Virtual Visionaries
We See What You Don't

2020/08/01
#Bikes4ERP_VVS

#BIKES4ERP

2020/08/01

Informatics 370 Deliverable 4 – Appendix A

Name (from left to right)	Surname	Student Number
Byron J.	Donald	17237913
Hansen	Li	17073708
Tapiwarufaro	Ndoro (Group Leader)	17210918
Cameron	King	17165904
Gareth	Phillips	17004162

Virtual Visionaries
GROUP 17

Table of Contents

Table of Figures

1.Data Flow Diagrams

Context Diagram

Sub-System 1-2

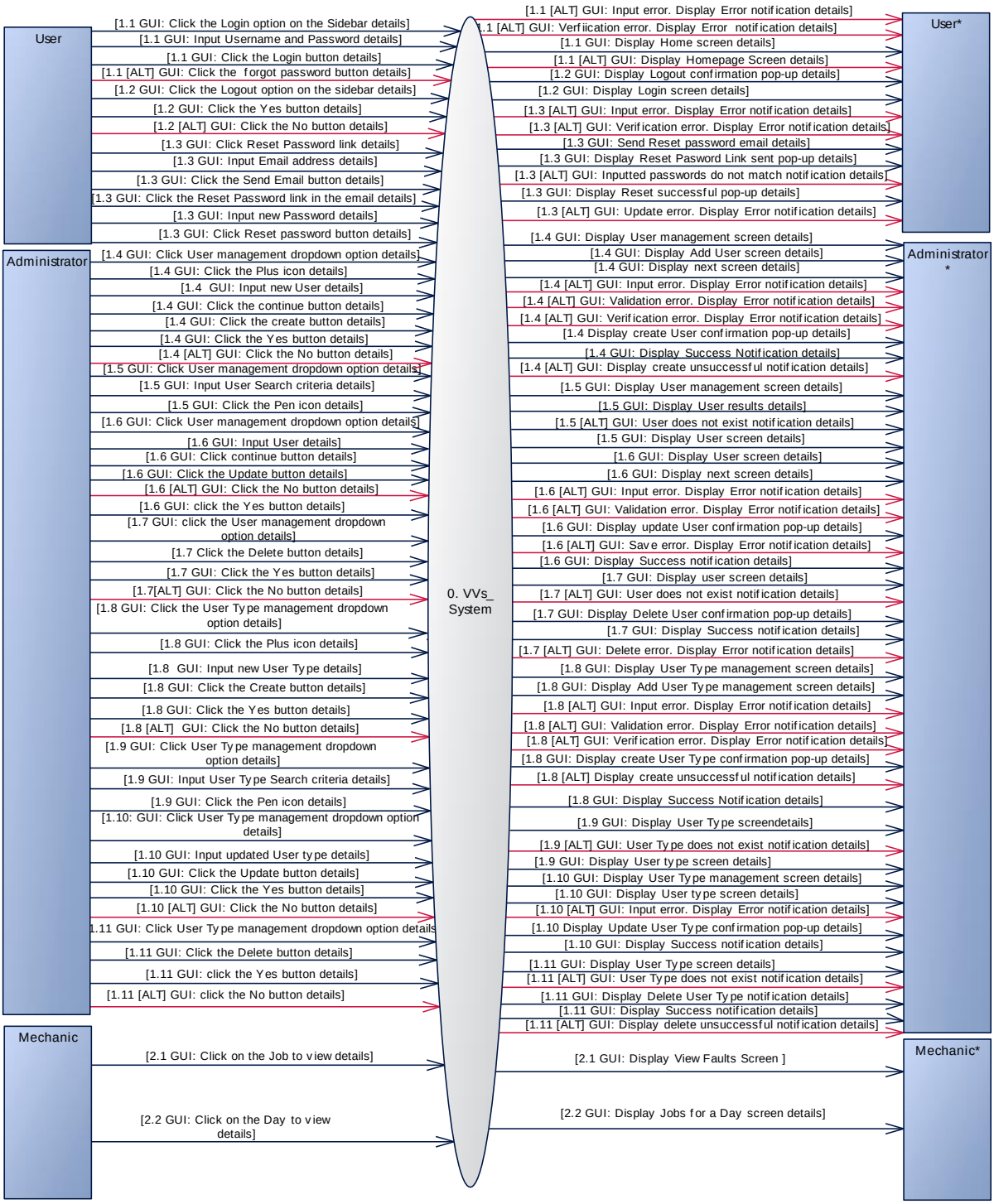


Figure 1 Context Diagram Sub-System 1-2

Sub-System 3-4

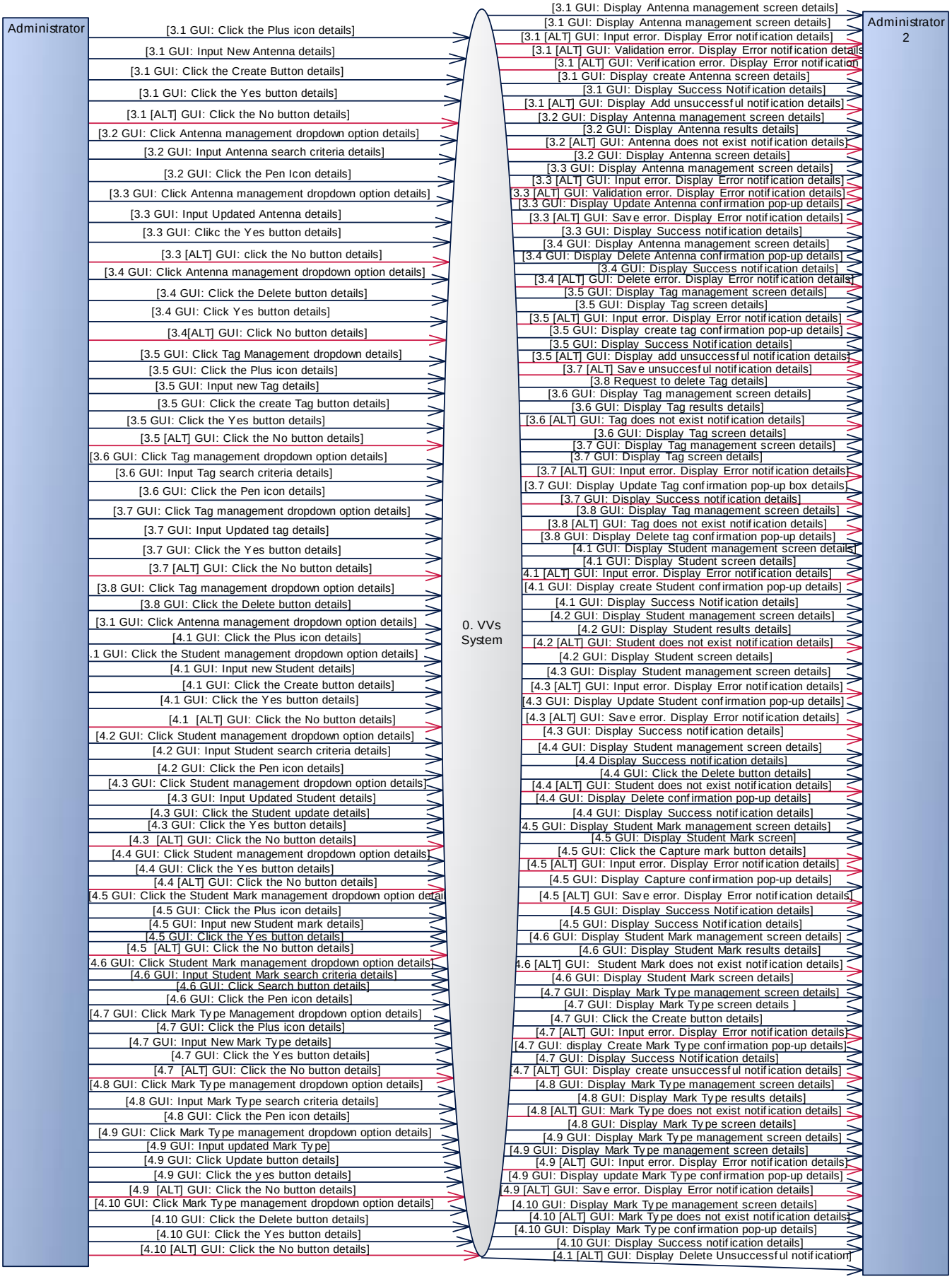


Figure 2 Context Diagram Sub-System 3-4

Sub-System 5-6

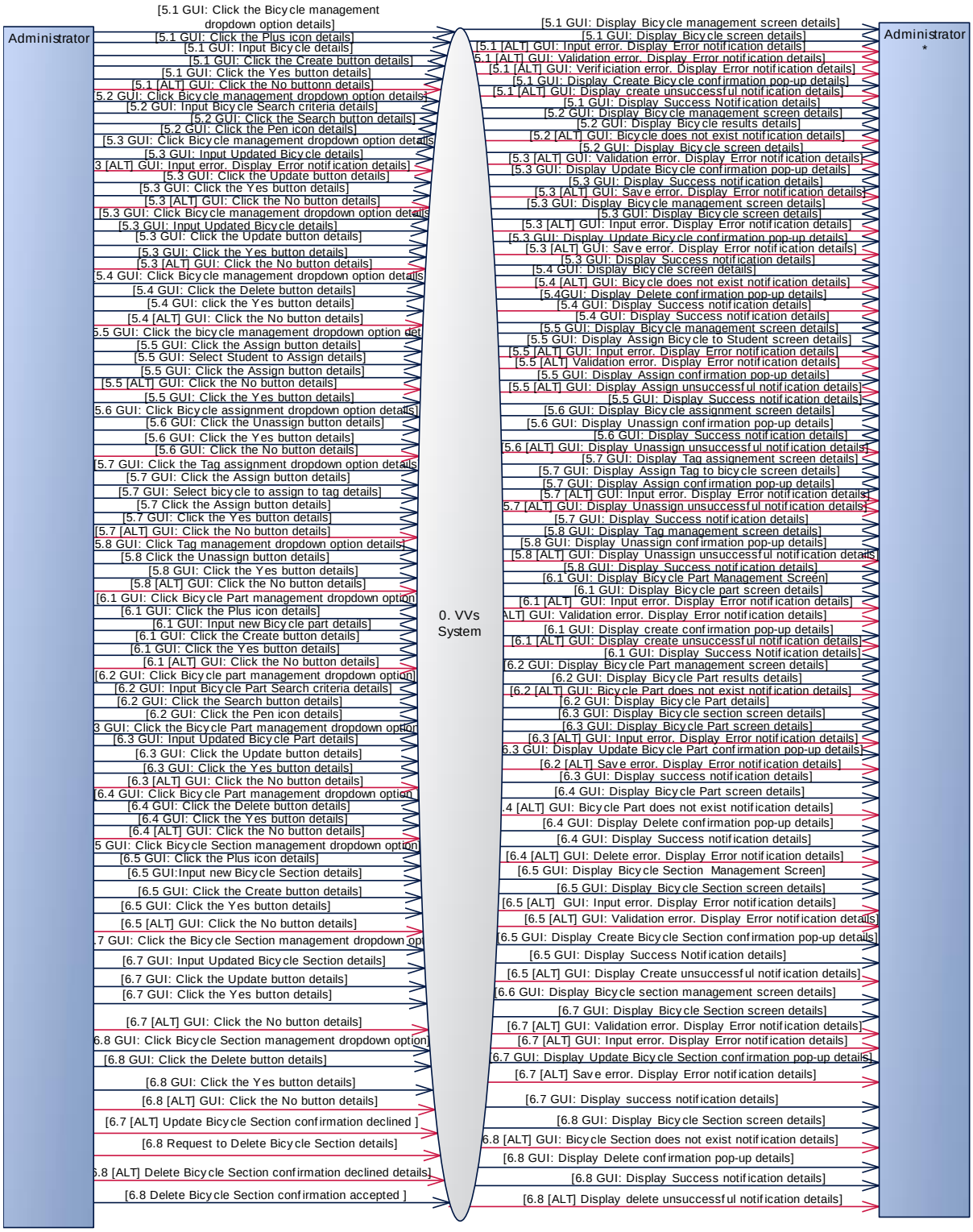


Figure 3 Context Diagram Sub-System 5-6

Sub-System 7-9

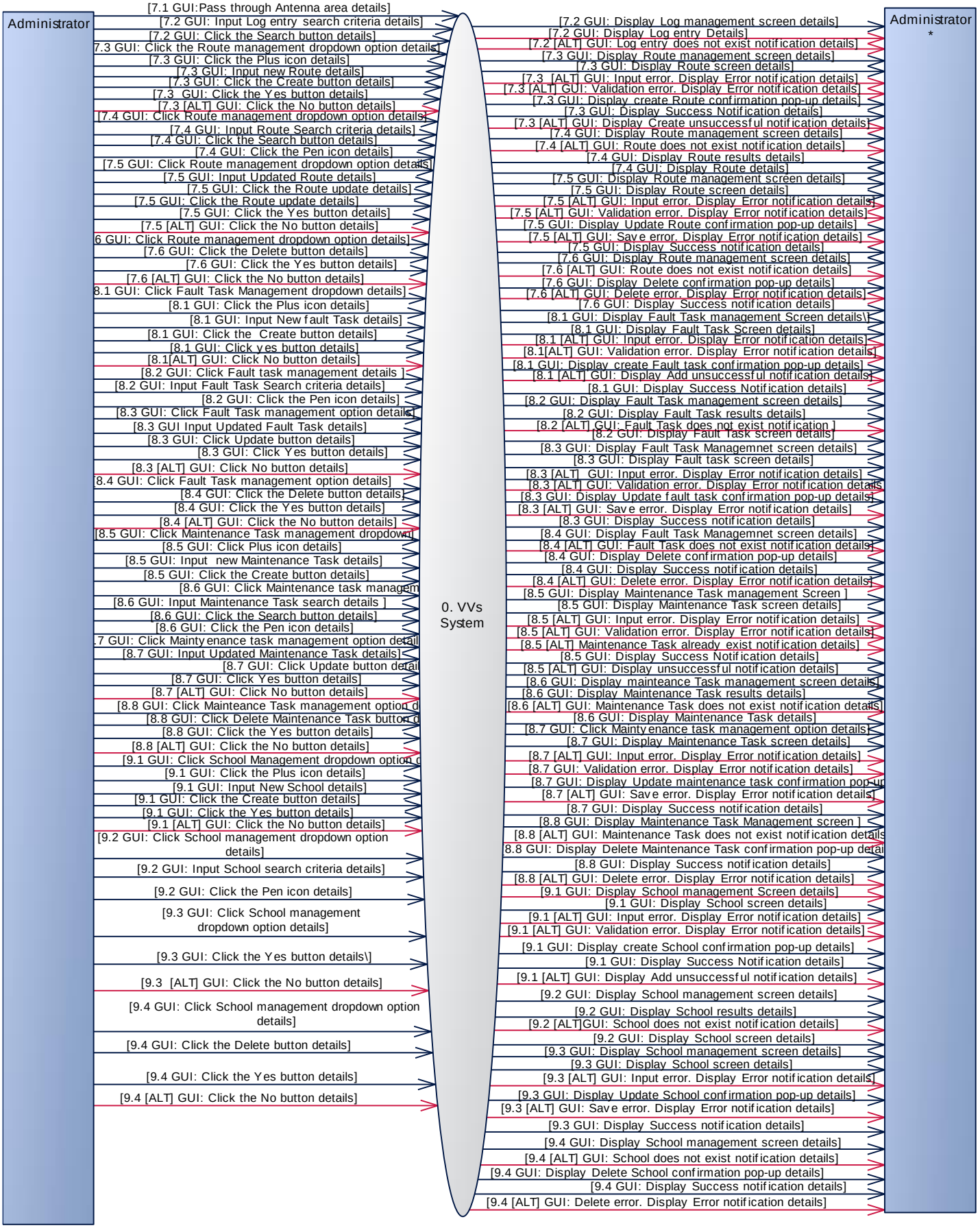


Figure 4 Context Diagram Sub-System 7-9

Sub-System 10-13

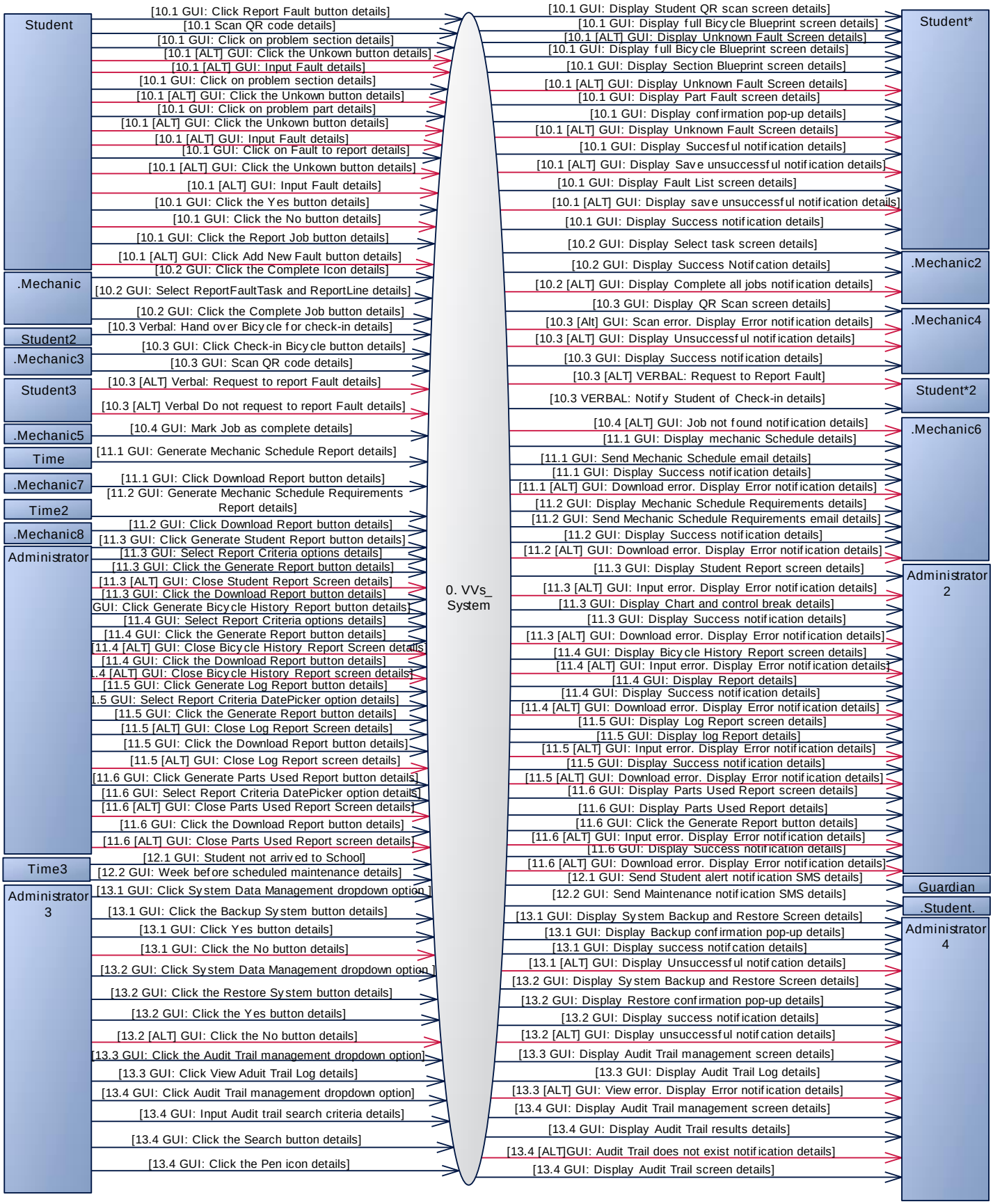


Figure 5 Context Diagram Sub-System 10-13

Decomposition Diagram

Sub-System 1: User

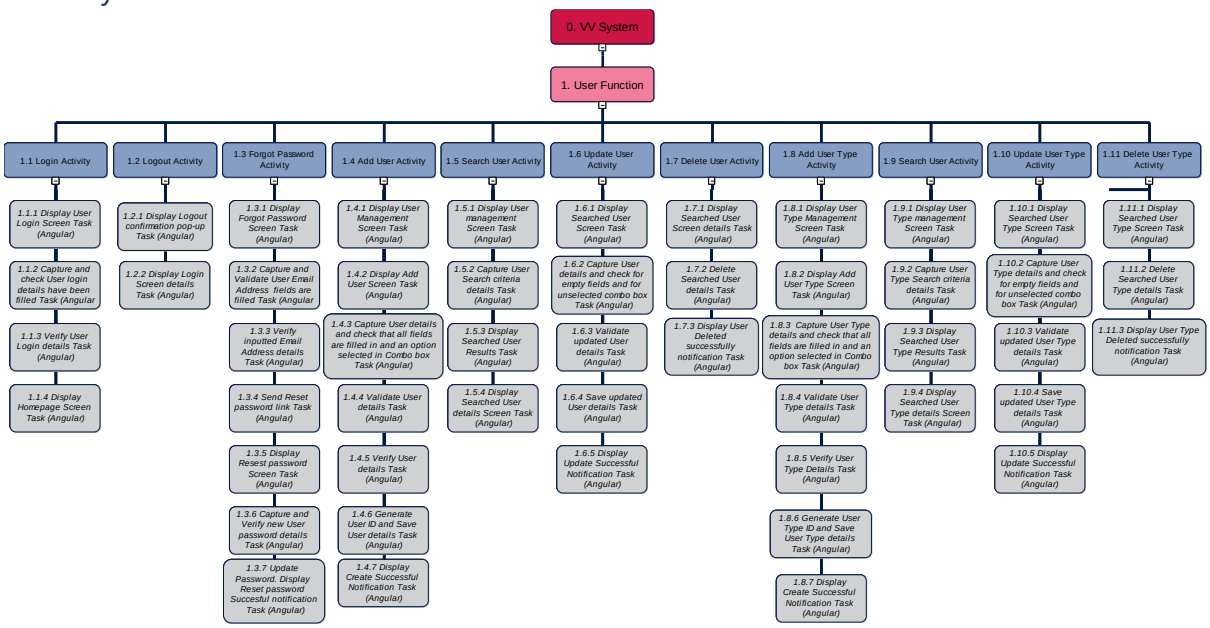


Figure 6 Decomposition Diagram Sub-System 1: User

Sub-System 2: Mechanic Schedule

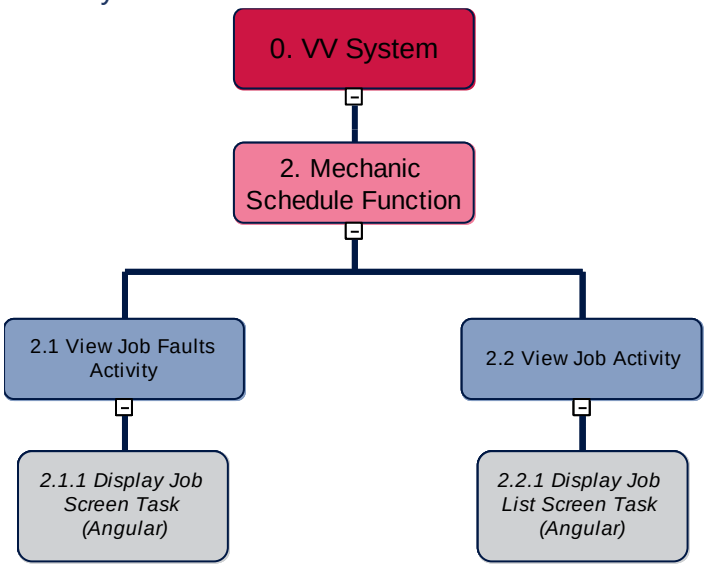


Figure 7 Decomposition Diagram Sub-System 2: Mechanic Schedule

Sub-System 3: Tracking Hardware

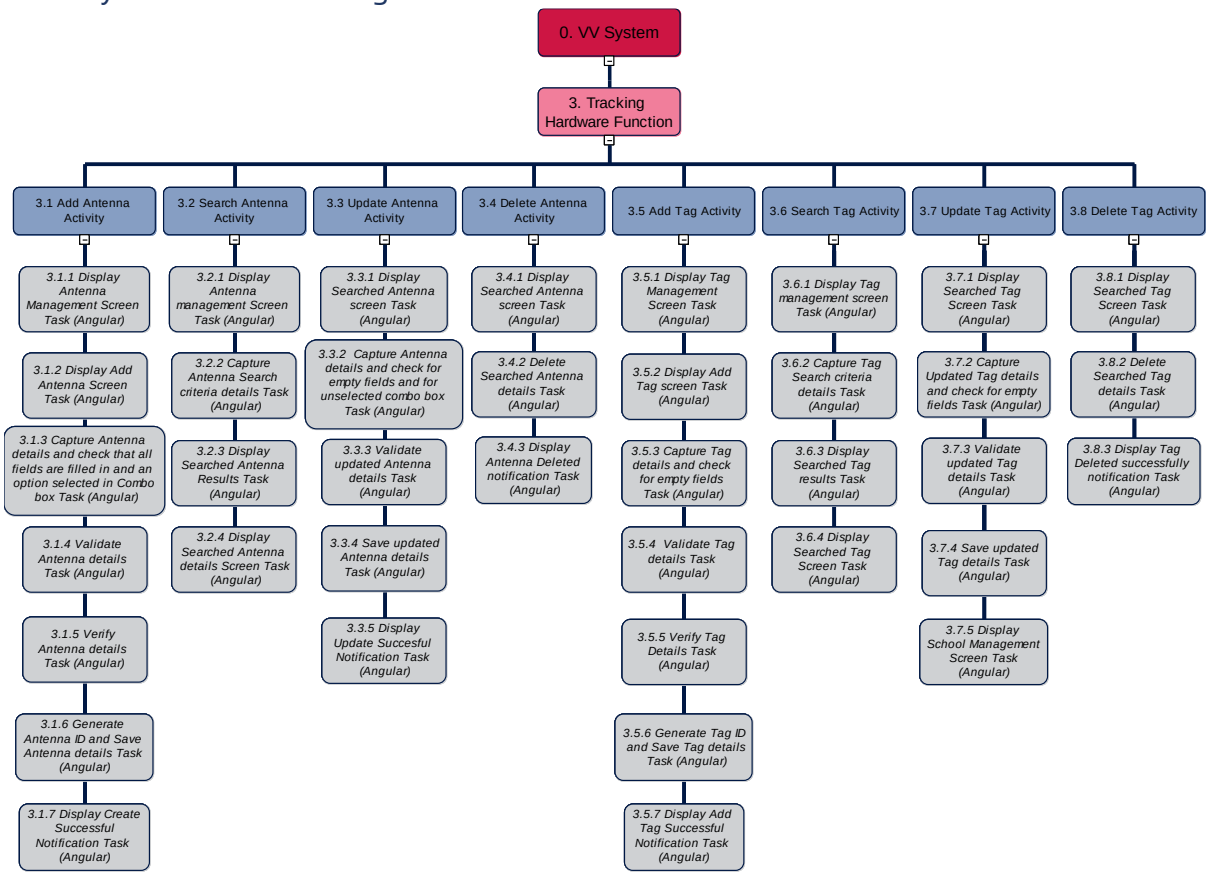


Figure 8 Decomposition Diagram Sub-System 3: Tracking Hardware

Sub-System 4: Student

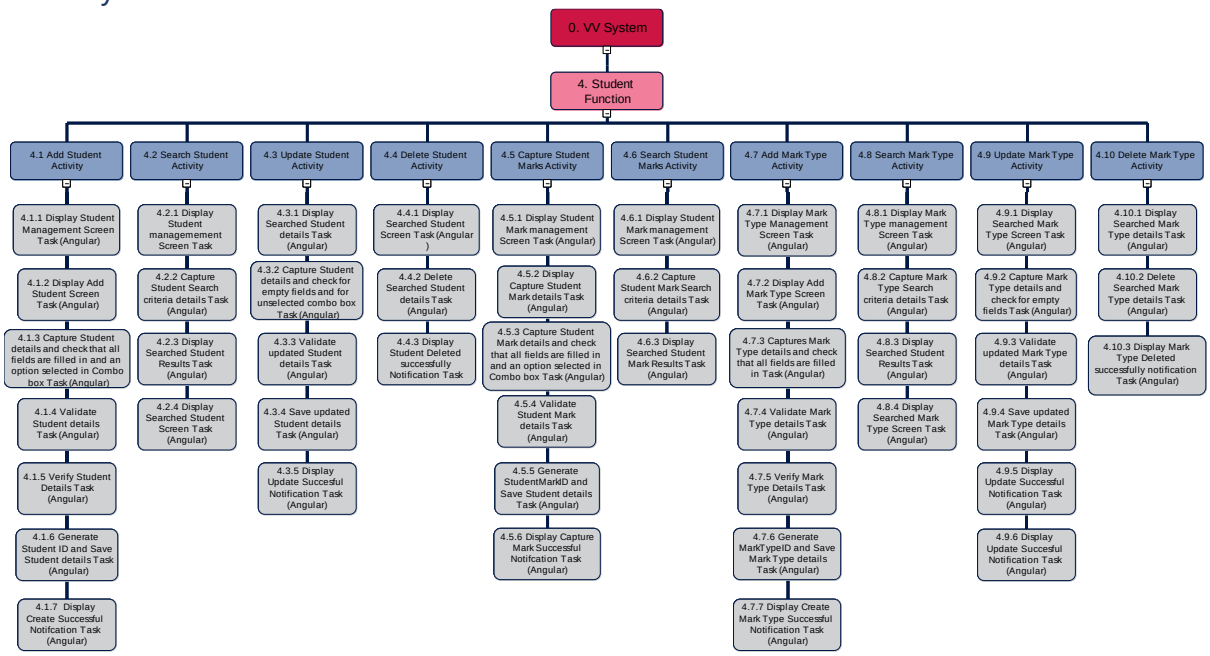


Figure 9 Decomposition Diagram Sub-System 4: Student

Sub-System 5: Bicycle

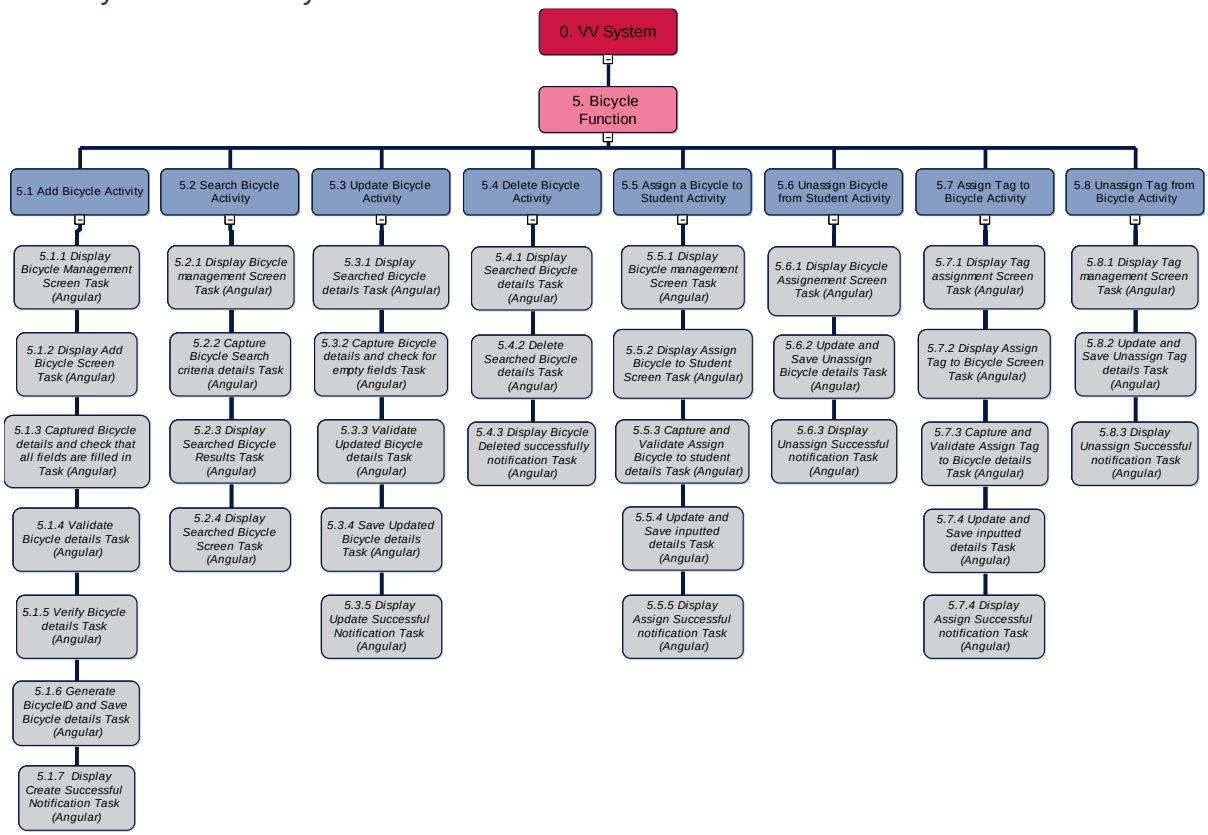


Figure 10 Decomposition Diagram Sub-System: Bicycle

Sub-System 6: Bicycle Part

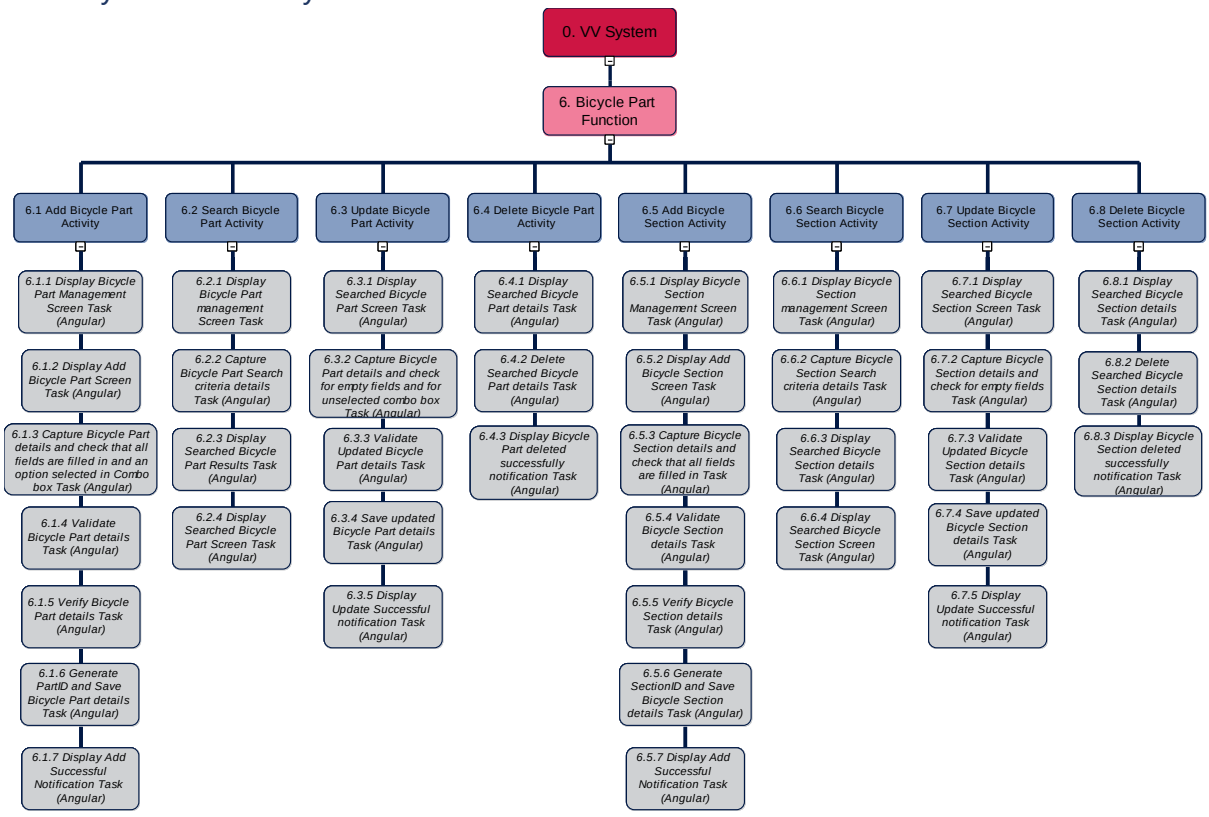


Figure 11 Decomposition Diagram Sub-System 6: Bicycle Part

Sub-System 7: Tracking

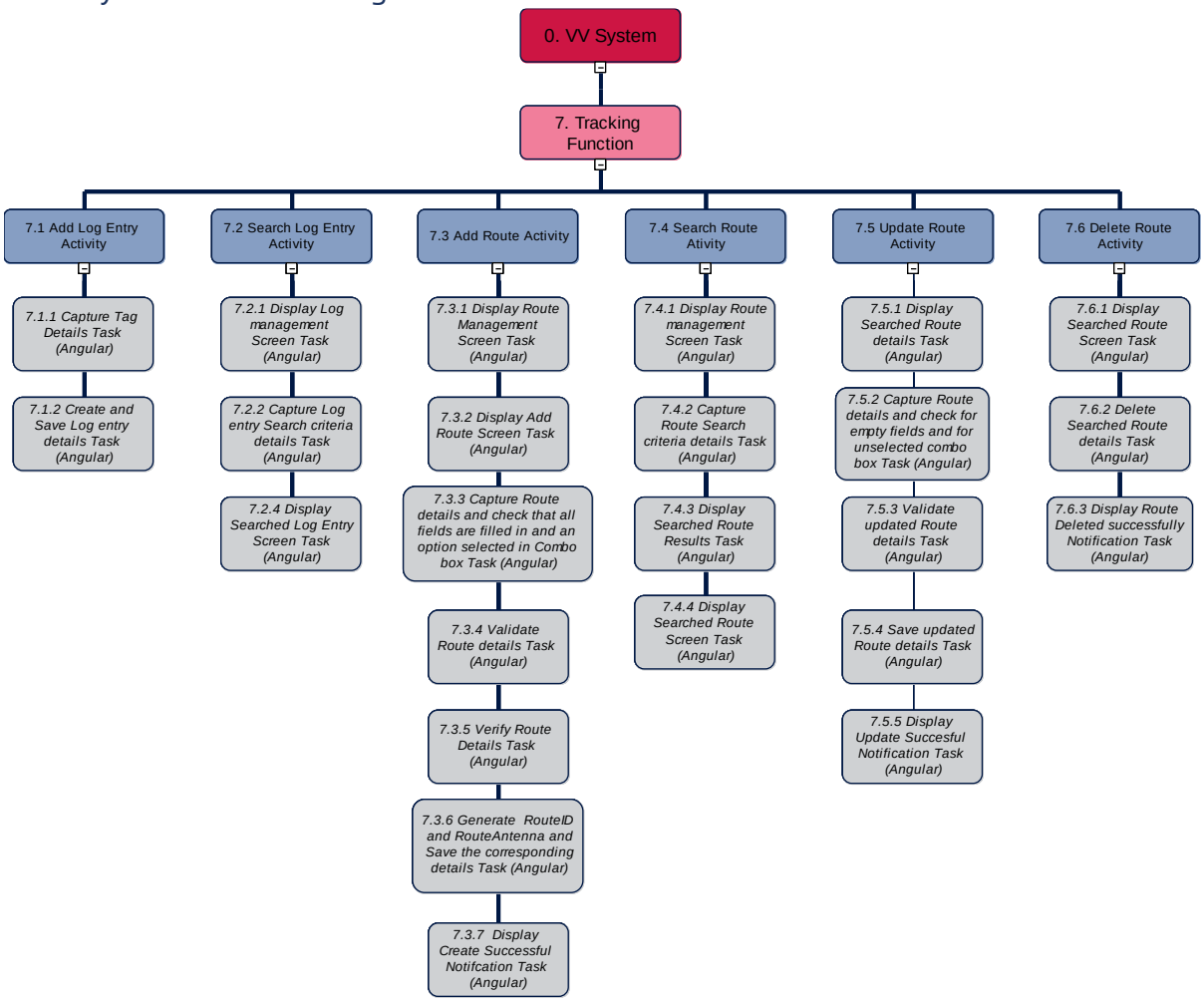


Figure 12 Decomposition Diagram Sub-System 7: Tracking

Sub-System 8: Job Tasks

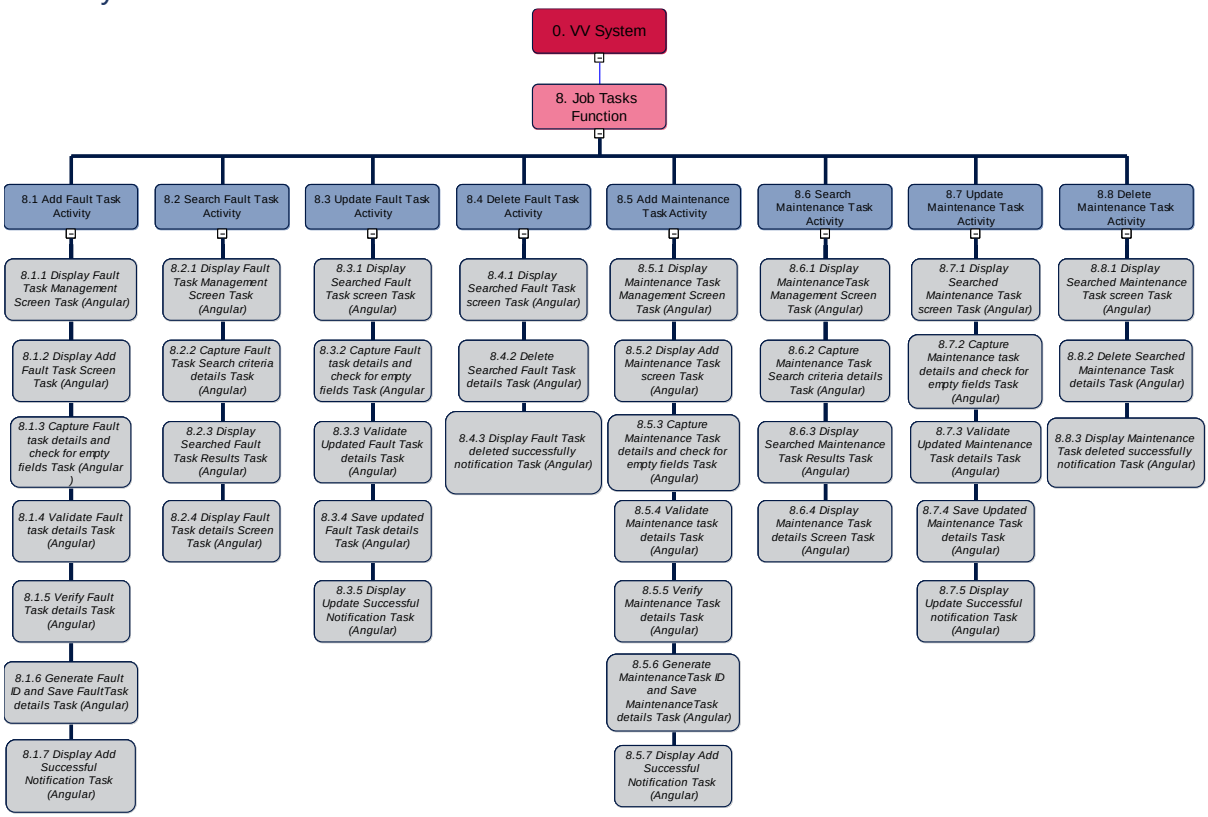


Figure 13 Decomposition Diagram Sub-System 8: Job Tasks

Sub-System 9: School

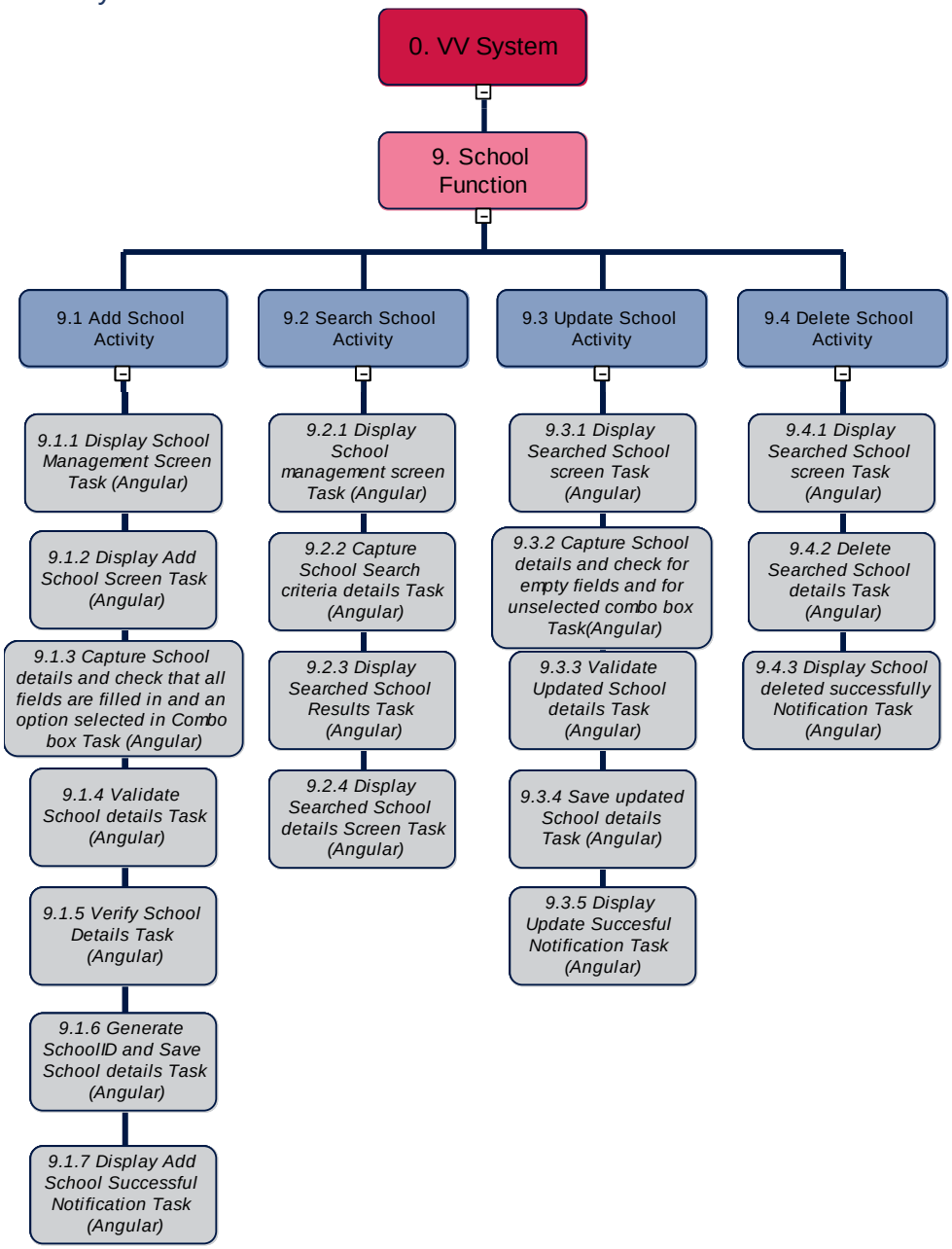


Figure 14 Decomposition Diagram Sub-System 9: School

Sub-System 10: Job

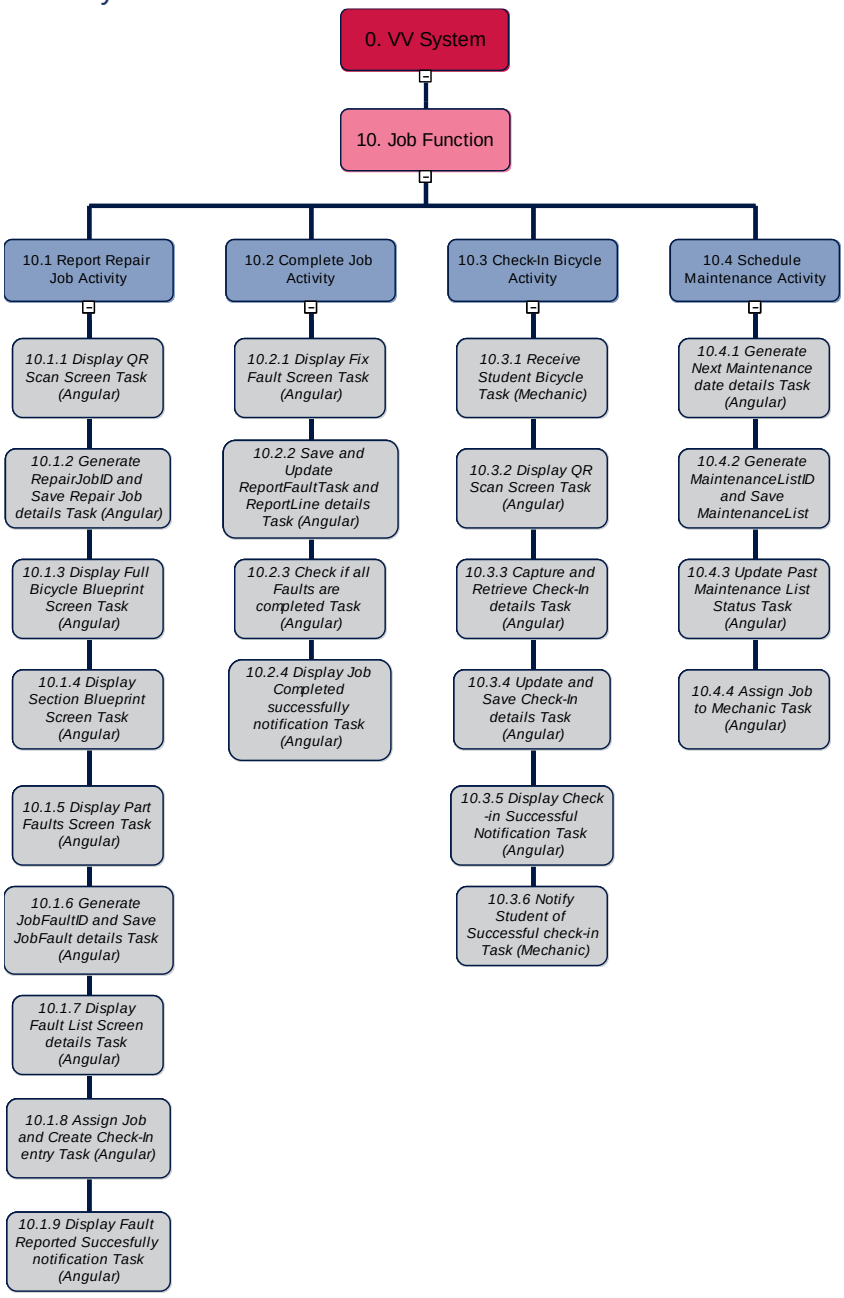


Figure 15 Decomposition Diagram Sub-System 10: Job

Sub-System 11: Reporting

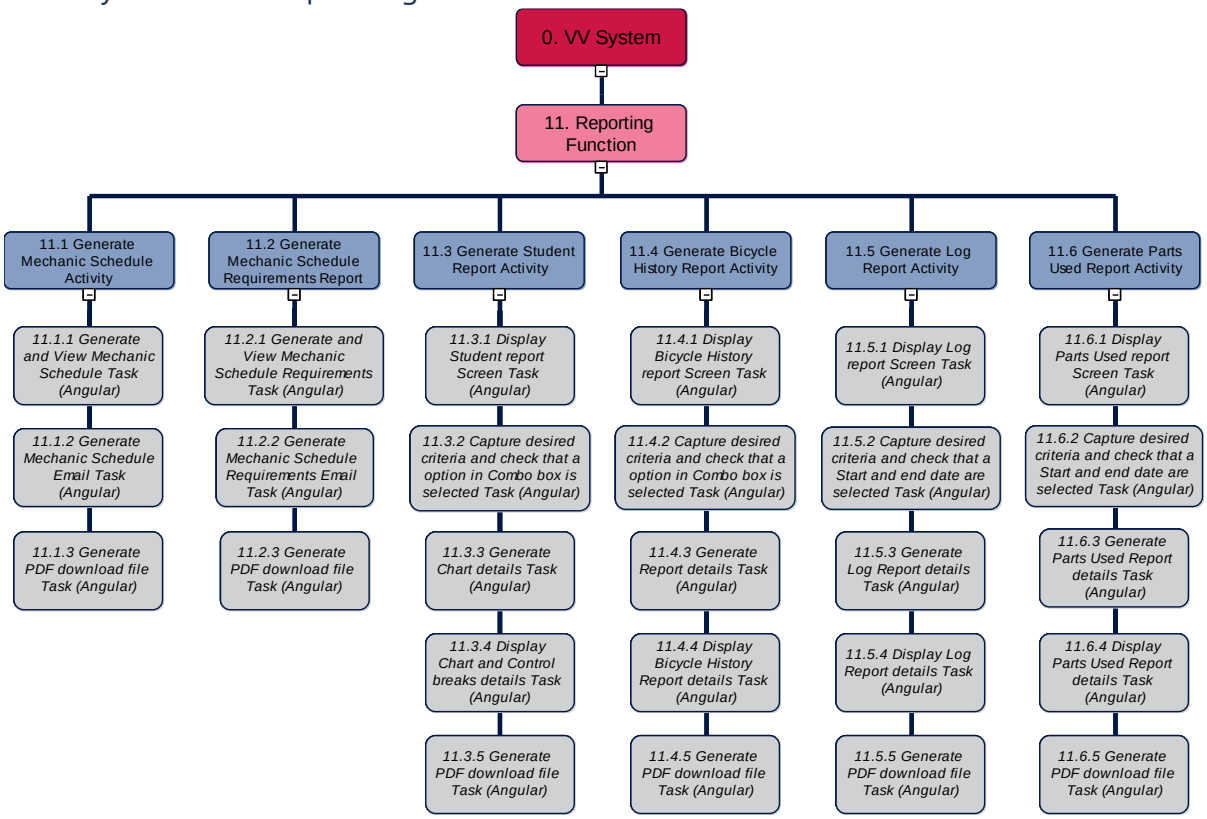


Figure 16 Decomposition Diagram Sub-System 11: Reporting

Sub-System 12: Notification

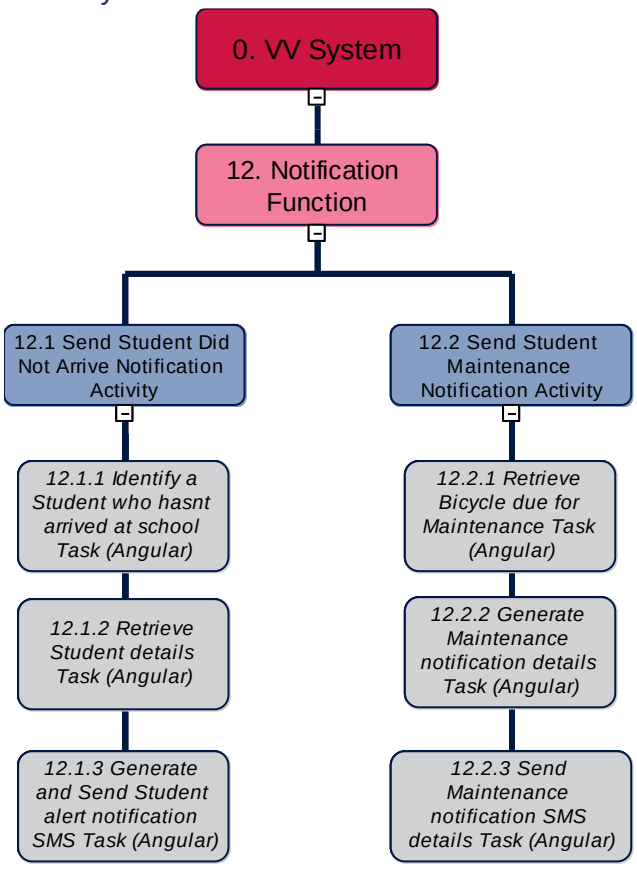


Figure 17 Decomposition Diagram Sub-System 12: Notification

Sub-System 13: Administration

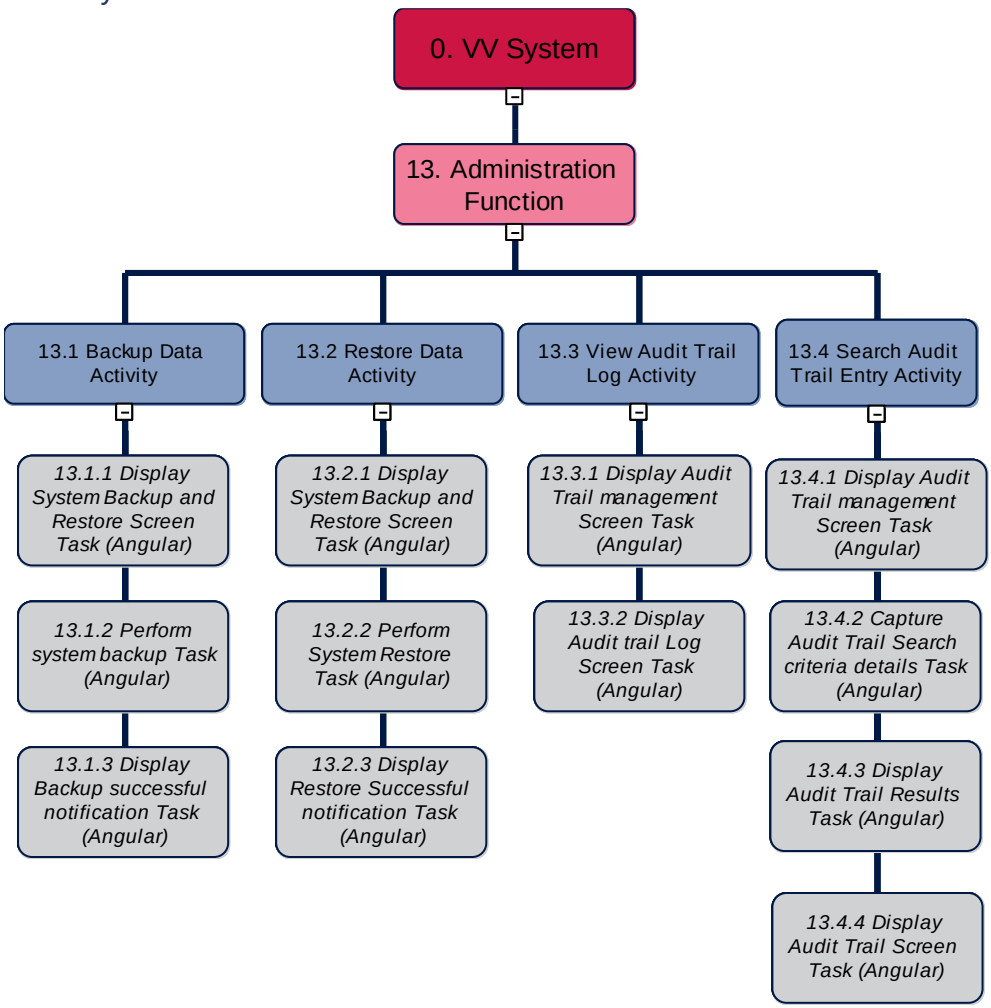


Figure 18 Decomposition Diagram Sub-System 13: Administration

High-Level Diagram

Sub System 1: User

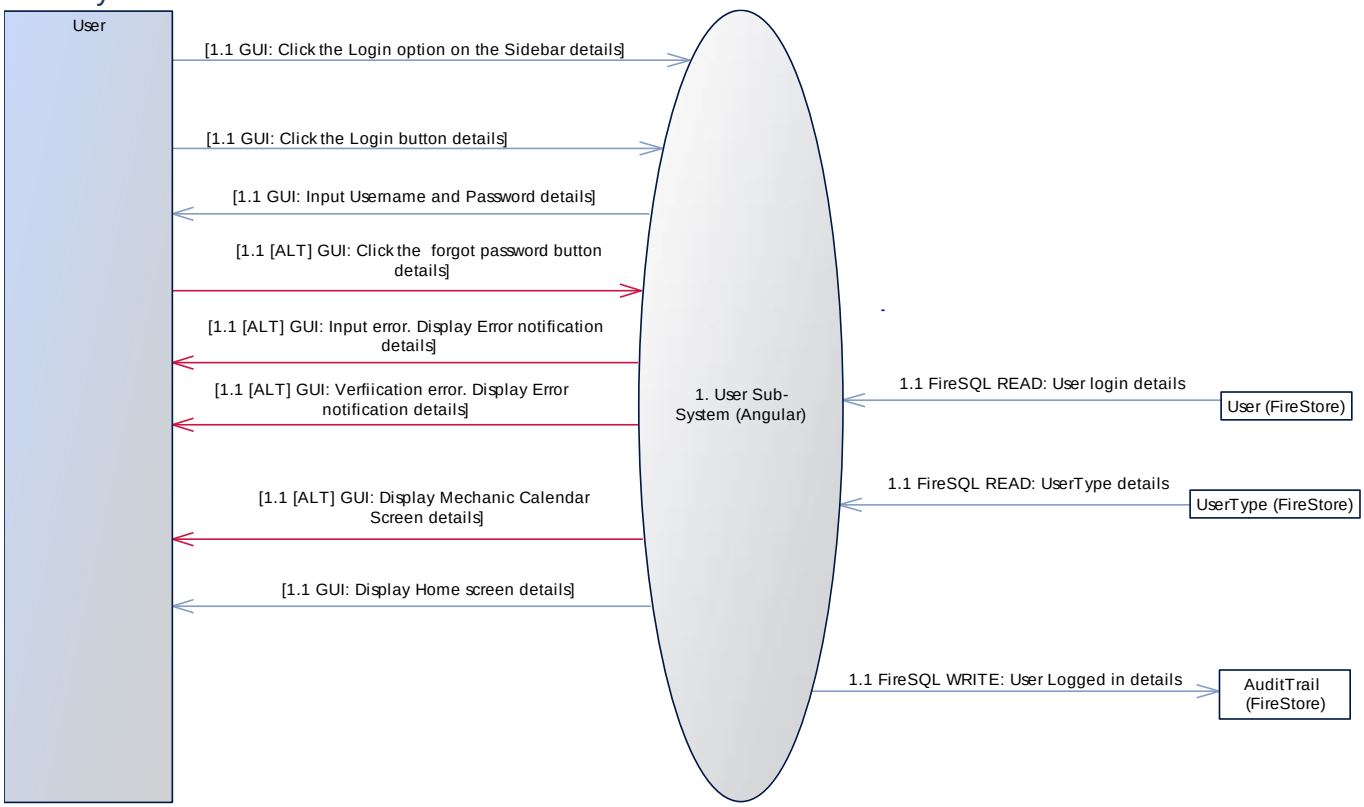


Figure 19 High-Level Diagram Sub-System 1: Part 1

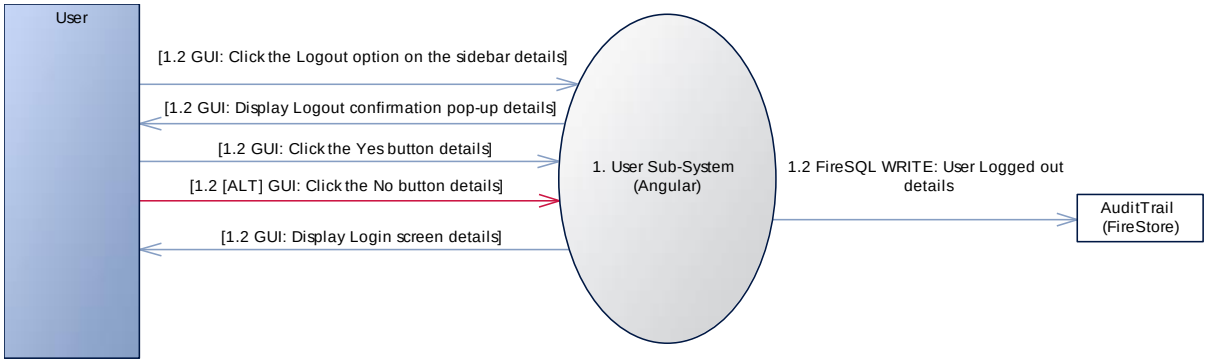


Figure 20 High-Level Diagram Sub-System 1: Part 2

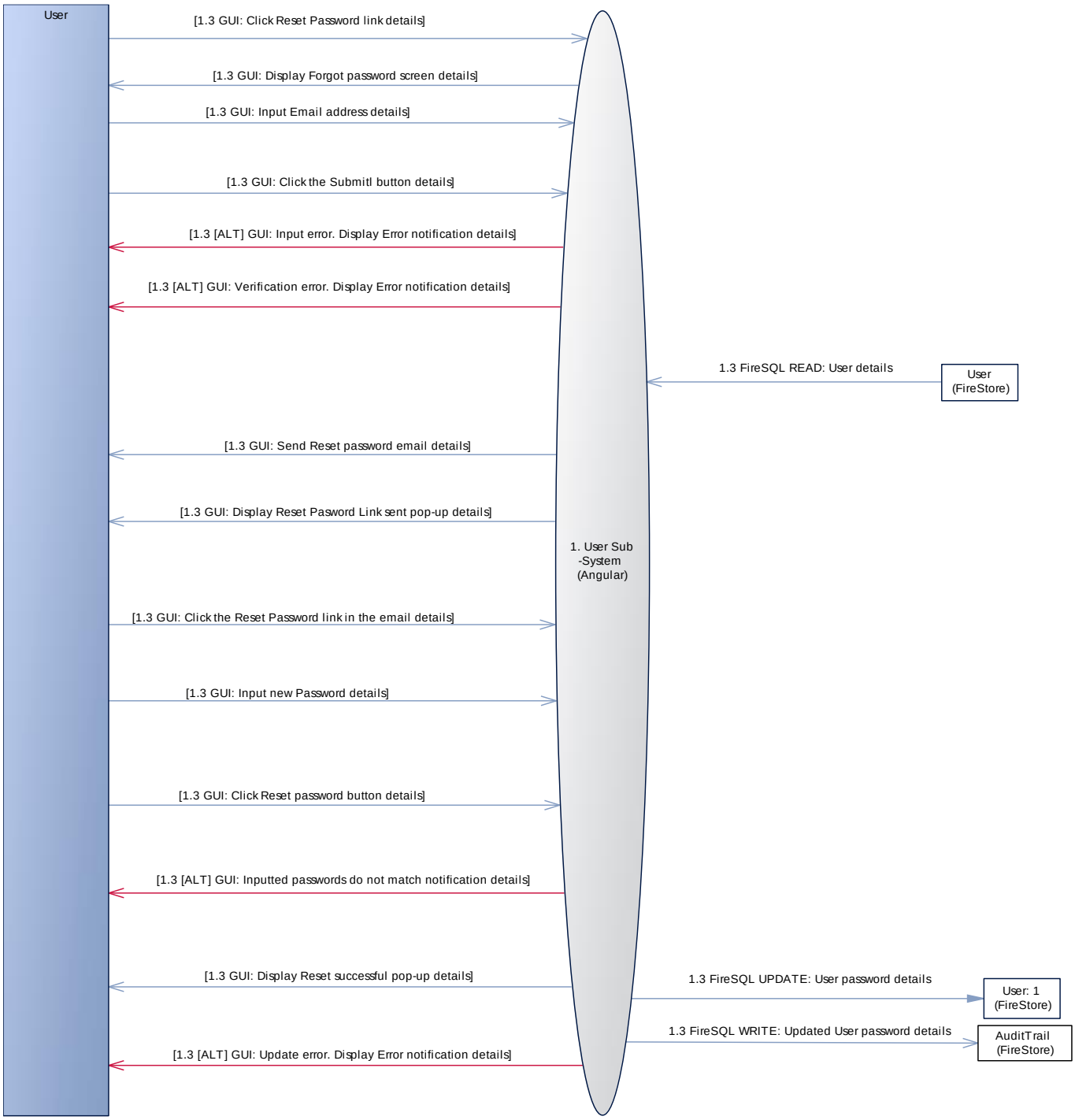


Figure 21 High-Level Diagram Sub-System 1: Part 3

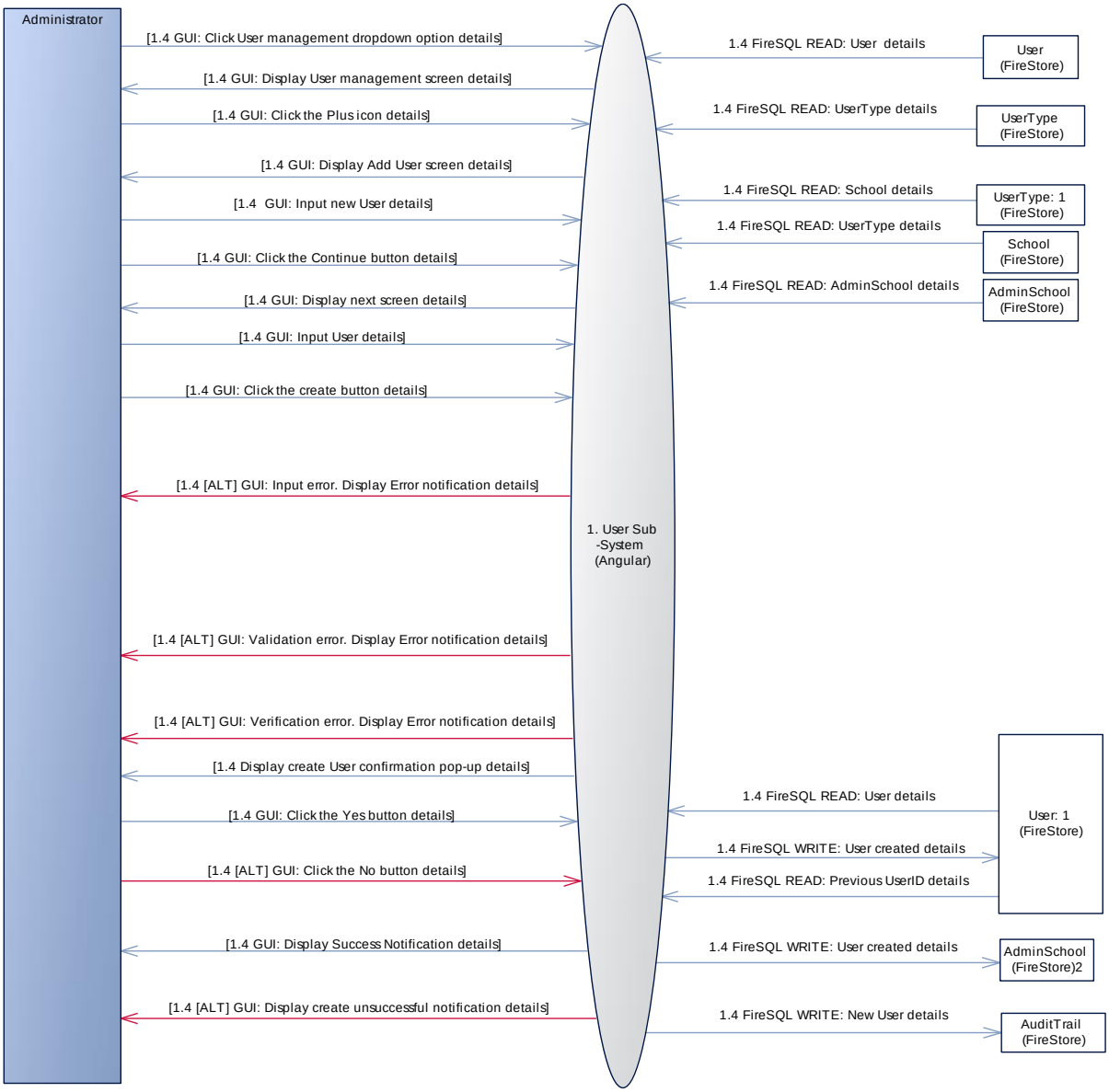


Figure 22 High-Level Diagram Sub-System 1: Part 4

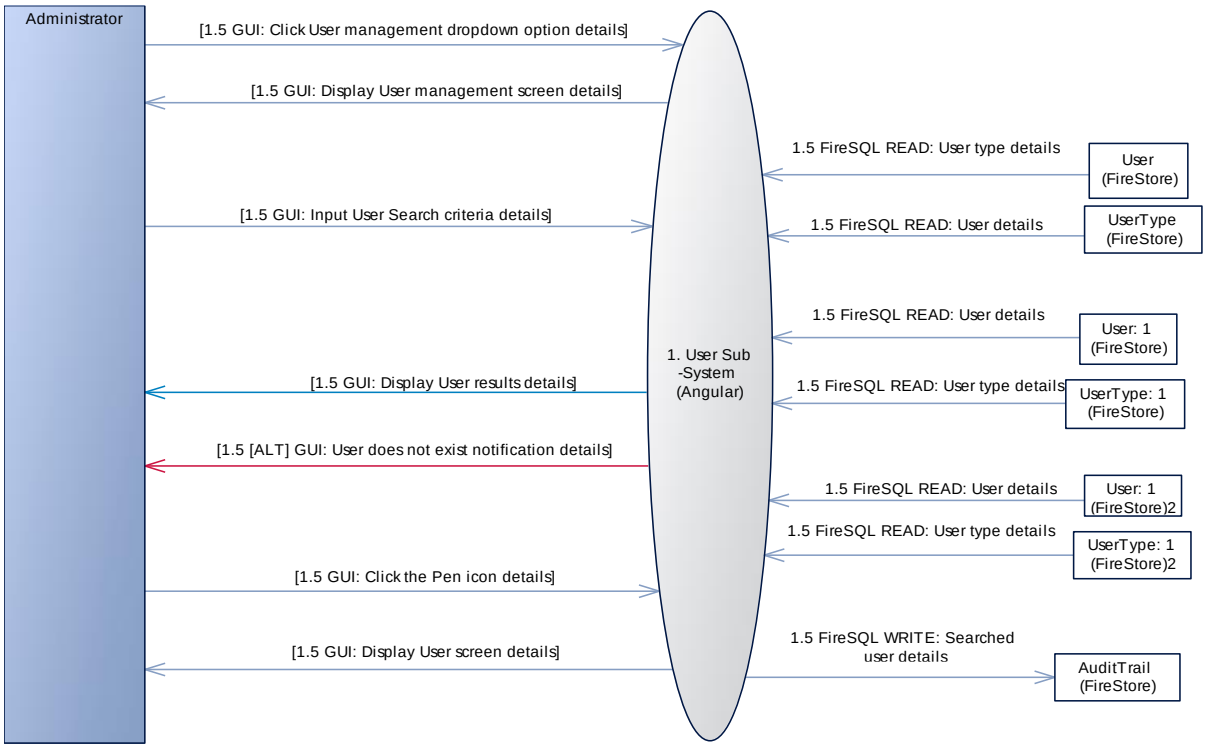


Figure 23 High-Level Diagram Sub-System 1: Part 5

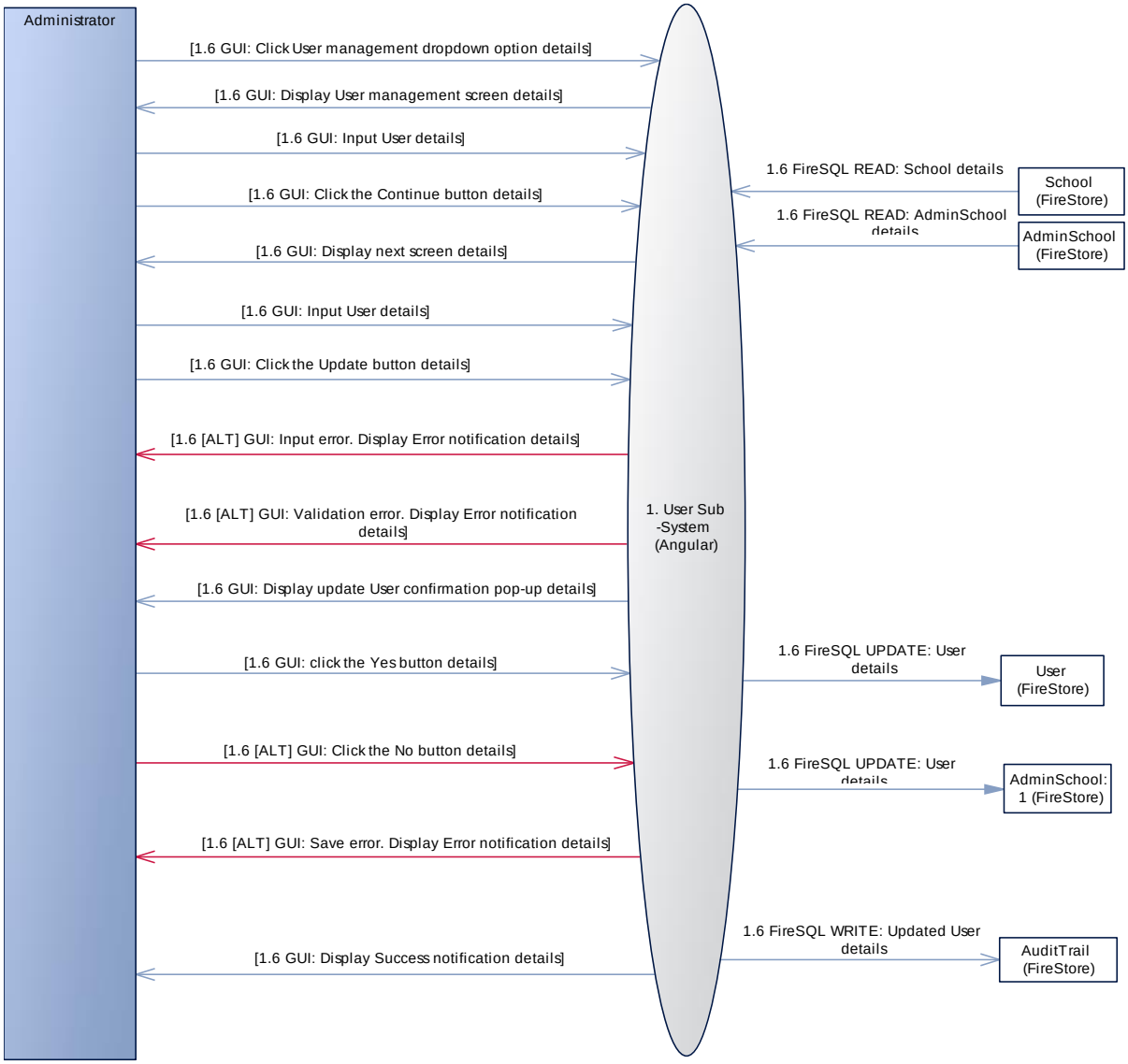


Figure 24 High-Level Diagram Sub-System 1: Part 6

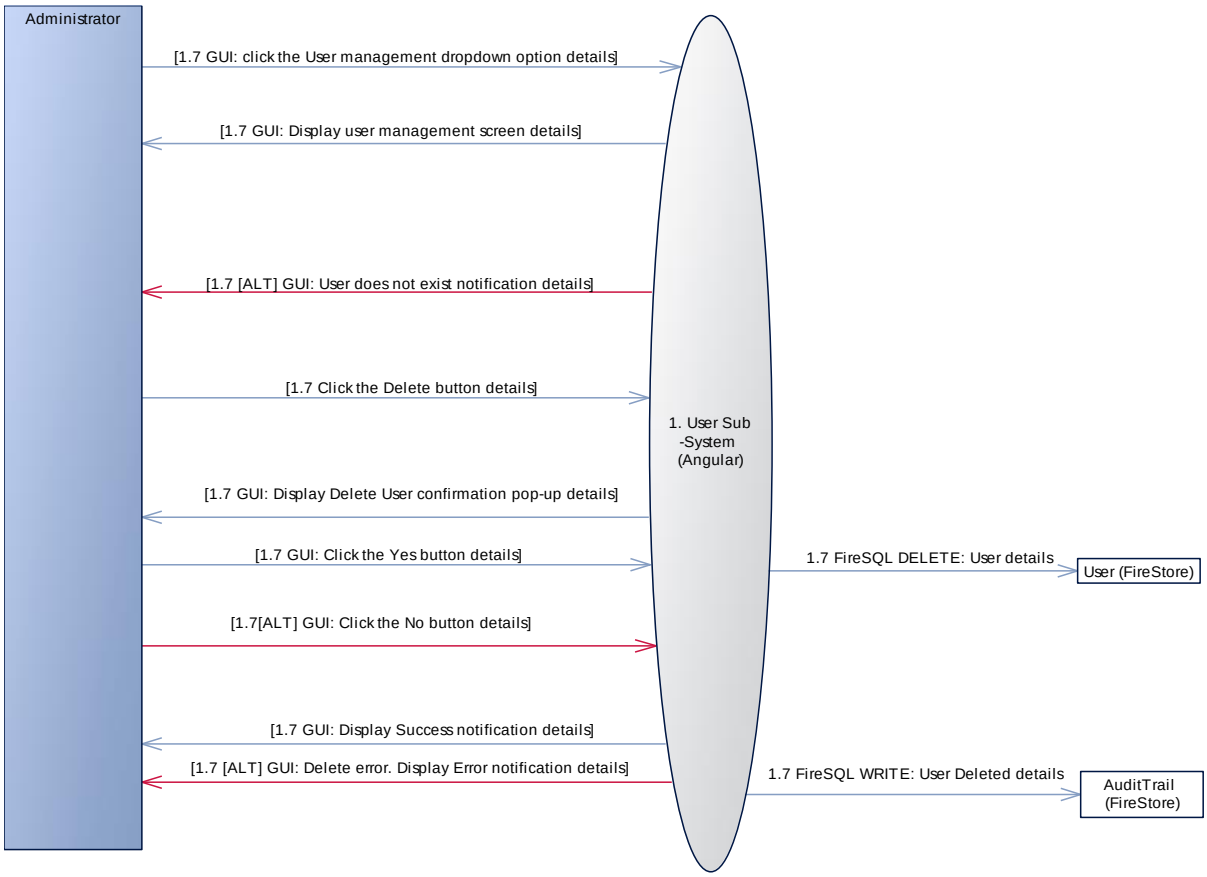


Figure 25 High-Level Diagram Sub-System 1: Part 7

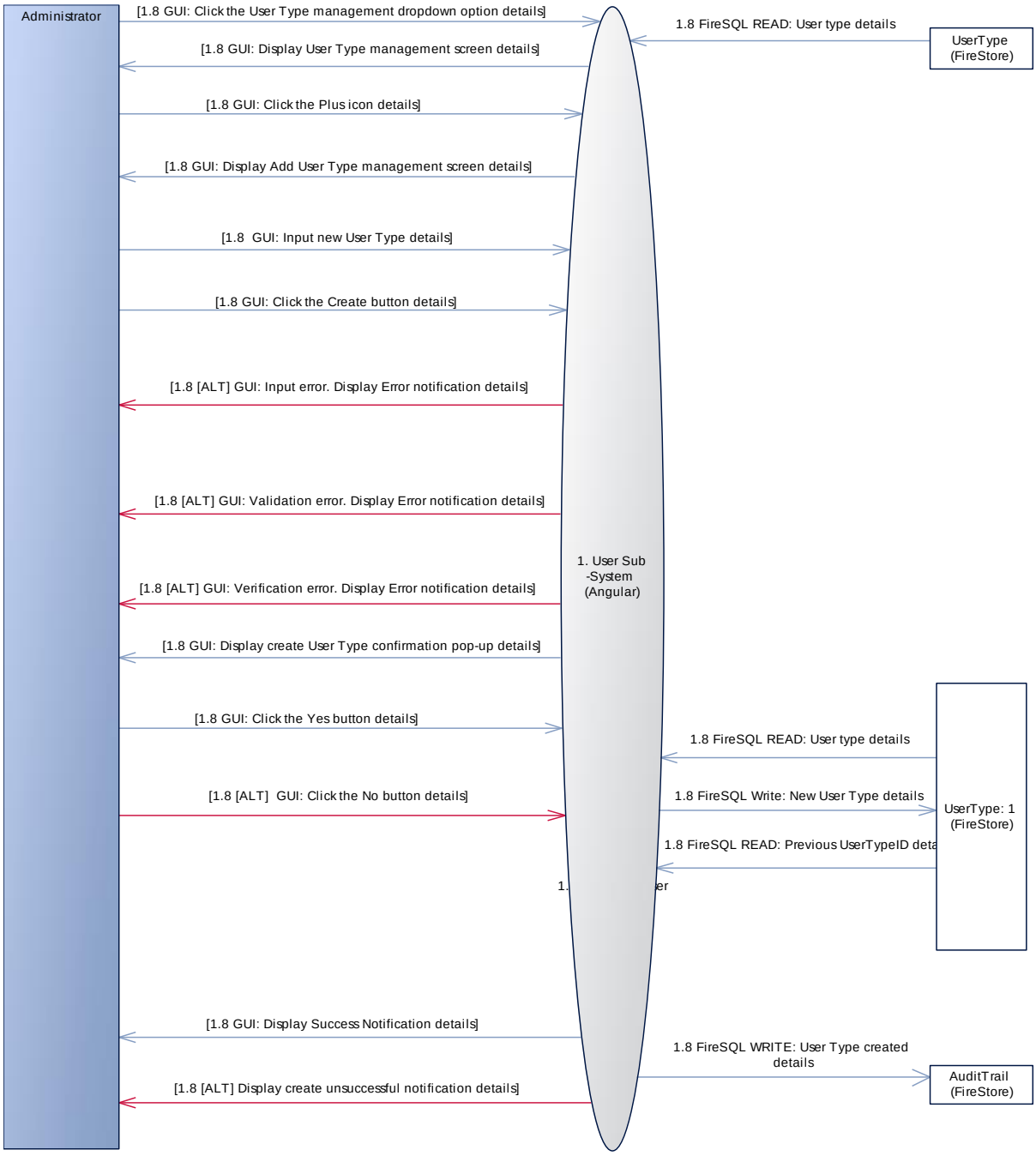


Figure 26 High-Level Diagram Sub-System 1: Part 8

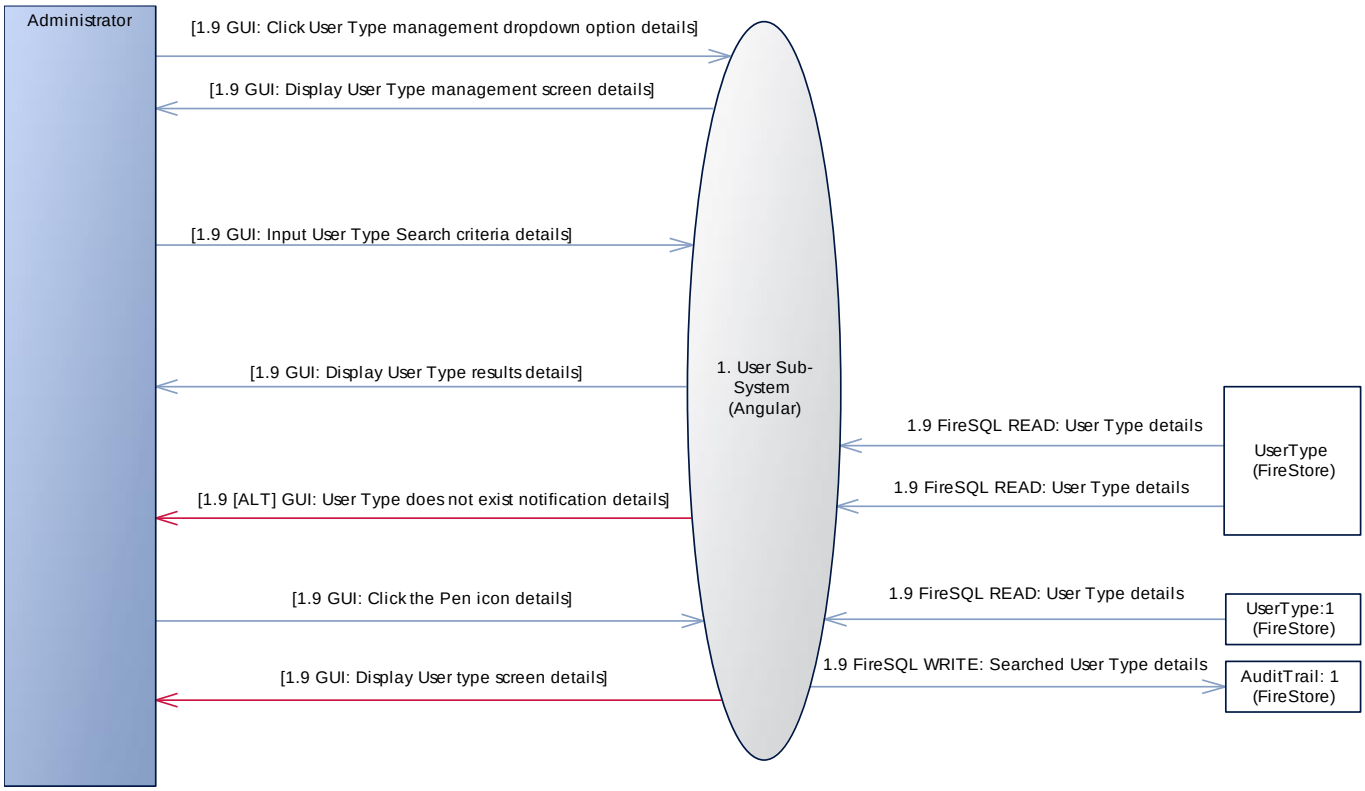


Figure 27 High-Level Diagram Sub-System 1: Part 9

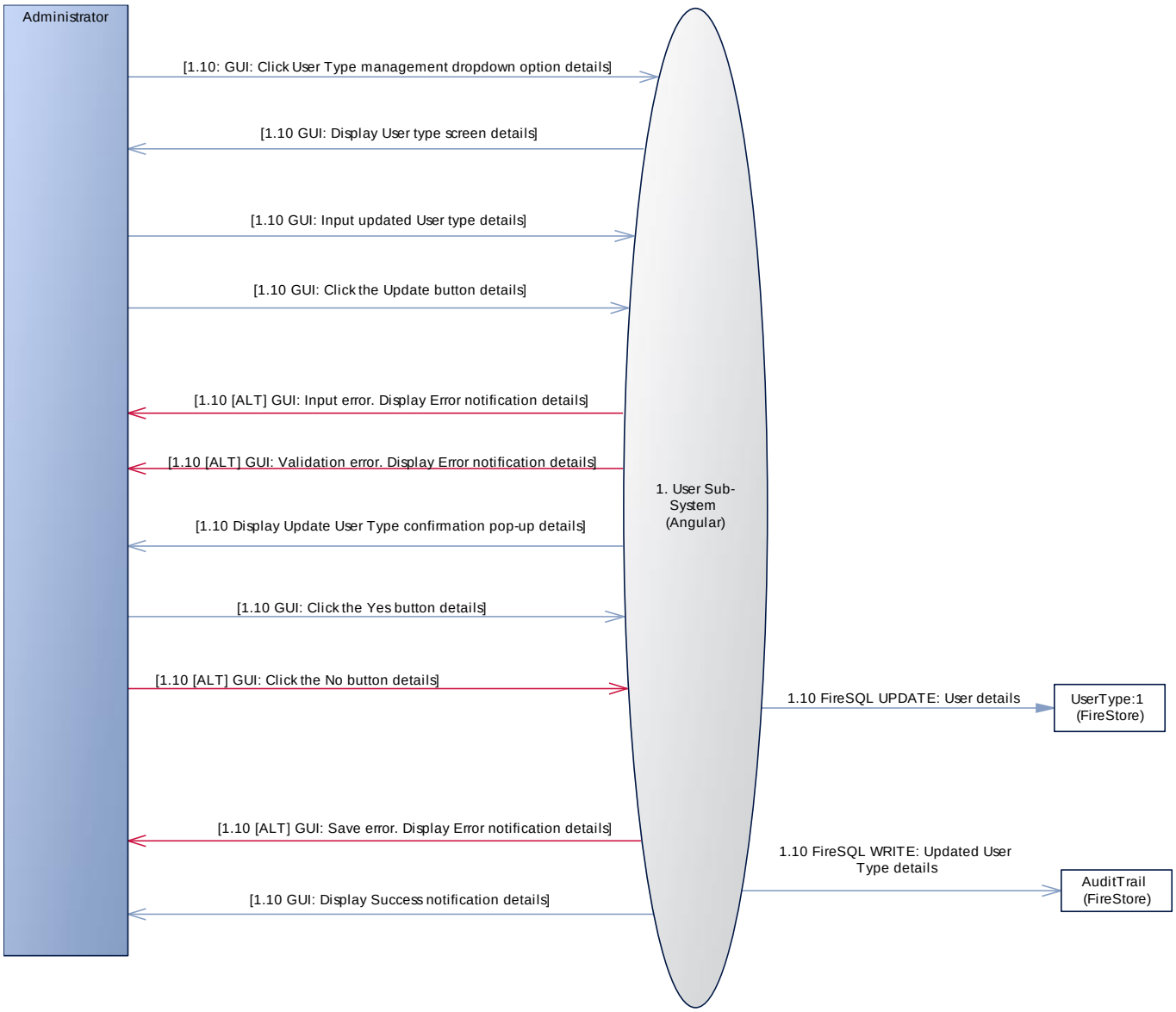


Figure 28 High-Level Diagram Sub-System 1: Part 10

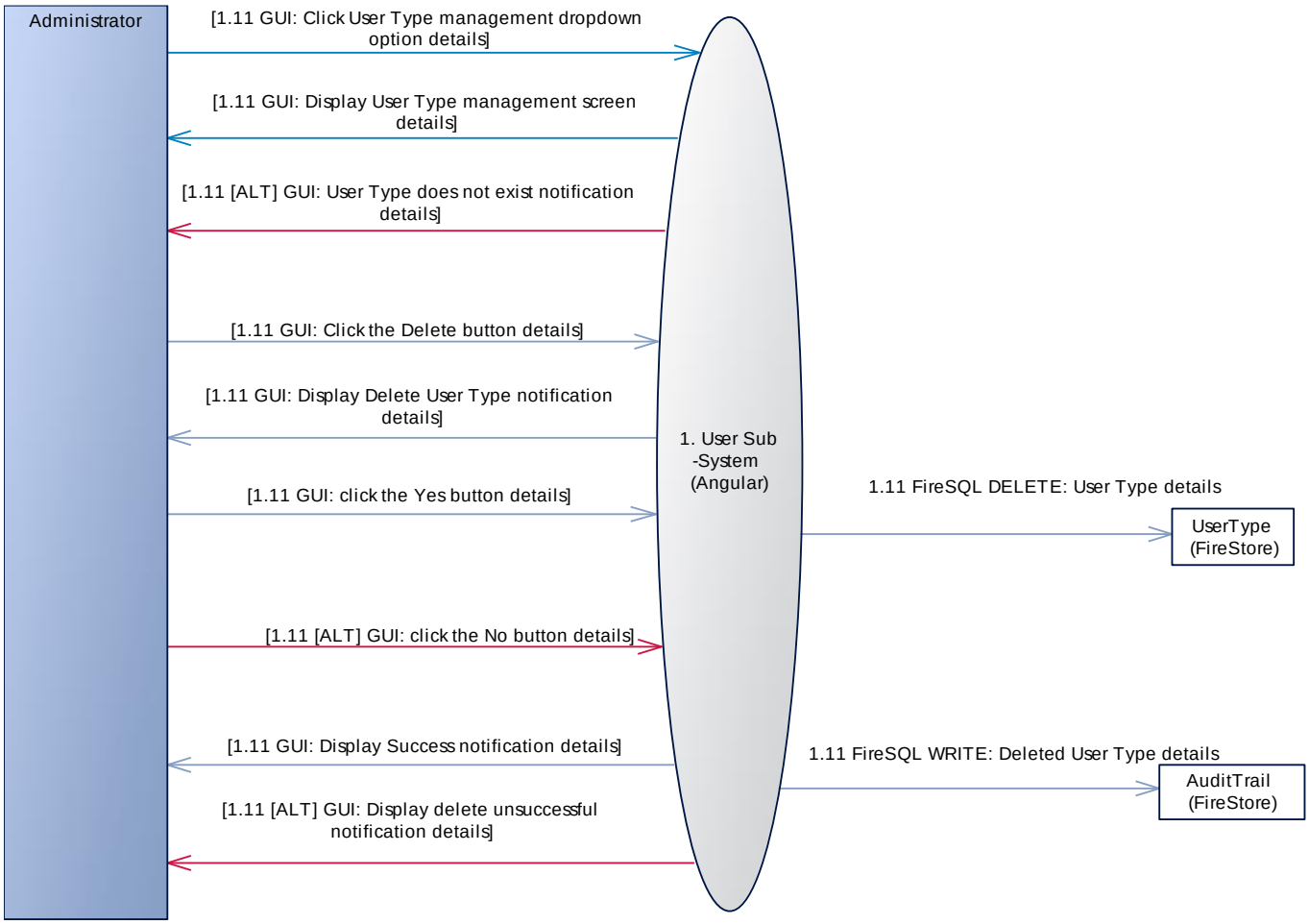


Figure 29 High-Level Diagram Sub-System 1: Part 11

Sub-System 2: Mechanic Schedule

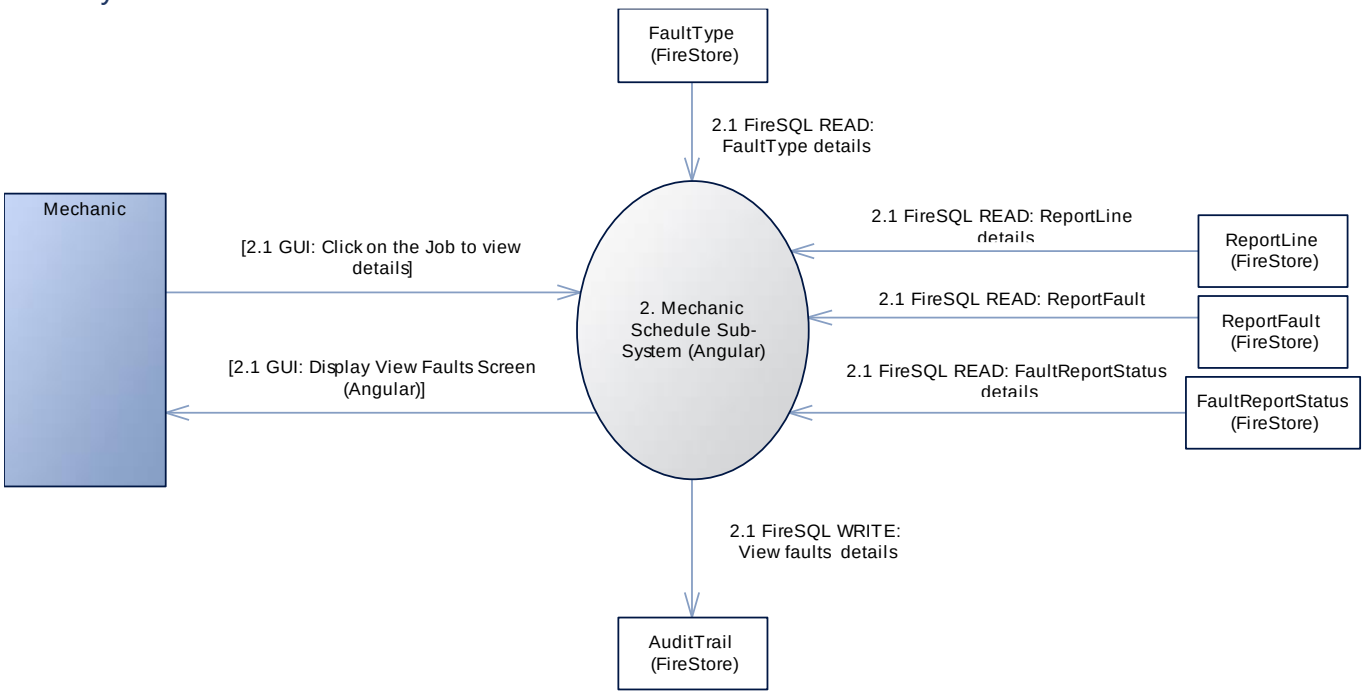


Figure 30 High-Level Diagram Sub-System 2: Part 1

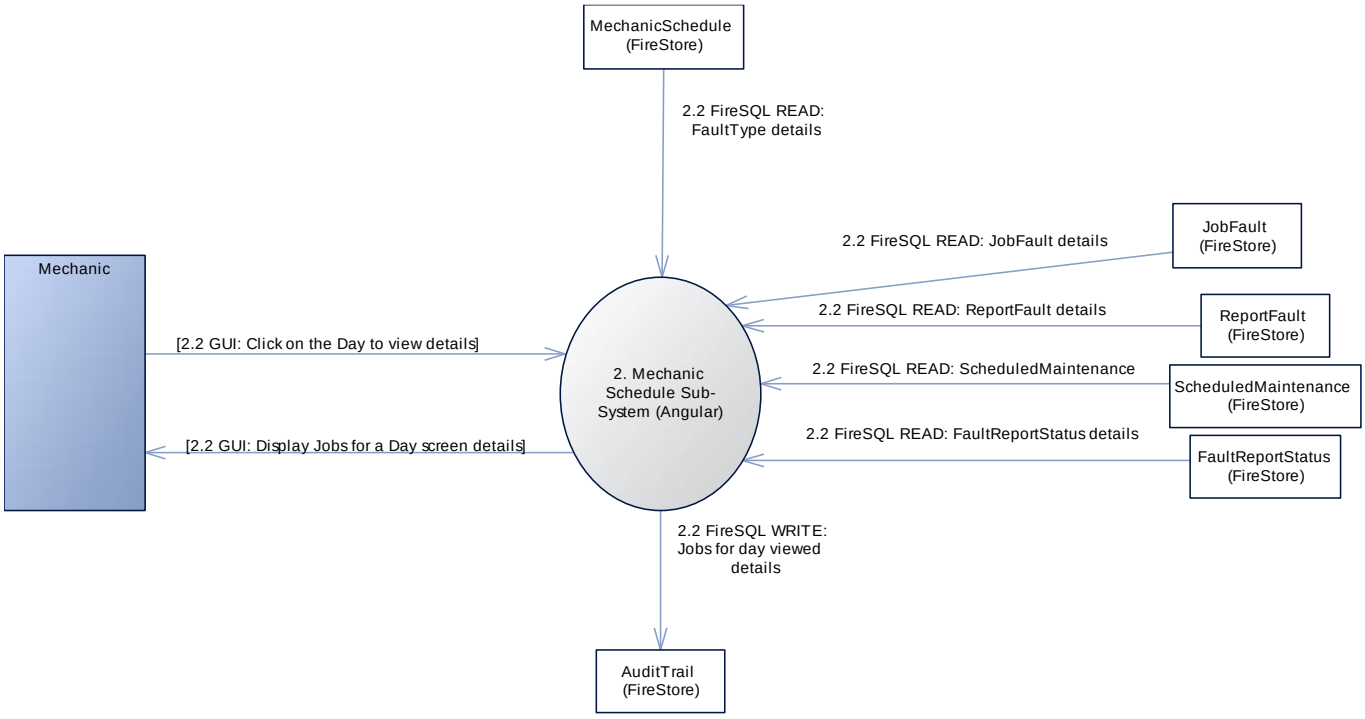


Figure 31 High-Level Diagram Sub-System 2: Part 2

Sub-System 3: Tracking Hardware

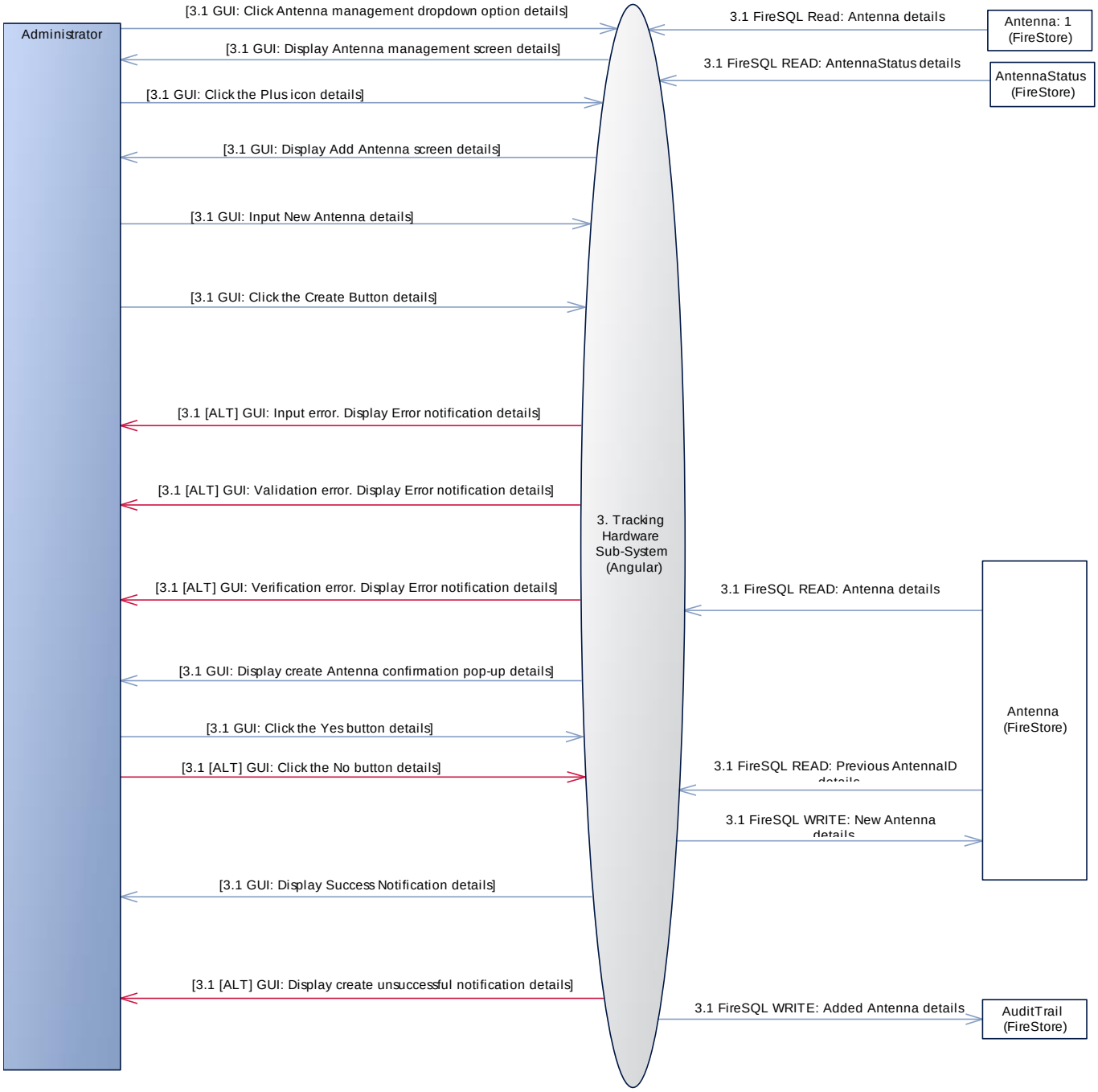


Figure 32 High-Level Diagram Sub-System 3: Part 1

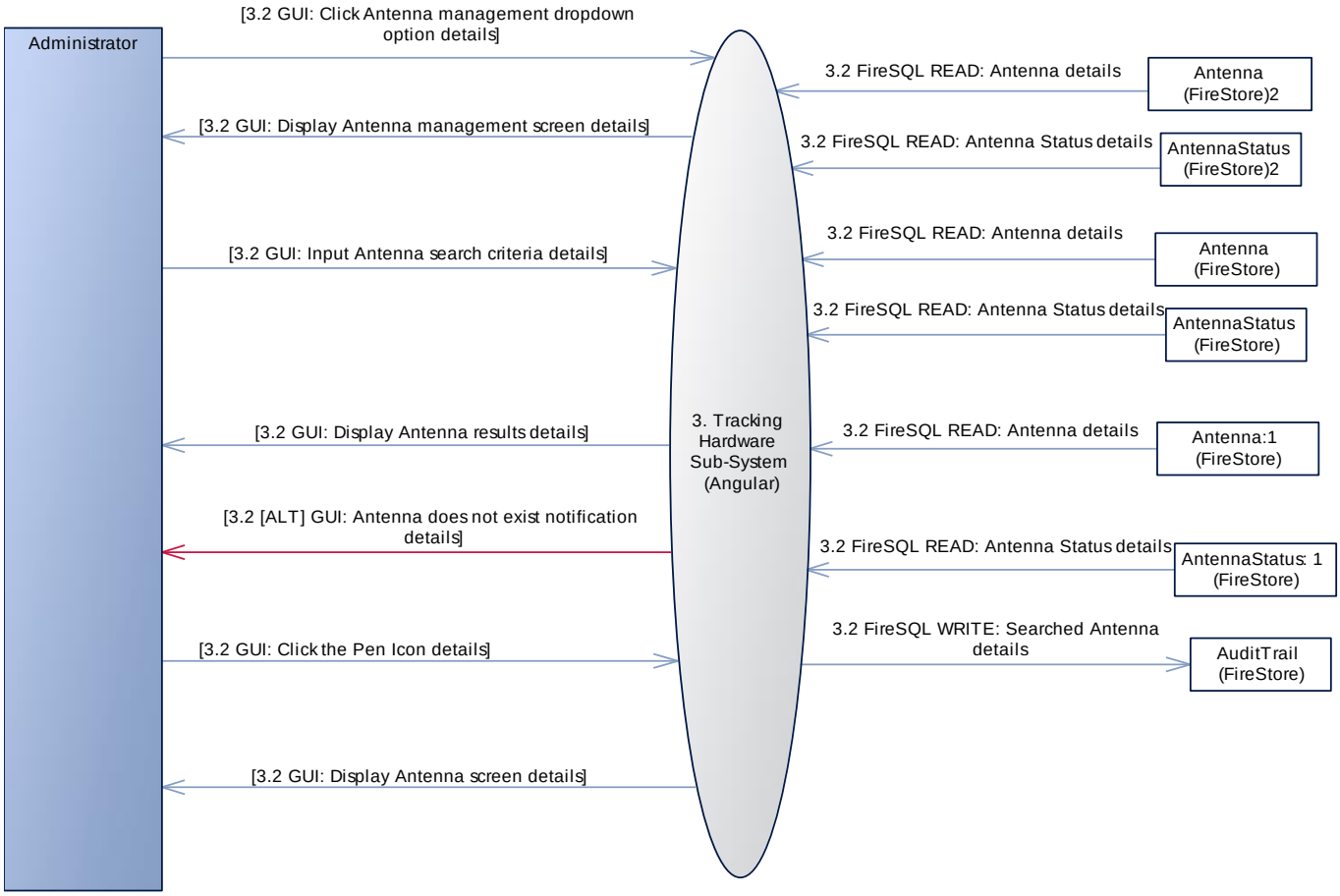


Figure 33 High-Level Diagram Sub-System 3: Part 2

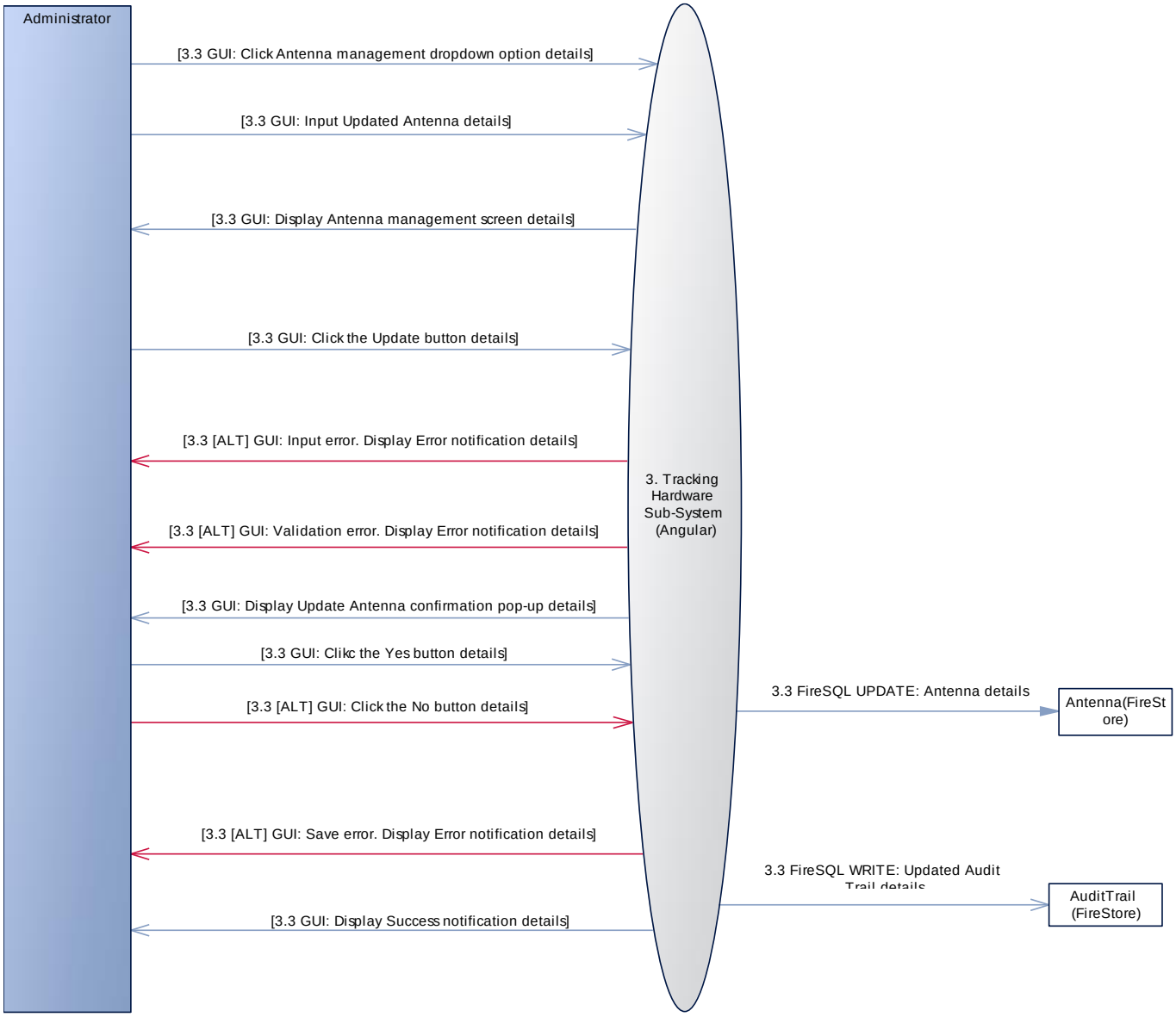


Figure 34 High-Level Diagram Sub-System 3: Part 3

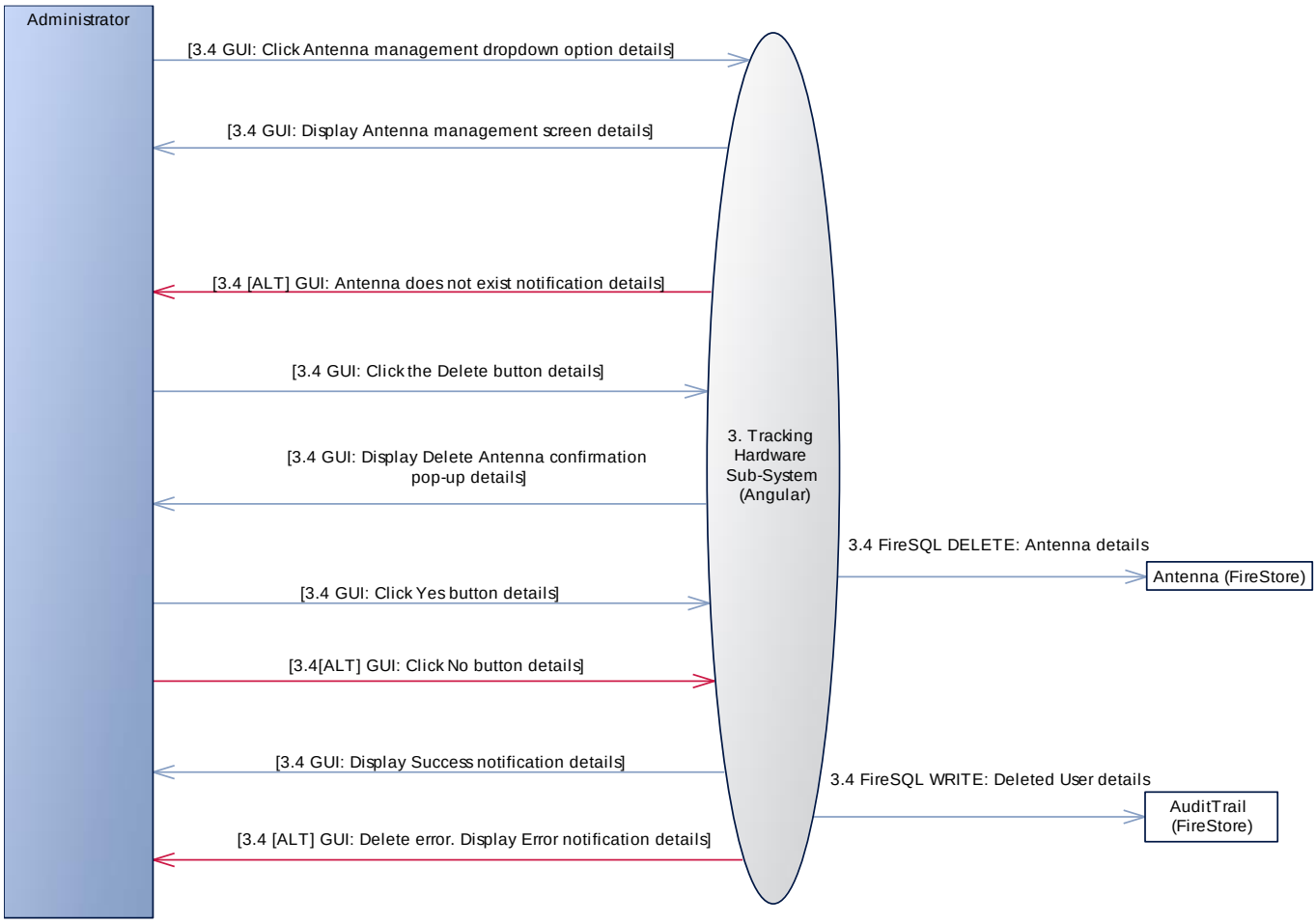


Figure 35 High-Level Diagram Sub-System 3: Part 4

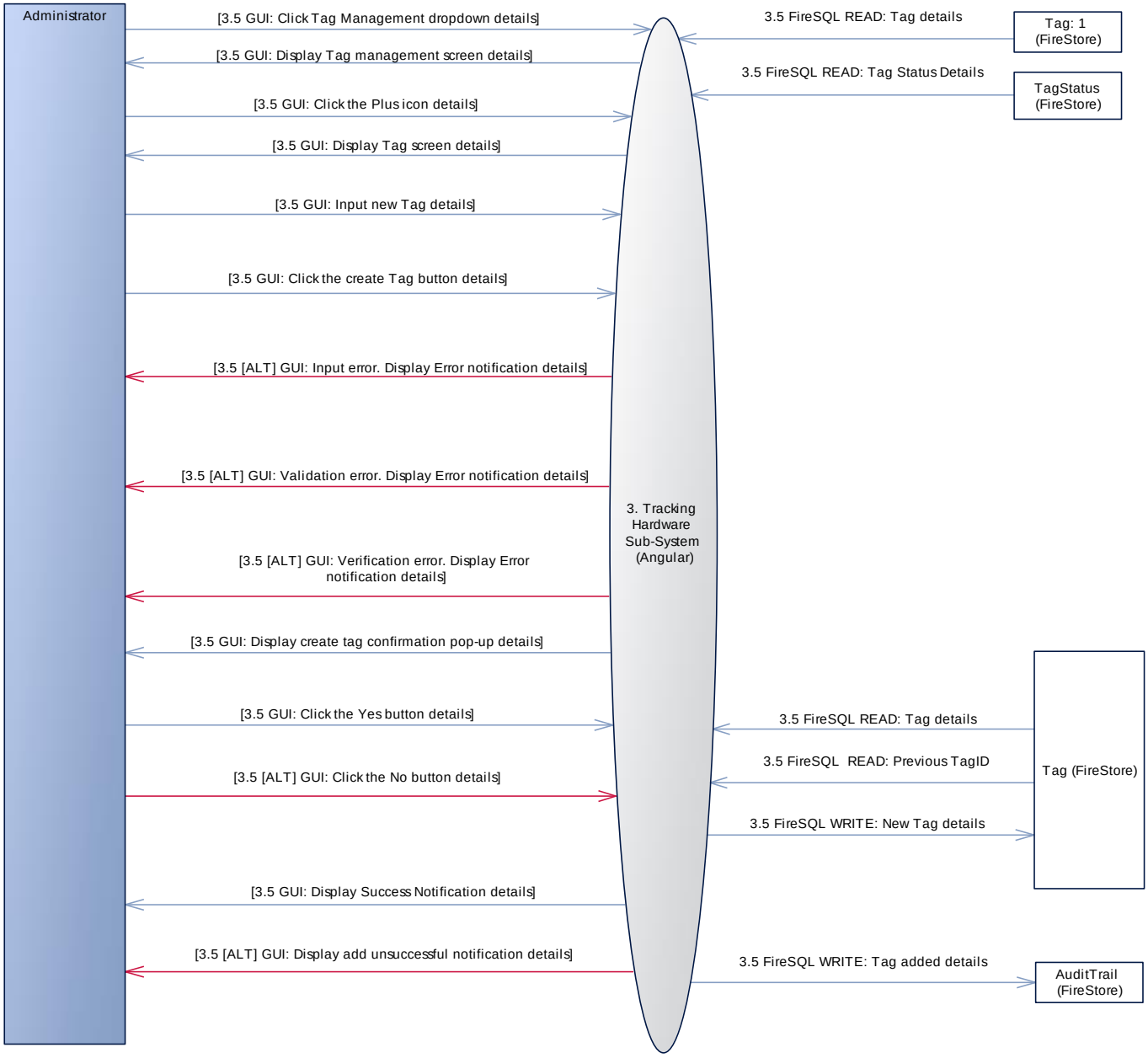


Figure 36 High-Level Diagram Sub-System 3: Part 5

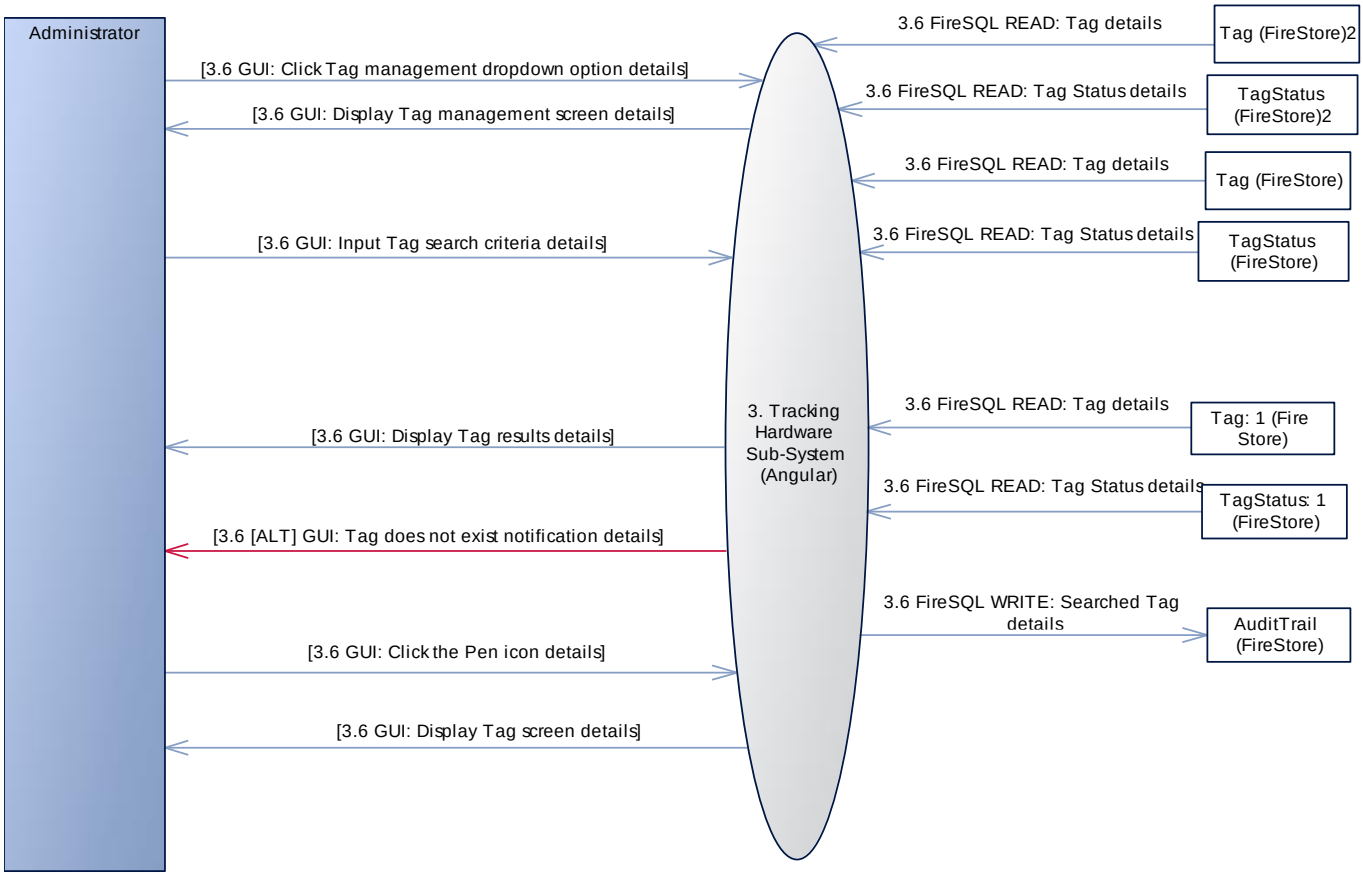


Figure 37 High-Level Diagram Sub-System 3: Part 6

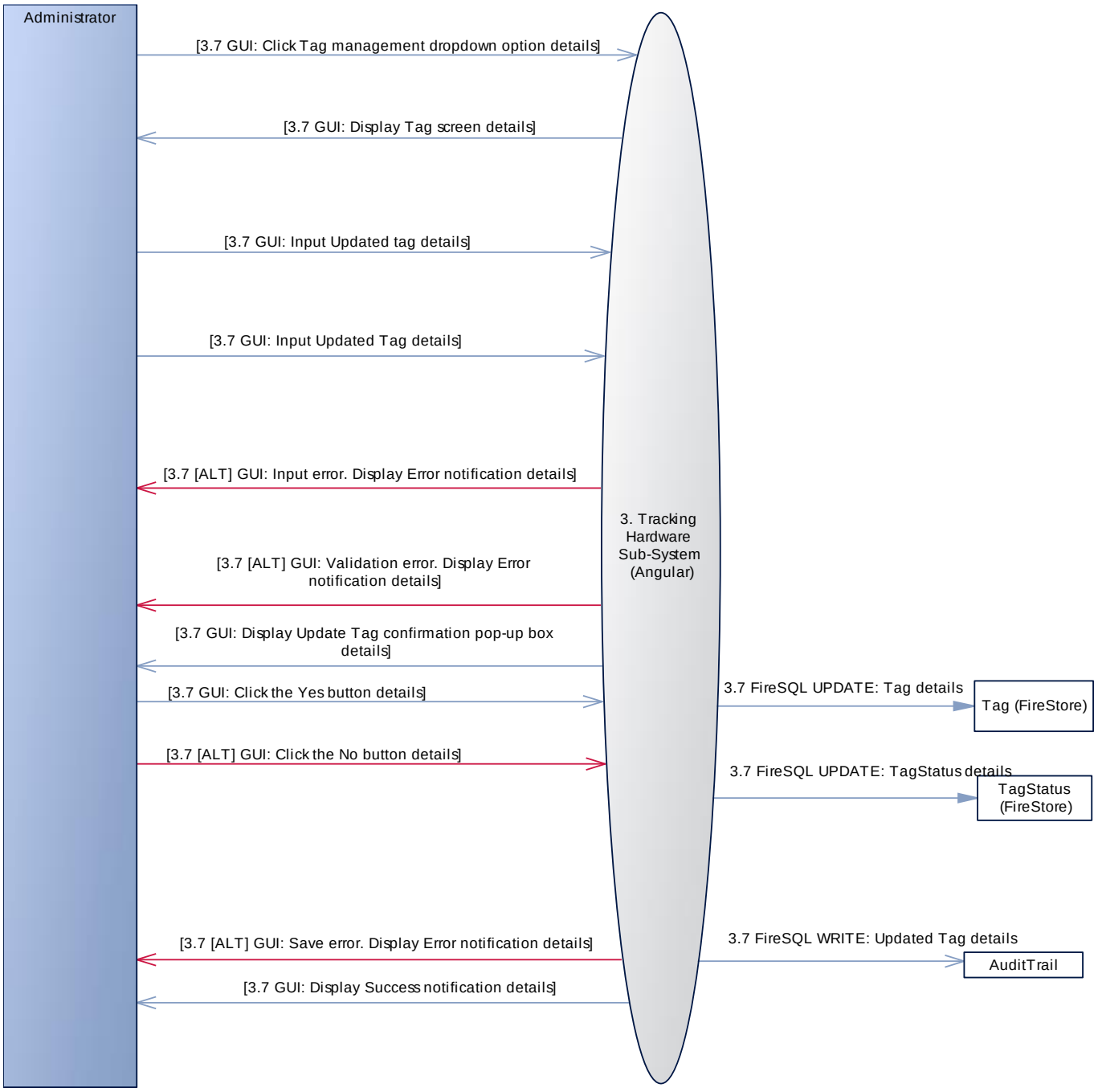


Figure 38 High-Level Diagram Sub-System 3: Part 7

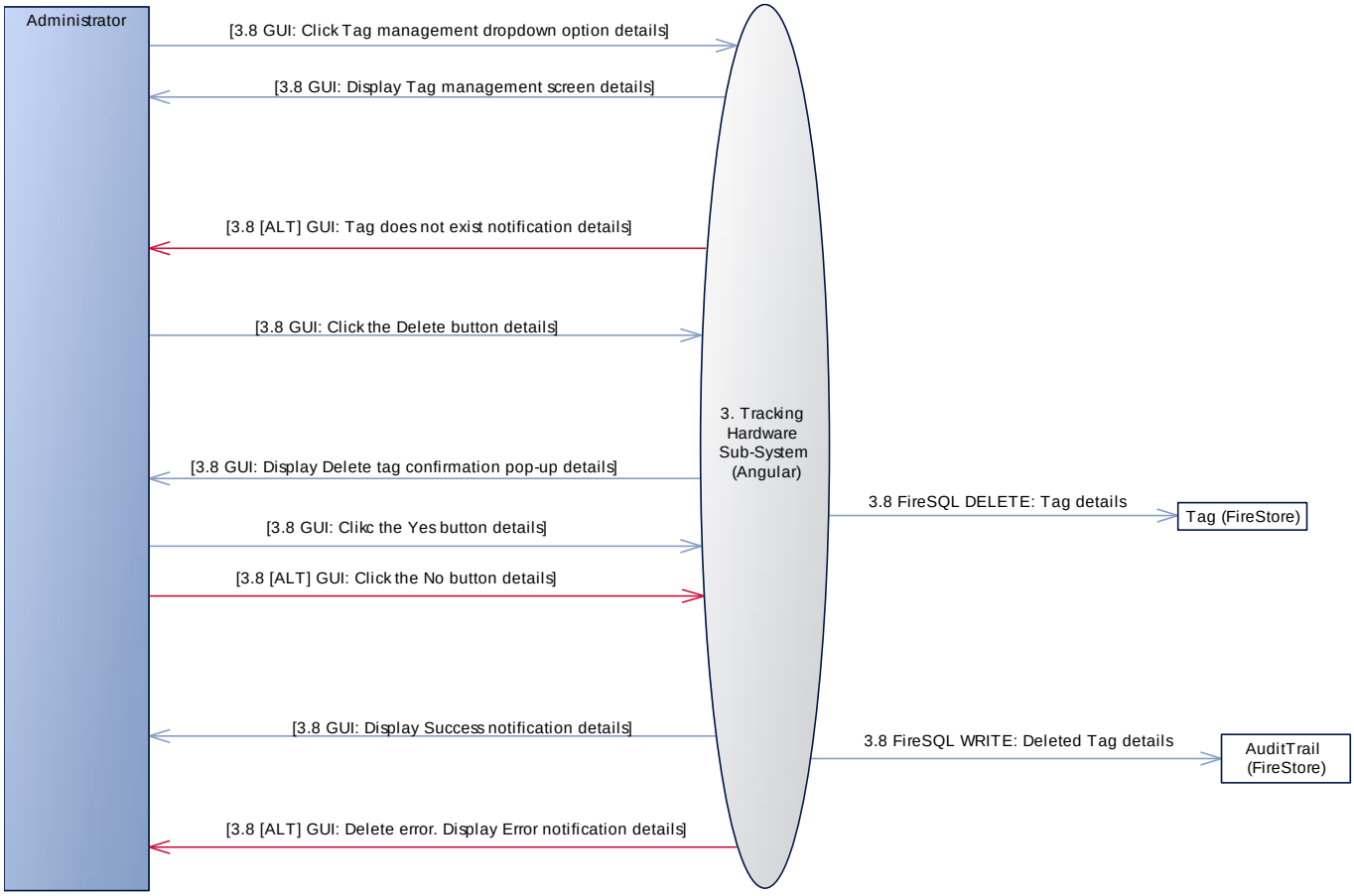


Figure 39 High-Level Diagram Sub-System 3: Part 8

Sub-System 4: Student

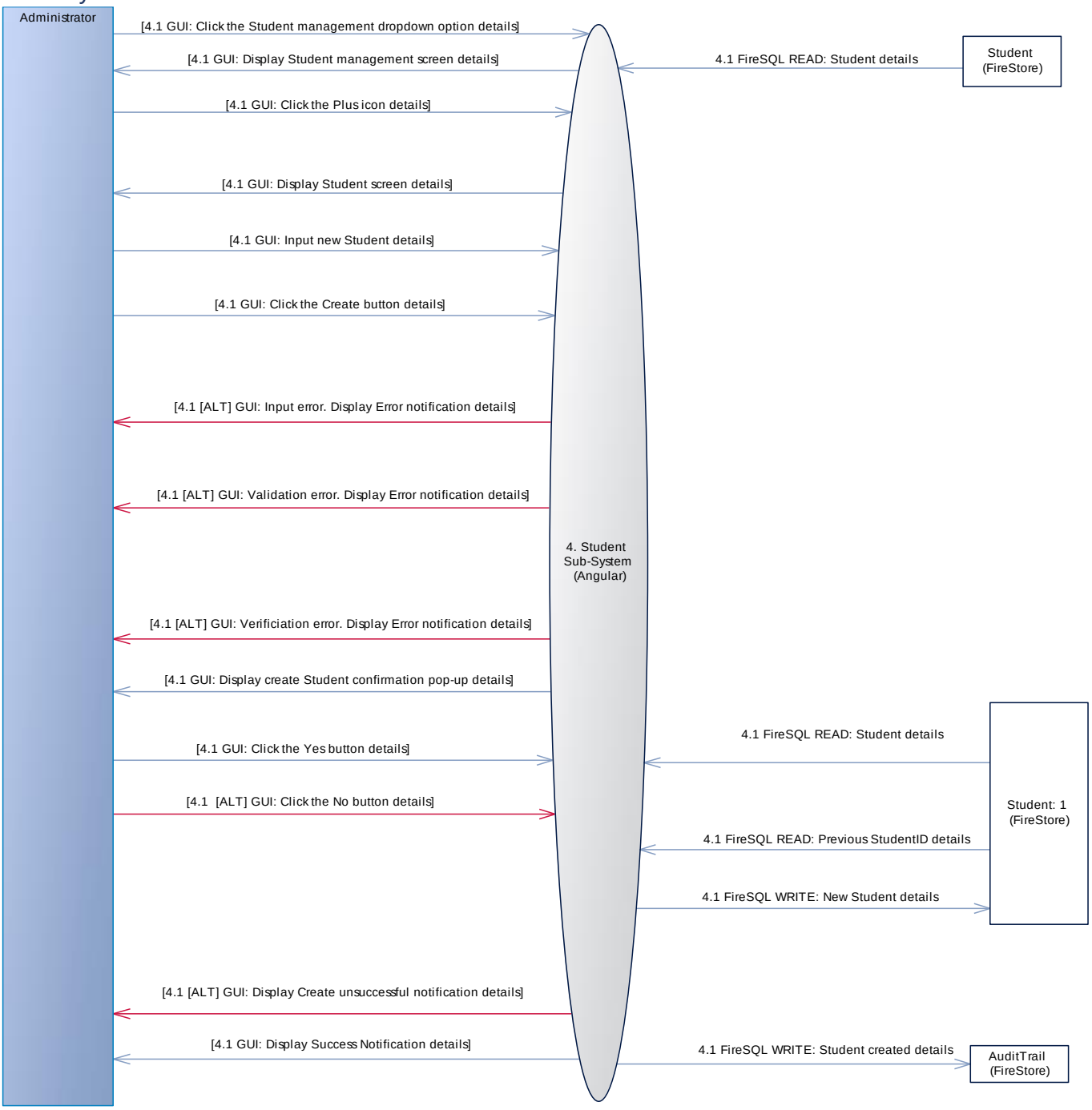


Figure 40 High-Level Diagram Sub-System 4: Part 1

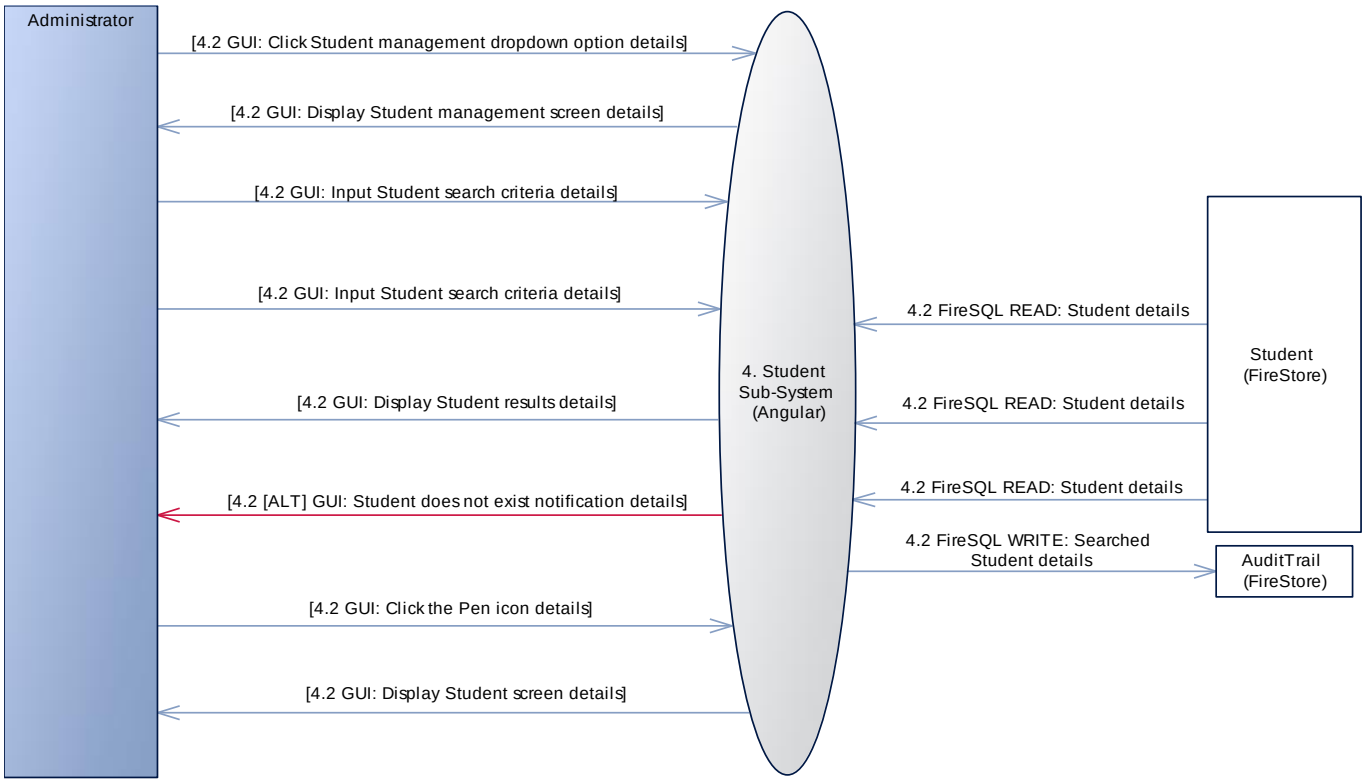


Figure 41 High-Level Diagram Sub-System 4: Part 2

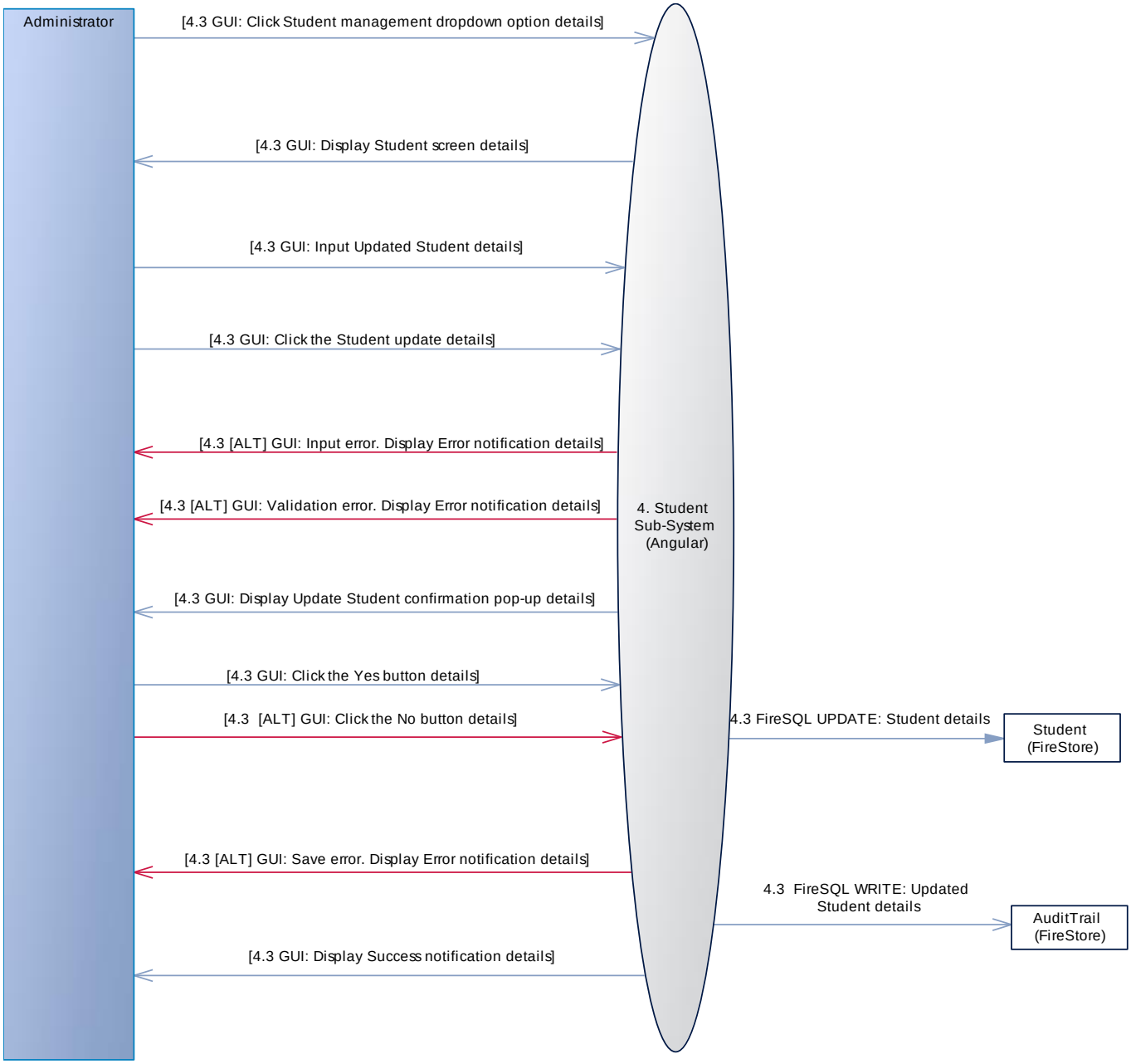


Figure 42 High-Level Diagram Sub-System 4: Part 3

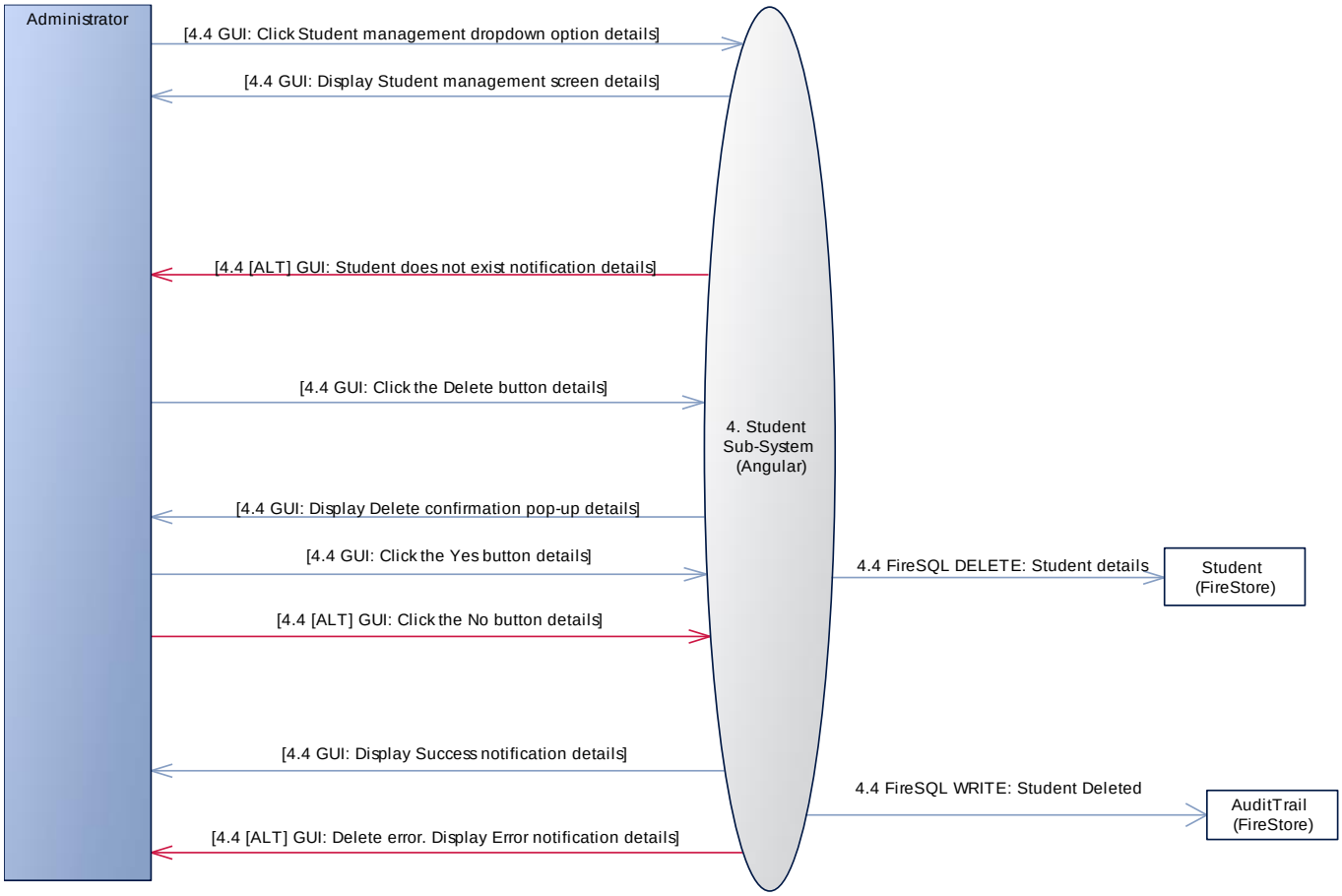


Figure 43 High-Level Diagram Sub-System 4: Part 4

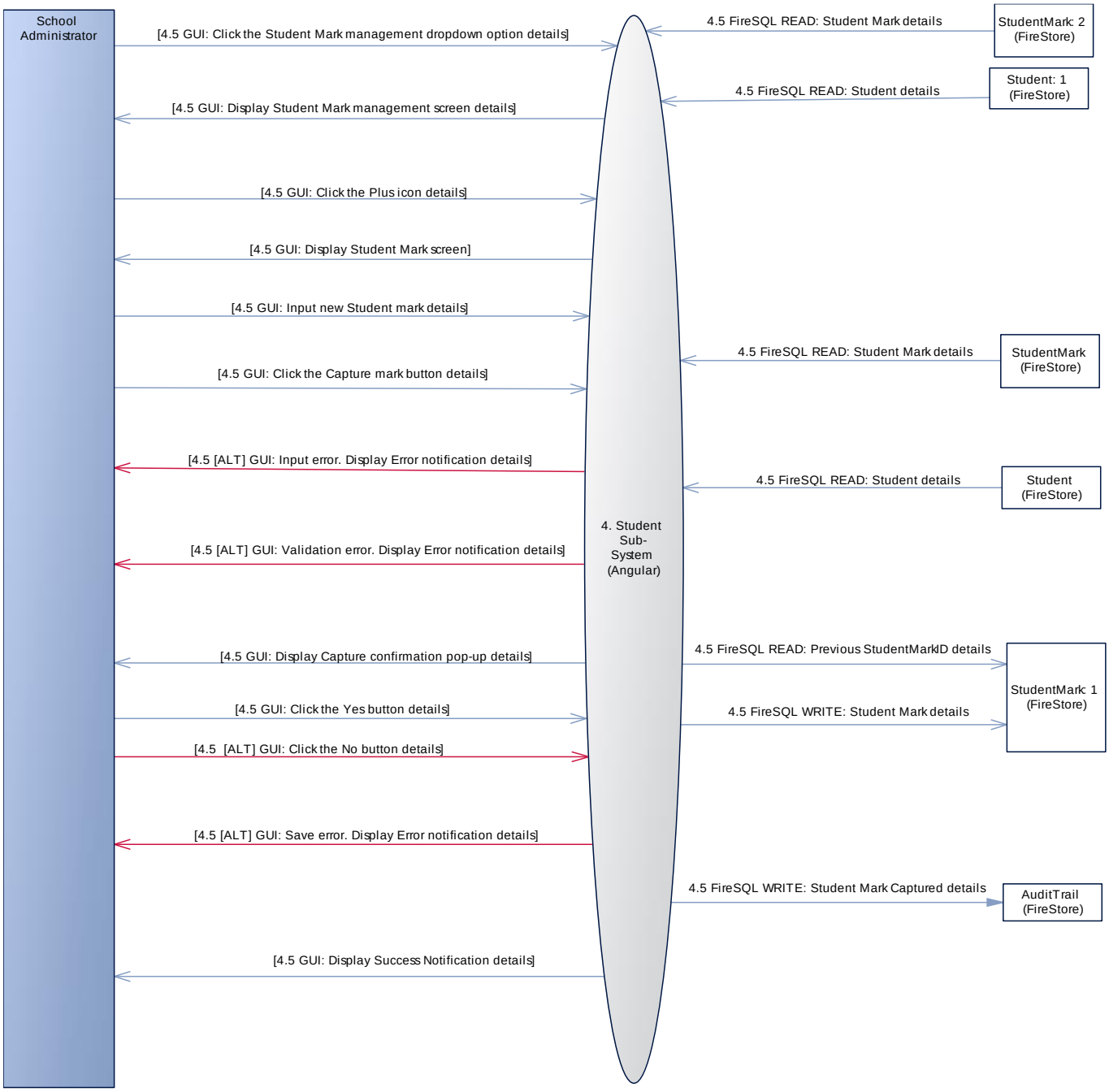


Figure 44 High-Level Diagram Sub-System 4: Part 5

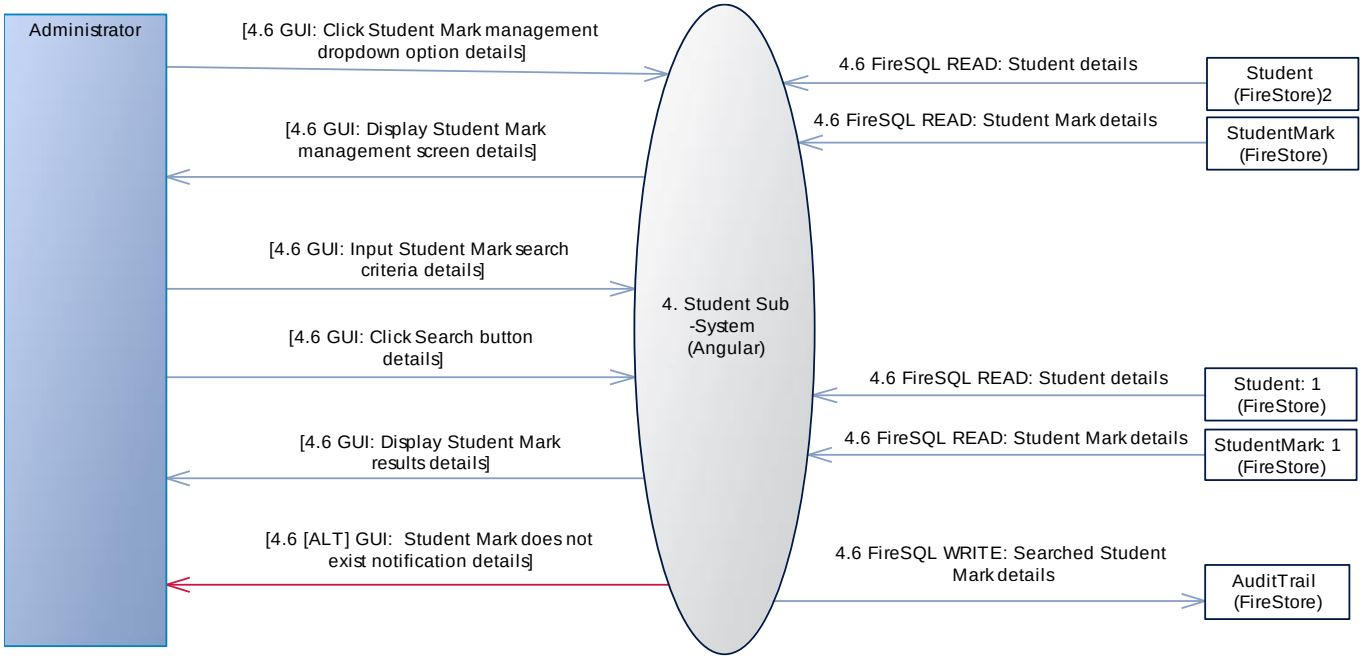


Figure 45 High-Level Diagram Sub-System 4: Part 6

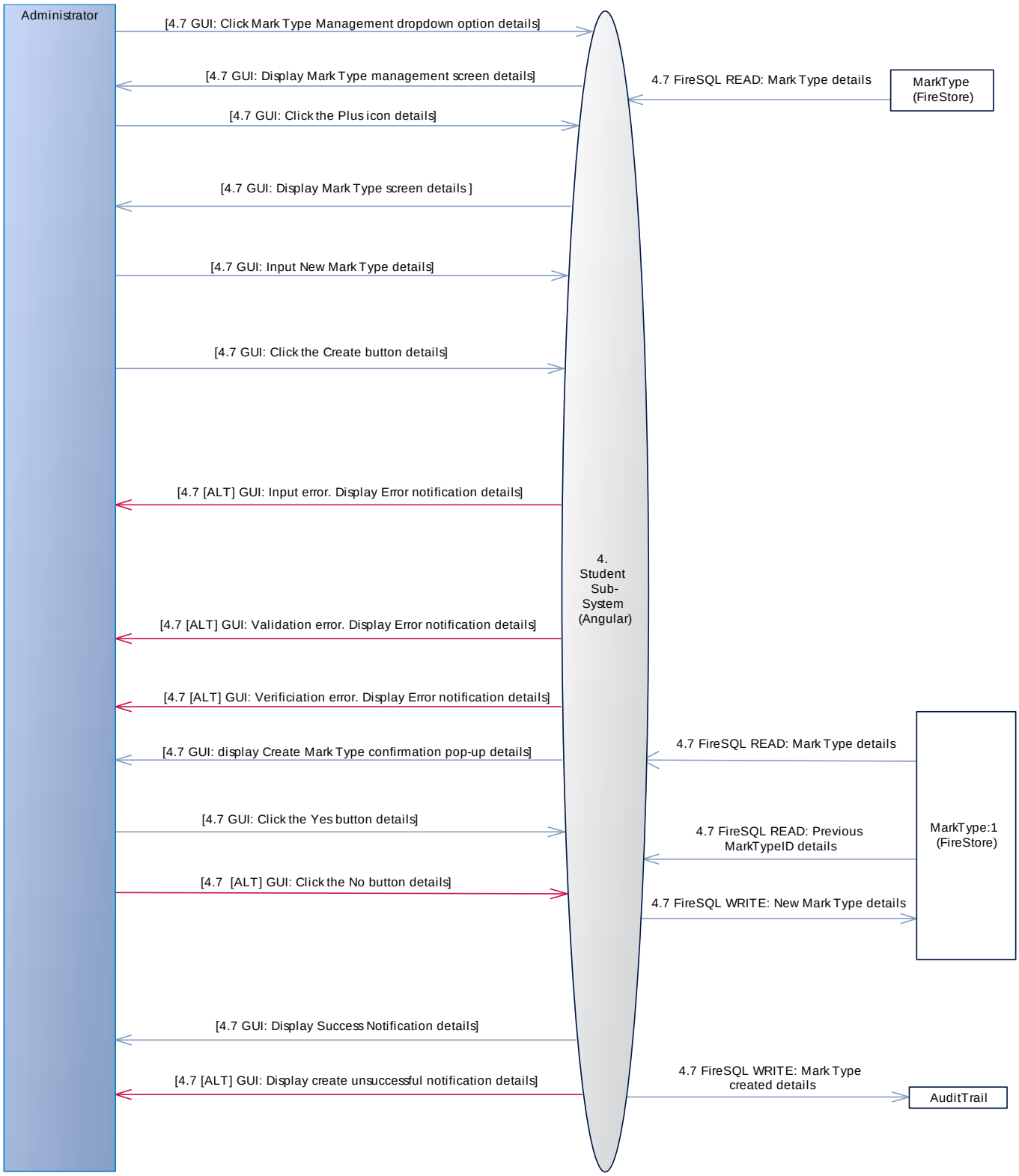


Figure 46 High-Level Diagram Sub-System 4: Part 7

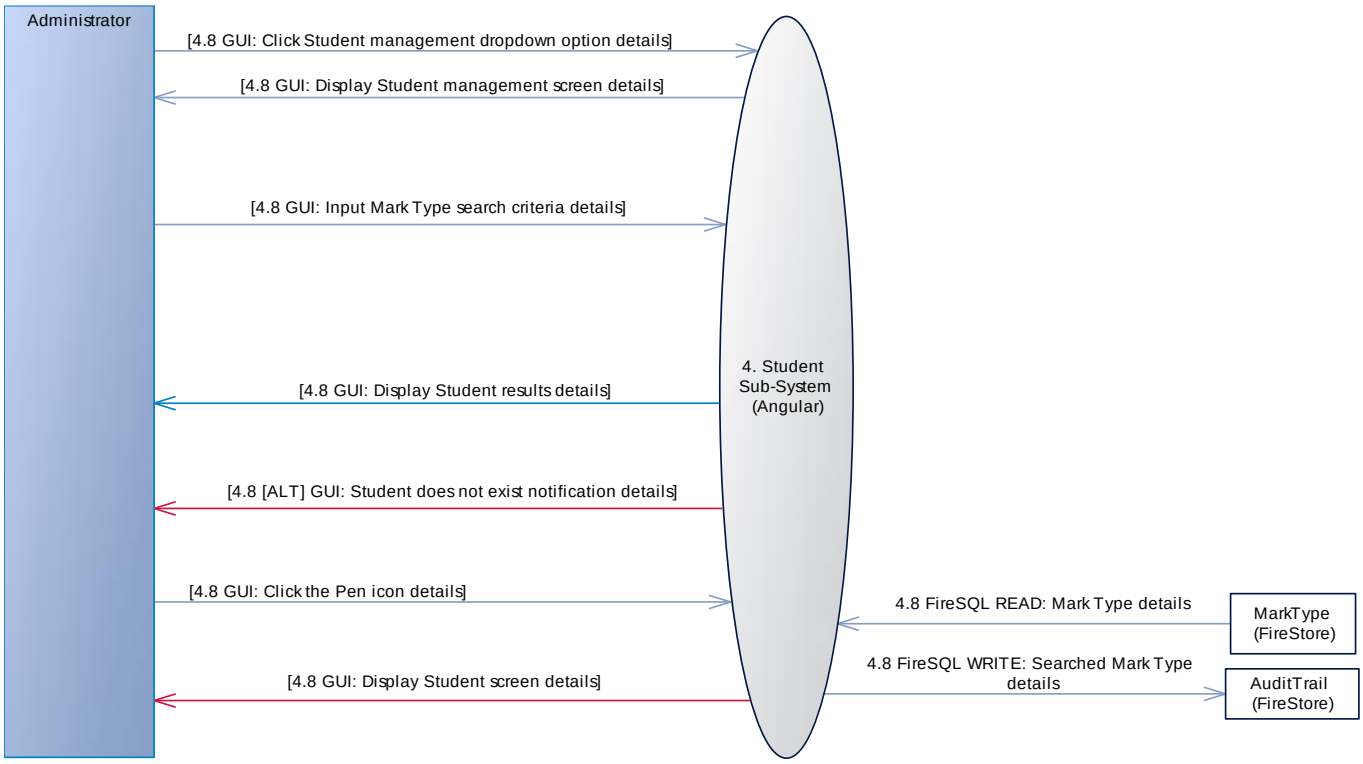


Figure 47 High-Level Diagram Sub-System 4: Part 8

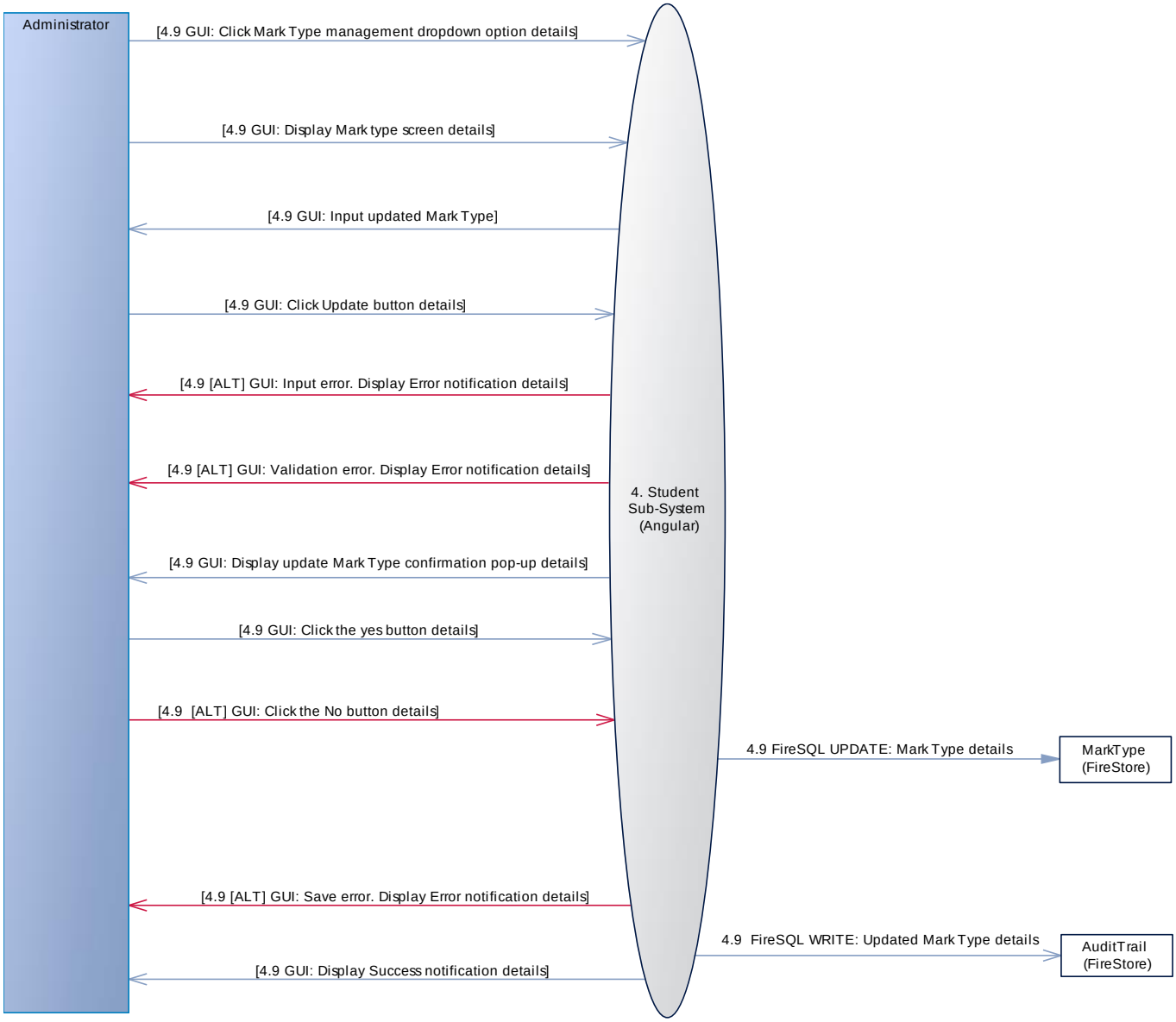


Figure 48 High-Level Diagram Sub-System 4: Part 9

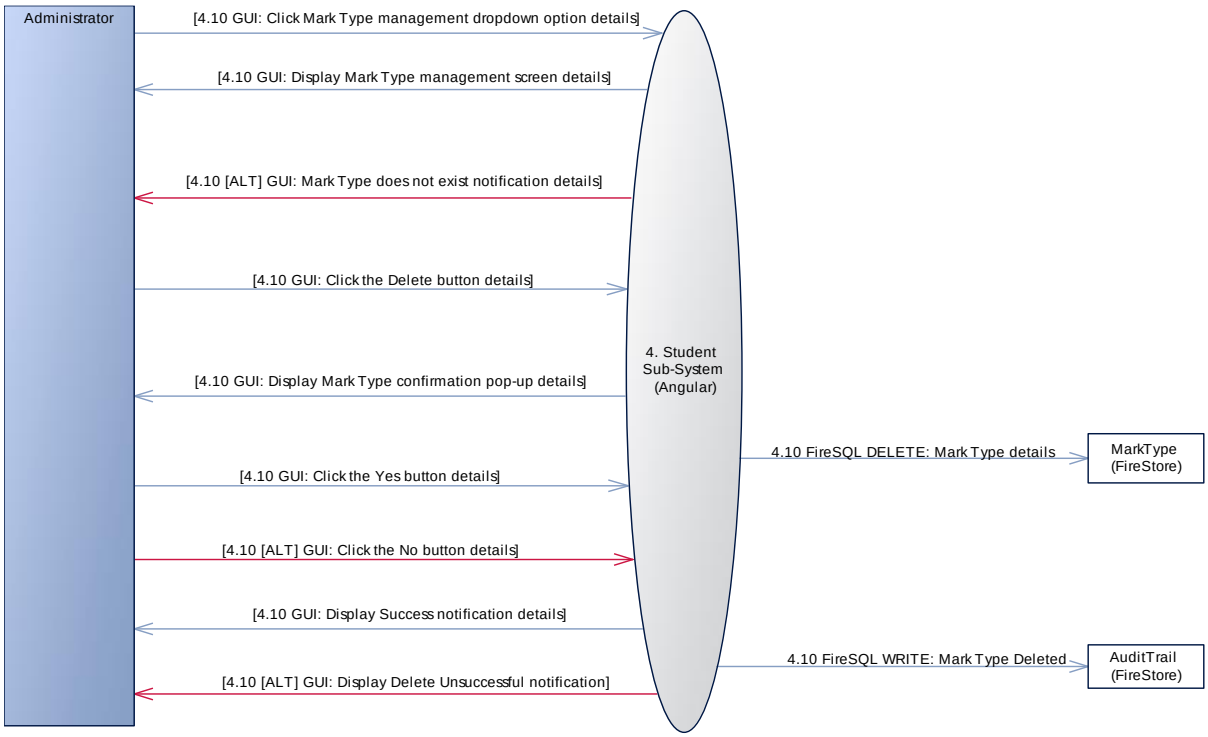


Figure 49 High-Level Diagram Sub-System 4: Part 10

Sub-System 5: Bicycle

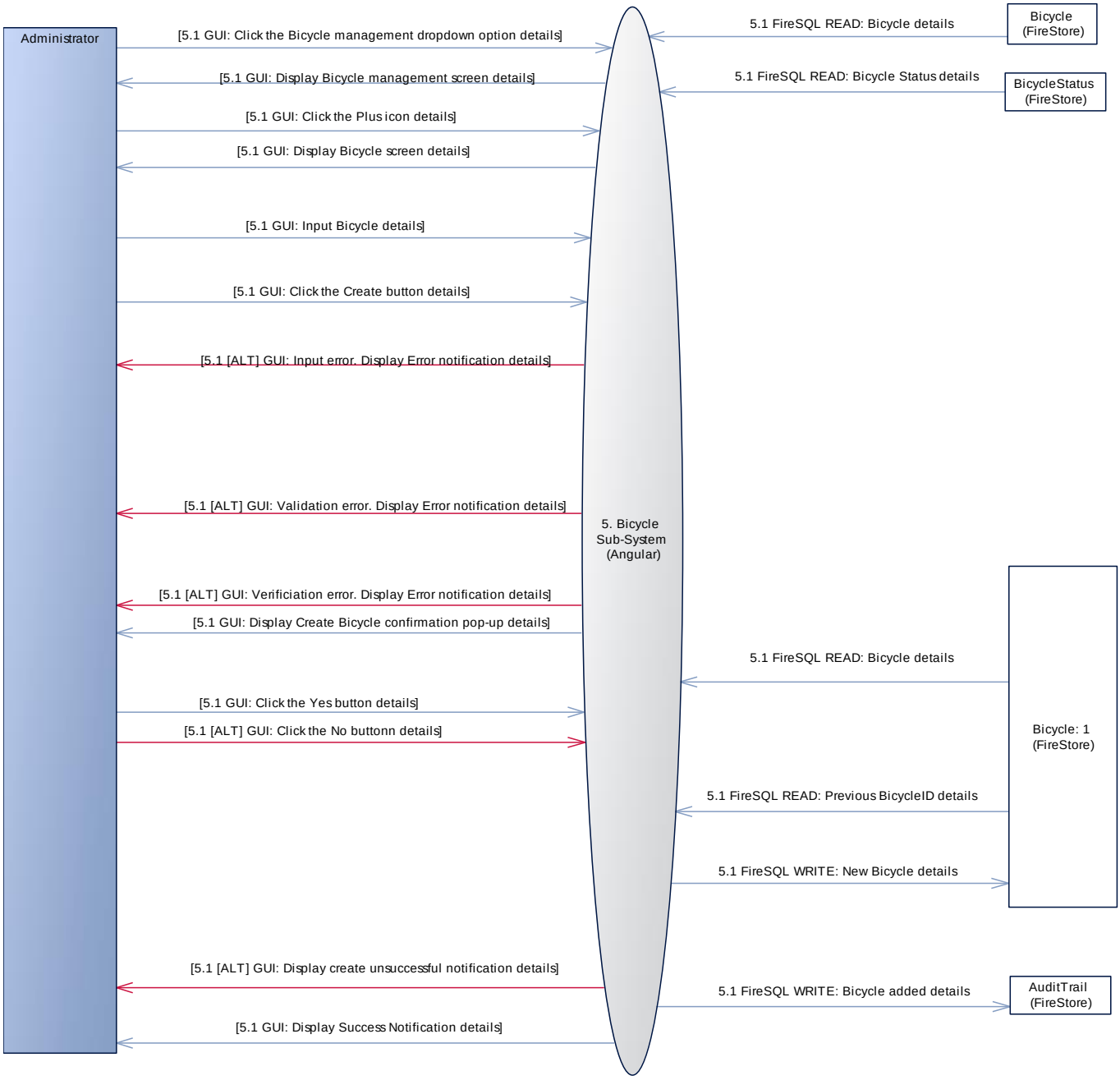


Figure 50 High-Level Diagram Sub-System 5: Part 1

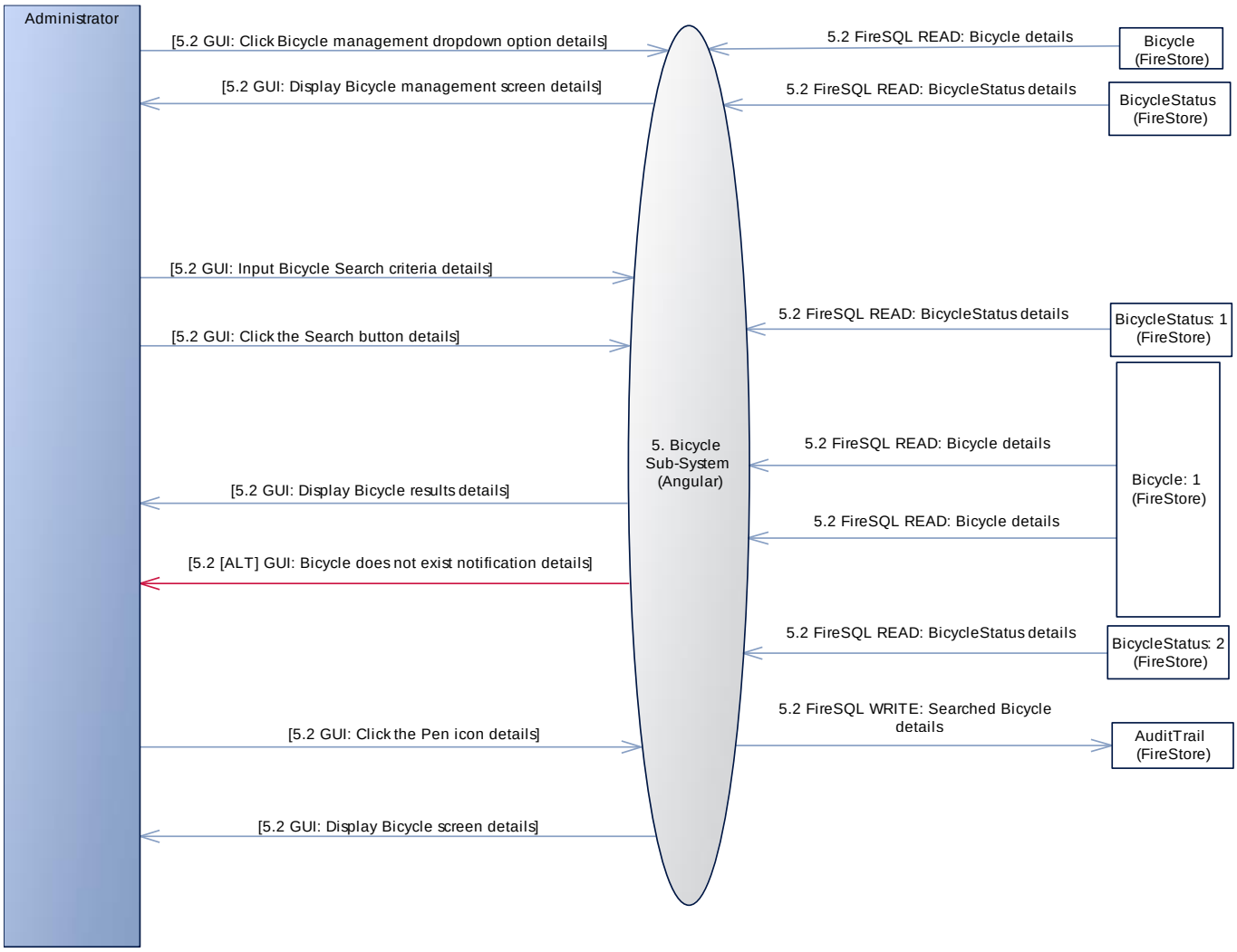


Figure 51 High-Level Diagram Sub-System 5: Part 2

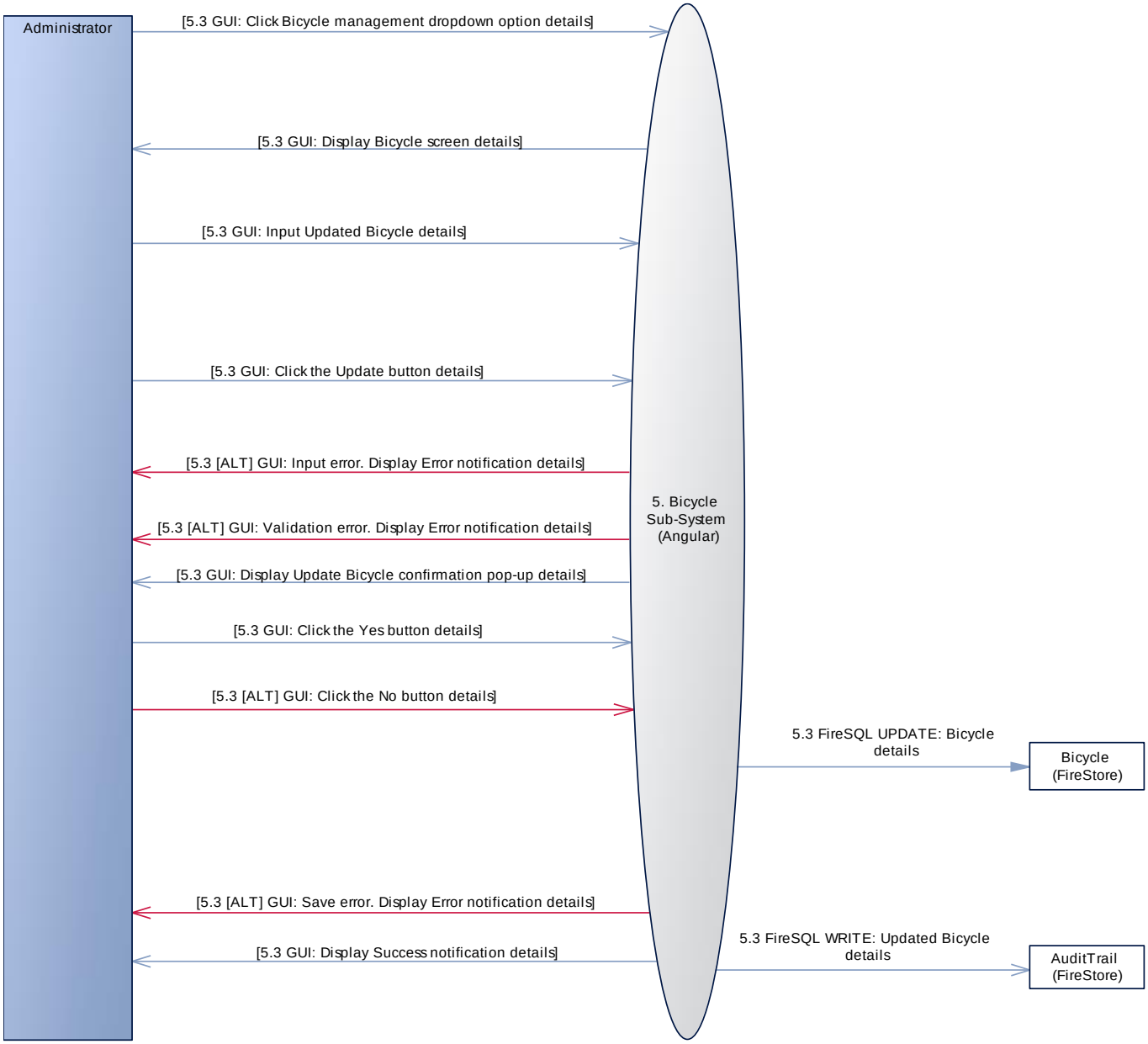


Figure 52 High-Level Diagram Sub-System 5: Part 3

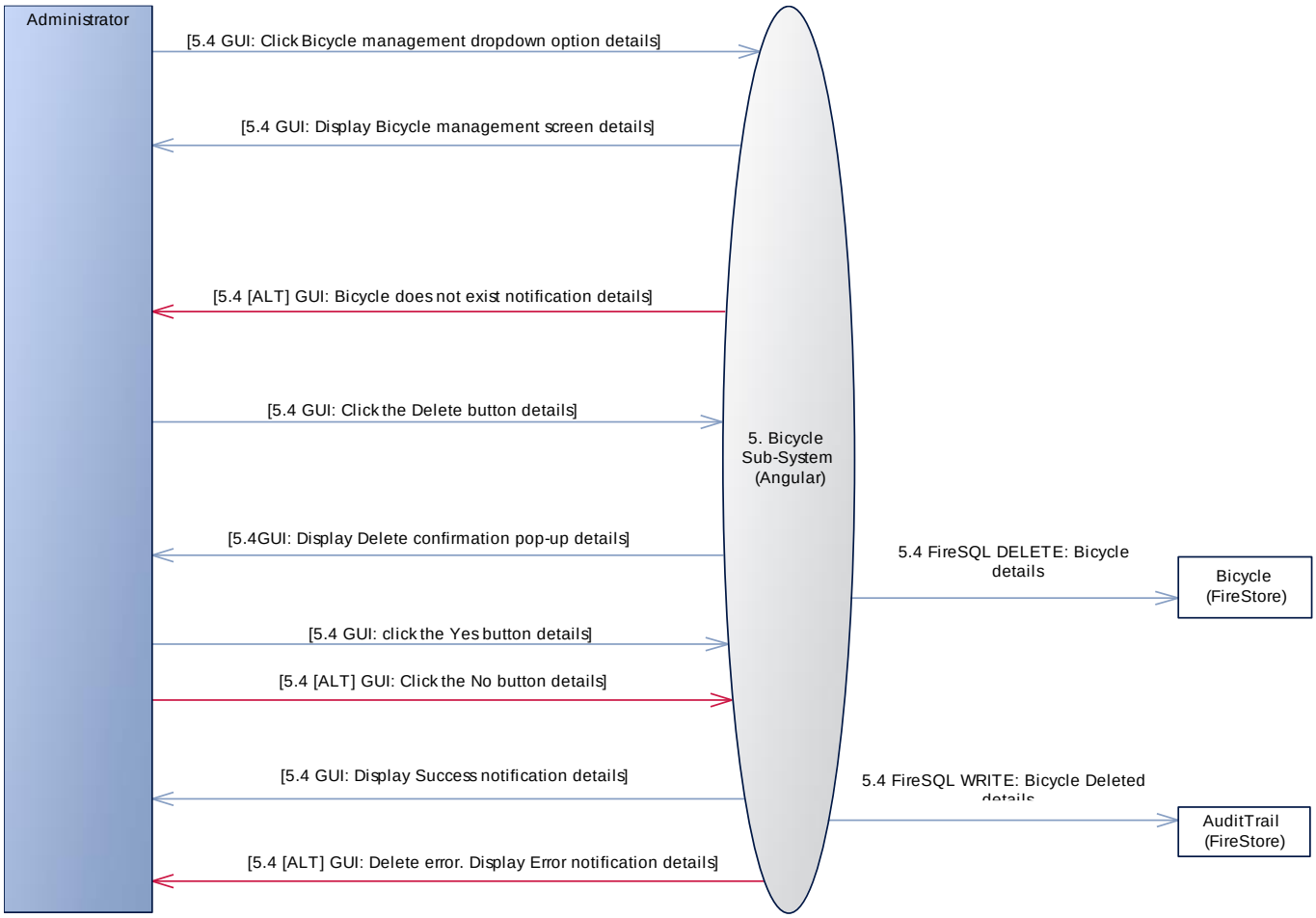


Figure 53 High-Level Diagram Sub-System 5: Part 4

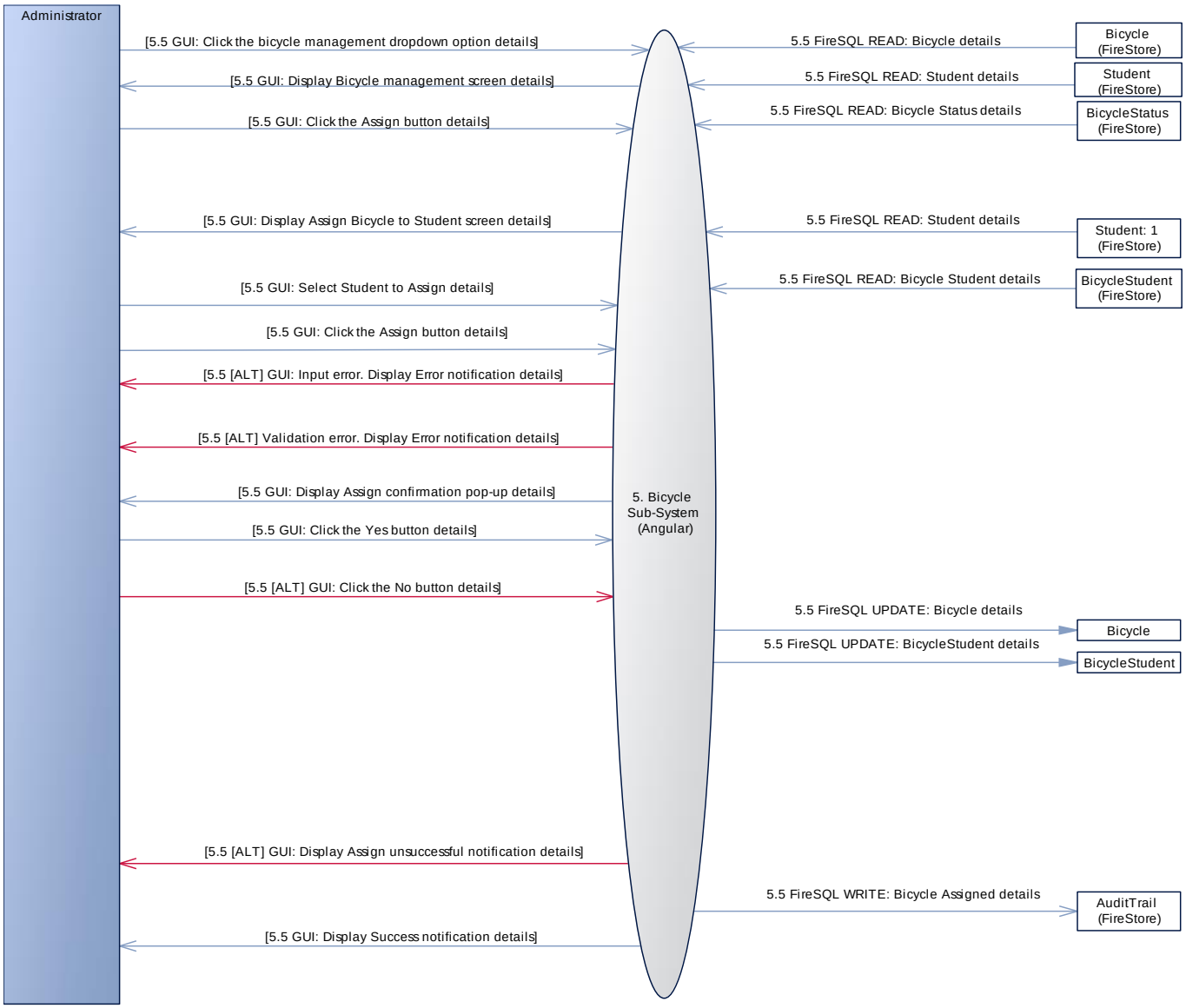


Figure 54 High-Level Diagram Sub-System 5: Part 5

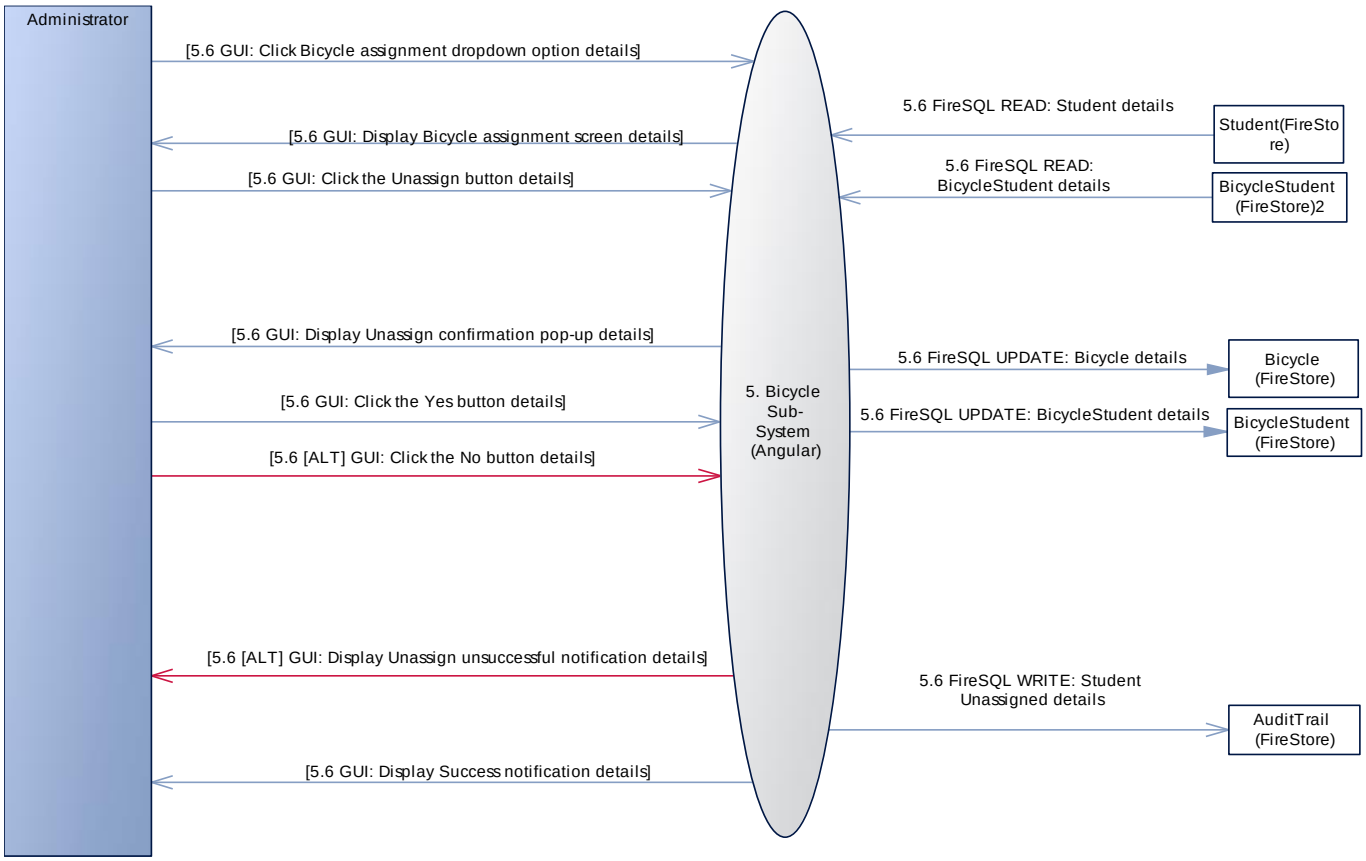


Figure 55 High-Level Diagram Sub-System 5: Part 6

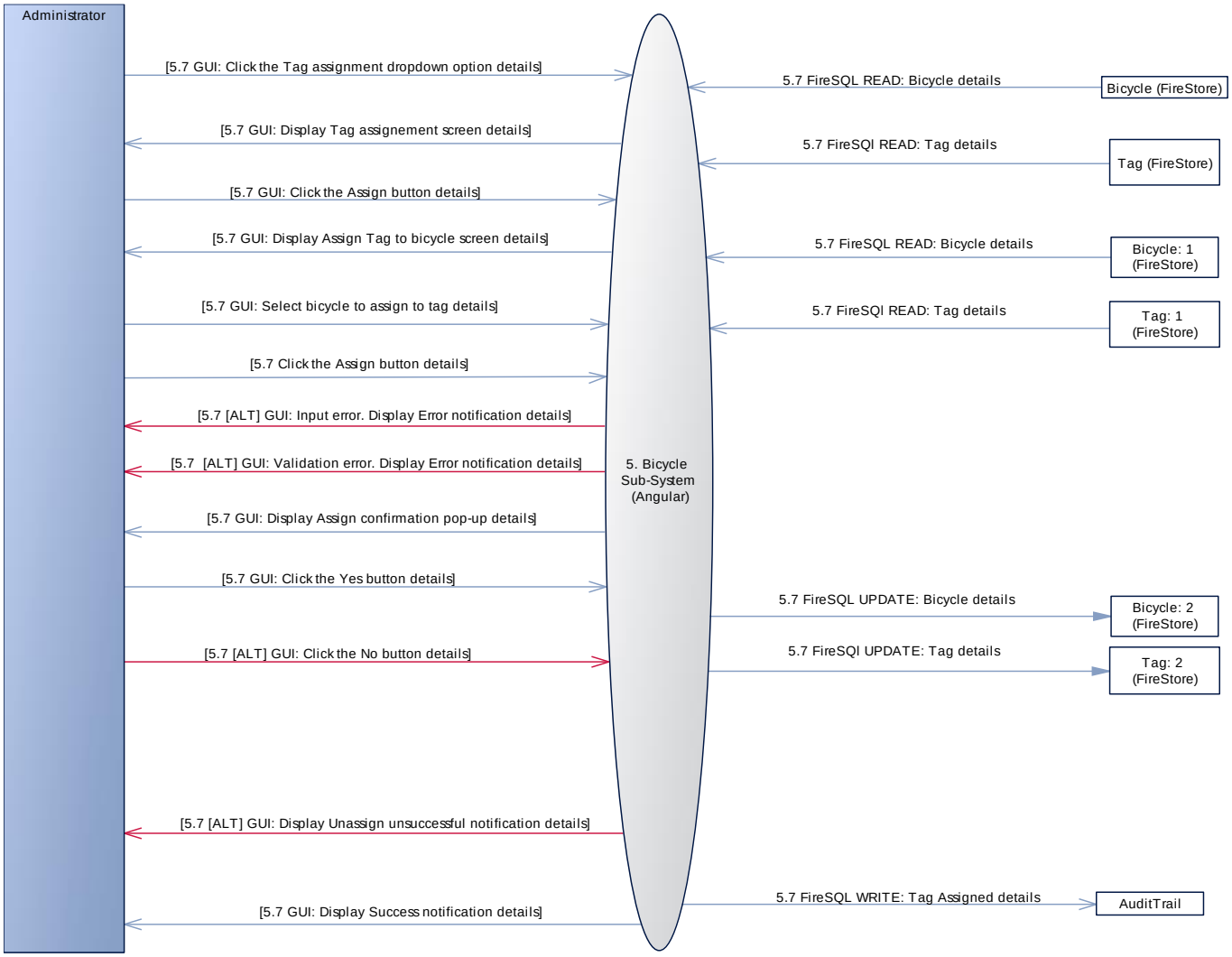


Figure 56 High-Level Diagram Sub-System 5: Part 7

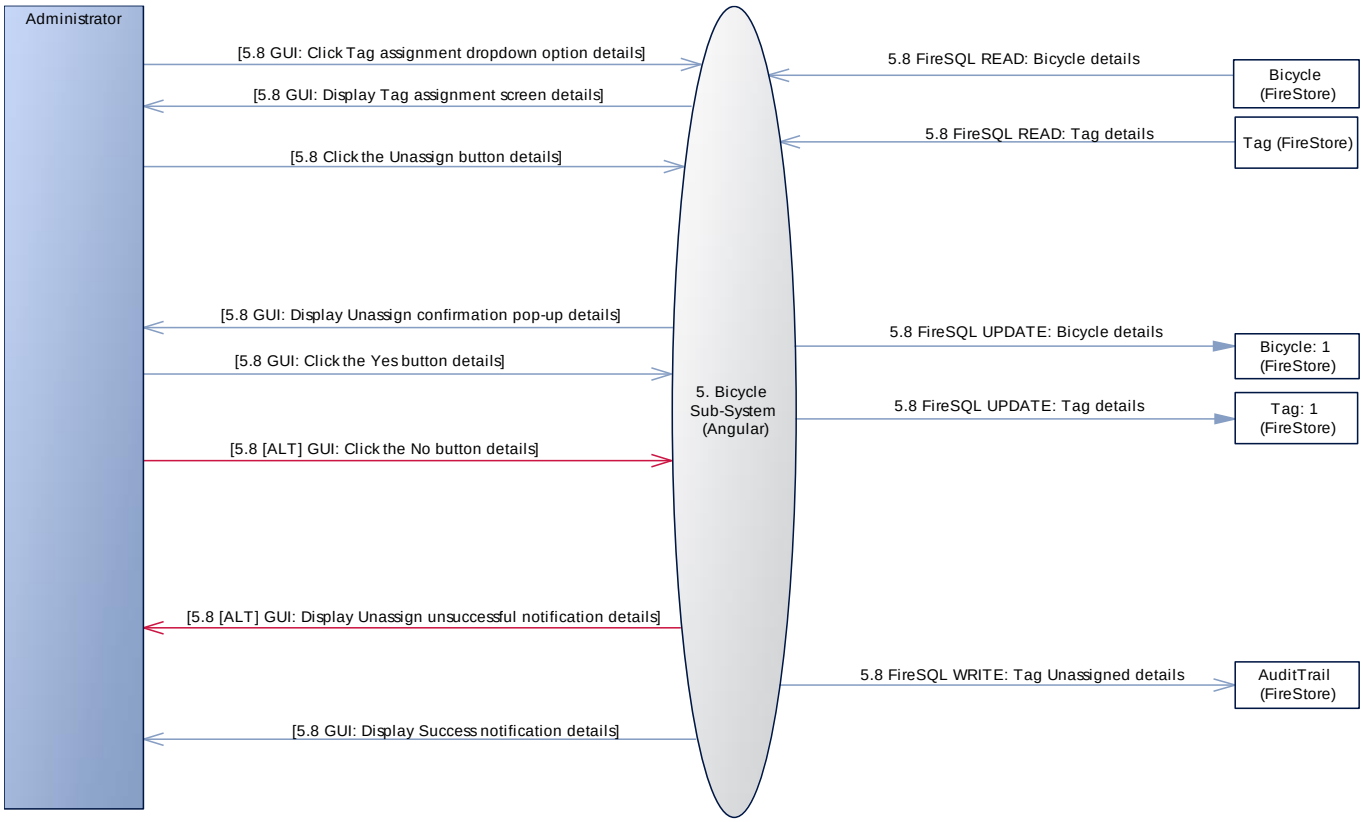


Figure 57 High-Level Diagram Sub-System 5: Part 8

Sub-System 6: Bicycle Part

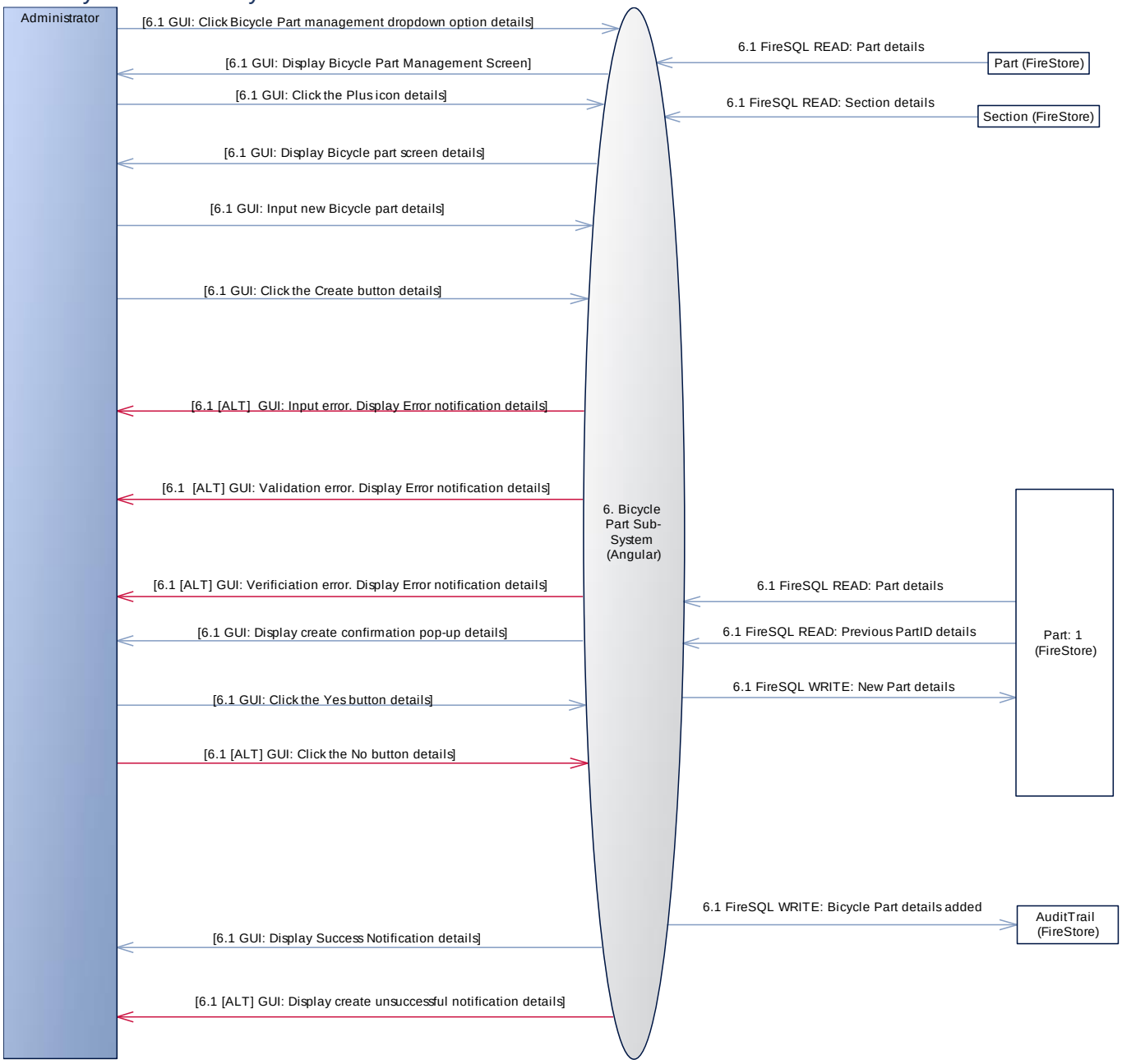


Figure 58 High-Level Diagram Sub-System 6: Part 1

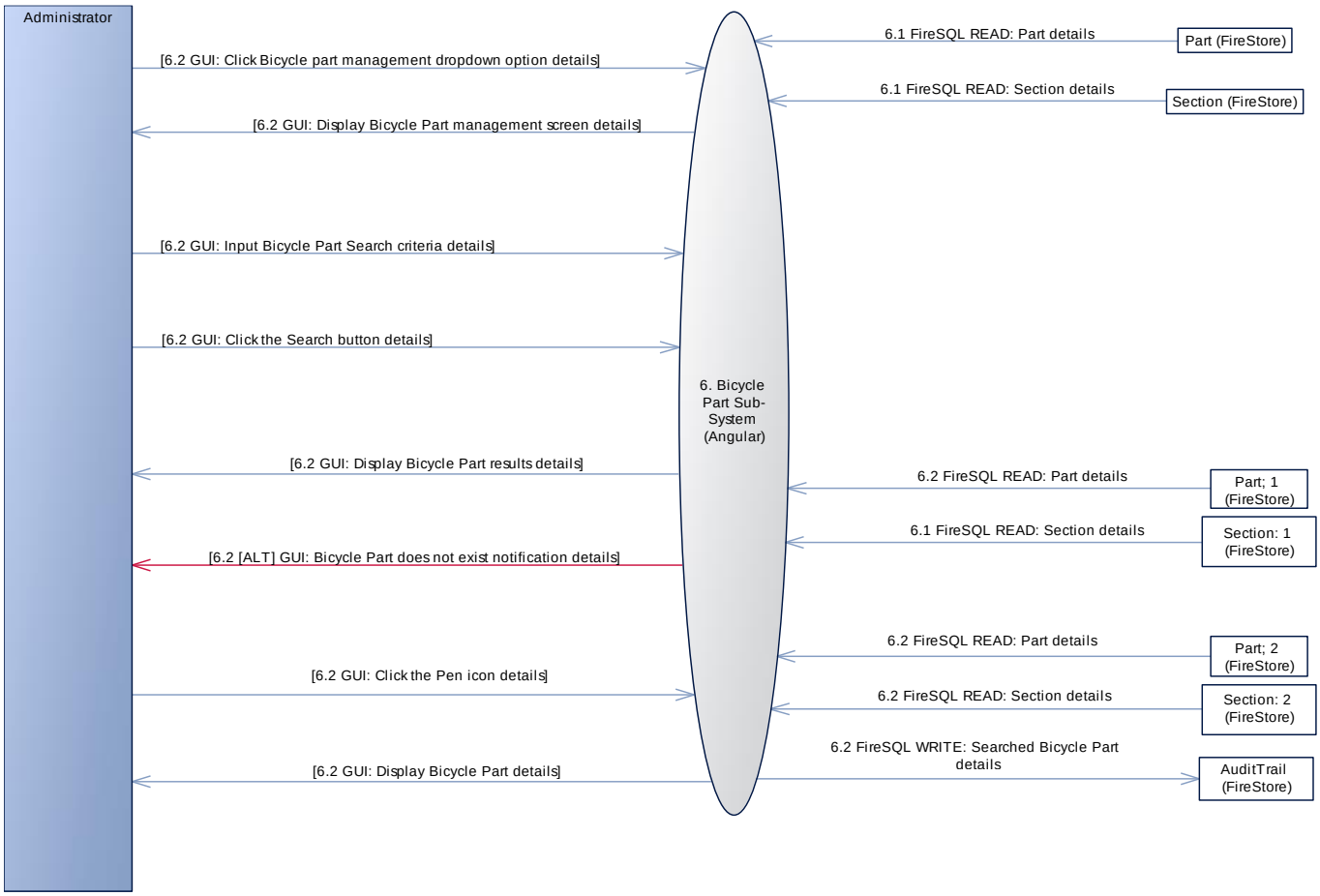


Figure 59 High-Level Diagram Sub-System 6: Part 2

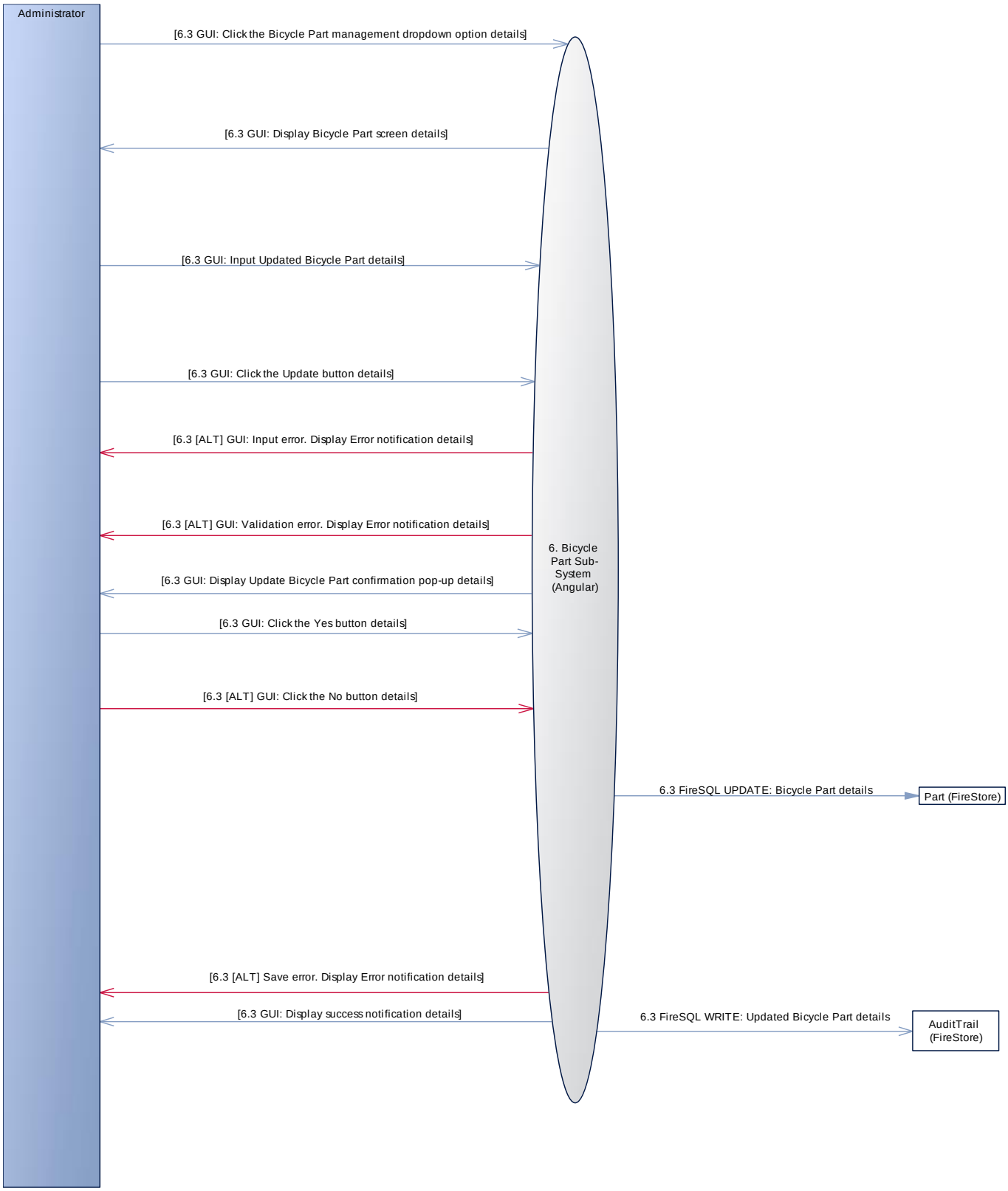


Figure 60 High-Level Diagram Sub-System 6: Part 3

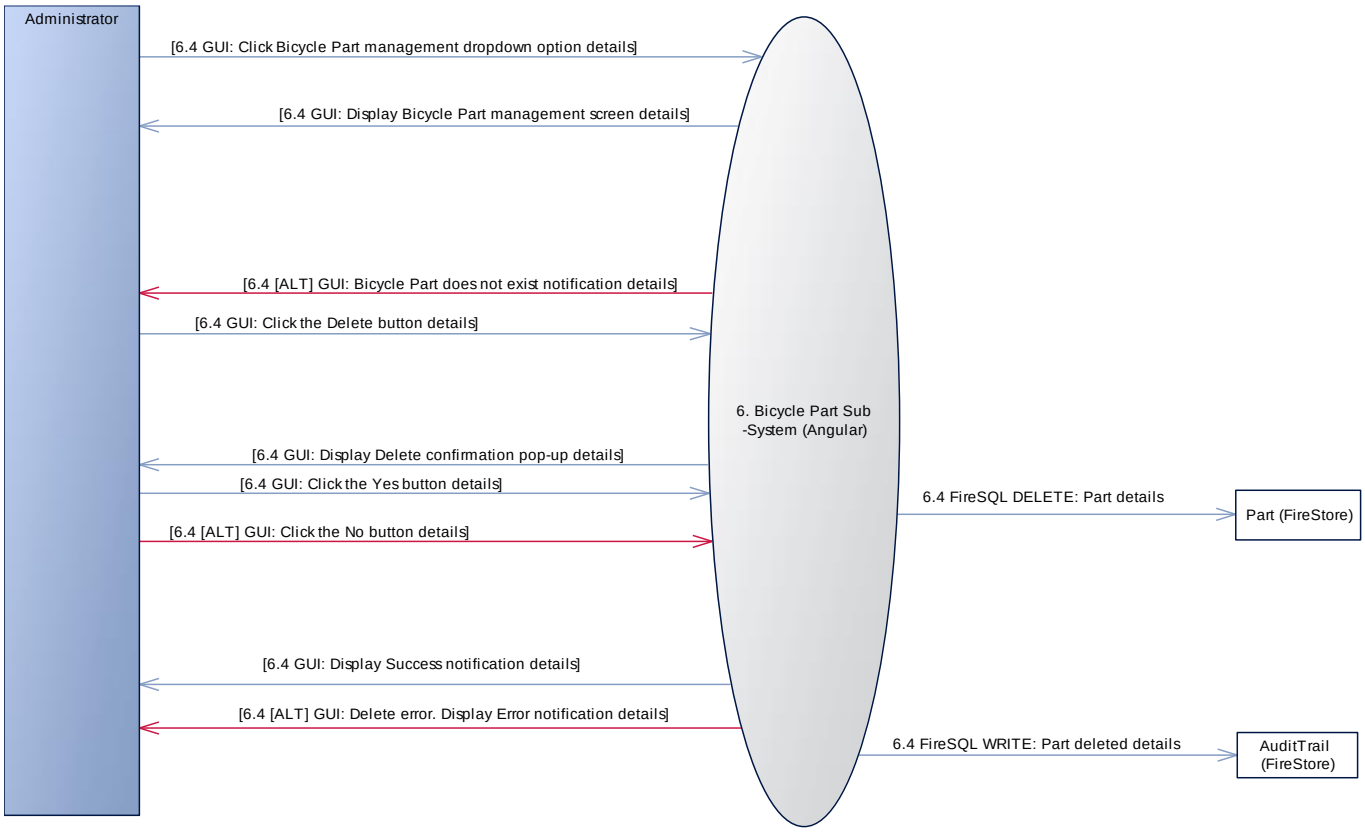


Figure 61 High-Level Diagram Sub-System 6: Part 4

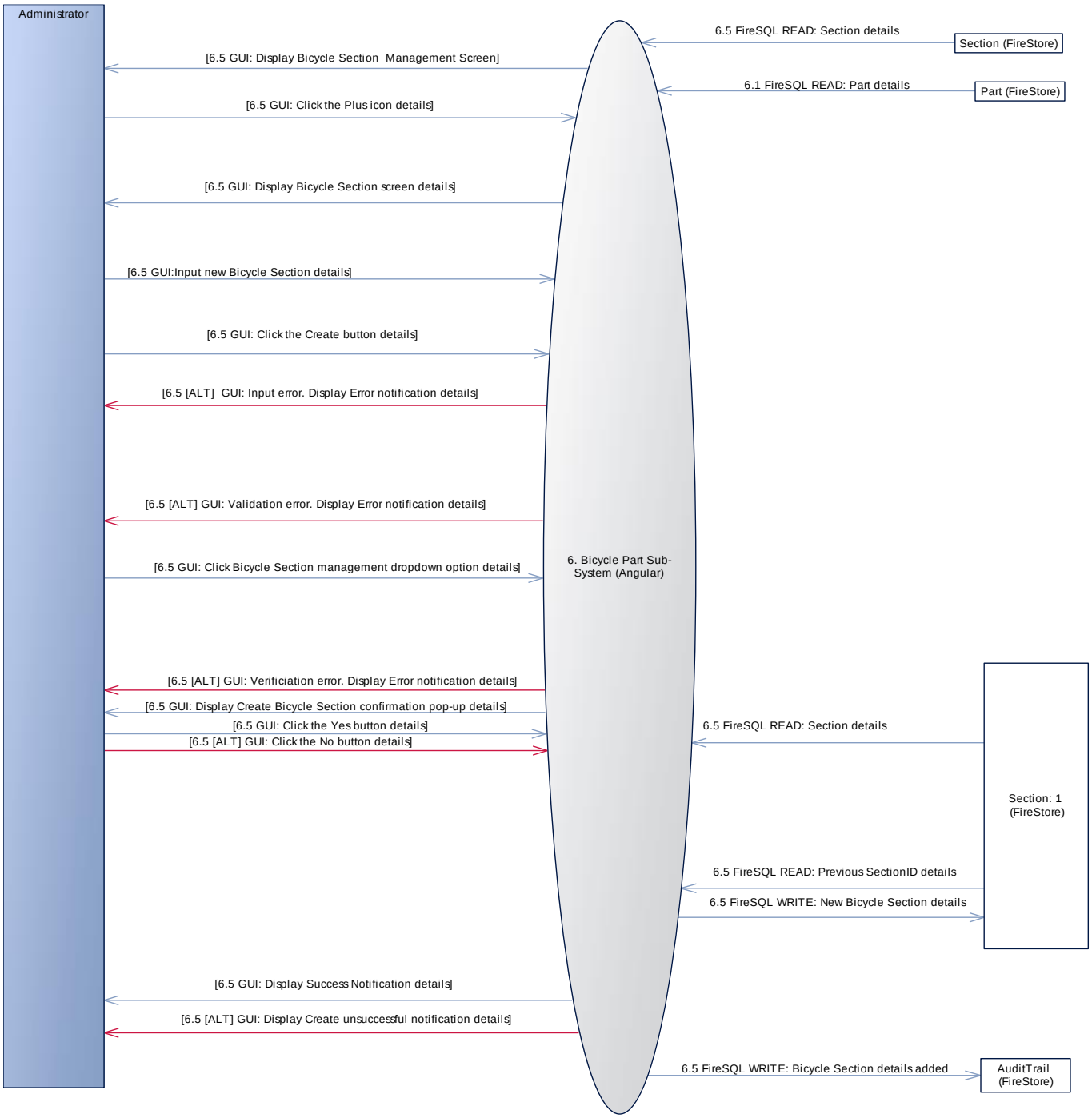


Figure 62 High-Level Diagram Sub-System 6: Part 5

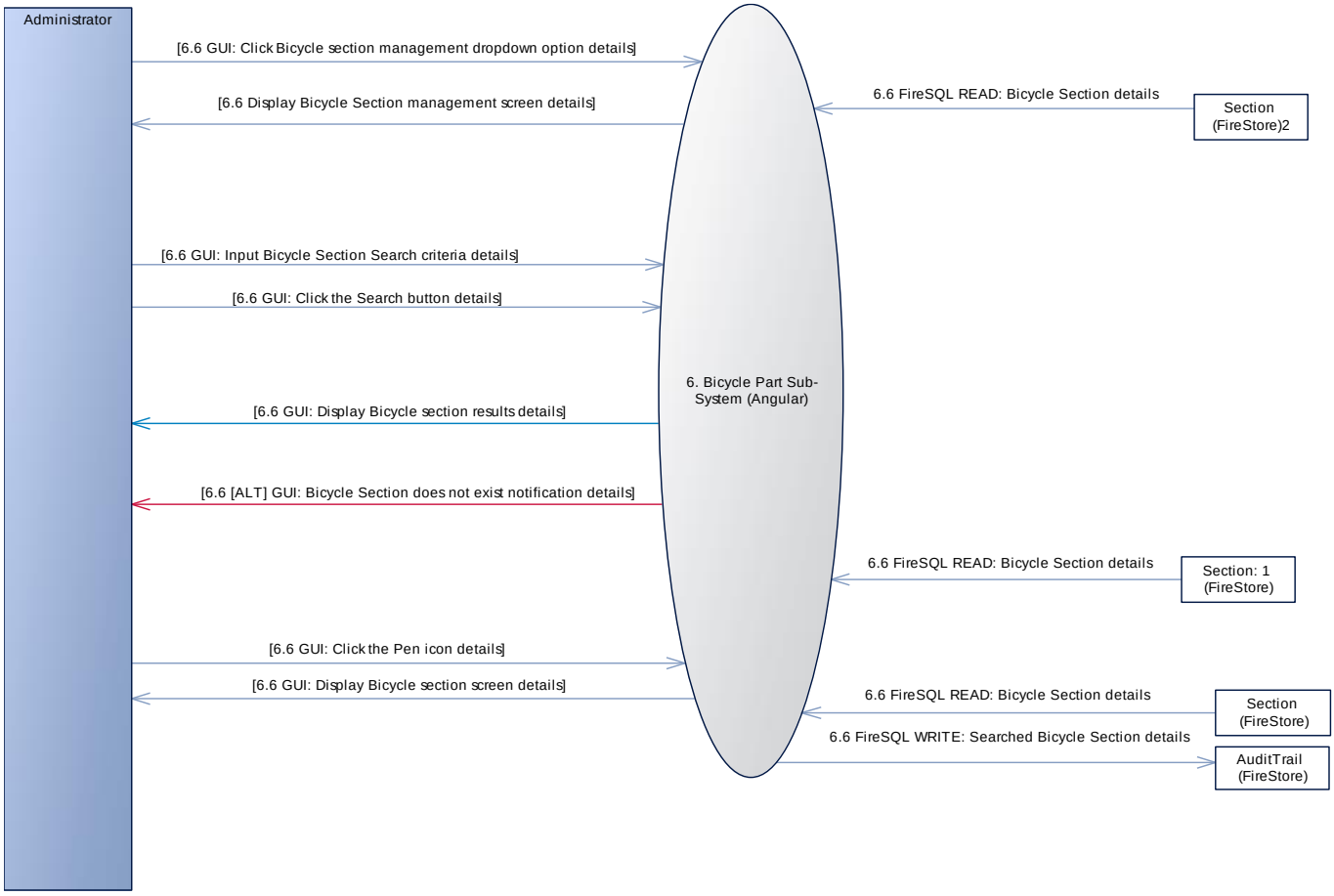


Figure 63 High-Level Diagram Sub-System 6: Part 6

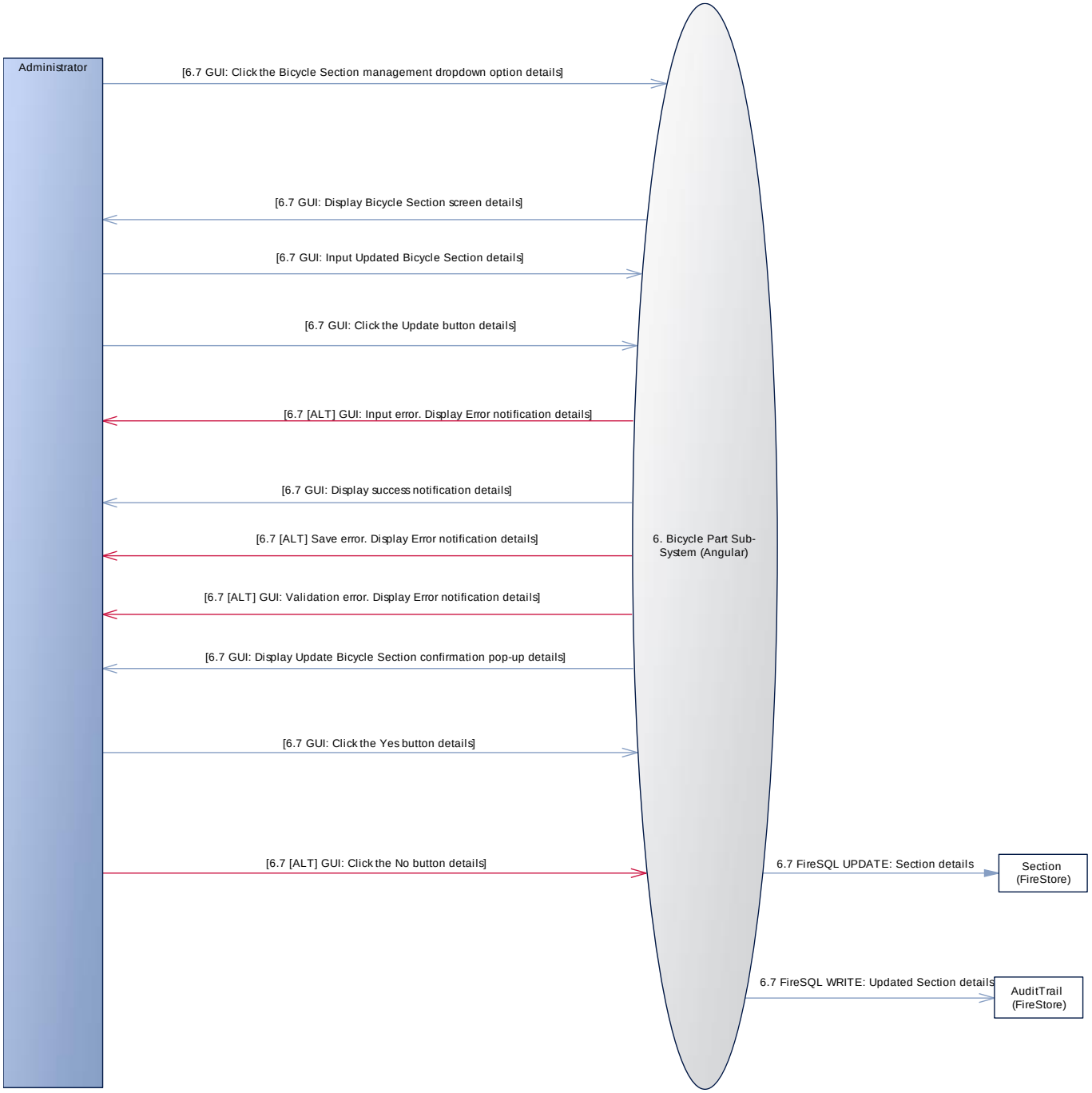


Figure 64 High-Level Diagram Sub-System 6: Part 7

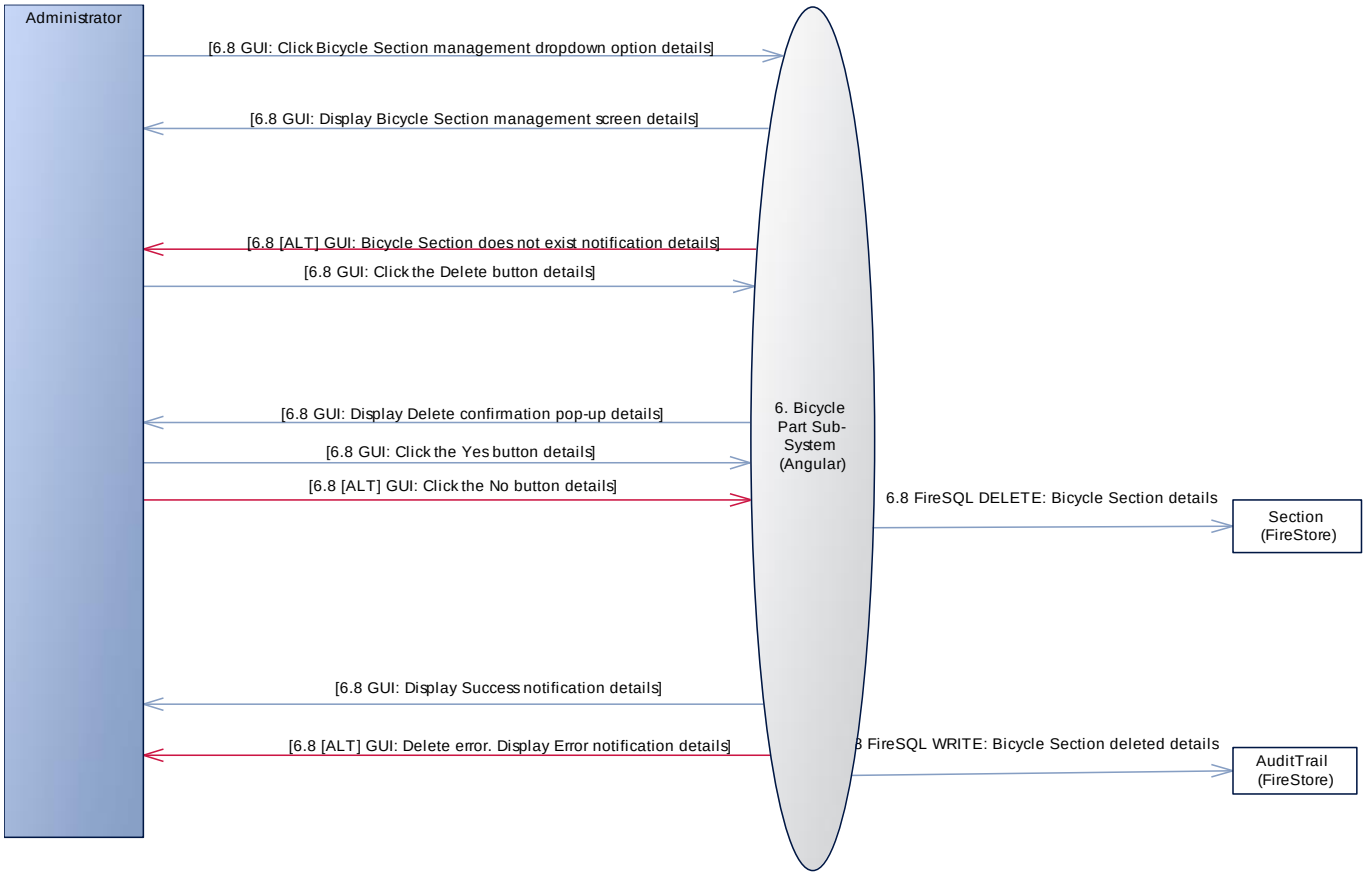


Figure 65 High-Level Diagram Sub-System 6: Part 8

Sub-System 7: Tracking

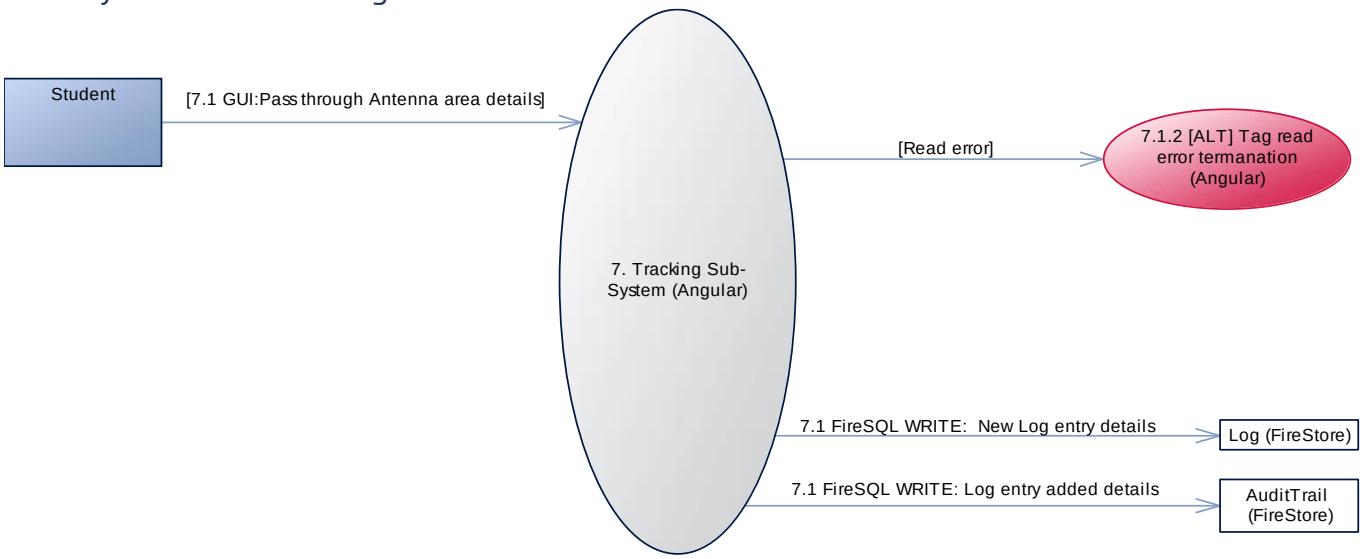


Figure 66 High-Level Diagram Sub-System 7: Part 1

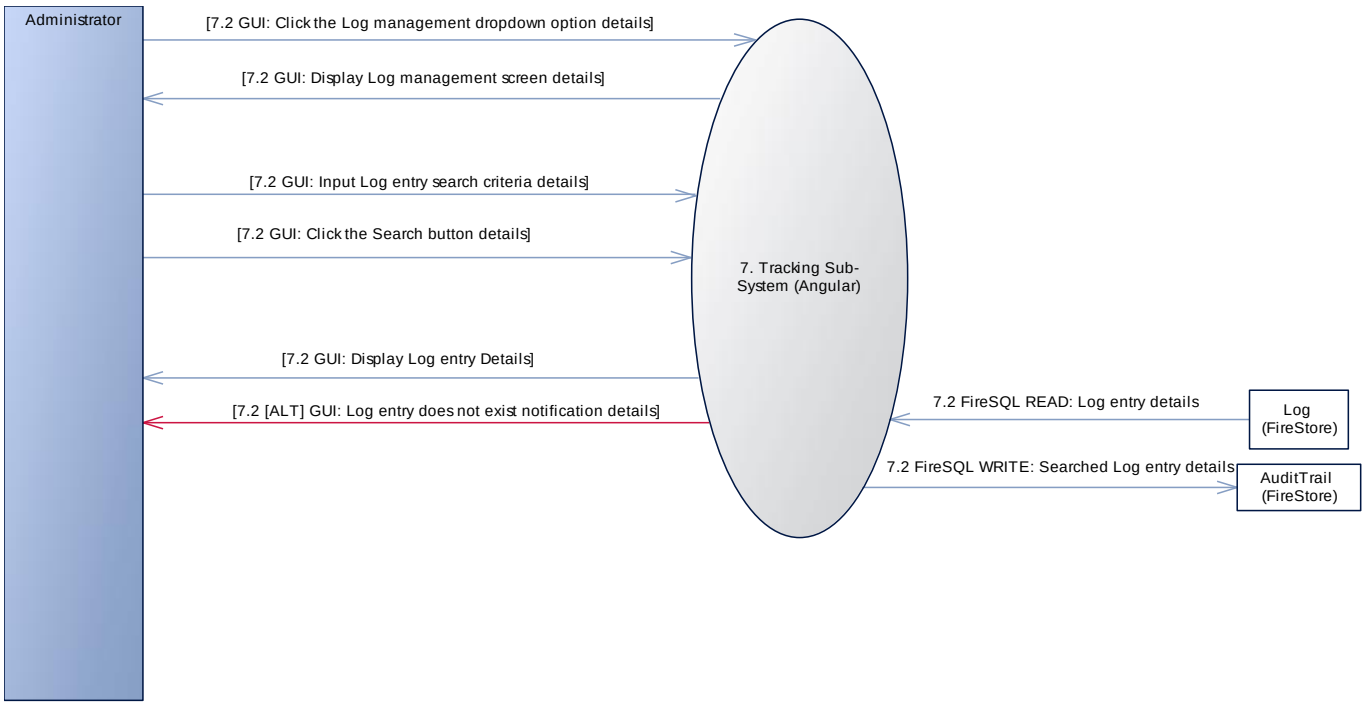


Figure 67 High-Level Diagram Sub-System 7: Part 2

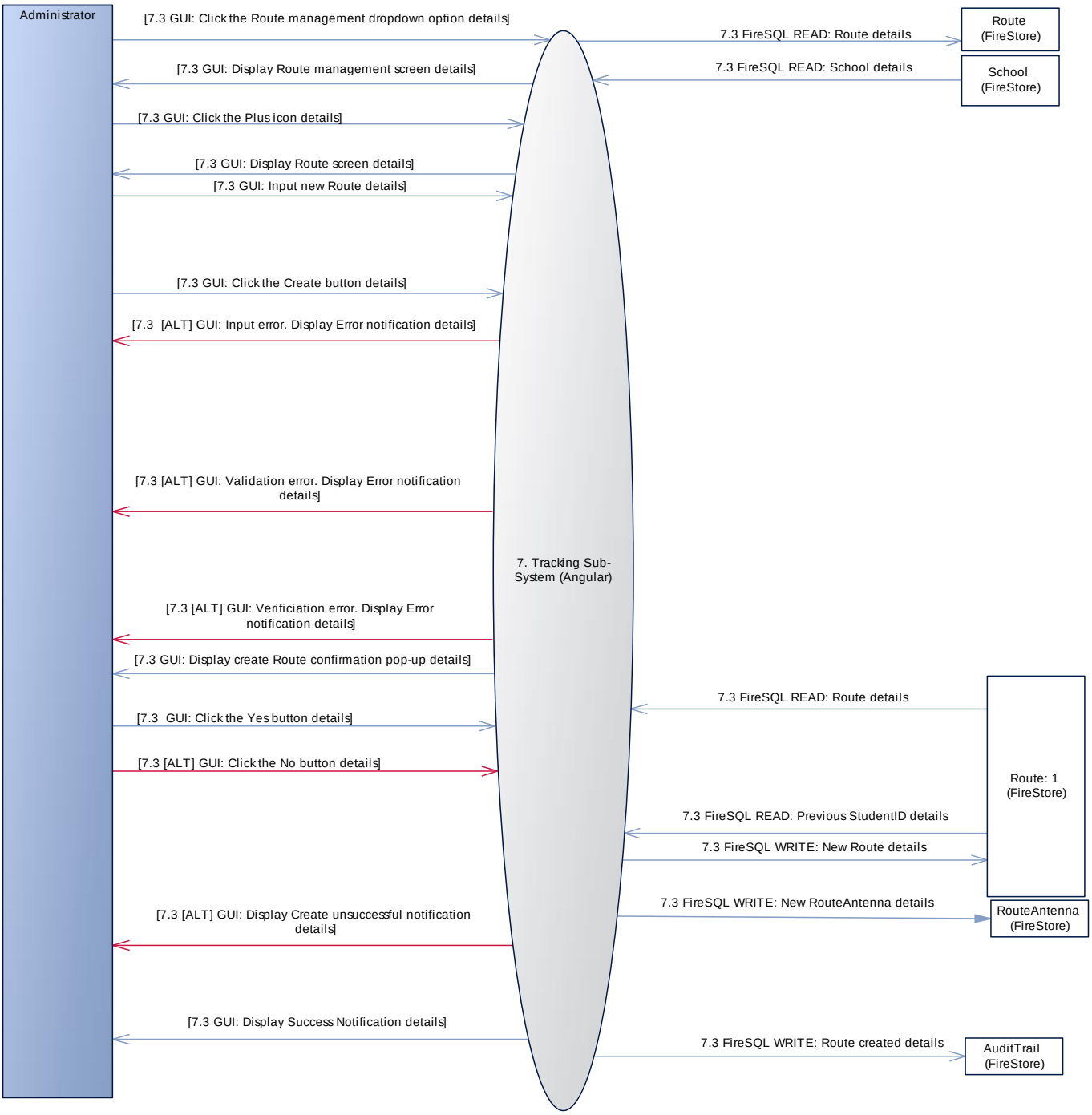


Figure 68 High-Level Diagram Sub-System 7: Part 3

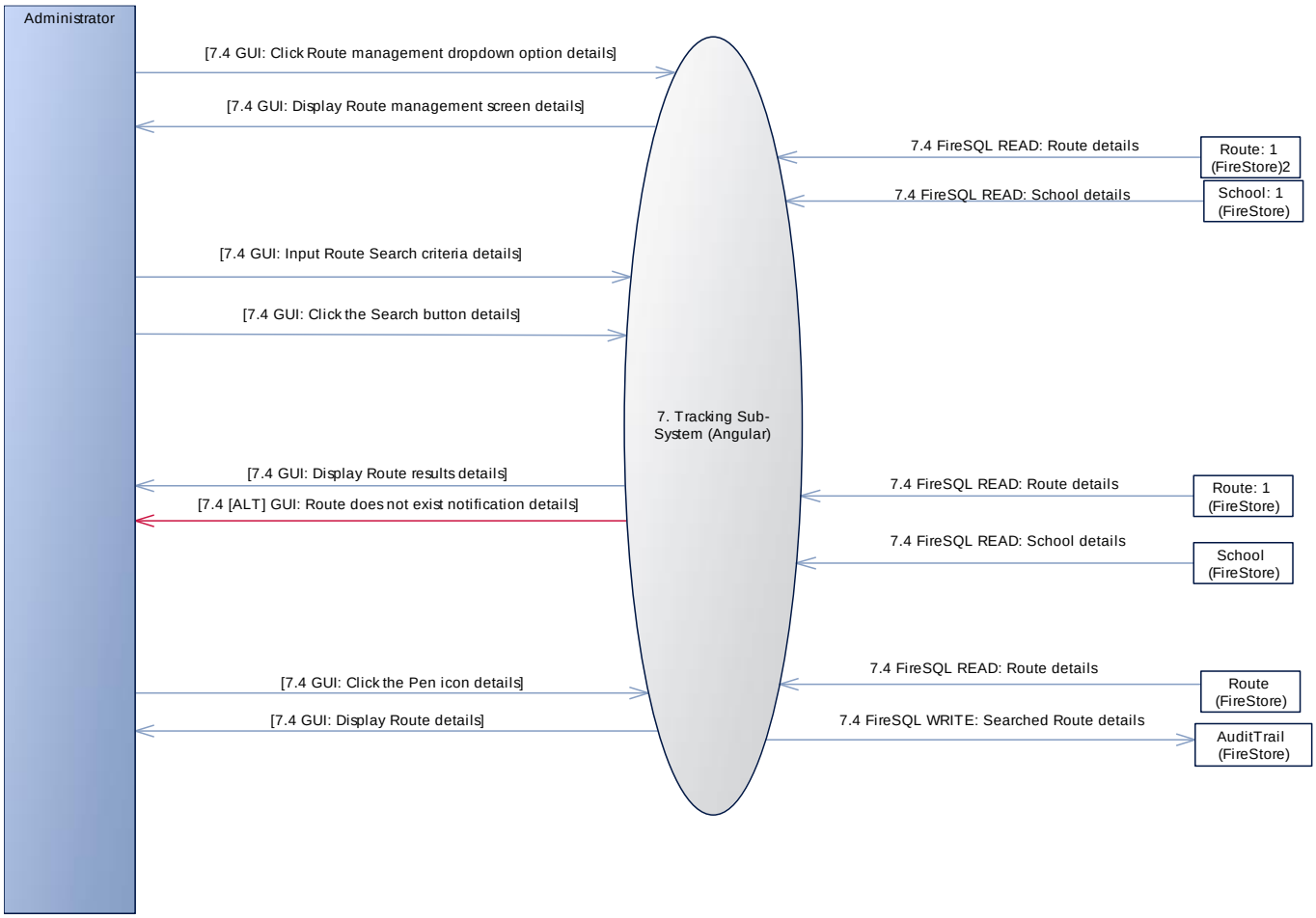


Figure 69 High-Level Diagram Sub-System 7: Part 4

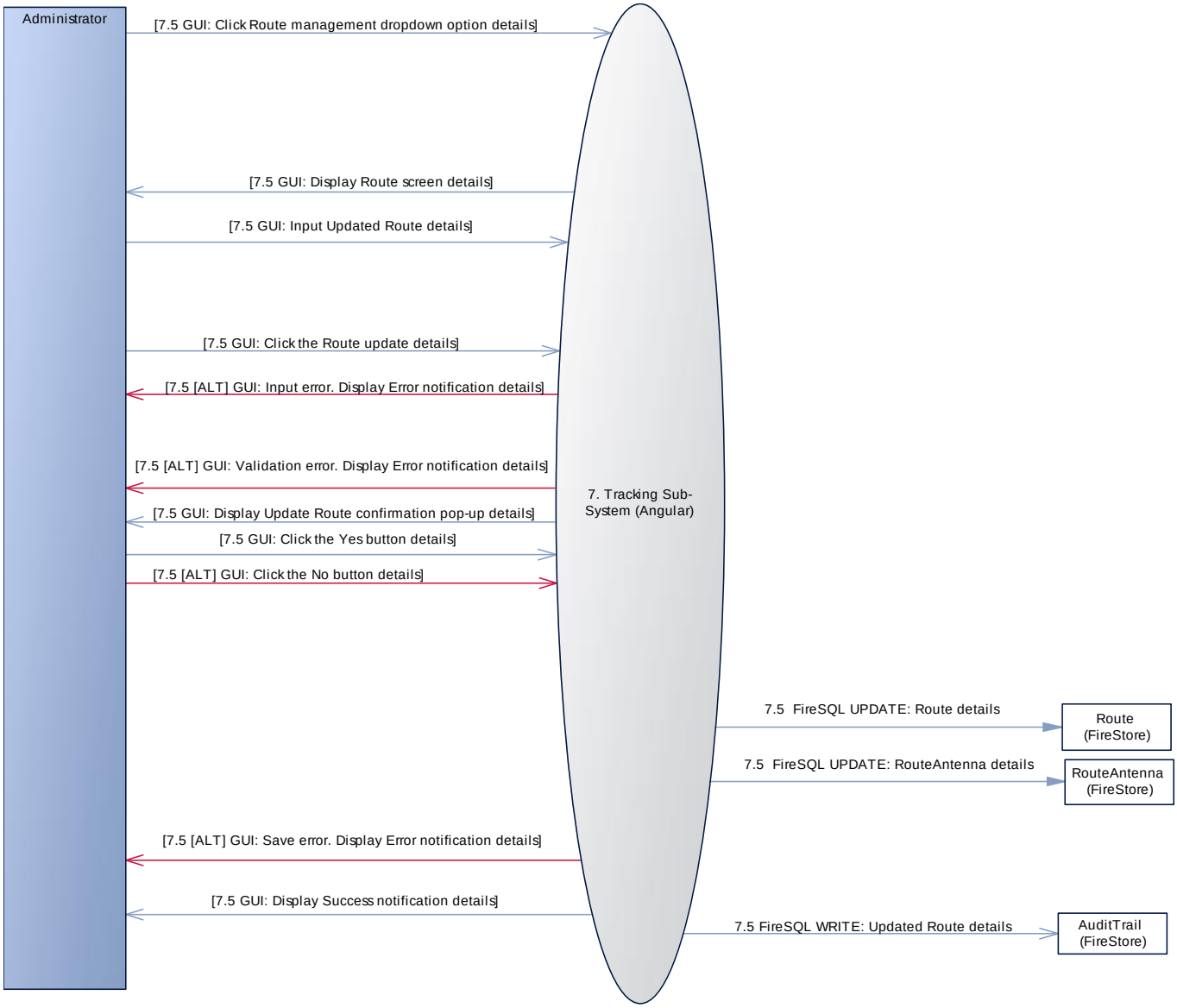


Figure 70 High-Level Diagram Sub-System 7: Part 5

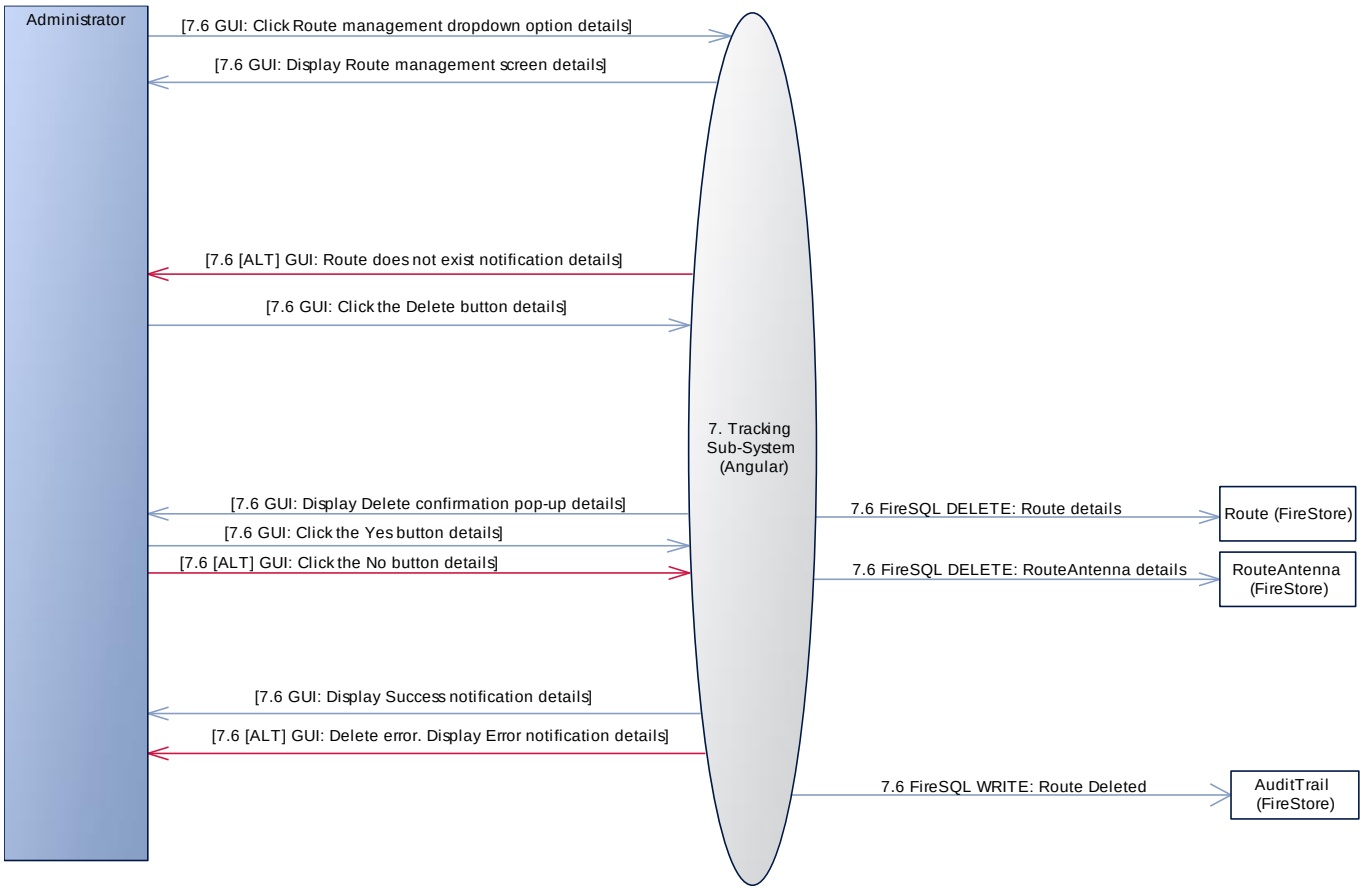


Figure 71 High-Level Diagram Sub-System 7: Part 6

Sub-System 8: Job Tasks

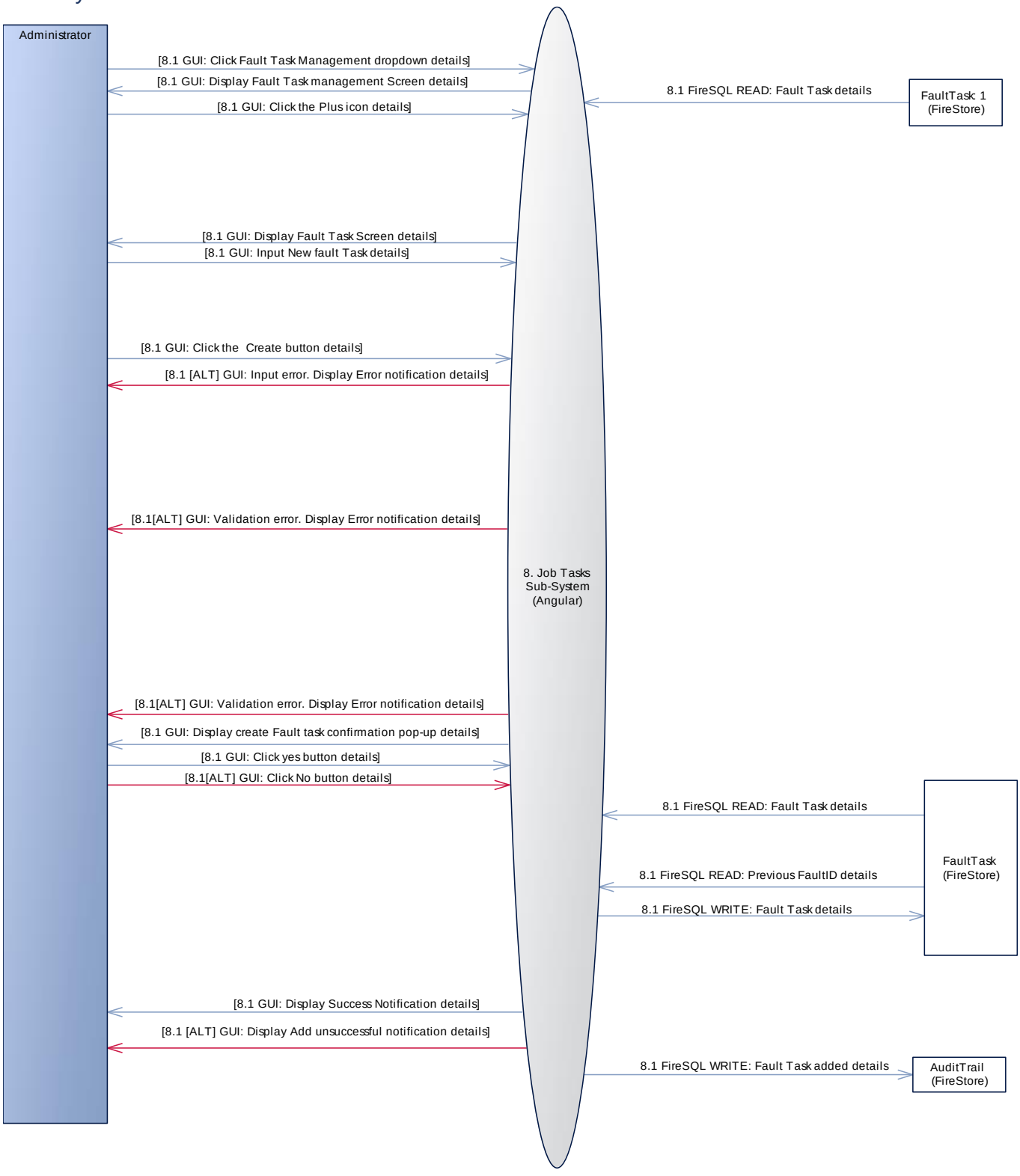


Figure 72 High-Level Diagram Sub-System 8: Part 1

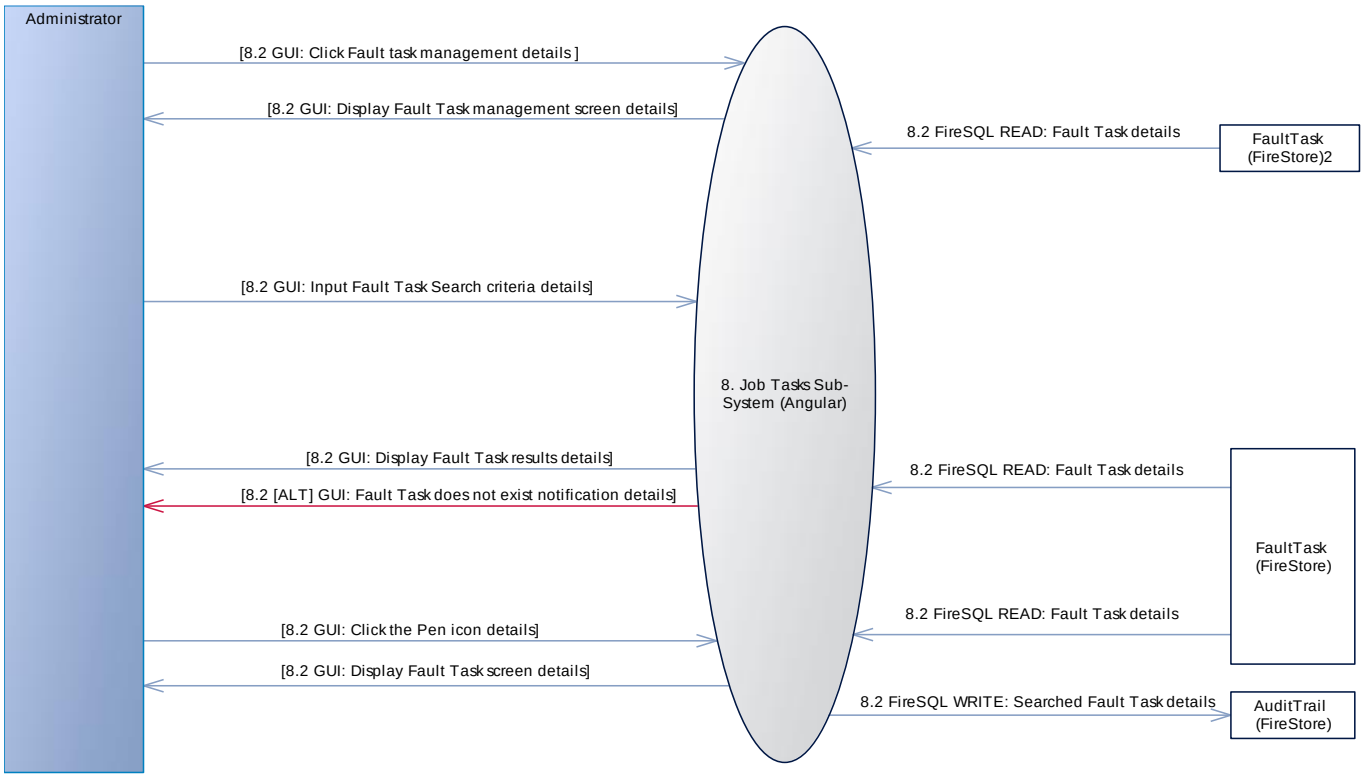


Figure 73 High-Level Diagram Sub-System 8: Part 2

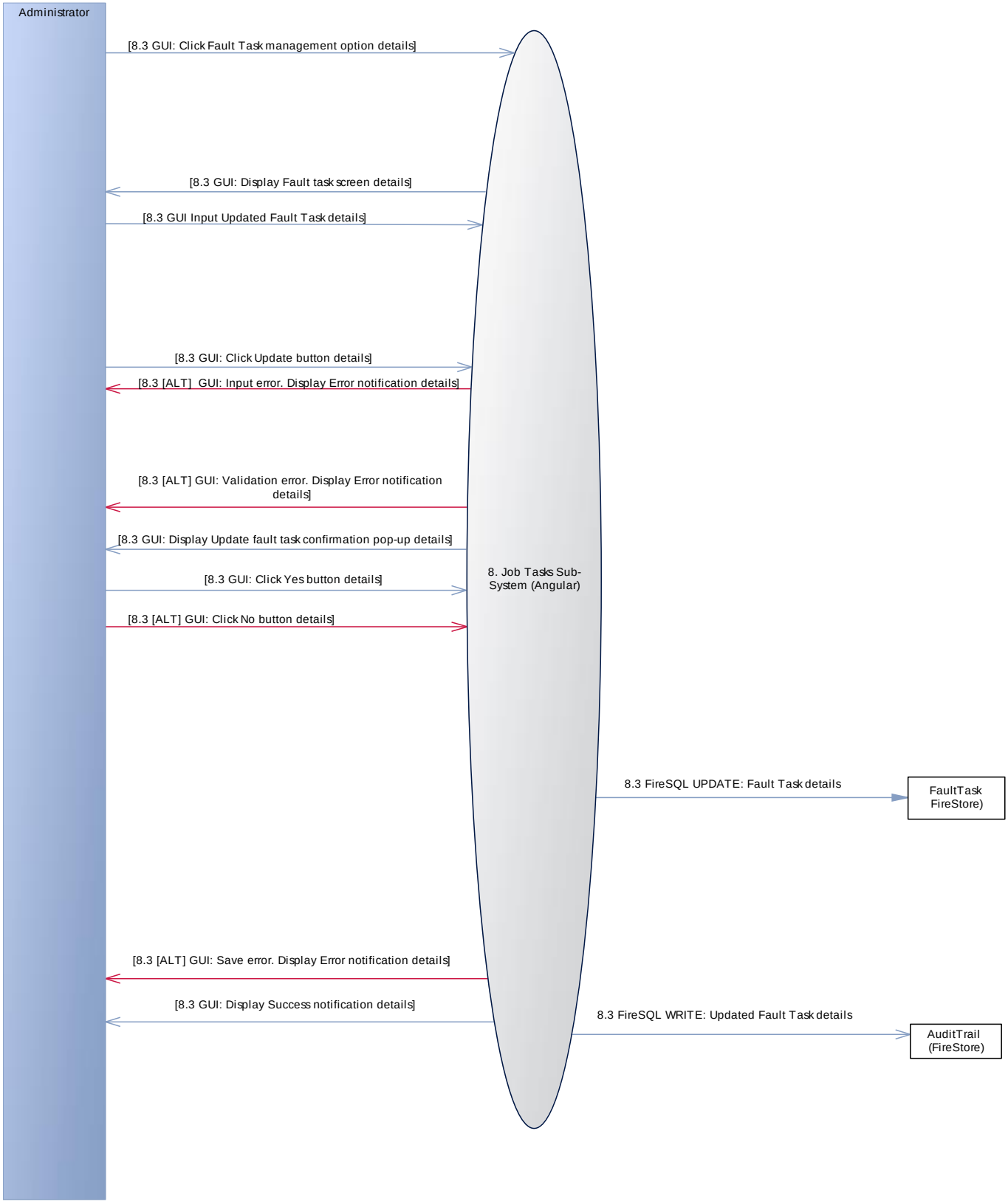


Figure 74 High-Level Diagram Sub-System 8: Part 3

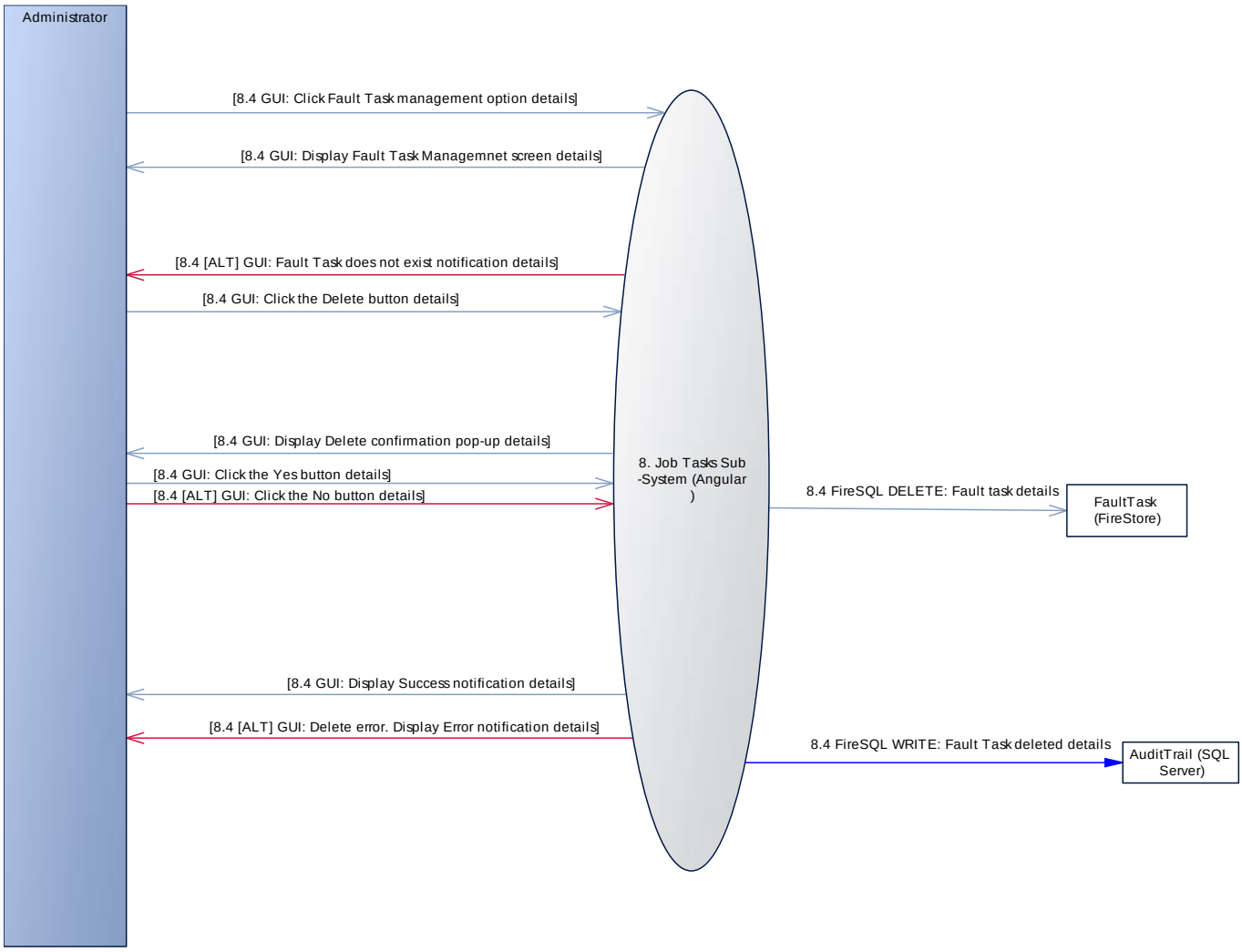


Figure 75 High-Level Diagram Sub-System 8: Part 4

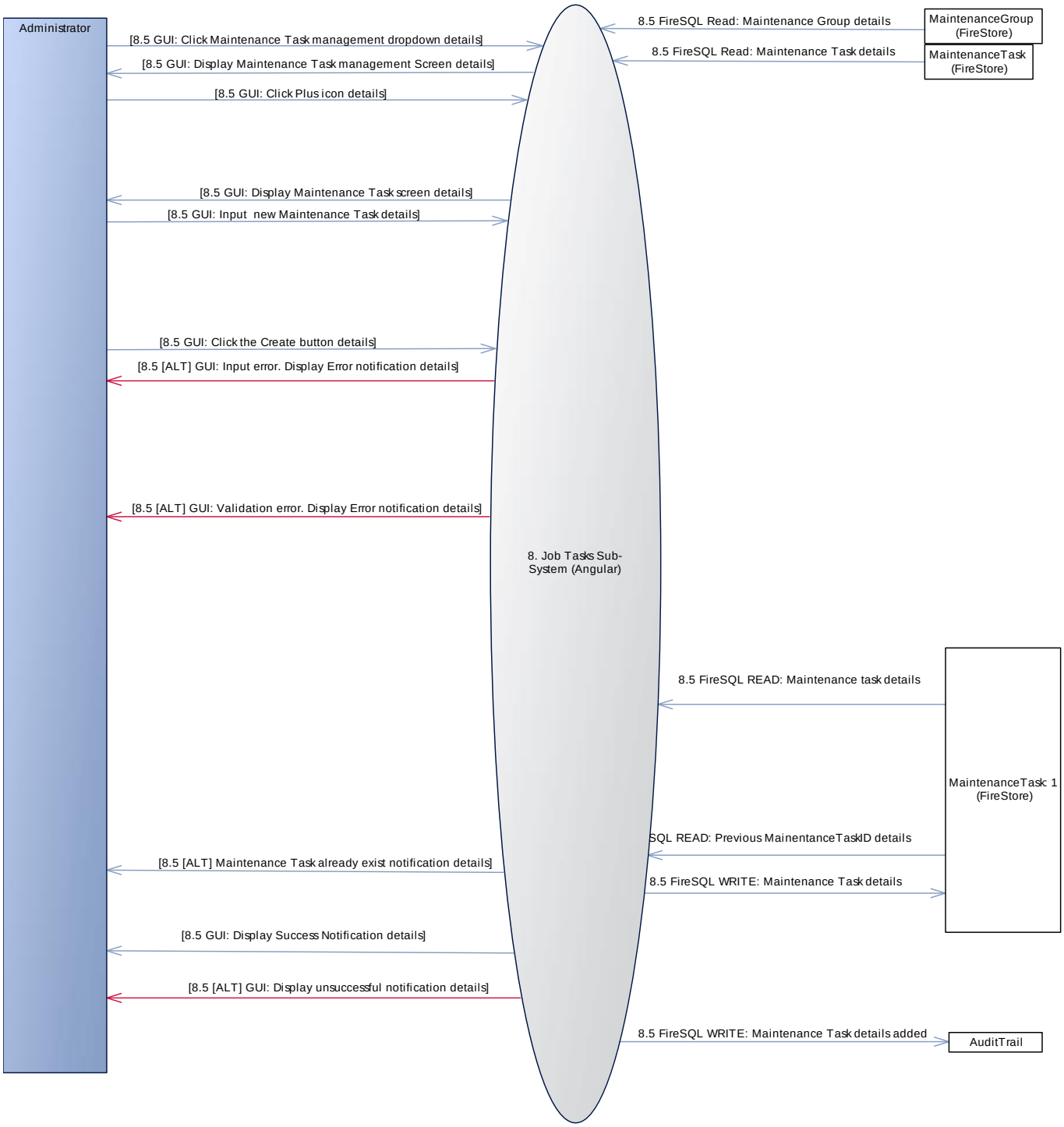


Figure 76 High-Level Diagram Sub-System 8: Part 5

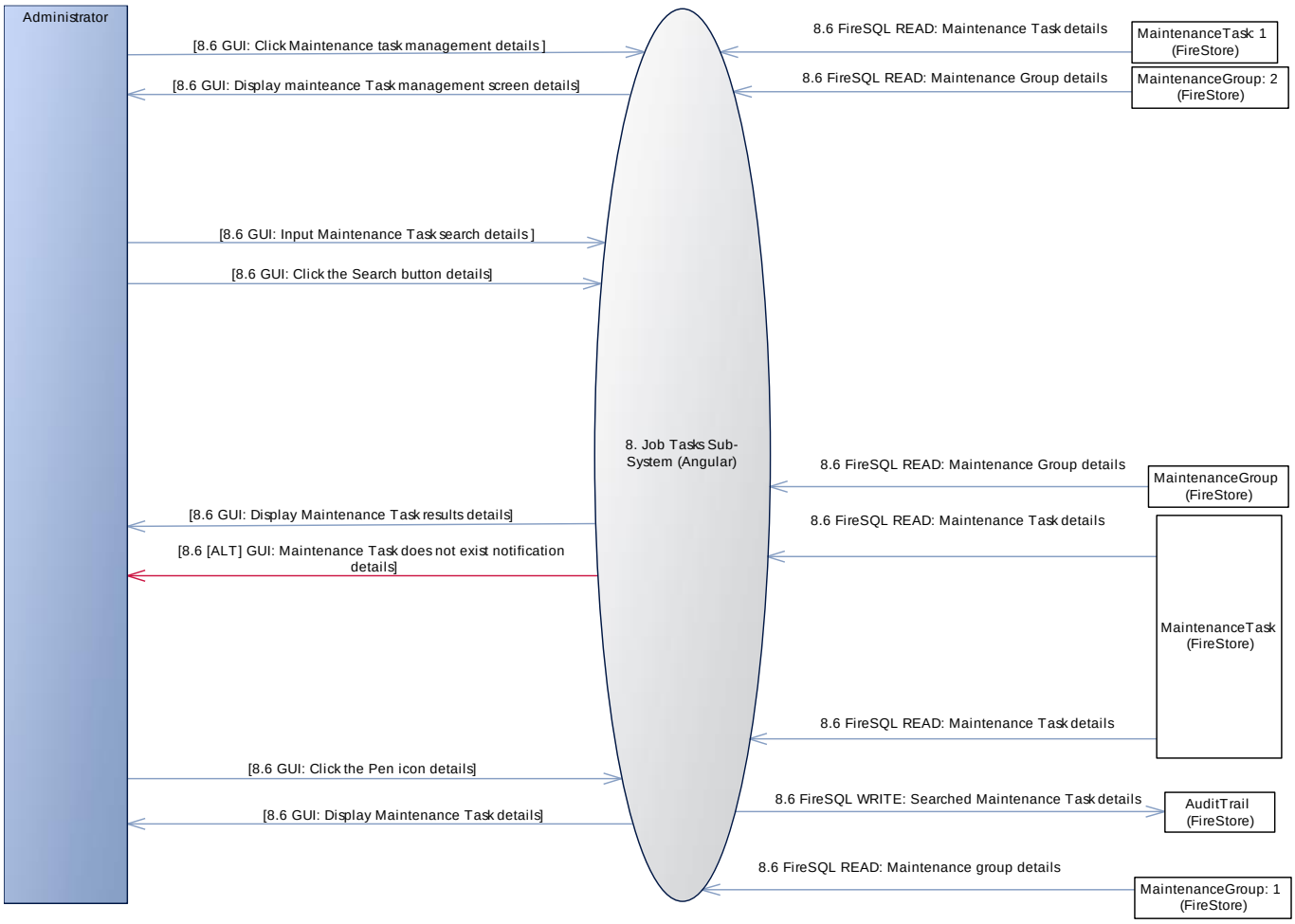


Figure 77 High-Level Diagram Sub-System 8: Part 6

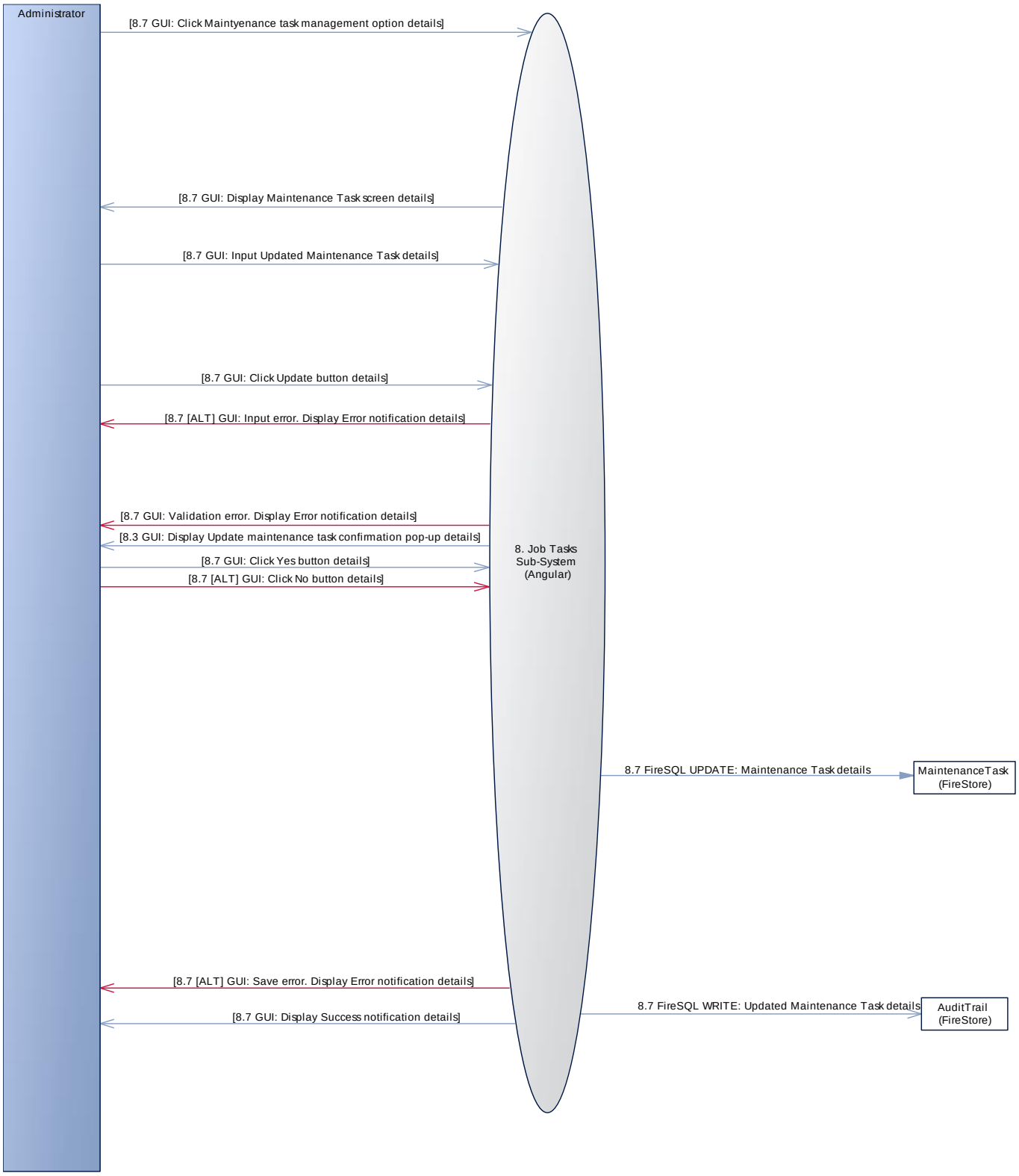


Figure 78 High-Level Diagram Sub-System 8: Part 7

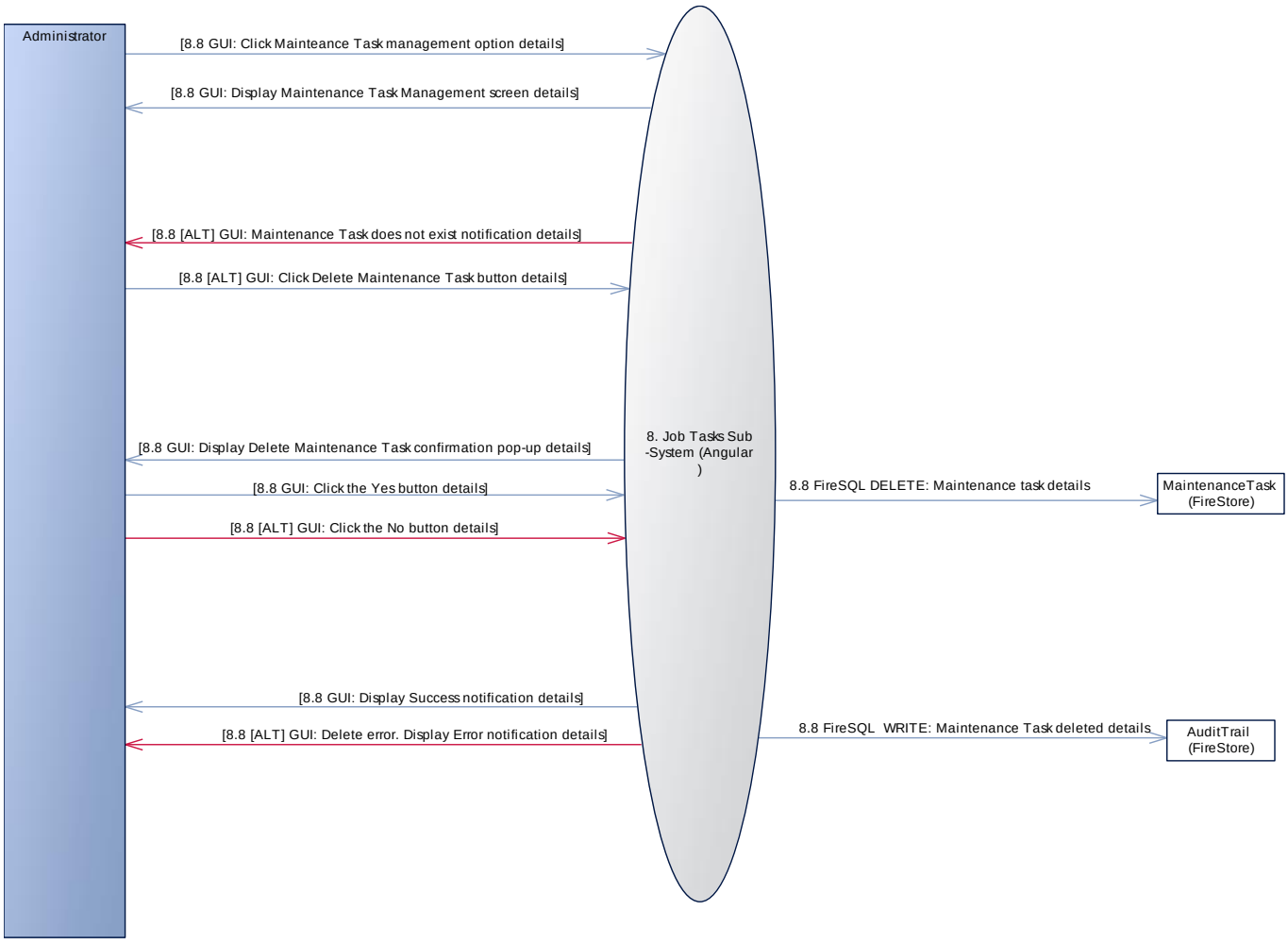


Figure 79 High-Level Diagram Sub-System 8: Part 8

Sub-System 9: School

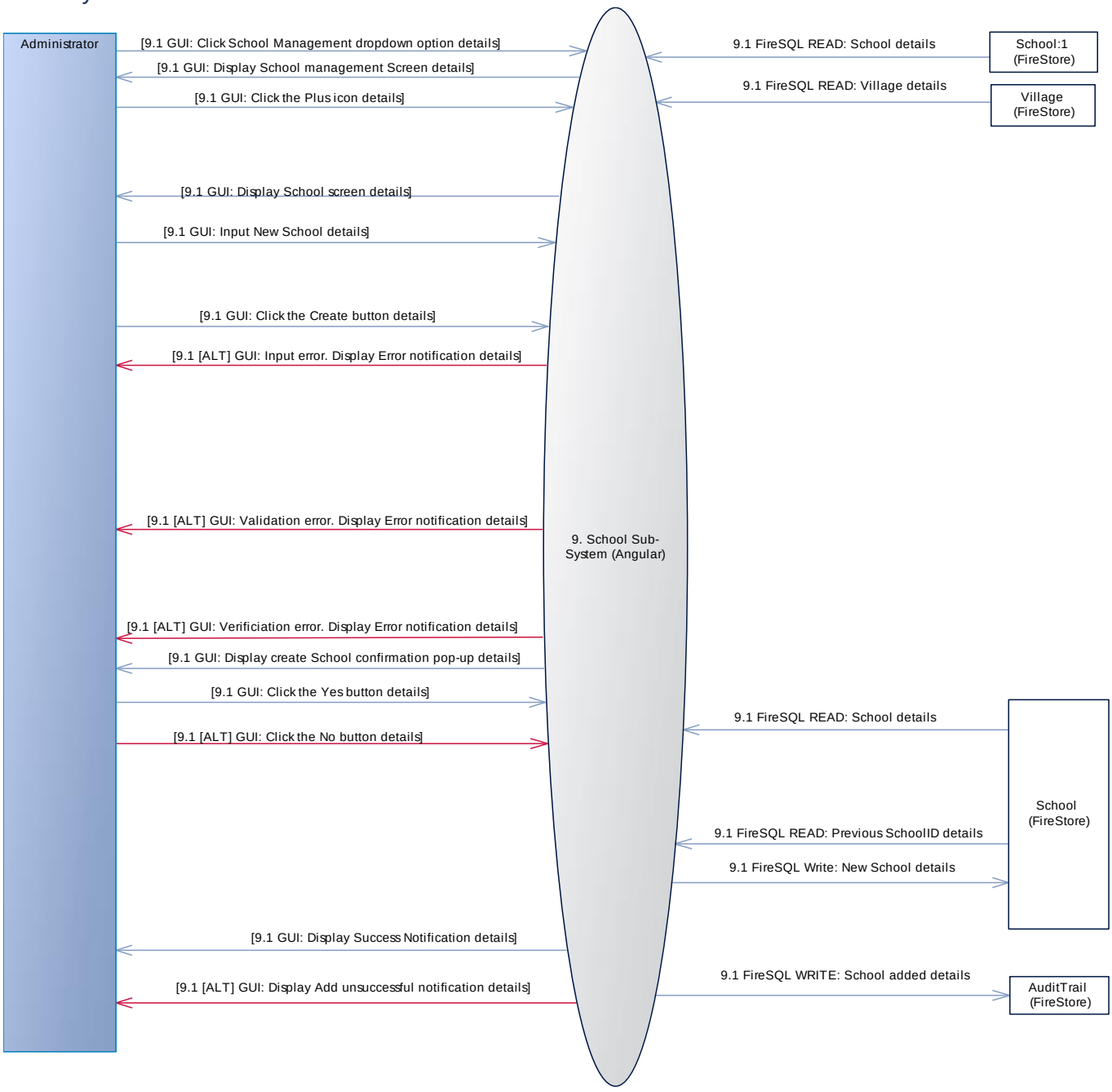


Figure 80 High-Level Diagram Sub-System 9: Part 1

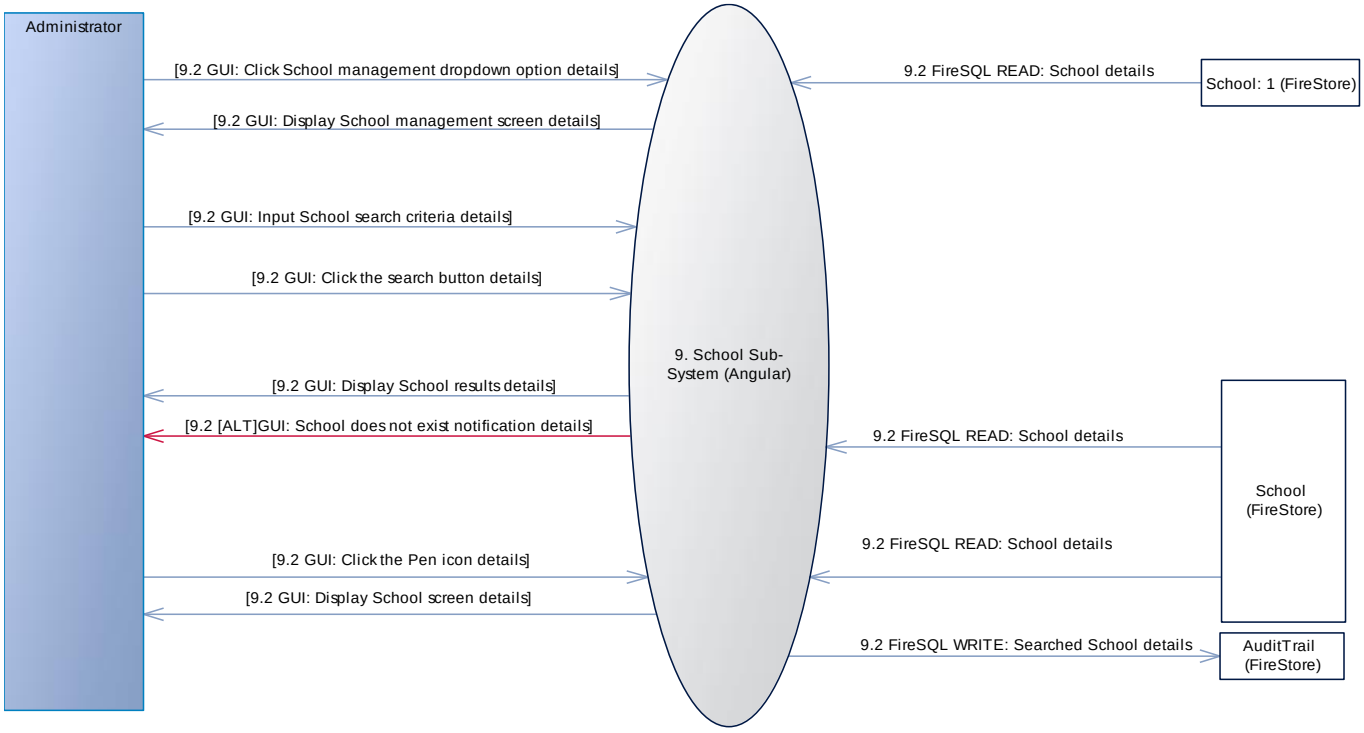


Figure 81 High-Level Diagram Sub-System 9: Part 2

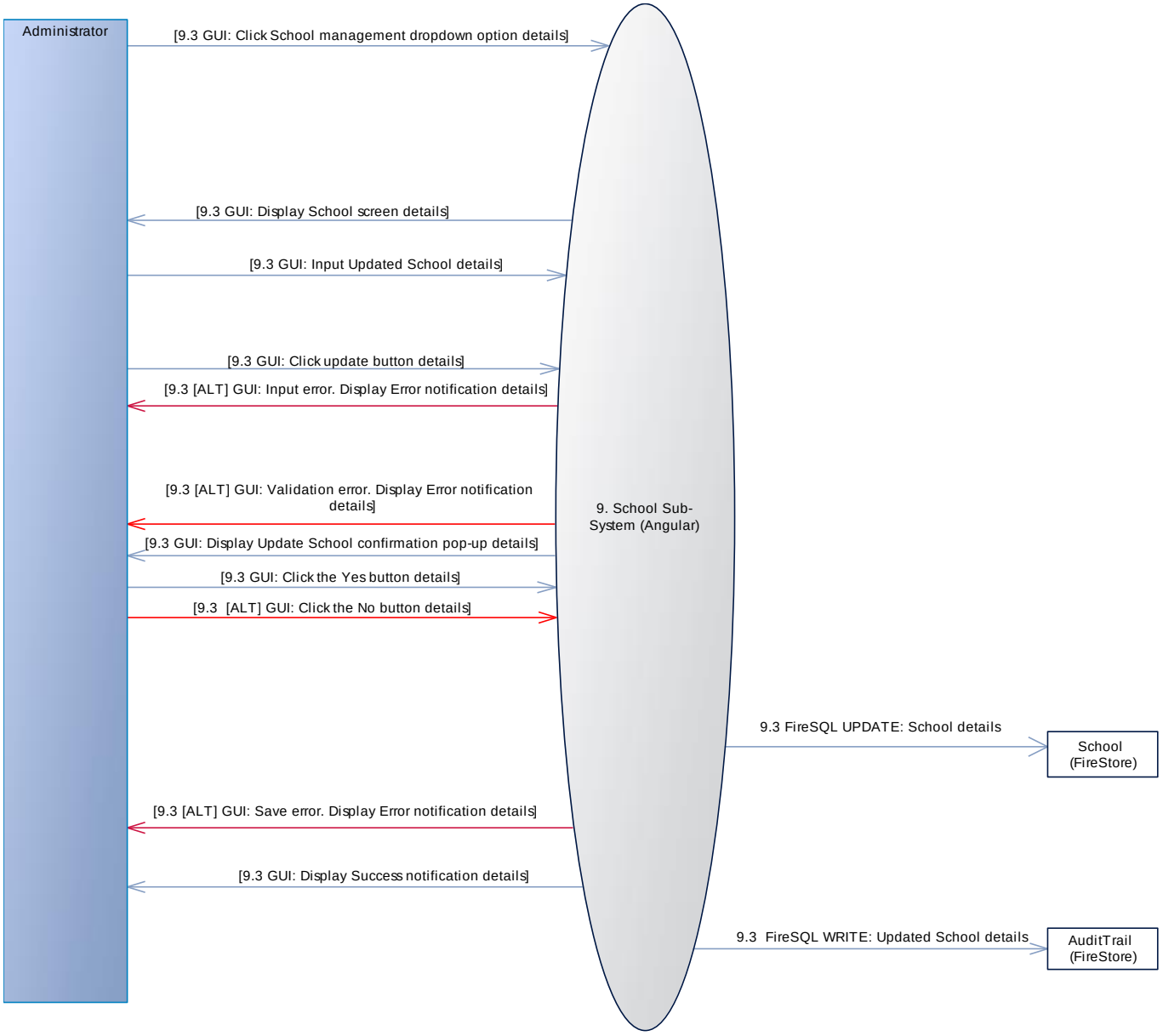


Figure 82 High-Level Diagram Sub-System 9: Part 3

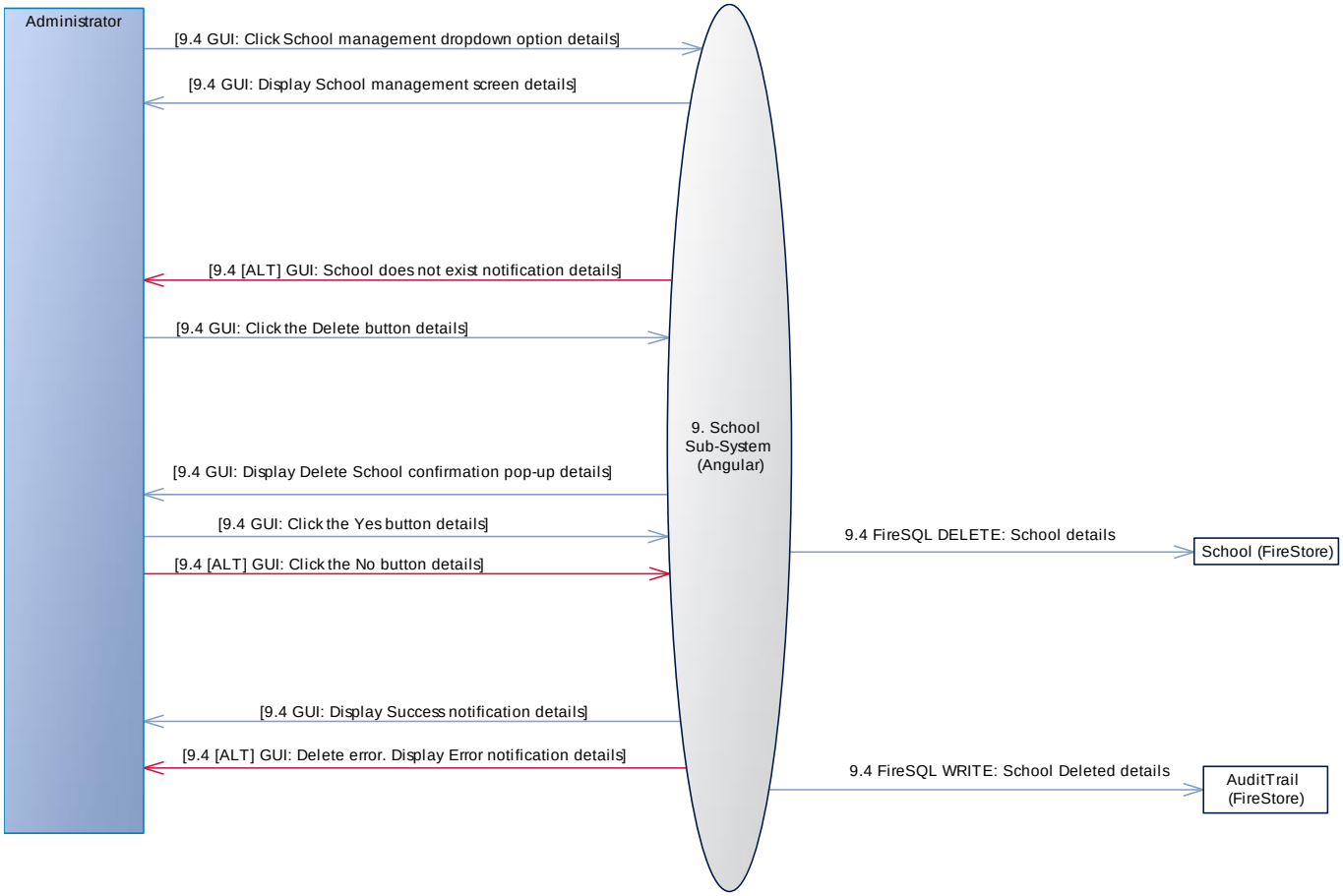


Figure 83 High-Level Diagram Sub-System 9: Part 4

Sub-System 10: Job

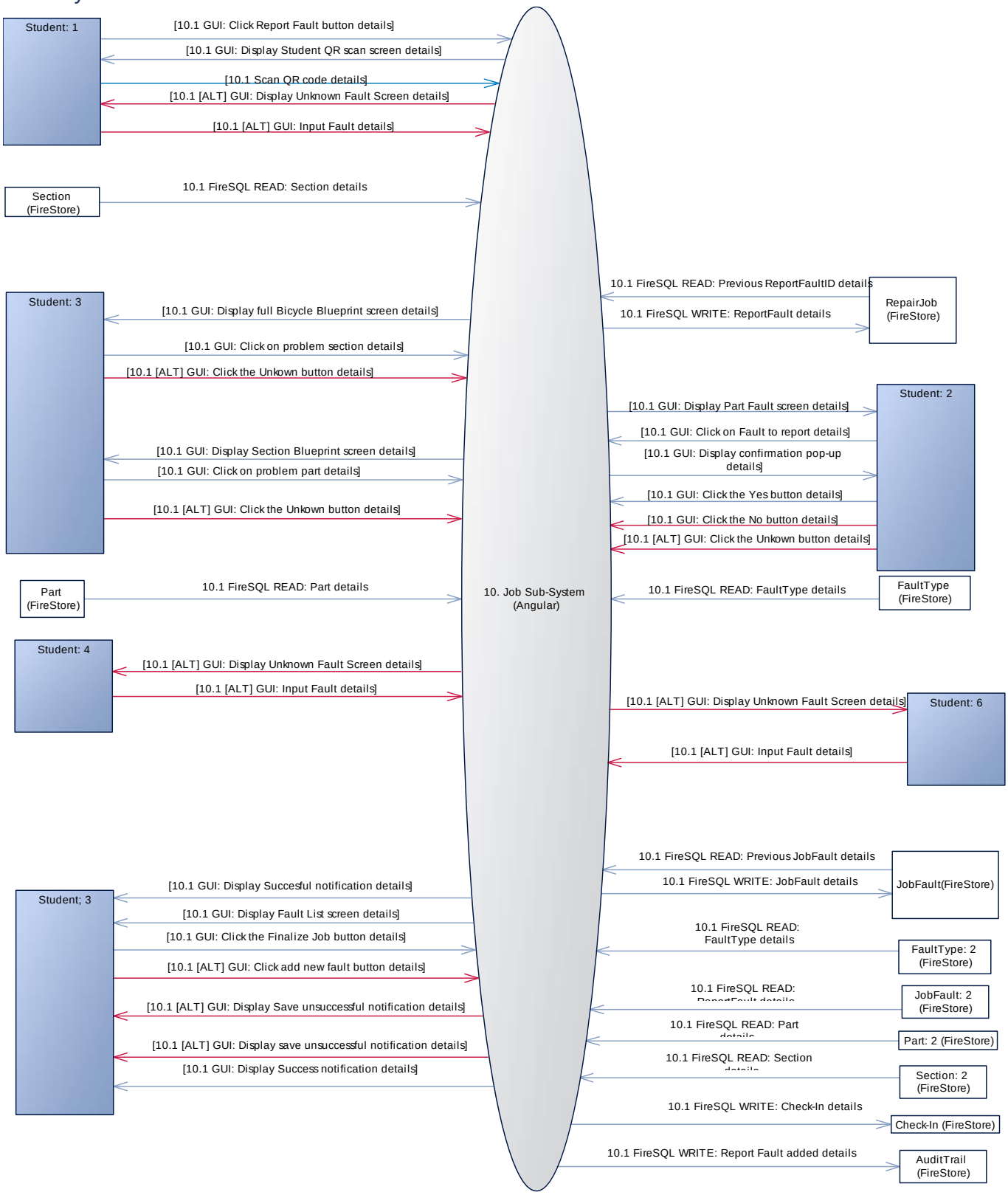


Figure 84 High-Level Diagram Sub-System 10: Part 1

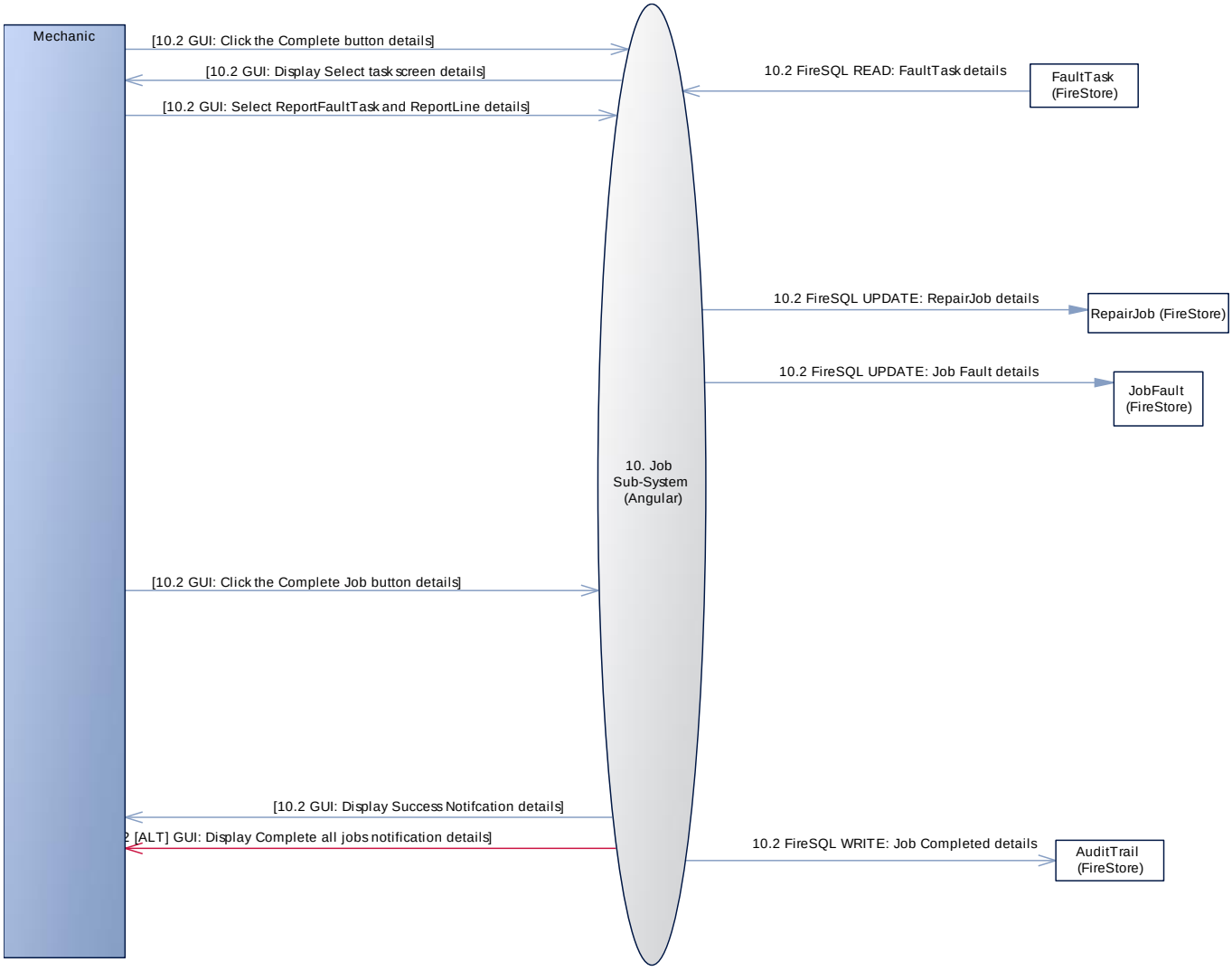


Figure 85 High-Level Diagram Sub-System 10: Part 2

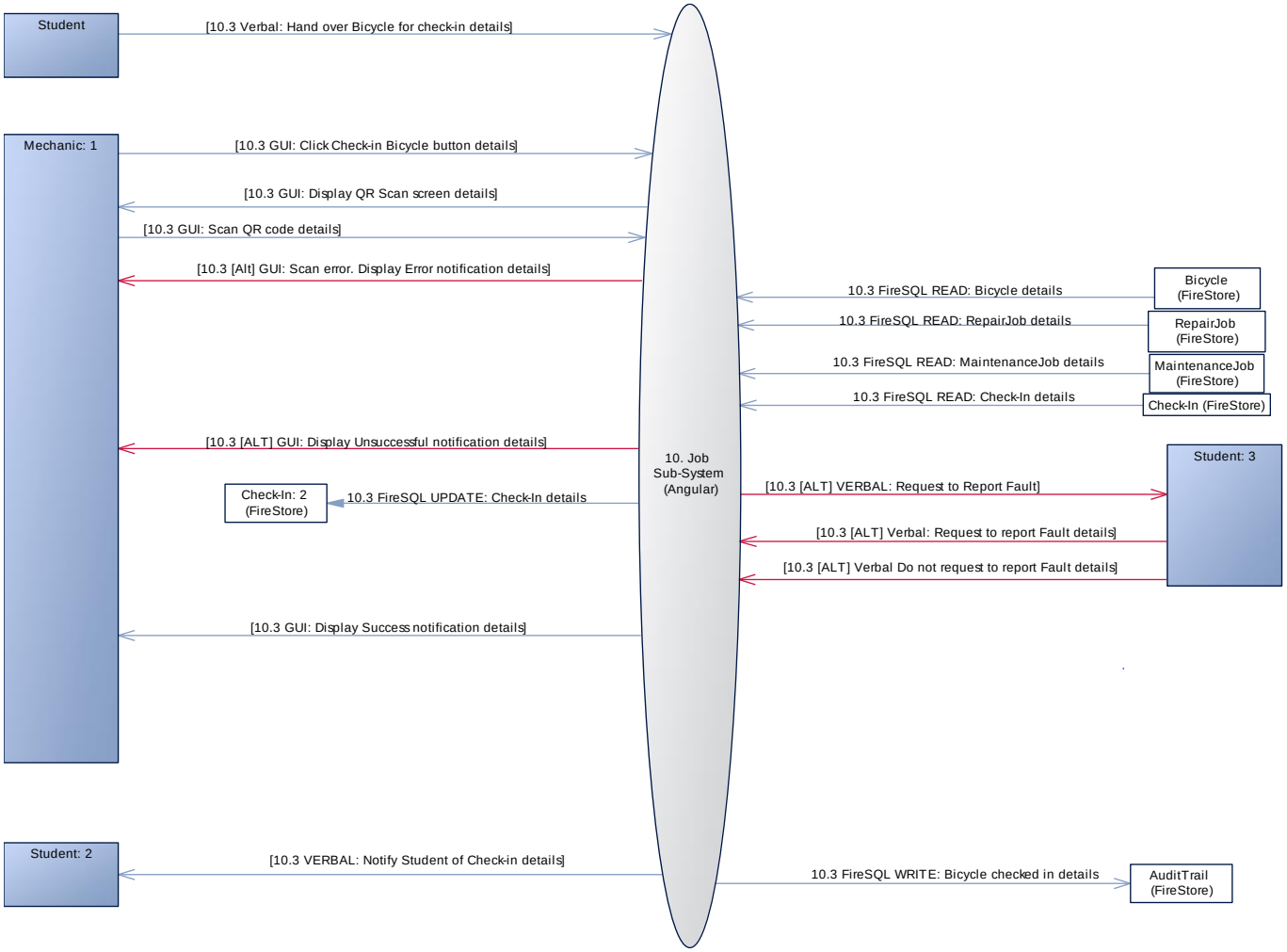


Figure 86 High-Level Diagram Sub-System 10: Part 3

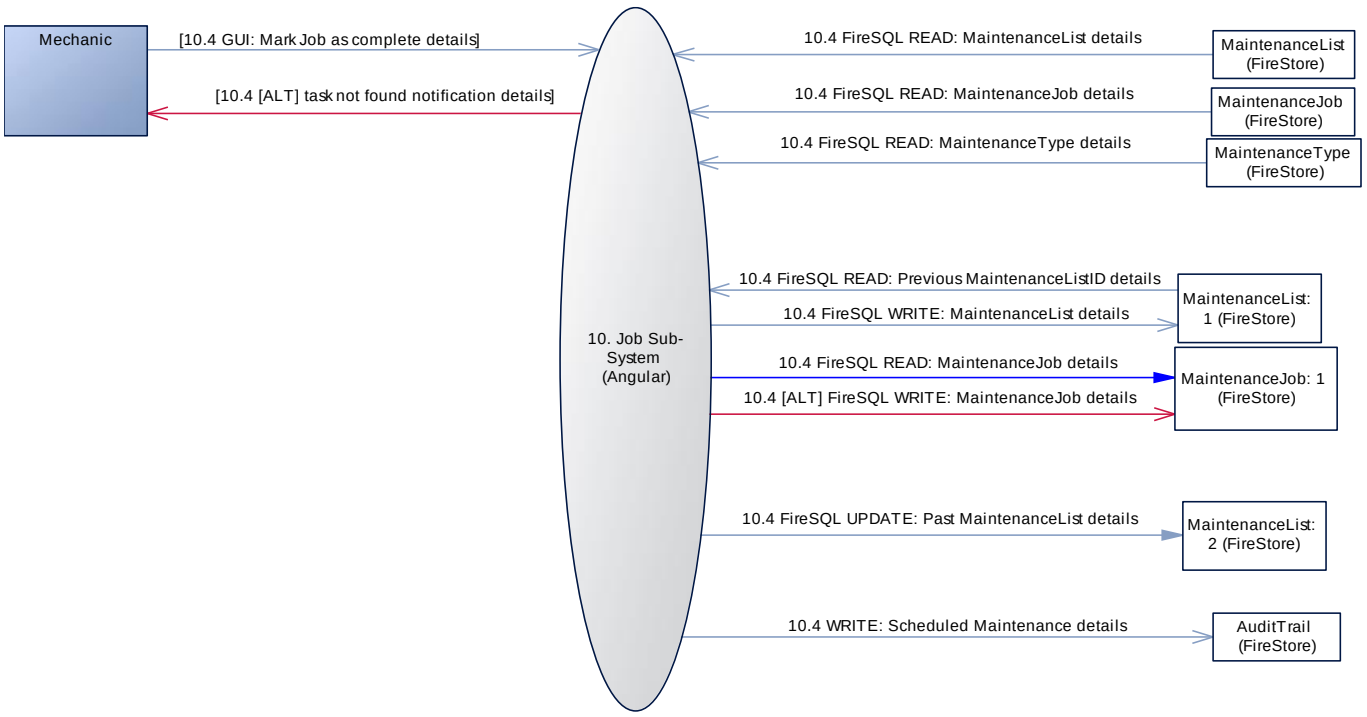


Figure 87 High-Level Diagram Sub-System 10: Part 4

Sub-System 11: Reporting

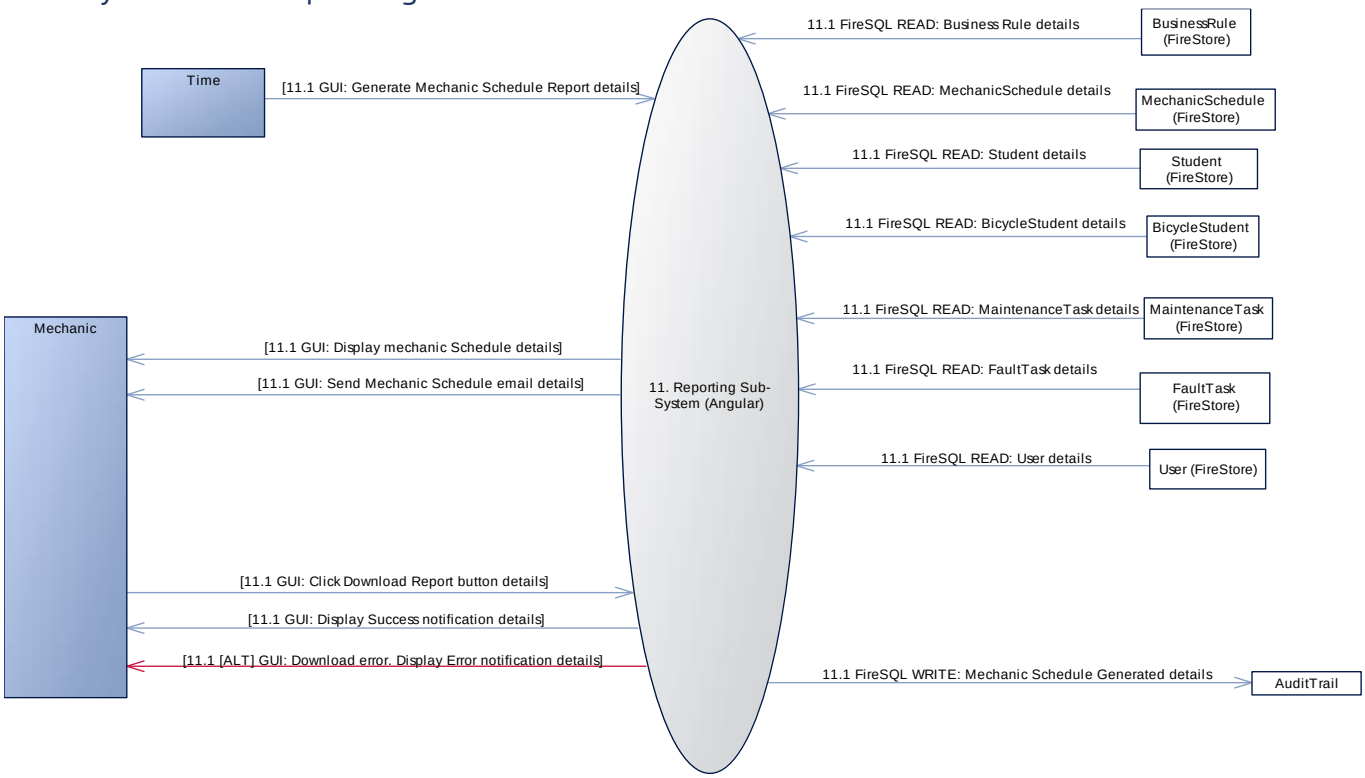


Figure 88 High-Level Diagram Sub-System 11: Part 1

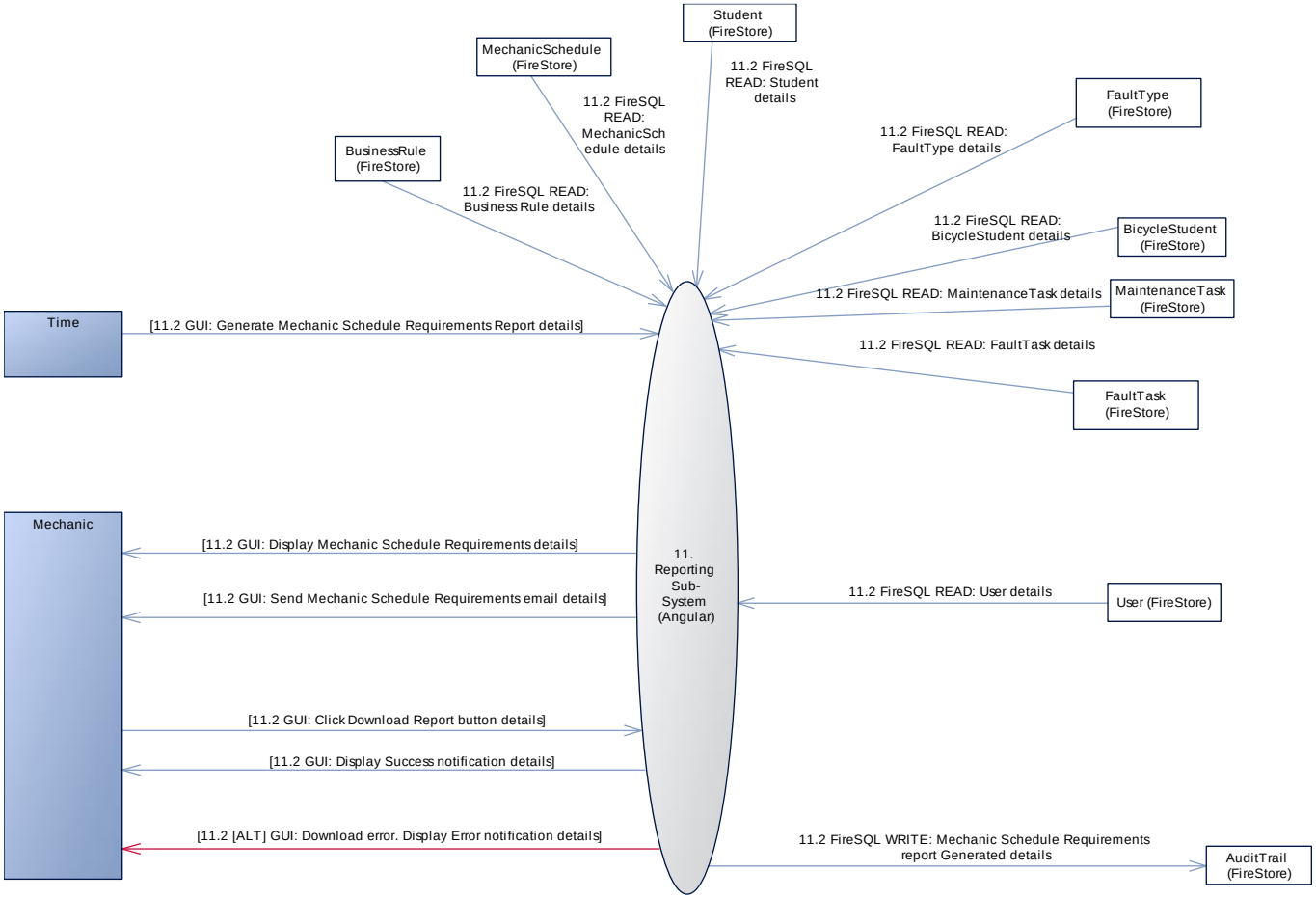


Figure 89 High-Level Diagram Sub-System 11: Part 2

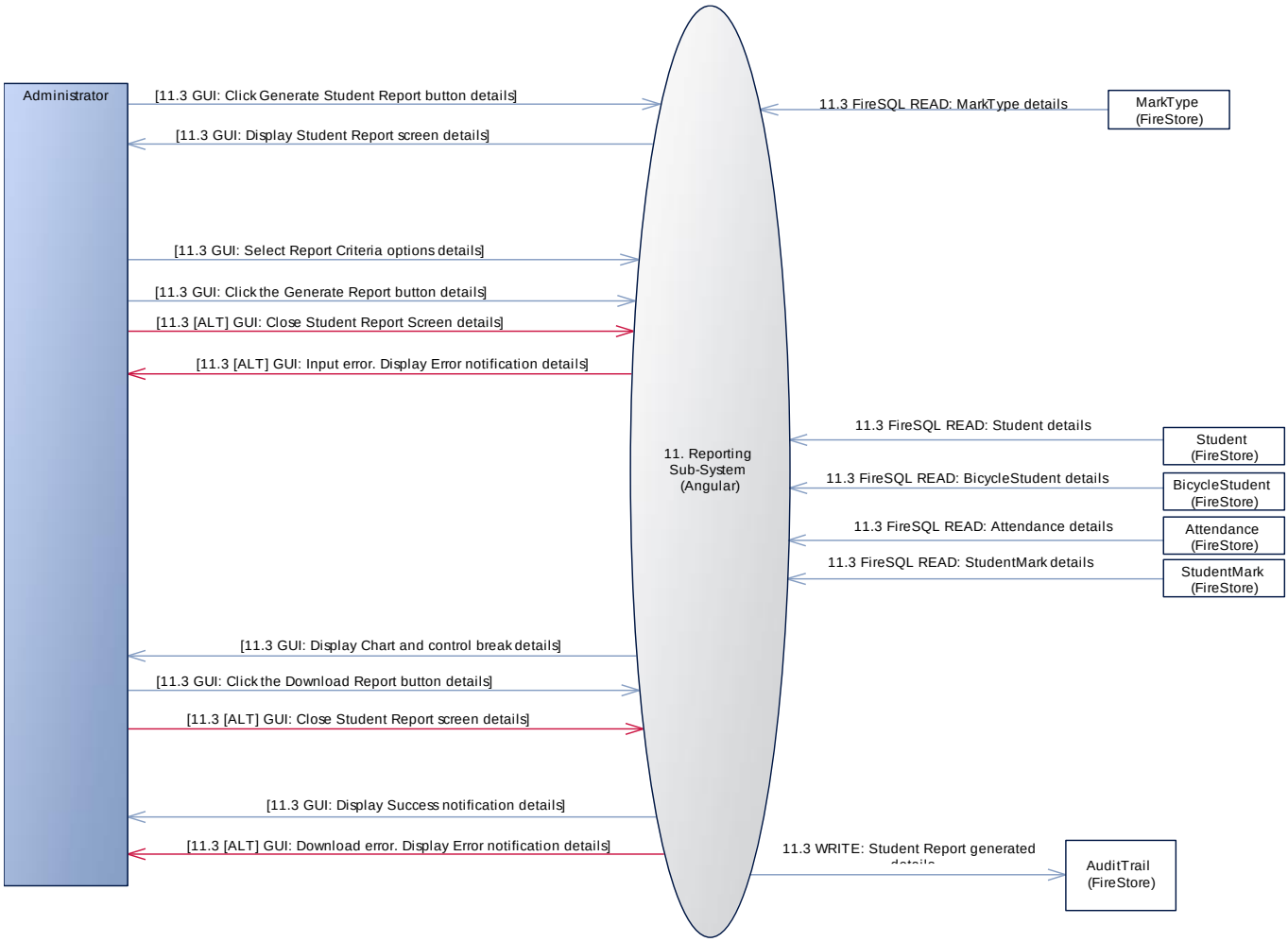


Figure 90 High-Level Diagram Sub-System 11: Part 3

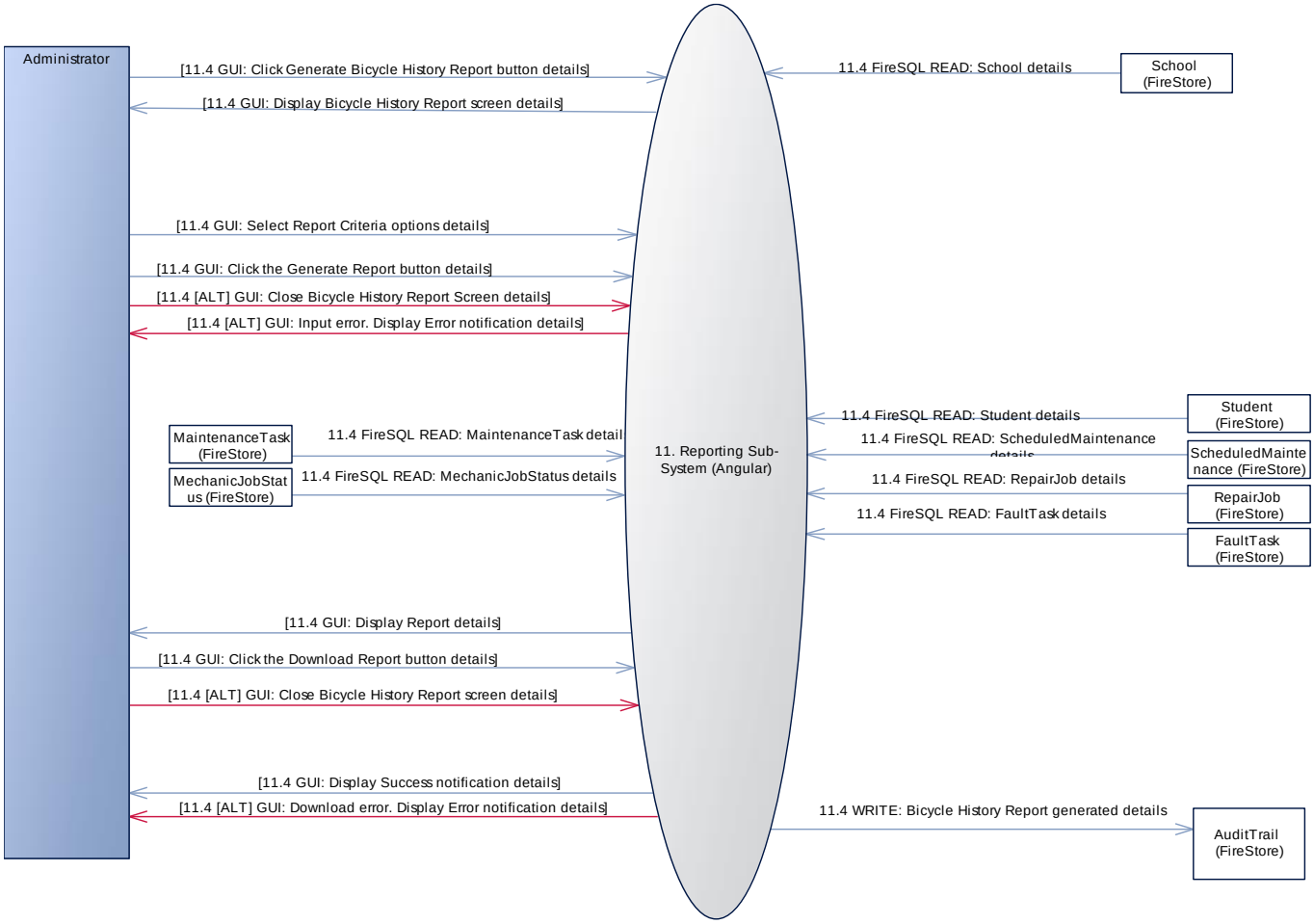


Figure 91 High-Level Diagram Sub-System 11: Part 4

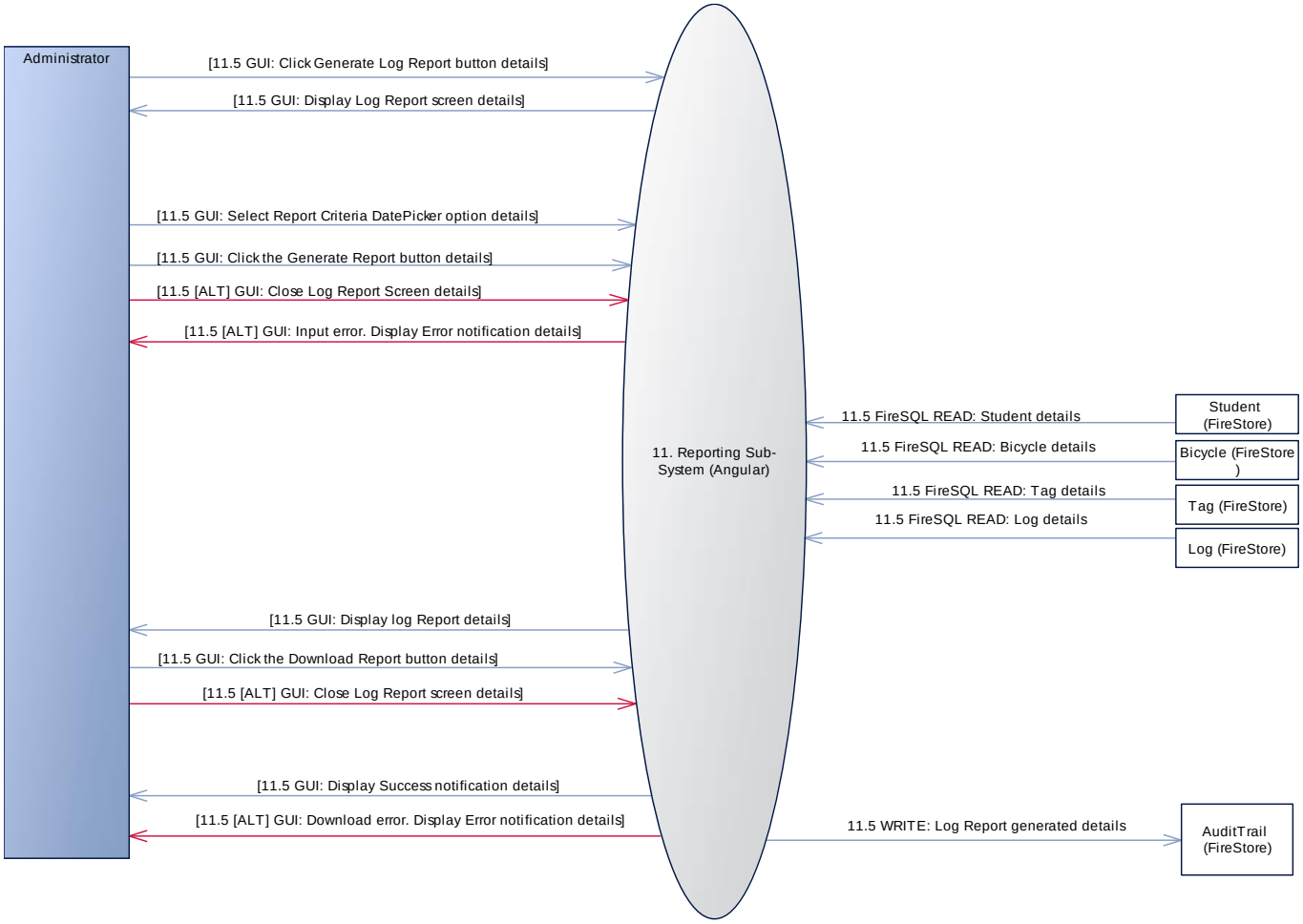


Figure 92 High-Level Diagram Sub-System 11: Part 5

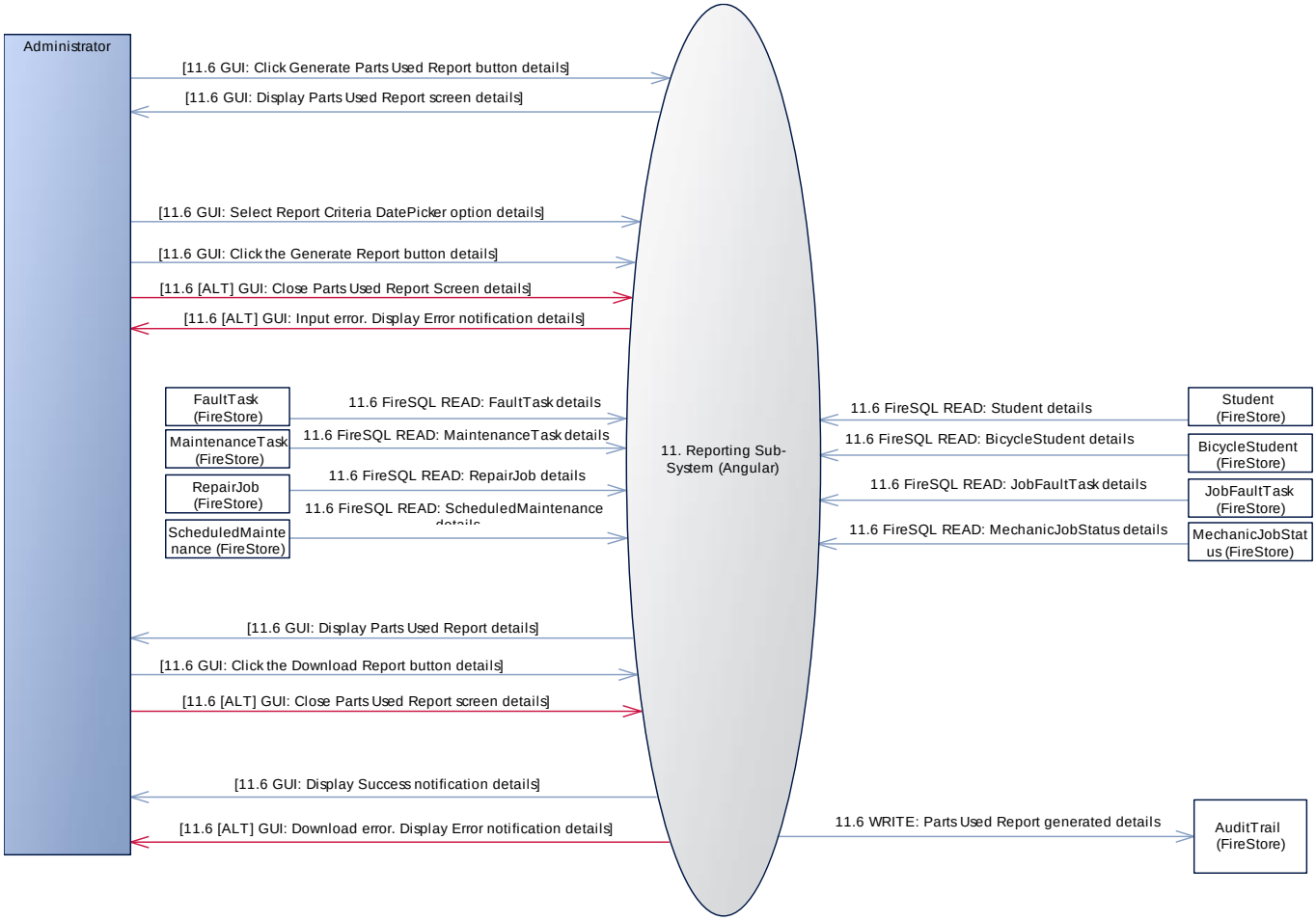


Figure 93 High-Level Diagram Sub-System 11: Part 6

Sub-System 12: Administration

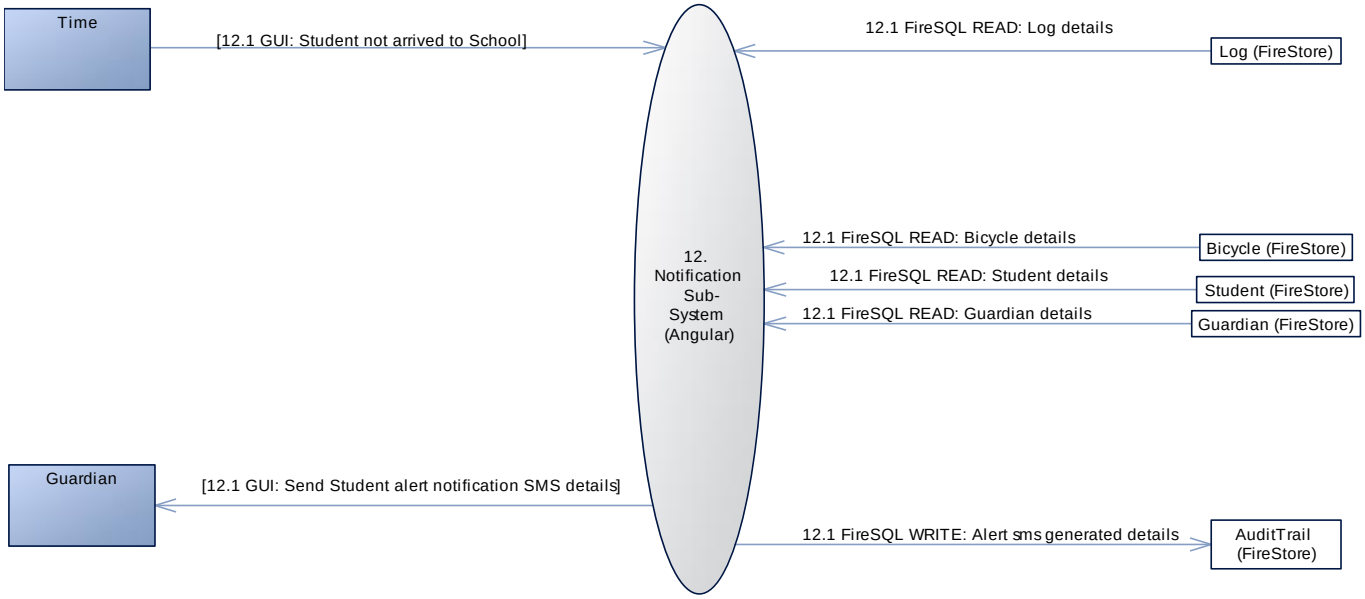


Figure 94 High-Level Diagram Sub-System 12: Part 1

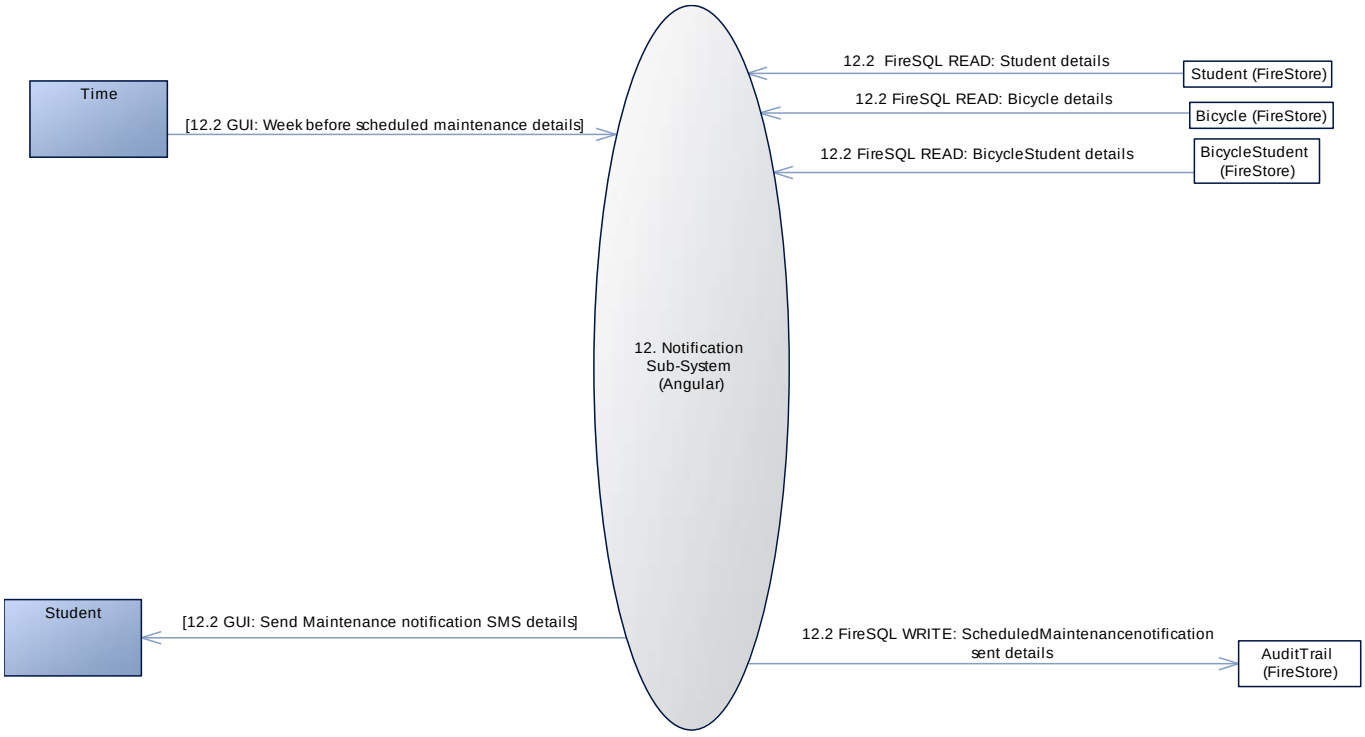


Figure 95 High-Level Diagram Sub-System 12: Part 2

Sub-System 13: Administration

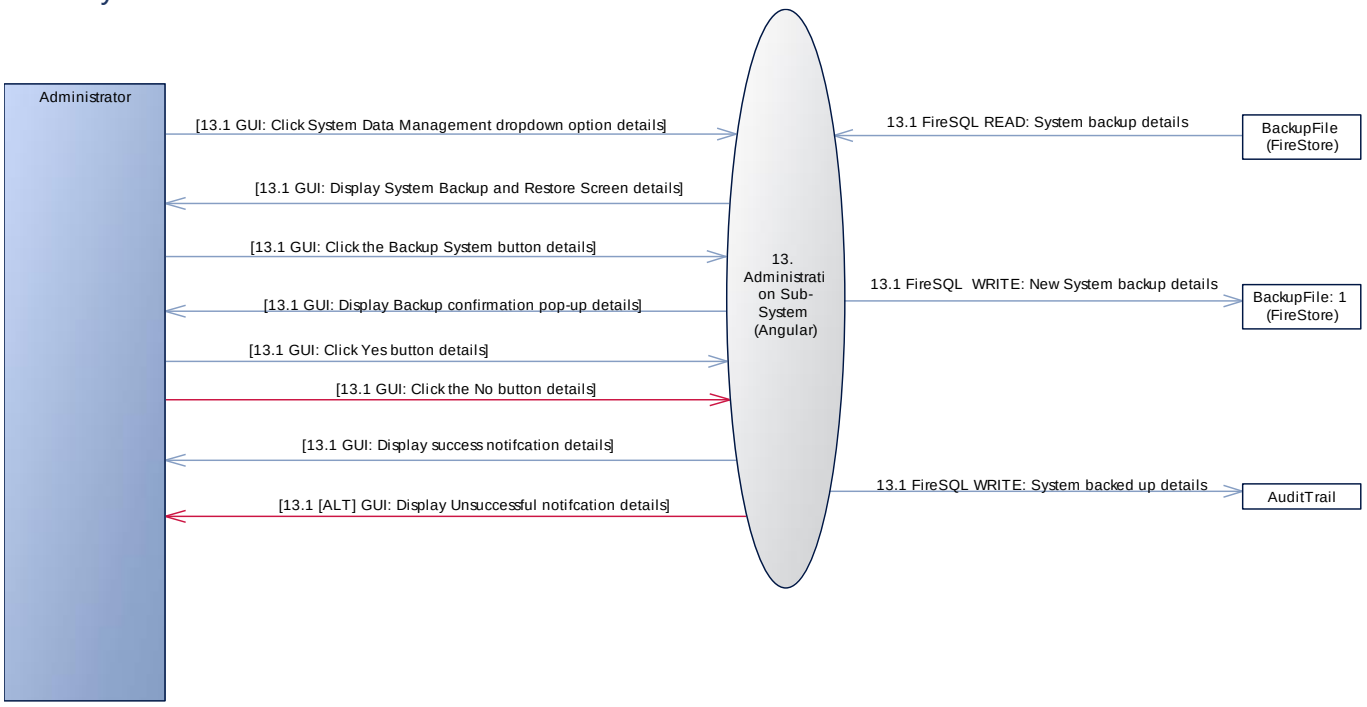


Figure 96 High-Level Diagram Sub-System 13: Part 1

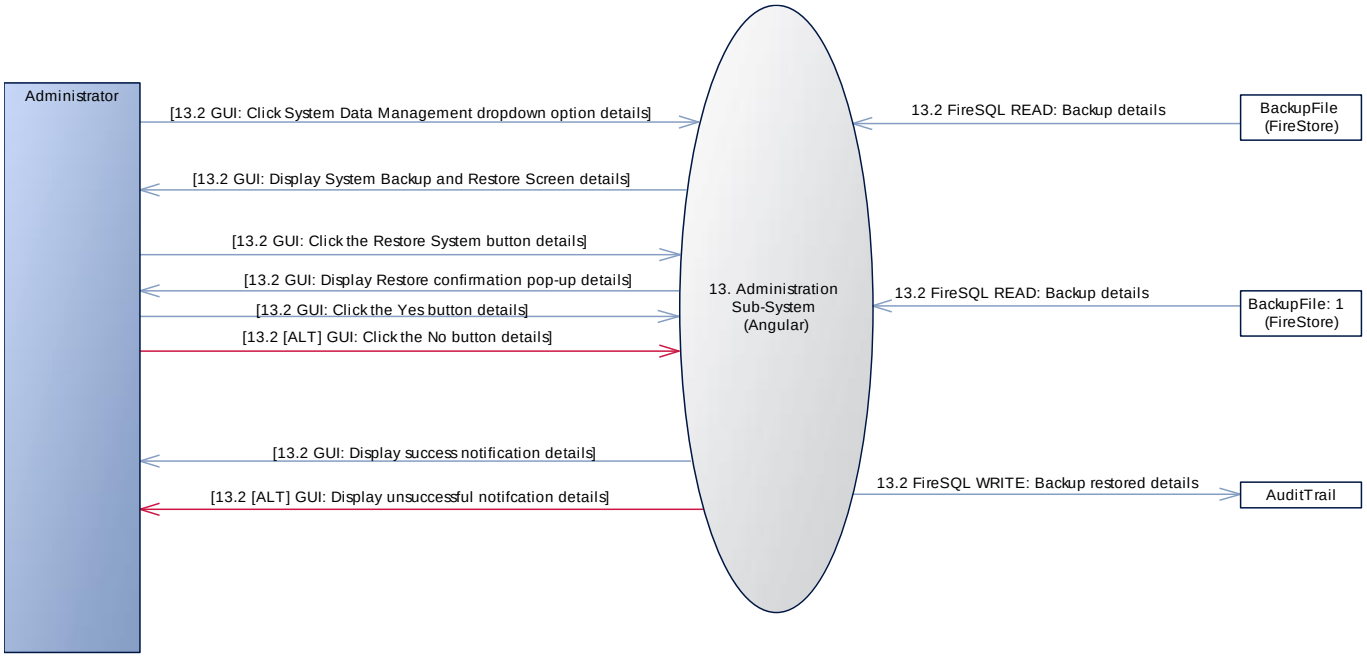


Figure 97 High-Level Diagram Sub-System 13: Part 2

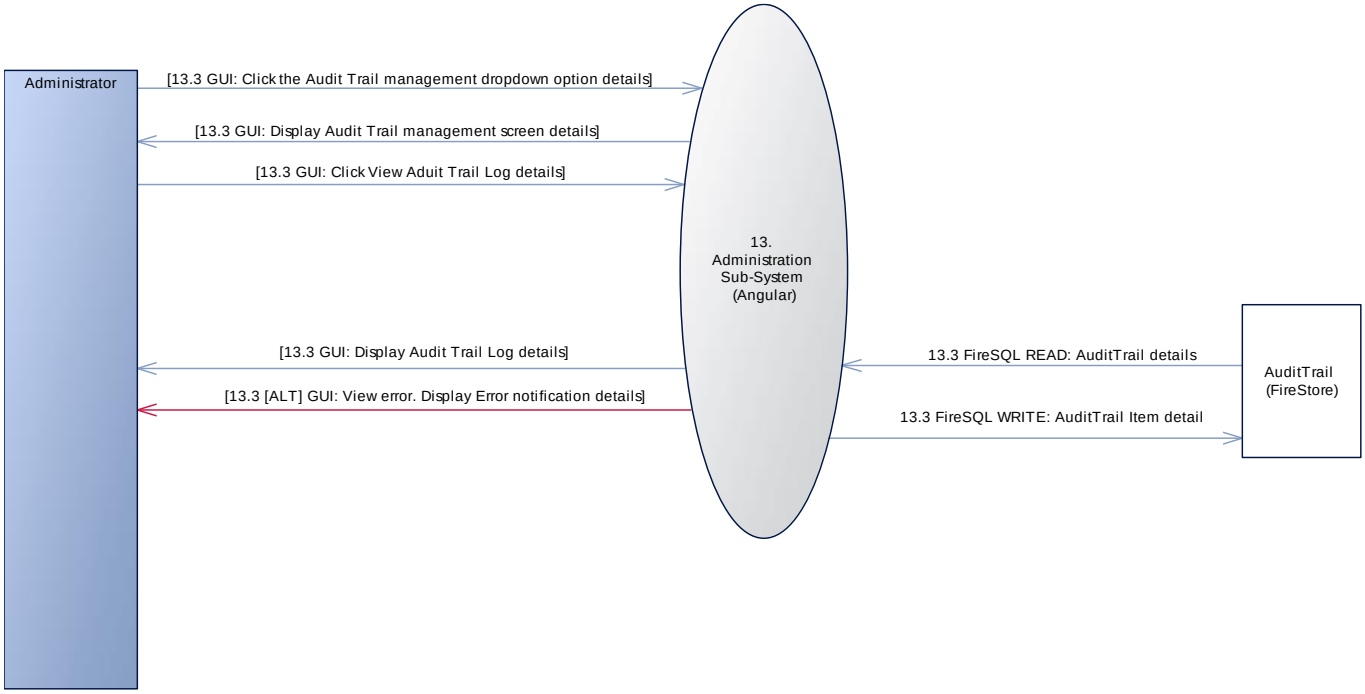


Figure 98 High-Level Diagram Sub-System 13: Part 3

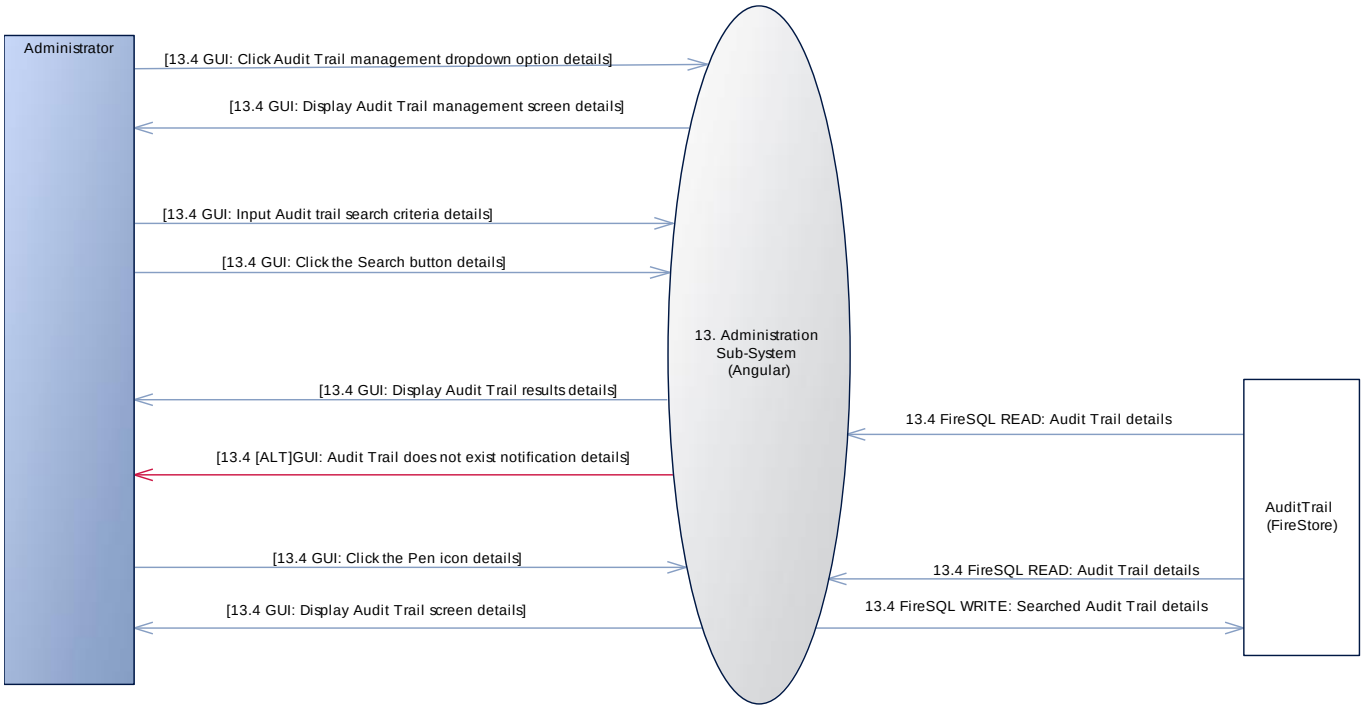


Figure 99 High-Level Diagram Sub-System 13: Part 4

Middle-Level Diagram

Sub System 1: User

1.1 Login

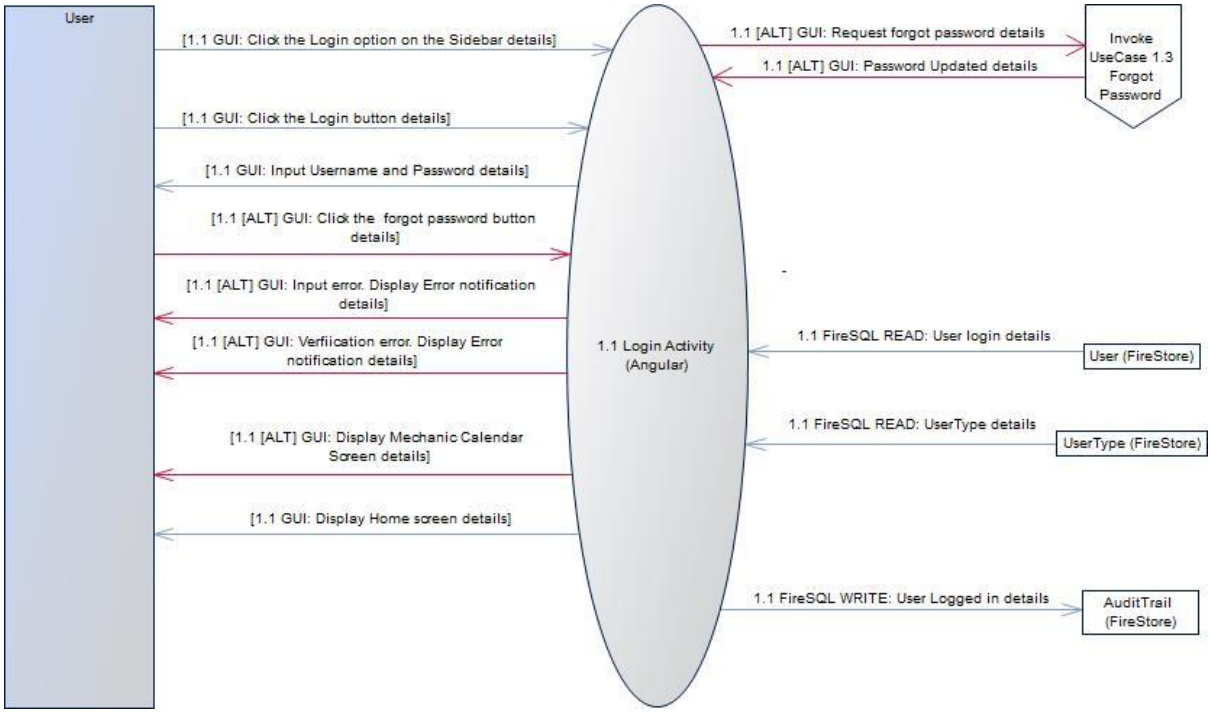


Figure 100 Middle-Level Diagram Sub-System 1: 1.1 login

1.2 Logout

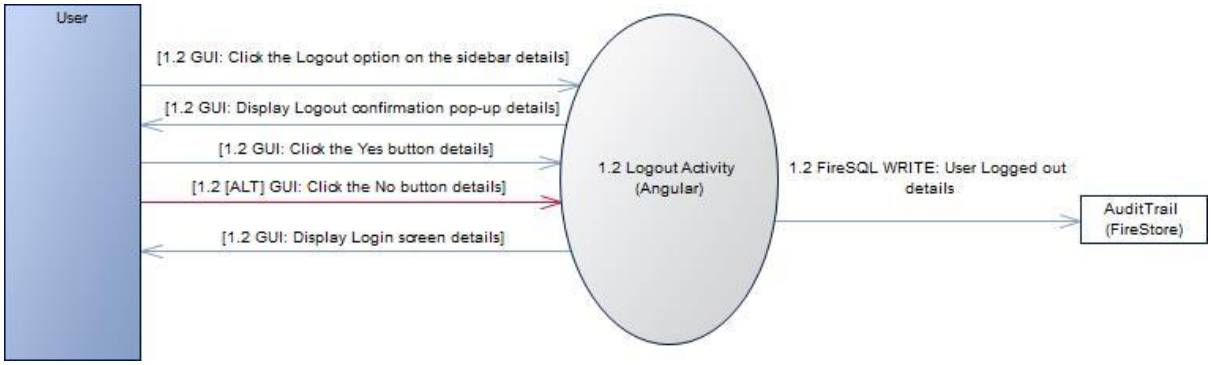


Figure 101 Middle-Level Diagram Sub-System 1: 1.2 Logout

1.3 Forgot Password

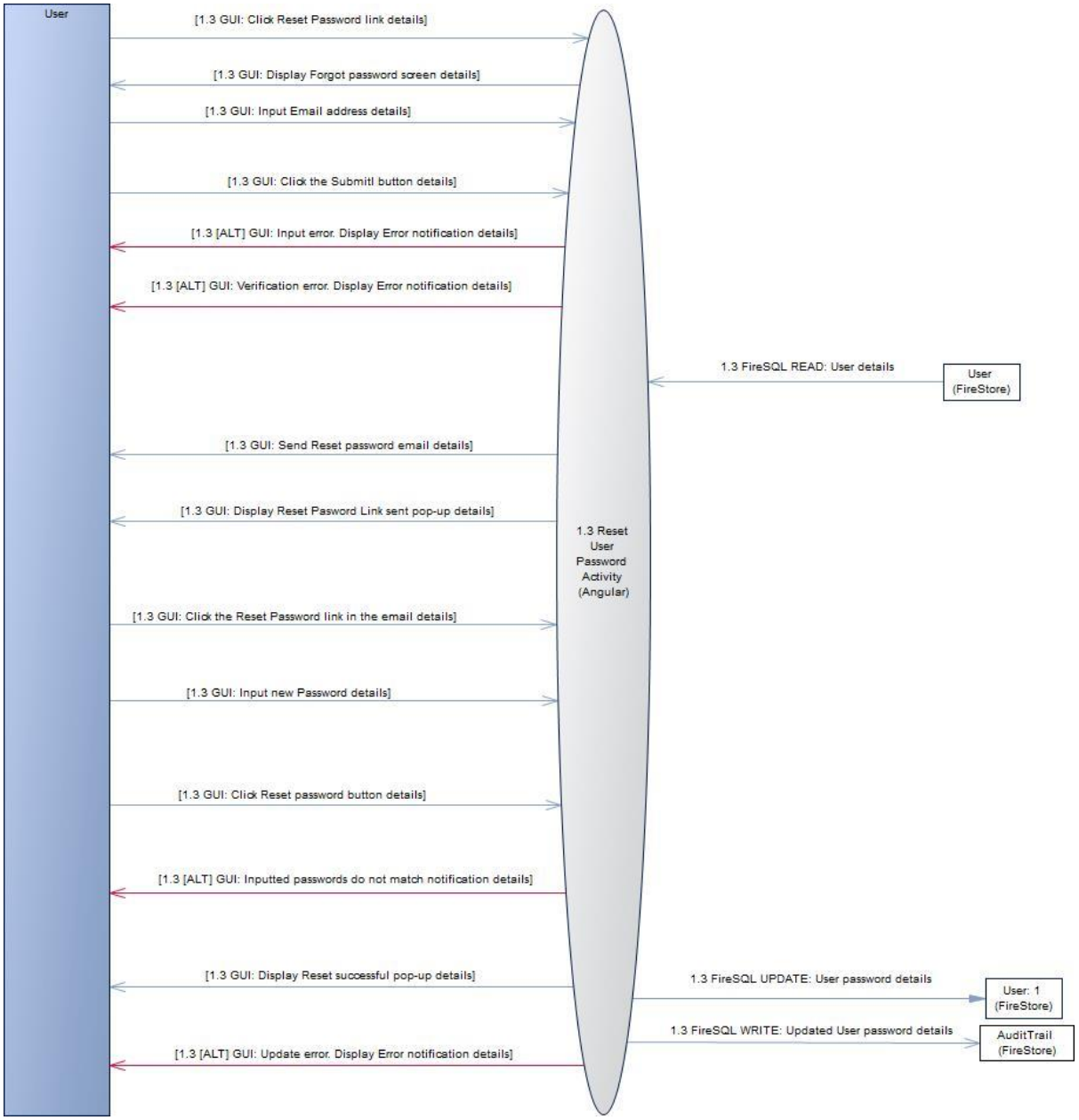


Figure 102 Middle-Level Diagram Sub-System 1: 1.3 Forgot Password

1.4 Add User

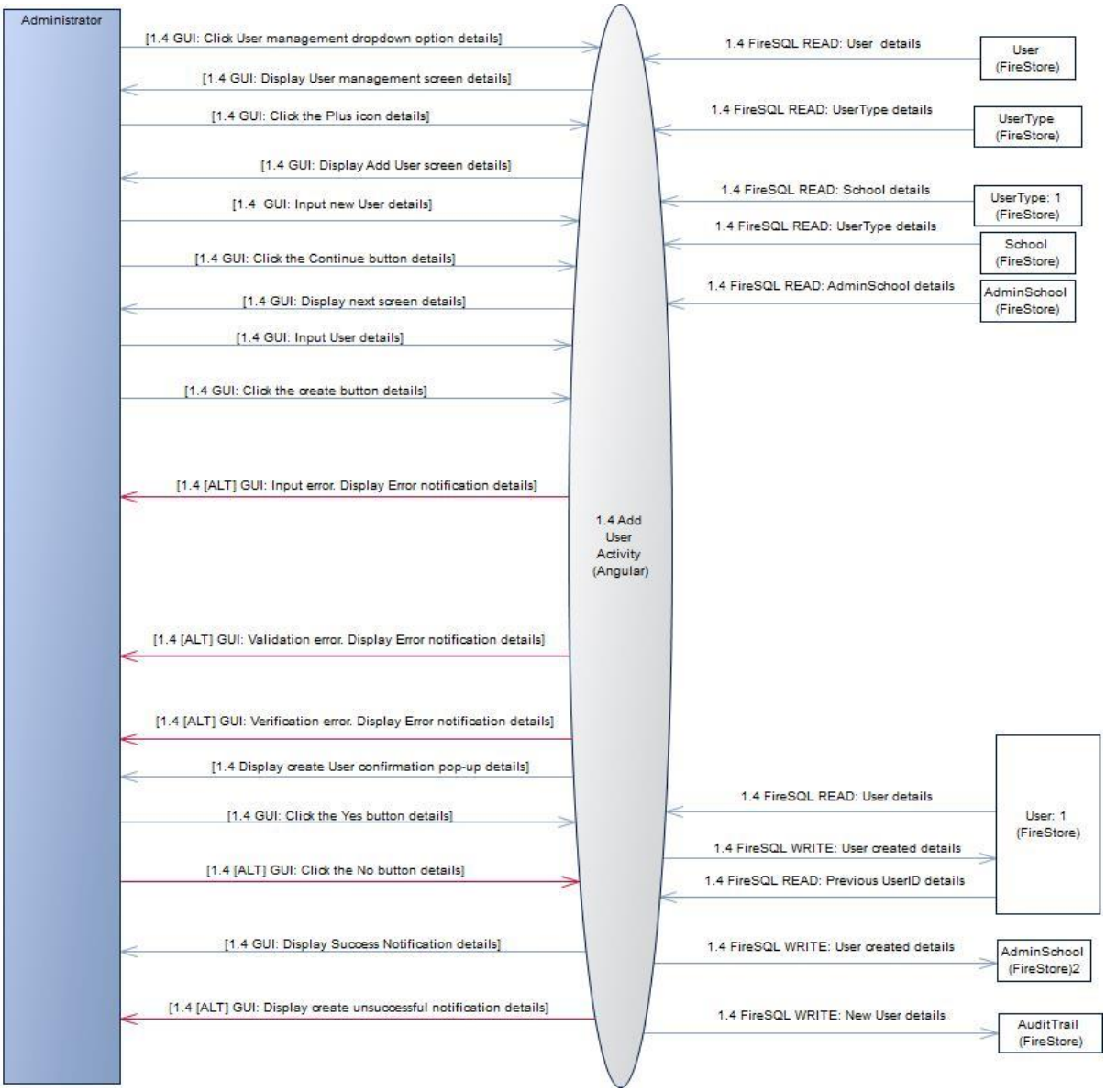


Figure 103 Middle-Level Diagram Sub-System 1: 1.4 Add User

1.5 Search User

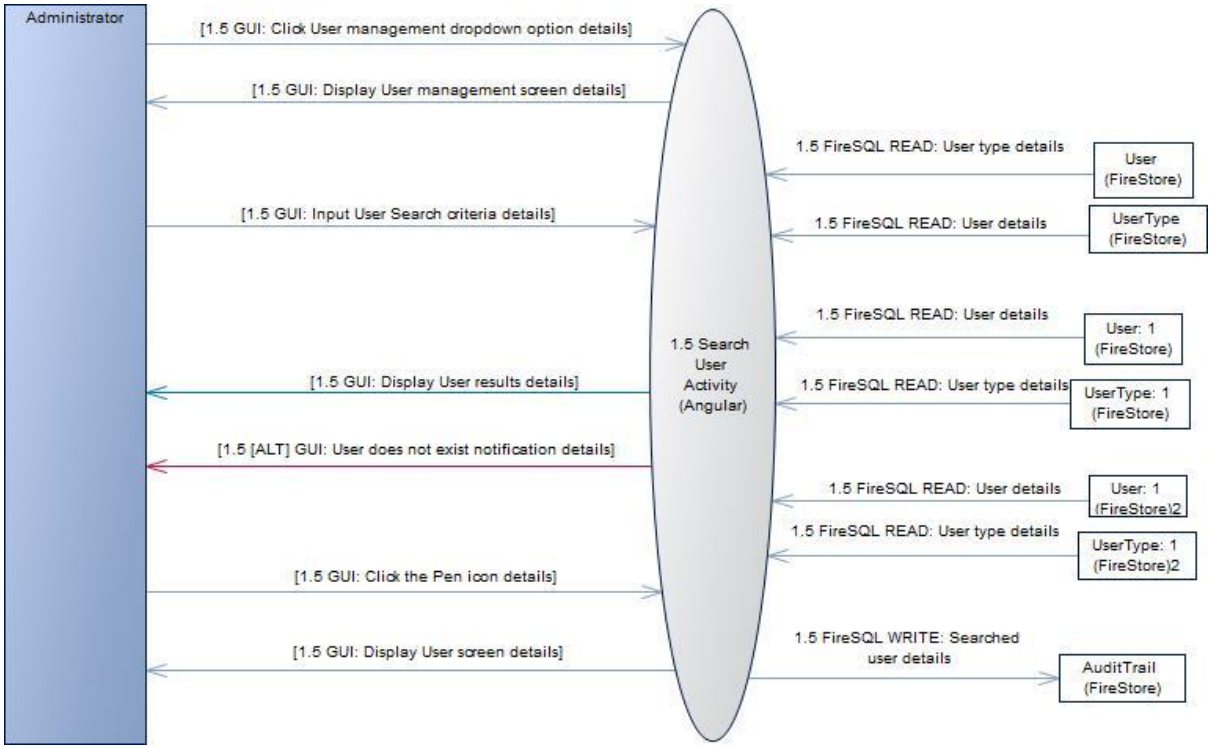


Figure 104 Middle-Level Diagram Sub-System 1: 1.5 Search User

1.6 Update User

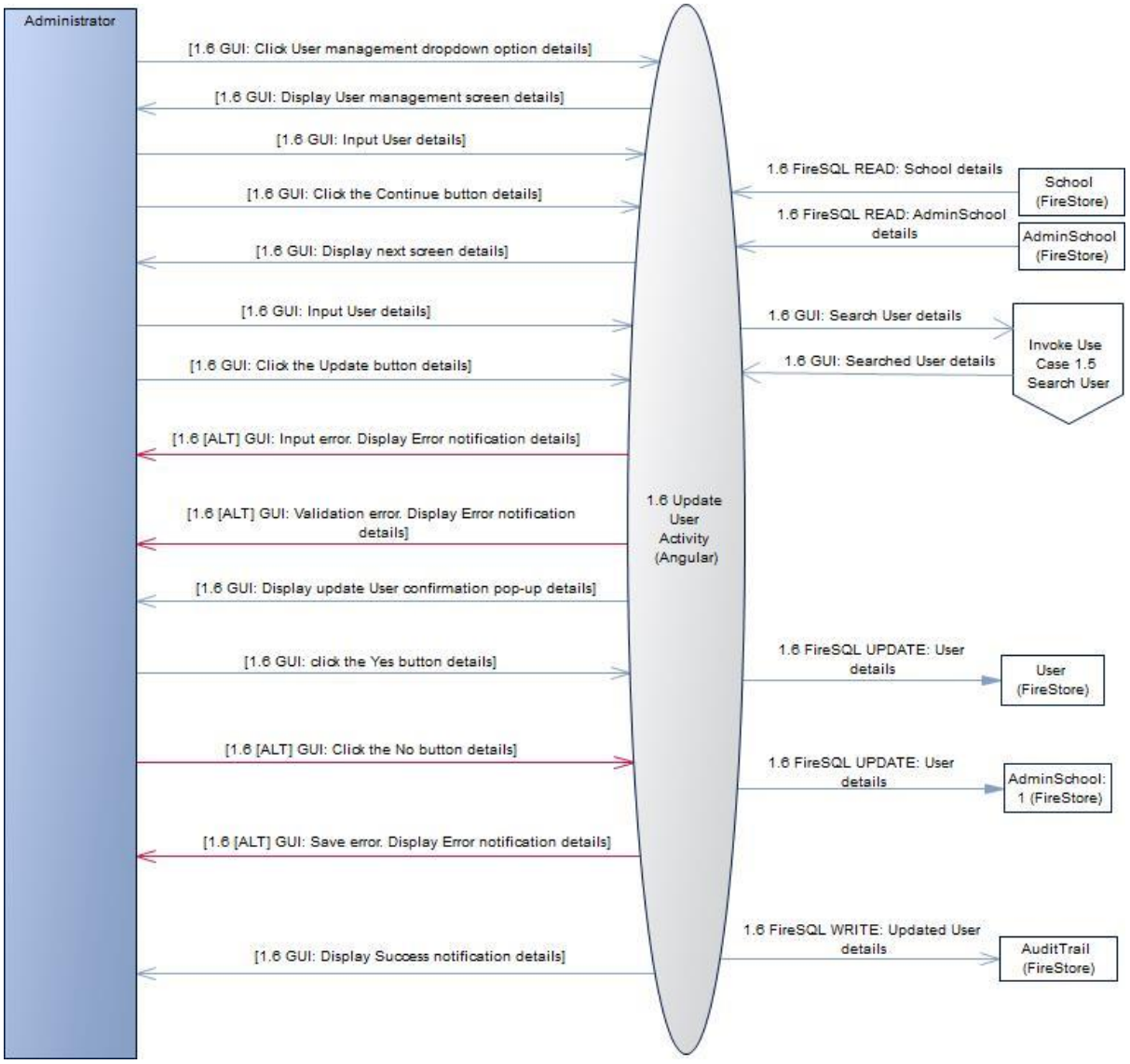


Figure 105 Middle-Level Diagram Sub-System 1: 1.6 Update User

1.7 Delete User

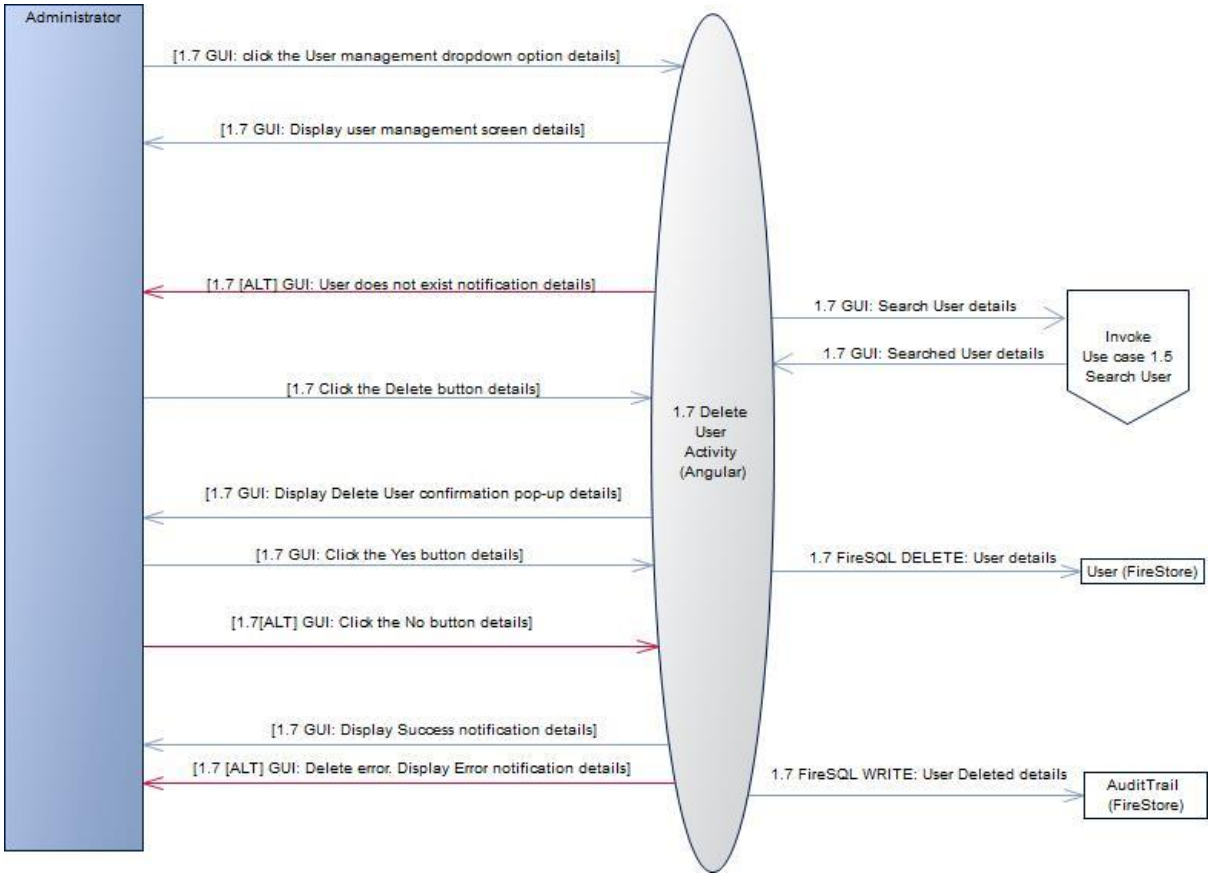


Figure 106 Middle-Level Diagram Sub-System 1: 1.7 Delete User

1.8 Add User Type

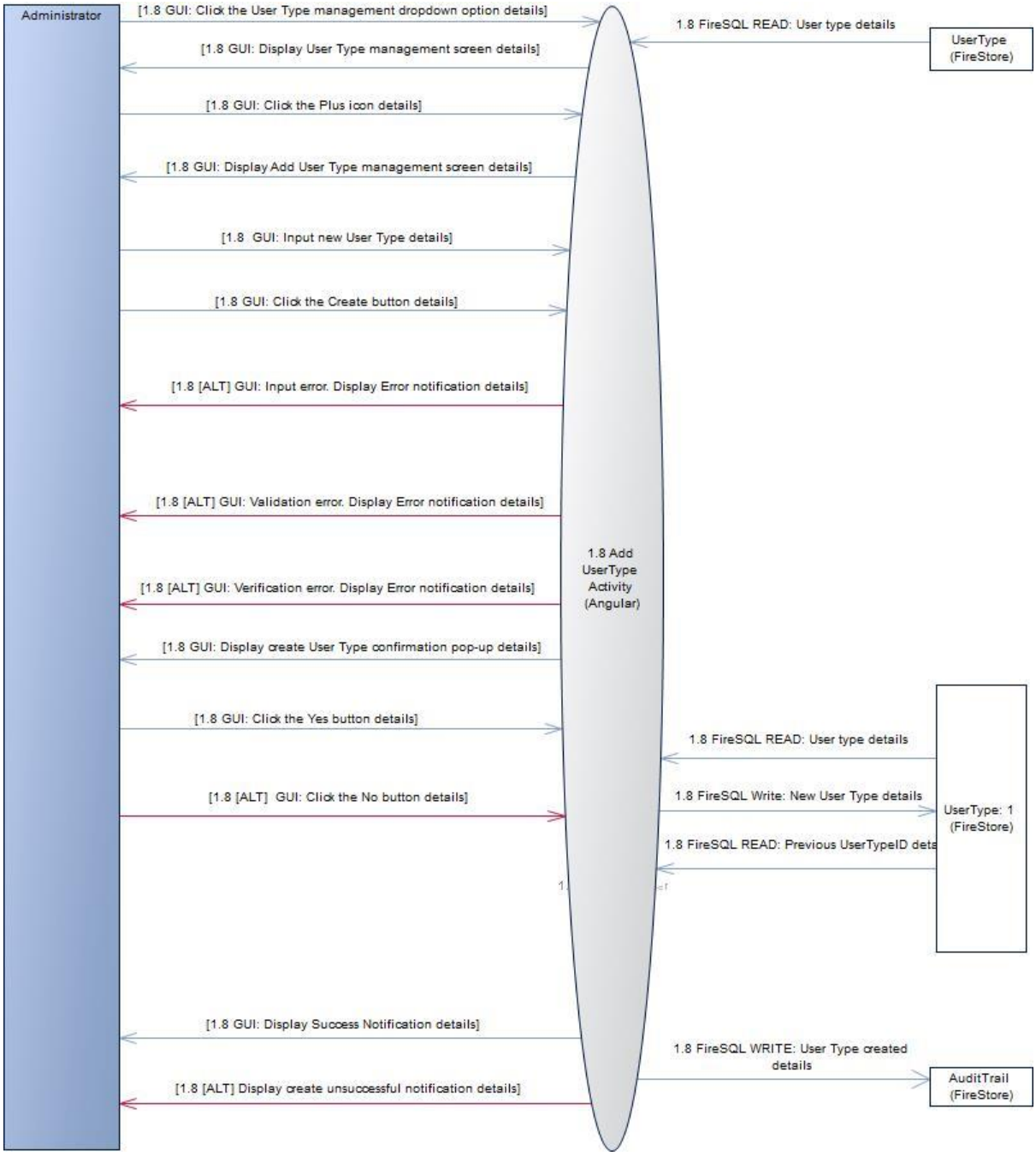


Figure 107 Middle-Level Diagram Sub-System 1: 1.8 Add User Type

1.9 Search User Type

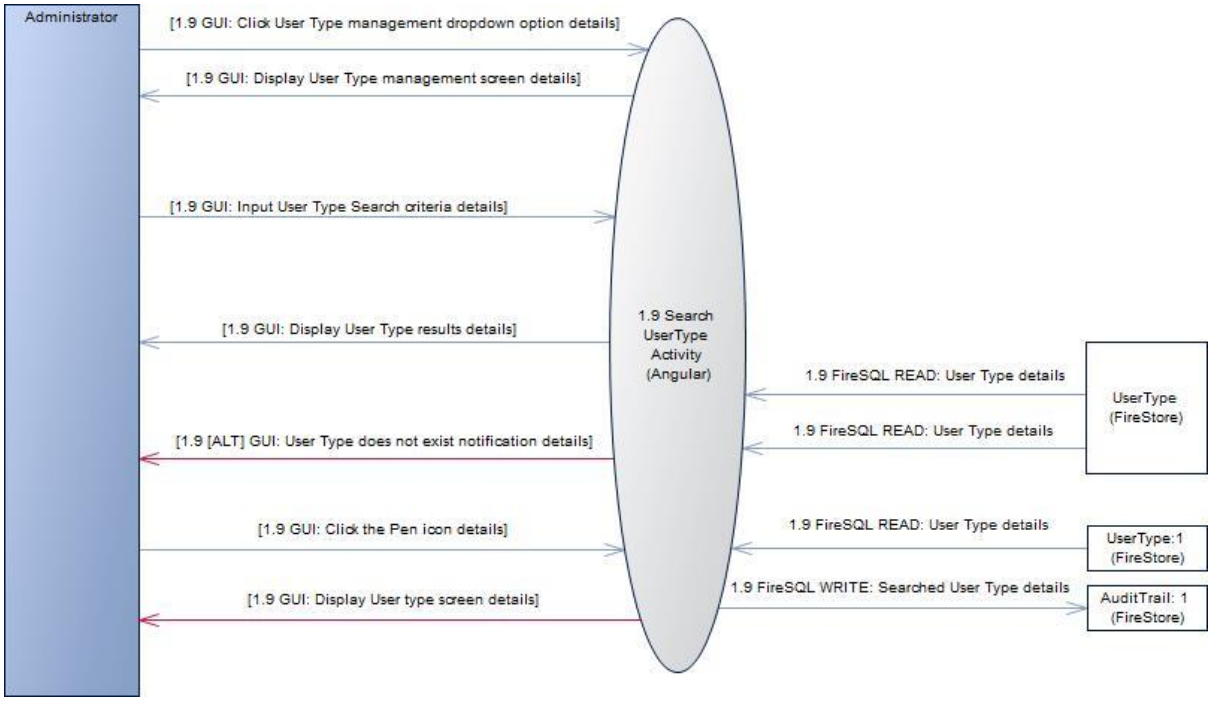


Figure 108 Middle-Level Diagram Sub-System 1: 1.9 Search User Type

1.10 Update User Type

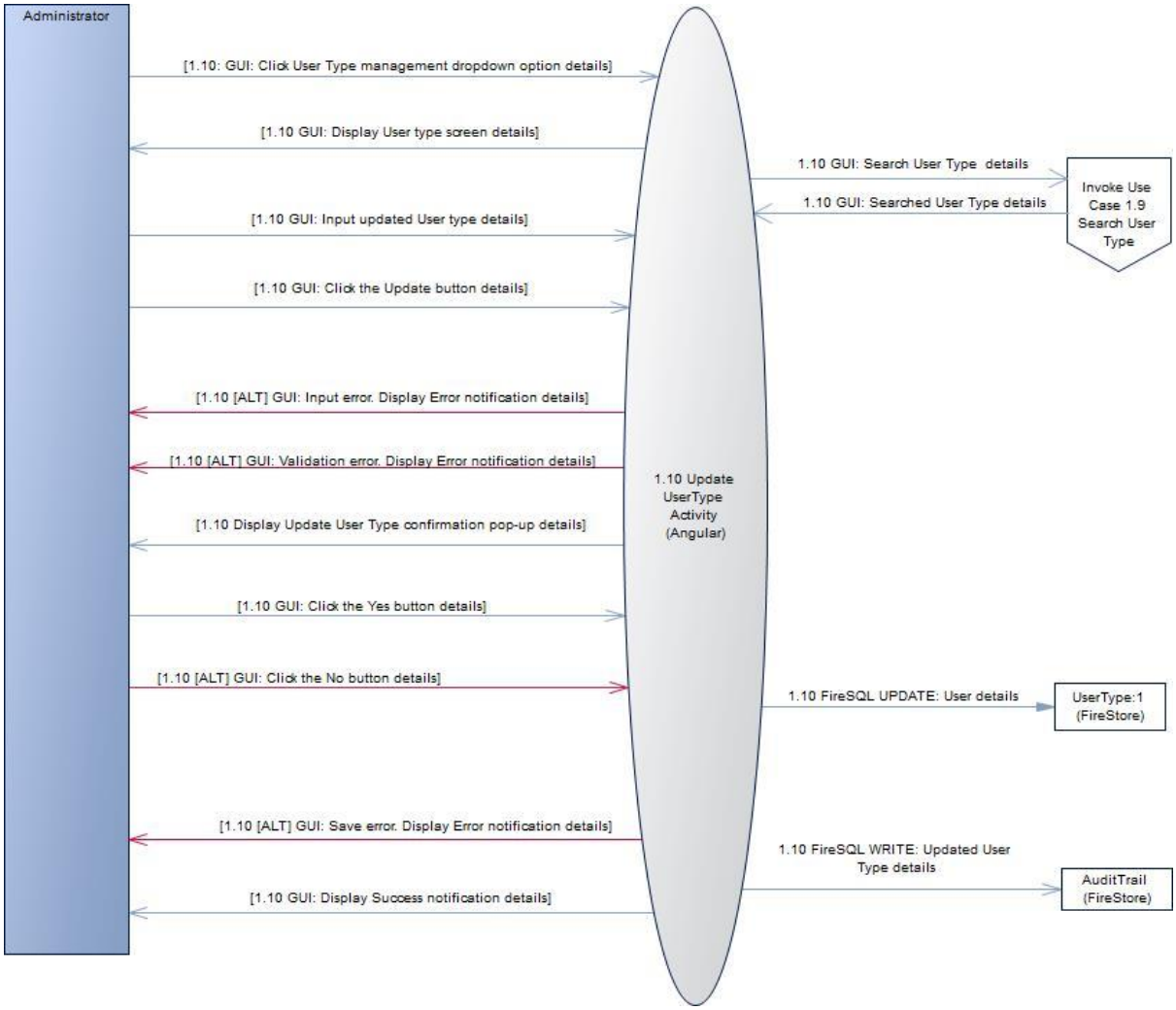


Figure 109 Middle-Level Diagram Sub-System 1: 1.10 Update User Type

1.11 Delete User Type

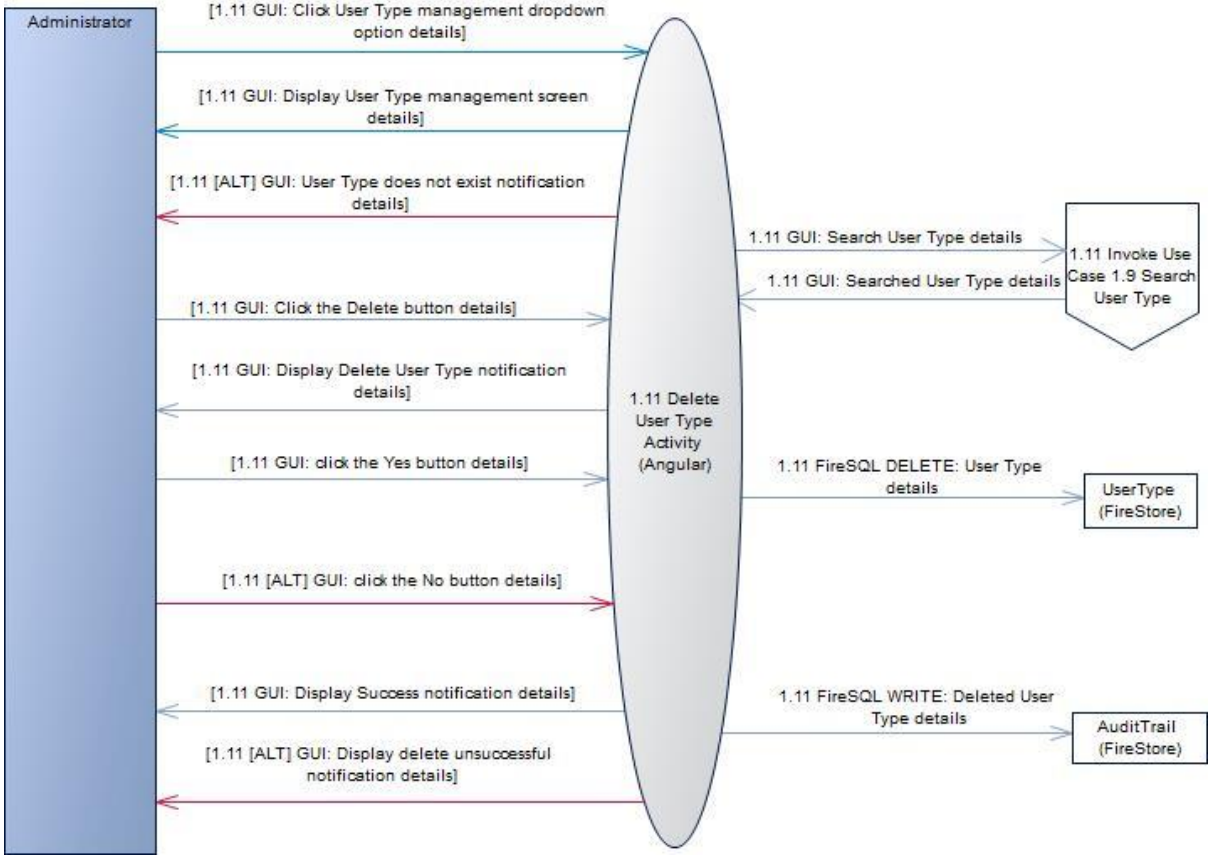


Figure 110 Middle-Level Diagram Sub-System 1: 1.11 Delete User Type

Sub-System 2: Mechanic Schedule

2.1 View Job

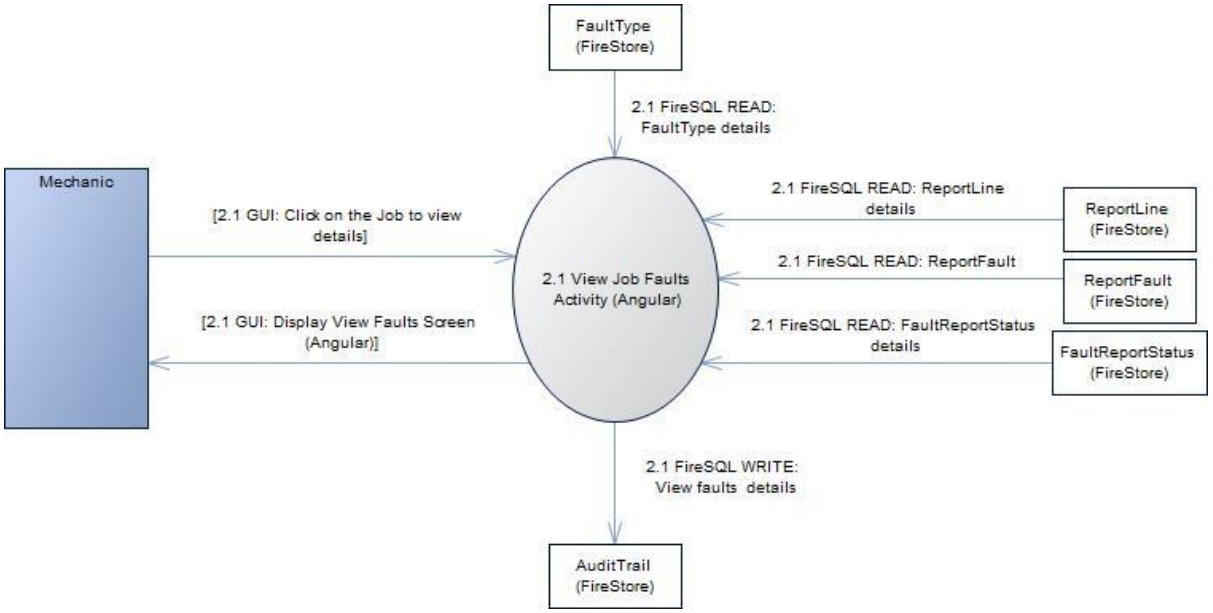


Figure 111 Middle-Level Diagram Sub-System 2: 2.1 View Job

2.2 View Job List

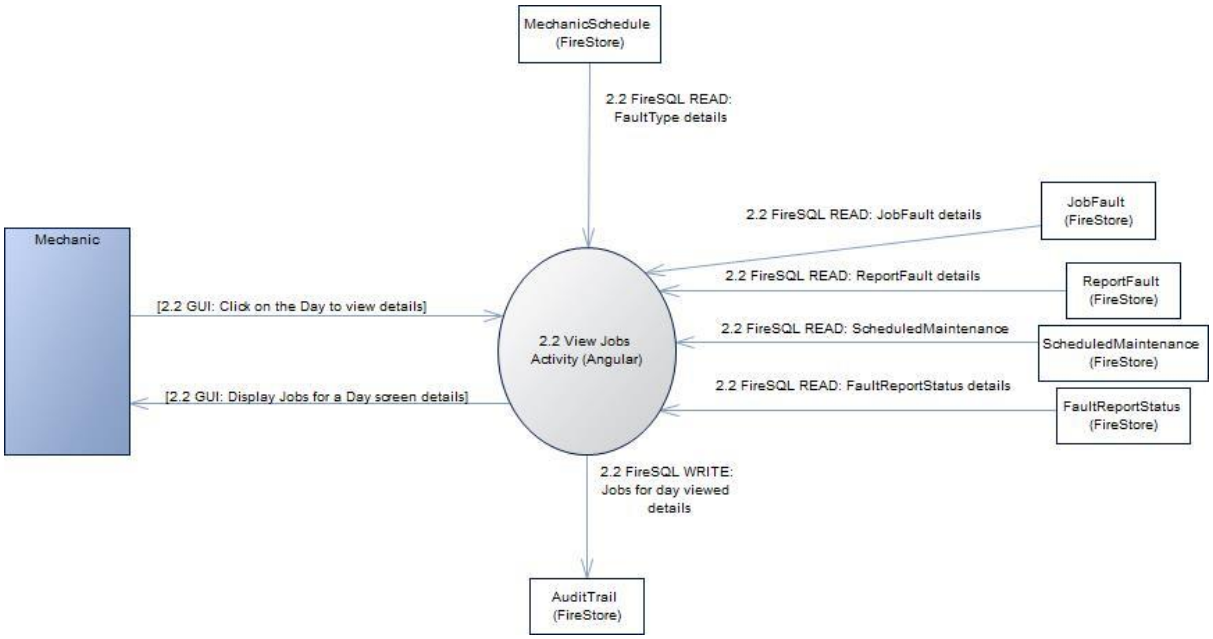


Figure 112 Middle-Level Diagram Sub-System 2: 2.2 View Job List

Sub-System 3: Tracking Hardware

3.1 Add Antenna

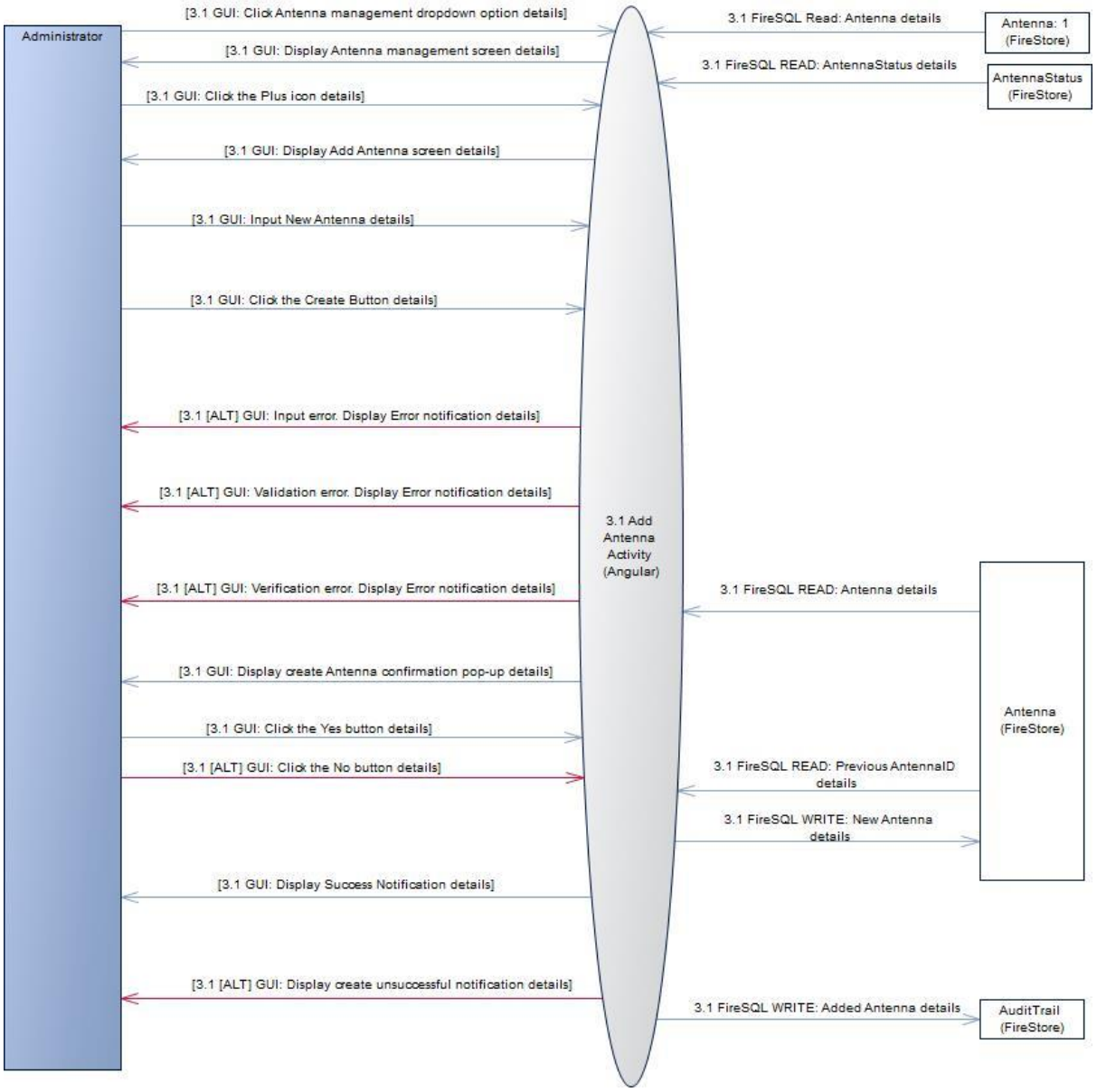


Figure 113 Middle-Level Diagram Sub-System 3: 3.1 Add Antenna

3.2 Search Antenna

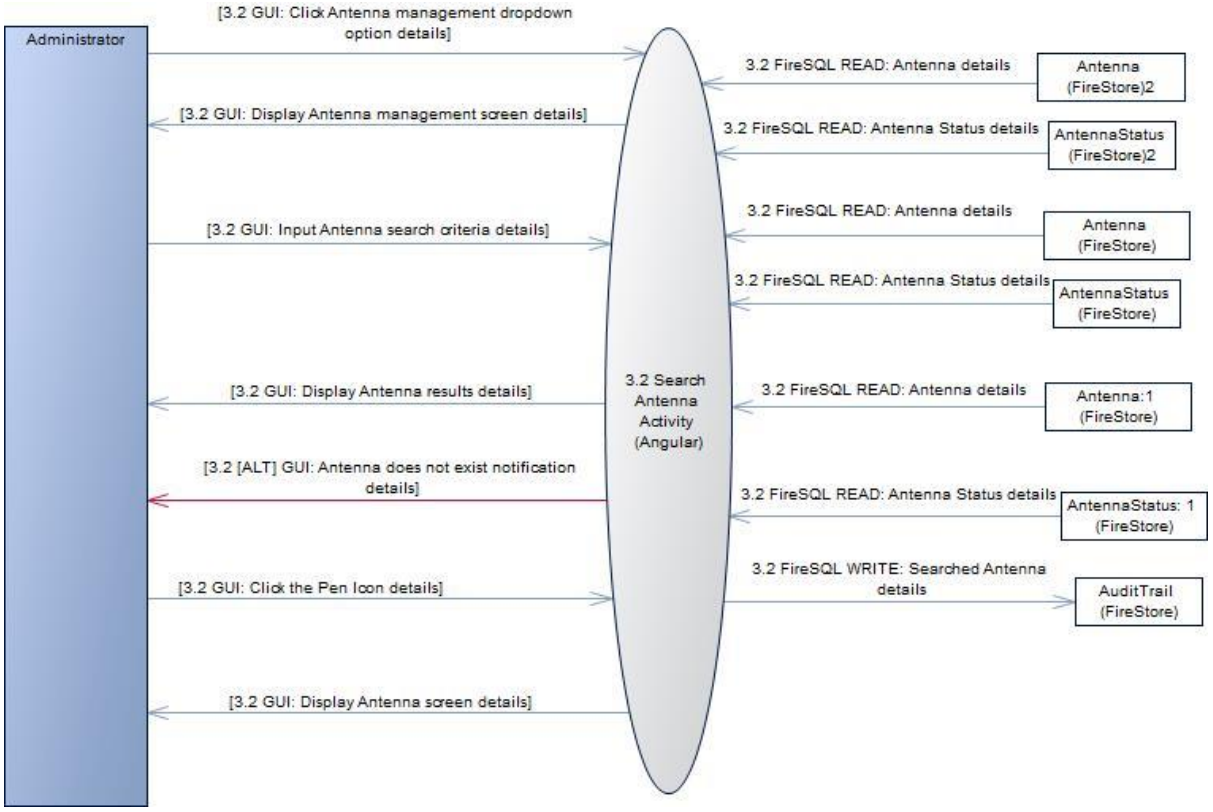


Figure 114 Middle-Level Diagram Sub-System 3: 3.2 Search Antenna

3.3 Update Antenna

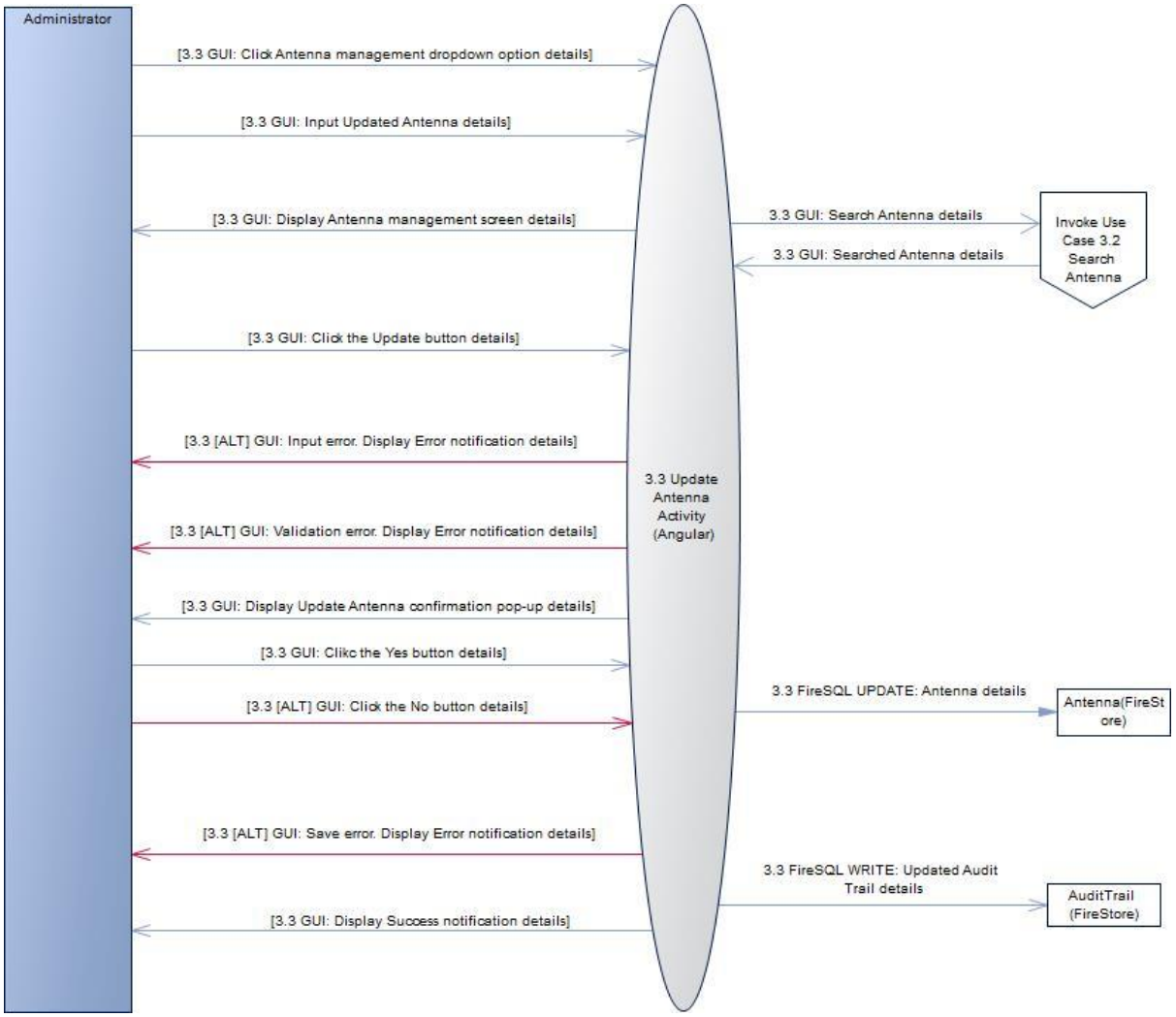


Figure 115 Middle-Level Diagram Sub-System 3: 3.3 Update Antenna

3.4 Delete Antenna

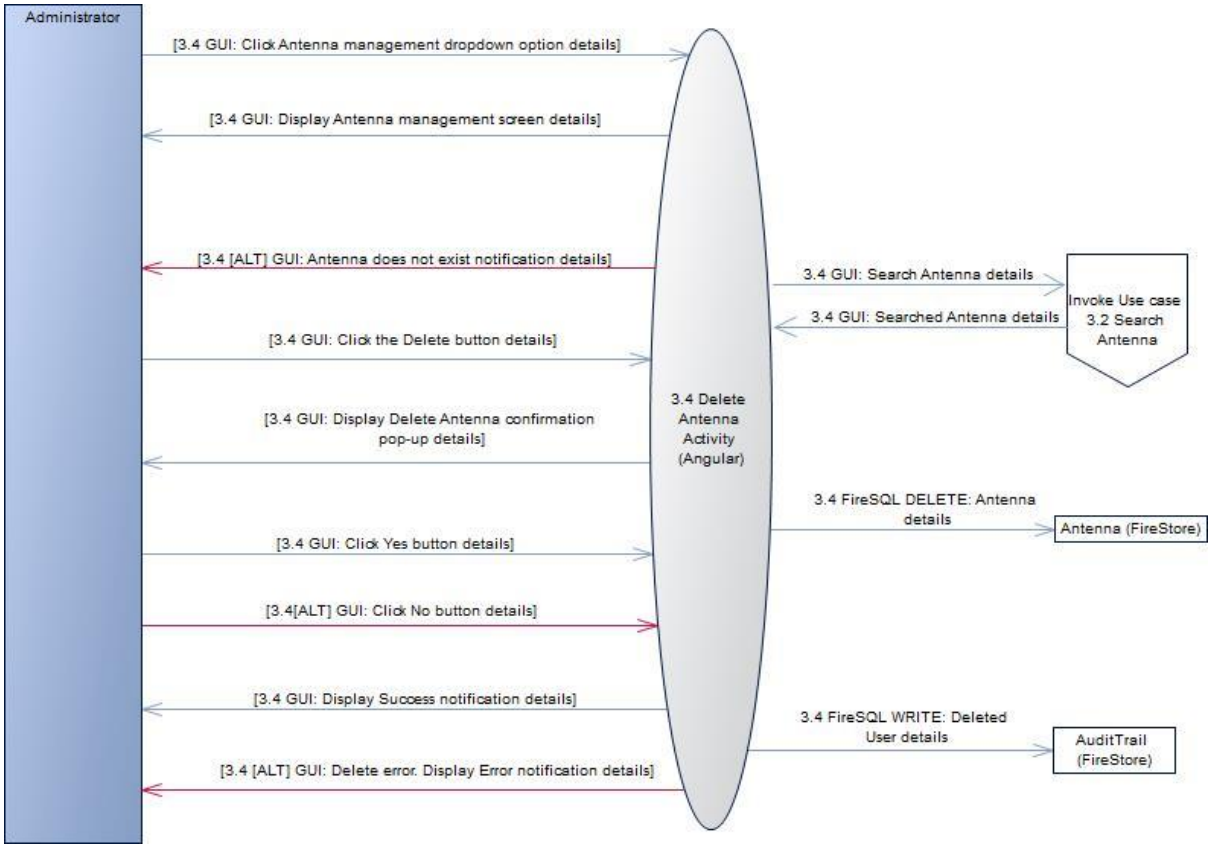


Figure 116 Middle-Level Diagram Sub-System 3: 3.4 Delete Antenna

3.5 Add Tag

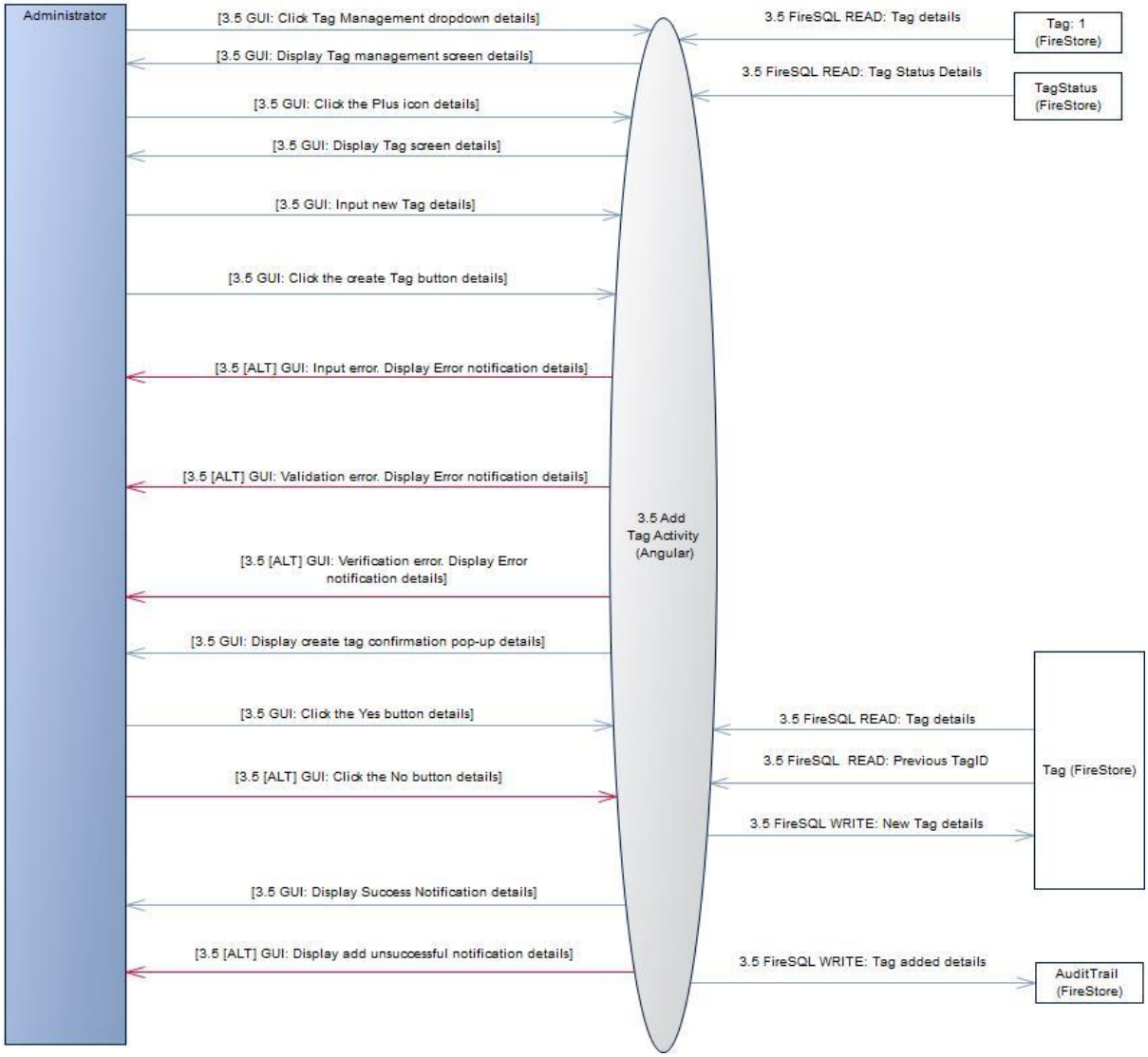


Figure 117 Middle-Level Diagram Sub-System 3: 3.5 Add Tag

3.7 Update Tag

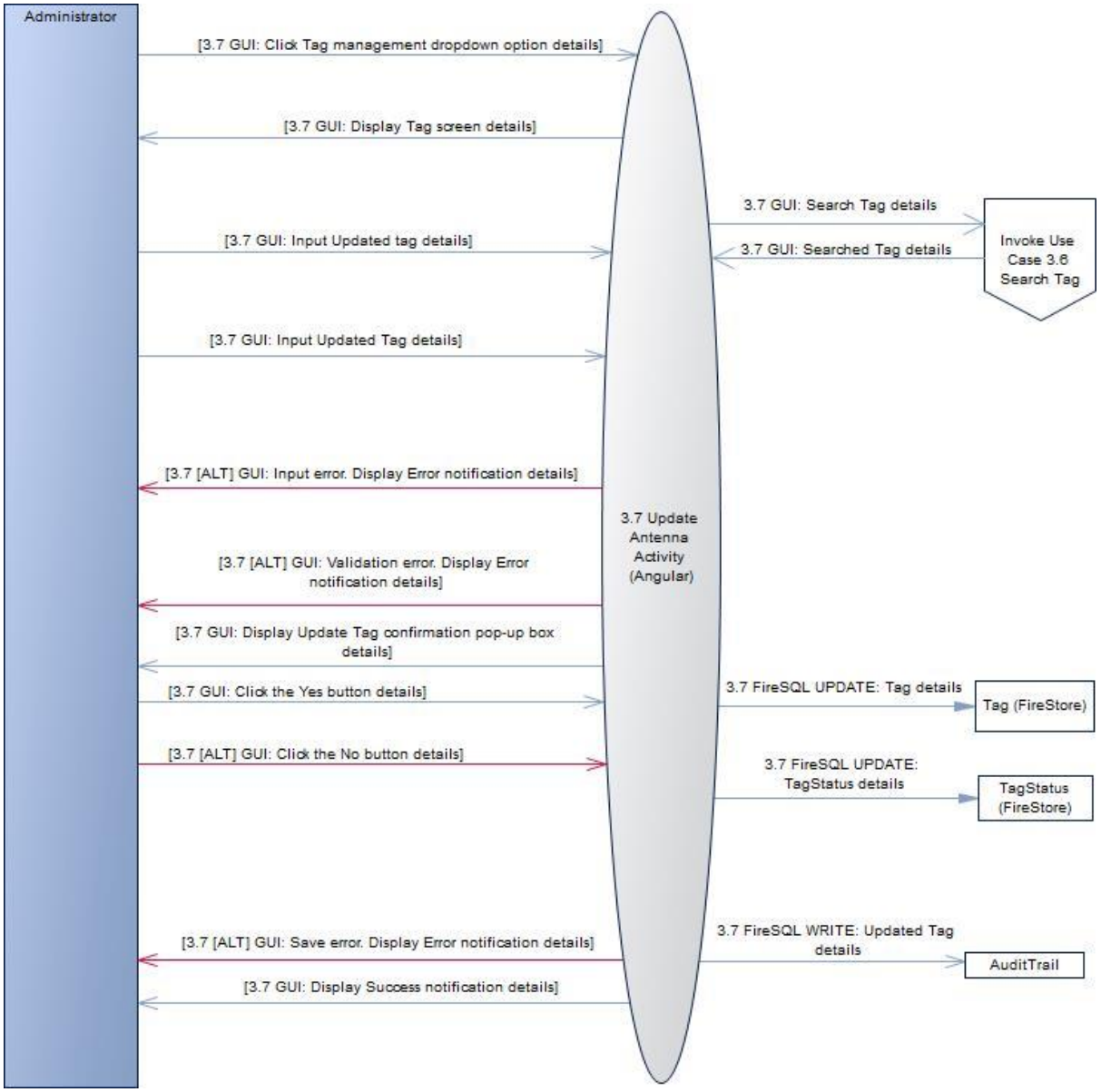


Figure 118 Middle-Level Diagram Sub-System 3: 3.7Update Tag

3.8 Delete Tag

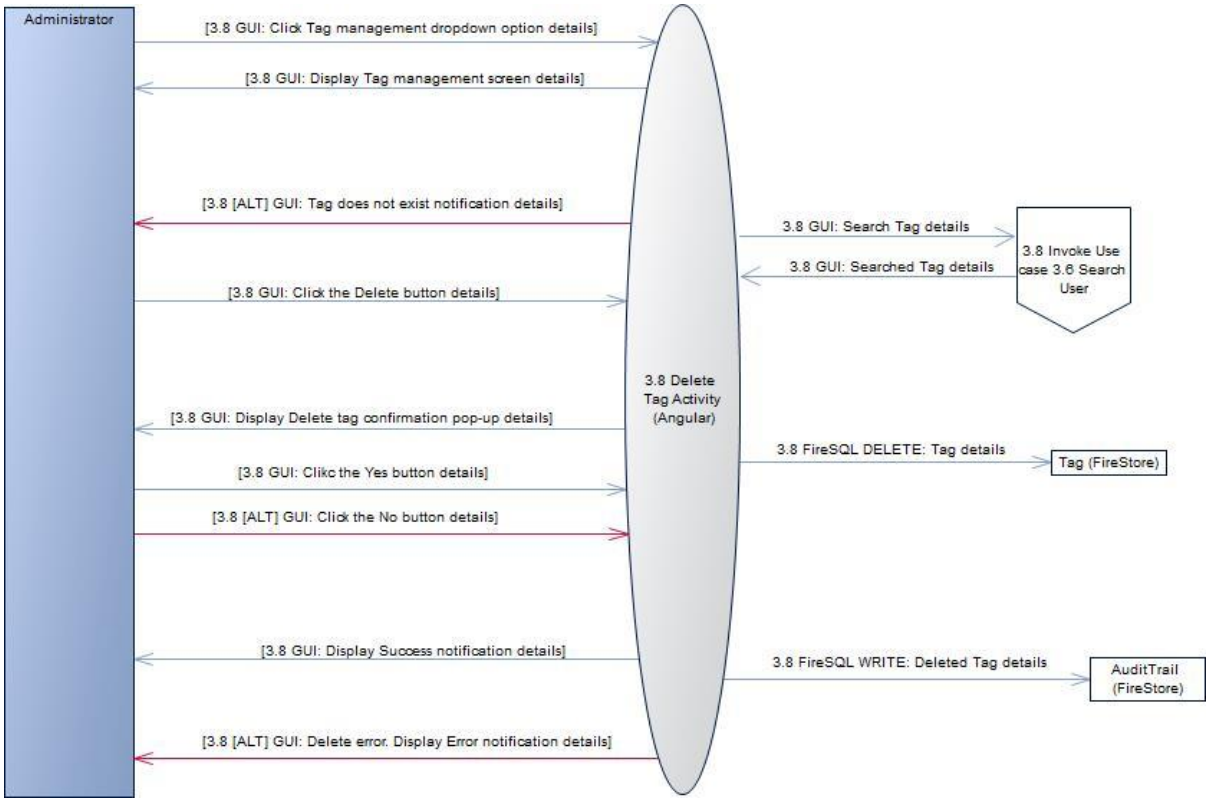


Figure 119 Middle-Level Diagram Sub-System 3: 3.8 Delete Tag

Sub-System 4: Student

4.1 Add Student

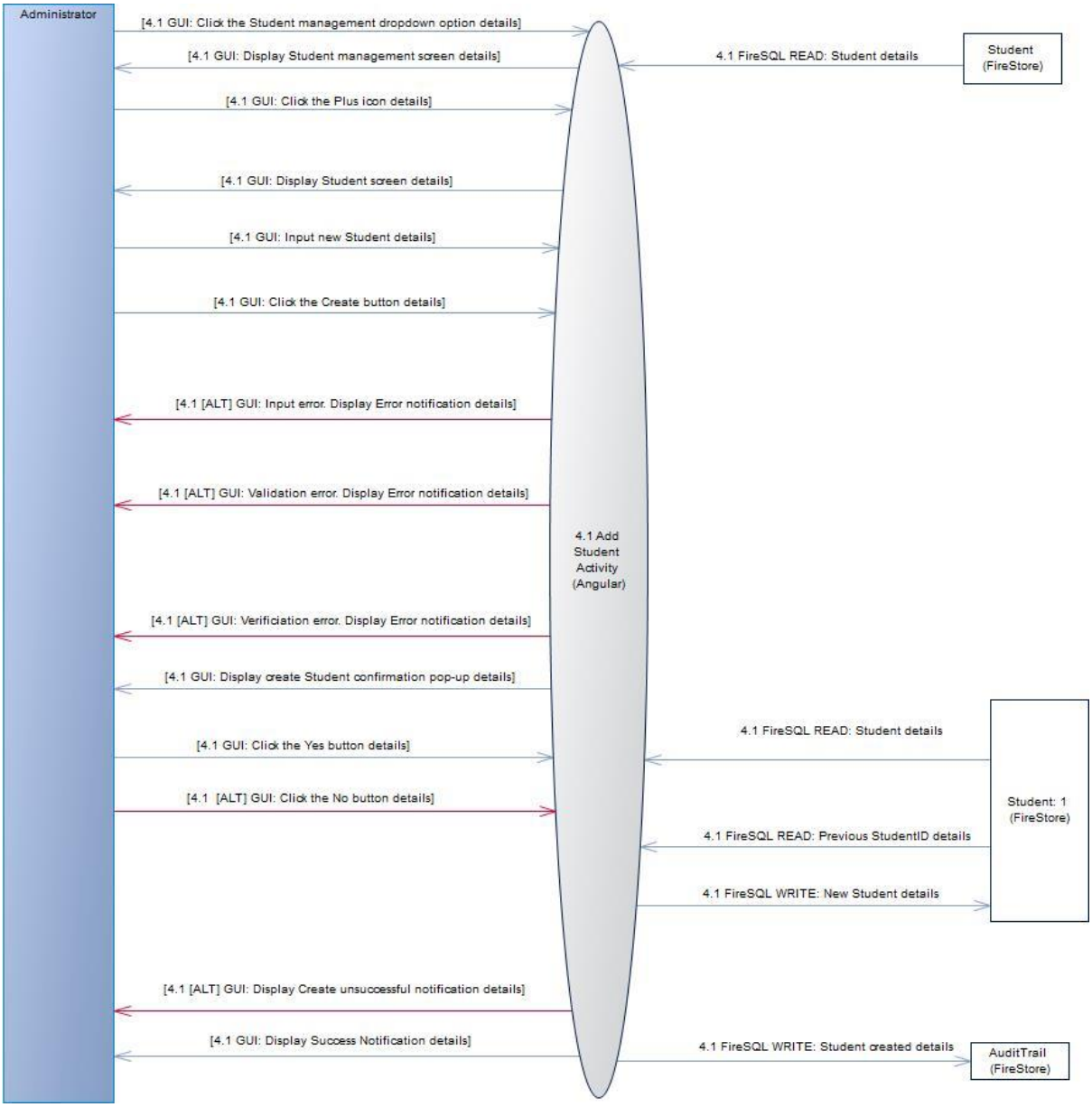


Figure 120 Middle-Level Diagram Sub-System 4: 4.1 Add Student

4.2 Search Student

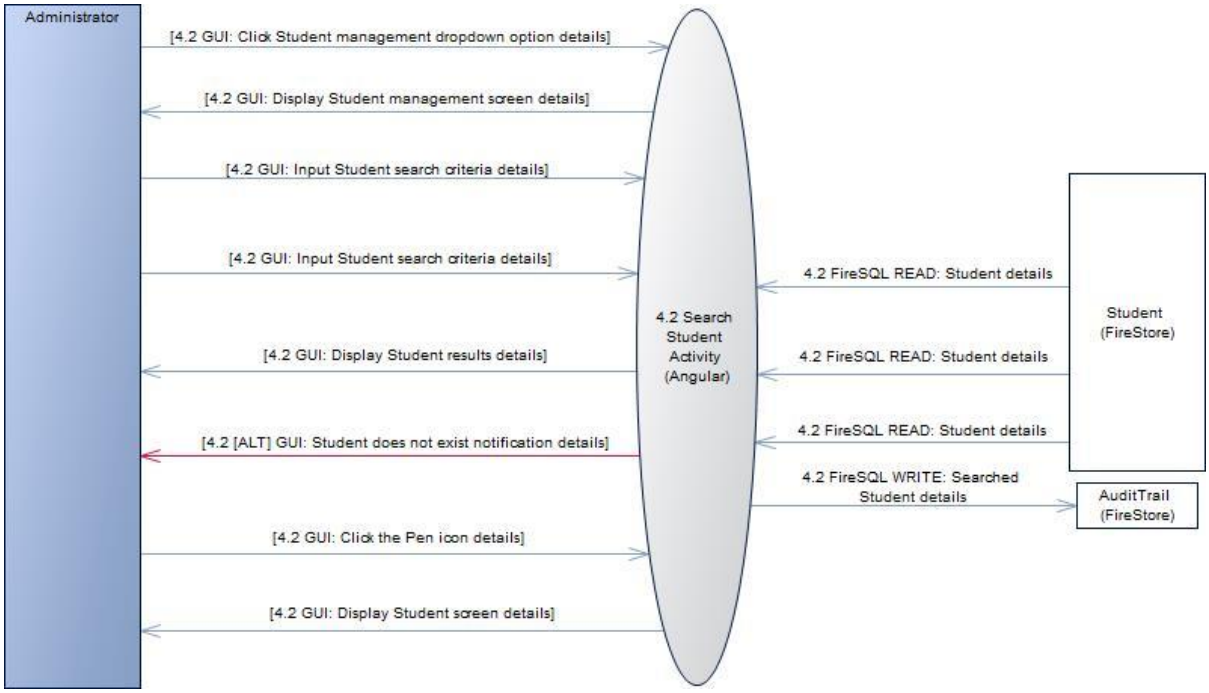


Figure 121 Middle-Level Diagram Sub-System 4: 4.2 Search Student

4.3 Update Student

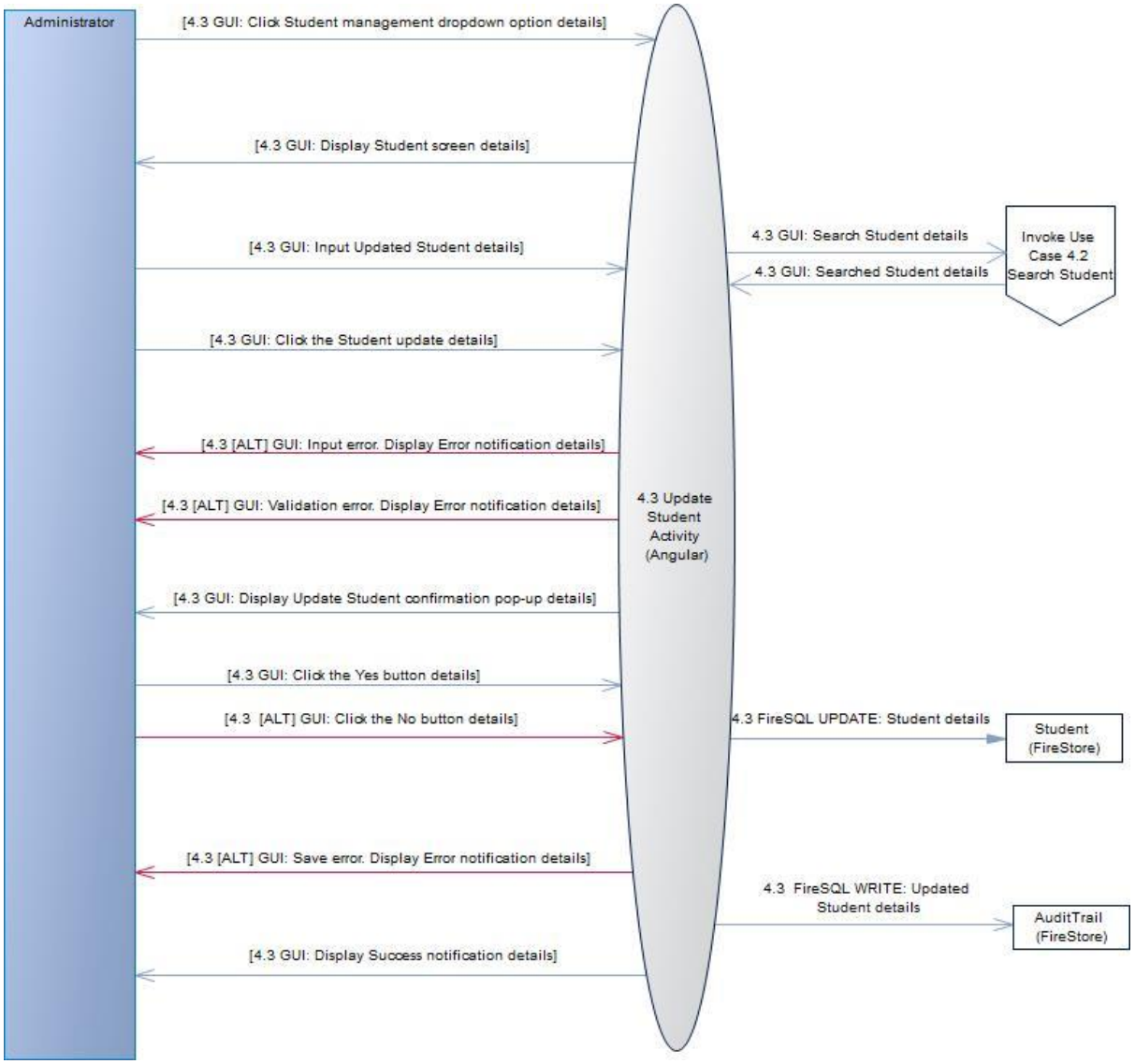


Figure 122 Middle-Level Diagram Sub-System 4: 4.3 Update Student

4.4 Delete Student

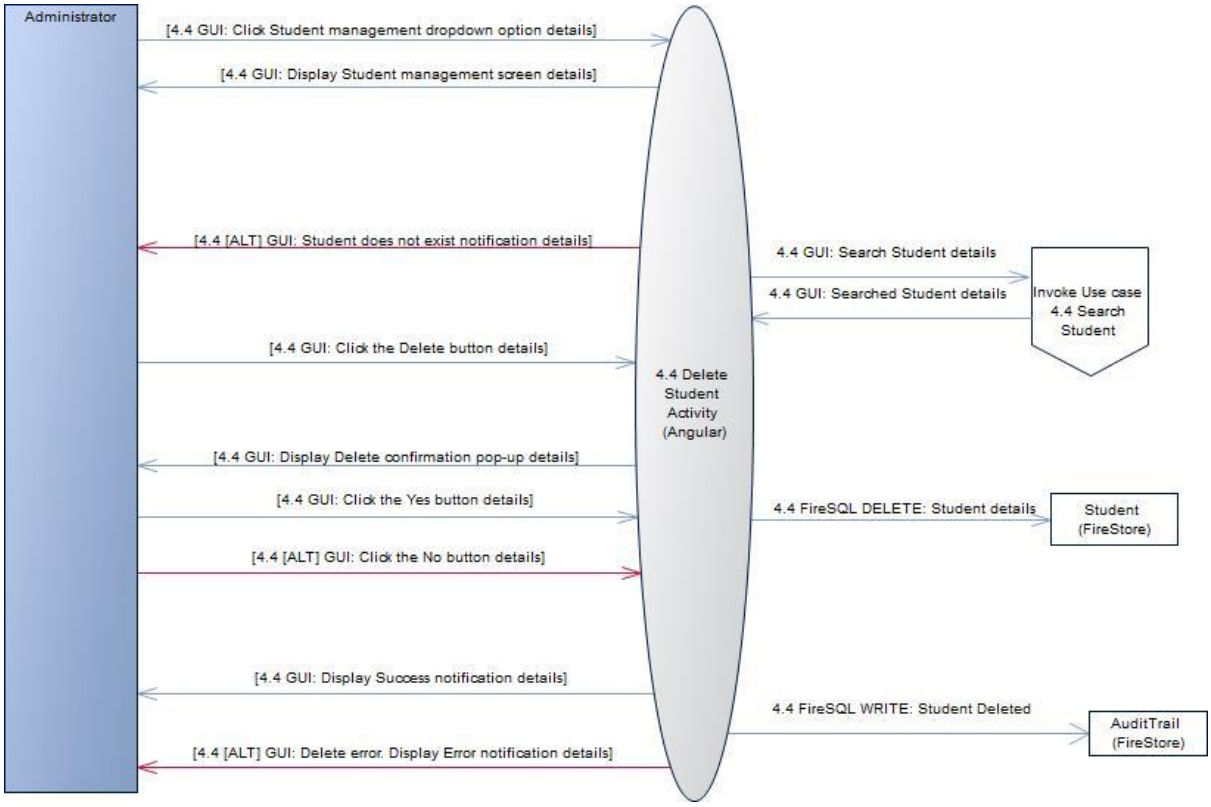


Figure 123 Middle-Level Diagram Sub-System 4: 4.4 Delete Student

4.5 Capture Student Mark

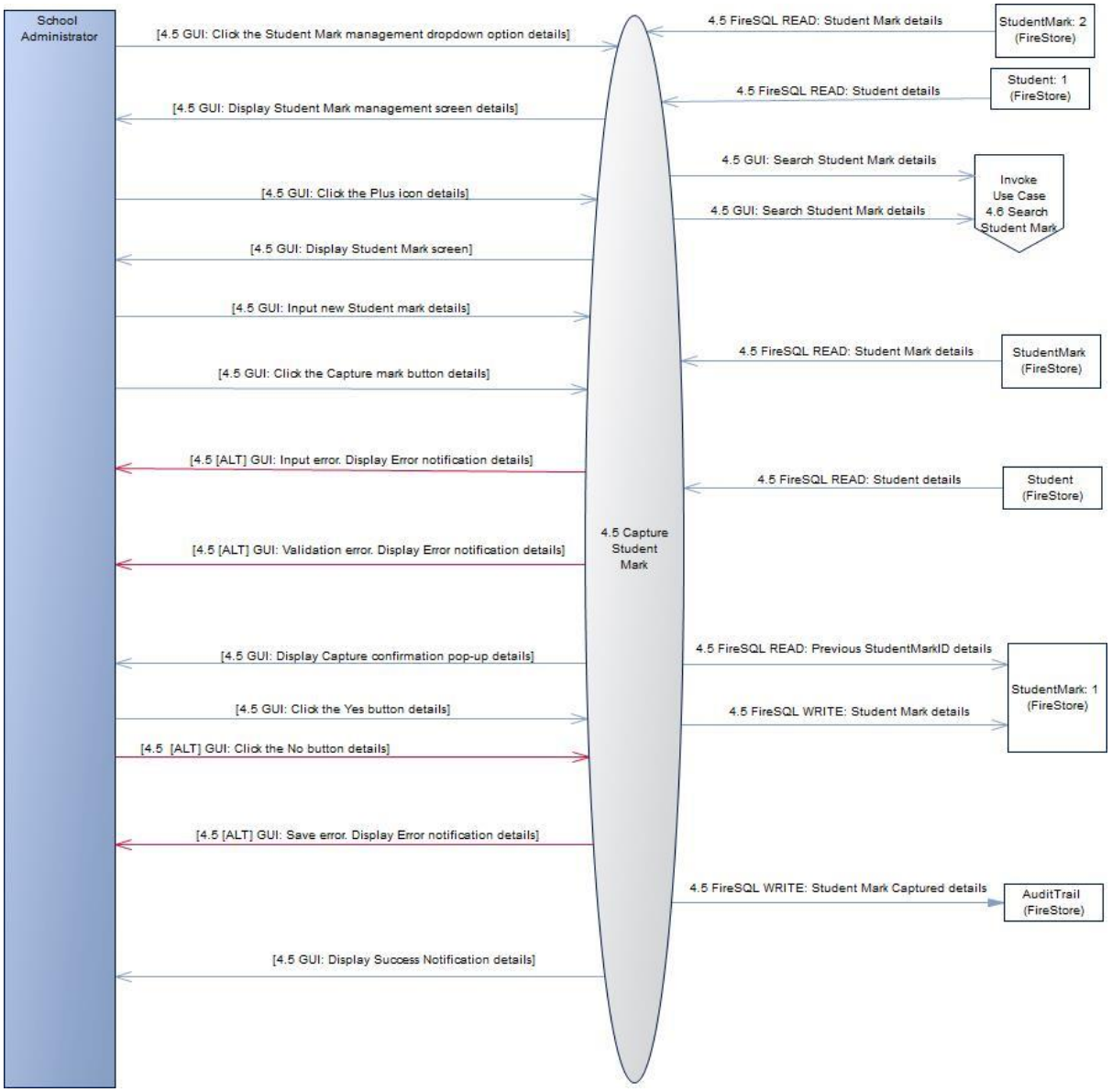


Figure 124 Middle-Level Diagram Sub-System 4: 4.5 Capture Student Mark

4.6 Search Student Mark

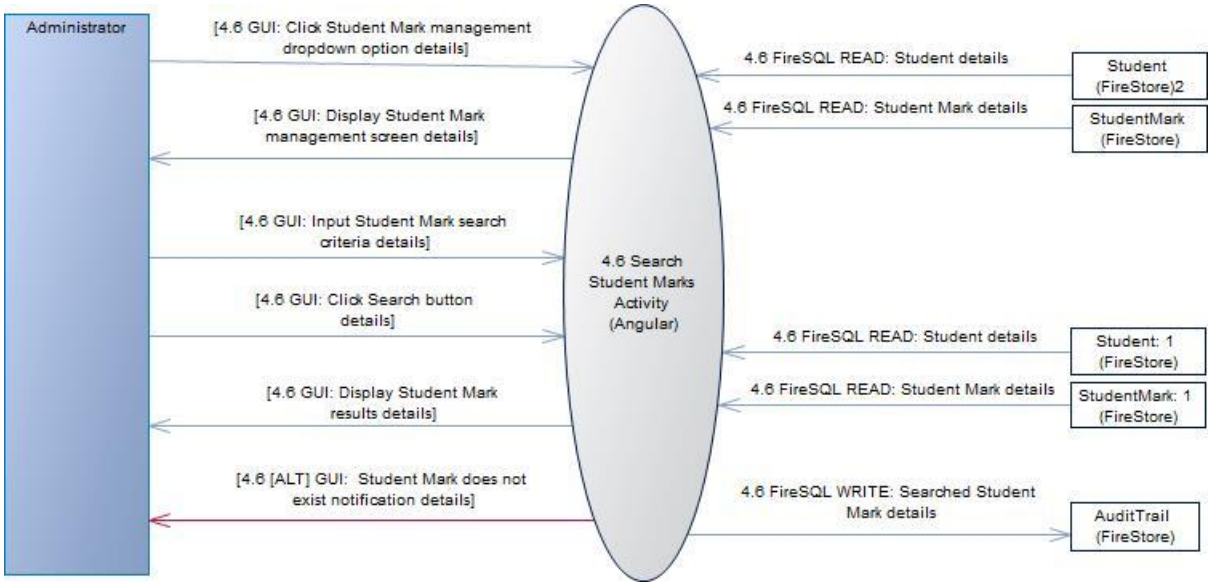


Figure 125 Middle-Level Diagram Sub-System 4: 4.6 Search Student Mark

4.7 Add Mark Type

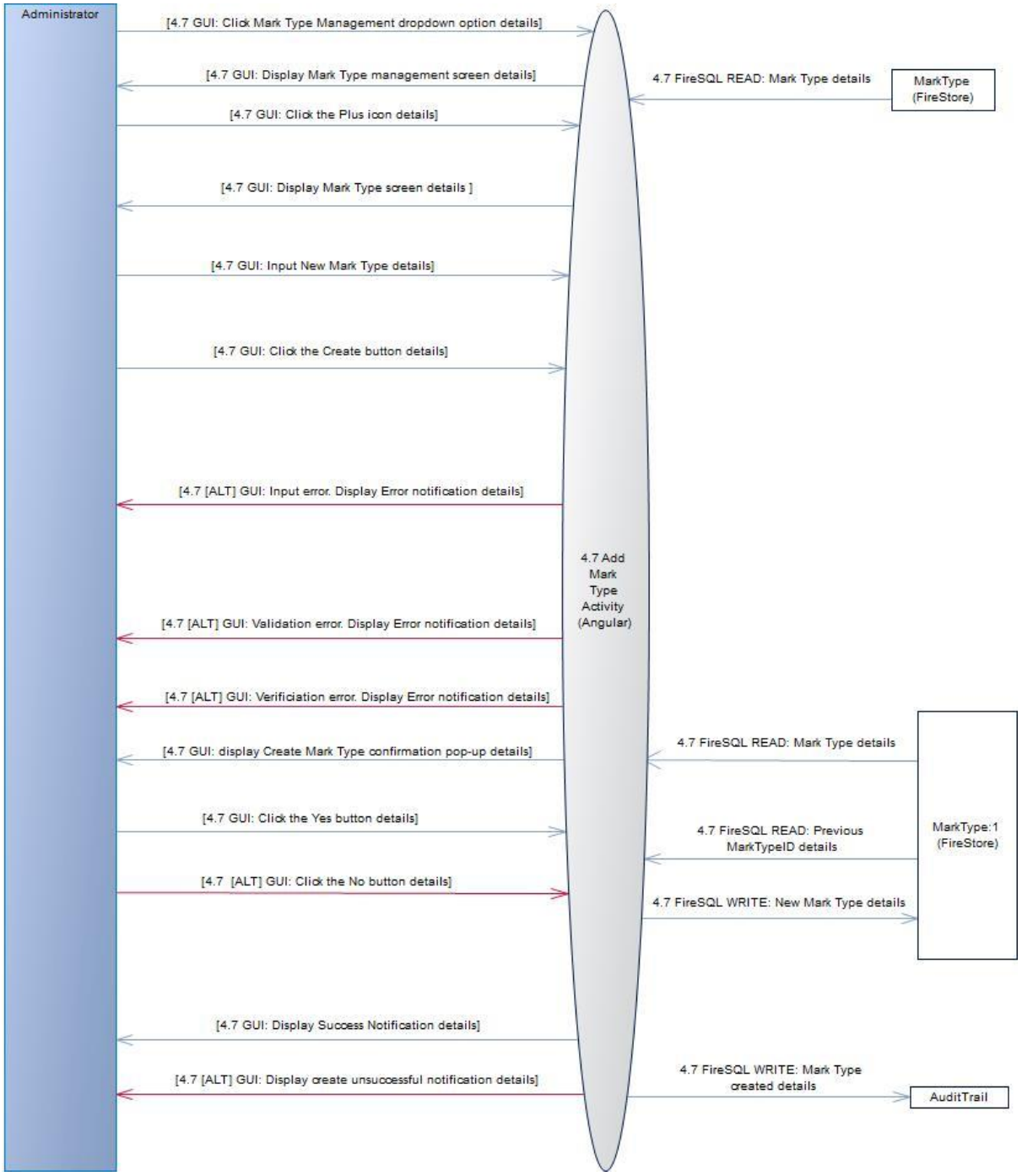


Figure 126 Middle-Level Diagram Sub-System 4: 4.7 Add Mark Type

4.8 Search Mark Type

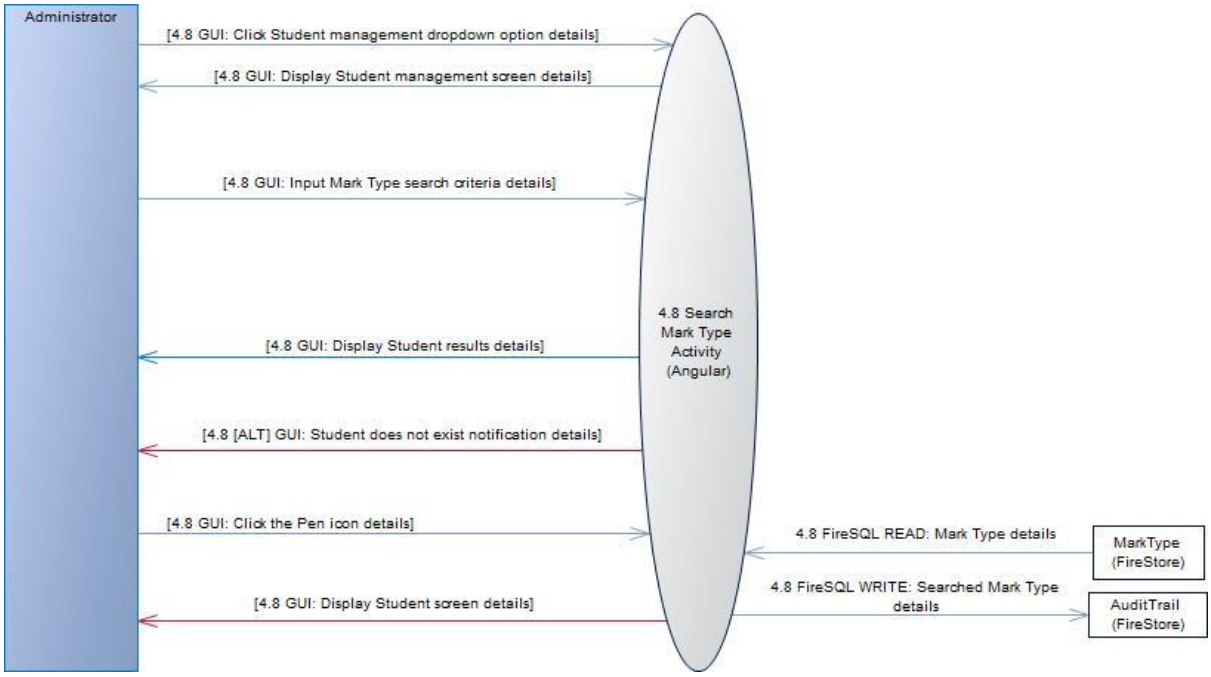


Figure 127 Middle-Level Diagram Sub-System 4: 4.8 Search Mark Type

4.9 Update Mark Type

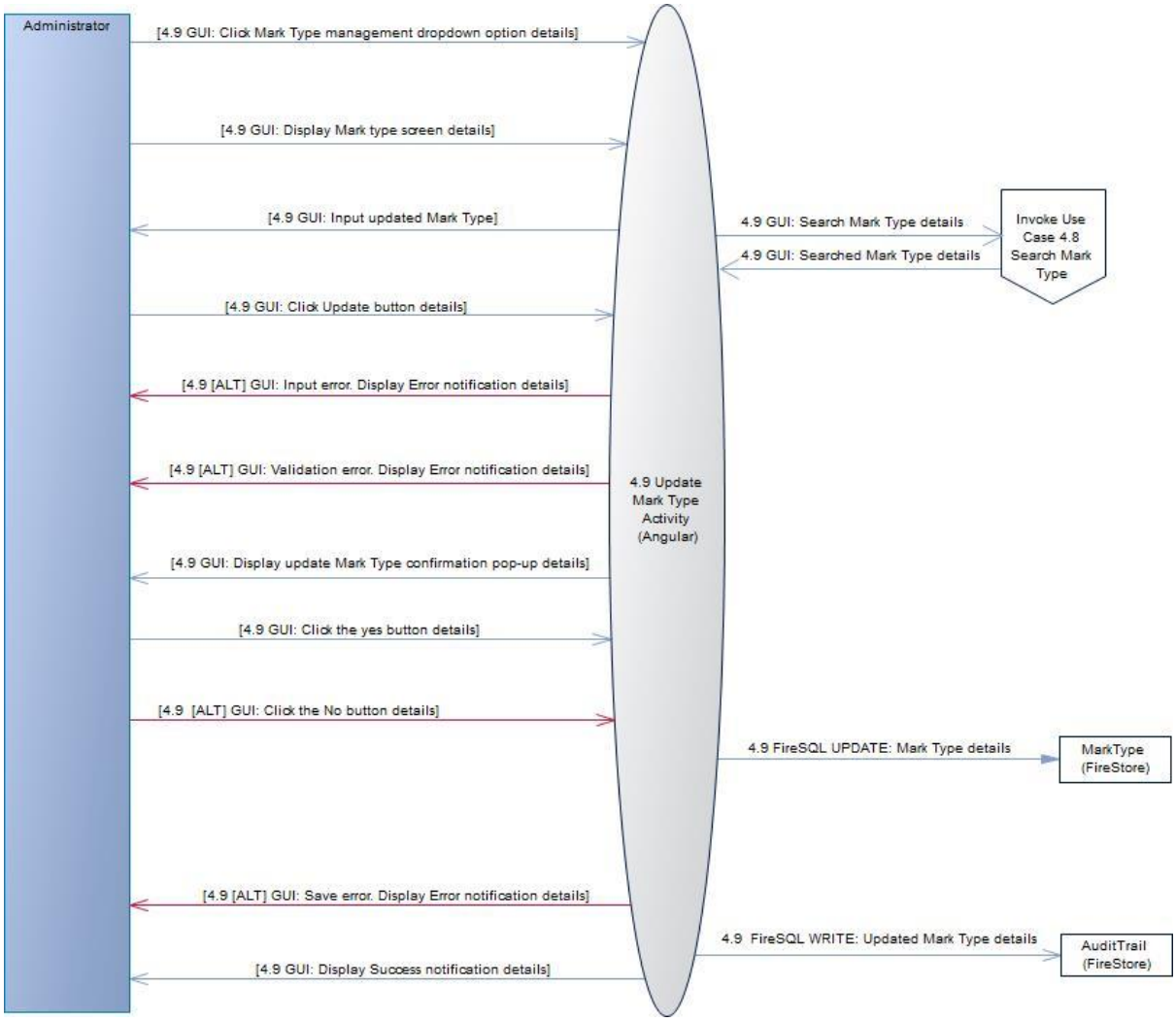


Figure 128 Middle-Level Diagram Sub-System 4: 4.9 Update Mark Type

4.10 Delete Mark Type

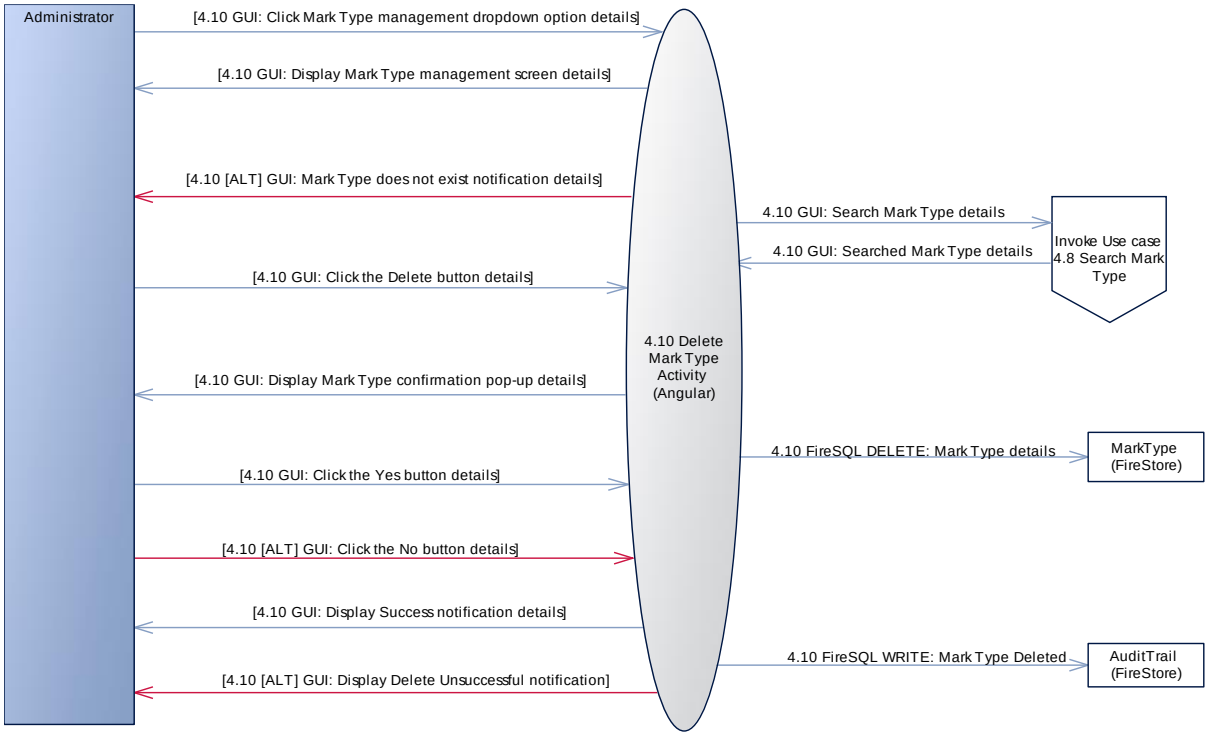


Figure 129 Middle-Level Diagram Sub-System 4: 4.10 Delete Mark Type

Sub-System 5: Bicycle

5.1 Add Bicycle

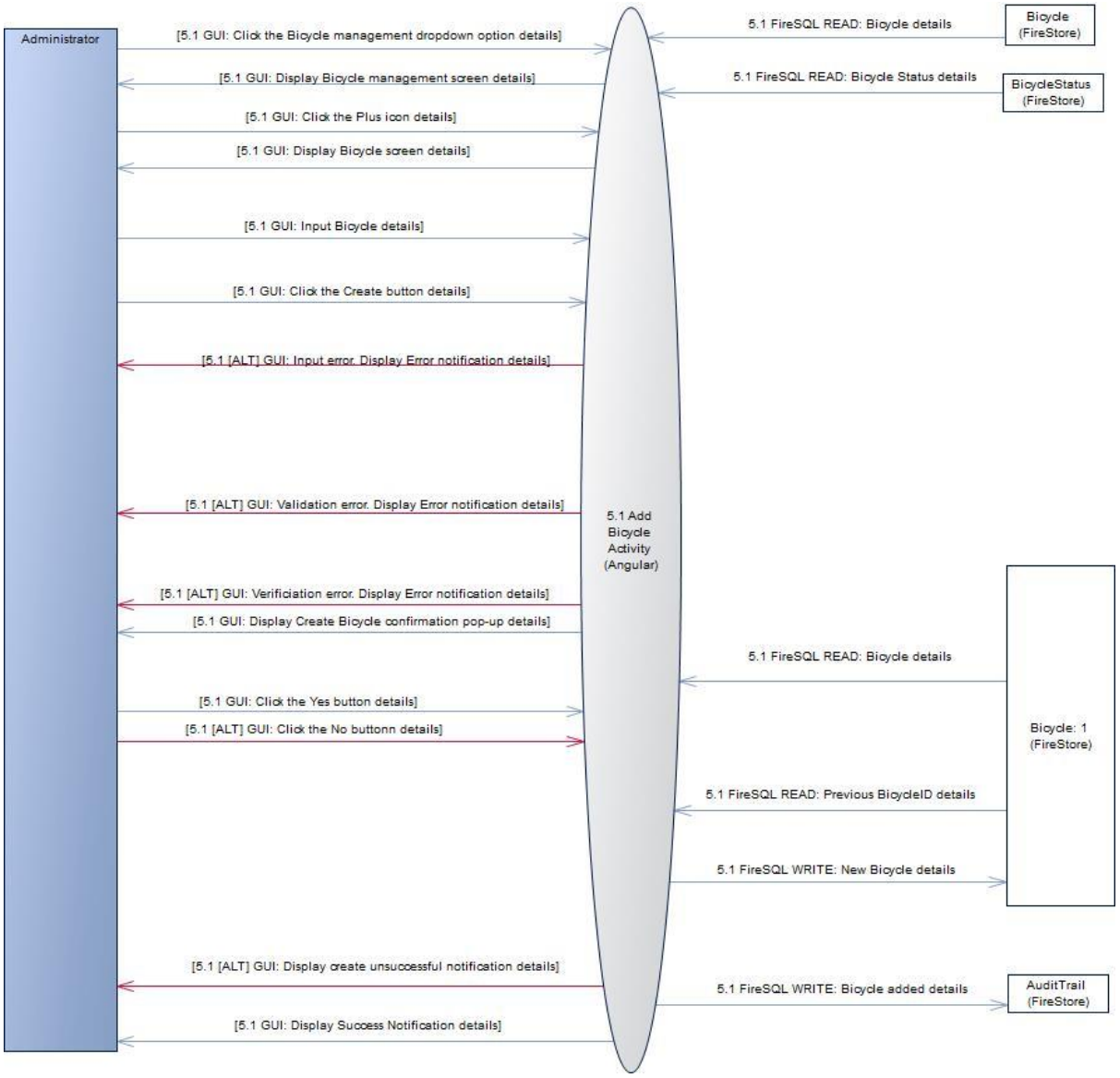


Figure 130 Middle-Level Diagram Sub-System 5: 5.1 Add Bicycle

5.2 Search Bicycle

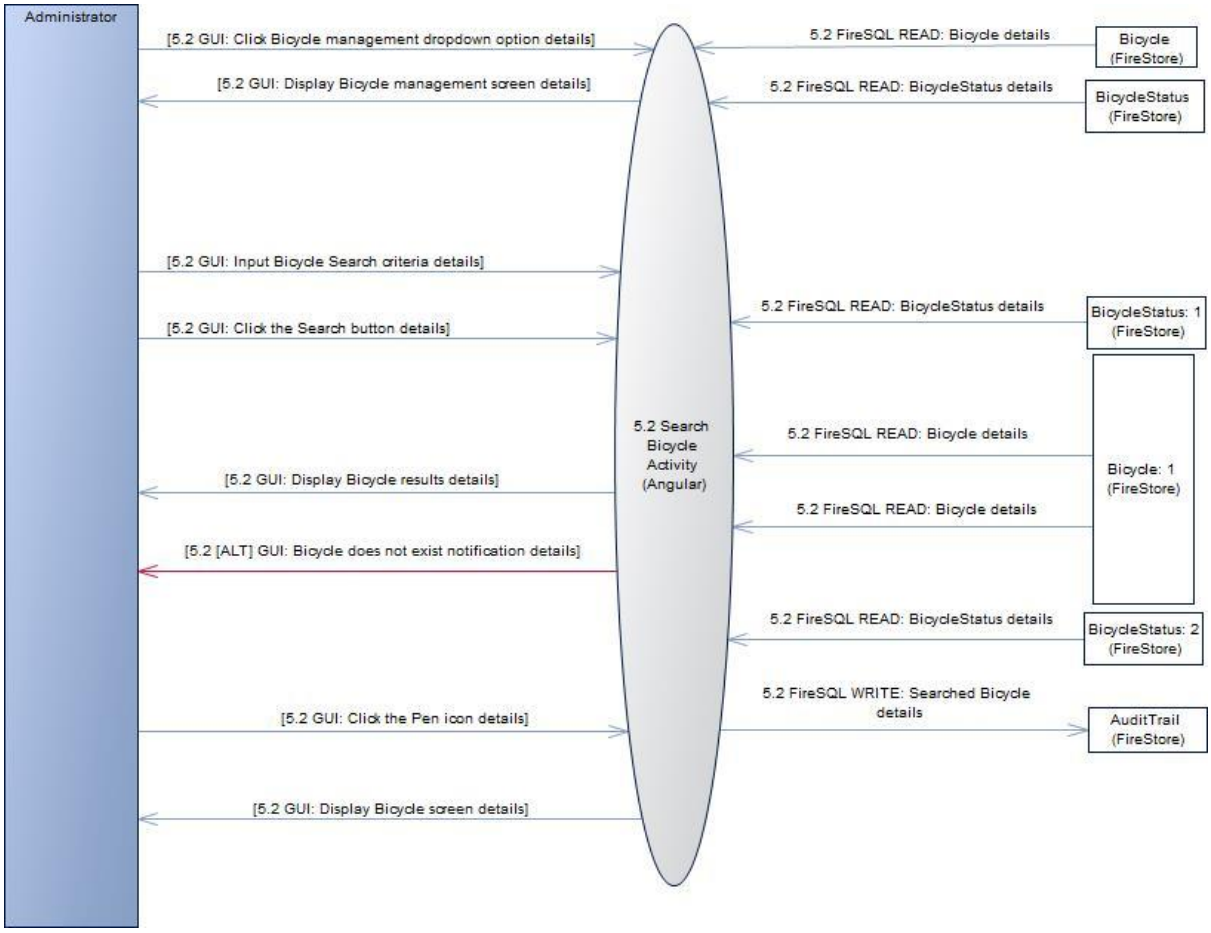


Figure 131 Middle-Level Diagram Sub-System 5: 5.2 Search Bicycle

5.3 Update Bicycle

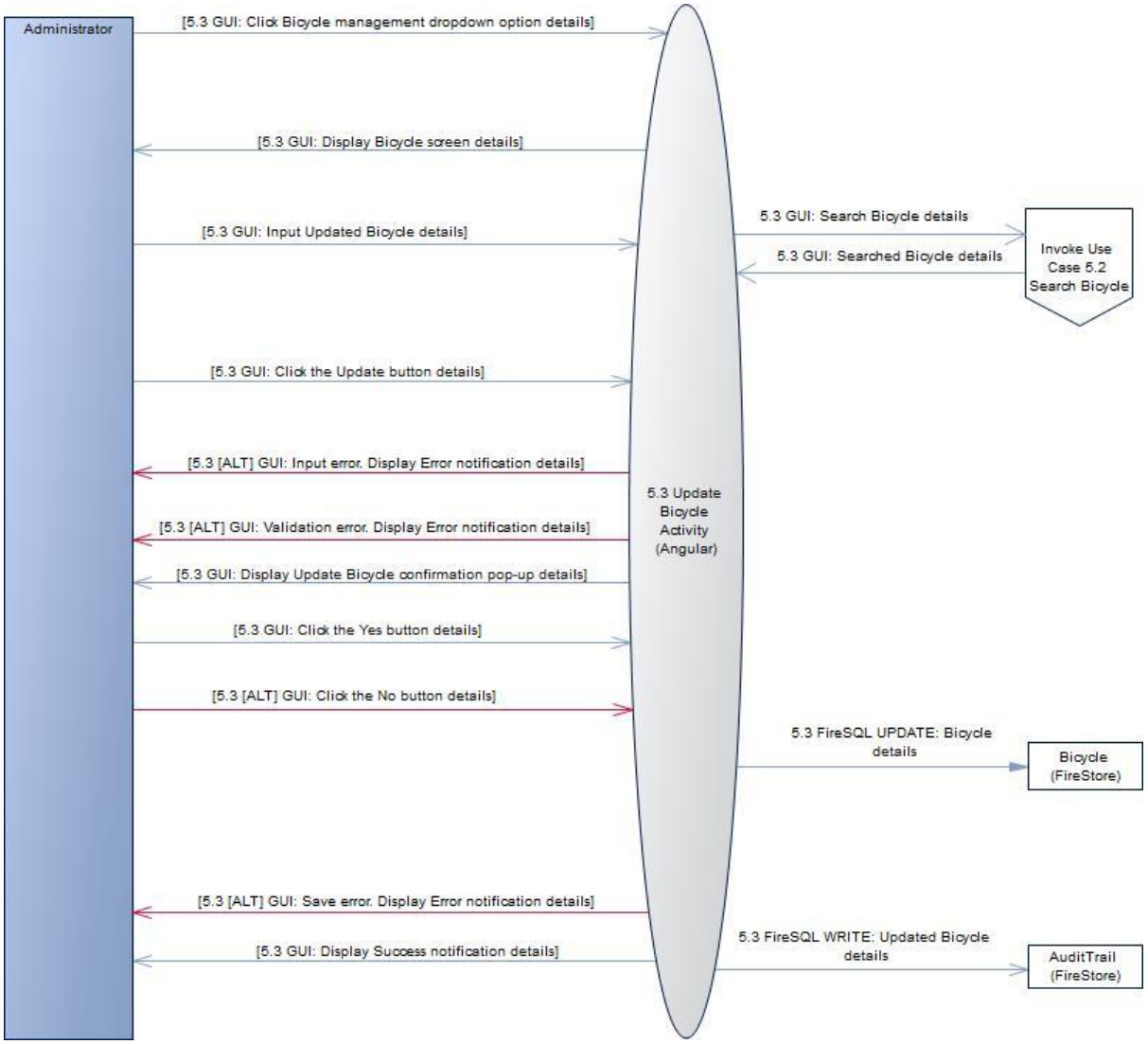


Figure 132 Middle-Level Diagram Sub-System 5: 5.3 update Bicycle

5.5 Delete Bicycle

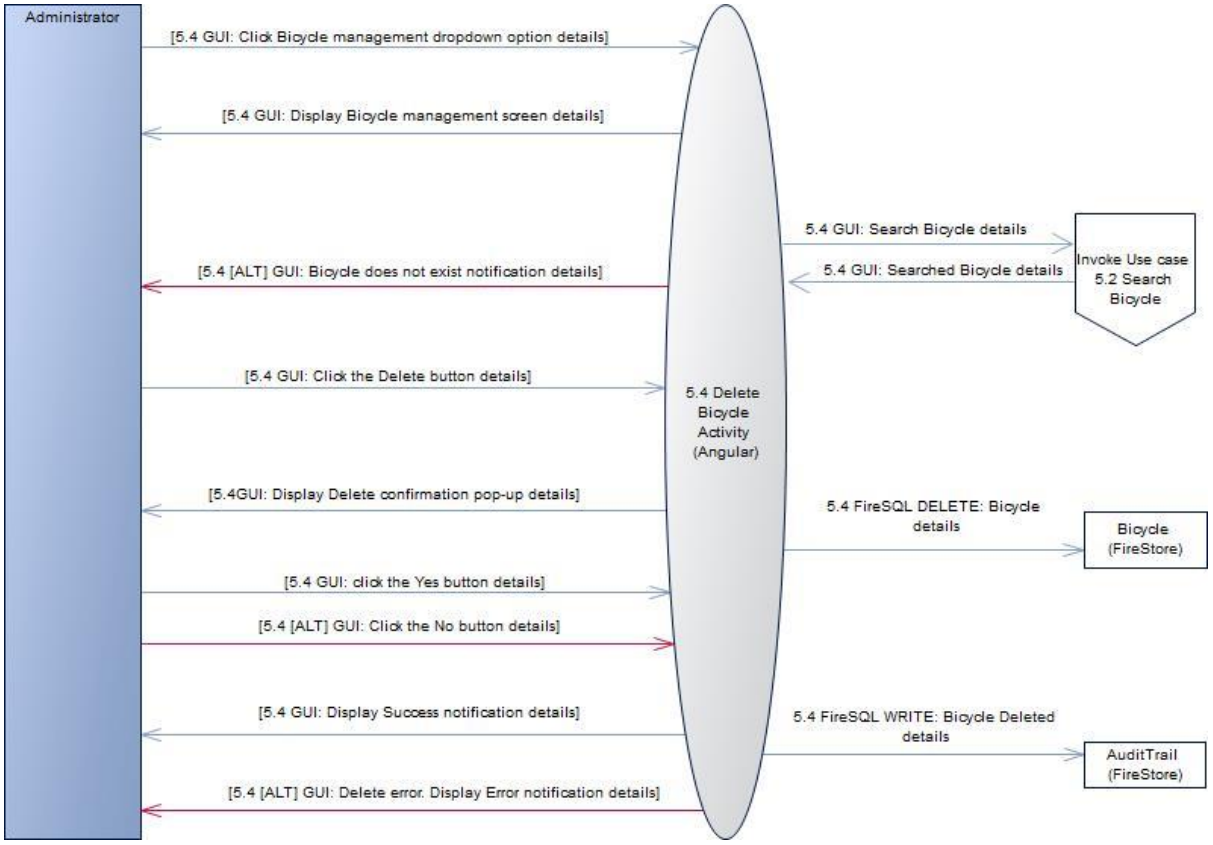


Figure 133 Middle-Level Diagram Sub-System 5: 5.5 Delete Bicycle

5.6 Assign Bicycle to Student

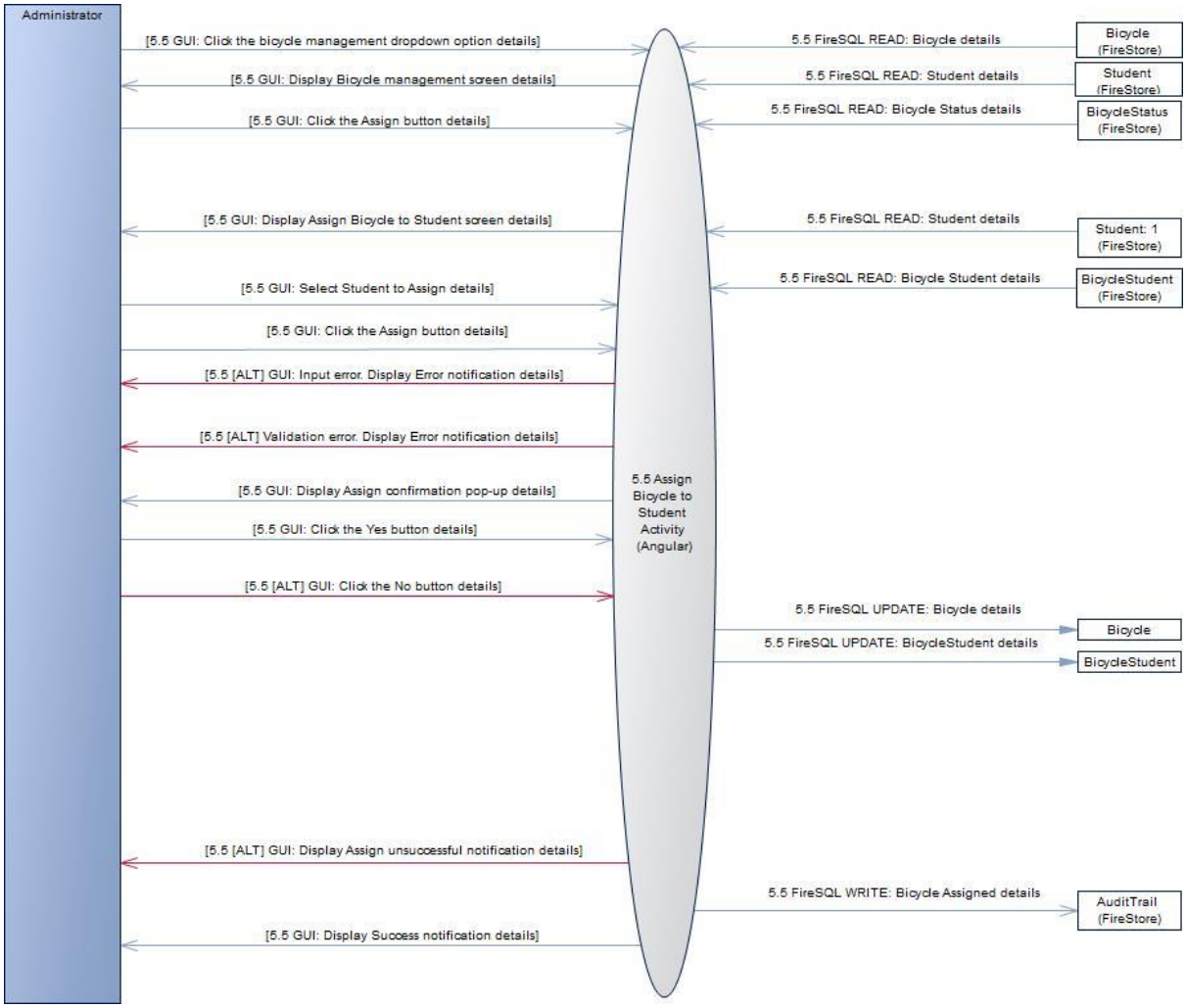


Figure 134 Middle-Level Diagram Sub-System 5: 5.6 Assign bicycle to Student

5.7 Unassign Bicycle from Student

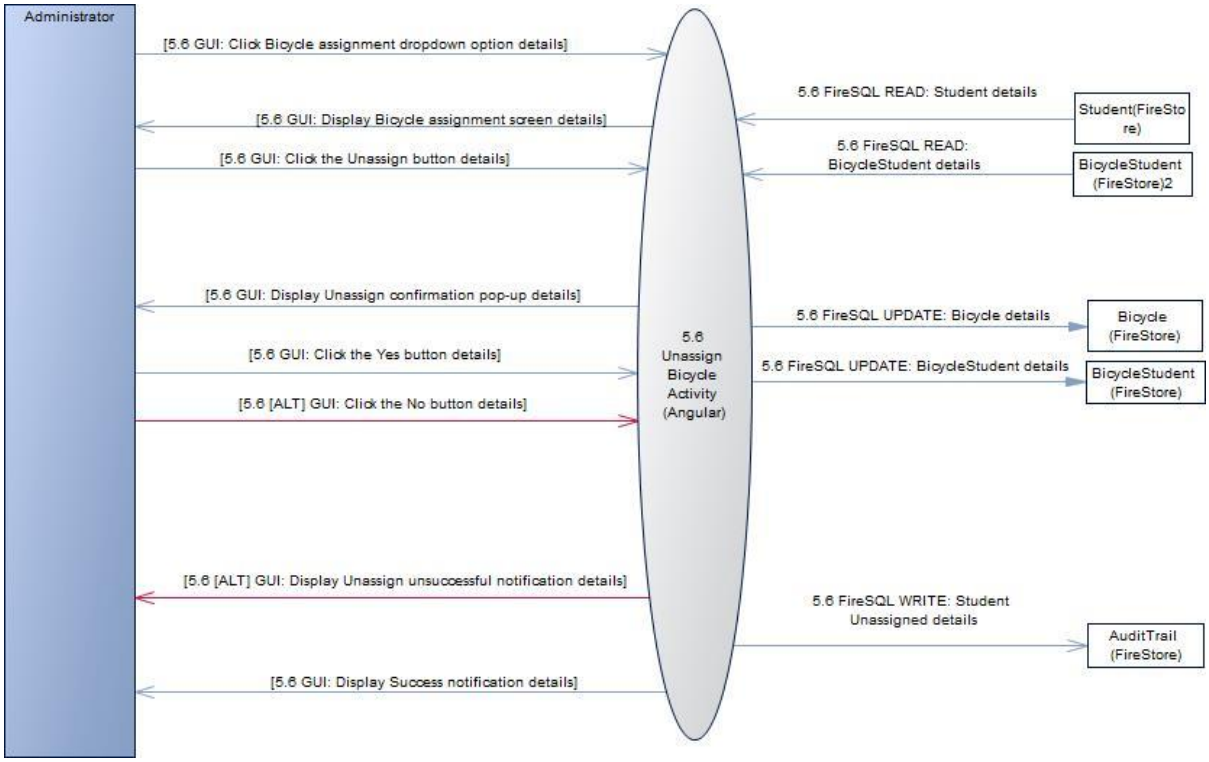


Figure 135 Middle-Level Diagram Sub-System 5: 5.7 unassign Bicycle from Student

5.7 Assign Tag to Bicycle

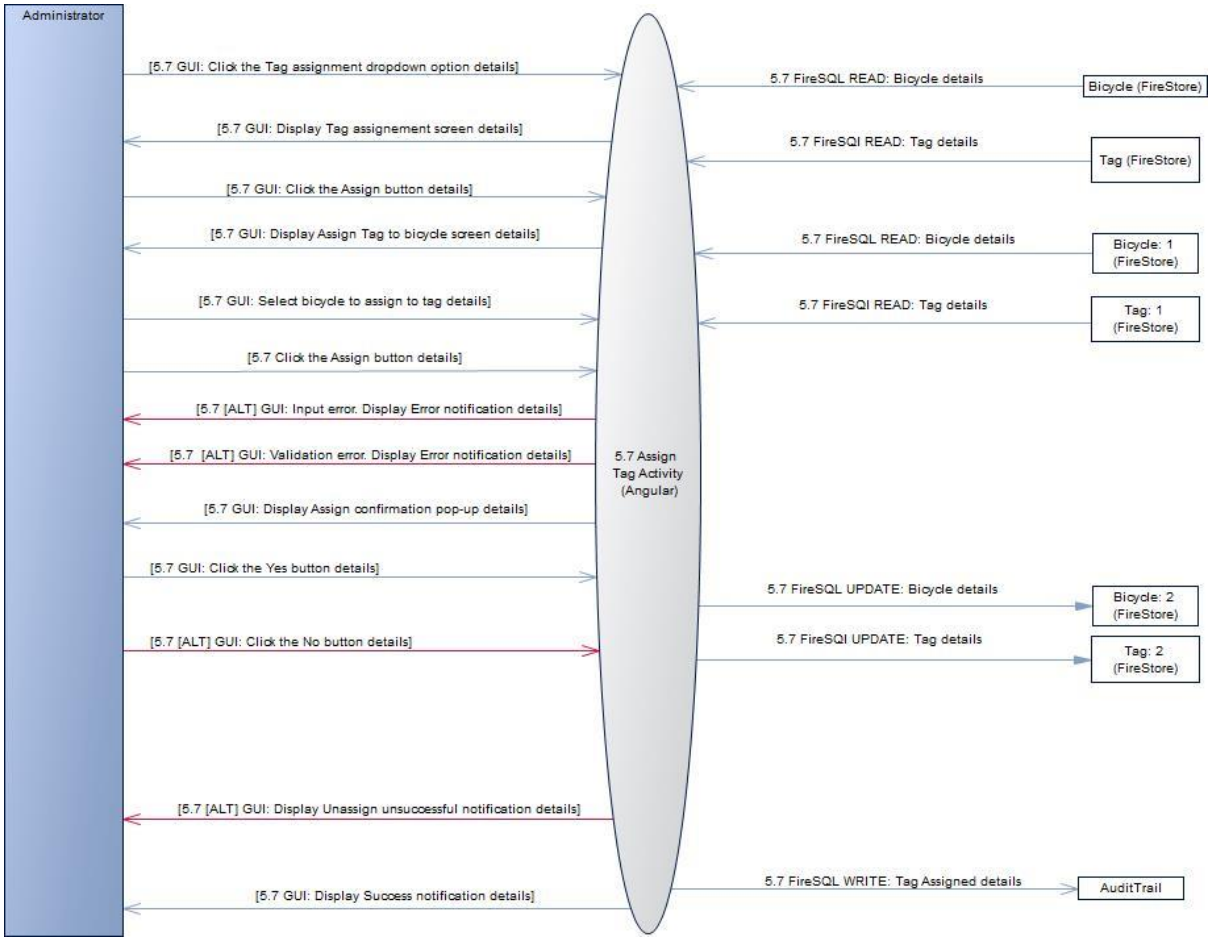


Figure 136 Middle-Level Diagram Sub-System 5: 5.7 Assign Tag to Bicycle

5.8 Unassign Tag from Bicycle

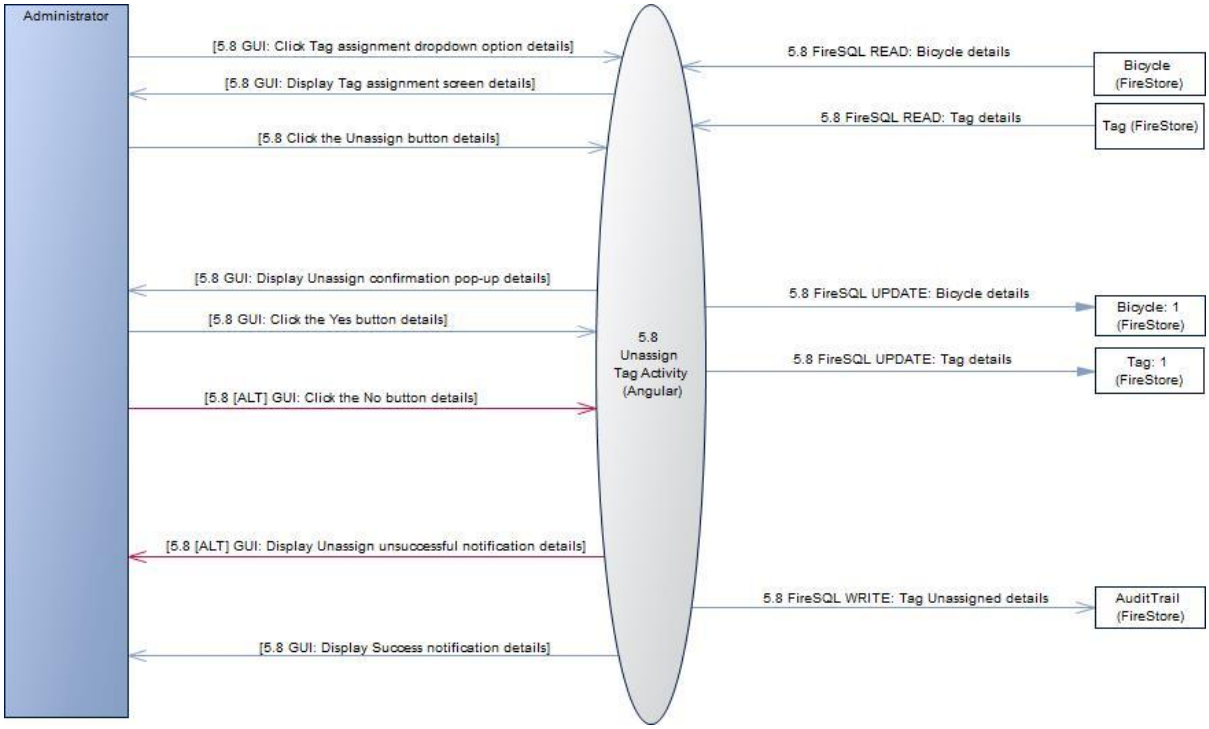


Figure 137 Middle-Level Diagram Sub-System 5: 5.8 Unassign Tag from Bicycle

Sub-System 6: Bicycle Part

6.1 Add Bicycle Part

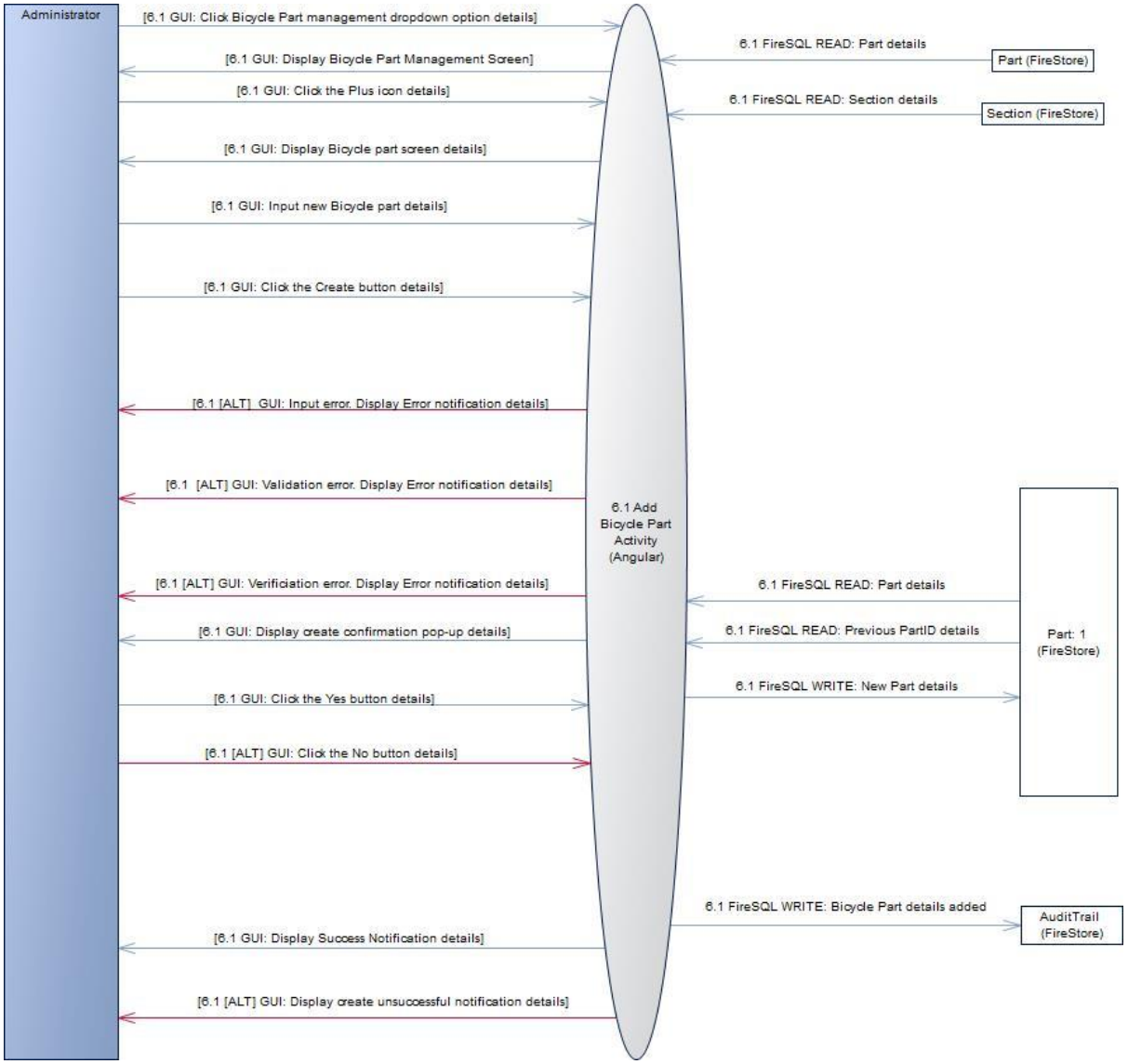


Figure 138 Middle-Level Diagram Sub-System 6: 6.1 Add Bicycle Part

6.2 Search Bicycle Part

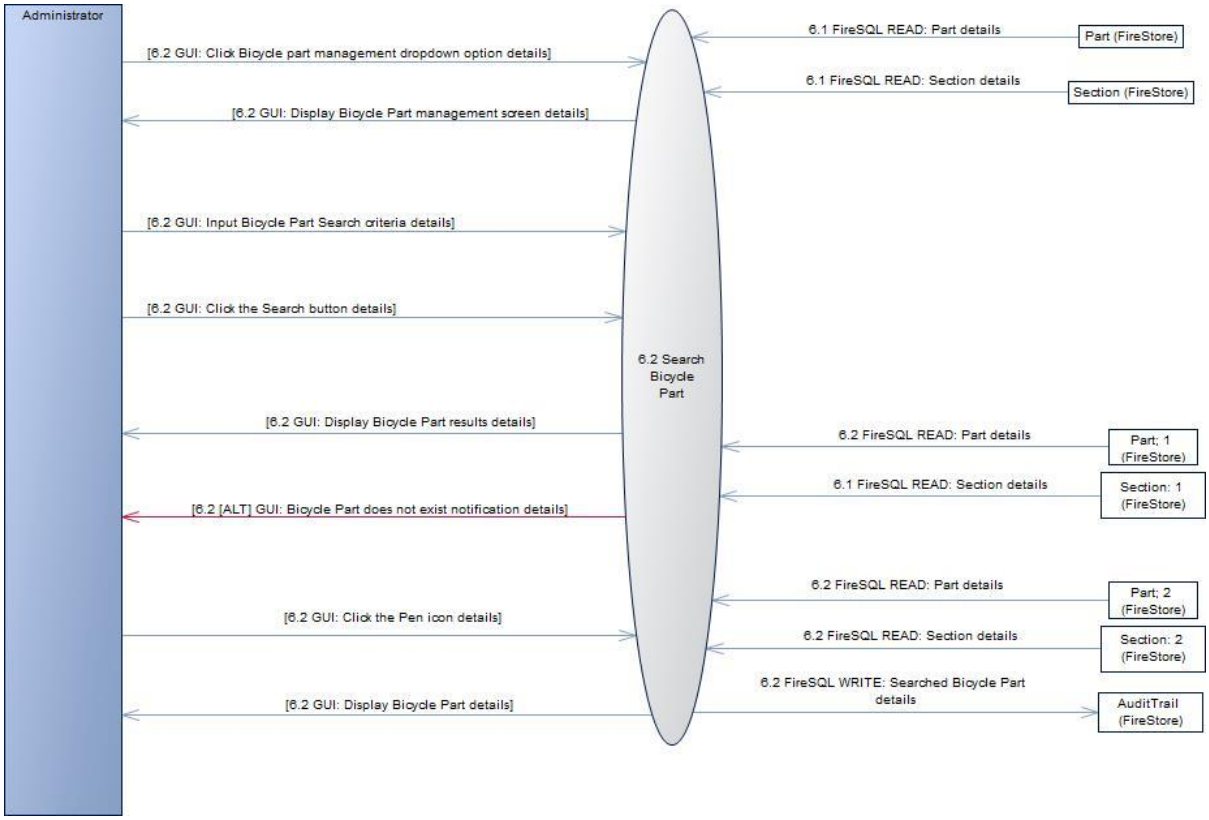


Figure 139 Middle-Level Diagram Sub-System 6: 6.2 Search Bicycle Part

6.3 Update Bicycle Part

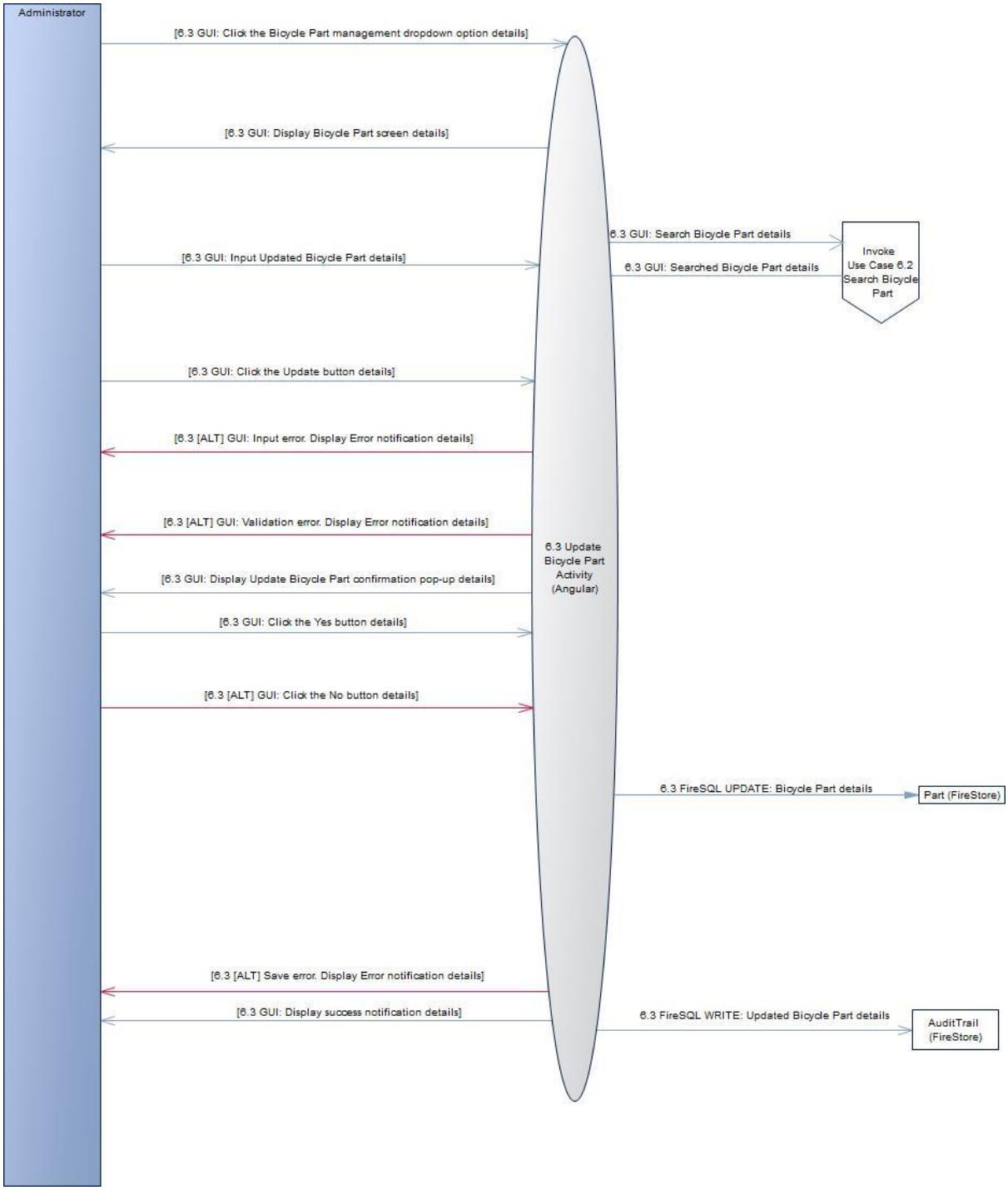


Figure 140 Middle-Level Diagram Sub-System 6: 6.3 Update Bicycle Part

6.4 Delete Bicycle Part

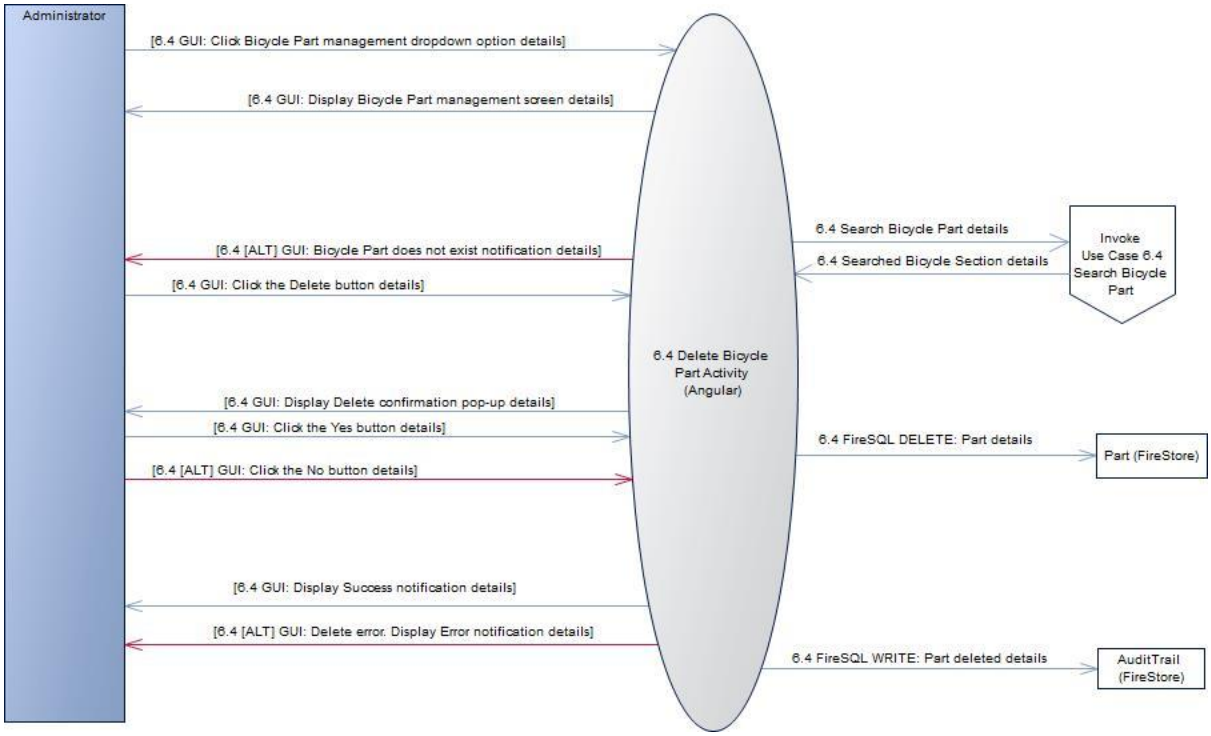


Figure 141 Middle-Level Diagram Sub-System 6: 6.4 Delete Bicycle Part

6.5 Add Bicycle Section

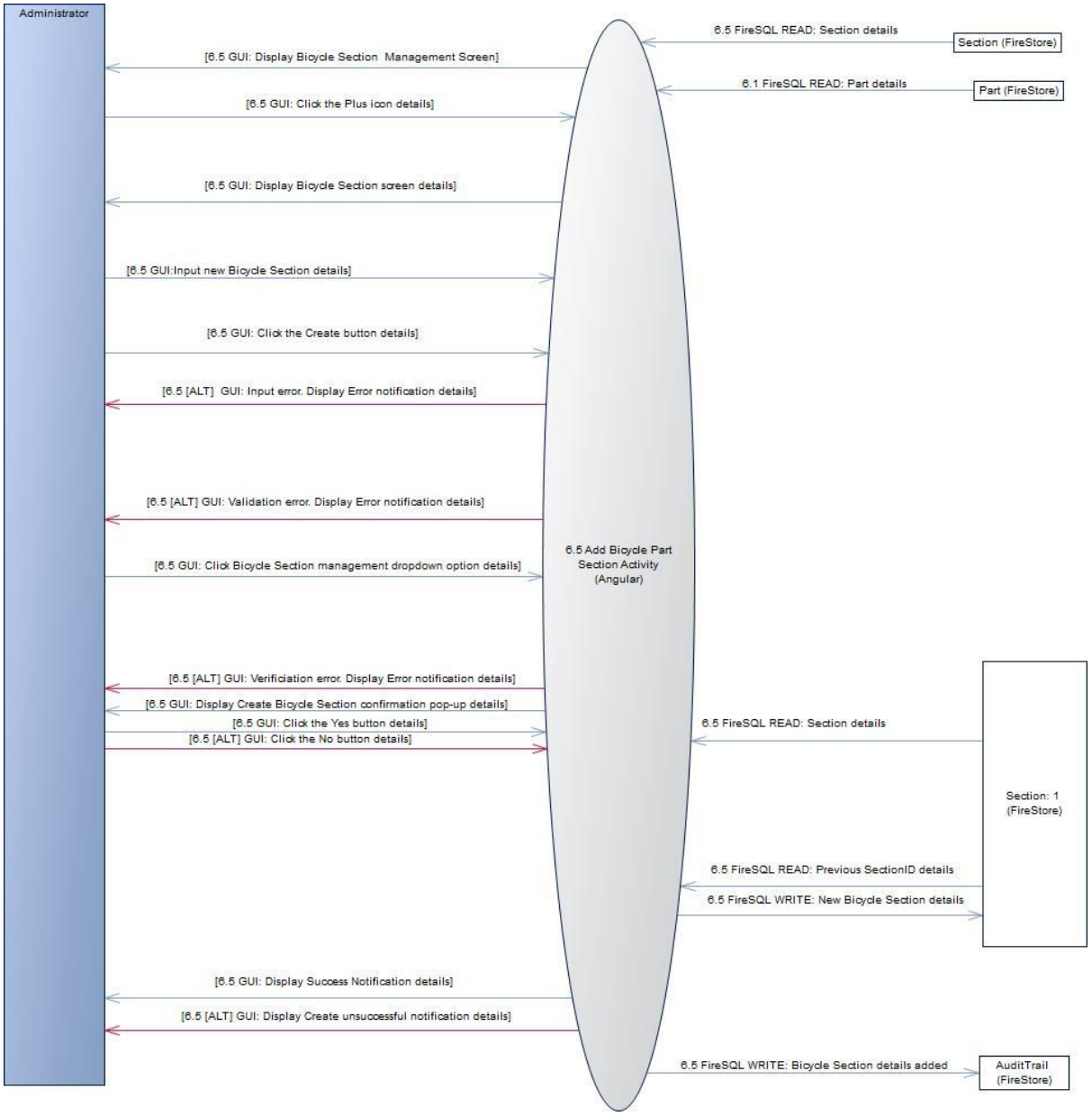


Figure 142 Middle-Level Diagram Sub-System 6: 6.5 Add Bicycle Section

6.6 Search Bicycle Section

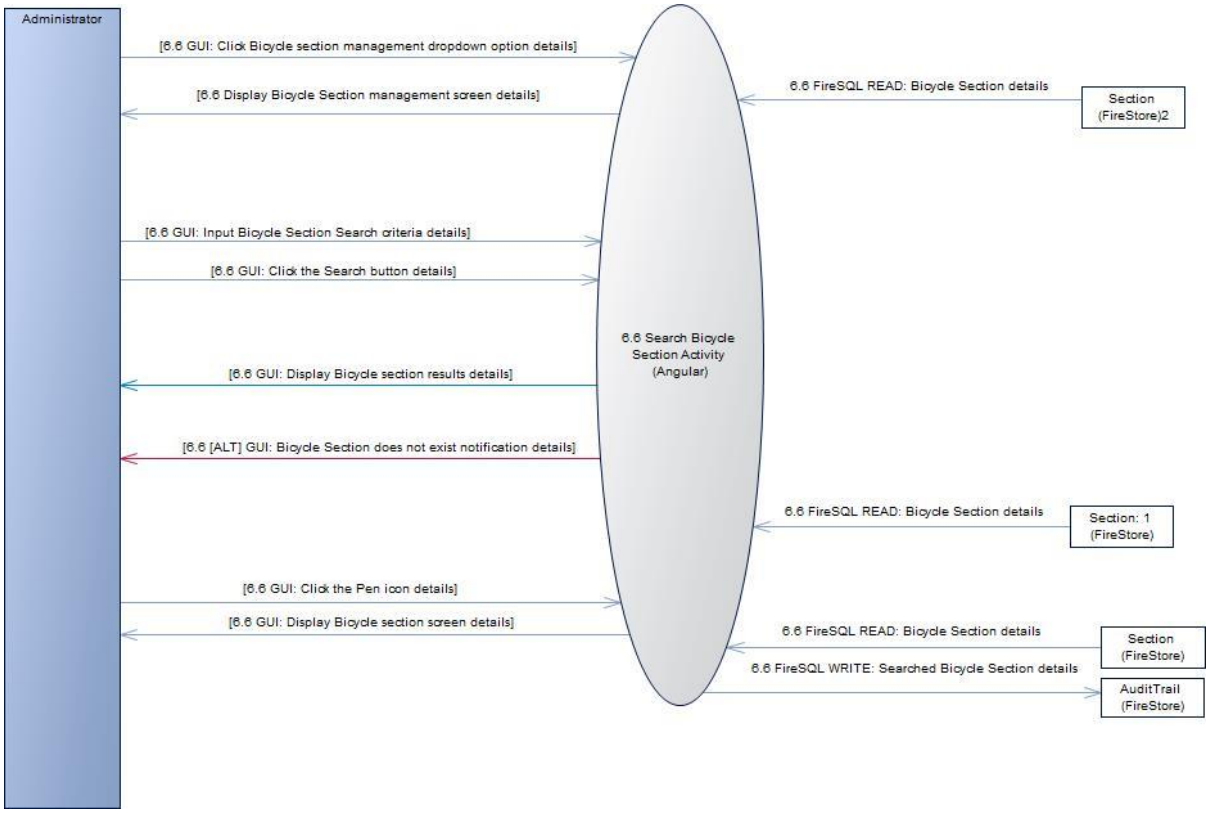


Figure 143 Middle-Level Diagram Sub-System 6: 6.6 Search Bicycle Section

6.7 Update Bicycle Section

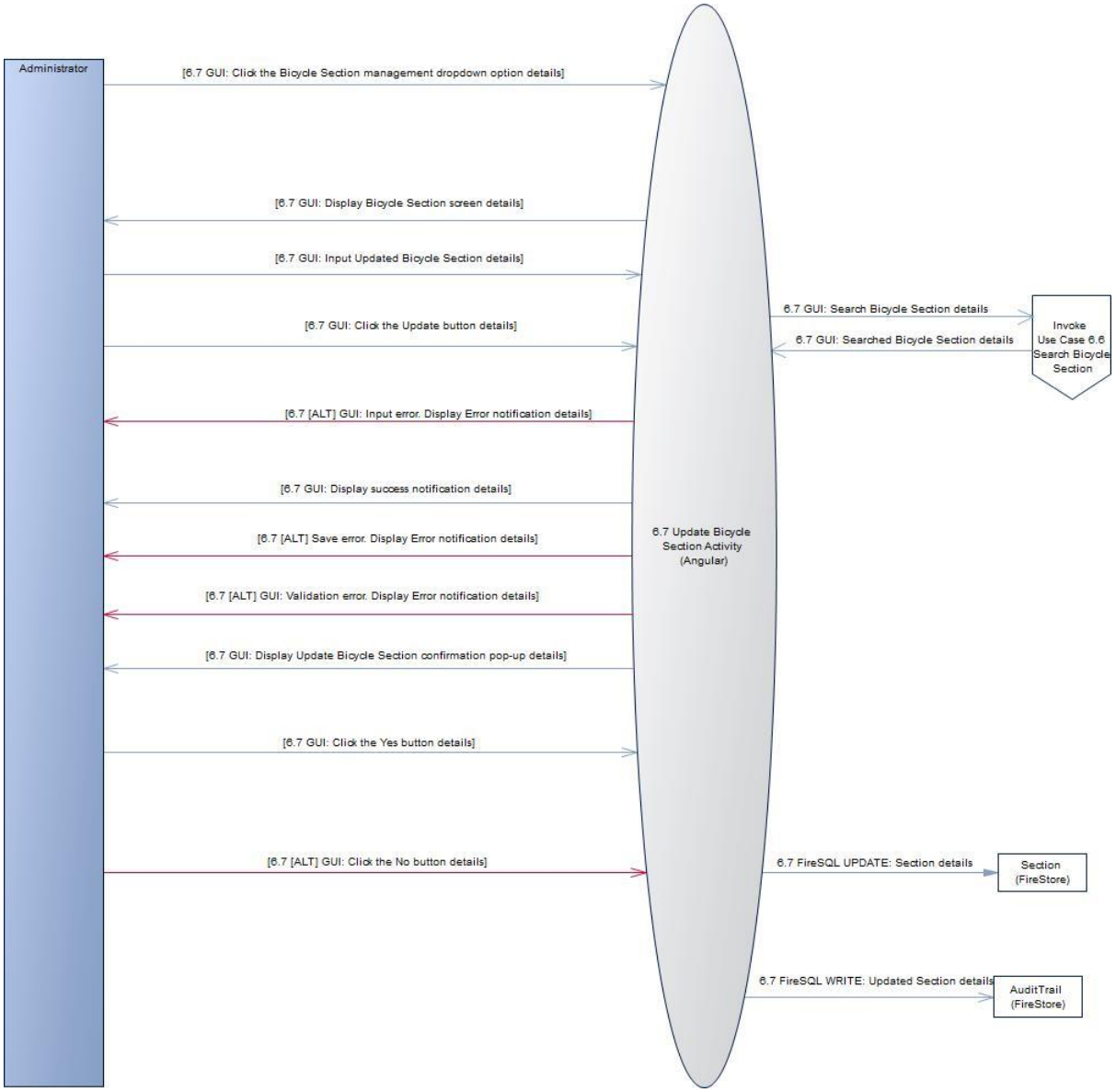


Figure 144 Middle-Level Diagram Sub-System 6: 6.7 Update Bicycle Section

6.8 Delete Bicycle Section

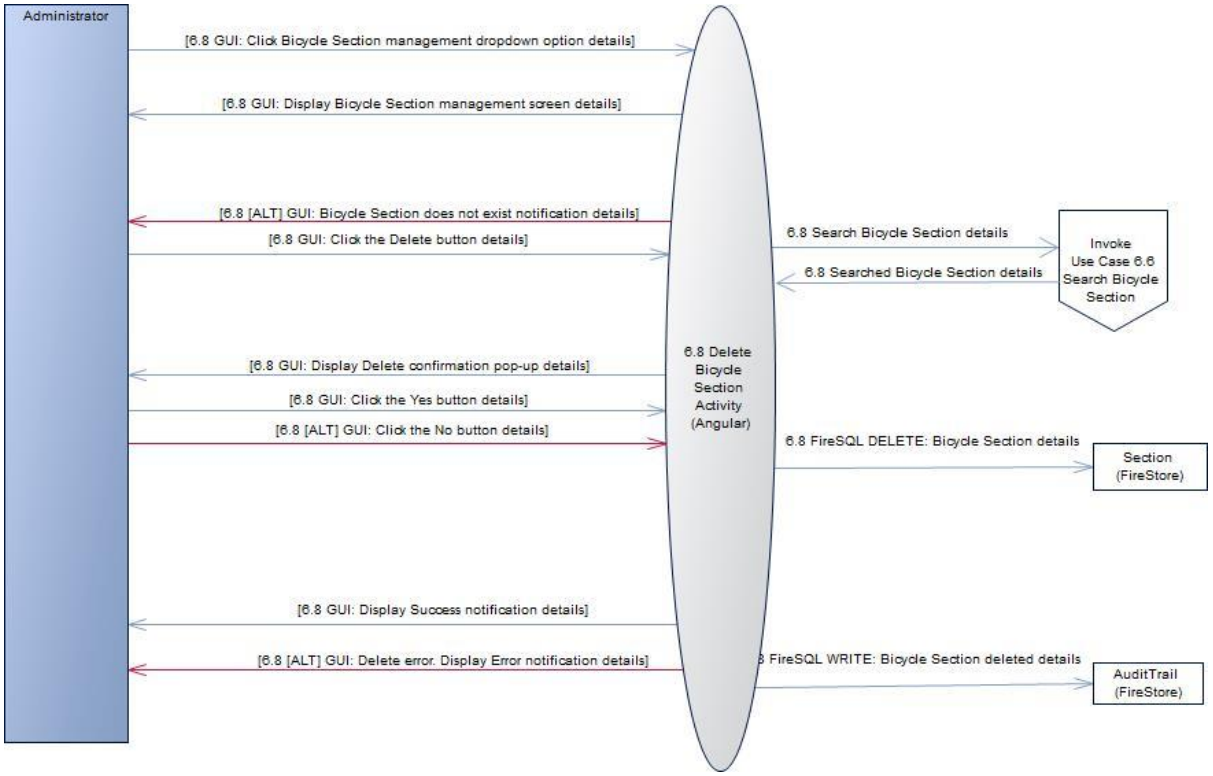


Figure 145 Middle-Level Diagram Sub-System 6: 6.8 Delete Bicycle Section

Sub-System 7: Tracking

7.1 Add Log Entry

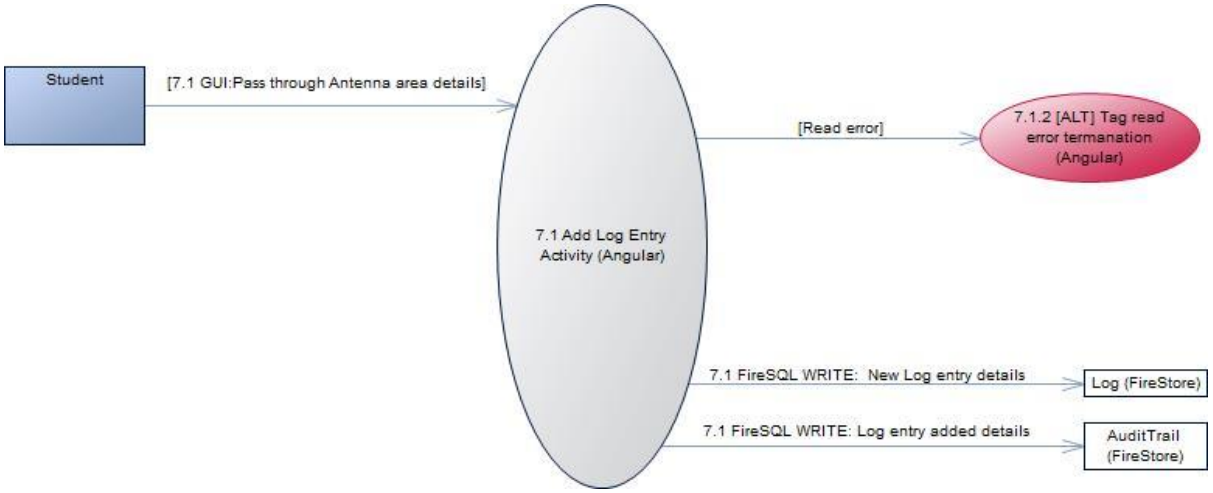


Figure 146 Middle-Level Diagram Sub-System 7: 7.1 Add Log Entry

7.2 Search Log Entry

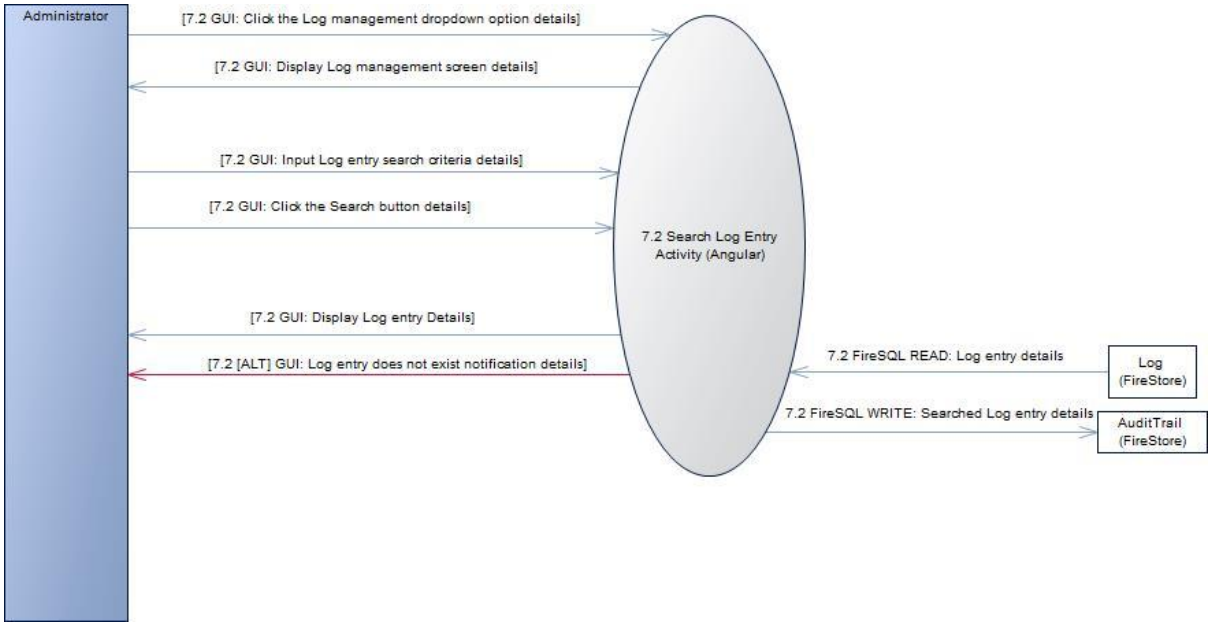


Figure 147 Middle-Level Diagram Sub-System 7: 7.2 Search Log Entry

7.3 Add Route

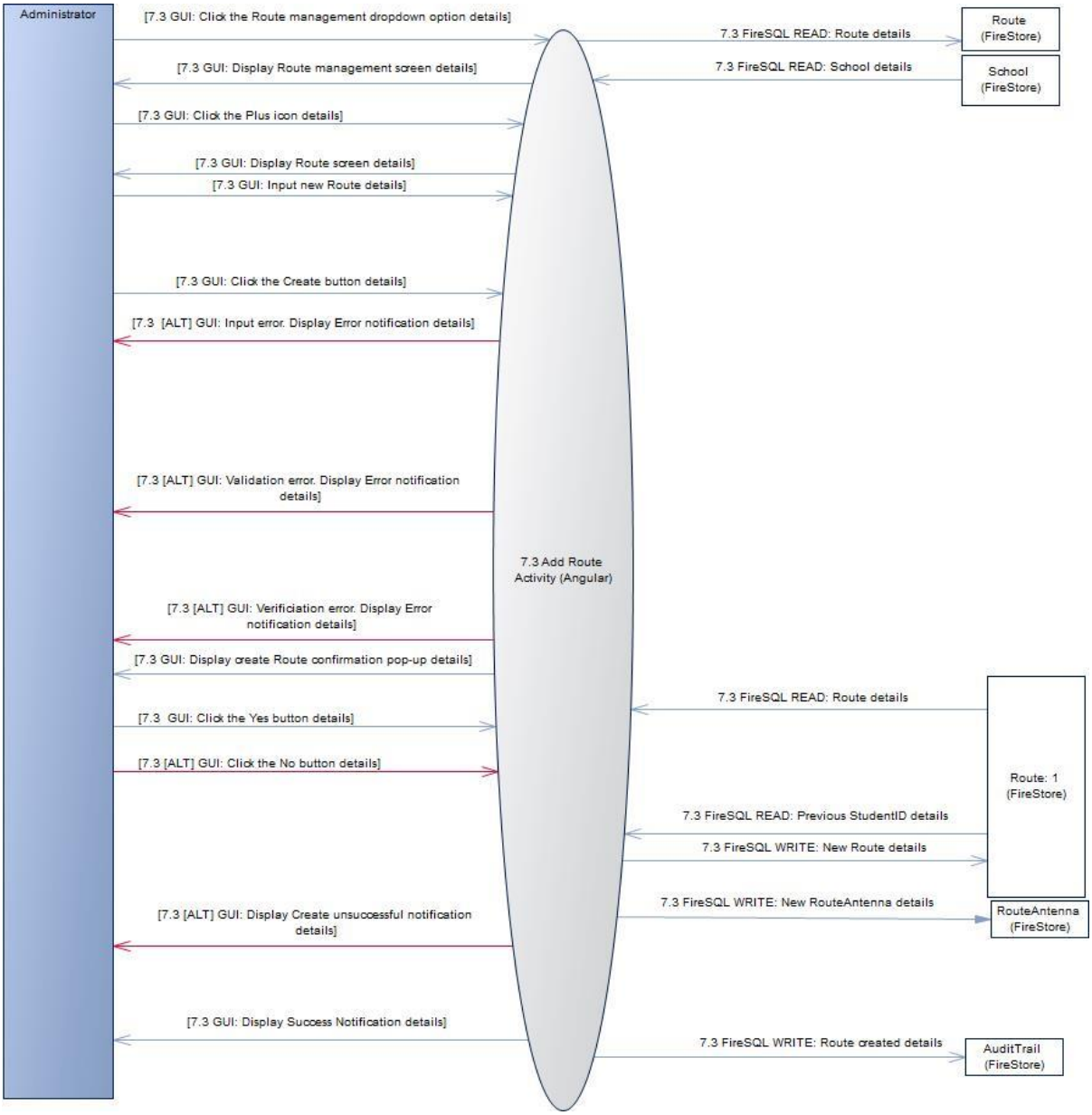


Figure 148 Middle-Level Diagram Sub-System 7: 7.3 Add Route

7.4 Search Route

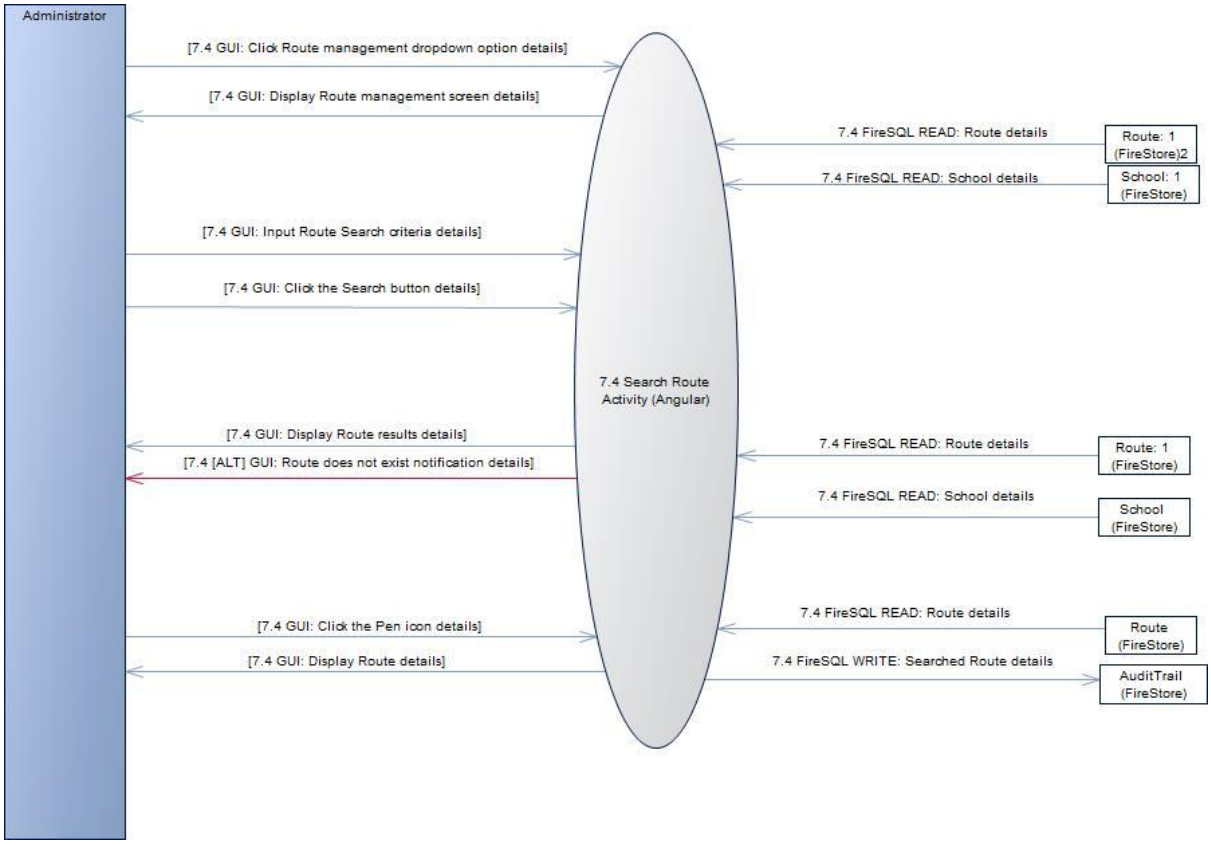


Figure 149 Middle-Level Diagram Sub-System 7: 7.4 Search Route

7.5 Update Route

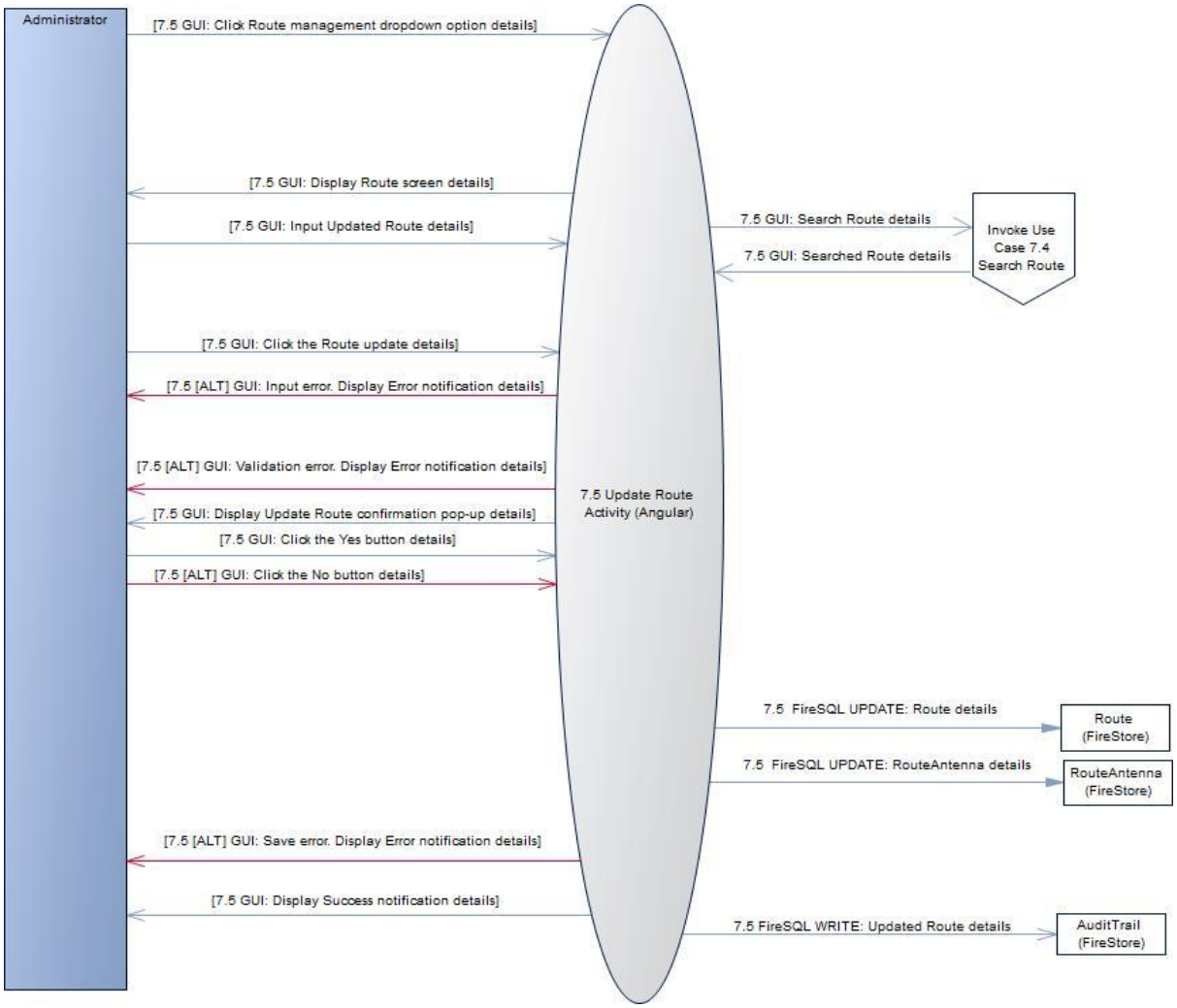


Figure 150 Middle-Level Diagram Sub-System 7: 7.5 Update Route

7.6 Delete Route

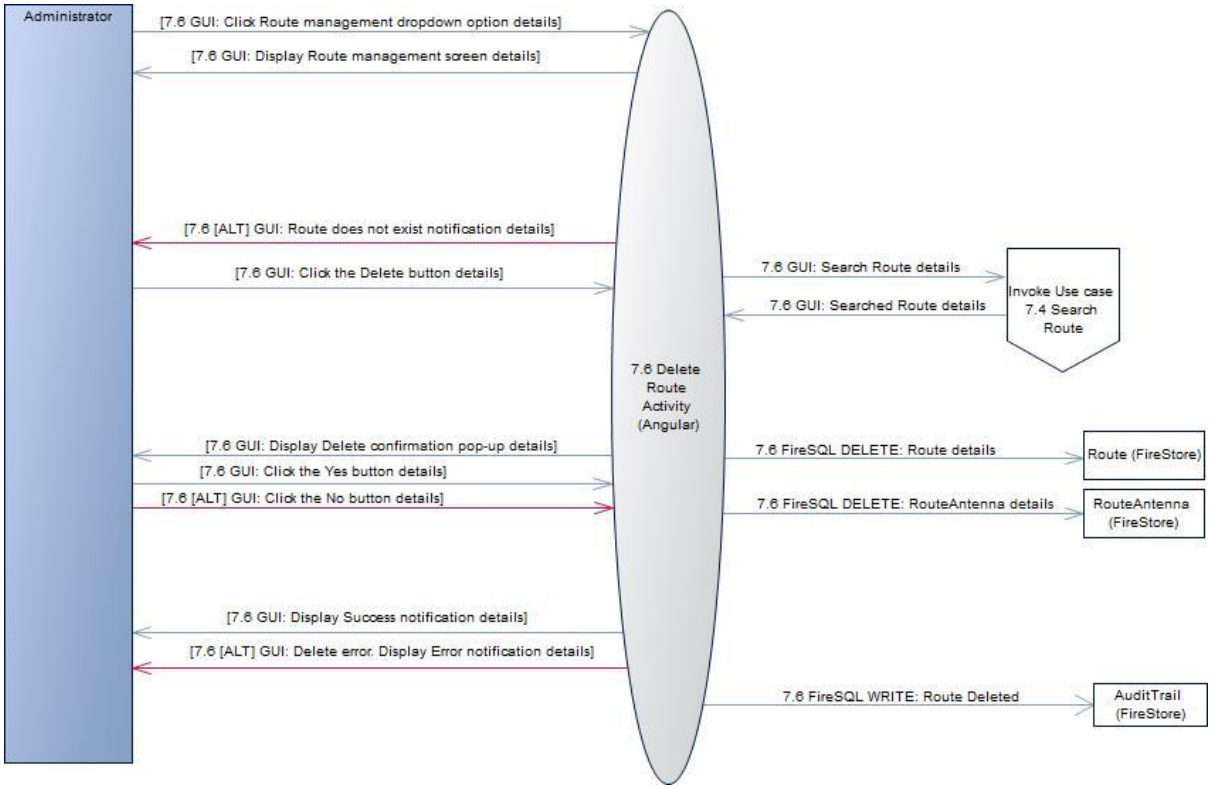


Figure 151 Middle-Level Diagram Sub-System 7: 7.6 Delete Route

Sub-System 8: Job Tasks

8.1 Add Fault Task

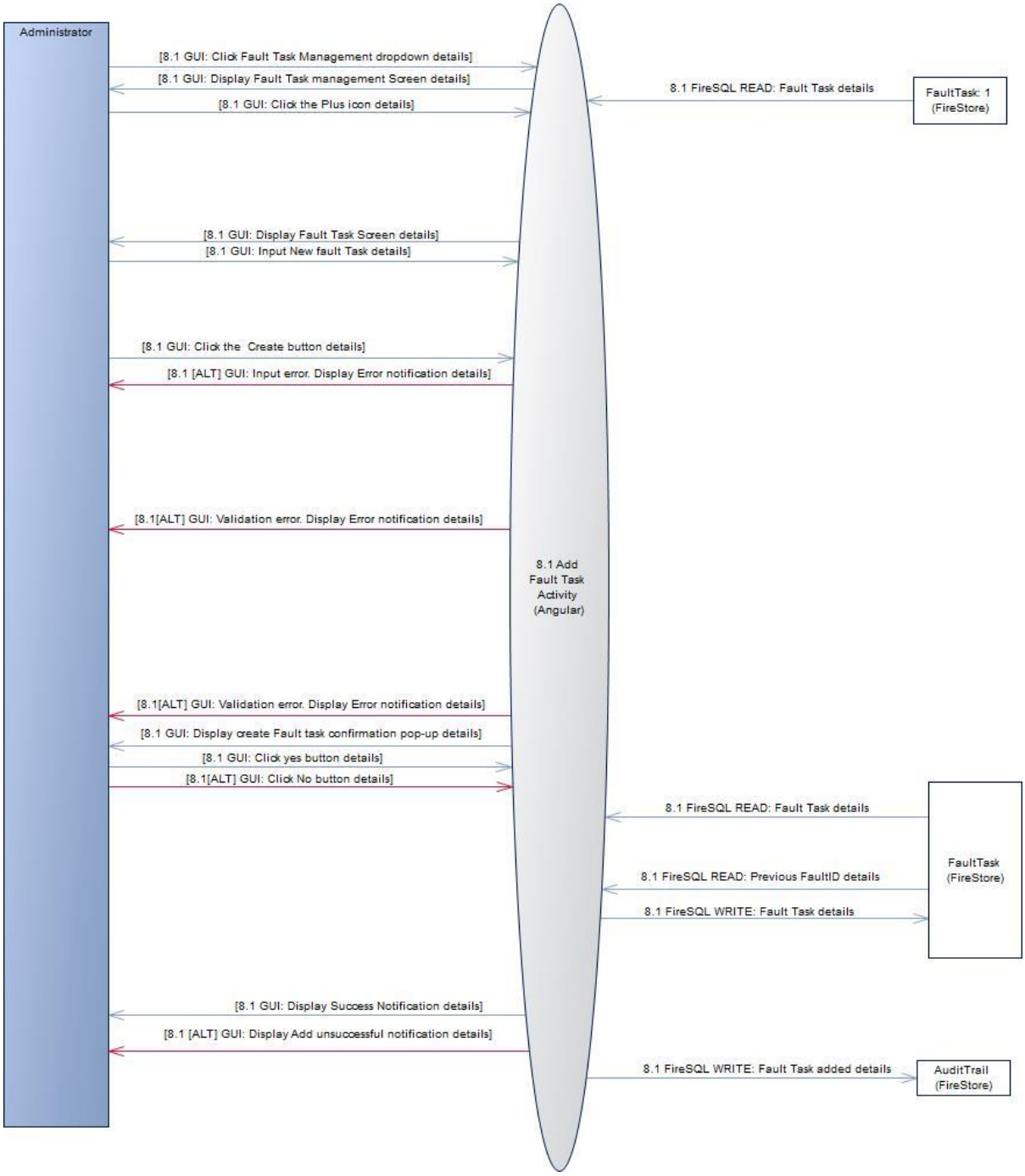


Figure 152 Middle-Level Diagram Sub-System 8: 8.1 Add Fault Task

8.2 Search Fault Task

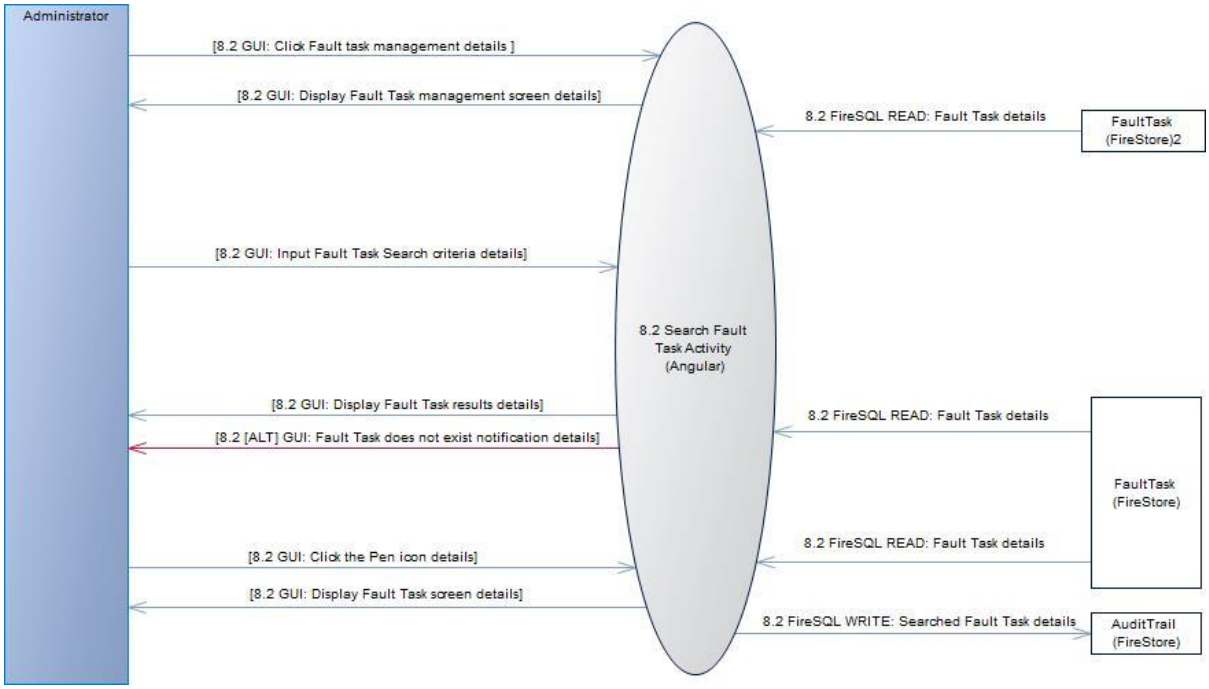


Figure 153 Middle-Level Diagram Sub-System 8: 8.2 Search Fault Task

8.3 Update Fault Task

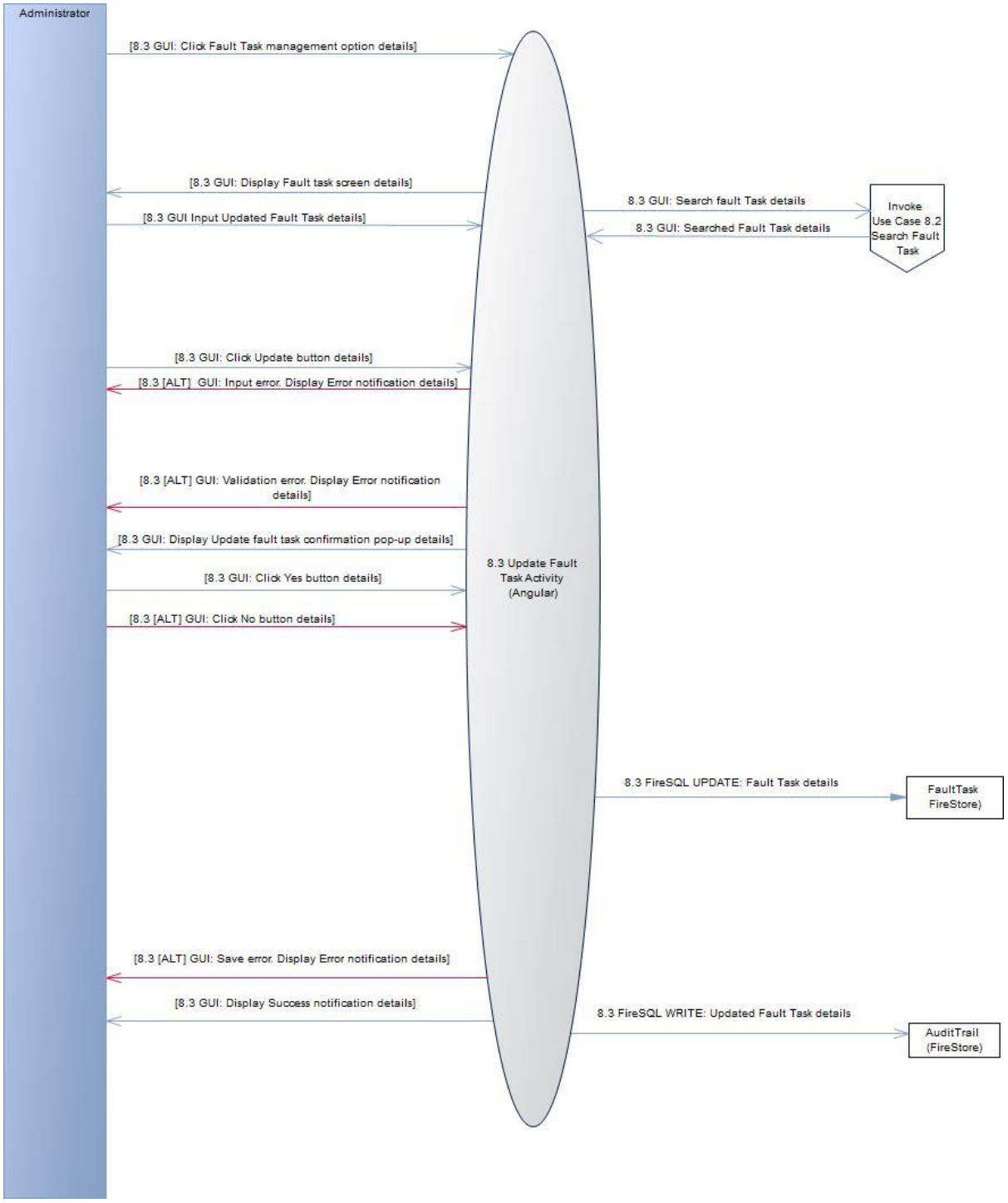


Figure 154 Middle-Level Diagram Sub-System 8: 8.3 Update Fault Task

8.4 Delete Fault Task

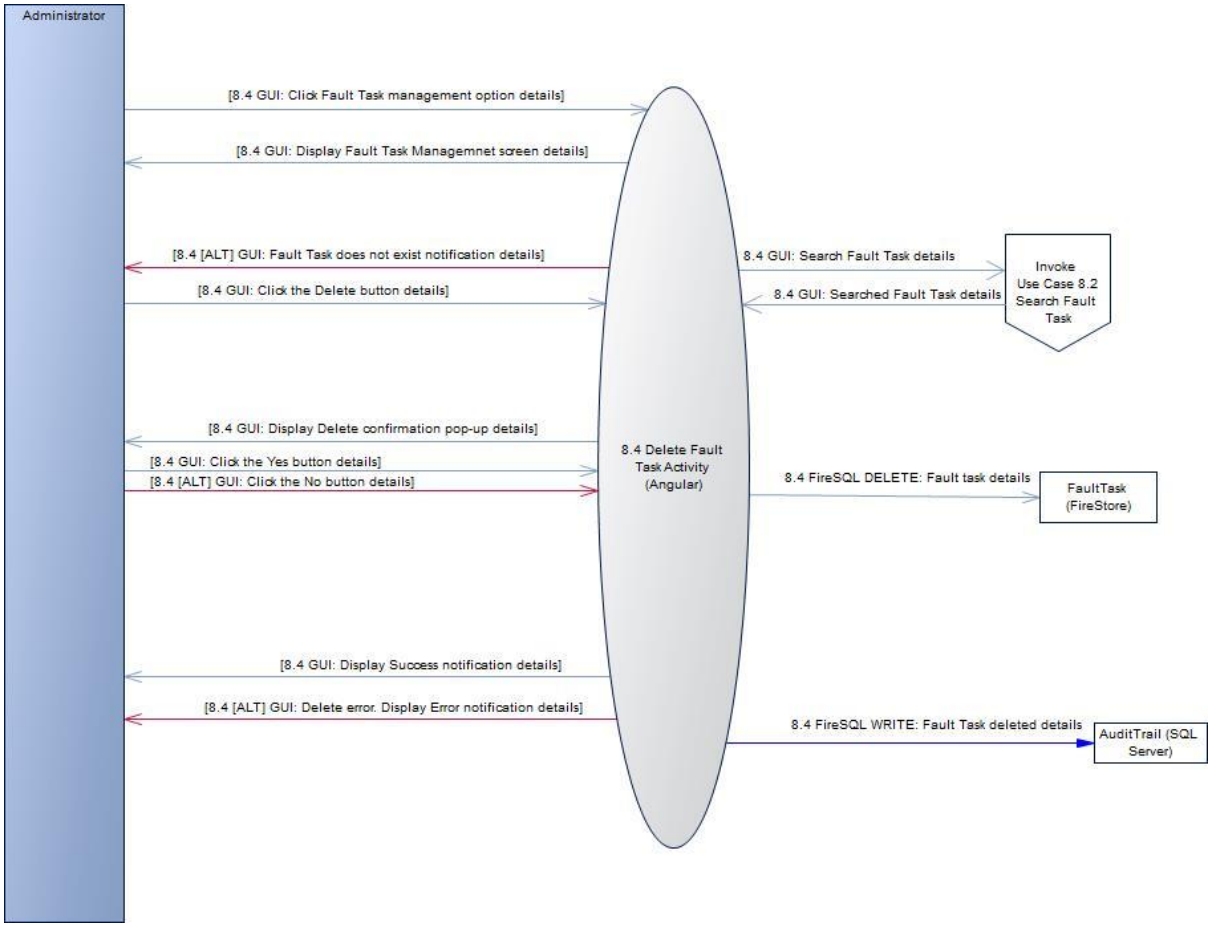


Figure 155 Middle-Level Diagram Sub-System 8: 8.4 Delete Fault Task

8.5 Add Maintenance Task

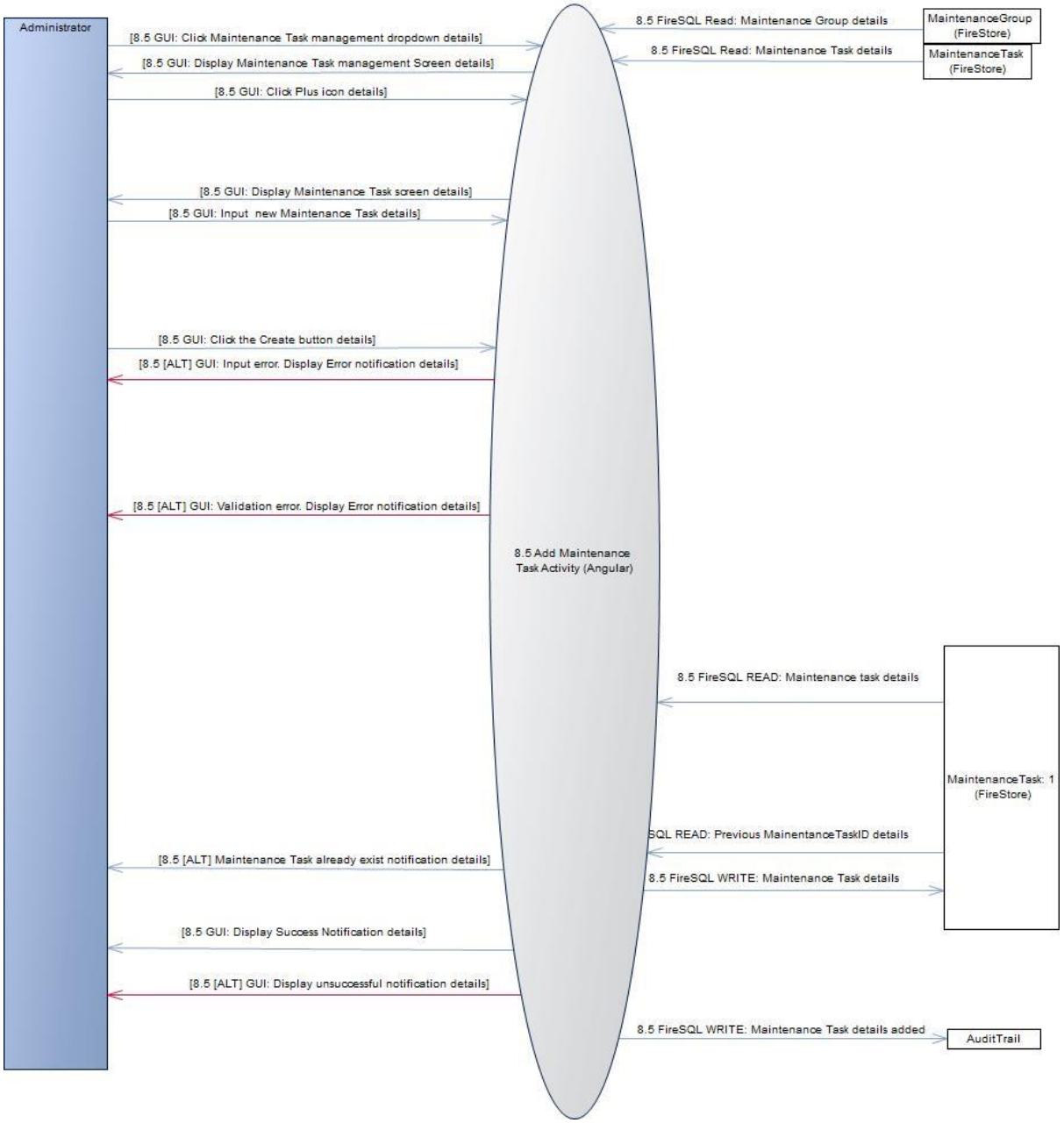


Figure 156 Middle-Level Diagram Sub-System 8: 8.5 Add Maintenance Task

8.6 Search Maintenance Task

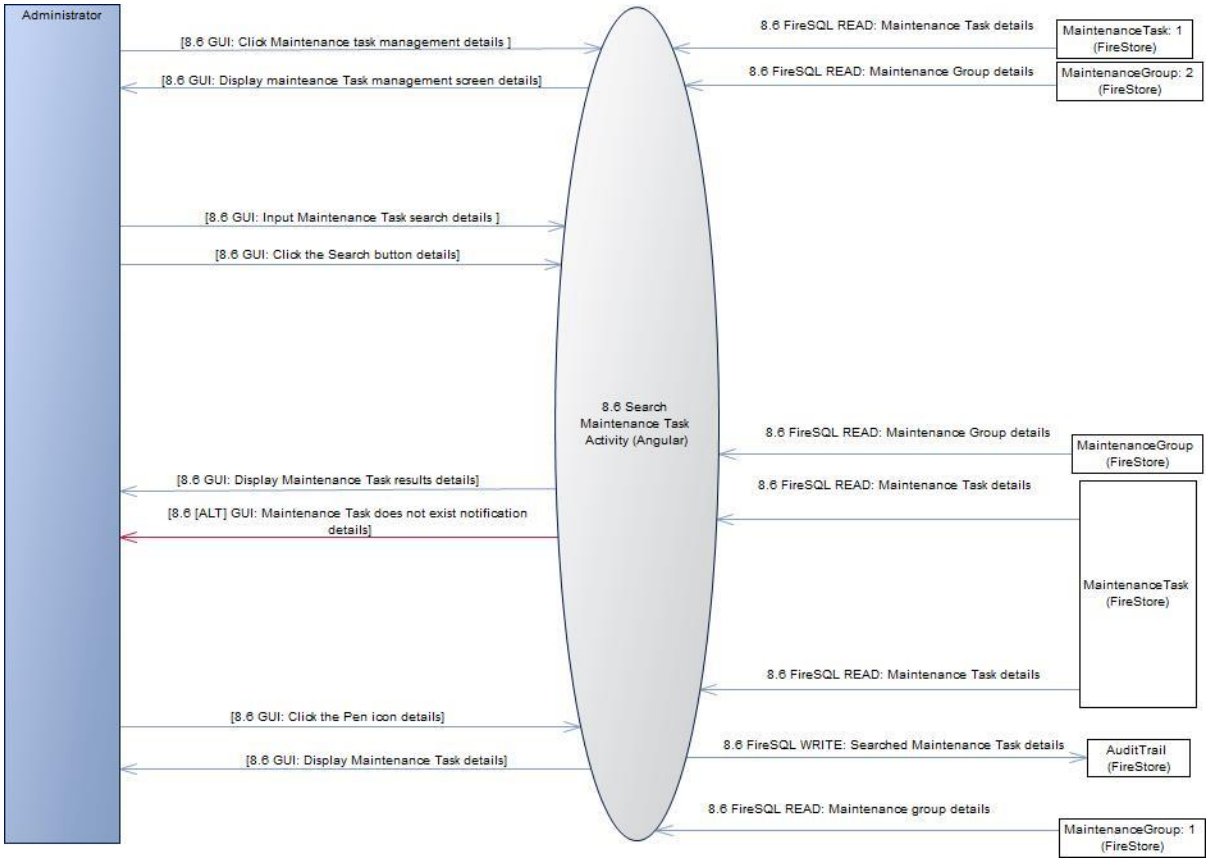


Figure 157 Middle-Level Diagram Sub-System 8: 8.6 Search Maintenance Task

8.7 Update Maintenance Task

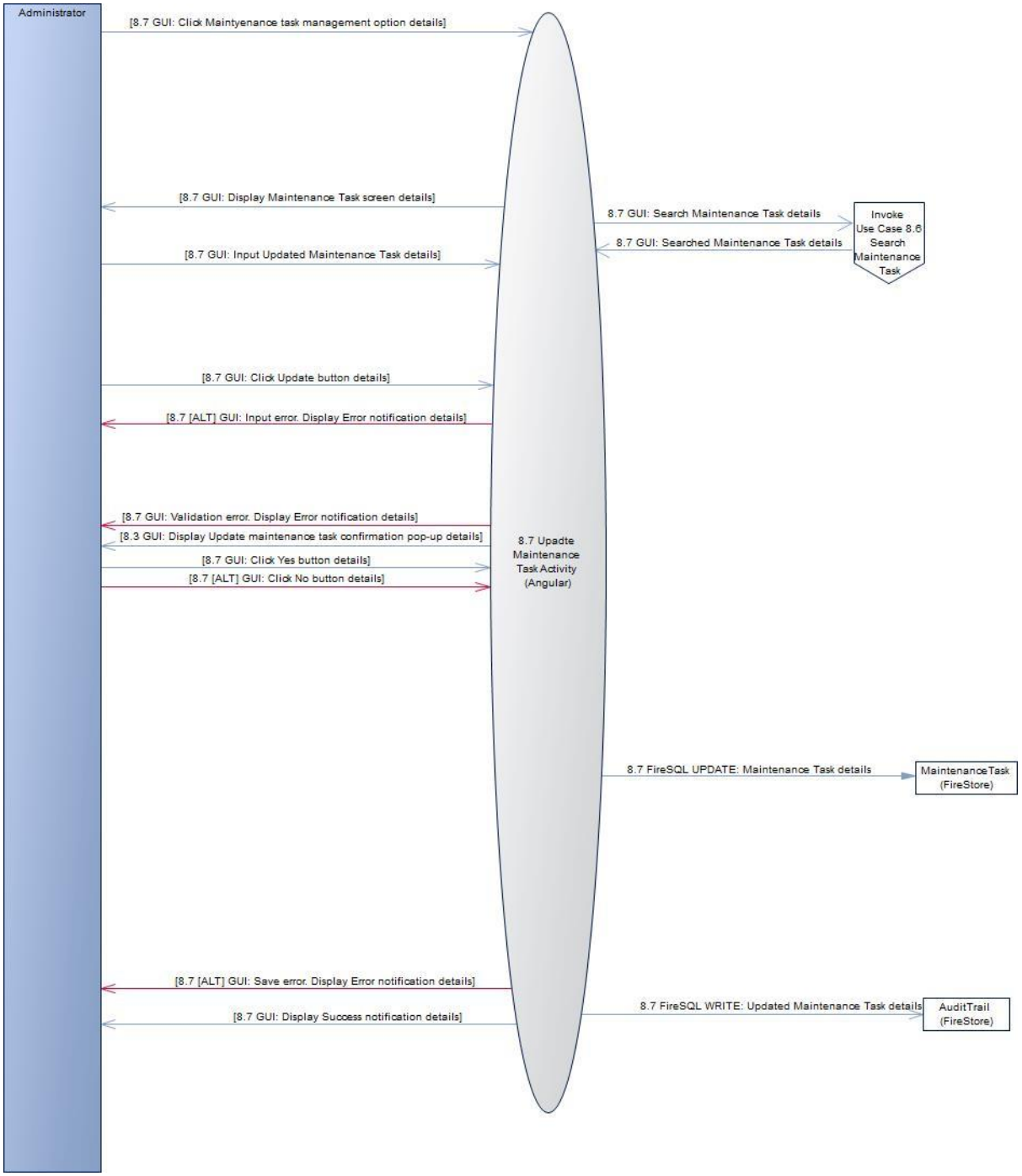


Figure 158 Middle-Level Diagram Sub-System 8: 8.7 Update Maintenance Task

8.8 Delete Maintenance Task

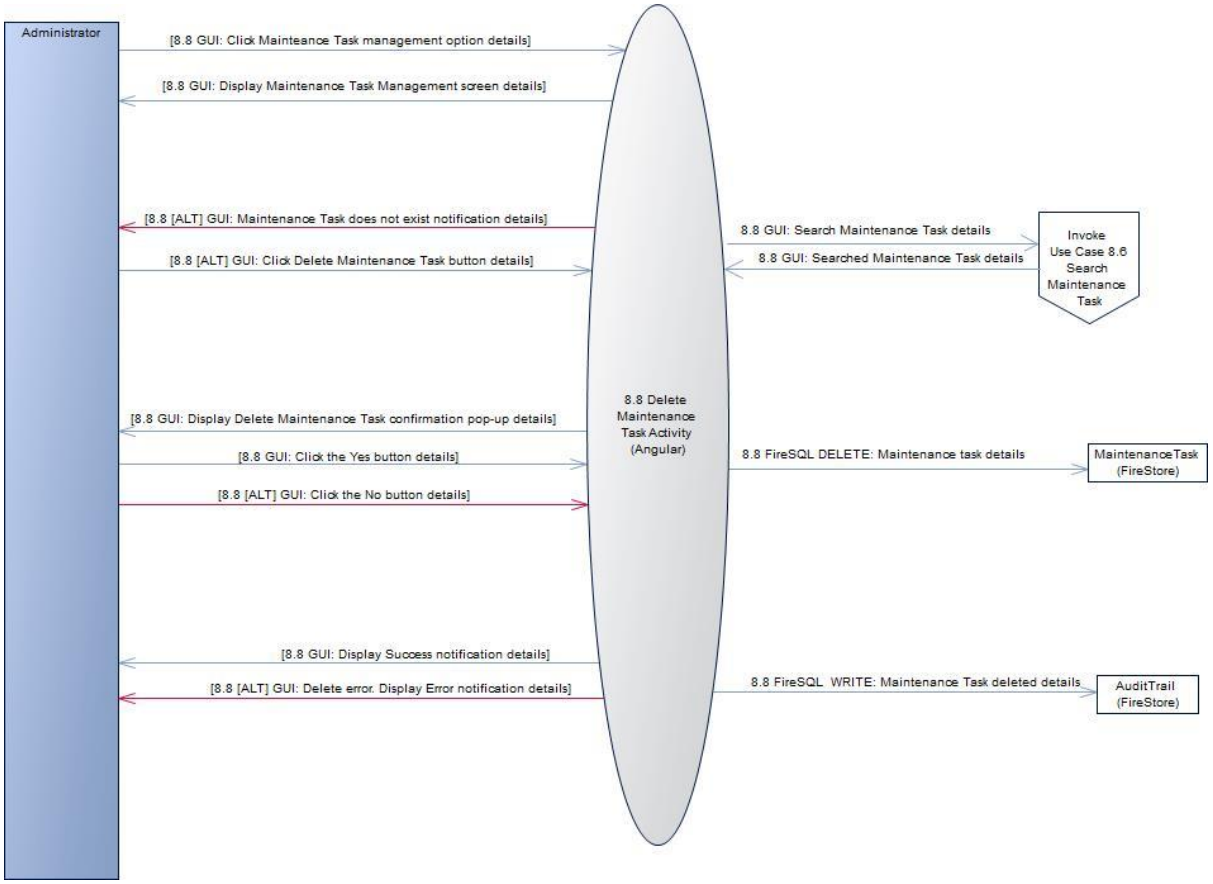


Figure 159 Middle-Level Diagram Sub-System 8: 8.8 Delete Maintenance Task

Sub-System 9: School

9.1 Add School

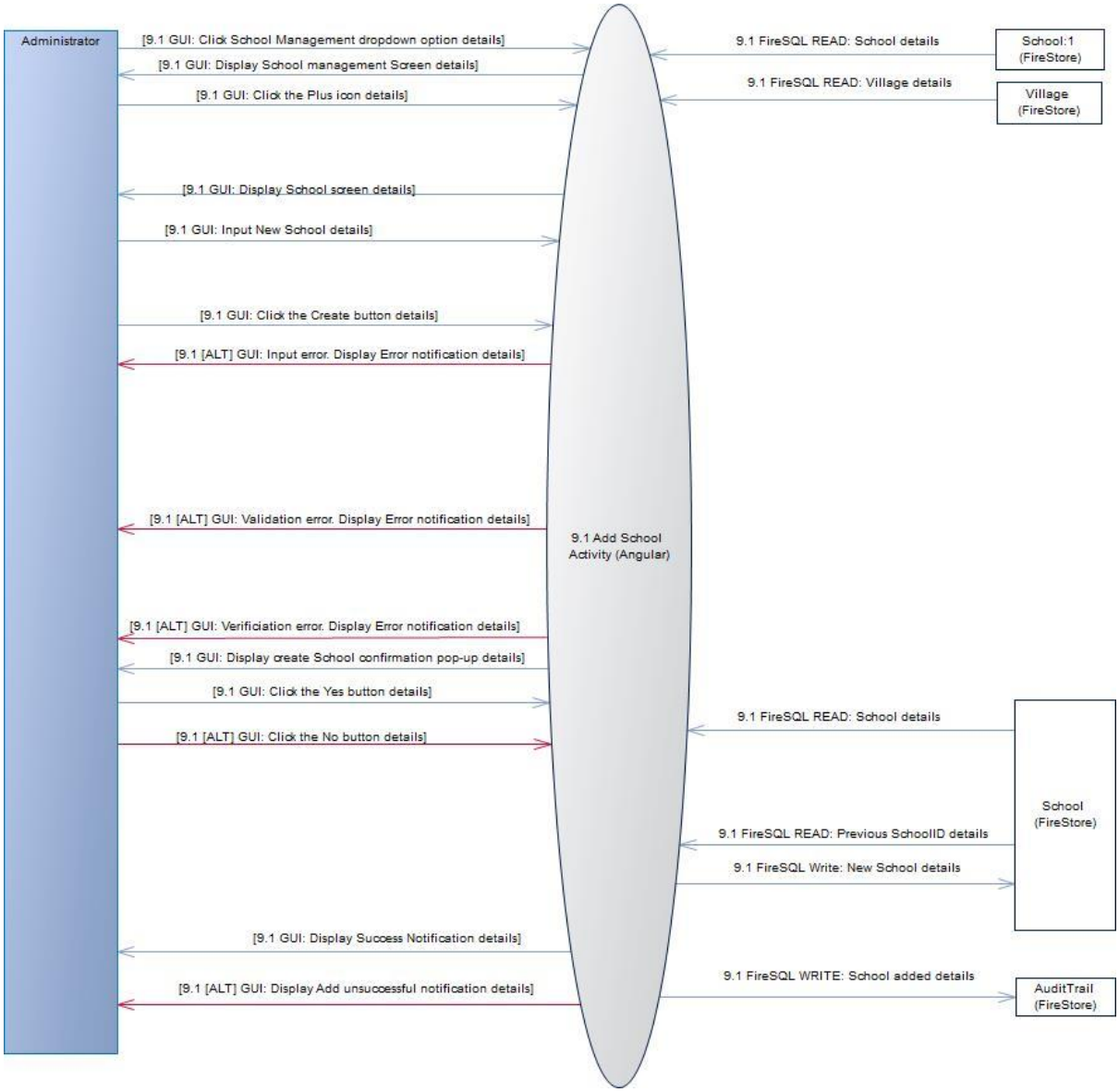


Figure 160 Middle-Level Diagram Sub-System 9: 9.1 Add School

9.2 Search School

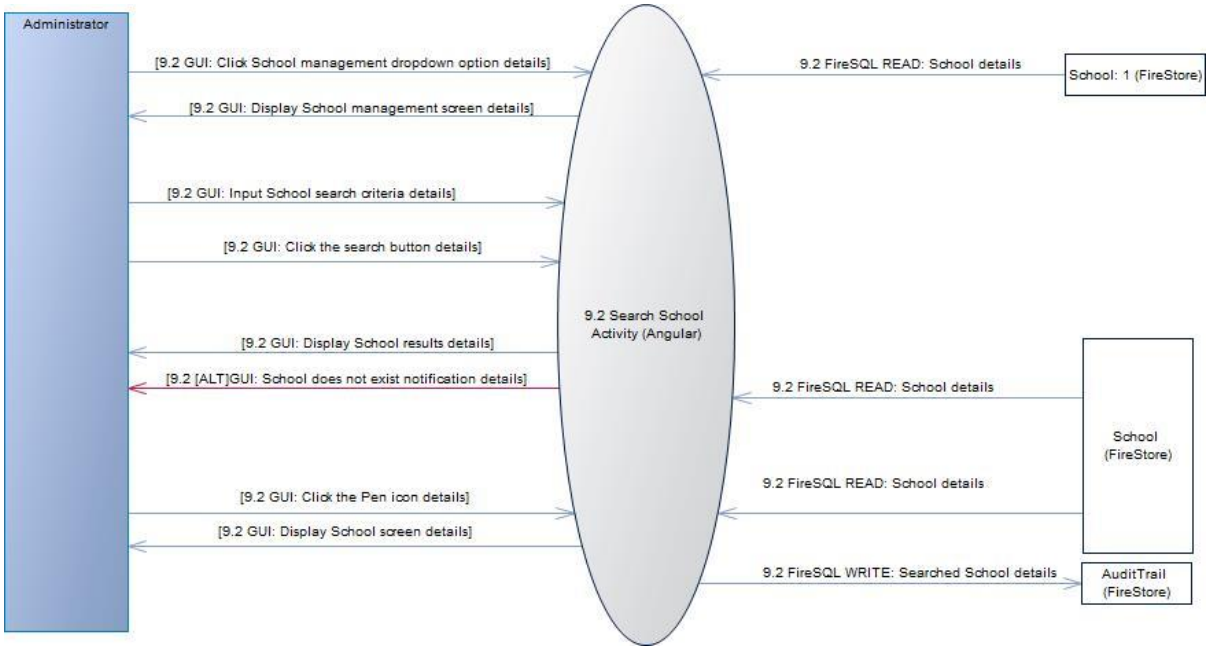


Figure 161 Middle-Level Diagram Sub-System 9: 9.2 Search School

9.3 Update School

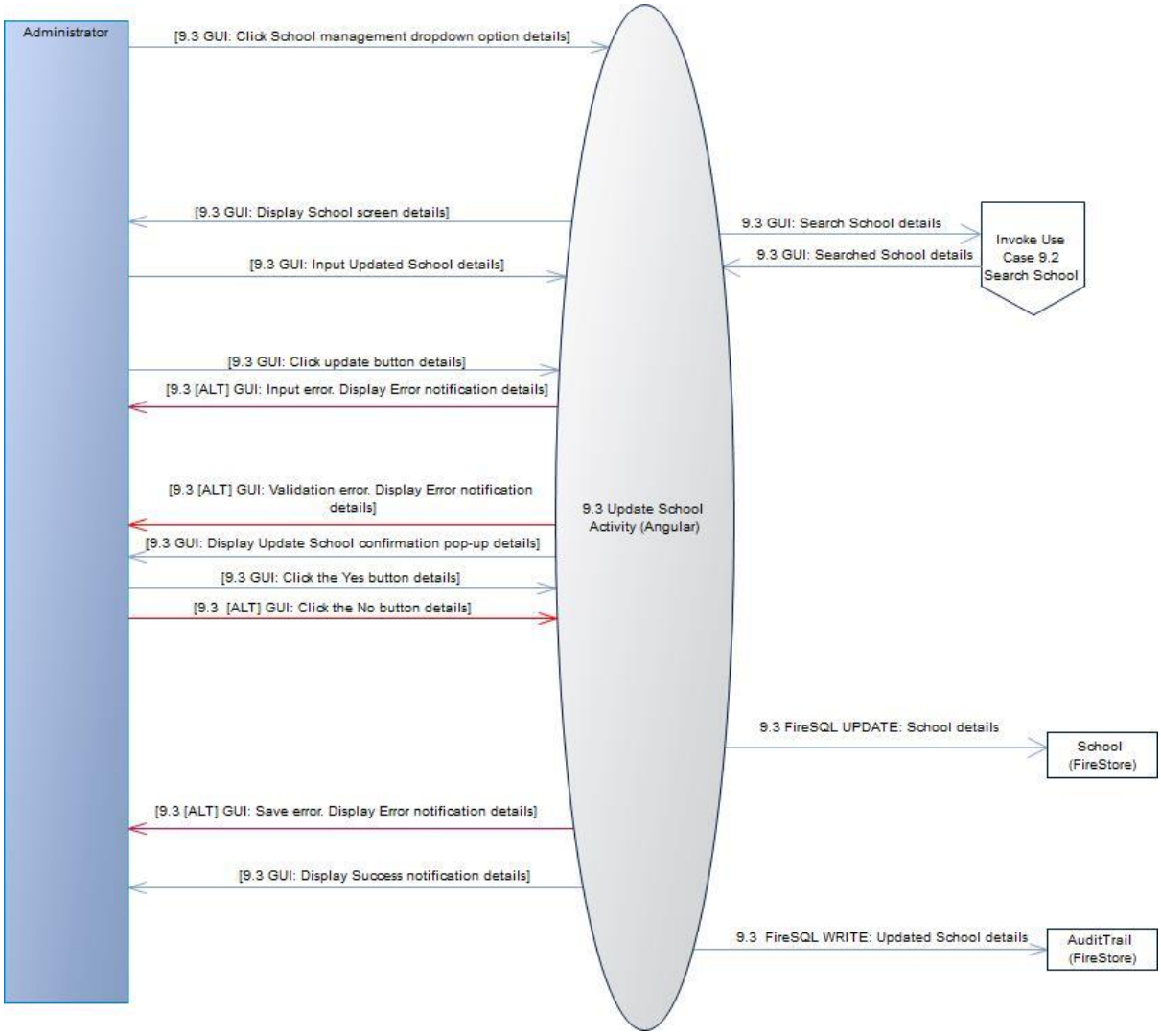


Figure 162 Middle-Level Diagram Sub-System 9: 9.3 Update School

9.4 Delete School

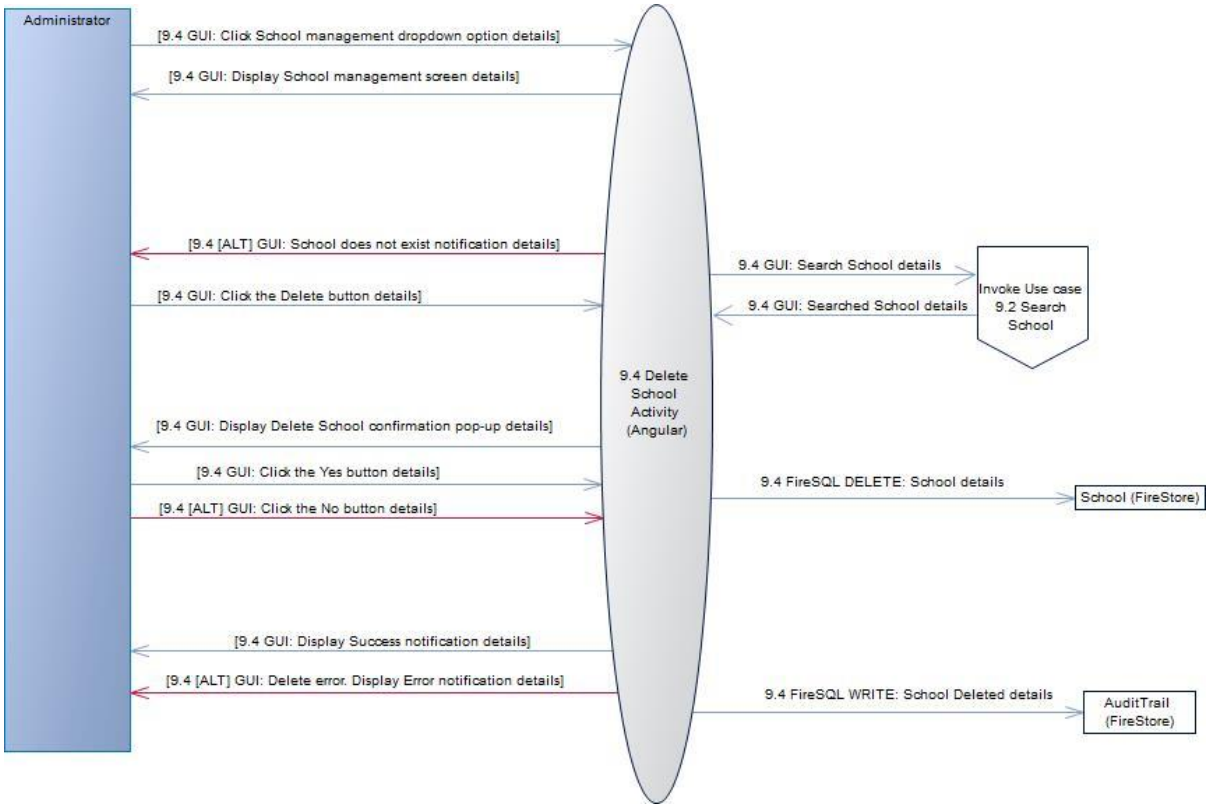


Figure 163 Middle-Level Diagram Sub-System 9: 9.4 Delete School

Sub-System 10: Job

10.1 Report Repair Job

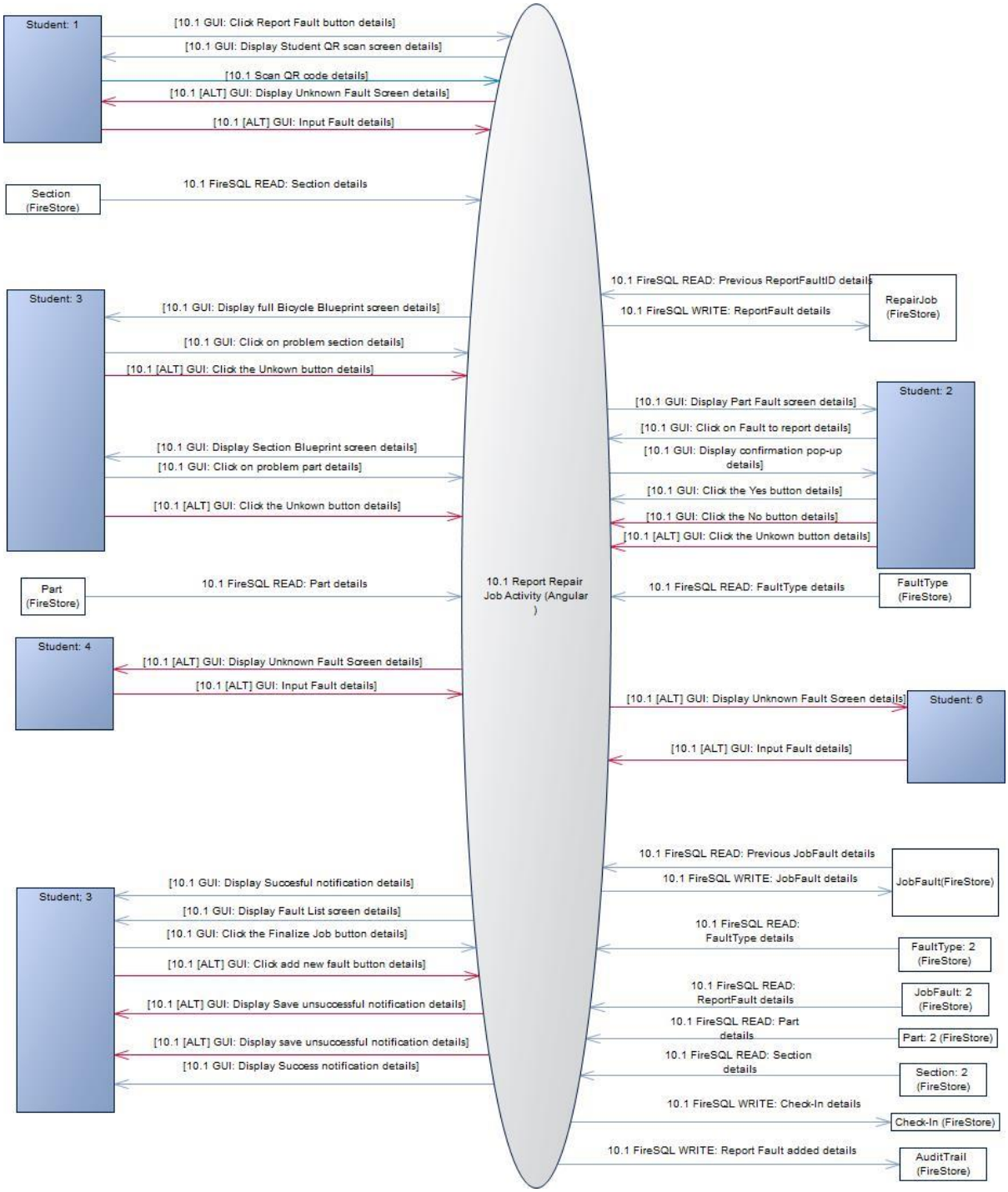


Figure 164 Middle-Level Diagram Sub-System 10: 10.1 Report Repair Job

10.2 Complete Job

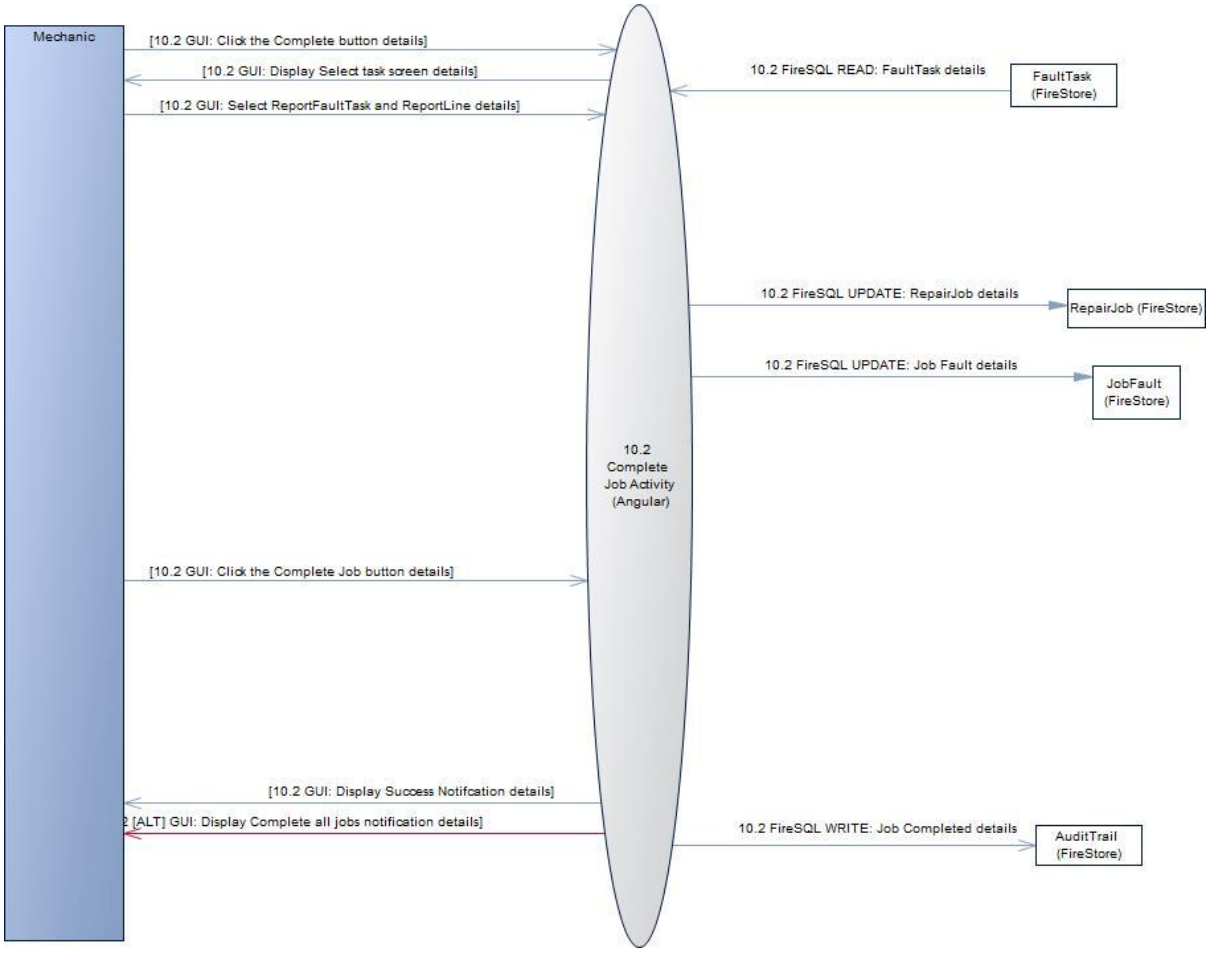


Figure 165 Middle-Level Diagram Sub-System 10: 10.2 Complete Job

10.3 Check-In

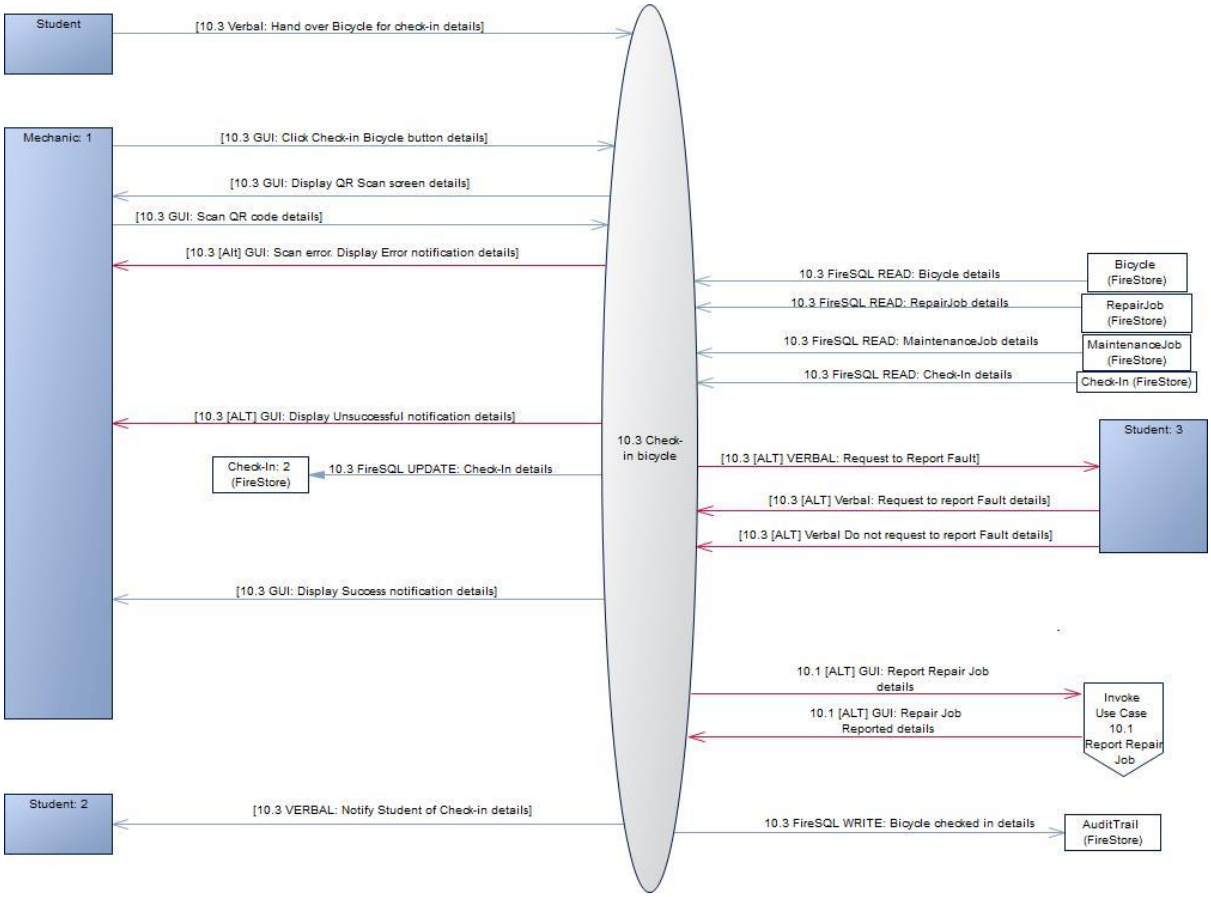


Figure 166 Middle-Level Diagram Sub-System 10: 10.3 Check-In

10.4 Report Maintenance

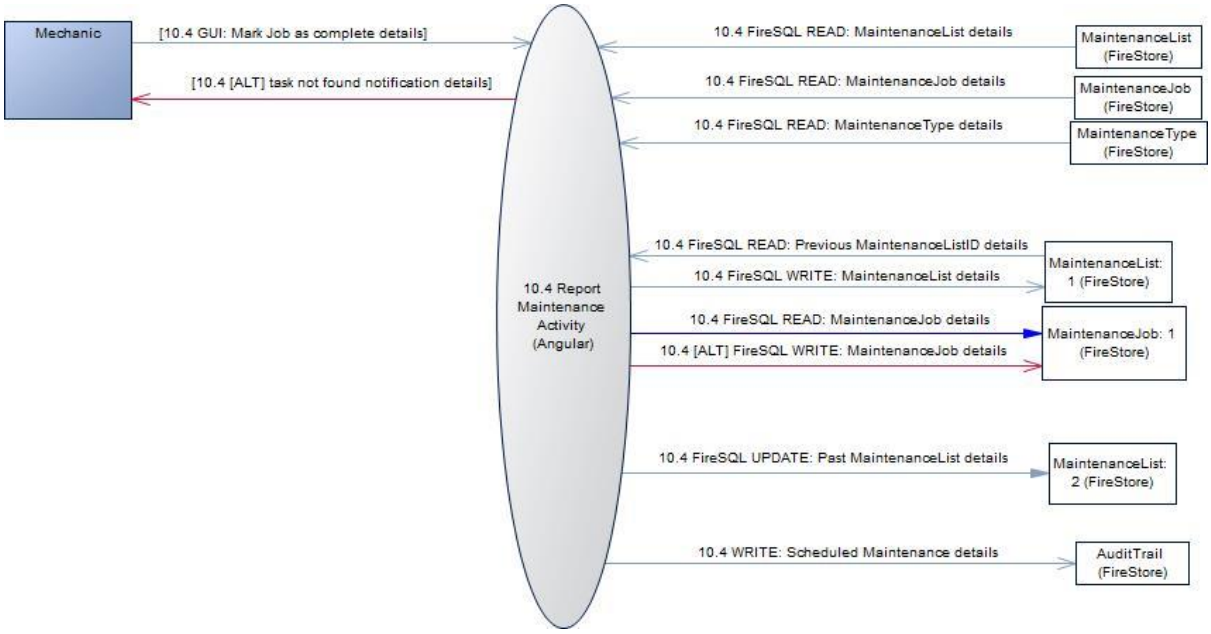


Figure 167 Middle-Level Diagram Sub-System 10: 10.4 Report Maintenance

Sub-System 11: Reporting

11.1 Generate Mechanic Schedule

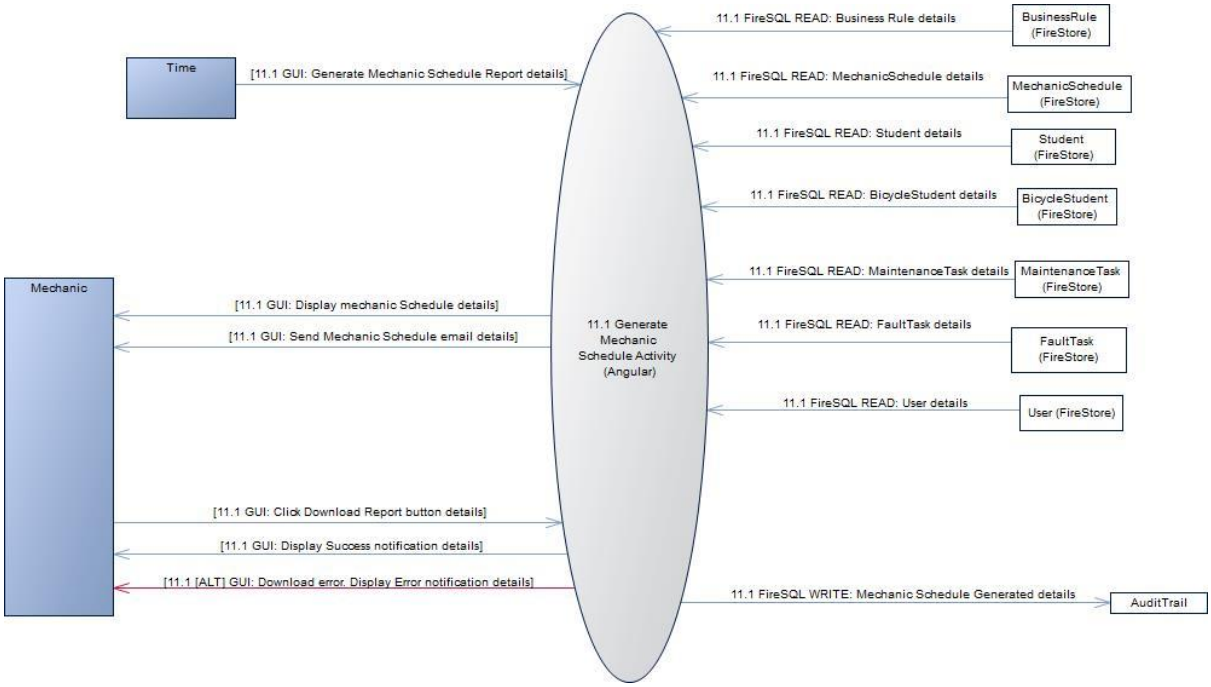


Figure 168 Middle-Level Diagram Sub-System 11: 11.1 Generate Mechanic Schedule

11.2 Generate Mechanic Schedule Requirements

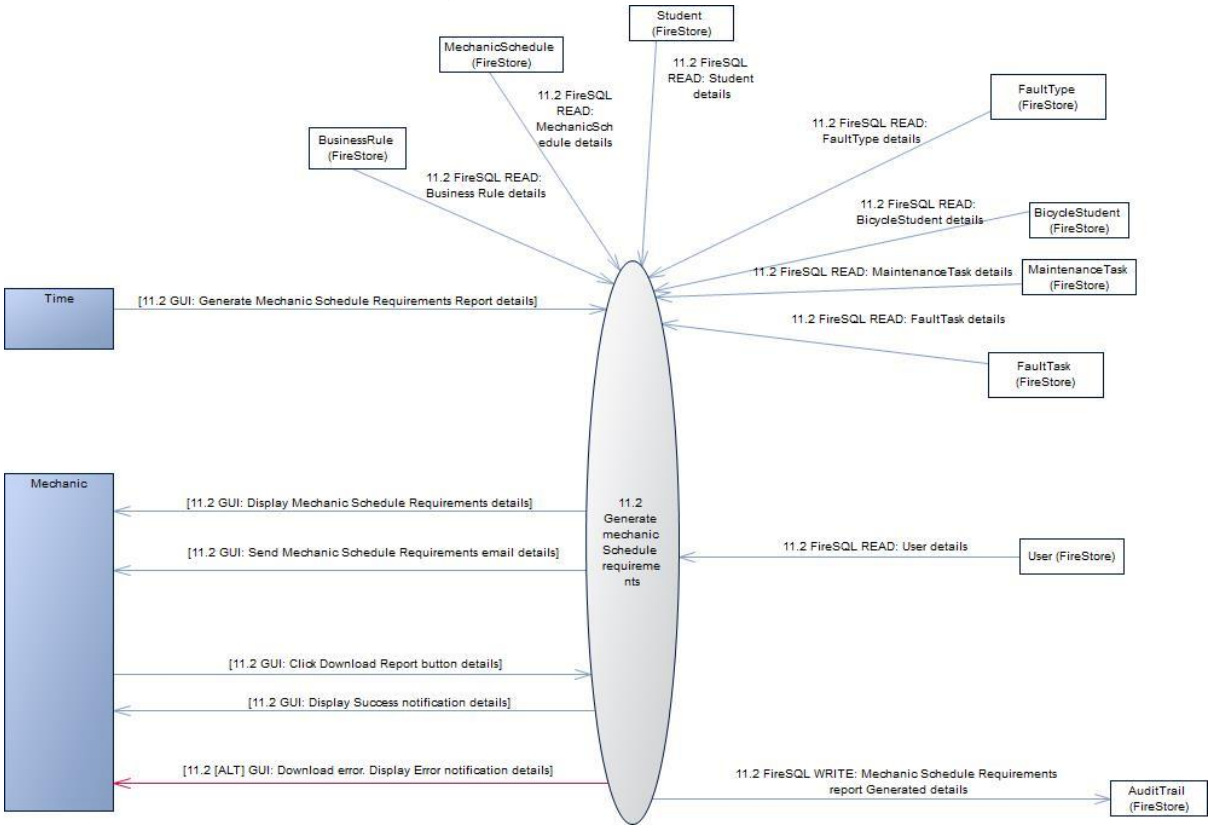


Figure 169 Middle-Level Diagram Sub-System 11: 11.2 Generate Mechanic Schedule Requirements

11.3 Generate Student Report

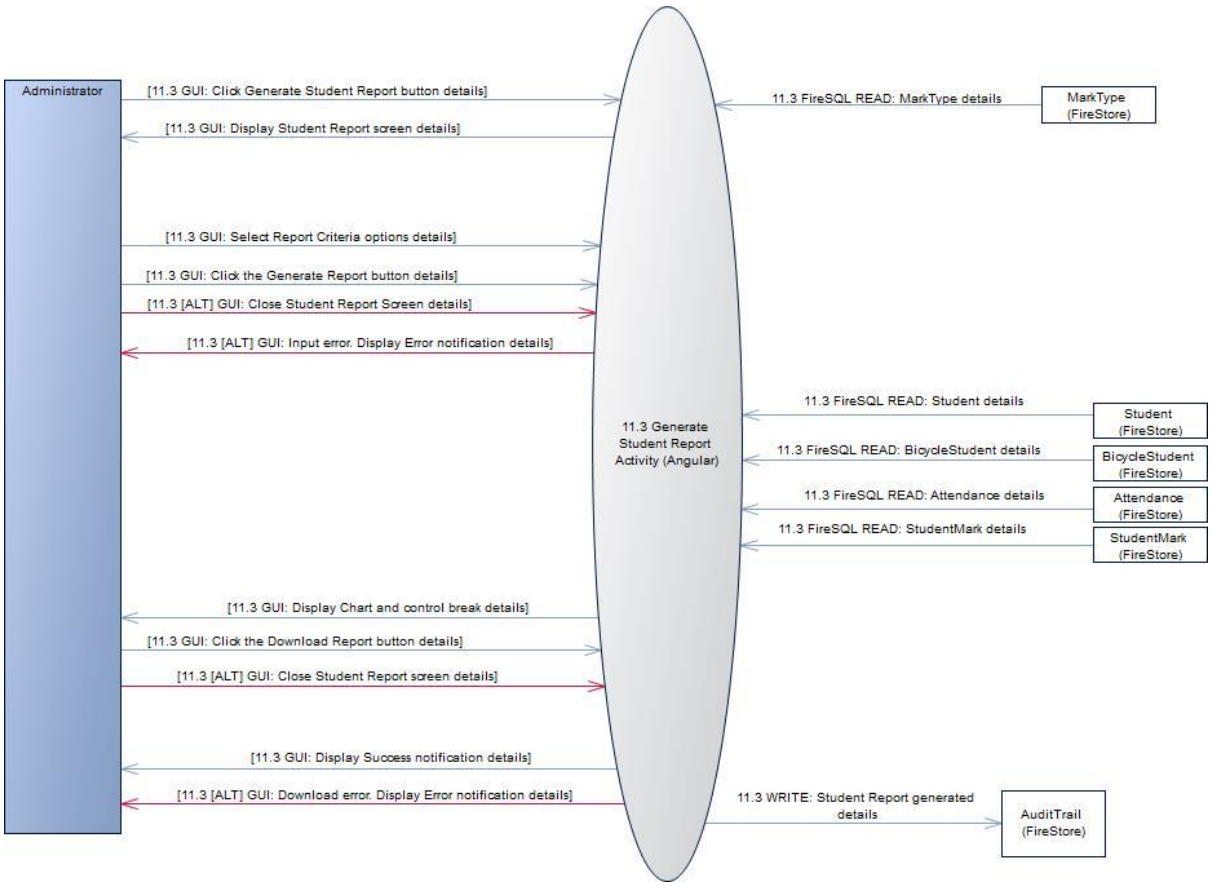


Figure 170 Middle-Level Diagram Sub-System 11: 11.3 Generate Student Report

11.4 Generate Bicycle History Report

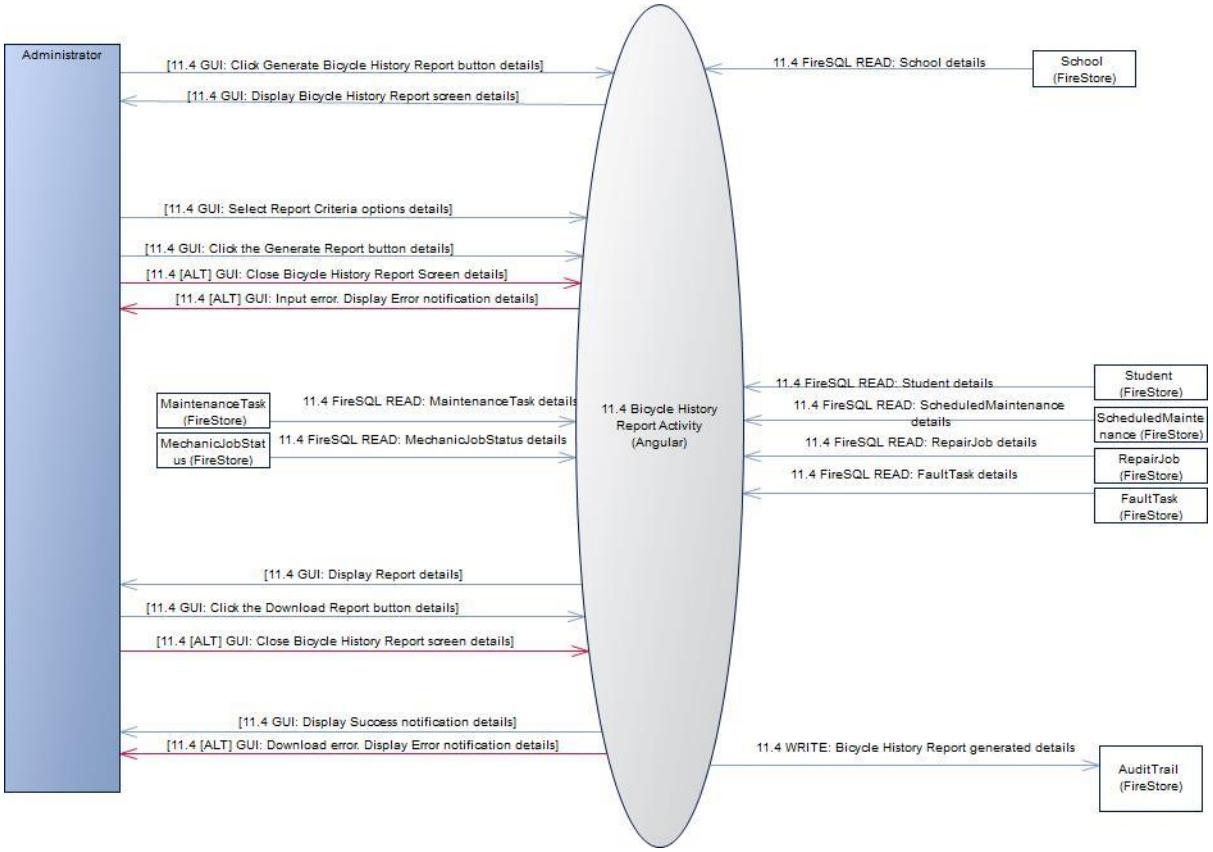


Figure 171 Middle-Level Diagram Sub-System 11: 11.4 Generate Bicycle History Report

11.5 Generate Log Report

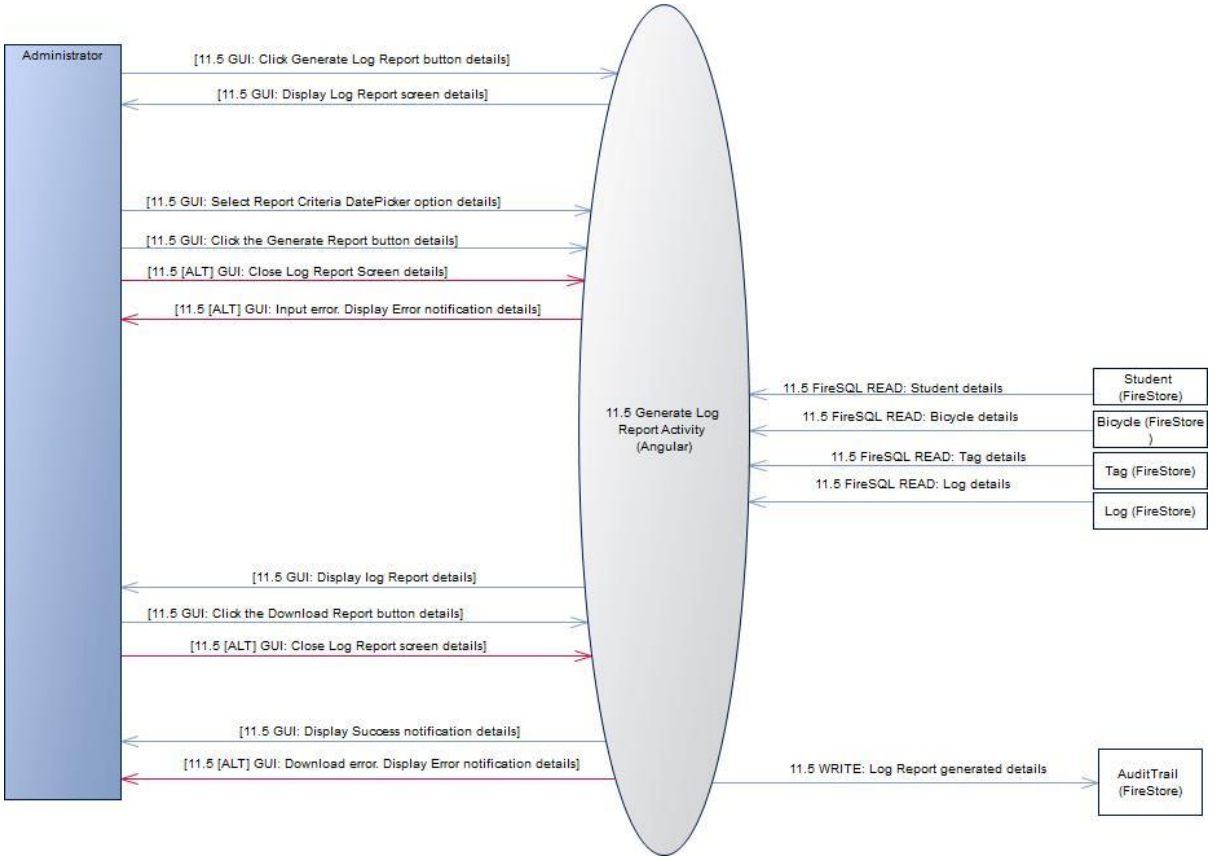


Figure 172 Middle-Level Diagram Sub-System 11: 11.5 Generate Log Report

11.6 Generate Parts Used Report

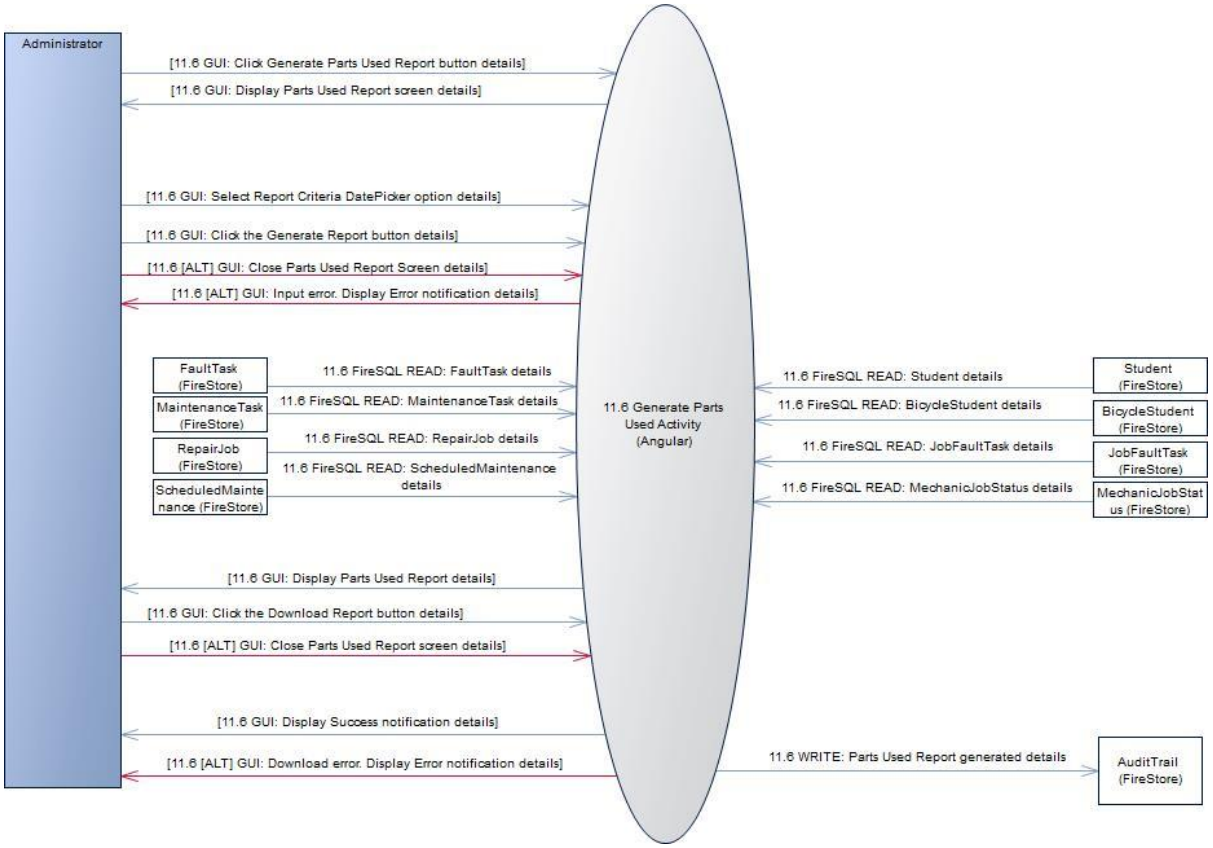


Figure 173 Middle-Level Diagram Sub-System 11: 11.6 Generate Parts Used Report

Sub-System 12: Administration

12.1 Send Student did not arrive notification

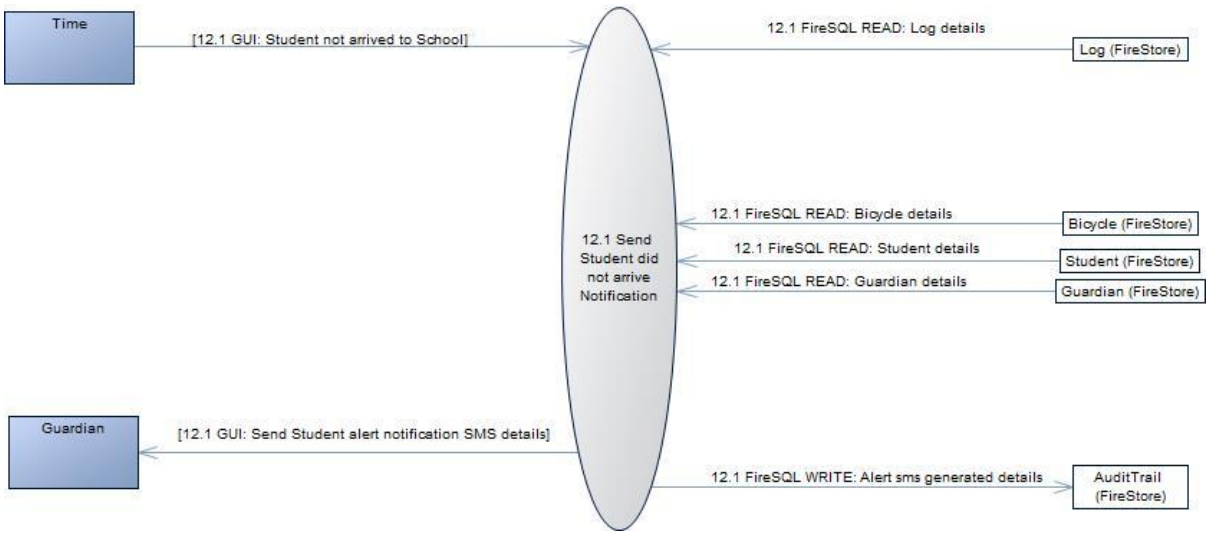


Figure 174 Middle-Level Diagram Sub-System 12: 12.1 Send Student did not arrive notification

12.2 Send Check-In Notification

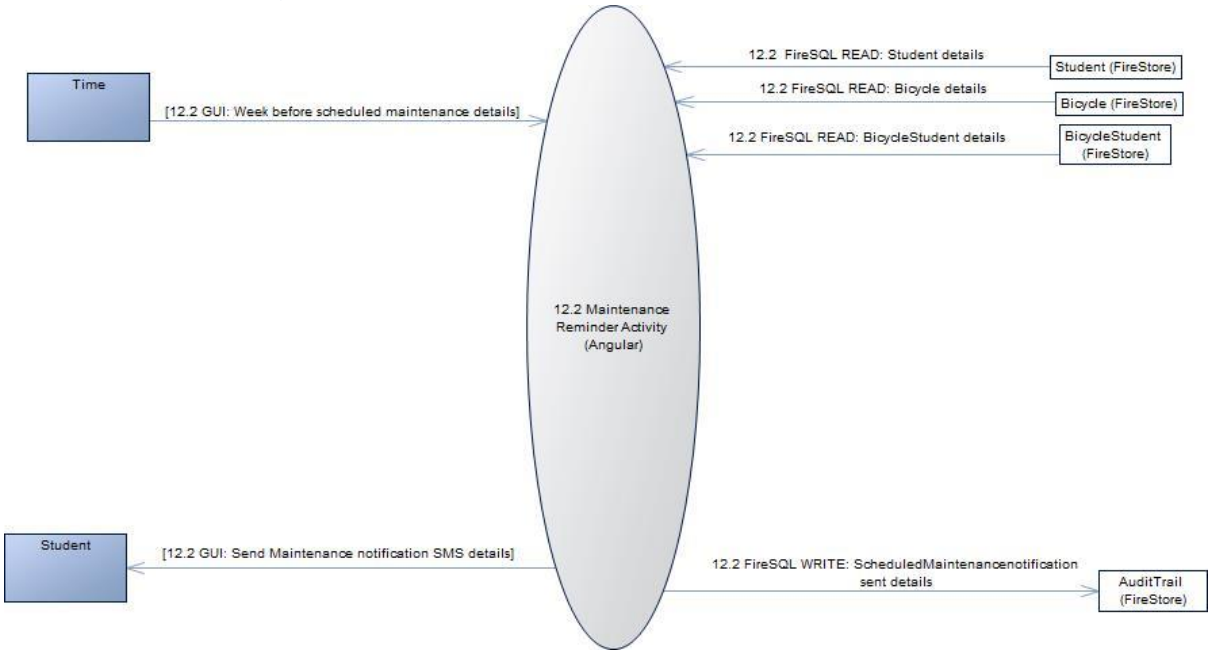


Figure 175 Middle-Level Diagram Sub-System 12: 12.2 Send Check-In Notification

Sub-System 13: Administration

13.1 Back-Up Data

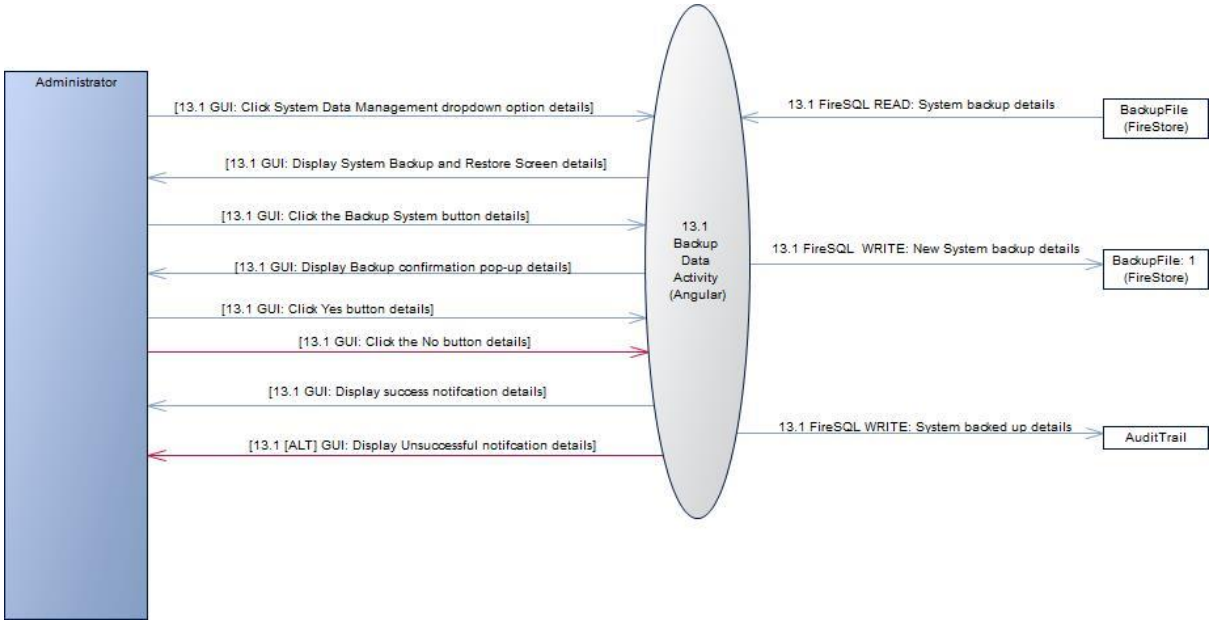


Figure 176 Middle-Level Diagram Sub-System 13: 13.1 Back-Up Data

13.2 Restore Data

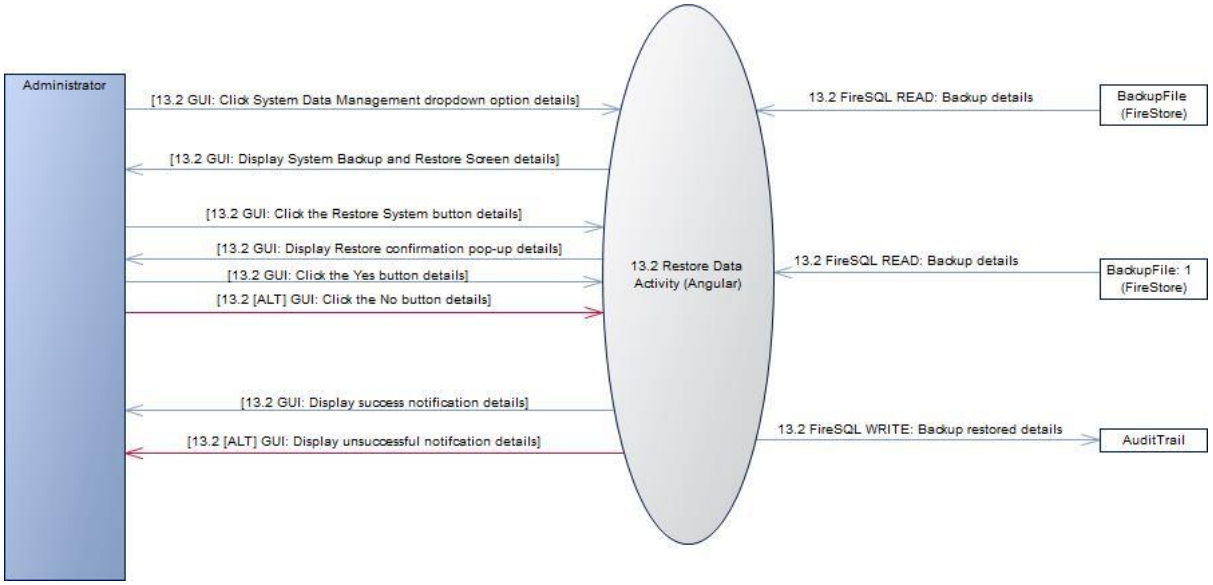


Figure 177 Middle-Level Diagram Sub-System 13: 13.2 Restore Data

13.3 View Audit Trail Log

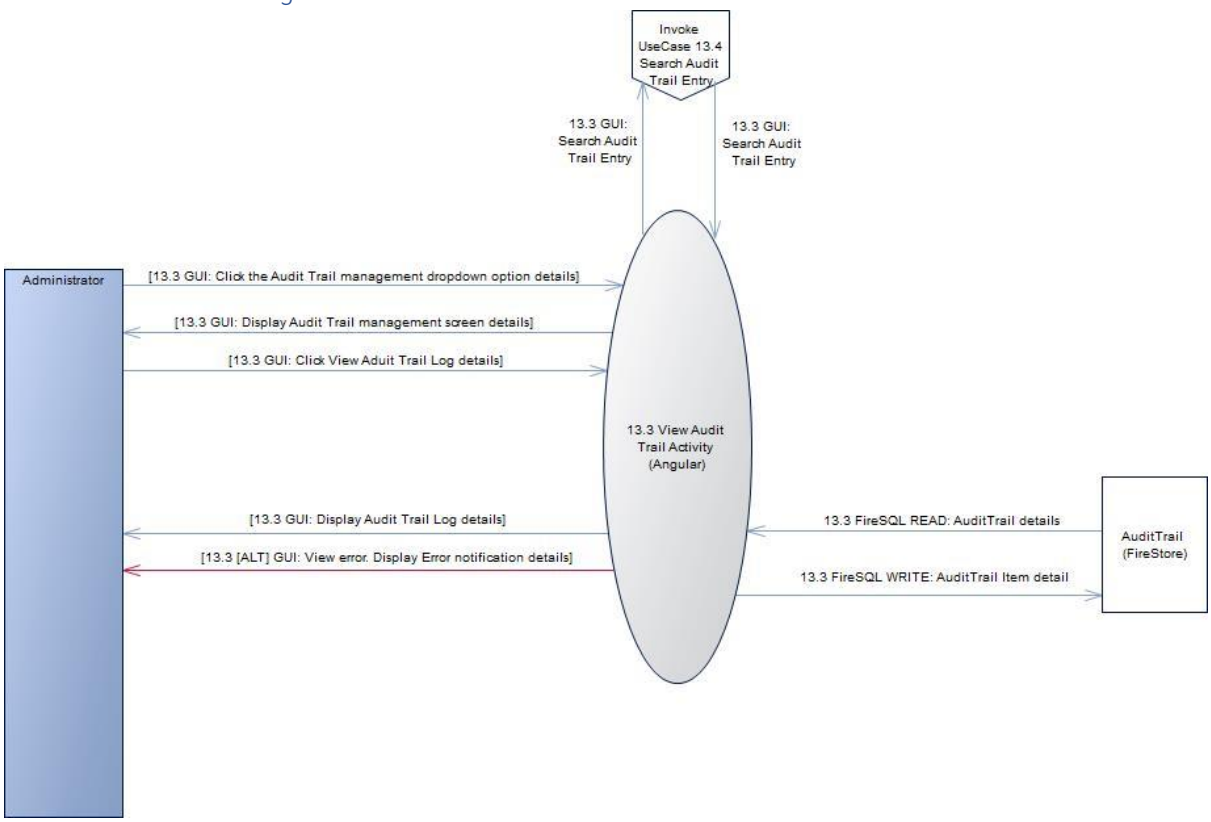
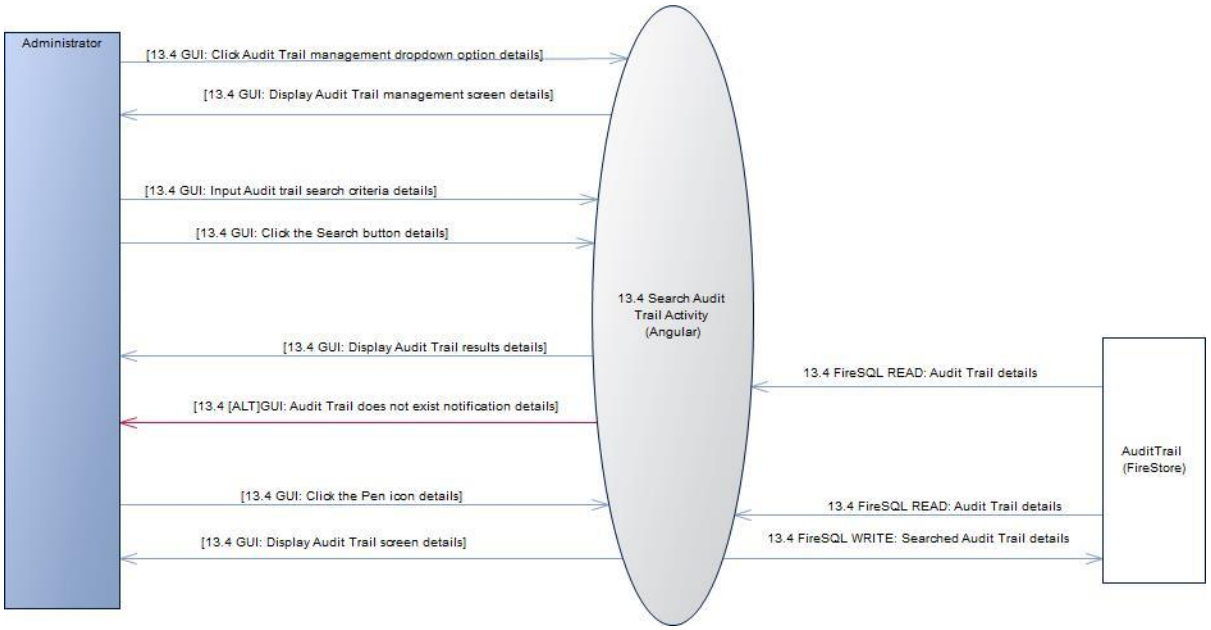


Figure 178 Middle-Level Diagram Sub-System 13: 13.3 View Audit Trail Log

13.4 Search Audit Trail Entry



Primitive Diagram

Sub System 1: User

1.1 Login

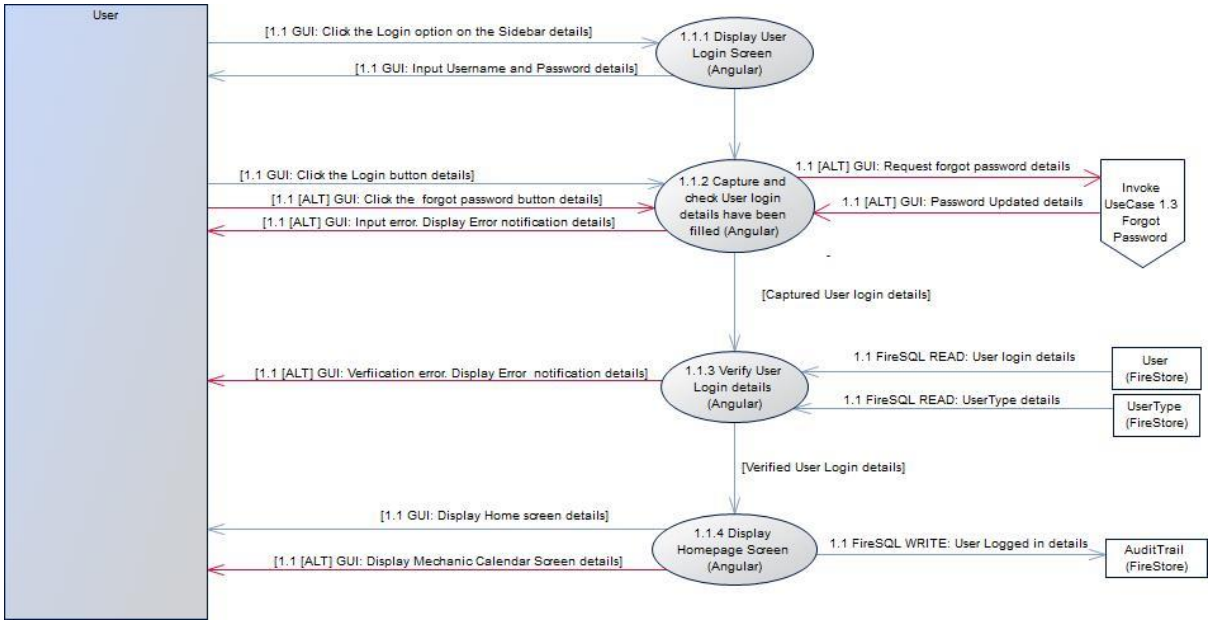


Figure 179 Primitive Diagram Sub-System 1: 1.1 Login

1.2 Logout

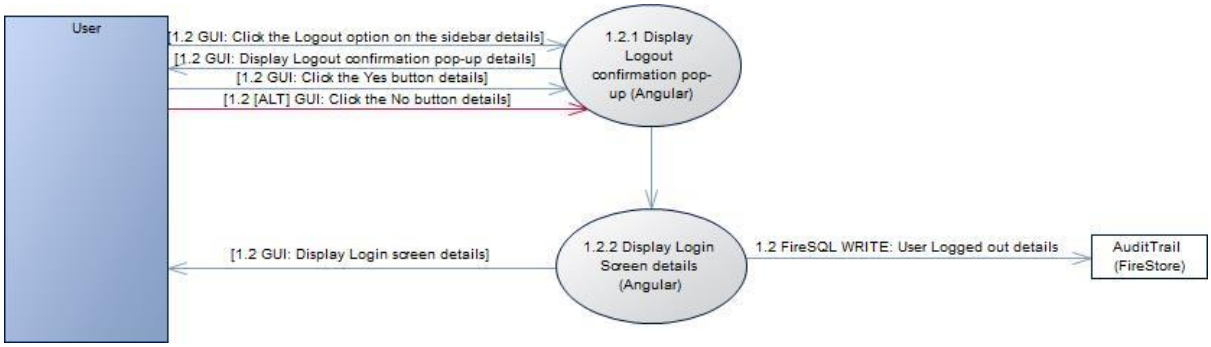


Figure 180 Primitive Diagram Sub-System 1: 1.2 Logout

1.3 Forgot Password

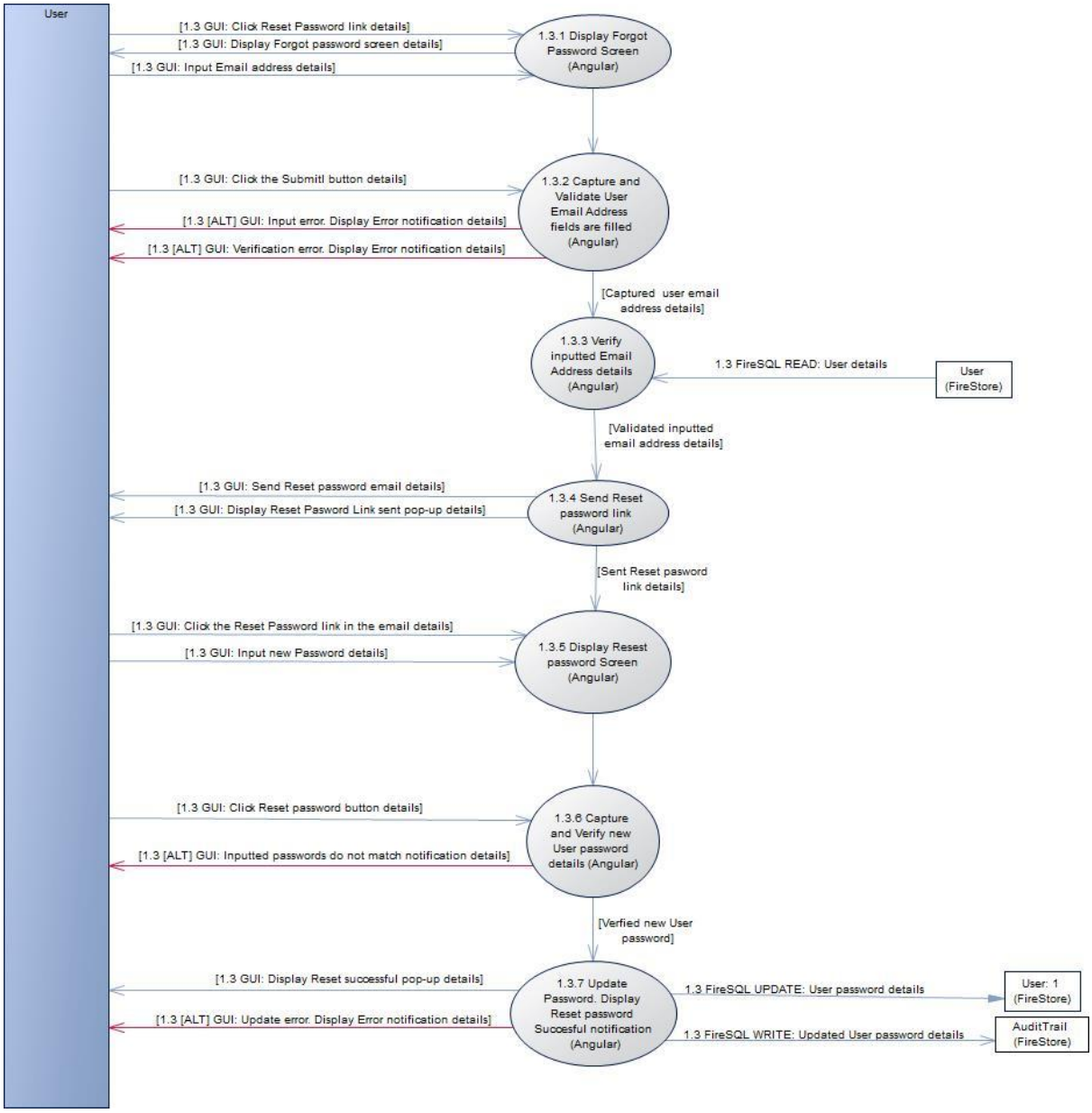


Figure 181 Primitive Diagram Sub-System 1: 1.3 Forgot Password

1.4 Add User

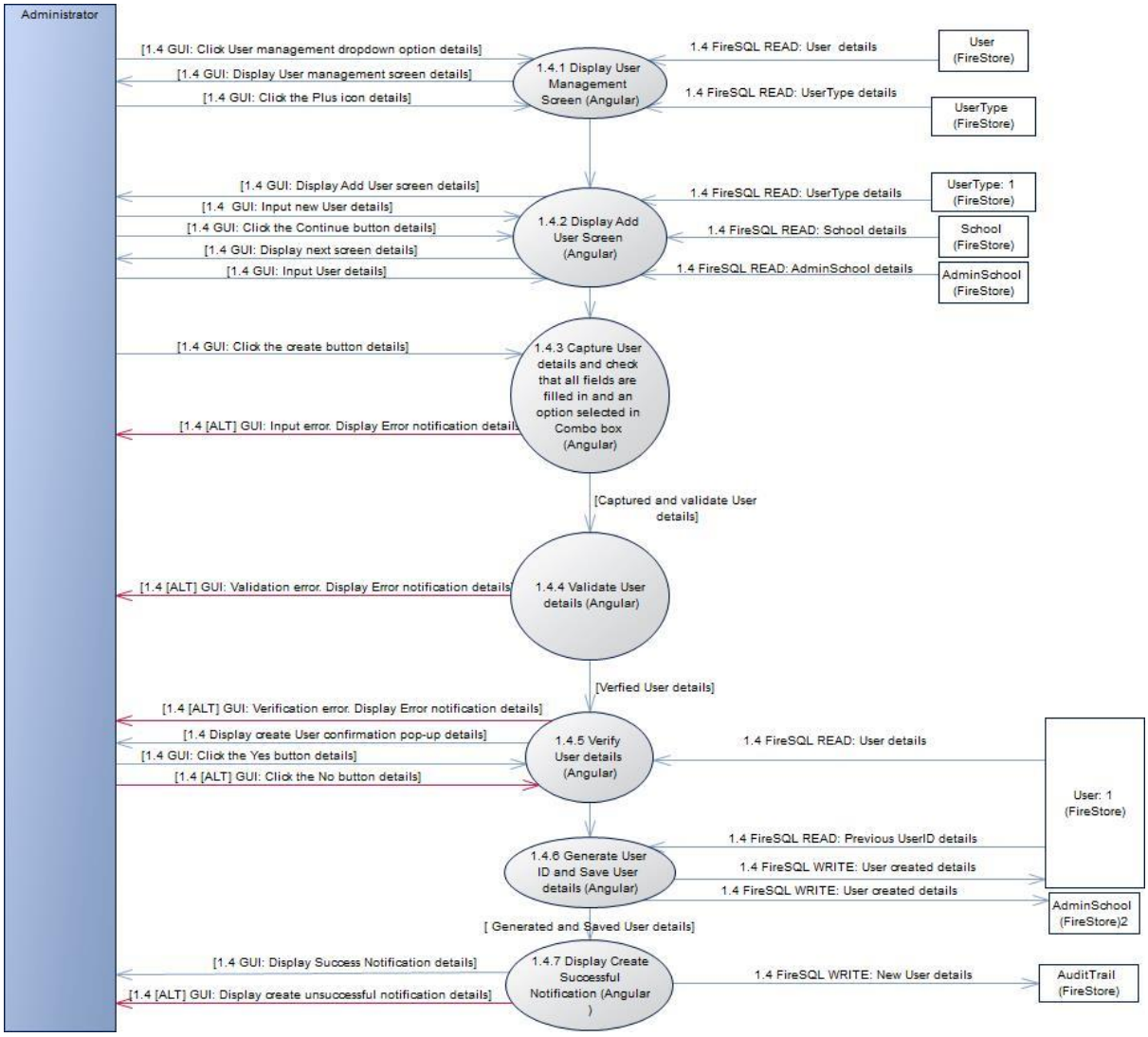


Figure 182 Primitive Diagram Sub-System 1: 1.4 Add User

1.5 Search User

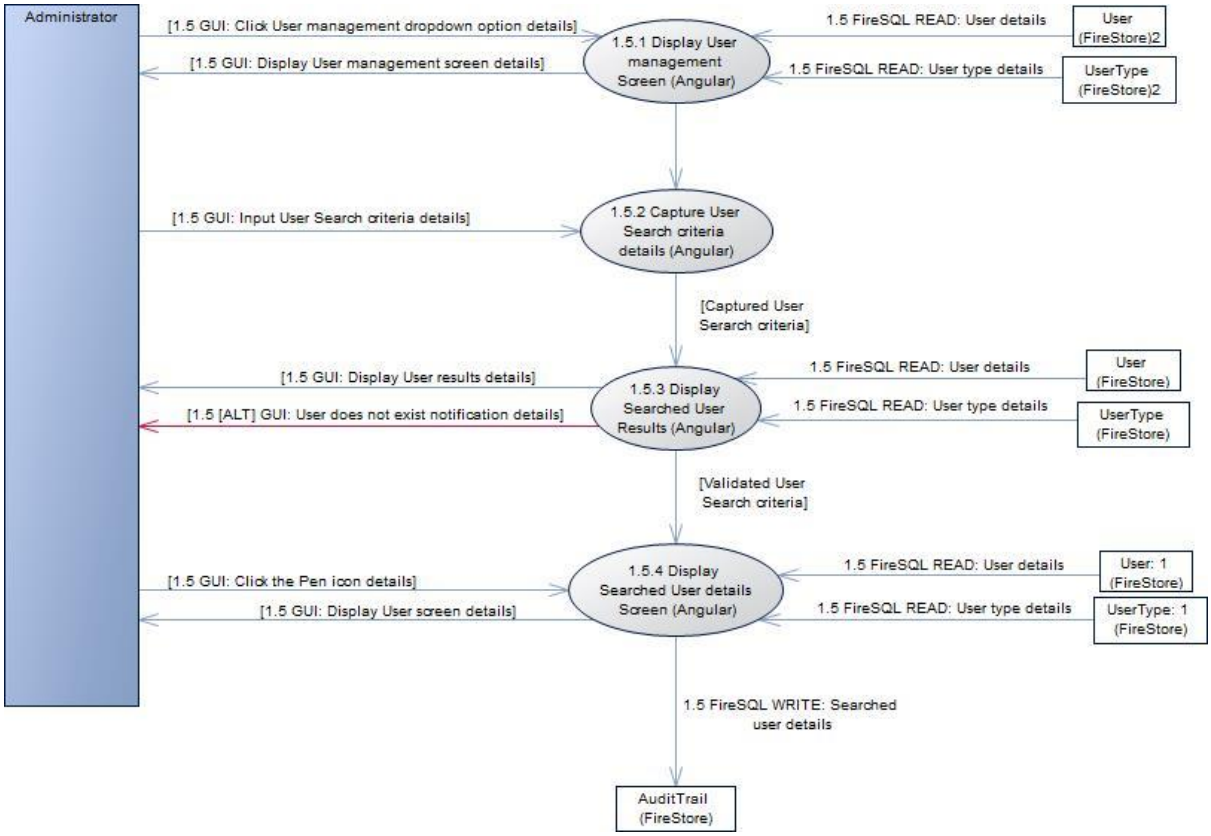


Figure 183 Primitive Diagram Sub-System 1: 1.5 Search User

1.6 Update User

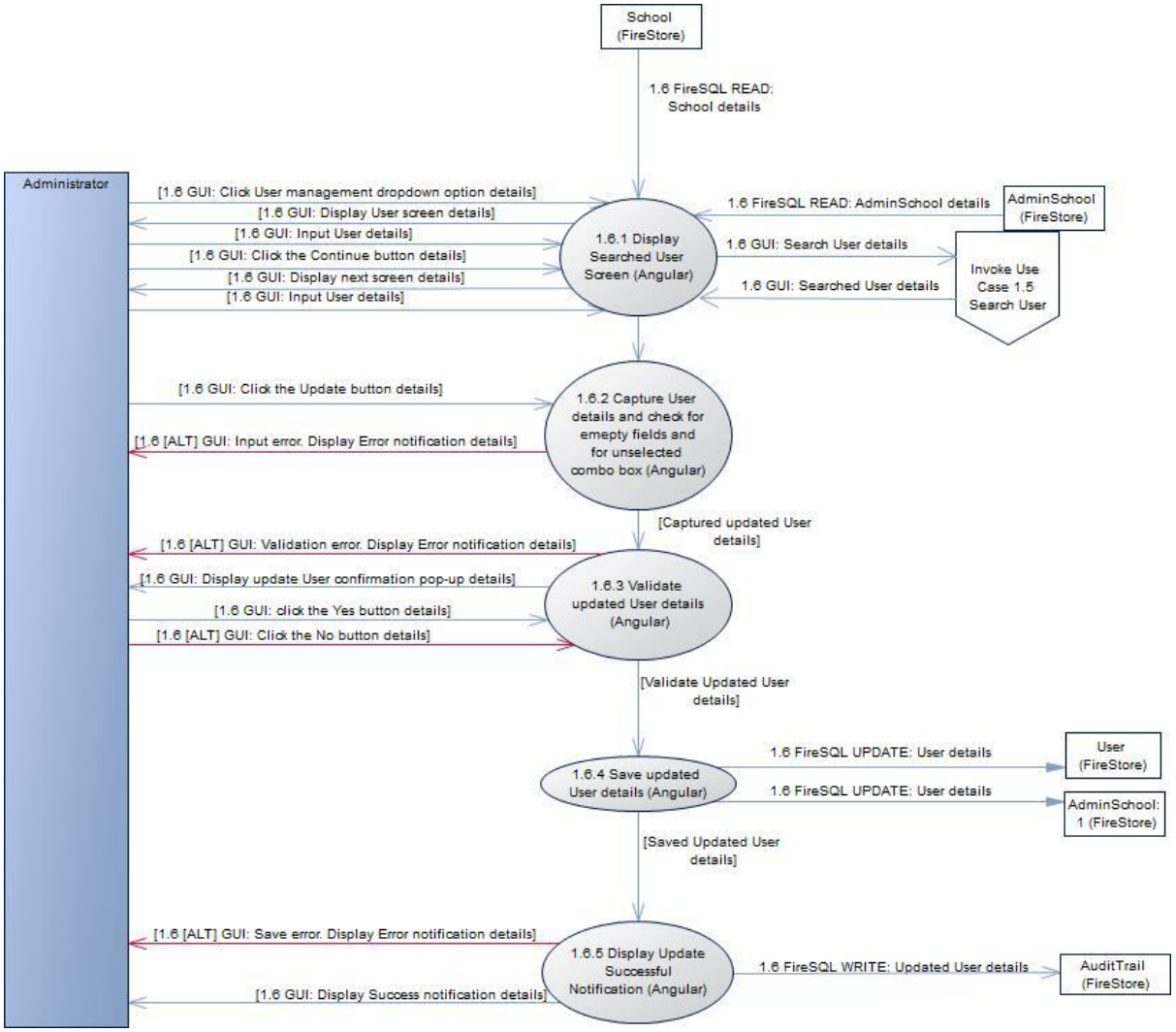


Figure 184 Primitive Diagram Sub-System 1: 1.6 Update User

1.7 Delete User

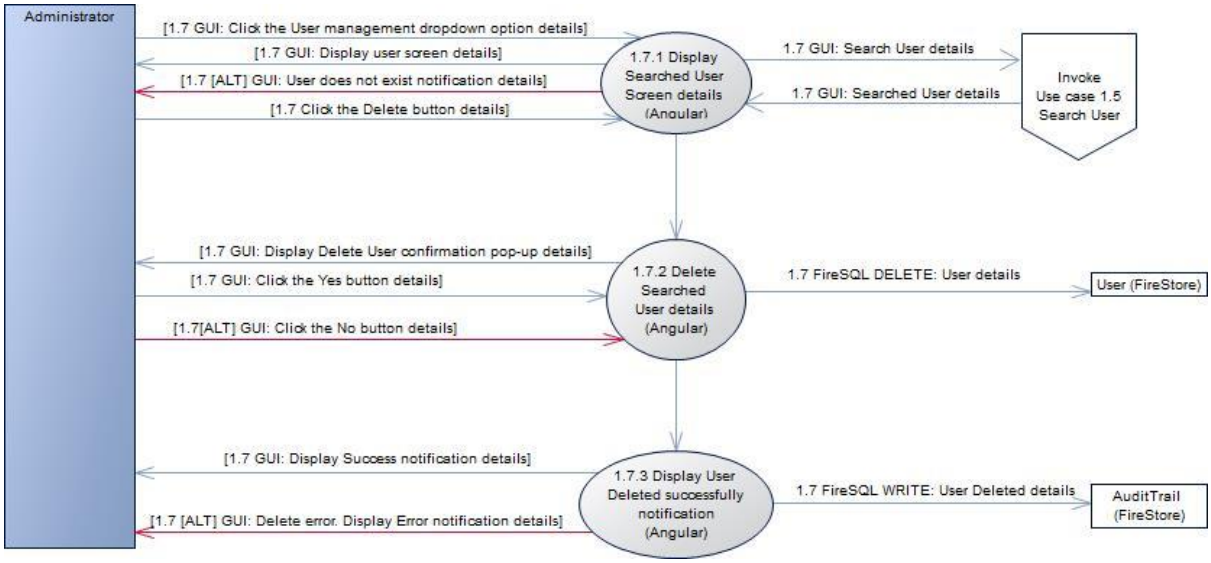


Figure 185 Primitive Diagram Sub-System 1: 1.7 Delete User

1.8 Add User Type

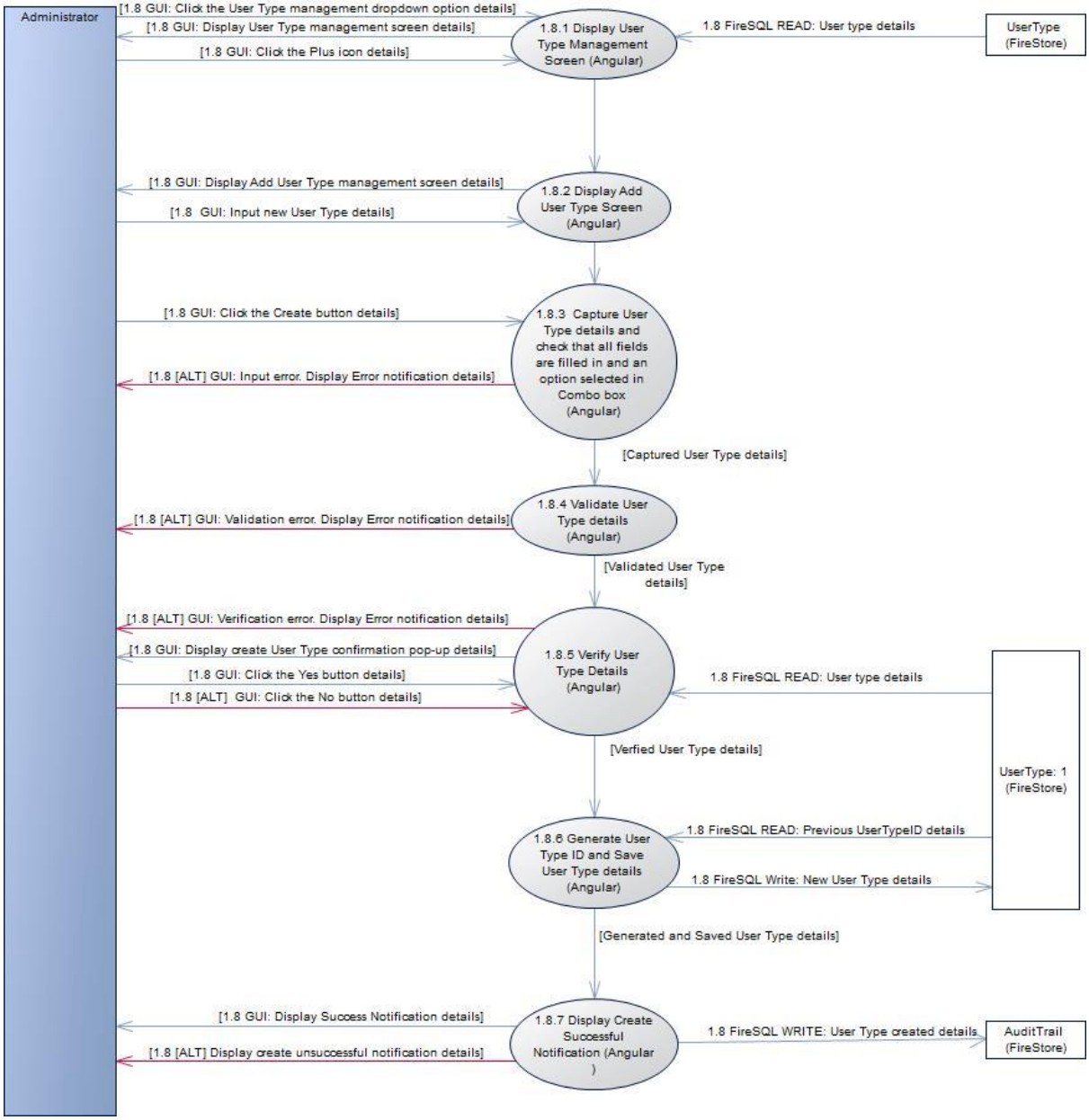


Figure 186 Primitive Diagram Sub-System 1: 1.8 Add User Type

1.9 Search User Type

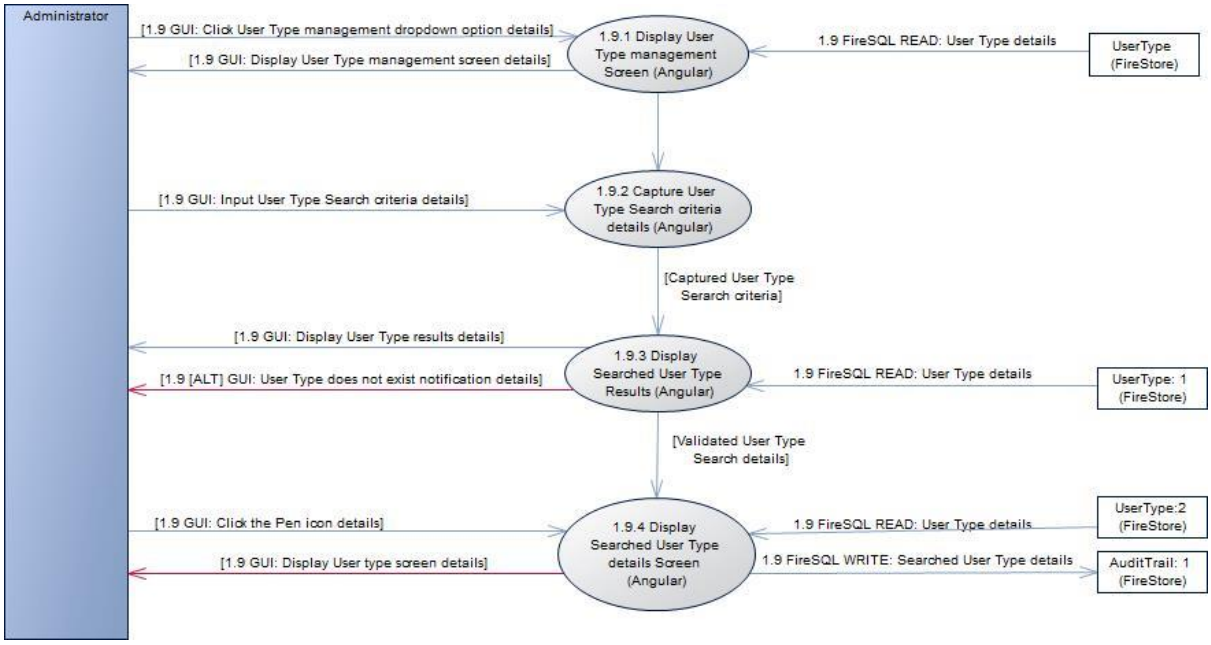


Figure 187 Primitive Diagram Sub-System 1: 1.9 Search User Type

1.10 Update User Type

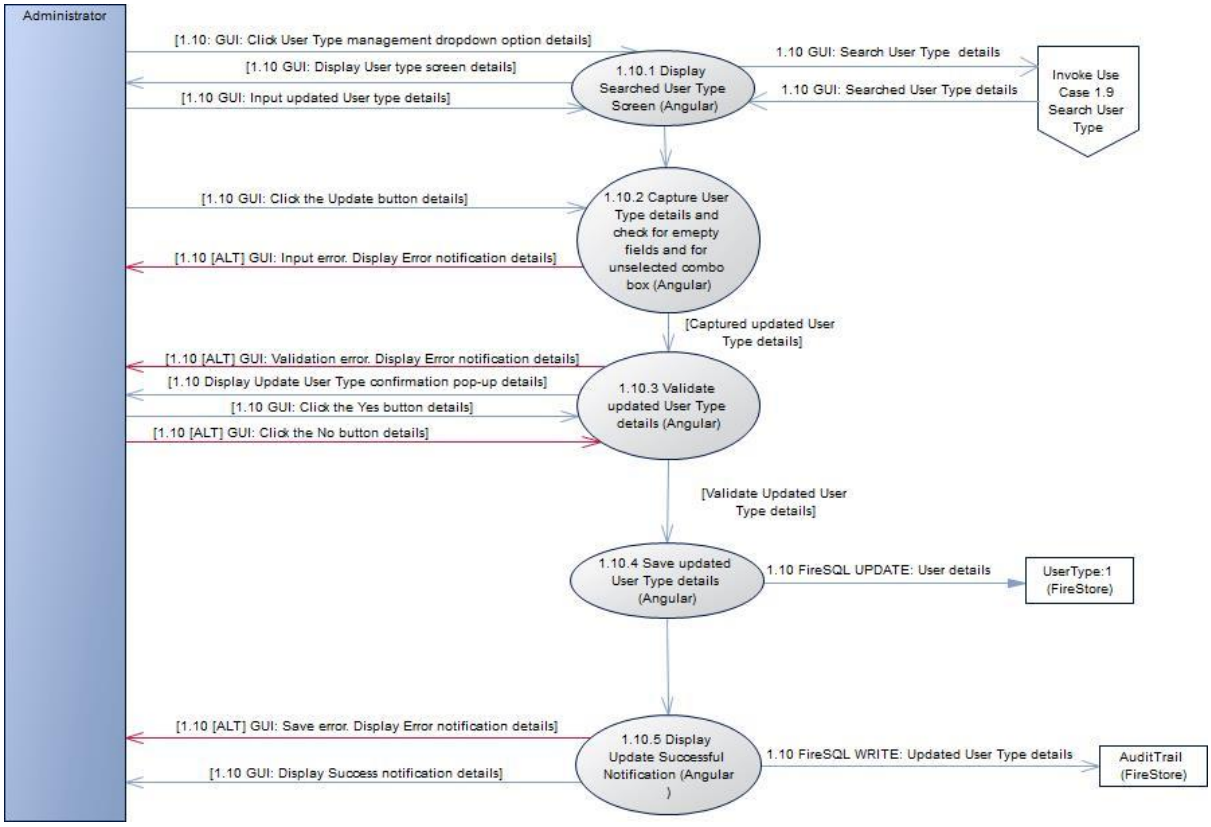


Figure 188 Primitive Diagram Sub-System 1: 1.10 Update User Type

1.11 Delete User Type

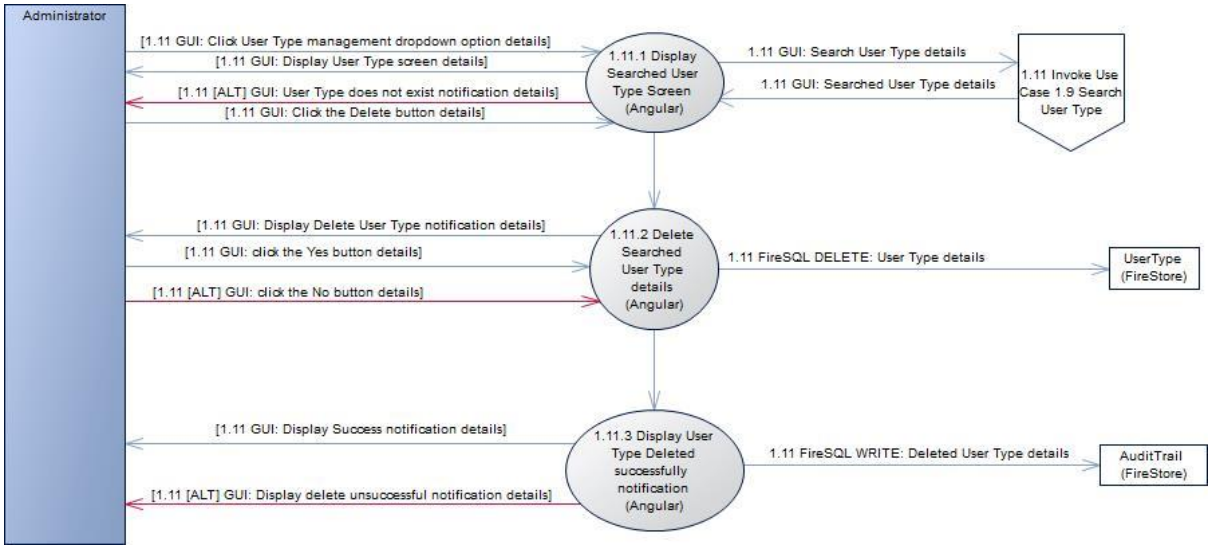


Figure 189 Primitive Diagram Sub-System 1: 1.11 Delete User Type

Sub-System 2: Mechanic Schedule

2.1 View Jobs

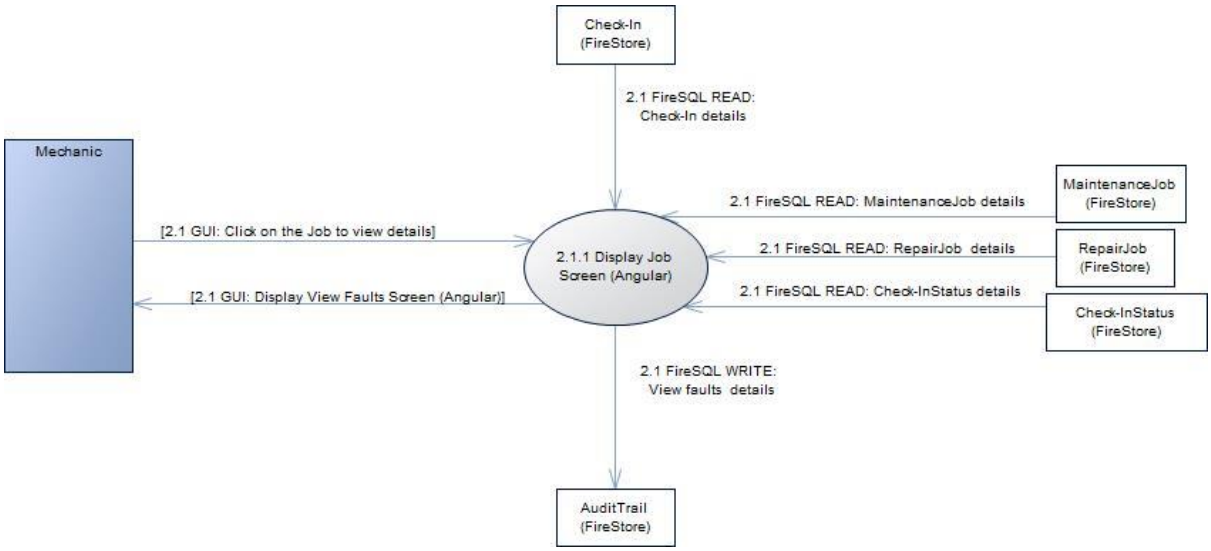


Figure 190 Primitive Diagram Sub-System 2: 2.1 View Jobs

2.2 View Job List

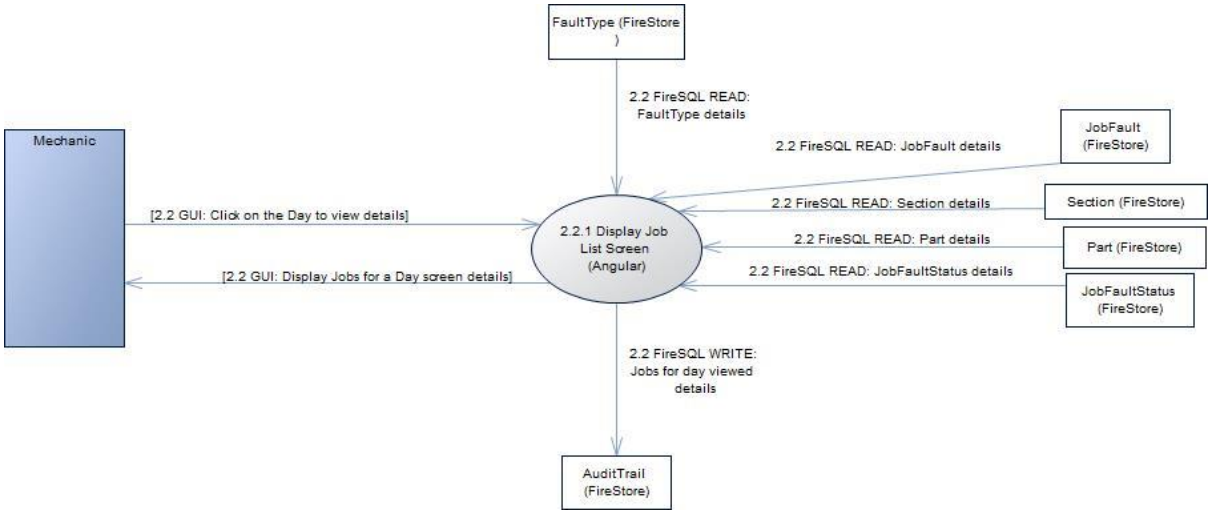


Figure 191 Primitive Diagram Sub-System 2: 2.1 View Job List

Sub-System 3: Tracking Hardware

3.1 Add Antenna

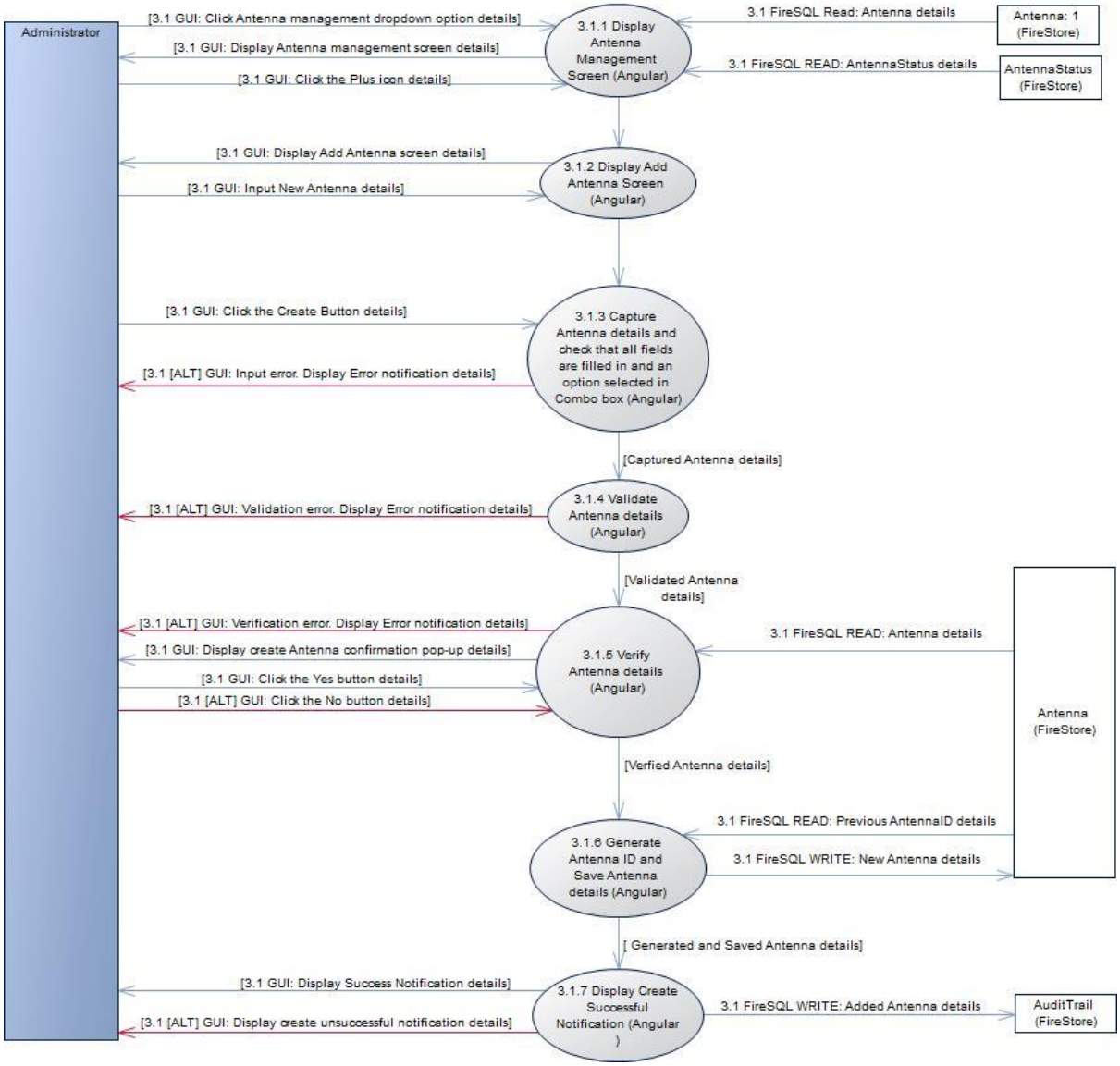


Figure 192 Primitive Diagram Sub-System 3: 3.1 Add Antenna

3.2 Search Antenna

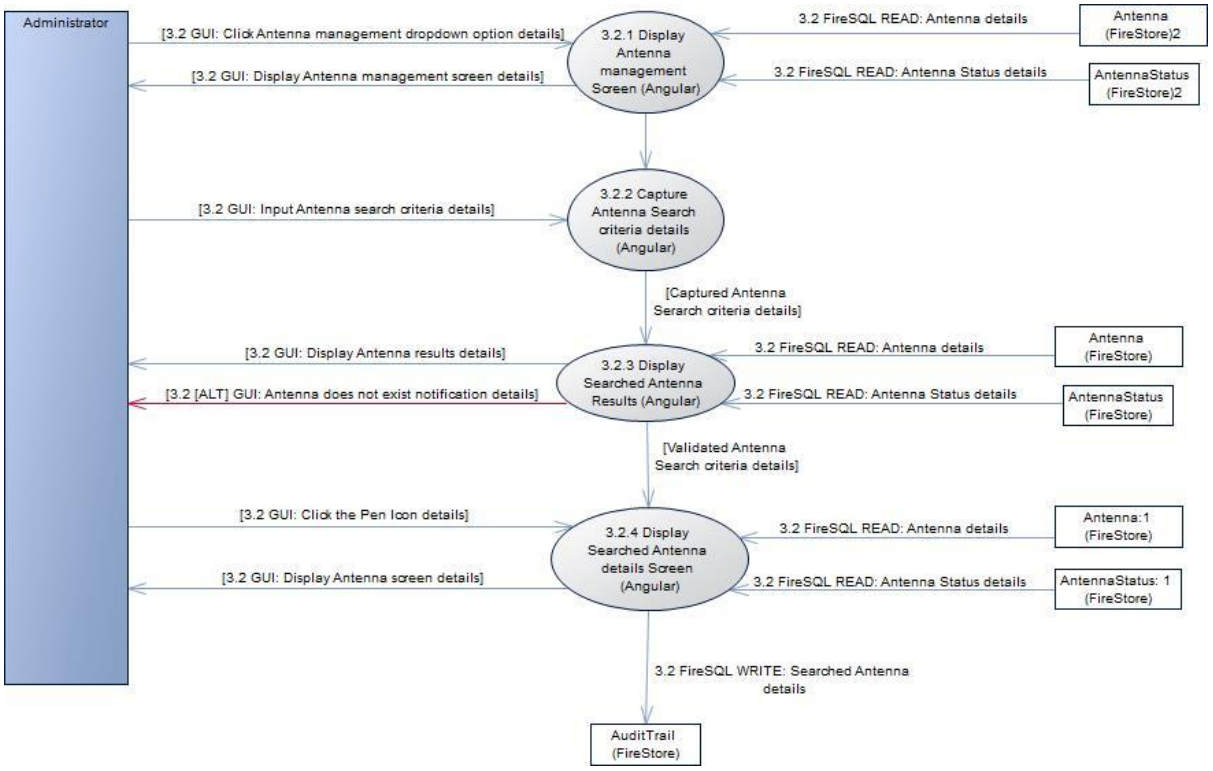


Figure 193 Primitive Diagram Sub-System 3: 3.2 Search Antenna

3.3 Update Antenna

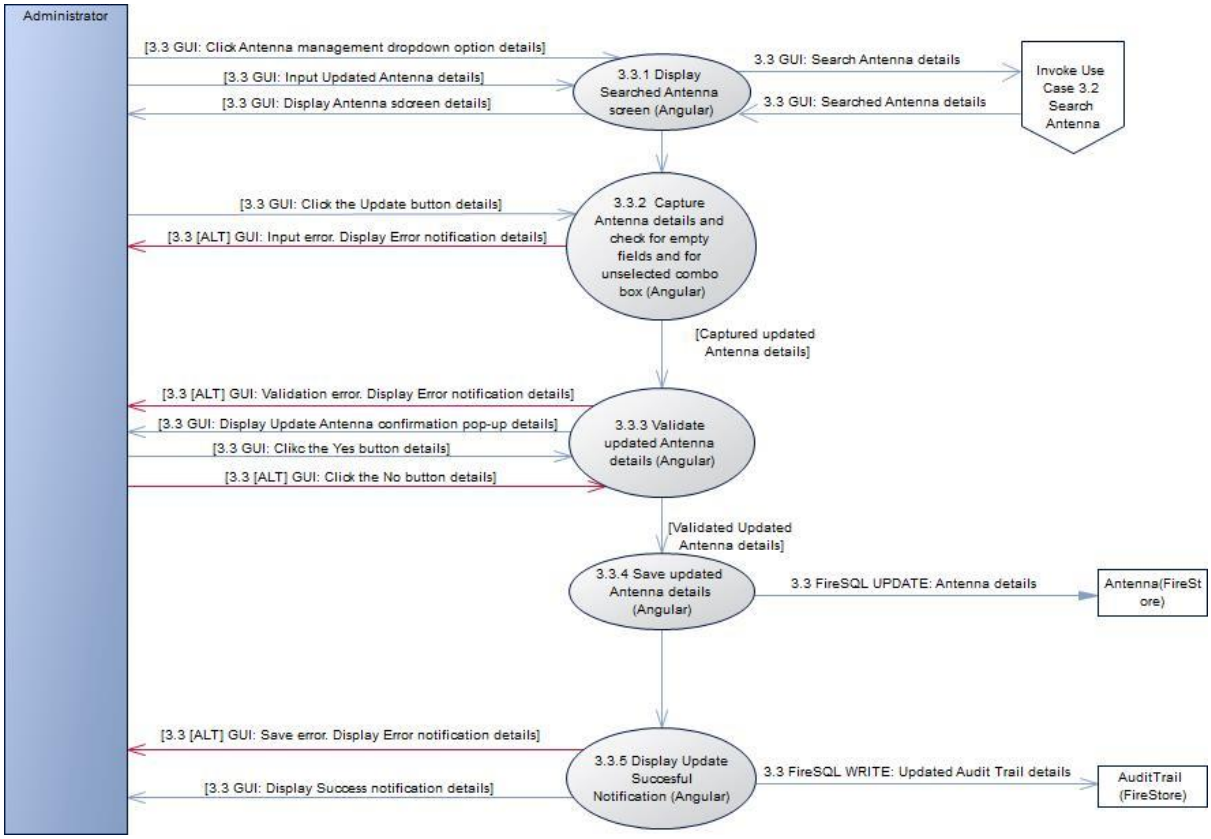


Figure 194 Primitive Diagram Sub-System 3: 3.3 Update Antenna

3.4 Delete Antenna

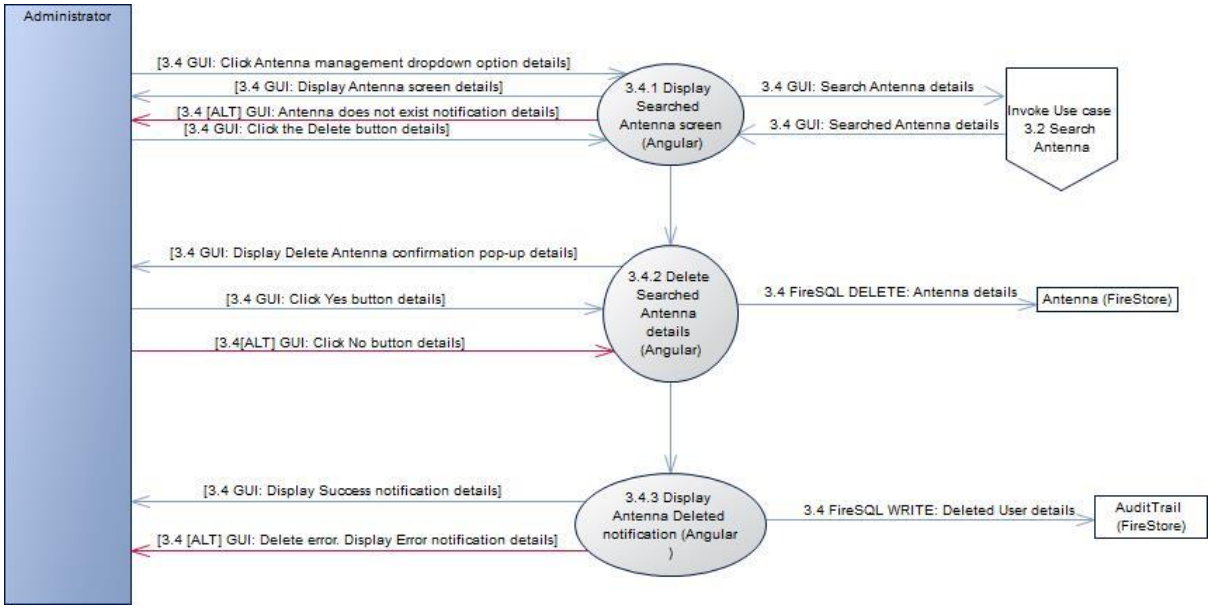


Figure 195 Primitive Diagram Sub-System 3: 3.4 Delete Antenna

3.5 Add Tag

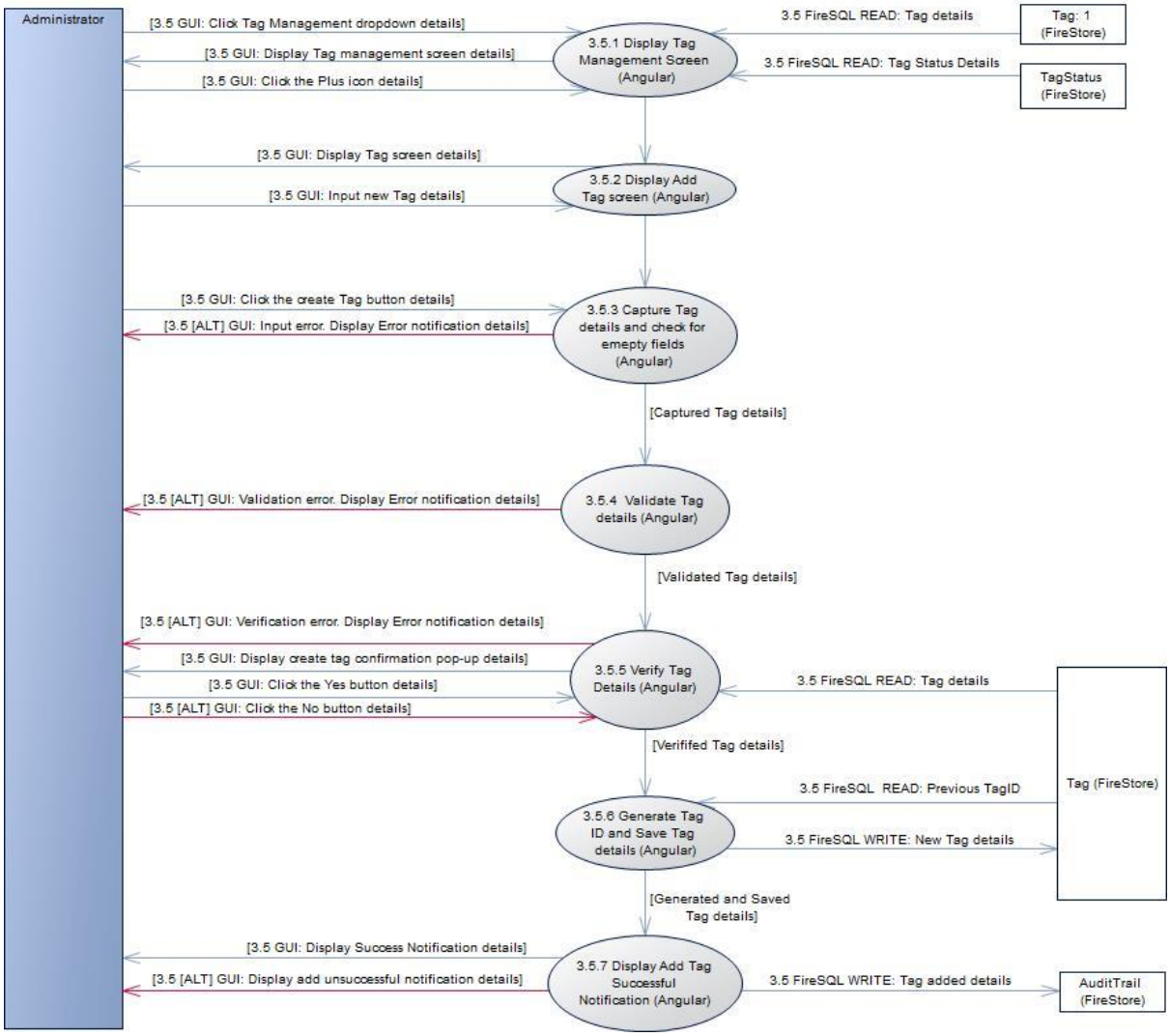


Figure 196 Primitive Diagram Sub-System 3: 3.5 Add Tag

3.6 Search Tag

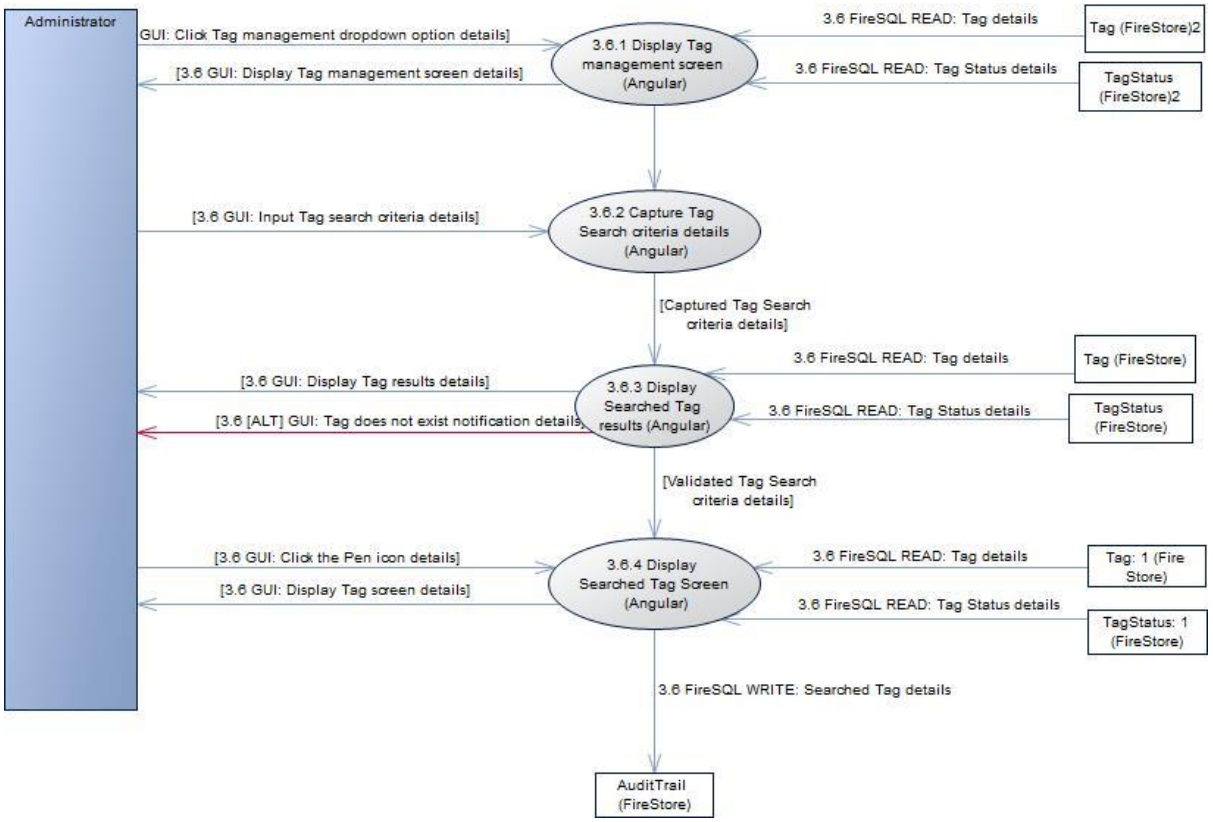


Figure 197 Primitive Diagram Sub-System 3: 3.6 Search Tag

3.7 Update Tag

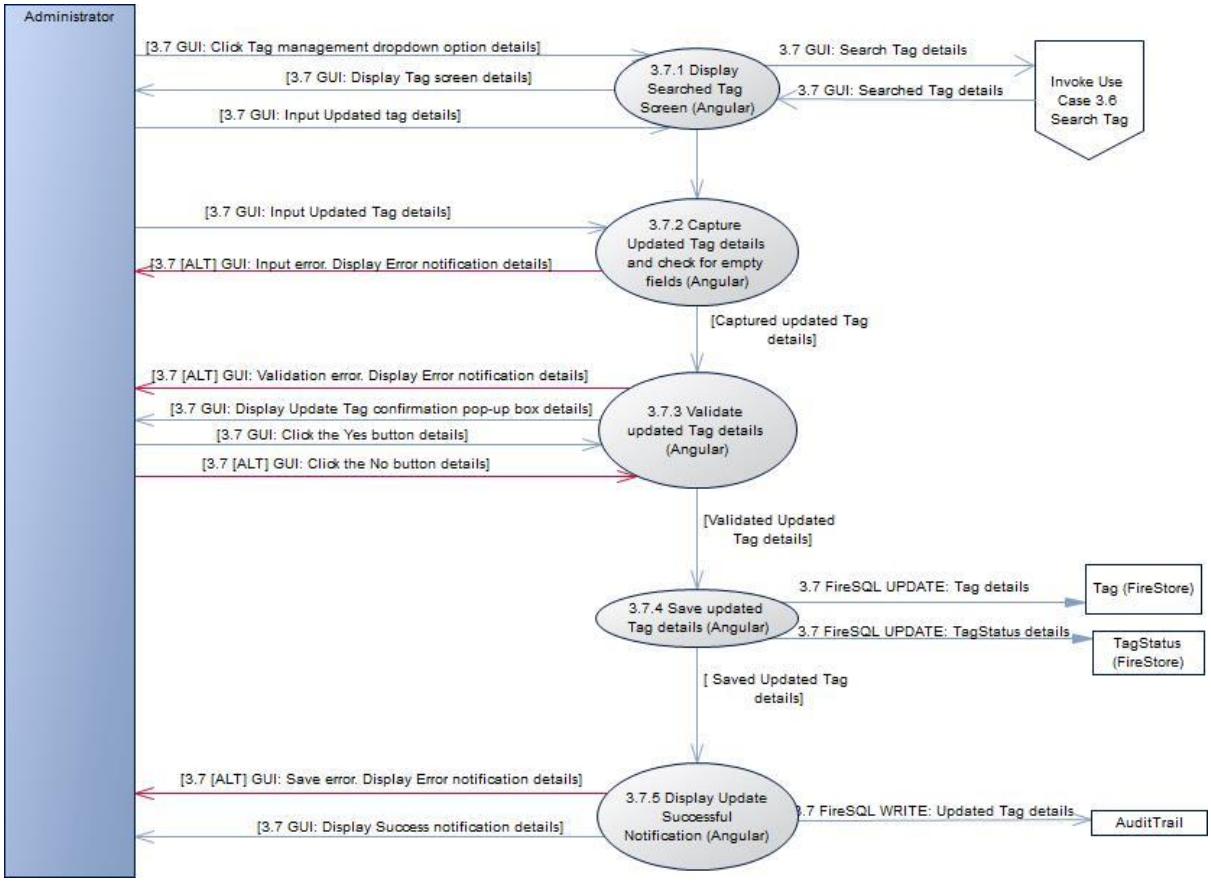


Figure 198 Primitive Diagram Sub-System 3: 3.7 Update Tag

3.8 Delete Tag

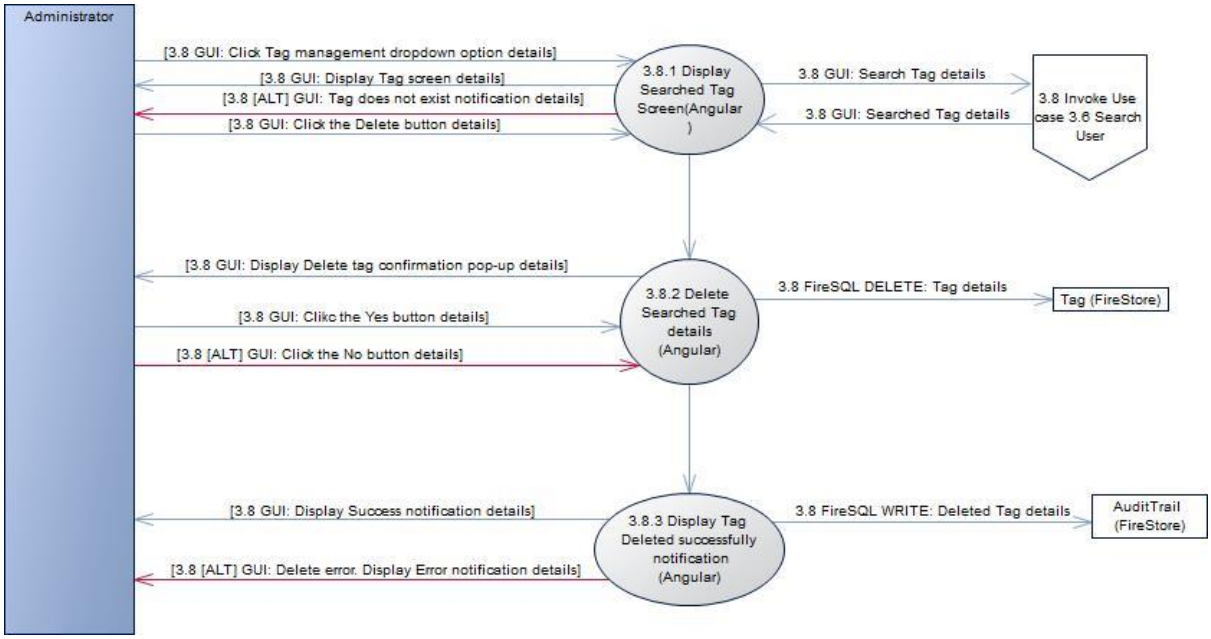


Figure 199 Primitive Diagram Sub-System 3: 3.8 Delete tag

Sub-System 4: Student

4.1 Add Student

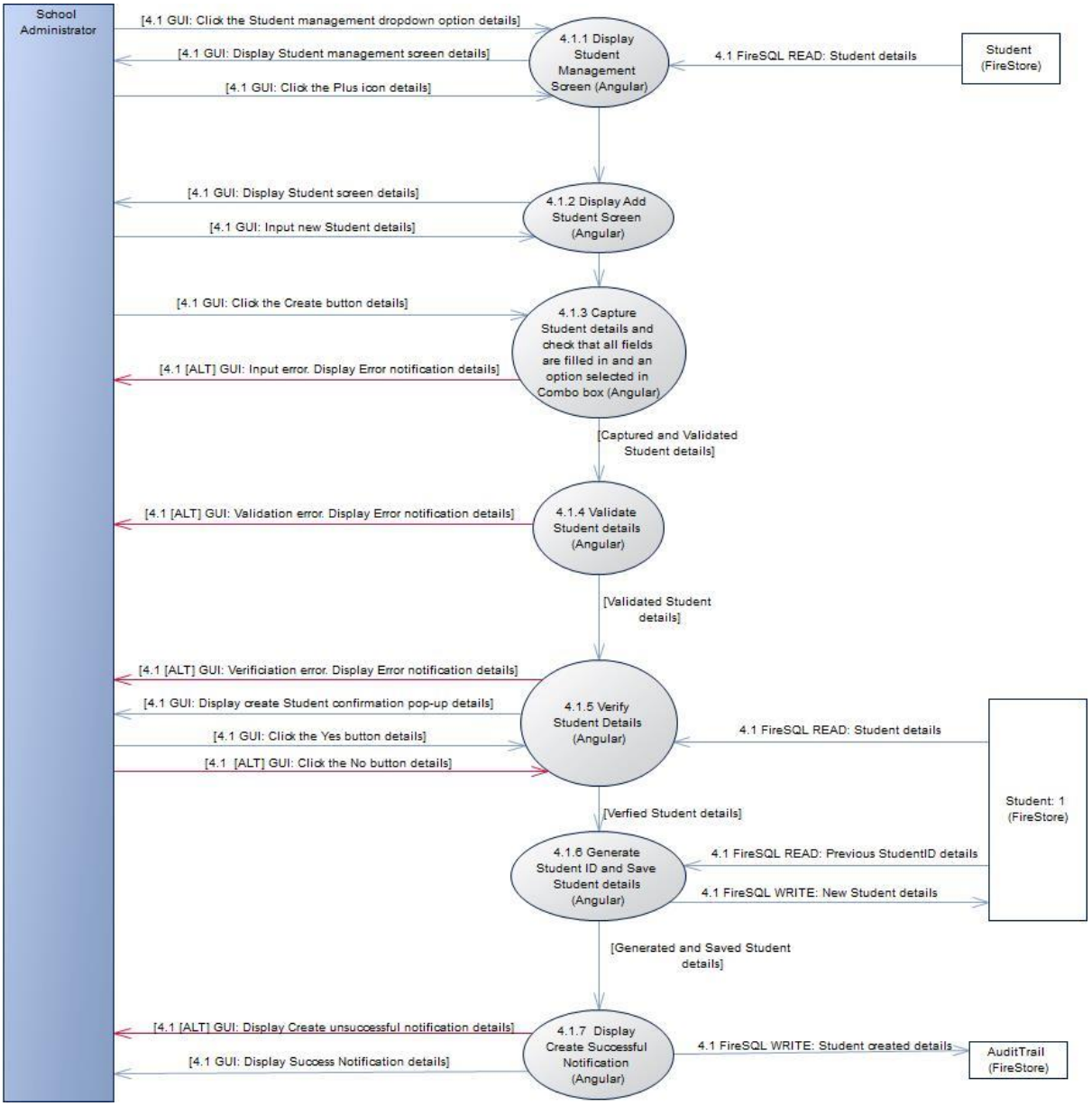


Figure 200 Primitive Diagram Sub-System 4: 4.1 Add Student

4.2 Search Student

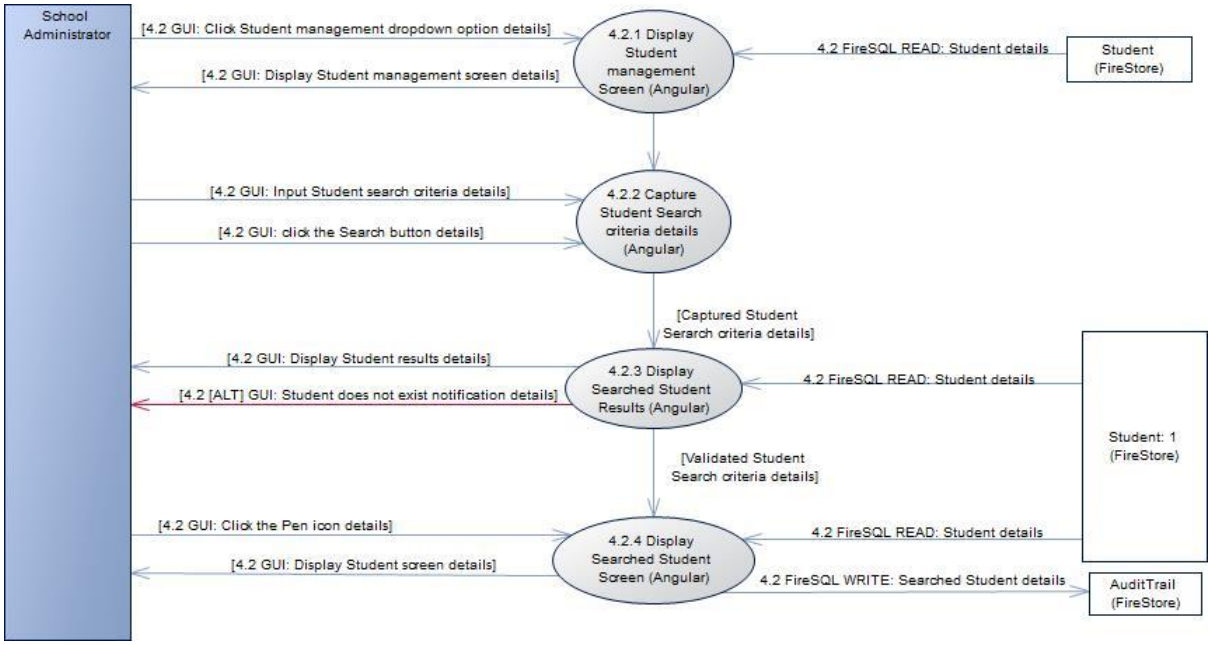


Figure 201 Primitive Diagram Sub-System 4: 4.2 Search Student

4.3 Update Student

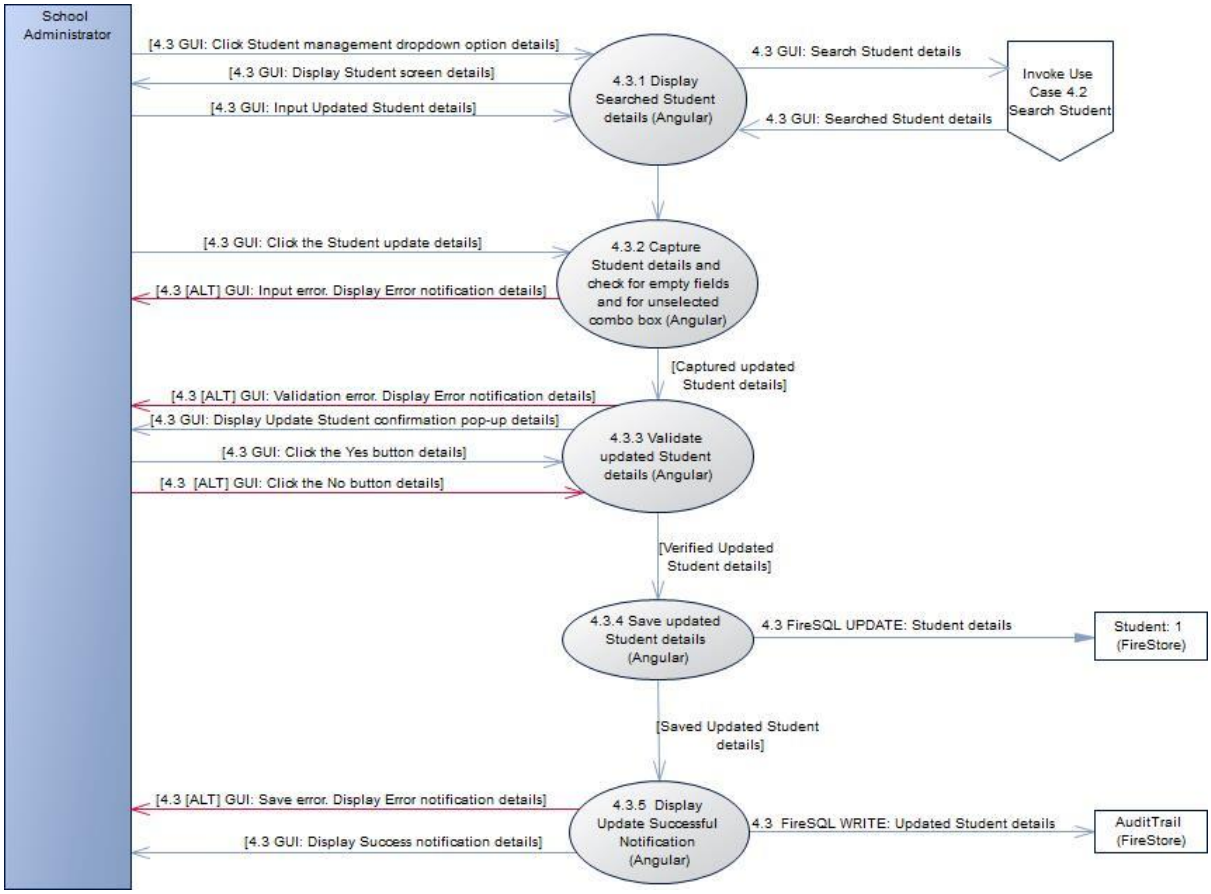


Figure 202 Primitive Diagram Sub-System 4: 4.3 Update Student

4.4 Delete Student

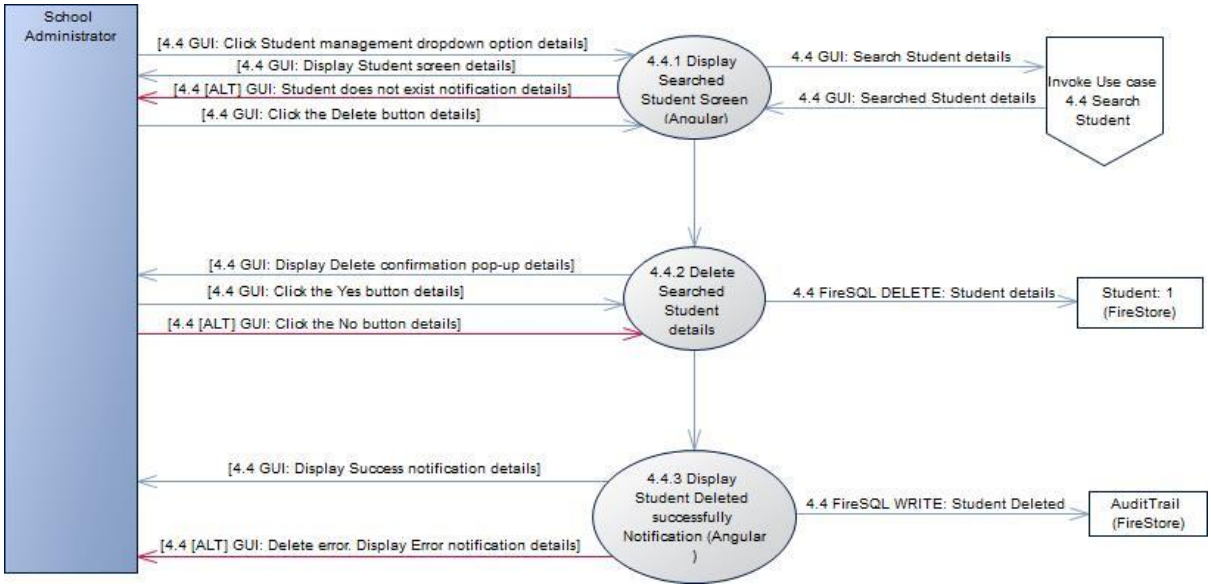


Figure 203 Primitive Diagram Sub-System 4: 4.4 Delete Student

4.5 Capture Student Mark

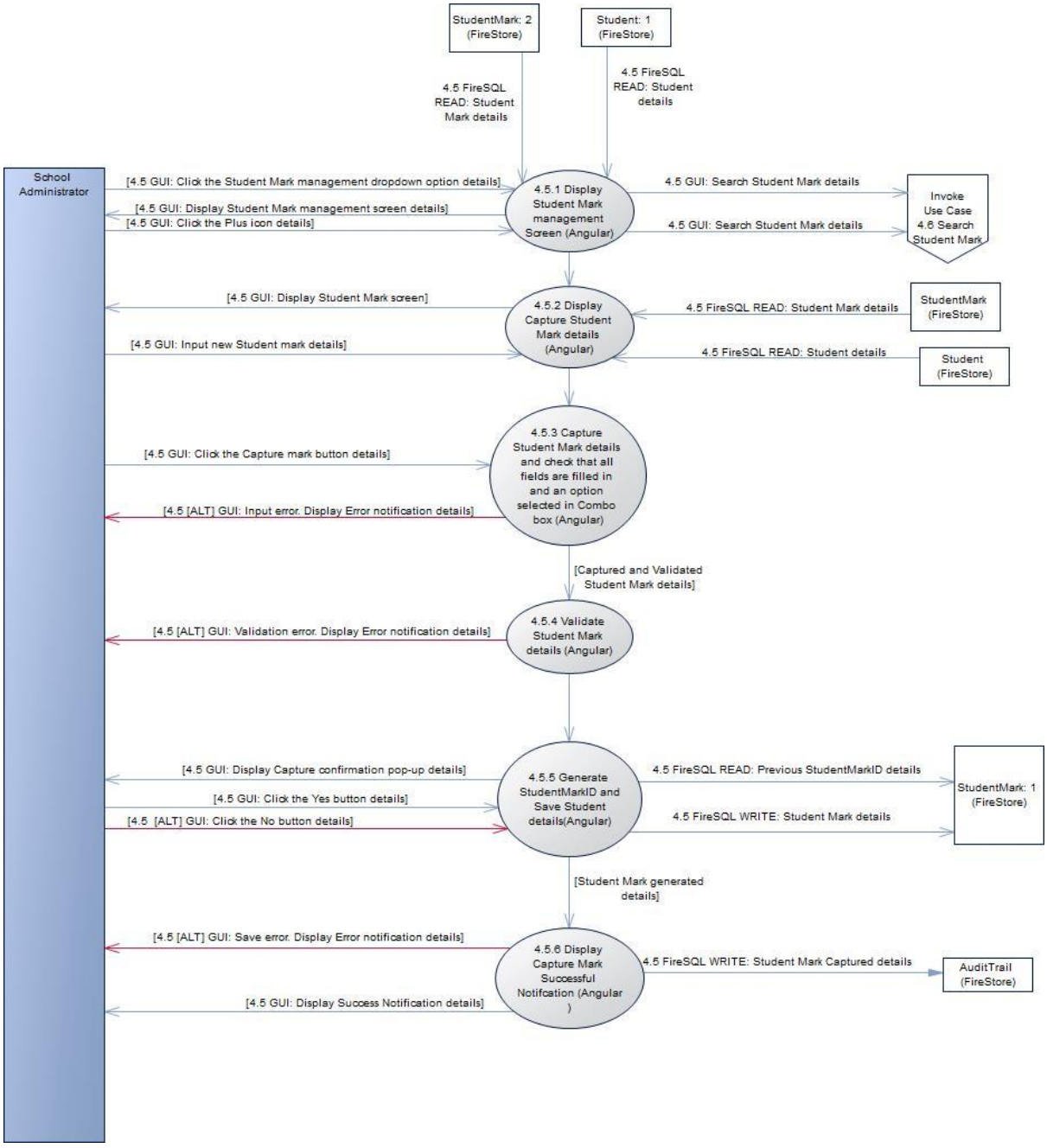


Figure 204 Primitive Diagram Sub-System 4: 4.5 Capture Student Mark

4.6 Search Student Mark

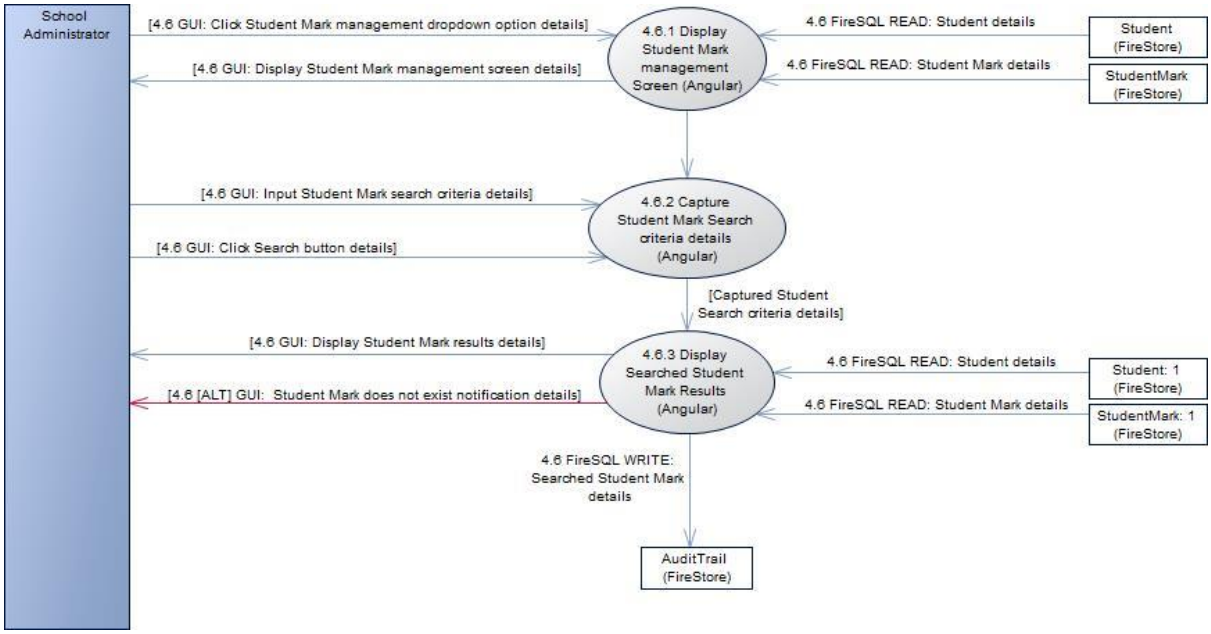


Figure 205 Primitive Diagram Sub-System 4: 4.6 Search Student Mark

4.7 Add Mark Type

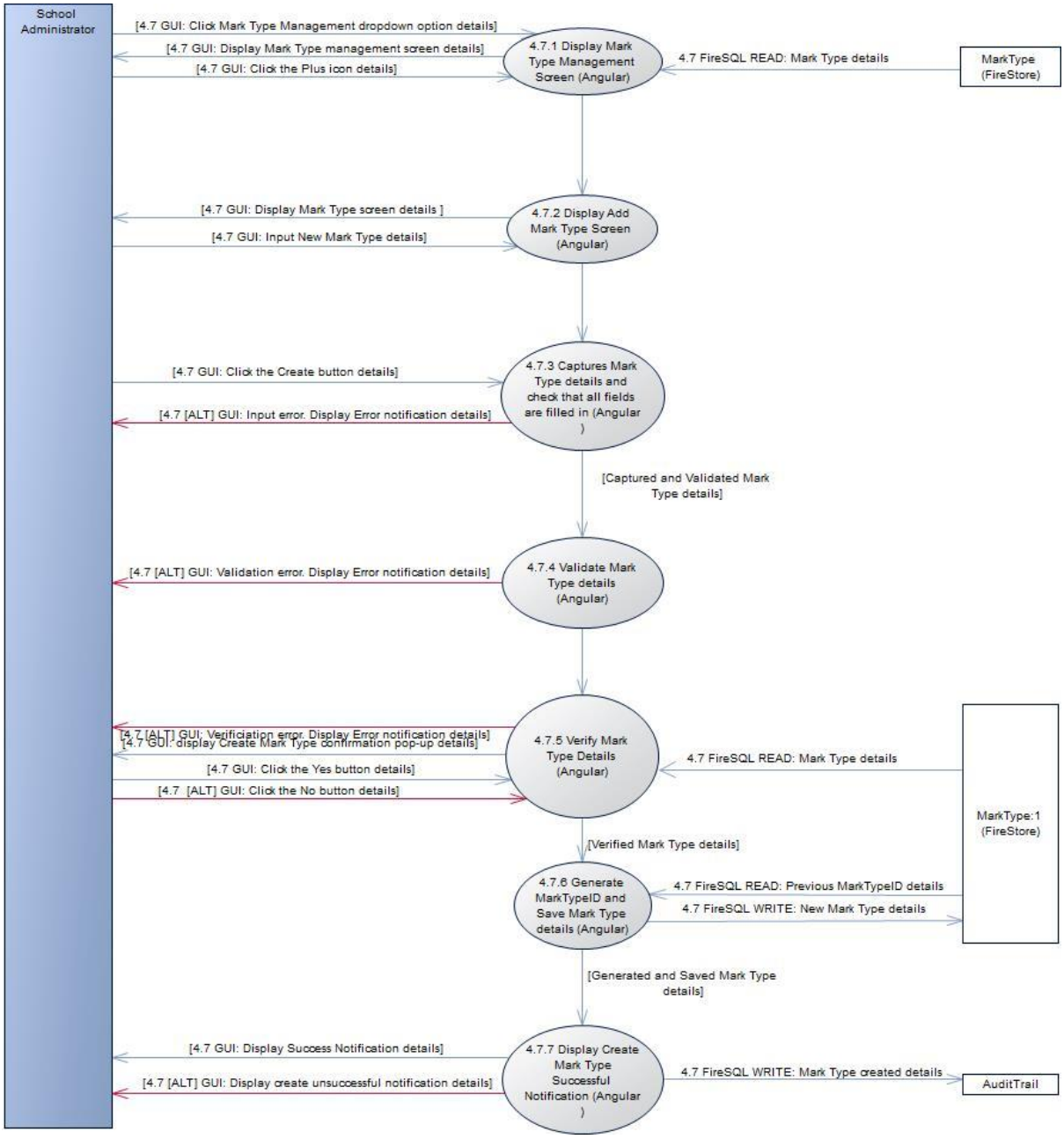


Figure 206 Primitive Diagram Sub-System 4: 4.7 Add Mark Type

4.8 Search Mark Type

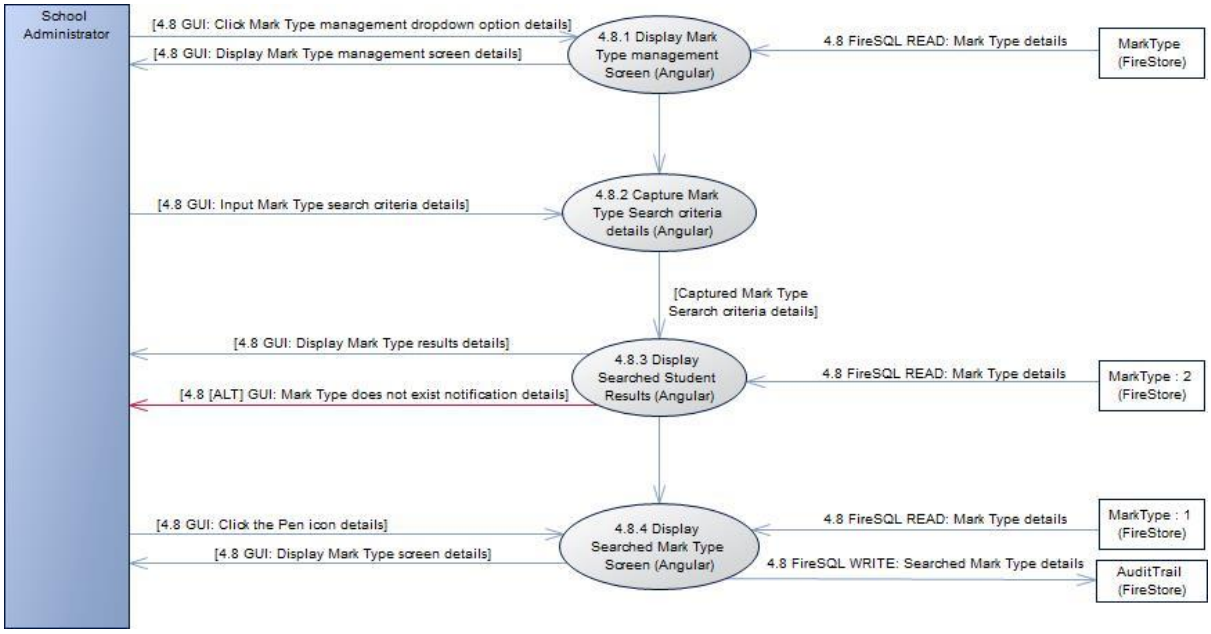


Figure 207 Primitive Diagram Sub-System 4: 4.8 Search Mark Type

4.9 Update Mark Type

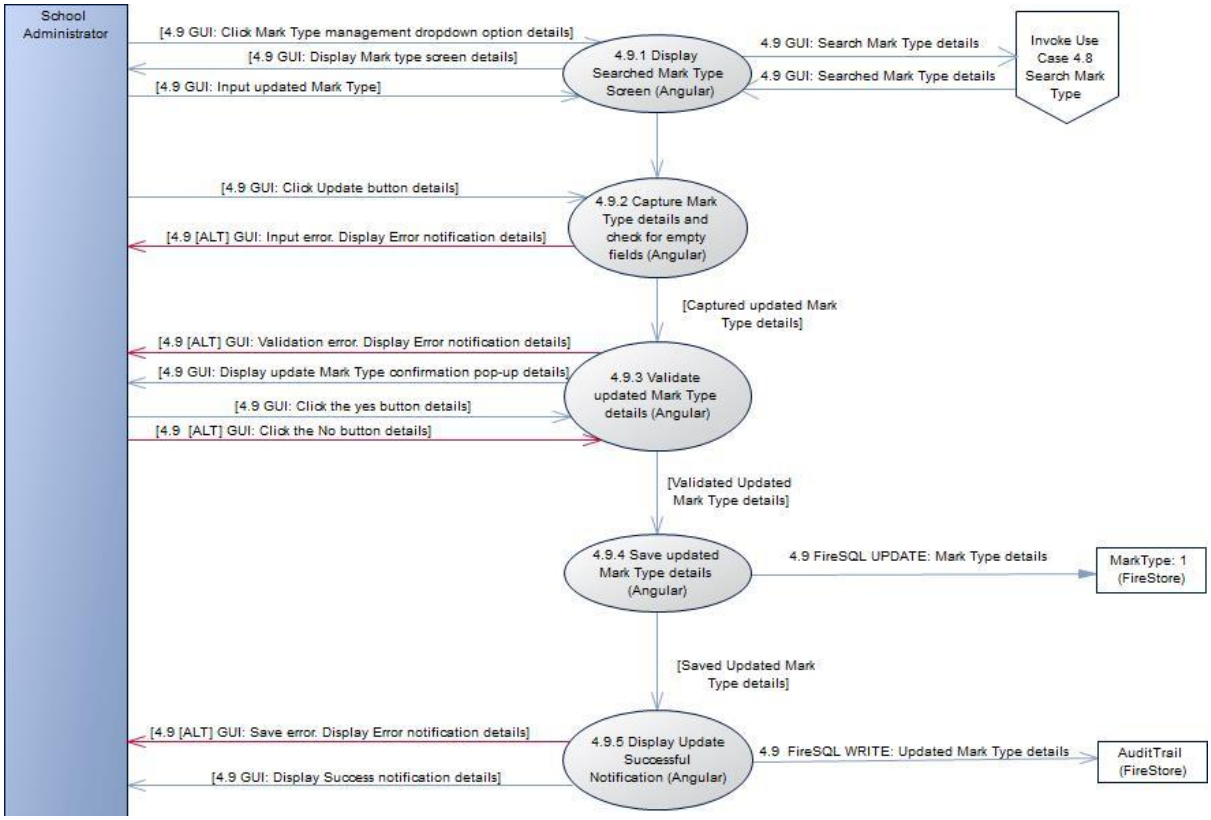


Figure 208 Primitive Diagram Sub-System 4: 4.9 Update Mark Type

4.10 Delete Mark Type

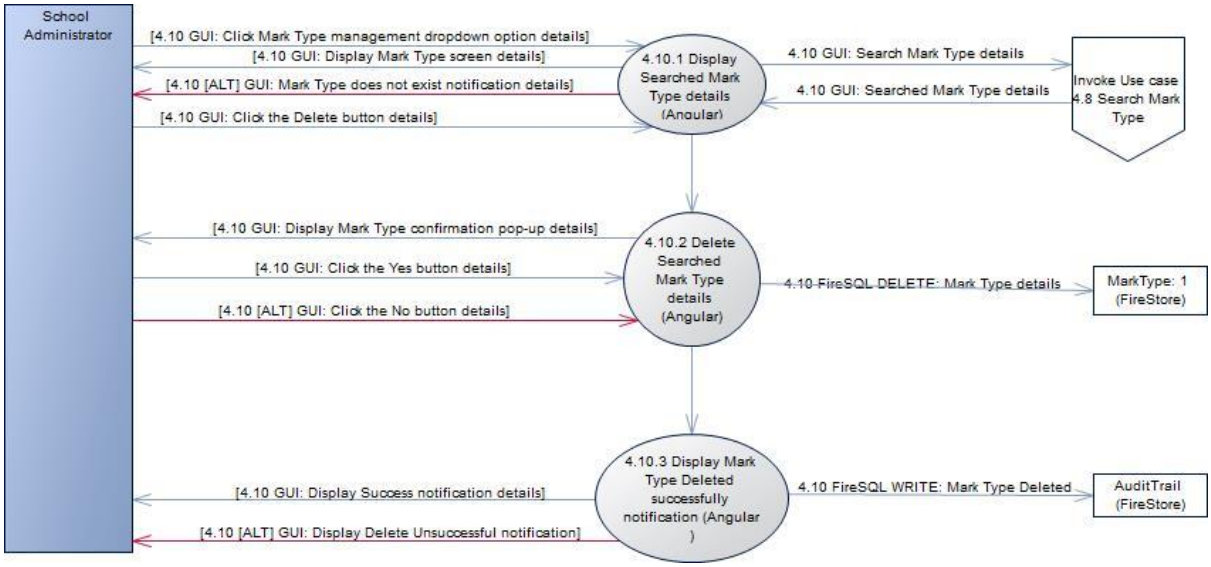


Figure 209 Primitive Diagram Sub-System 4: 4.10 Delete Mark Type

Sub-System 5: Bicycle

5.1 Add Bicycle

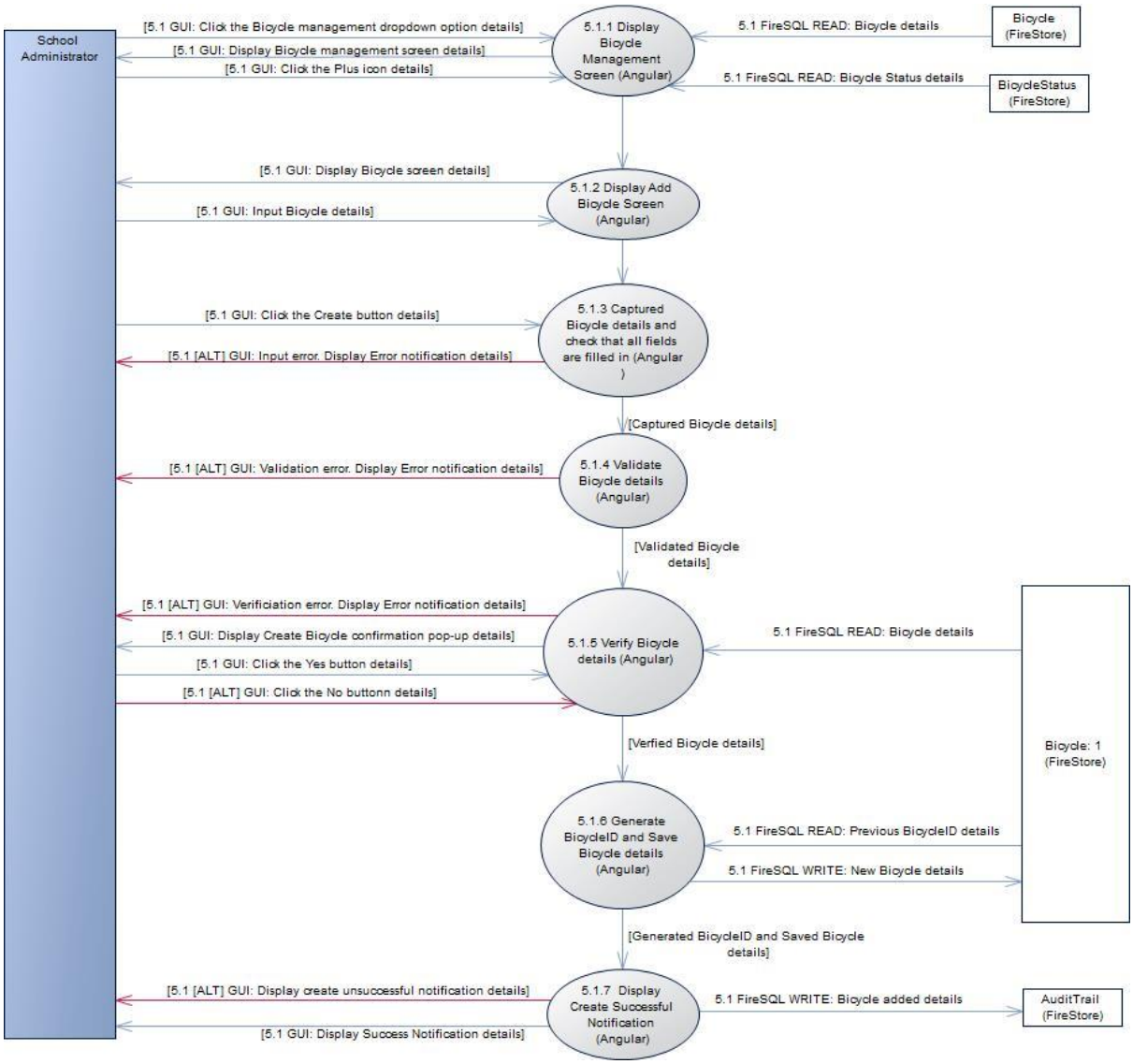


Figure 210 Primitive Diagram Sub-System 5: 5.1 Add Bicycle

5.2 Search Bicycle

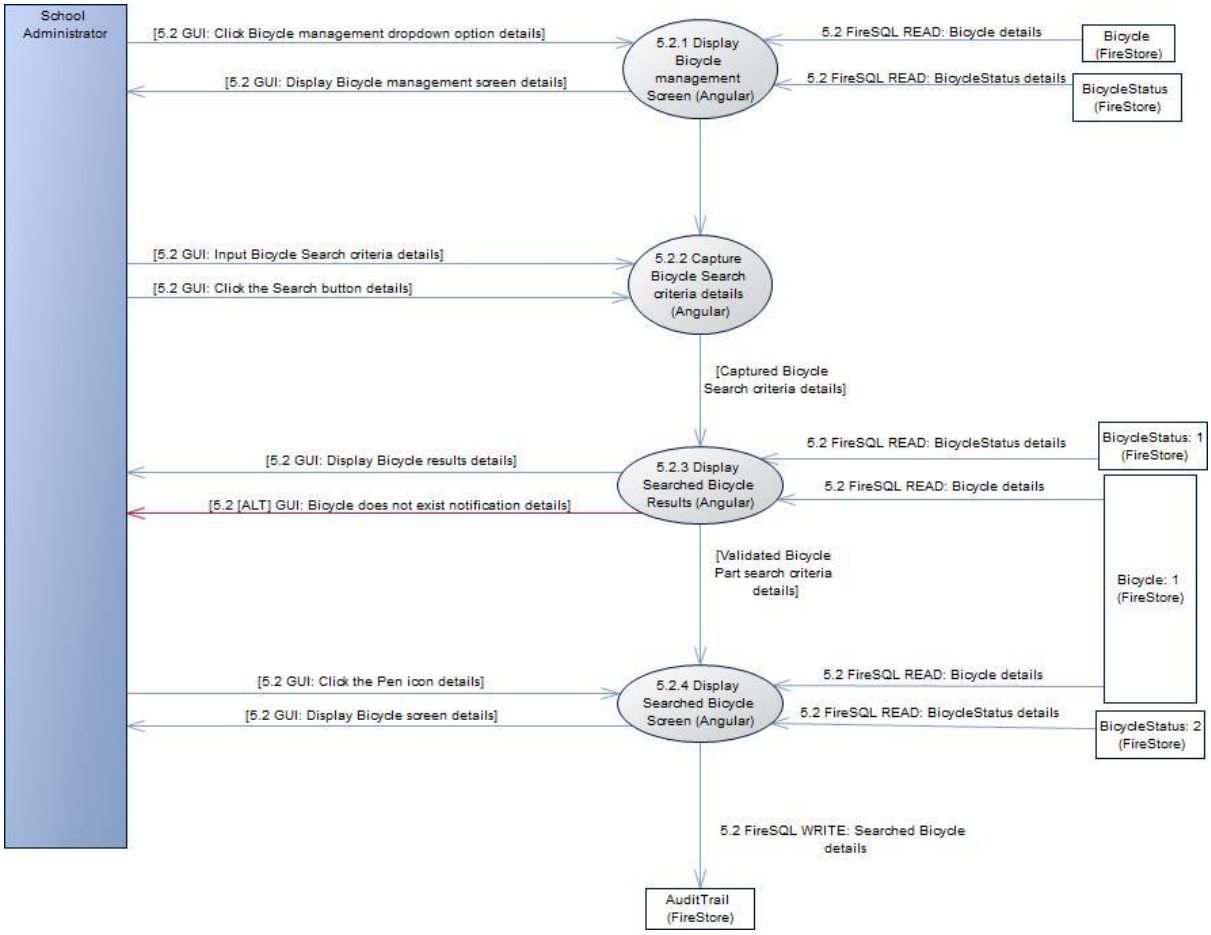


Figure 211 Primitive Diagram Sub-System 5: 5.2 Search Bicycle

5.3 Update Bicycle

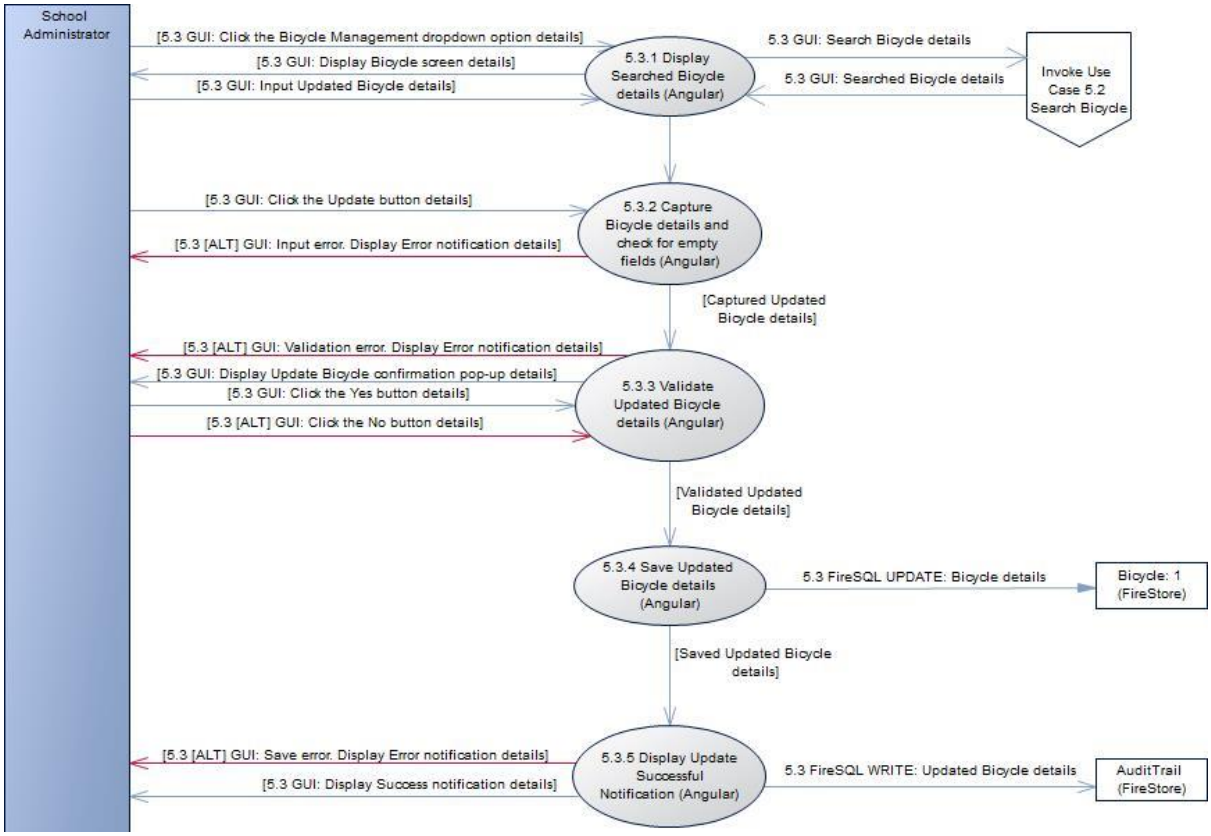


Figure 212 Primitive Diagram Sub-System 5: 5.3 Update Bicycle

5.4 Delete Bicycle

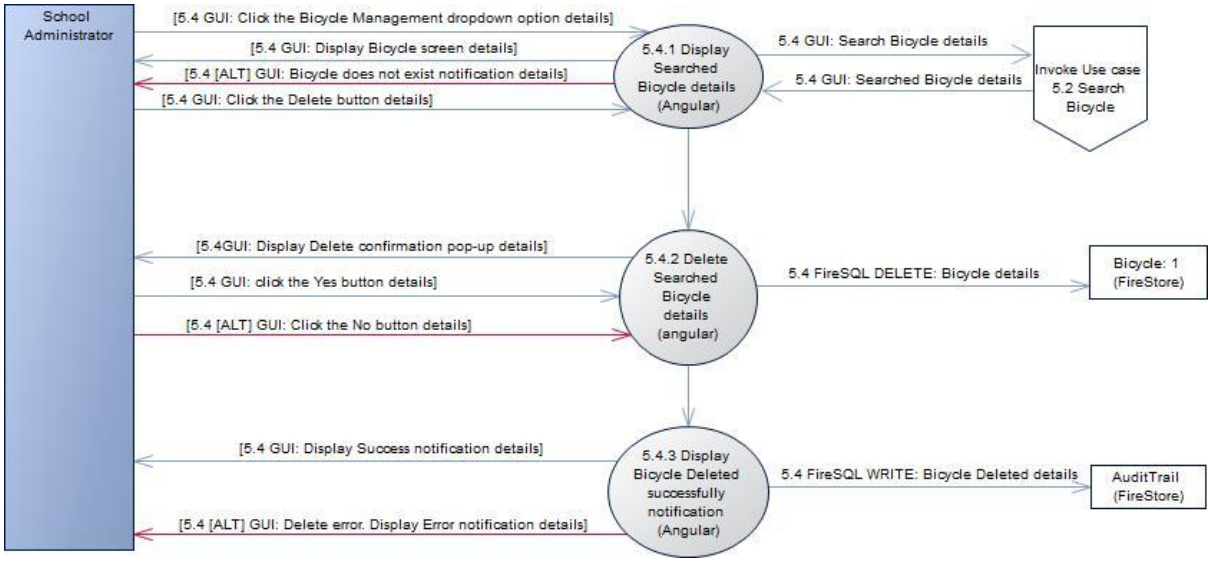


Figure 213 Primitive Diagram Sub-System 5: 5.4 Delete Bicycle

5.5 Assign Bicycle to Student

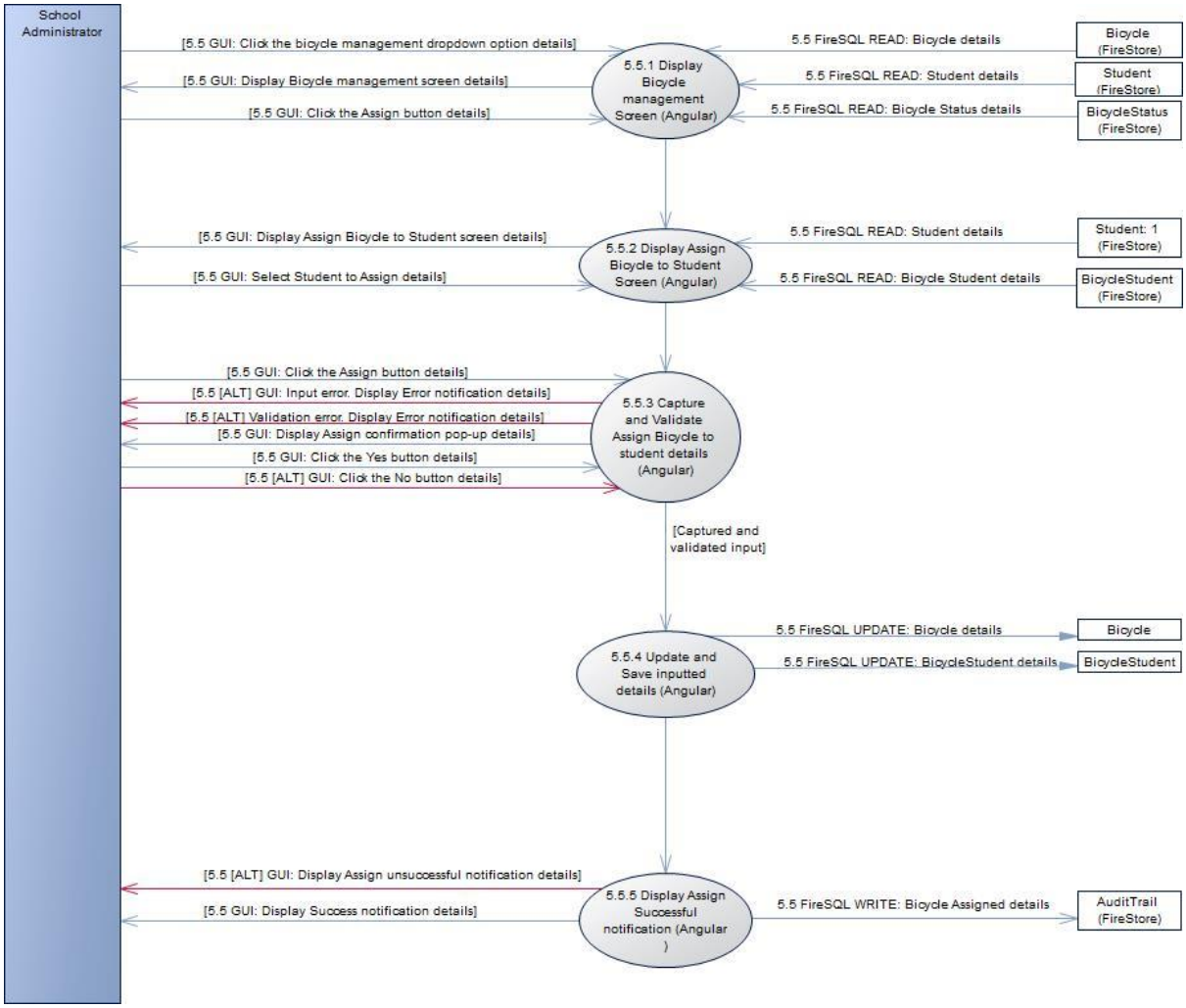


Figure 214 Primitive Diagram Sub-System 5: 5.5 Assign Bicycle to Student

5.6 Unassign Bicycle from Student

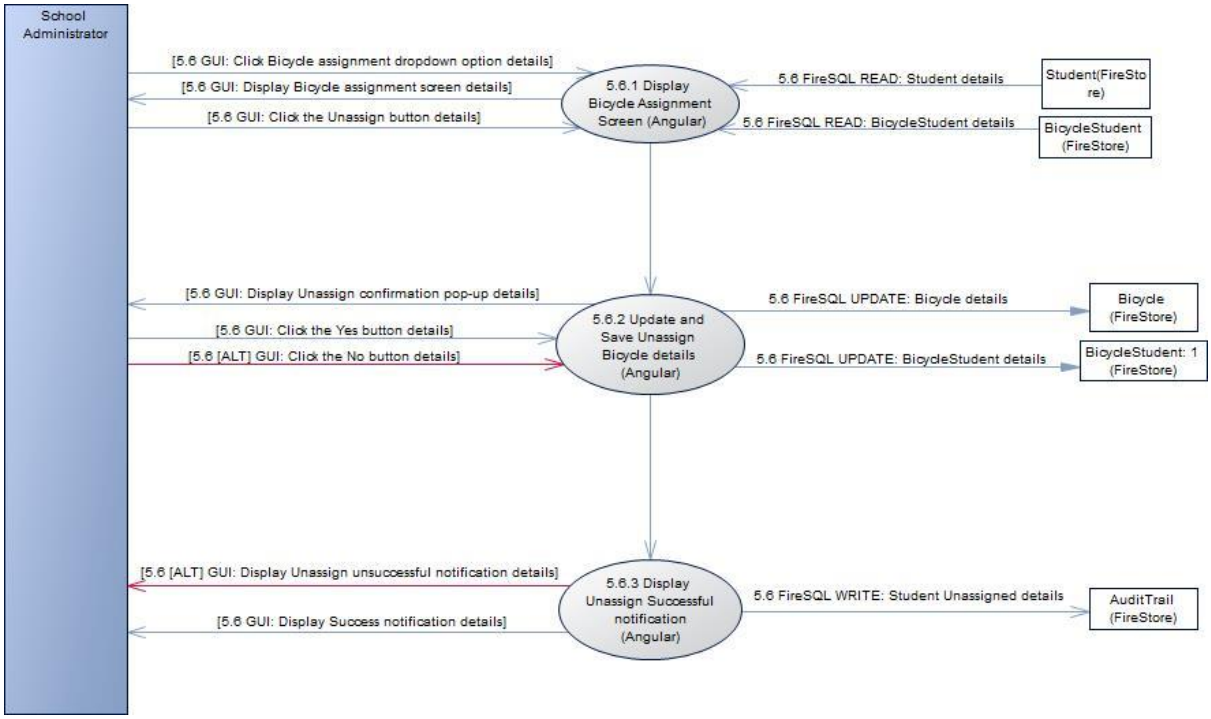


Figure 215 Primitive Diagram Sub-System 5: 5.6 Unassign Bicycle from Student

5.7 Assign Tag to Bicycle

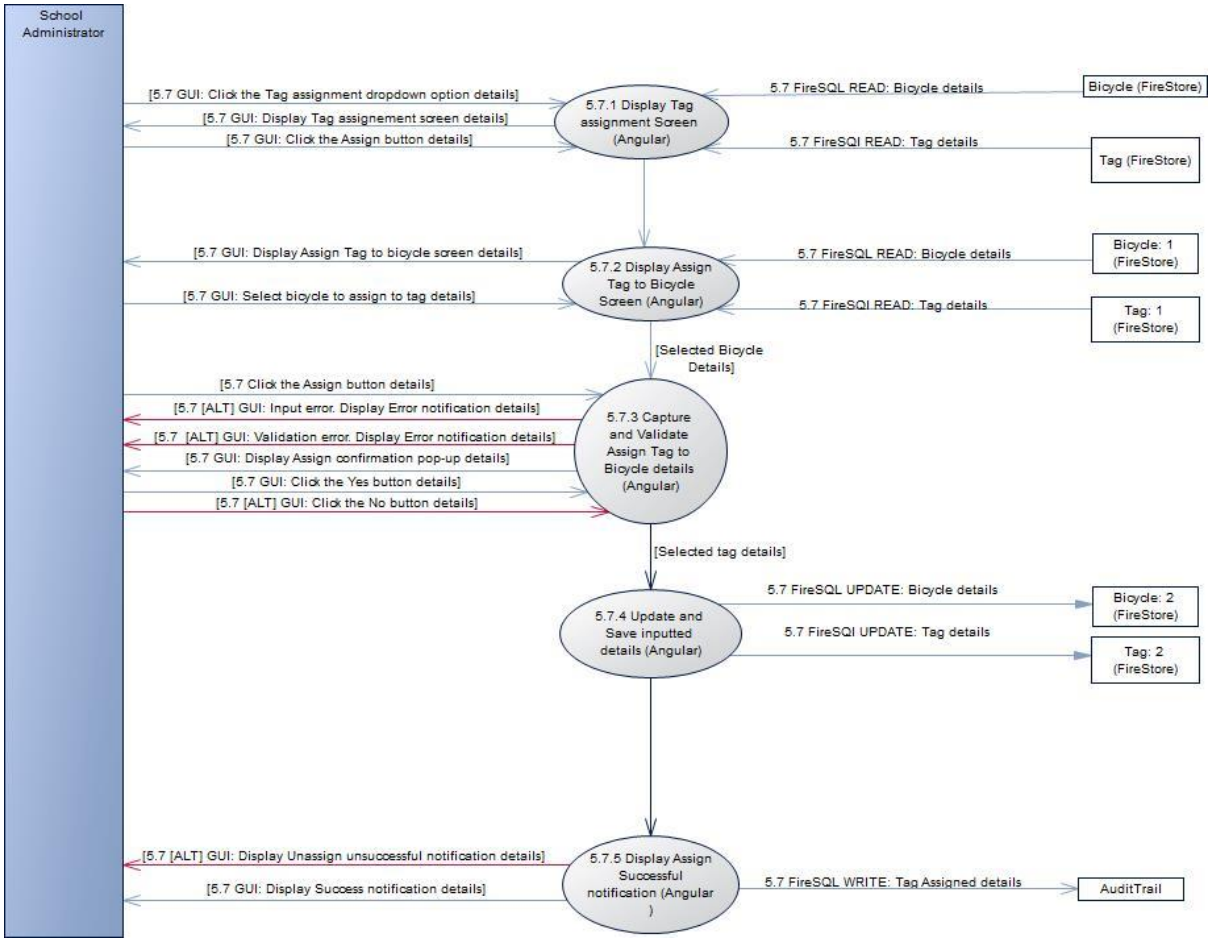


Figure 216 Primitive Diagram Sub-System 5: 5.7 Assign tag to Bicycle

5.8 Unassign Tag from Bicycle

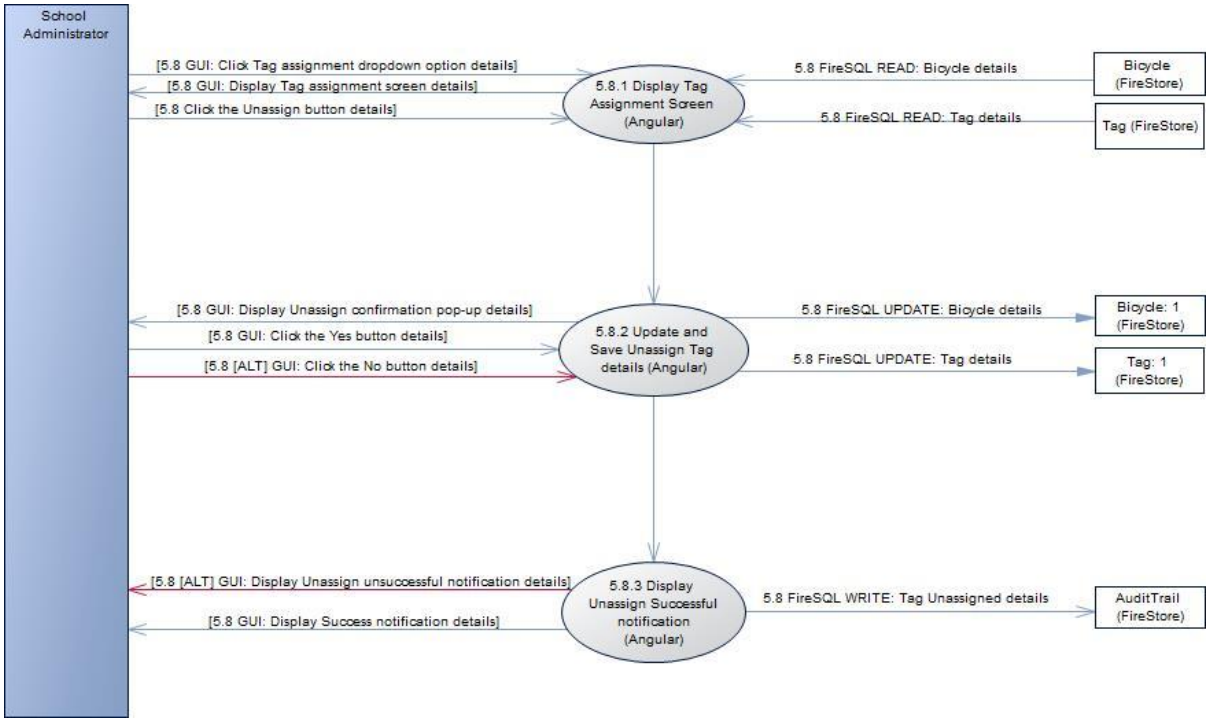


Figure 217 Primitive Diagram Sub-System 5: 5.8 Unassign Tag from Bicycle

Sub-System 6: Bicycle Part

6.1 Add Bicycle Part

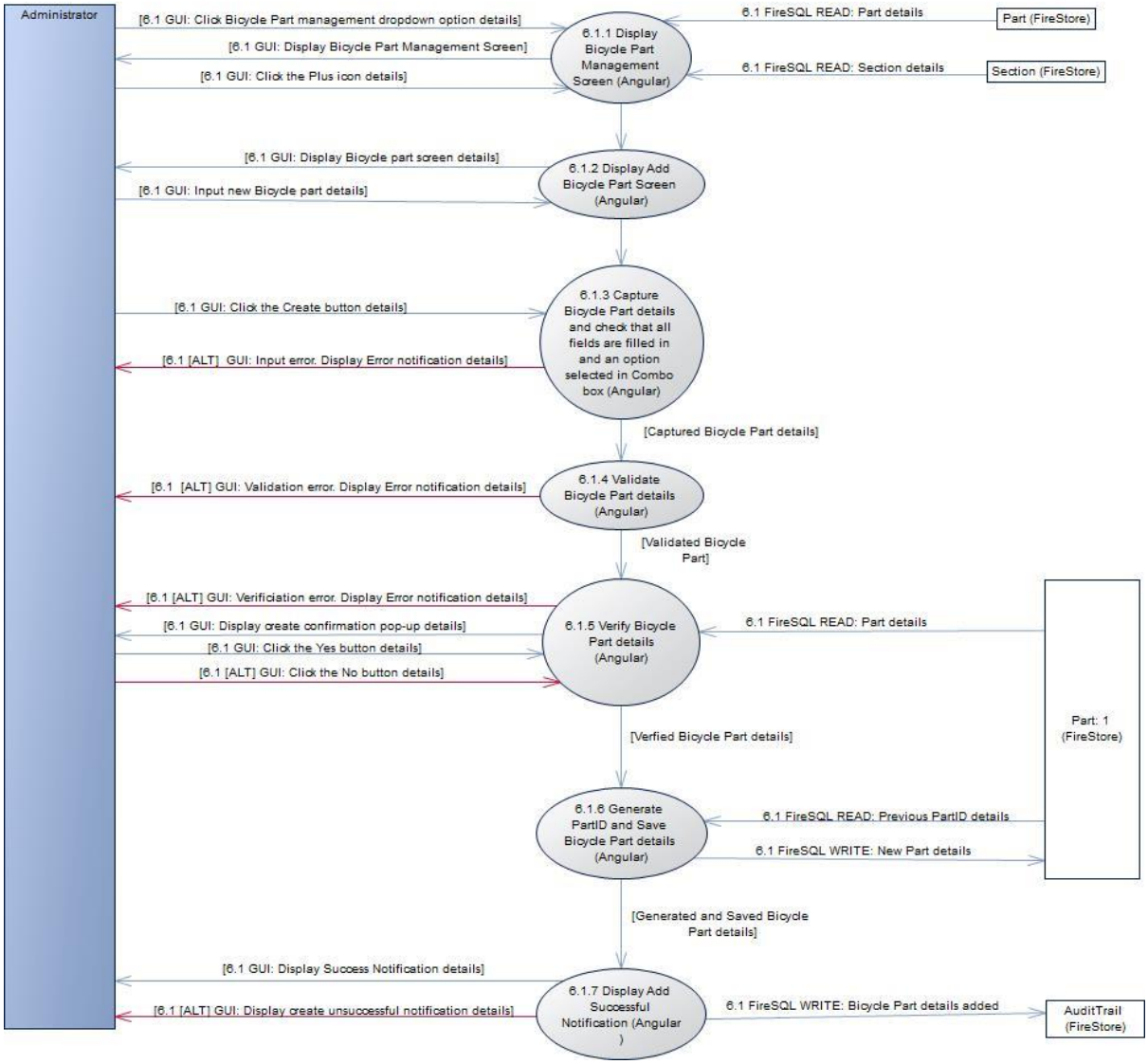


Figure 218 Primitive Diagram Sub-System 6: 6.1 Add bicycle Part

6.2 Search Bicycle Part

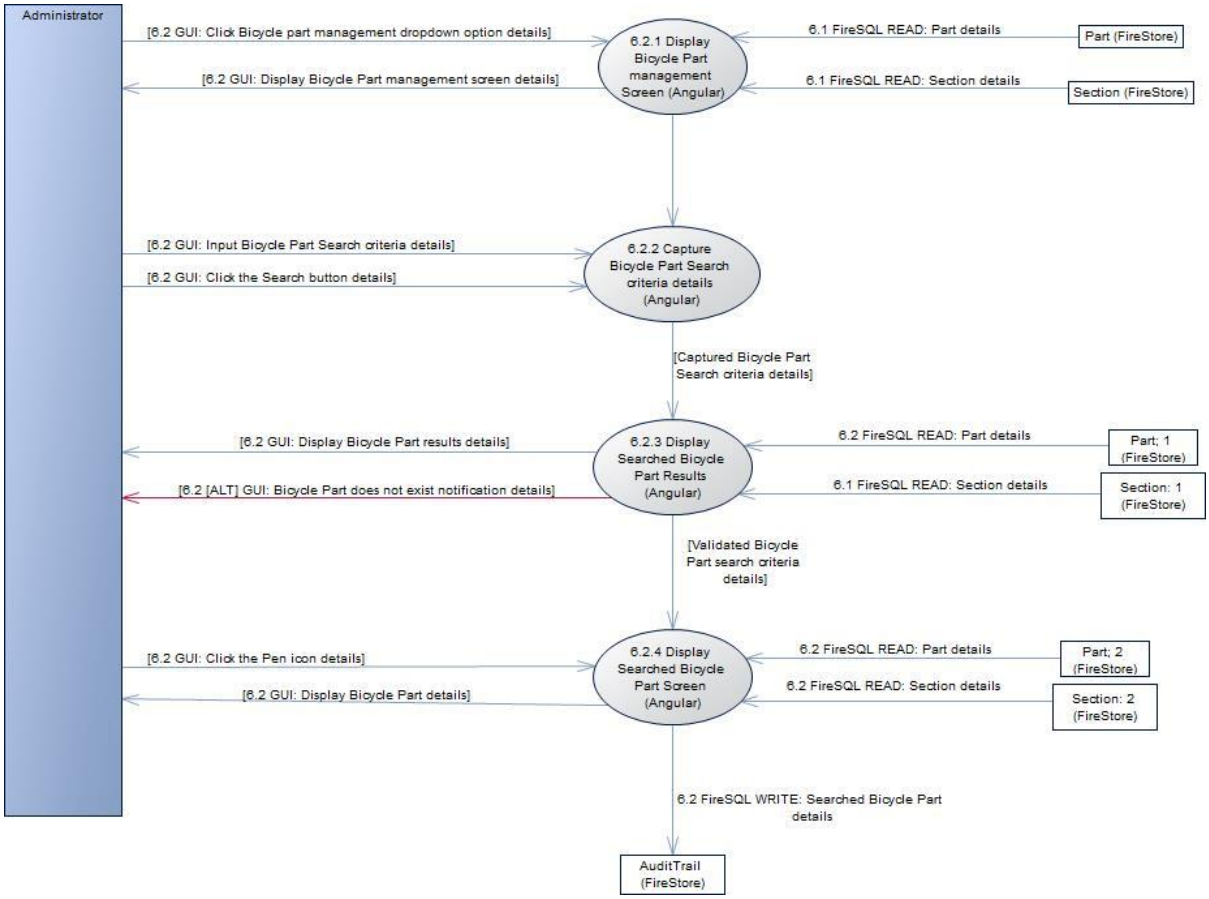


Figure 219 Primitive Diagram Sub-System 6: 6.2 Search Bicycle Part

6.3 Update Bicycle Part

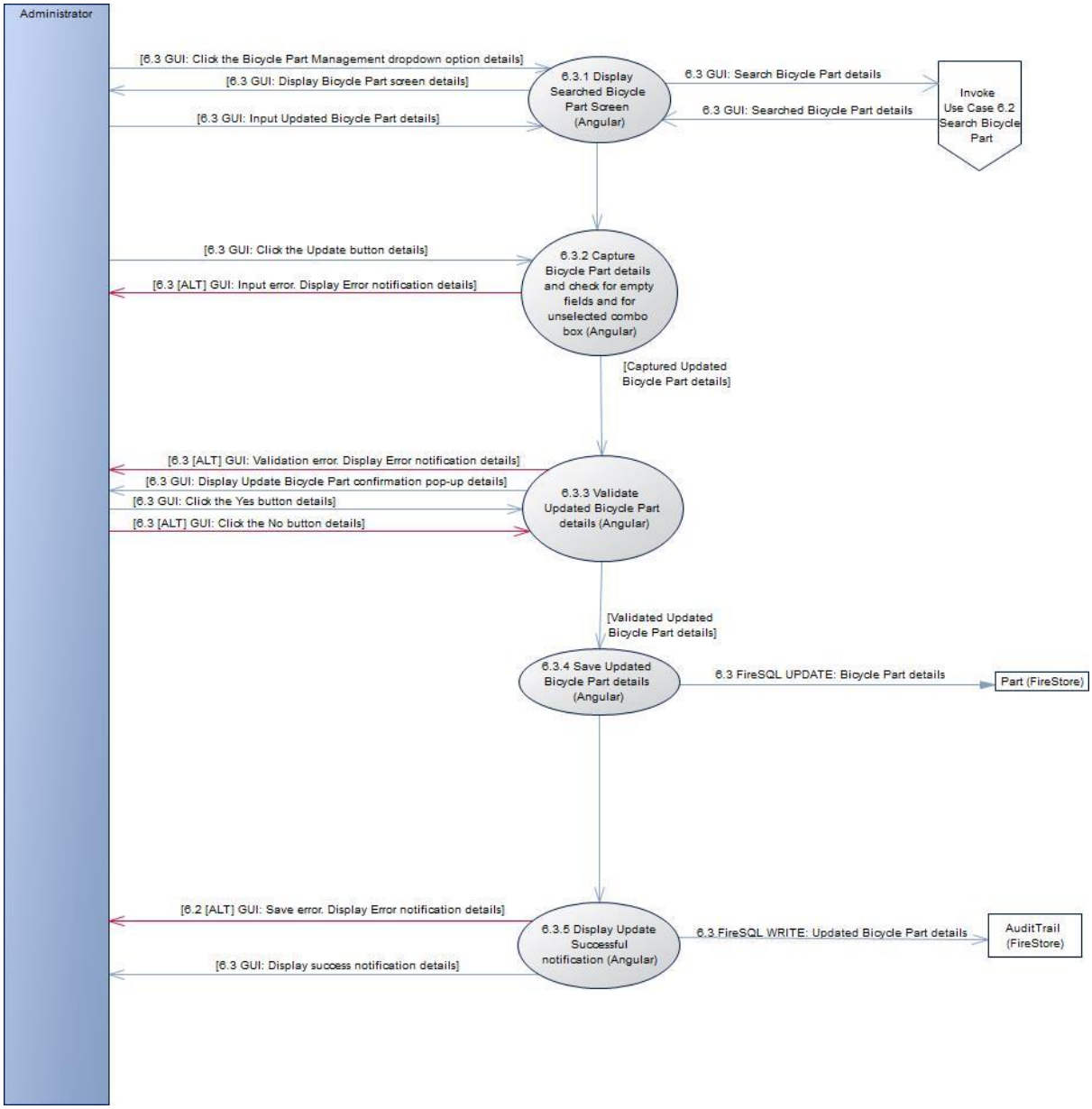


Figure 220 Primitive Diagram Sub-System 6: 6.3 Update Bicycle Part

6.4 Delete Bicycle Part

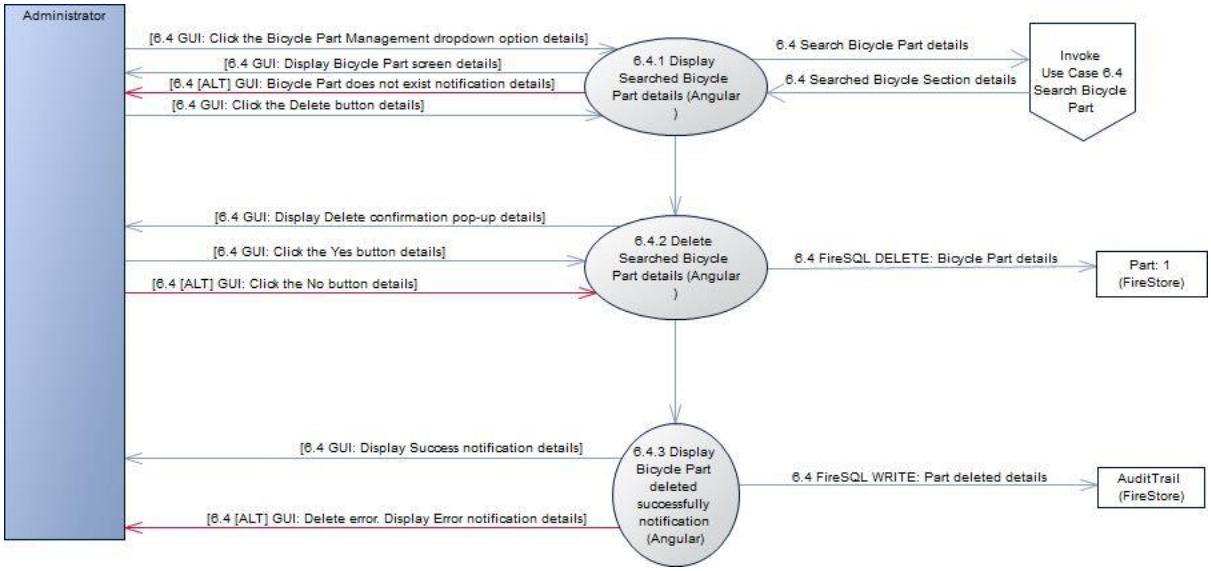


Figure 221 Primitive Diagram Sub-System 6: 6.4 Delete Bicycle Part

6.5 Add Bicycle Section

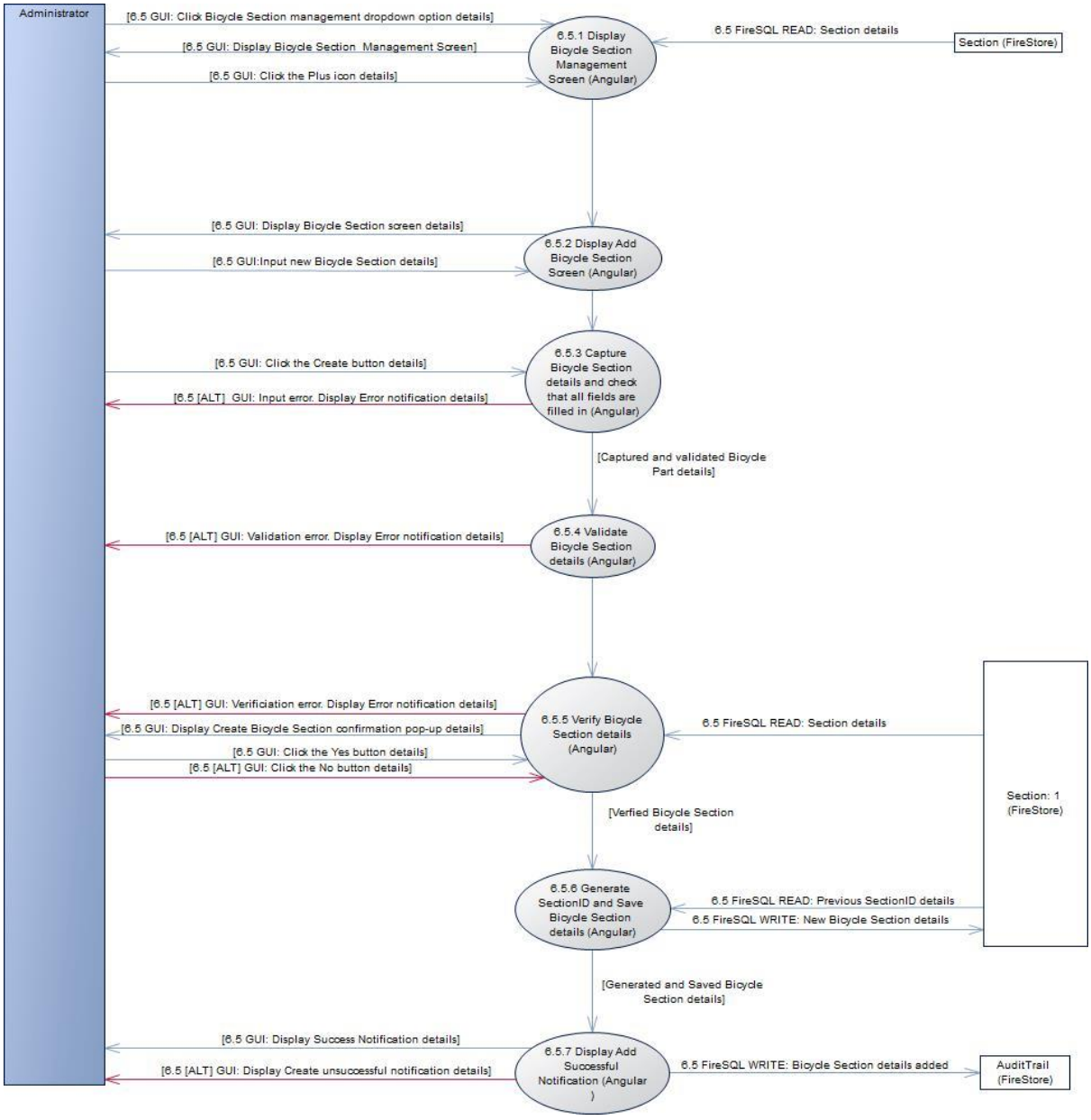


Figure 222 Primitive Diagram Sub-System 6: 6.5 Add Bicycle Section

6.6 Search Bicycle Section

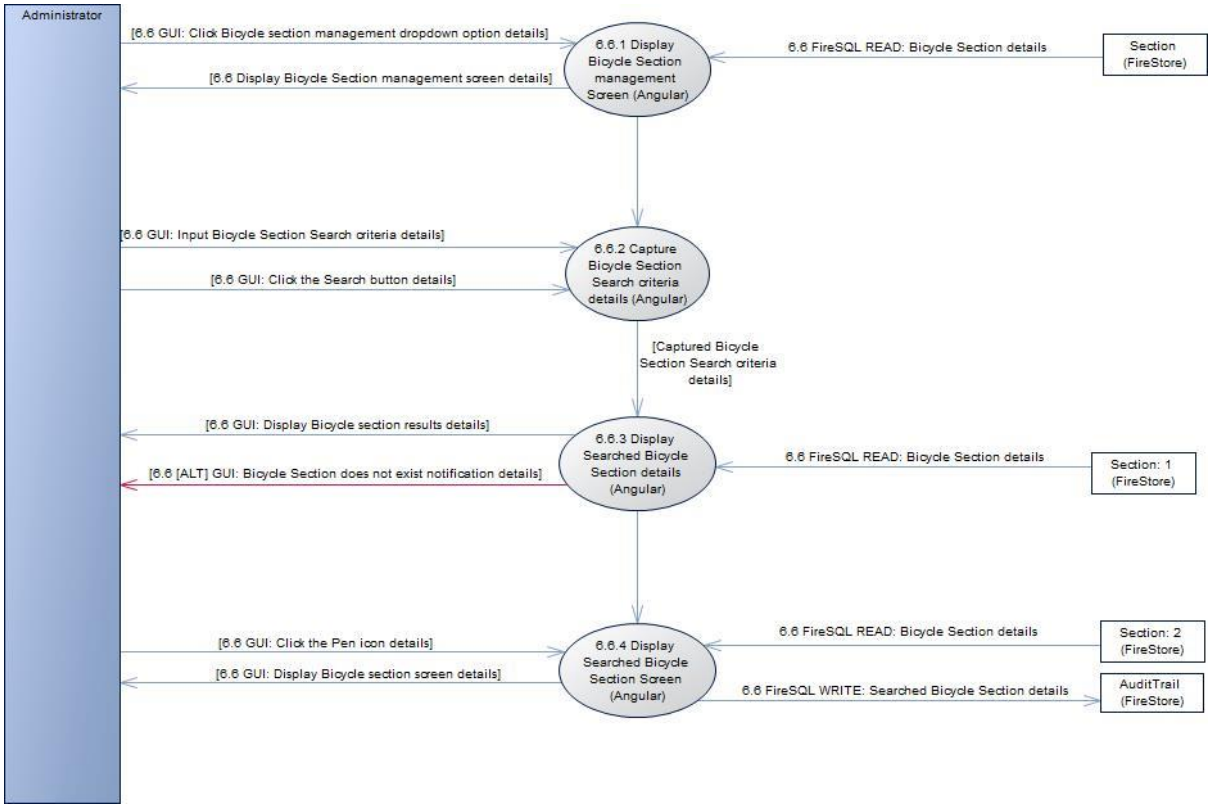


Figure 223 Primitive Diagram Sub-System 6: 6.6 Search Bicycle Section

6.7 Update Bicycle Section

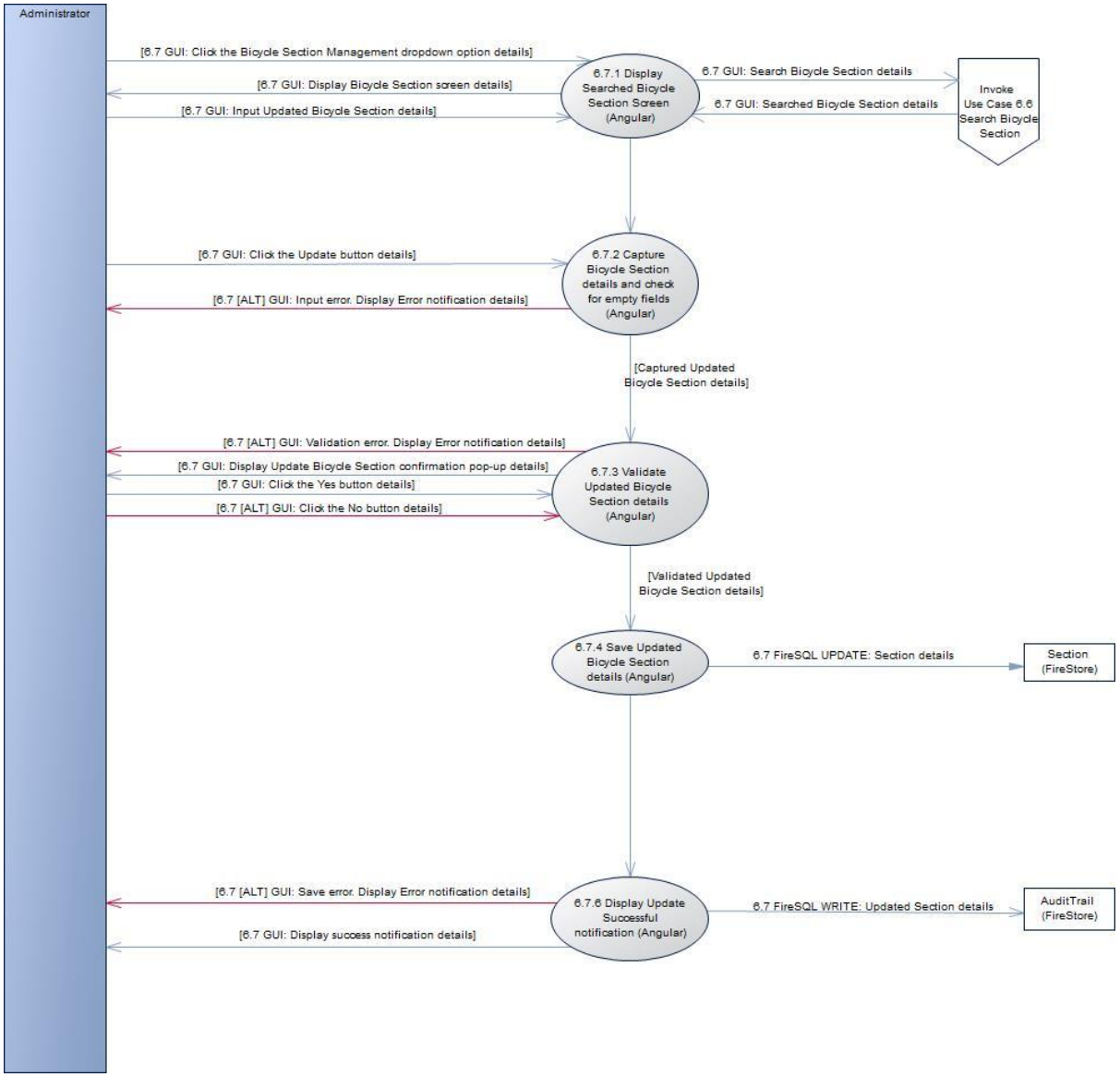


Figure 224 Primitive Diagram Sub-System 6: 6.6 Update Bicycle Section

6.8 Delete Bicycle Section

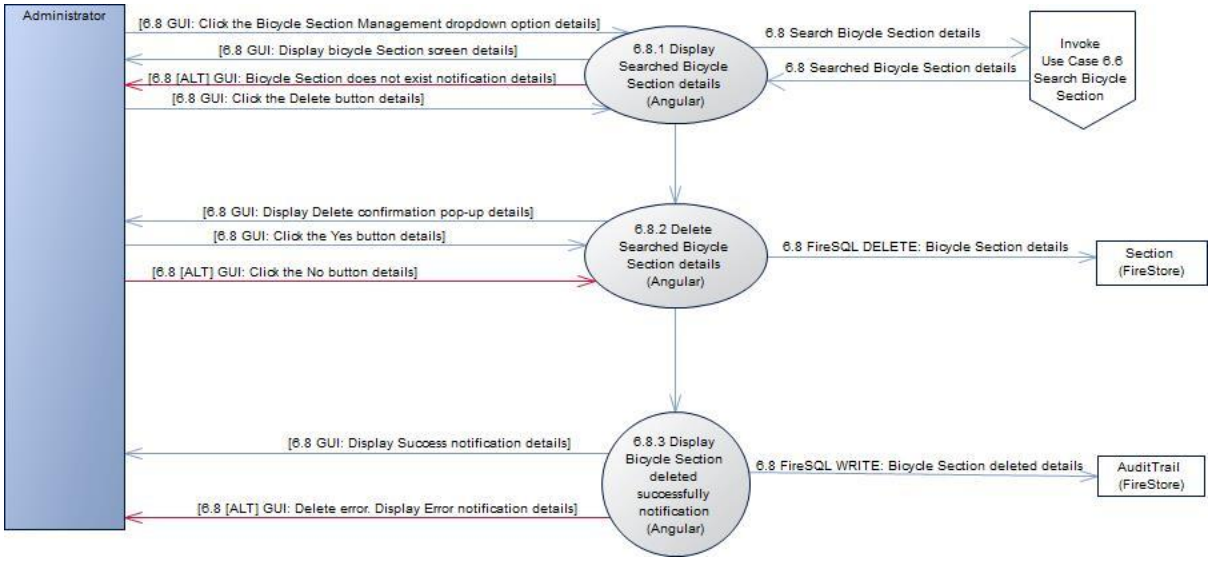


Figure 225 Primitive Diagram Sub-System 6: 6.8 Delete Bicycle Section

Sub-System 7: Tracking

7.1 Add Log Entry

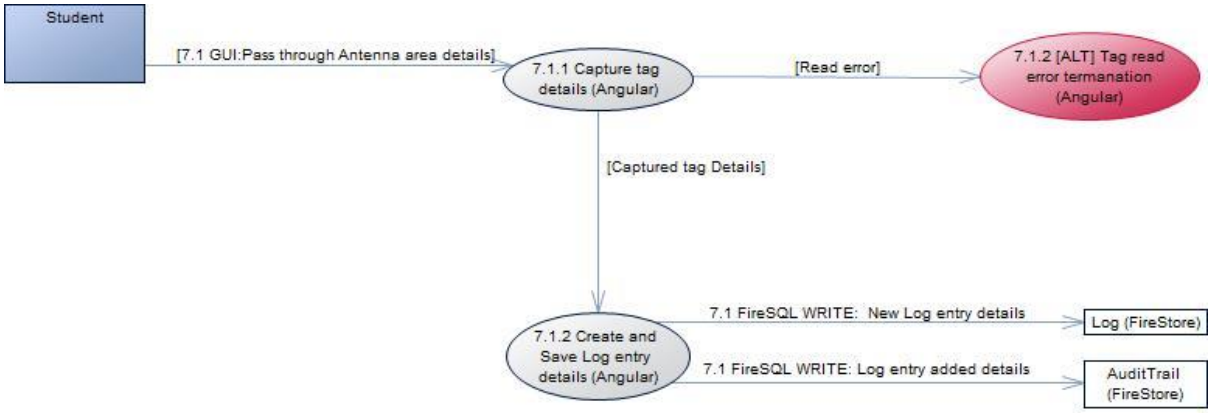


Figure 226 Primitive Diagram Sub-System 7: 7.1 Add Log Entry

7.2 Search Log Entry

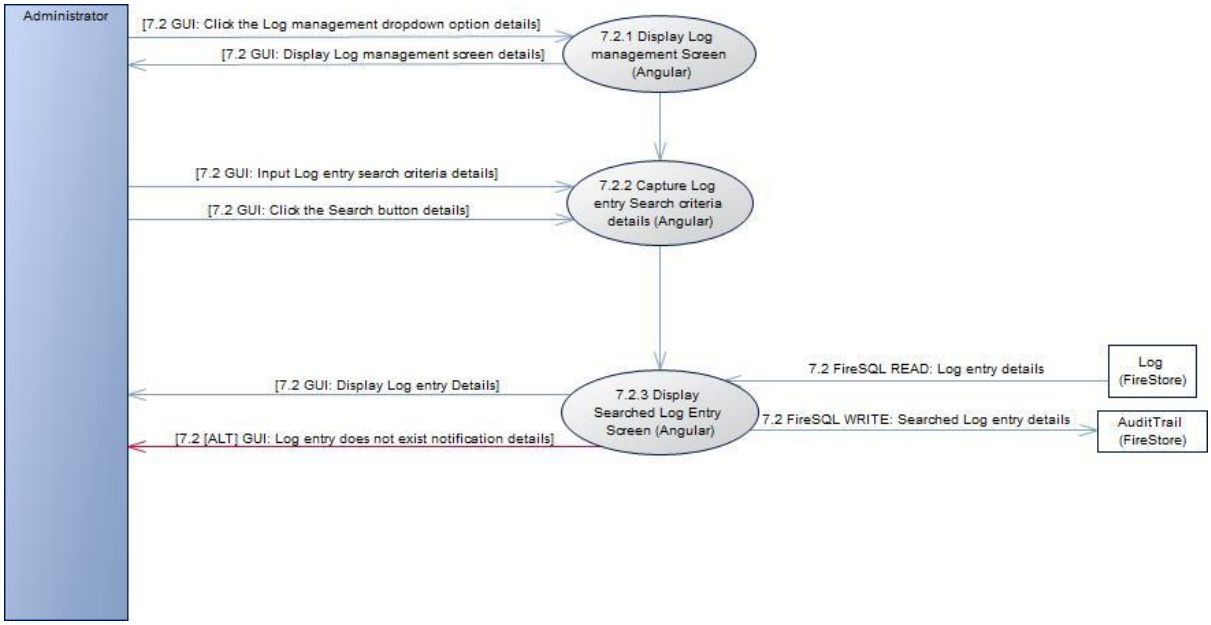


Figure 227 Primitive Diagram Sub-System 7: 7.2 Search Log Entry

7.3 Add Route

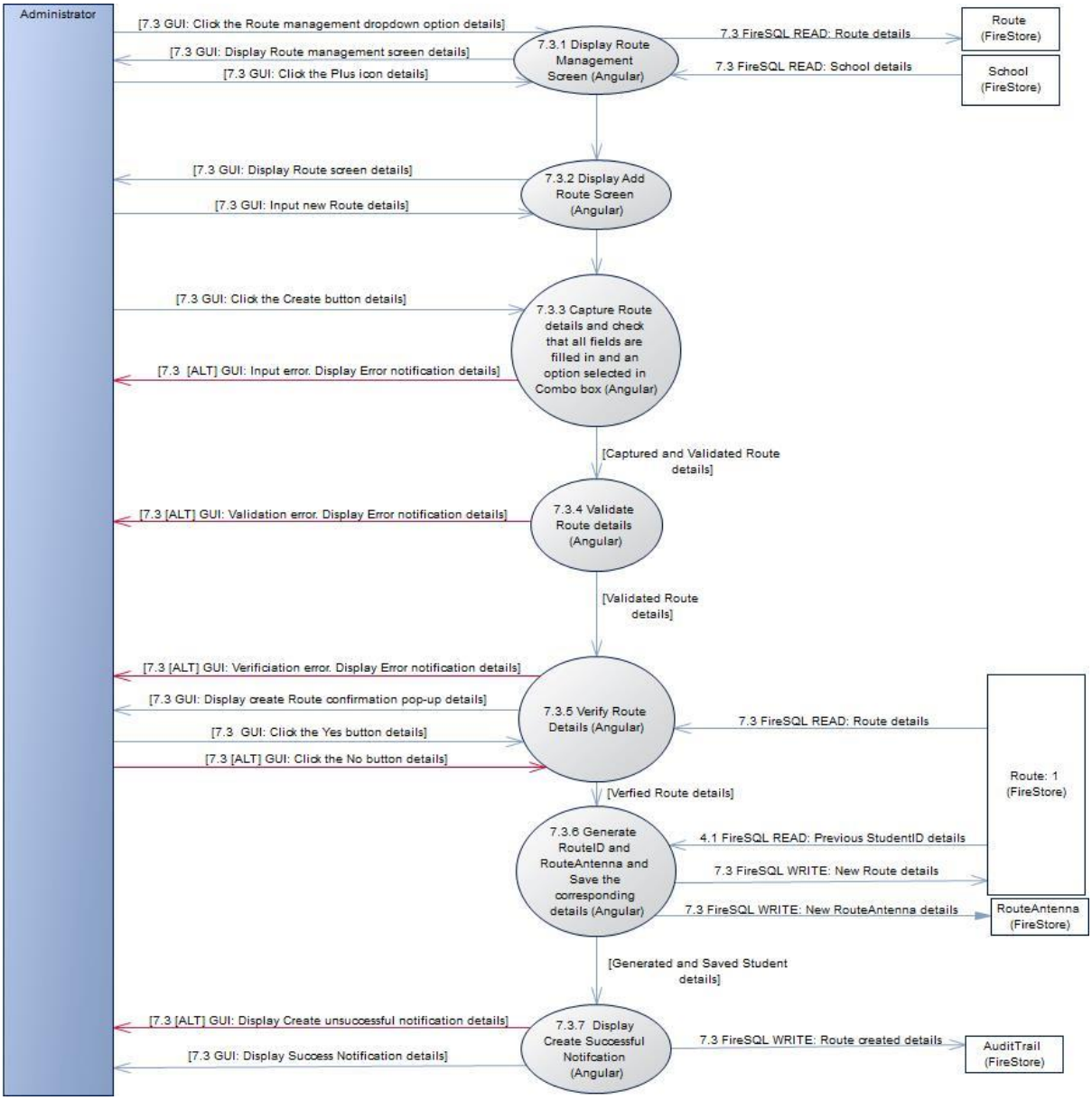


Figure 228 Primitive Diagram Sub-System 7: 7.3 Add Route

7.4 Search Route

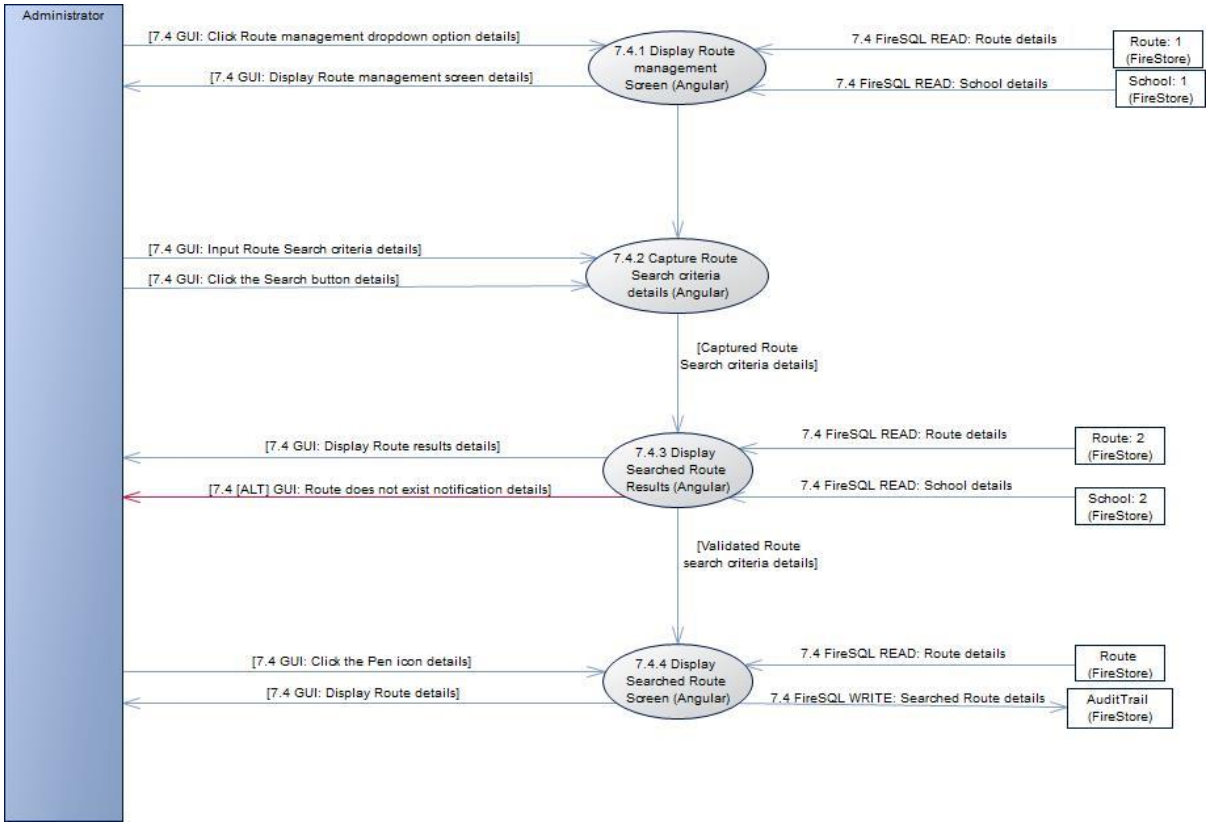


Figure 229 Primitive Diagram Sub-System 7: 7.4 Search Route

7.5 Update Route

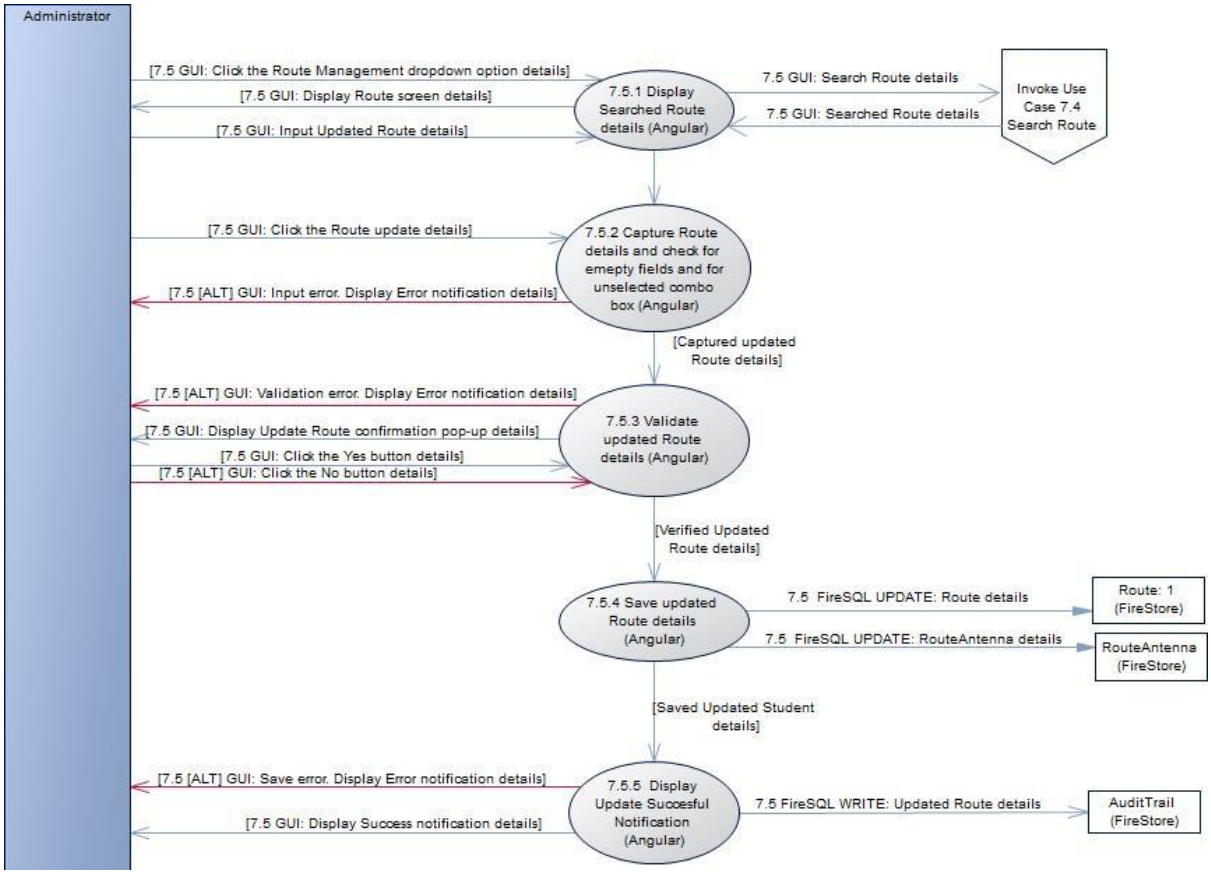


Figure 230 Primitive Diagram Sub-System 7: 7.5 Update Route

7.6 Delete Route

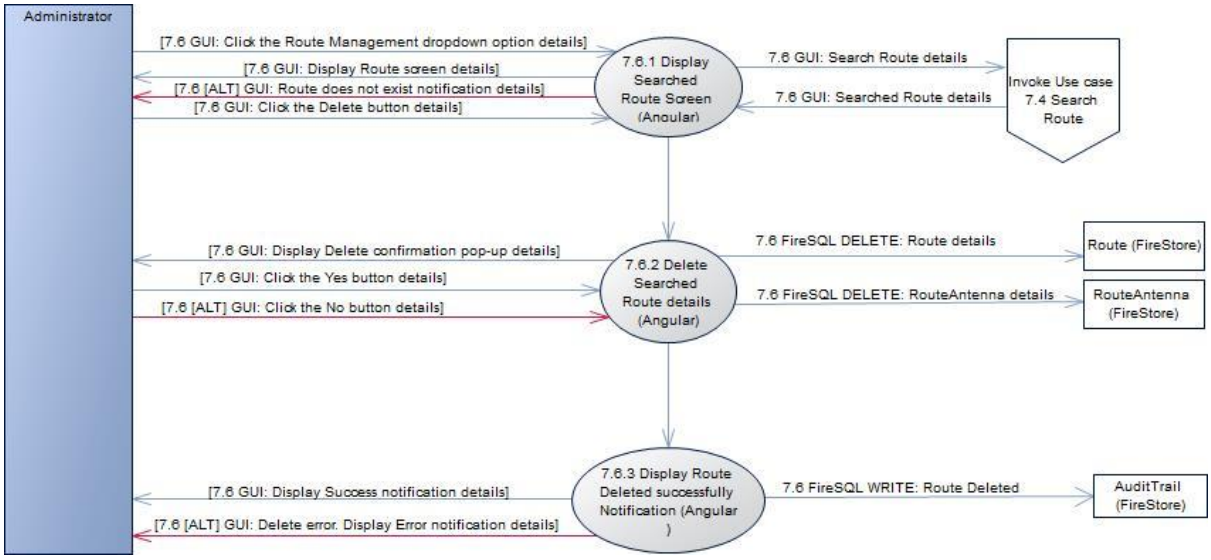


Figure 231 Primitive Diagram Sub-System 7: 7.6 Delete Route

Sub-System 8: Job Tasks

8.1 Add Fault Task

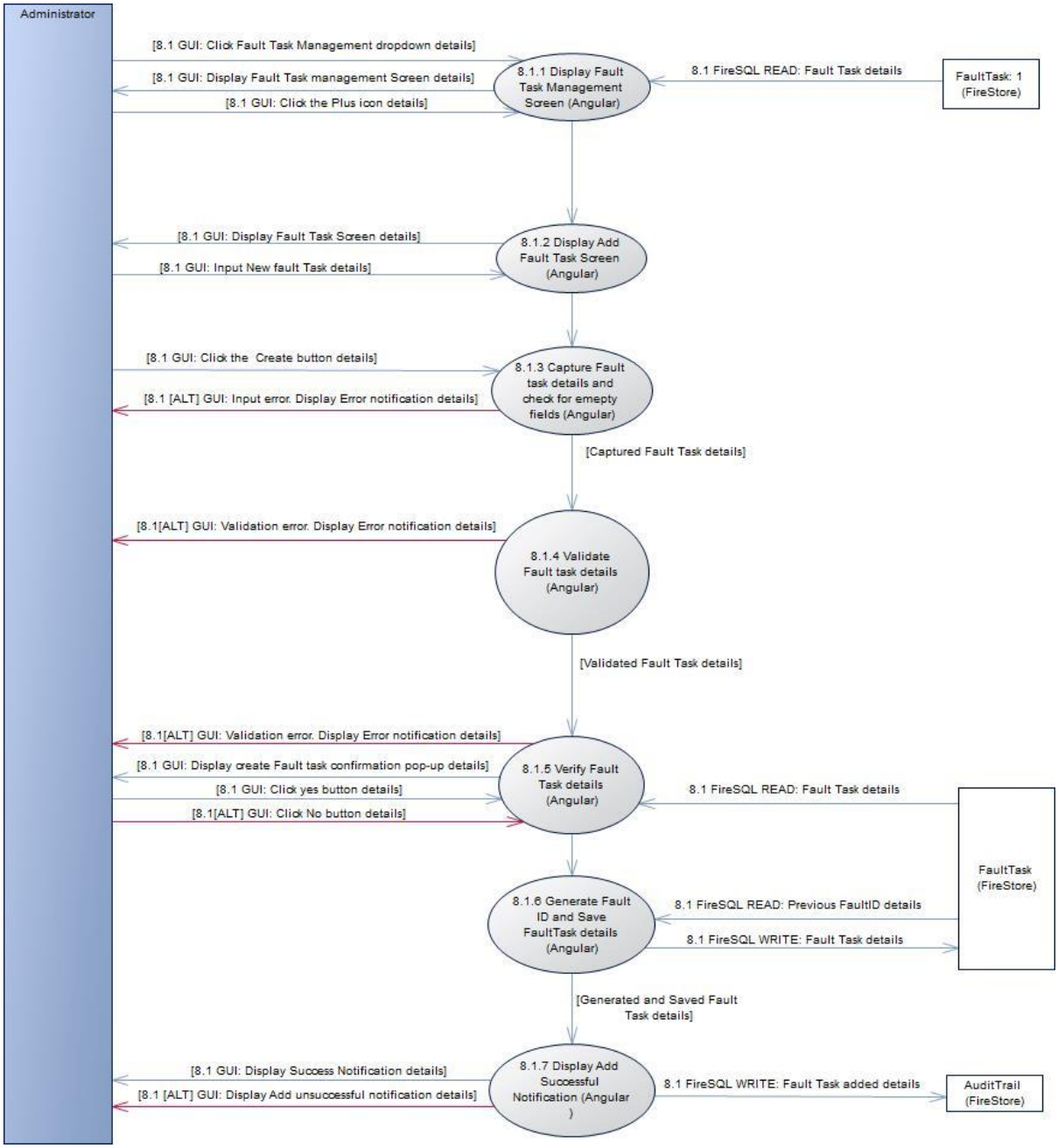


Figure 232 Primitive Diagram Sub-System 8: 8.1 Add Fault Task

8.2 Search Fault Task

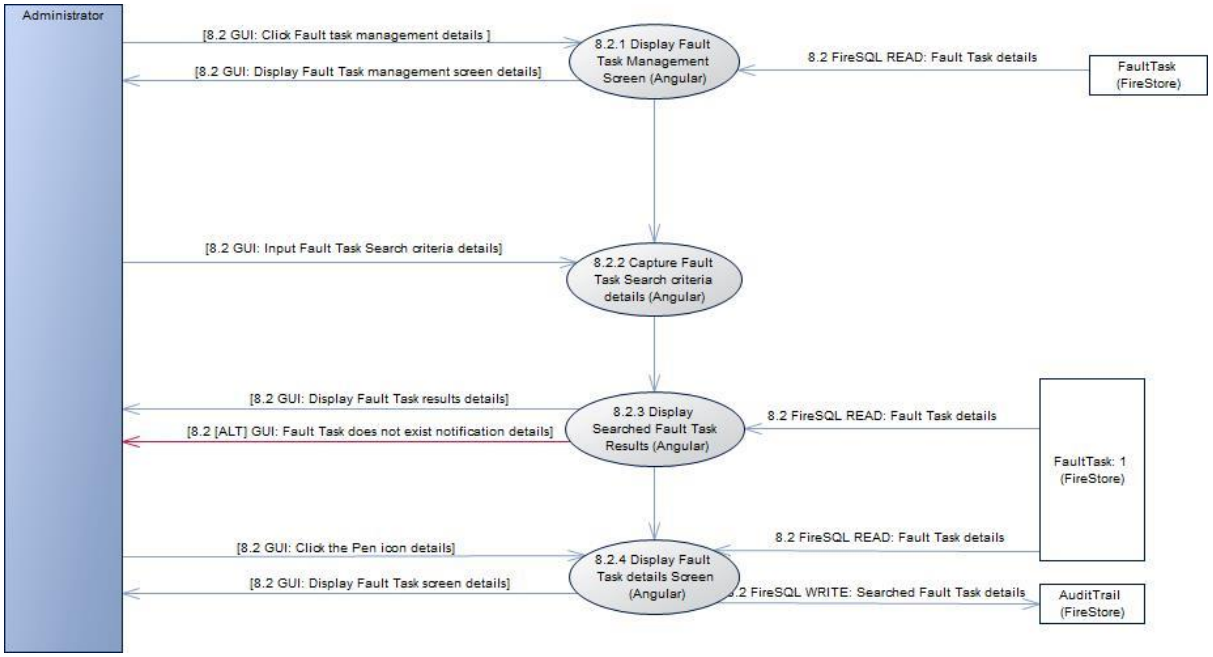


Figure 233 Primitive Diagram Sub-System 8: 8.2 Search Fault Task

8.3 Update Fault Task

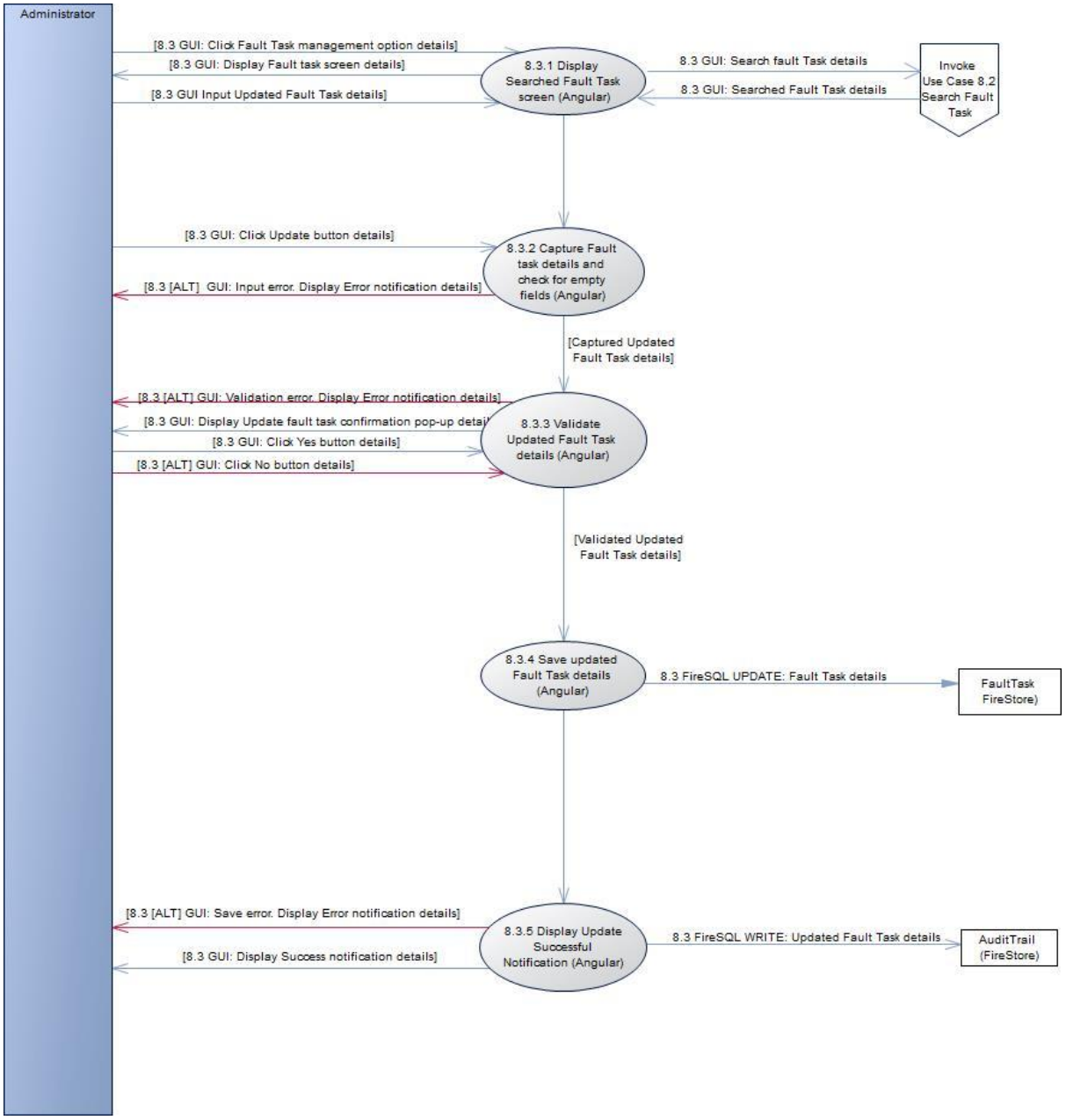


Figure 234 Primitive Diagram Sub-System 8: 8.3 Update Fault Task

8.4 Delete Fault Task

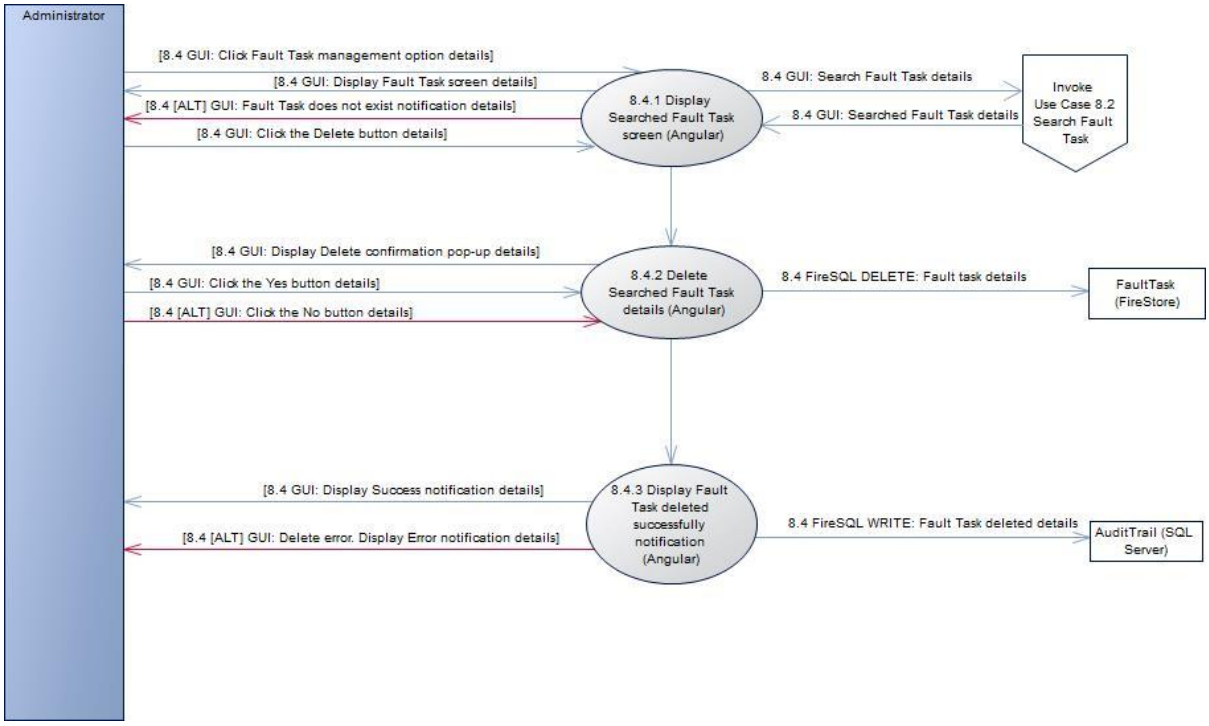


Figure 235 Primitive Diagram Sub-System 8: 8.4 Delete Fault Task

8.5 Add Maintenance Task

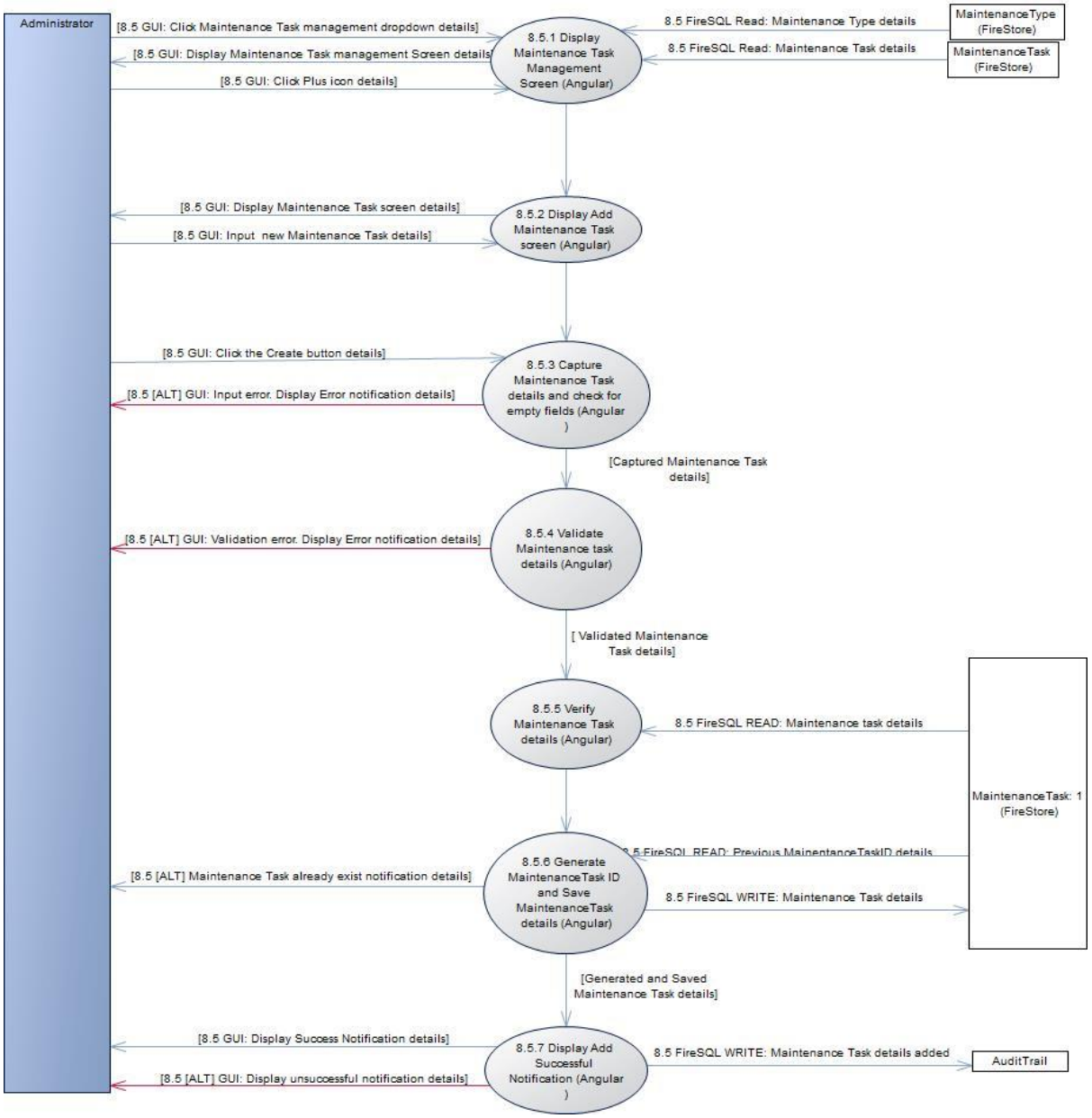


Figure 236 Primitive Diagram Sub-System 8: 8.5 Add Maintenance Task

8.6 Search Maintenance Task

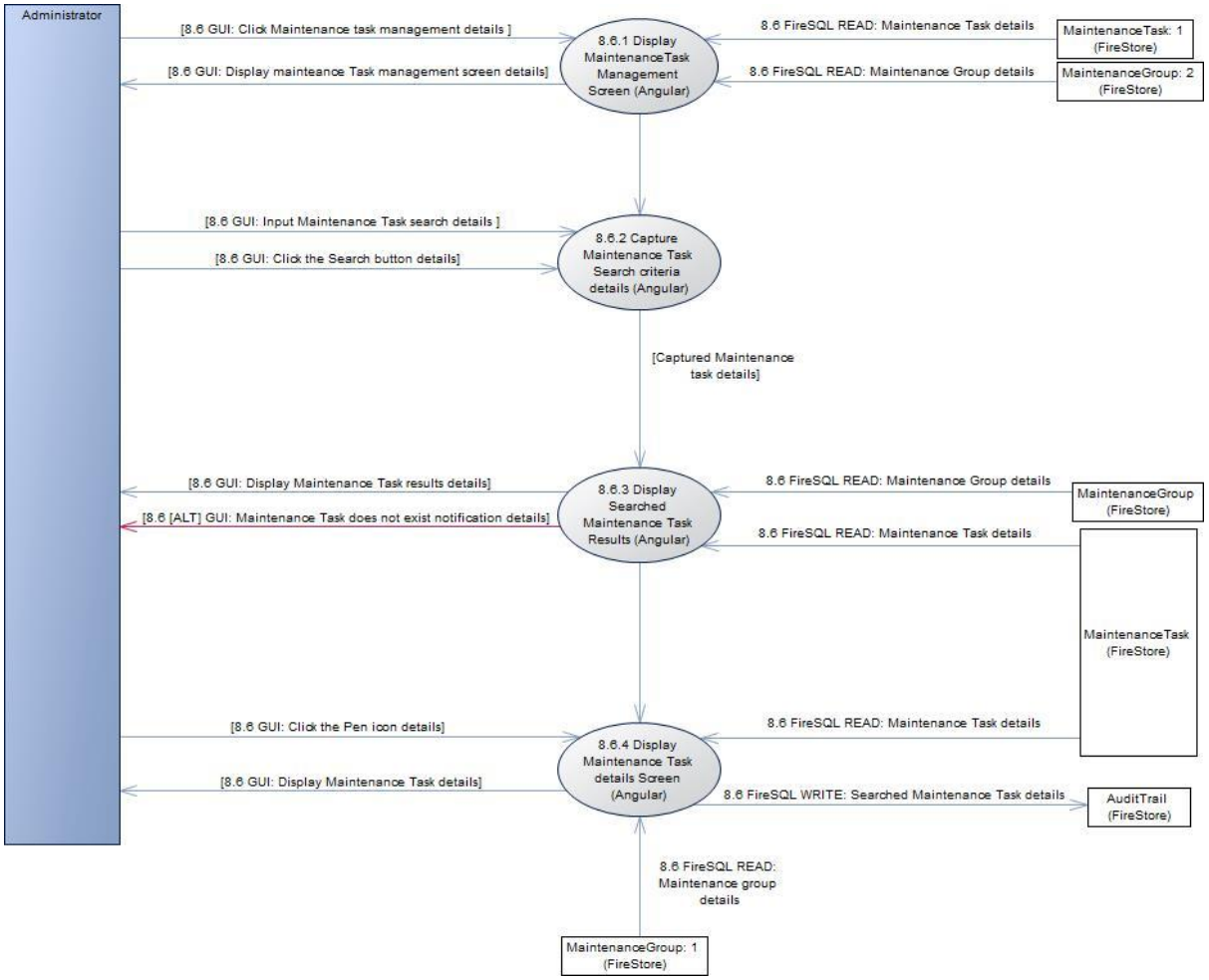


Figure 237 Primitive Diagram Sub-System 8: 8.6 Search Maintenace Task

8.7 Update Maintenance Task

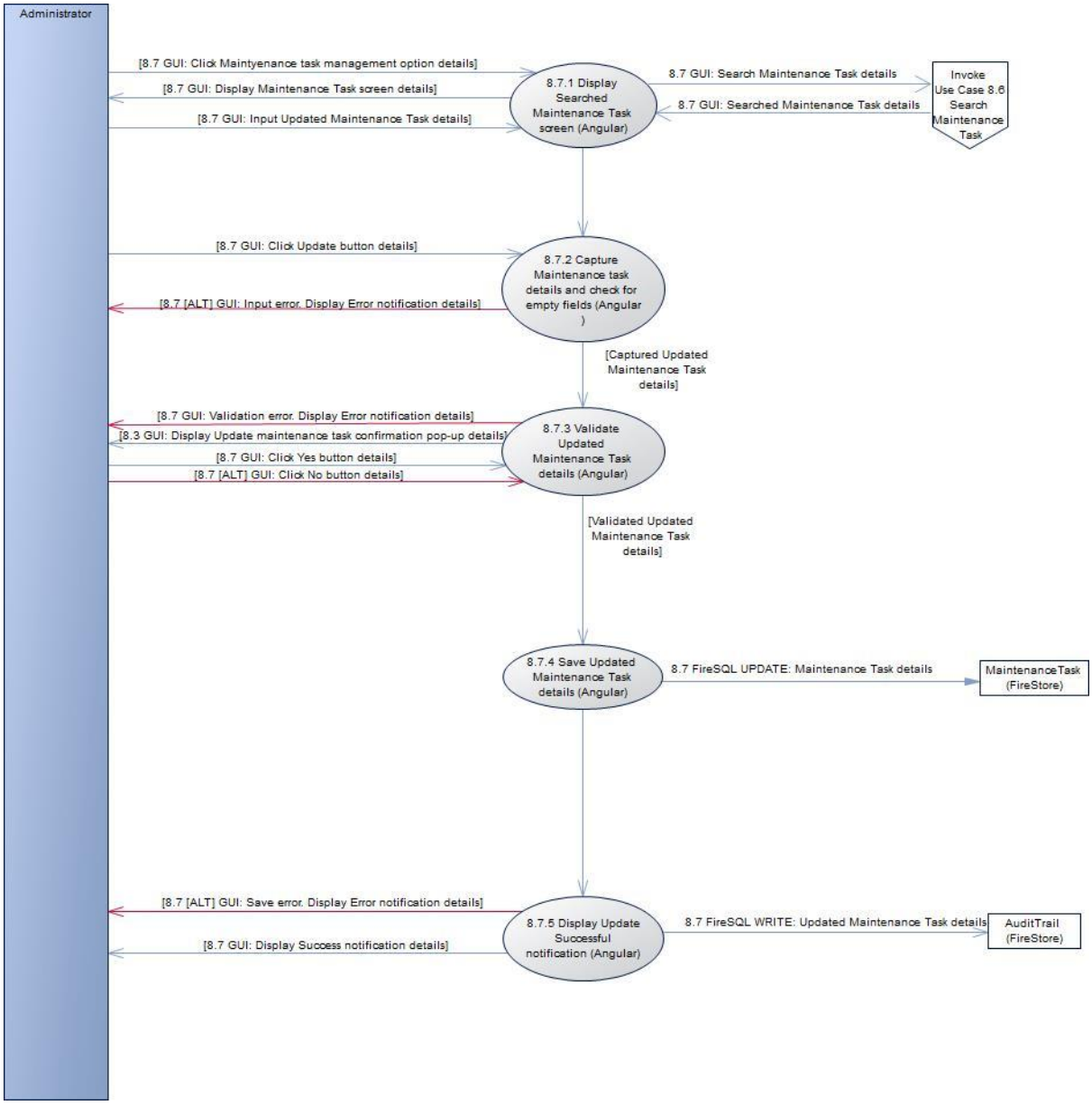


Figure 238 Primitive Diagram Sub-System 8: 8.7 Update Maintenance Task

8.8 Delete Maintenance Task

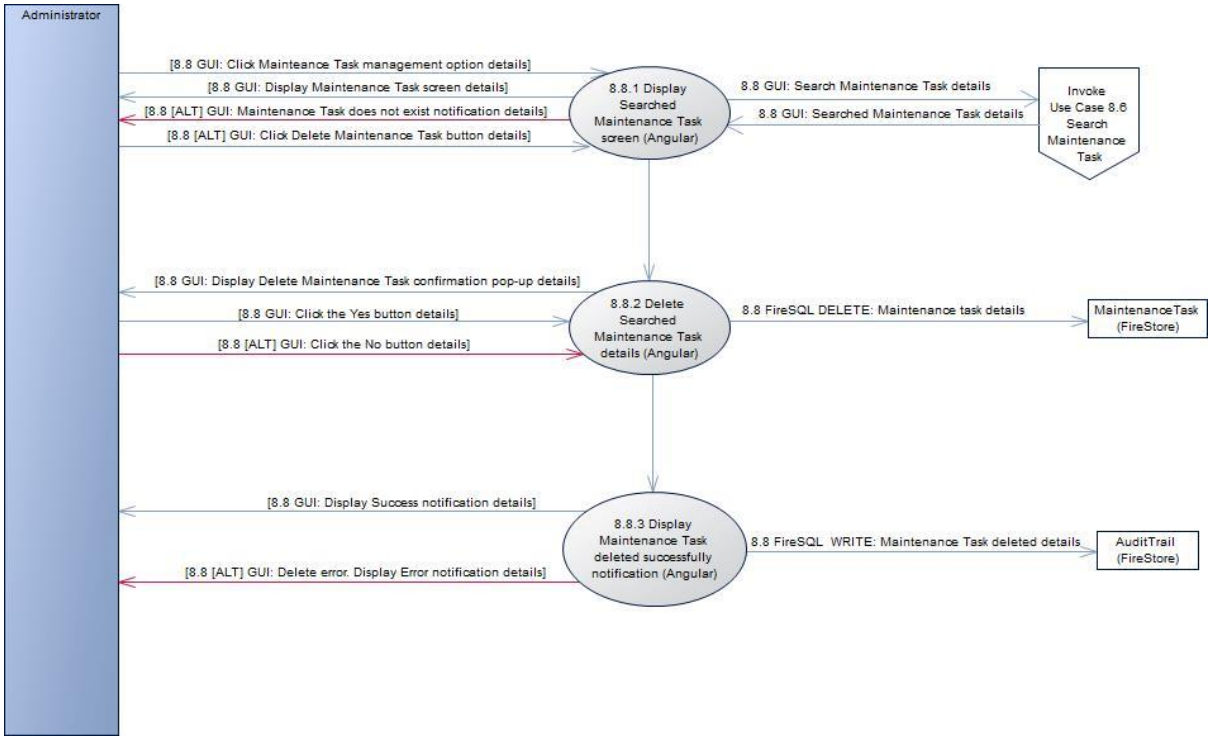


Figure 239 Primitive Diagram Sub-System 8: 8.8 Delete Maintenance Task

Sub-System 9: School

9.1 Add School

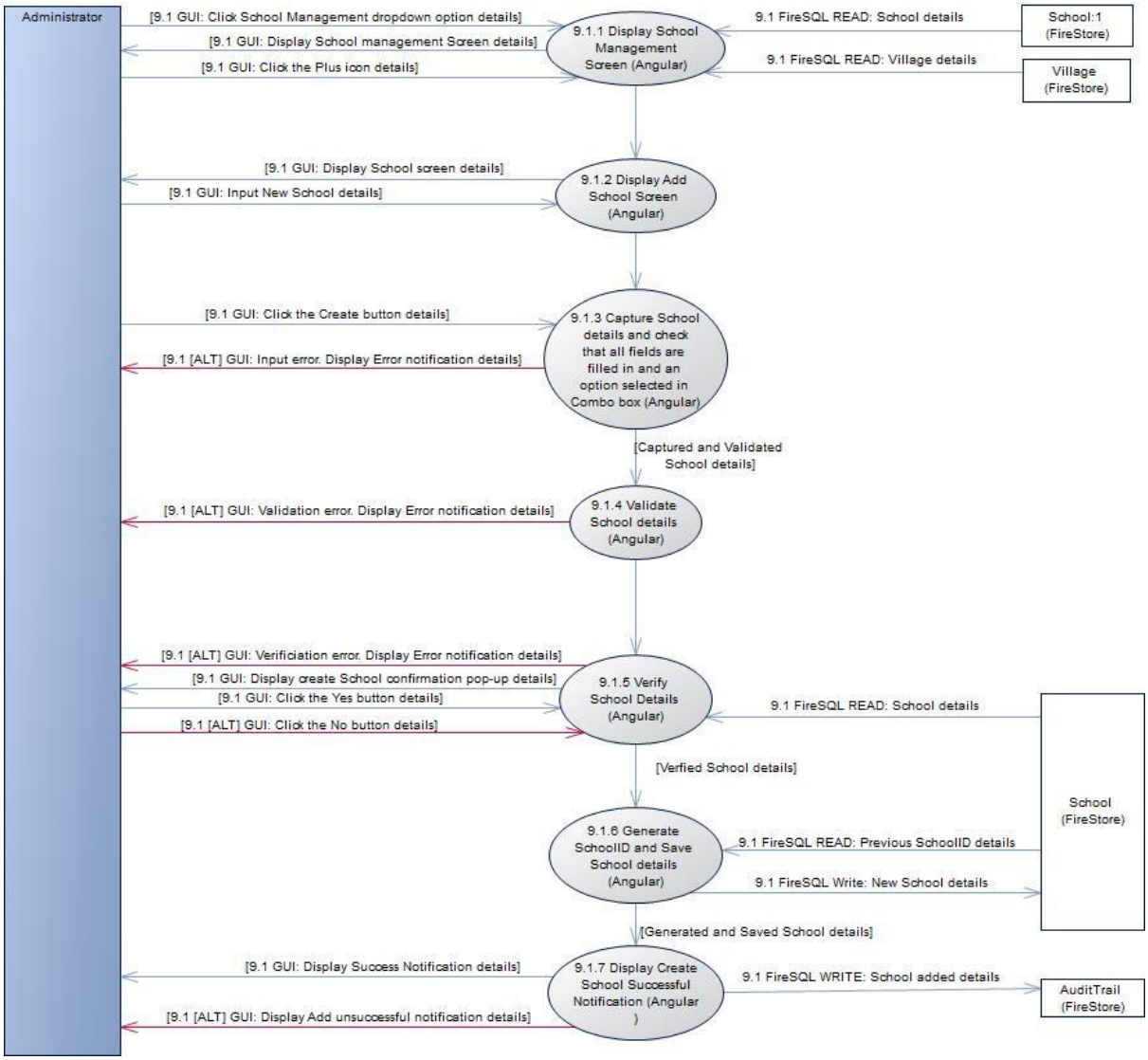


Figure 240 Primitive Diagram Sub-System 9: 9.1 Add School

9.2 Search School

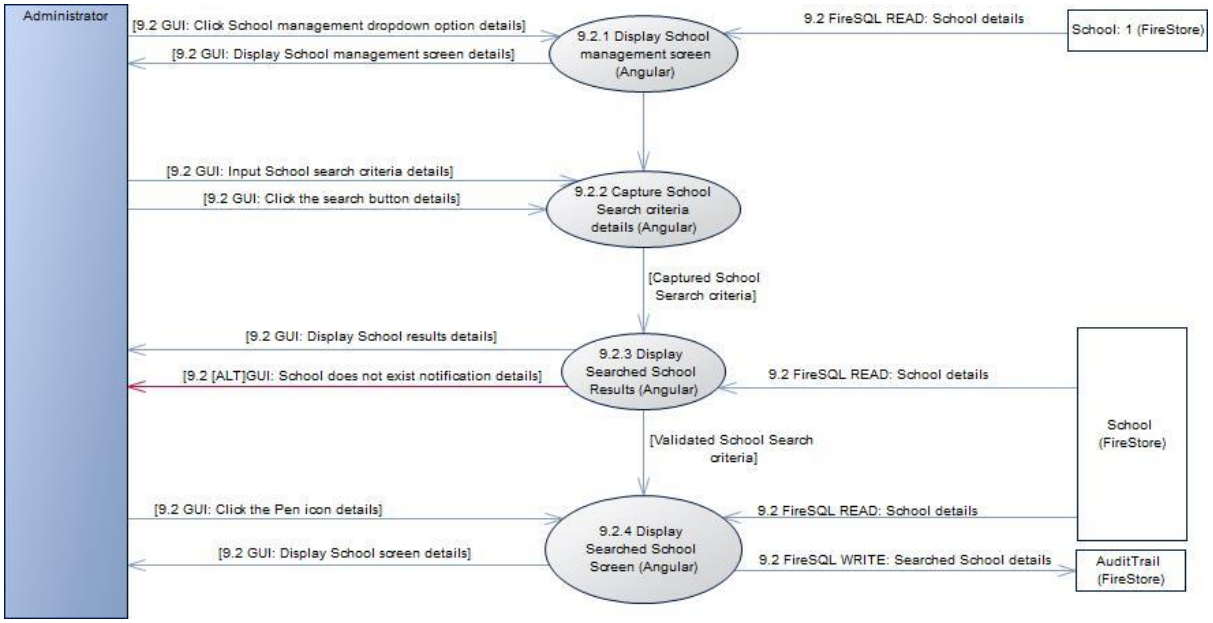


Figure 241 Primitive Diagram Sub-System 9: 9.2 Search School

9.3 Update School

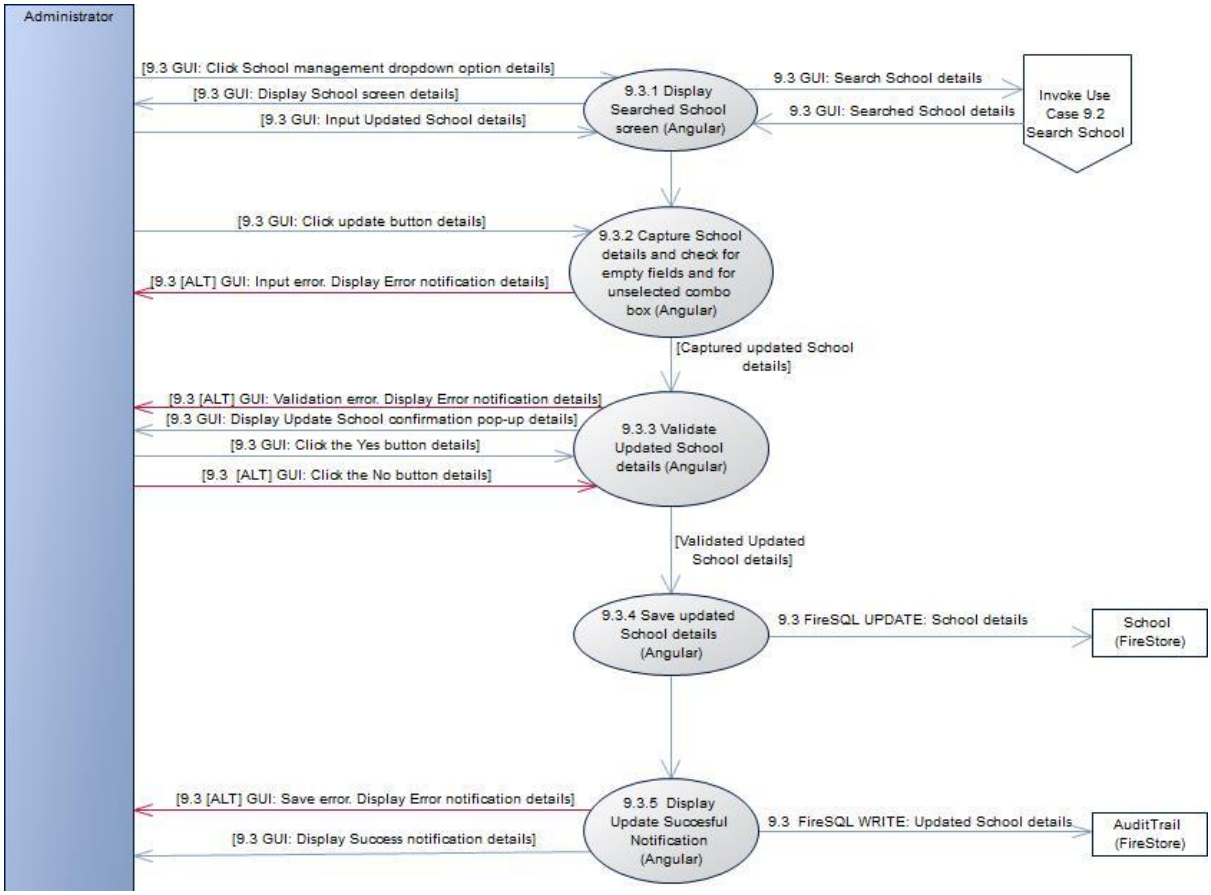


Figure 242 Primitive Diagram Sub-System 9: 9.3 Update School

9.4 Delete School

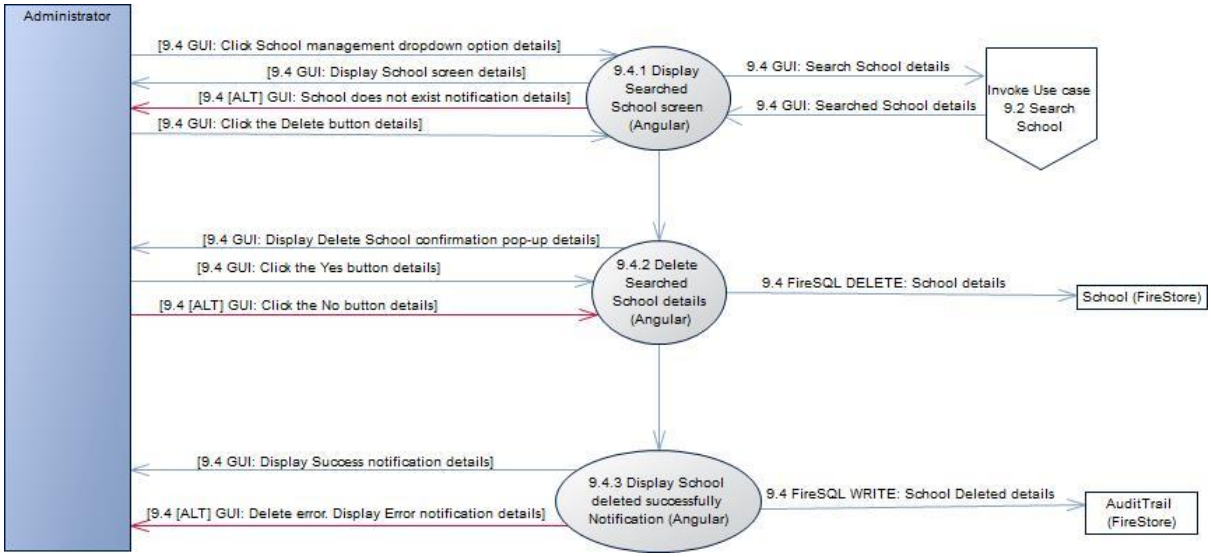


Figure 243 Primitive Diagram Sub-System 9: 9.4 Delete School

Sub-System 10: Job

10.1 Report Repair Job

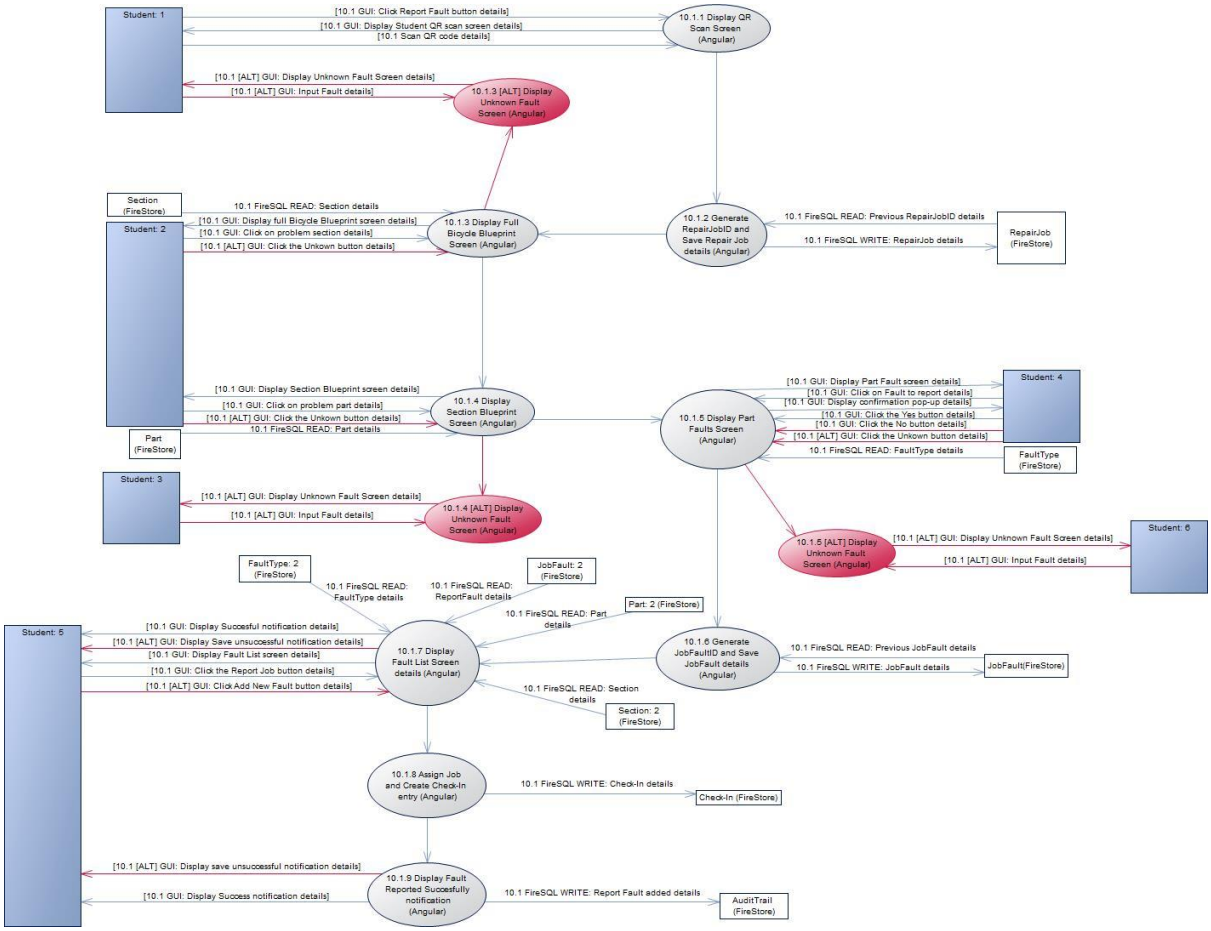


Figure 244 Primitive Diagram Sub-System 10: 10.1 Report Repair Job

10.2 Complete Job

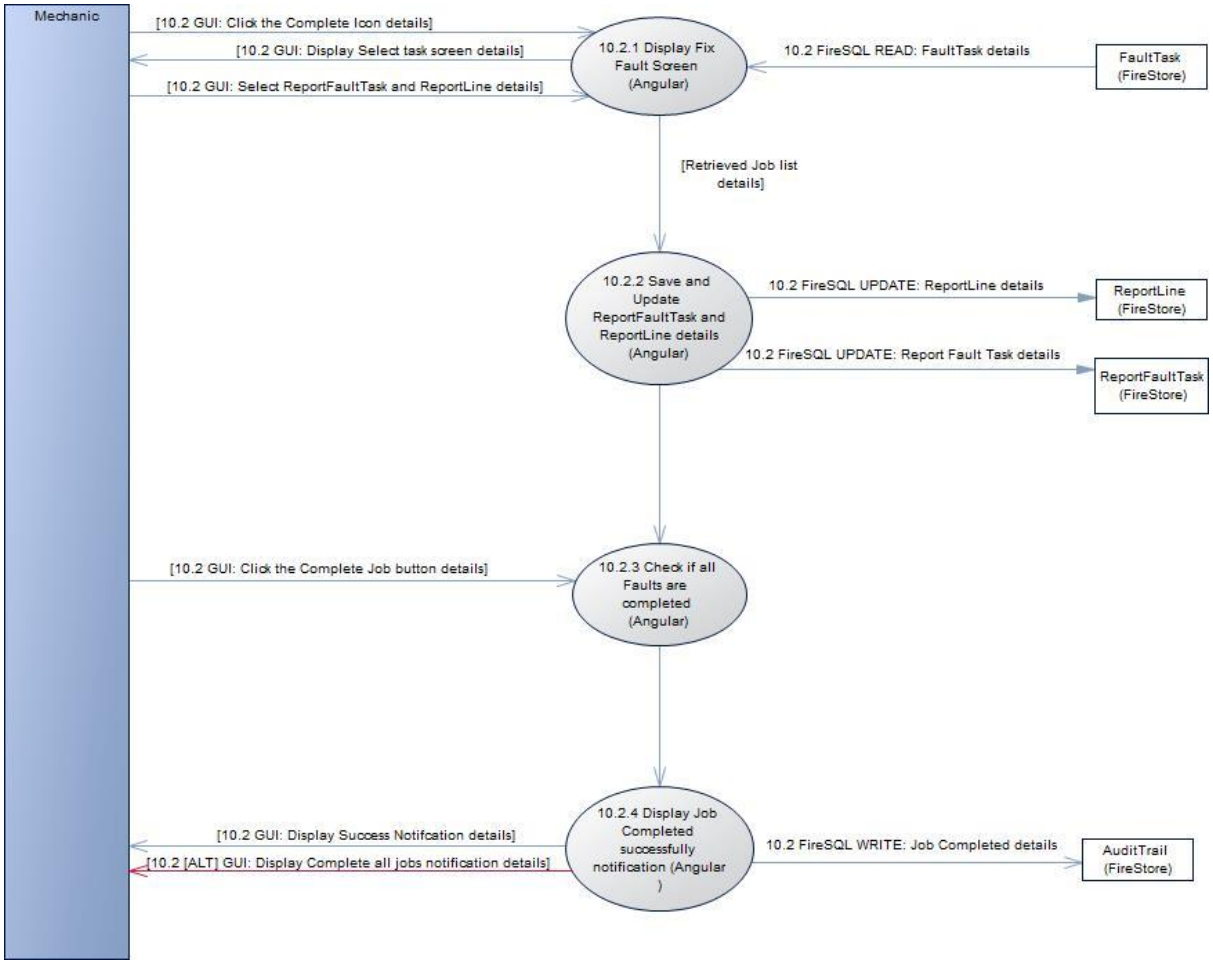


Figure 245 Primitive Diagram Sub-System 10: 10.2 Complete Job

10.3 Check-In Bicycle

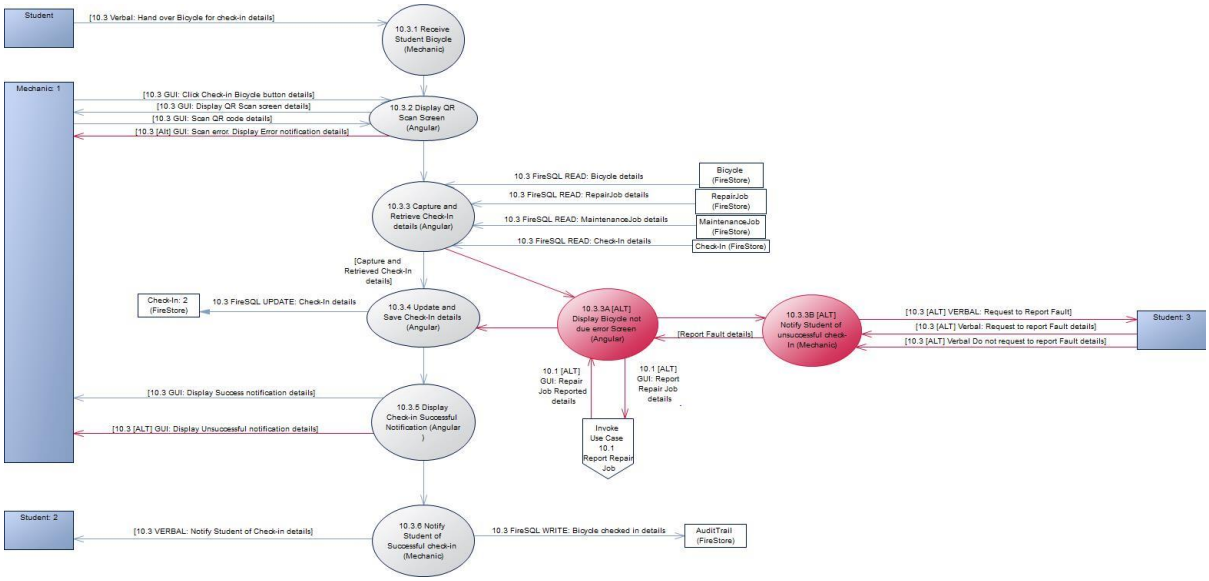


Figure 246 Primitive Diagram Sub-System 10: 10.3 Check-In

10.4 Schedule Maintenance

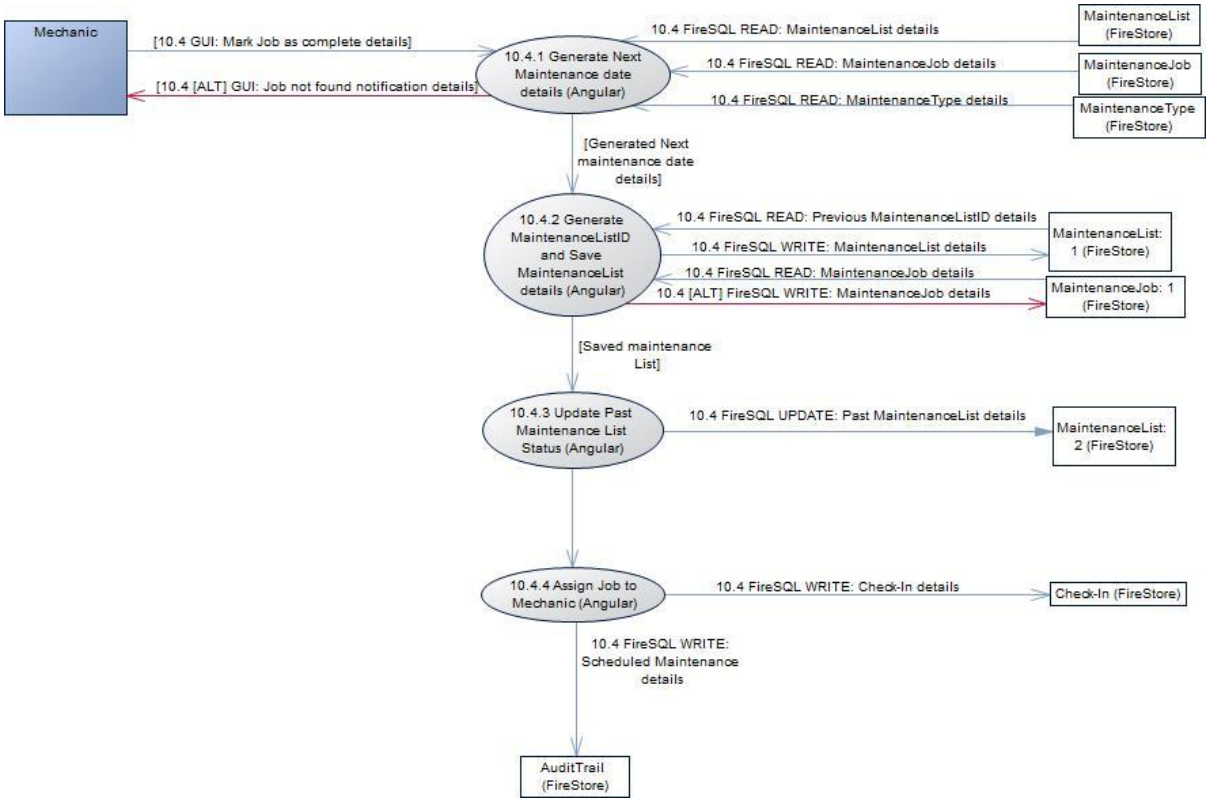


Figure 247 Primitive Diagram Sub-System 10: 10.4 Schedule Maintenance

Sub-System 11: Reporting

11.1 Generate Mechanic Schedule

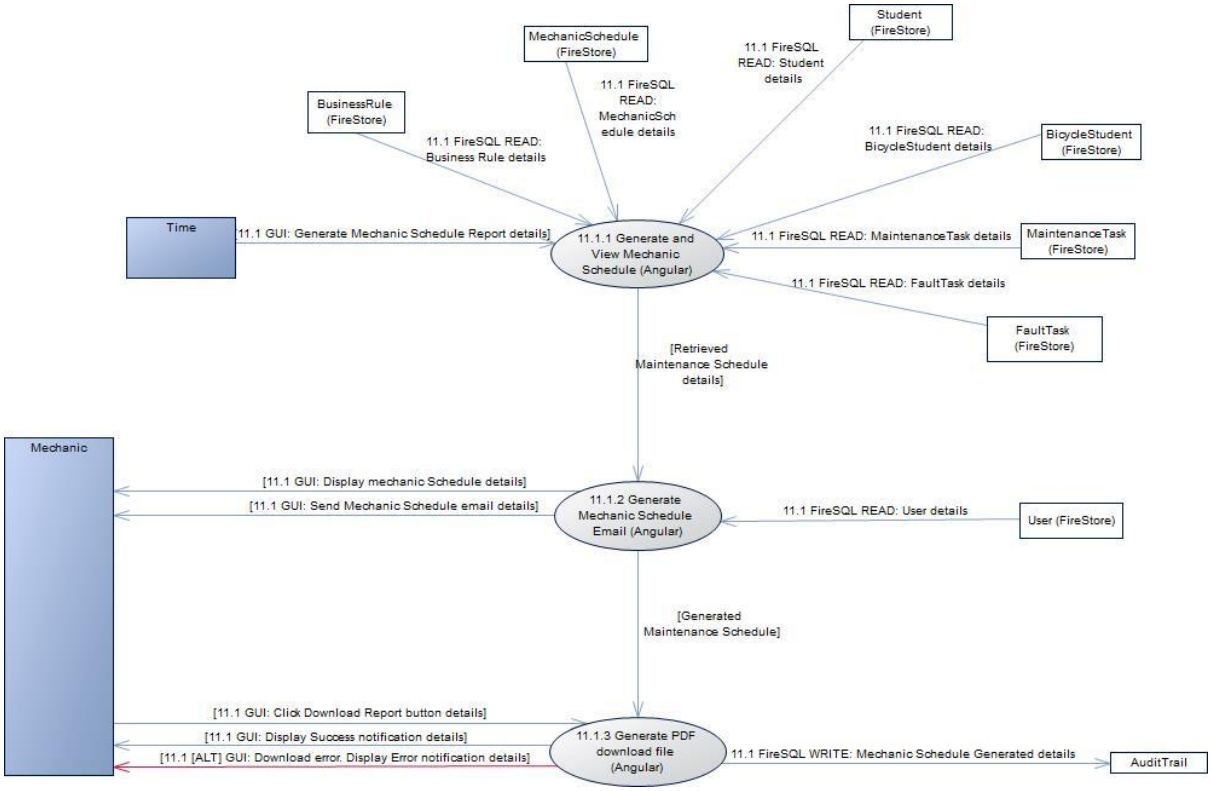


Figure 248 Primitive Diagram Sub-System 11: 11.1 Generate Mechanic Schedule

11.2 Generate Mechanic Schedule Requirements Report

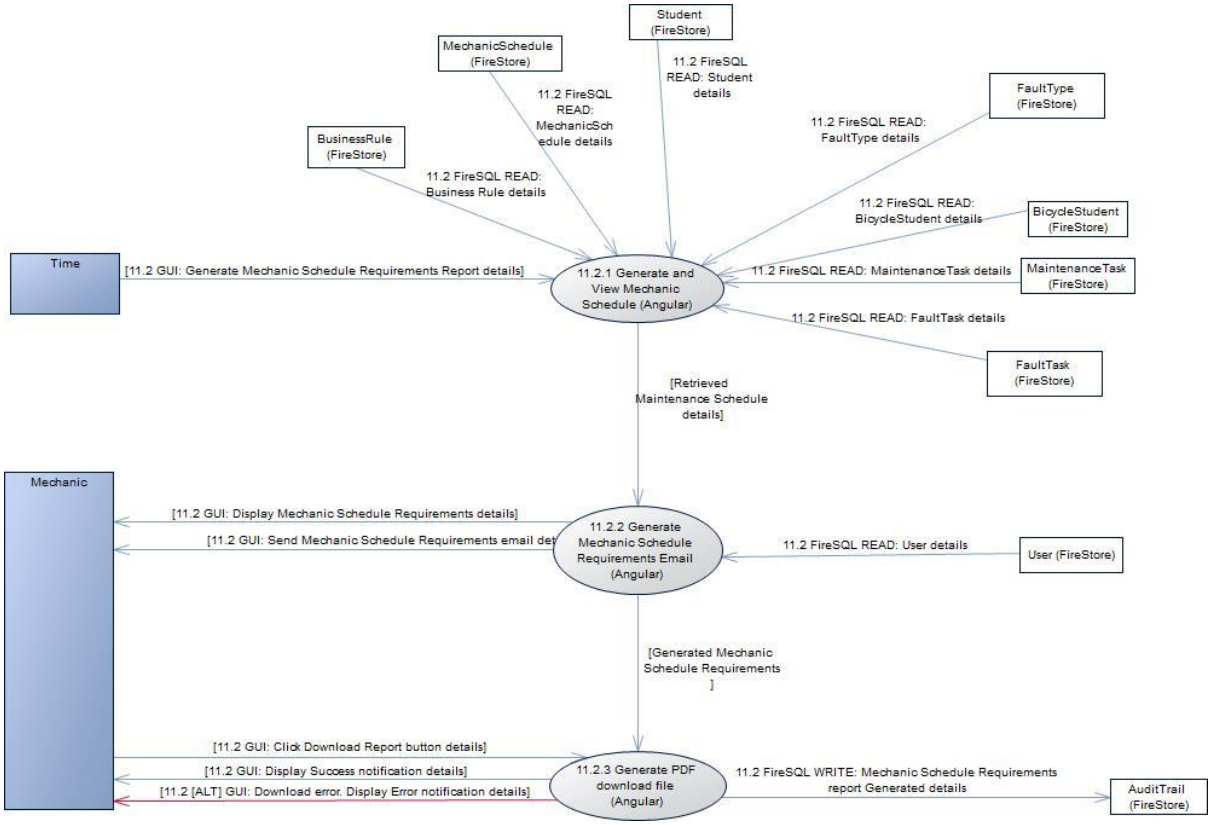


Figure 249 Primitive Diagram Sub-System 11: 11.2 Generate Mechanic Schedule Requirements Report

11.3 Generate Student Report

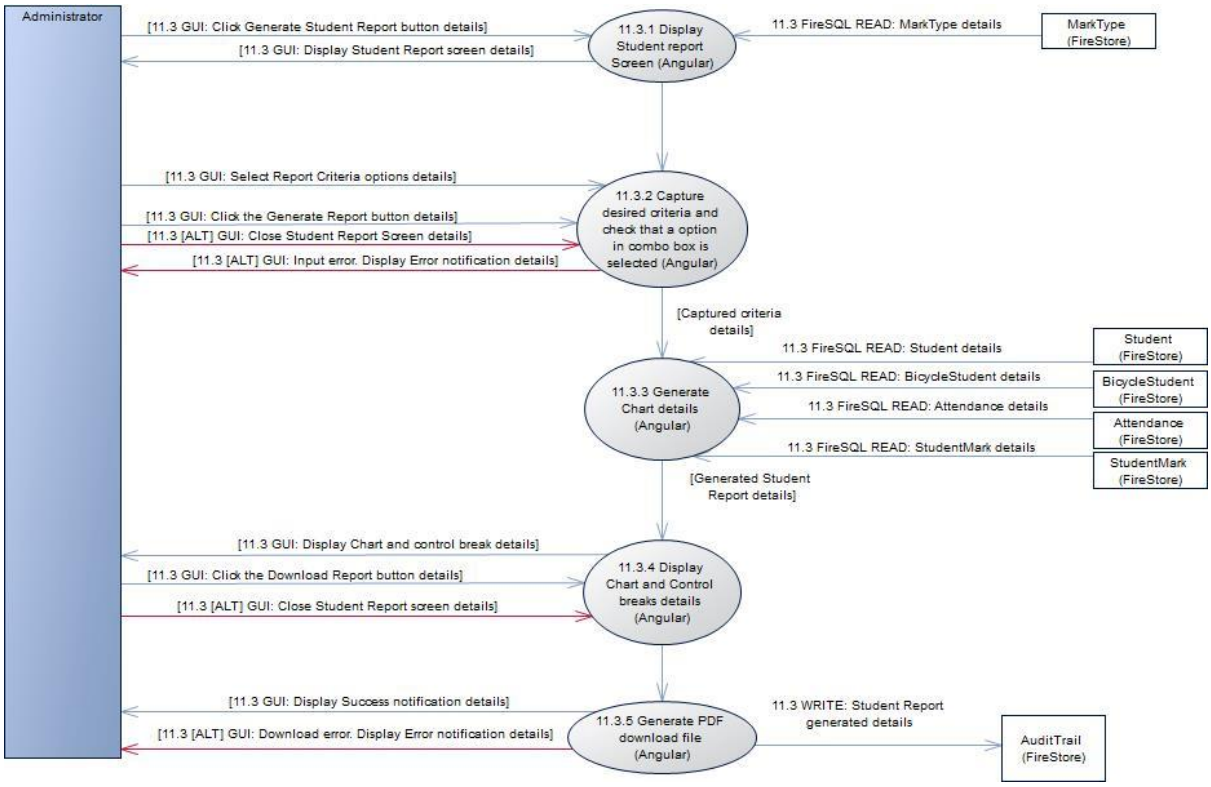


Figure 250 Primitive Diagram Sub-System 11: 11.3 generate Student Report

11.4 Generate Bicycle History Report

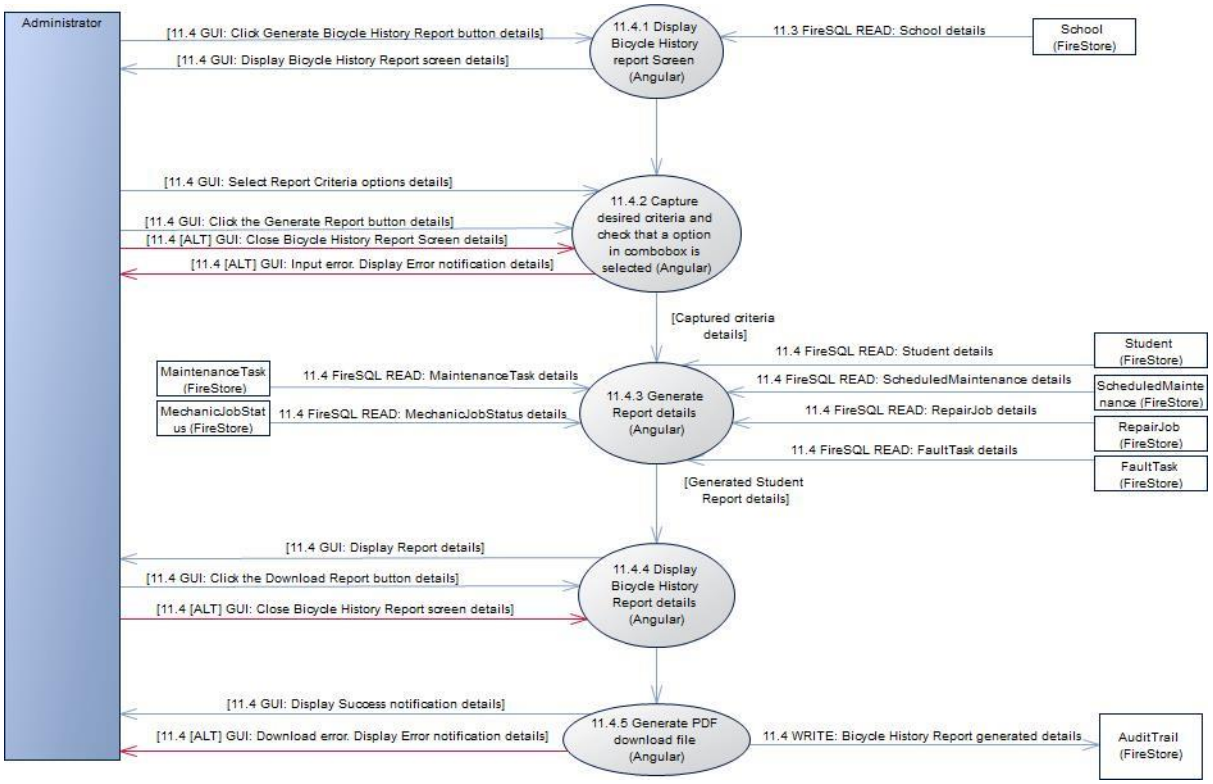


Figure 251 Primitive Diagram Sub-System 11: 11.4 Generate Bicycle History Report

11.5 Generate Log Report

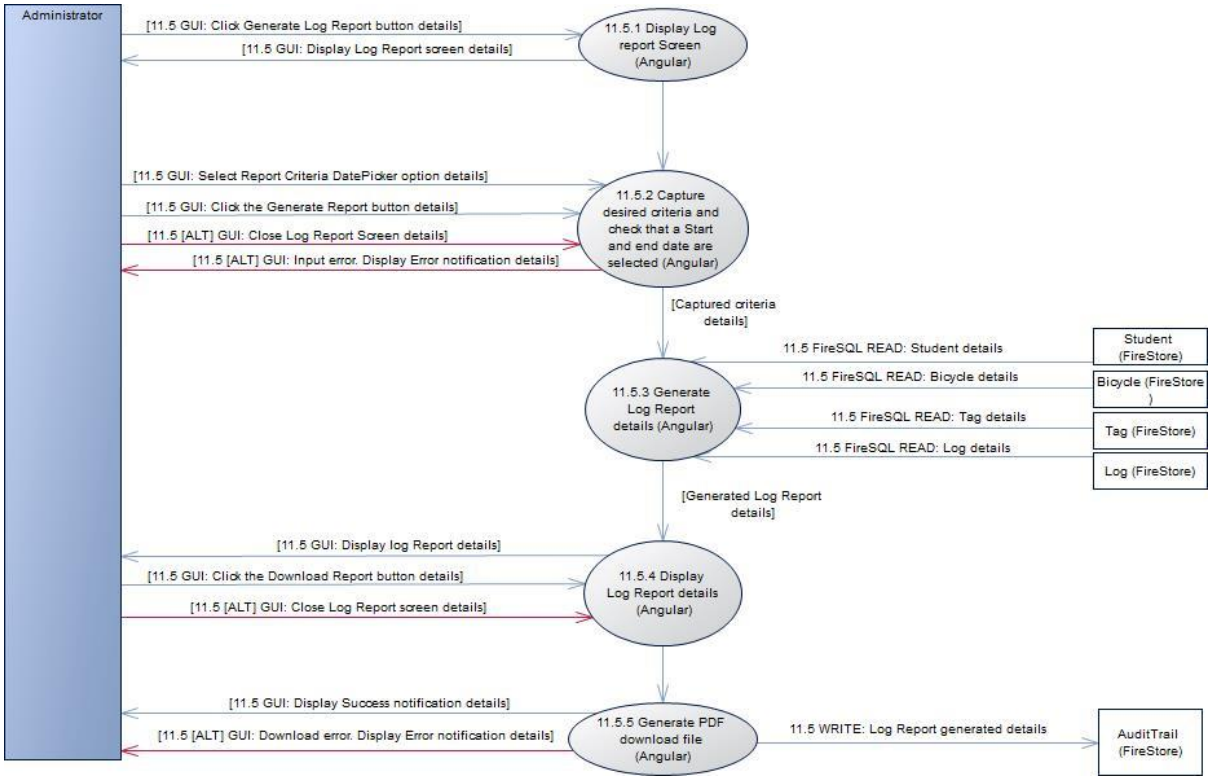


Figure 252 Primitive Diagram Sub-System 11: 11.5 Generate Log Report

11.6 Generate Parts Used Report

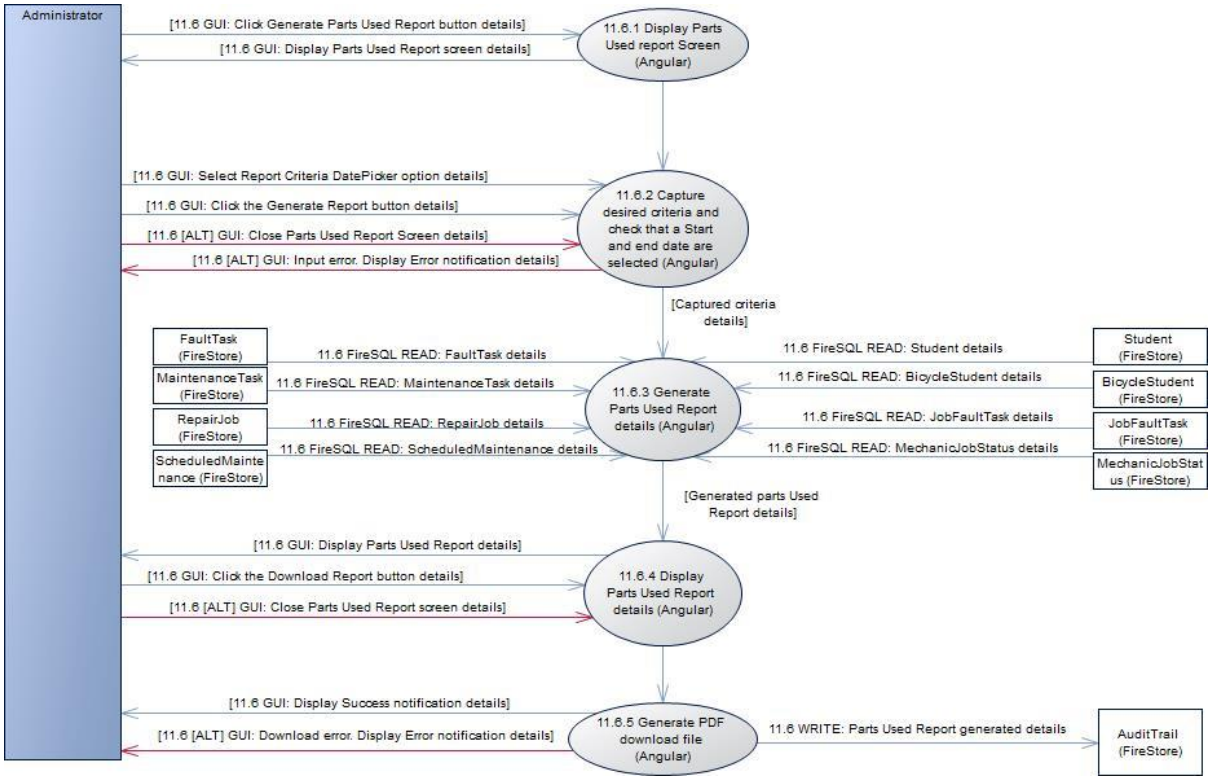


Figure 253 Primitive Diagram Sub-System 11: 11.6 Generate Parts Used Report

Sub-System 12: Notification

12.1 Send Student did not arrive notification

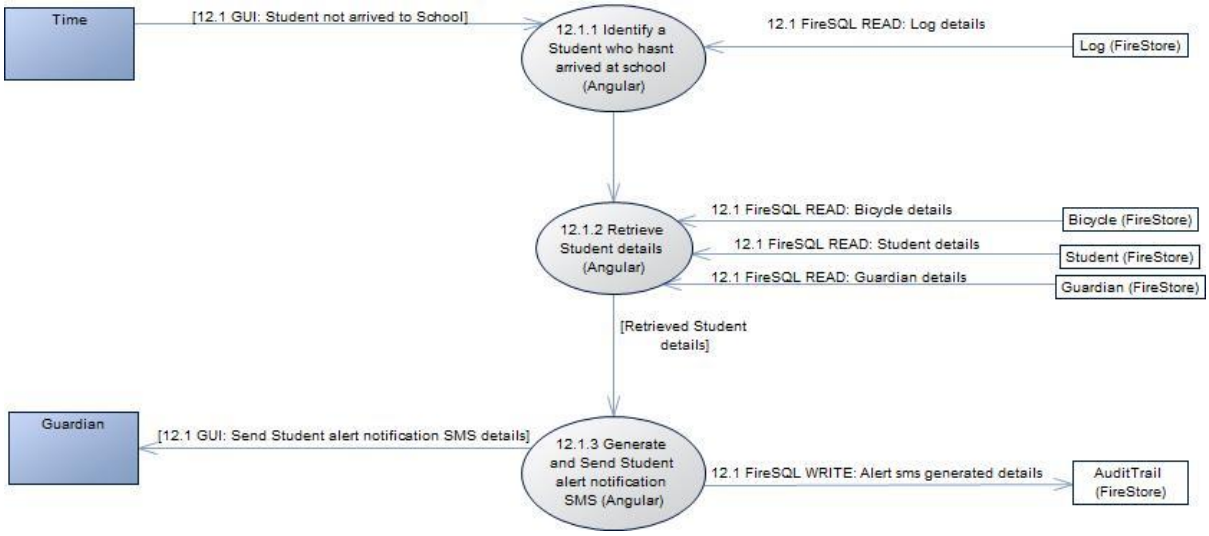


Figure 254 Primitive Diagram Sub-System 12: 12.1 Send Student did not arrive notification

12.2 Send Check-in notification

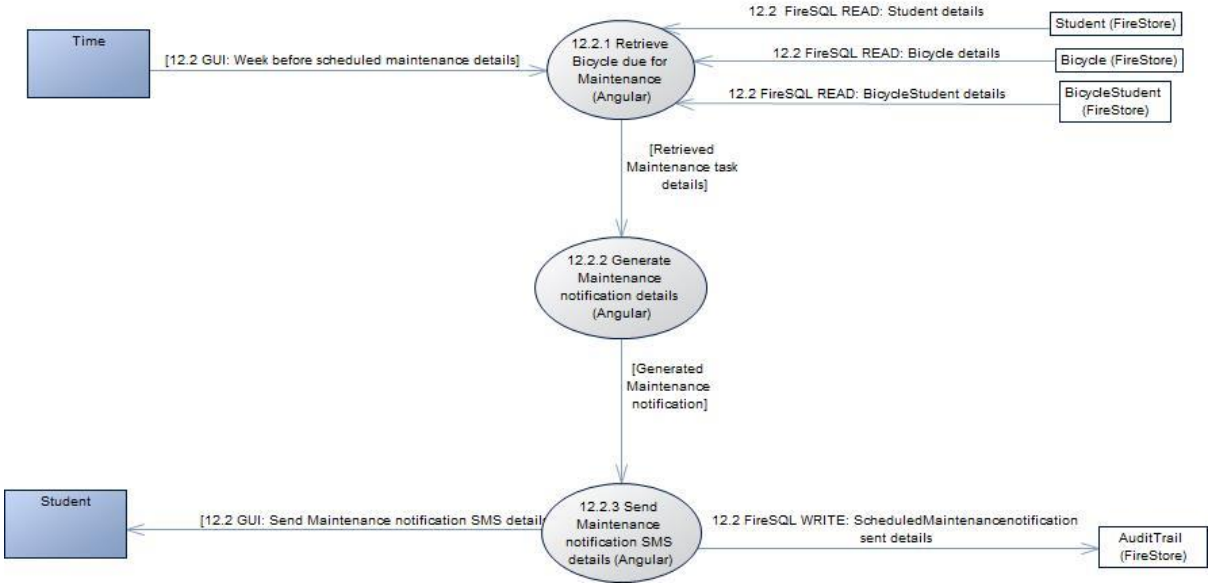


Figure 255 Primitive Diagram Sub-System 12: 12.2 Send Check-In Notification

Sub-System 13: Administration

13.1 Backup Data

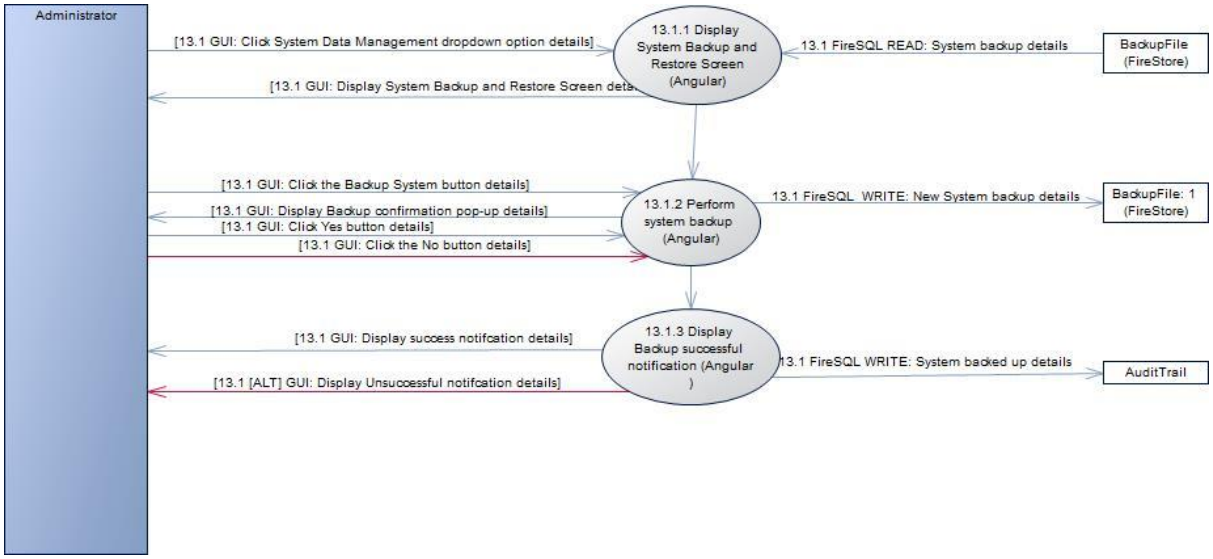


Figure 256 Primitive Diagram Sub-System 13: 13.1 Backup Data

13.2 Restore Data

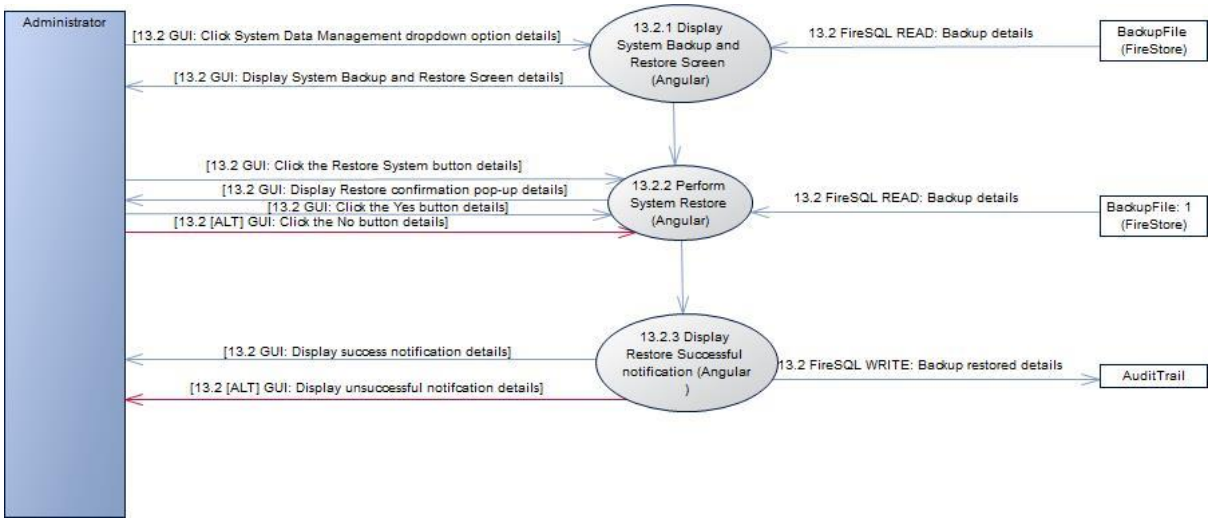


Figure 257 Primitive Diagram Sub-System 13: 13.1 Restore Data

13.3 View Audit Trail Log

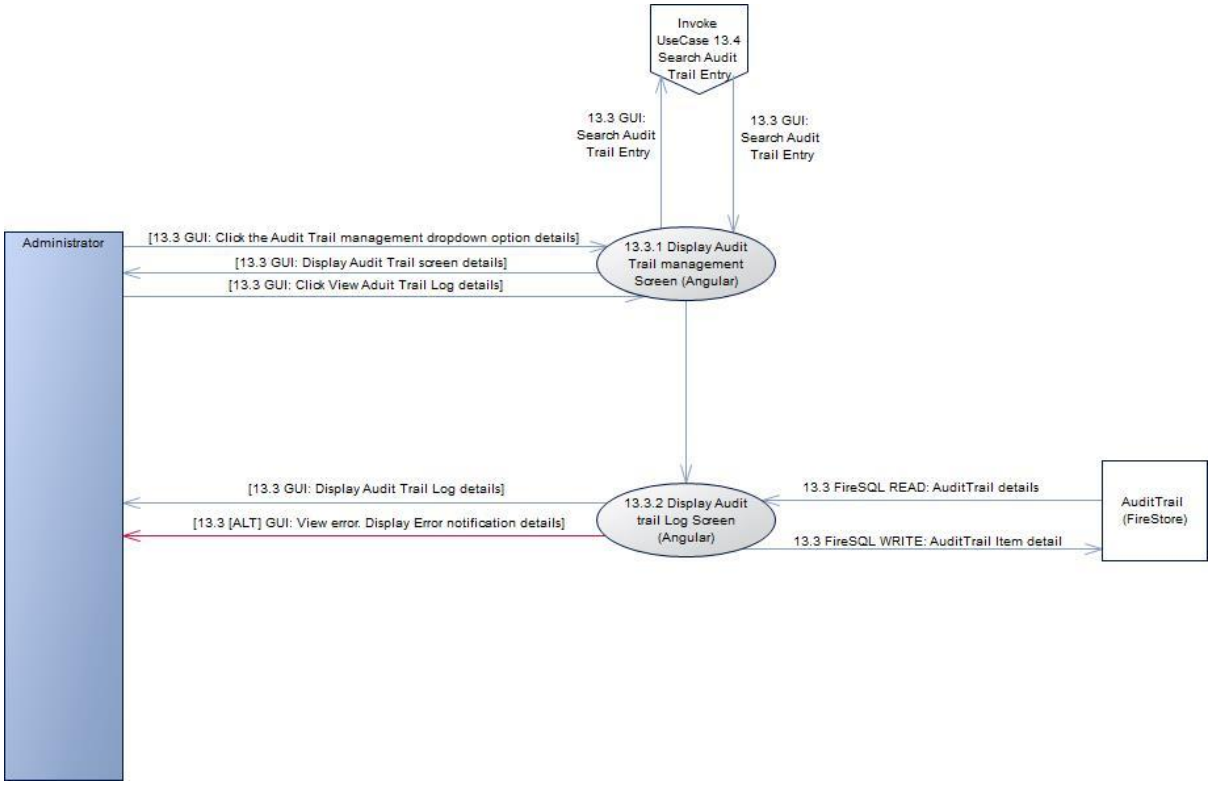


Figure 258 Primitive Diagram Sub-System 13: 13.3 View Audit Trail Log

13.4 Search Audit Trail Entry

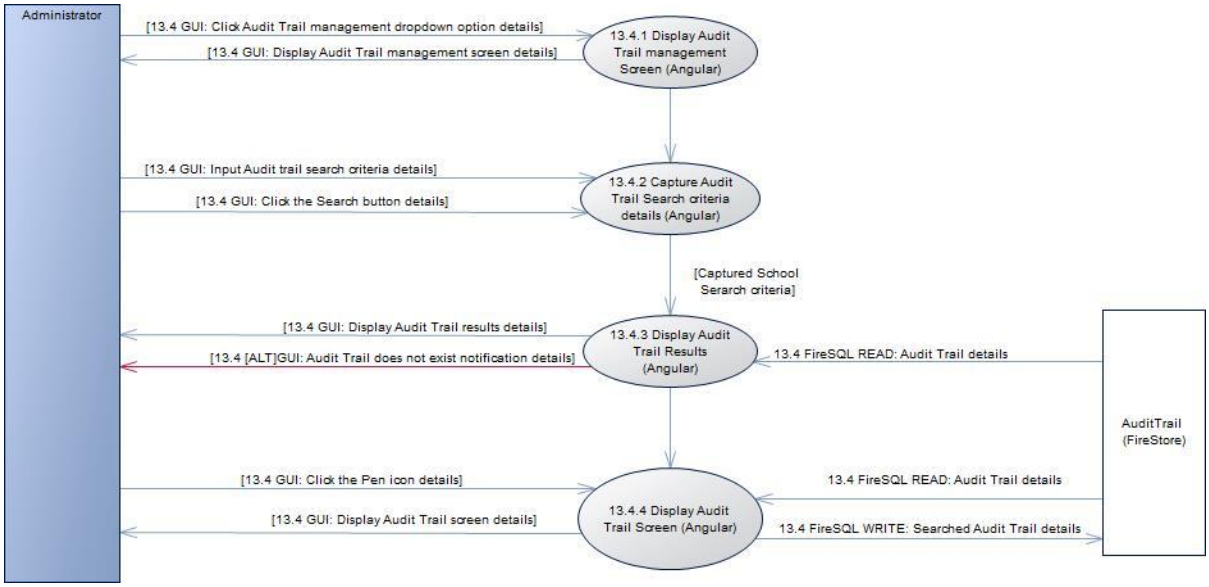


Figure 259 Primitive Diagram Sub-System 13: 134 Search Audit Trail Entry

2. Pseudo Code

Use Case 1.1: Login

Process 1.1.1: Display User Login screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Login" navbar-item.

// Login Screen

Page.Load "Login" screen.

Display

"Welcome Back!" Label

"Please log into your account" Label

"Username" Label

"Password" Label

"Password" Text box

"Username" Text box

"Login" Button

"Forgot Password" Link

Process 1.1.2: Capture and checker User login details have been filled

// Login screen

IF (Onlick = "Login")

IF (Fields fill = true)

Proceed to next Process.

ELSE

Proceed to Alt step 1

ELSE IF (Onlick = "Forgot Password")

Proceed to Alt Step 2

Alt Step 1

// Login screen

Alert.Show "Input Error"

Display below input field of error

"Please complete this field." Text

"X" Button

Alt Step 2

//Login screen

INVOKE USE CASE 1.3 Forgot Password

Process 1.1.3: Verify User login details.

Verification {

FireSQL READ Username from User table

FireSQL READ Password from User table

The system will compare the inputted username and password with the Username and Password in the database.

IF (password = Password && username = Username)

Return True

ELSE IF

Return False

}

IF (Verification = true)

Proceed to next Process

ELSE IF (Verification = False)

Proceed to Alt step

Alt Step

//Login screen

Alert.Show "Input Error"

Display below input field of error

"The Login Details were incorrect, Please Try Again." Text

"X" Button

Process 1.1.4: Display Homepage screen

//Homepage

Page.Load "Homepage" screen.

Use Case 1.2: Logout

Process 1.2.1: Display Logout confirmation popup

//Homepage

Page.Load "Homepage" screen.

Onclick "Logout" navbar-item.

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to Logout?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

ELSE IF (onclick = "No"

Proceed to Alt step

Alt step

// Homepage

Page.Load "Homepage" screen

Process 1.2.2: Display Login screen details

// Login Screen

Page.Load "Login" screen.

Display

"Welcome Back!" Label

"Please log into your account" Label

"Username" Label

"Password" Label

"Password" Text box

"Username" Text box

"Login" Button

"Forgot Password" Link

Use Case 1.3: Forgot Password

Process 1.3.1: Display Forgot Password screen

// Login Screen

Page.Load "Login" screen.

Display

"Welcome Back!" Label

"Please log into your account" Label

"Username" Label

"Password" Label

"Password" Text box

"Username" Text box

"Login" Button

"Forgot Password" Link

Onclick "Forgot Password" Link

// Forgot Password

Page.Load "Forgot Password" screen.

Display

"Forgot Password" Label

"Please enter your email" Label

"Email" Label

"Email" Text box

"Login" Button

"Cancel" Button

Process 1.3.2: Capture and validate User email address field is filled

// Login screen

(Onlick = "Submit")

IF (Fields fill = true)

IF (Validation = true)

Proceed to next Process

ELSE IF

Proceed to Alt step 2

ELSE

Proceed to Alt step 1

Alt Step 1

// Forgot Password screen

Alert.Show "Input Error"

Display below input field of error

"Please complete this field." Text

"X" Button

Alt Step 2

// Forgot Password screen

Alert.Show "Input Error"

Display below input field of error

"Invalid input." Text

"X" Button

Process 1.3.3: Verify inputted User email address details

Verification {

FireSQL READ Email from the User table

The system will compare the inputted email with existing Emails in the User table

IF (Email = email)

Return True

ELSE IF

Return False

}

IF (Verification = true)

Proceed to next Process

ELSE IF (Verification = false)

Proceed to Alt step

Process 1.3.4: Send Rest Password link

//Forgot Password screen

The system will send a Reset Password link to the inputted email.

Modal.Load "Successful" modal

Display

"Successful" Header

"Forgot Password email sent." Text

"Okay" Button

Process 1.3.5: Display rest password screen

//User Email

Onclick "Reset Password" Link

// Reset Password screen

Page.Load "Reset Password" screen.

Display

"Password Reset" Label

"Please enter your new password" Label

"Password" Label

"Confirm Password" Label

"Password" Text box

"Confirm Password" Text box

"Submit" Button

"Cancel" Button

Process 1.3.6: Capture and Verify new User password details

// Reset Password screen

Onclick "Submit" button

IF (Passwords = match)

Proceed to next Process

ELSE IF

Proceed to Alt step

Alt Step

//Reset Password screen

Alert.Show "Input Error"

Display below input field of error

"Password do not match." Text

"X" Button

Process 1.3.7: Update Password. Display Reset password Successful notification

// Reset Password screen

FireSQL UPDATE Password in the User table

FireSQL WRITE Updated use password details

IF (Reset password = success)

Modal.Load "Successful" modal

Display

"Successful" Header

"Password successfully updated." Text

"Okay" Button

ELSE IF

Proceed to Alt step

Alt Step

// Forgot Password screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Reset password failed. Please try again. Text

"Okay" Button

Use Case 1.4: Add User

Process 1.4.1: Display User Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "User Management" navbar-dropdown.

Display "User Management" navbar-dropdown-items.

Onclick "User Management" navbar-dropdown-item.

// User Management Screen

Page.Load "User Management" screen.

Display

"User Management" Label

"Add User" Button

"Search" Text box

"Search" Button

Display and Populate

"User Management" Table

"First Name" Table Header

FireSQL READ FirstName from User table

"Surname" Table Header

FireSQL READ Surname from User table

"Surname" Table Header

FireSQL READ UserType from UserType table

"Action" Table Header

"Pencil / View" Button

Process 1.4.2: Display add User screen

// User Management screen

Onclick "Add" button.

// Create User screen

Page.Load "Create User" screen.

Display

"User" Label

"Name" Label

"Surname" Label

"Email" Label

"Username" Label

"New Password" Label

"Confirm Password" Label

"Phone Number" Label

"Name" Text box

"Surname" Text box

"Email" Text box

"New Password" Text box

"Confirm Password" Text box

"User Type" Combo box

FireSQL READ UserType from UserType table

"Continue" Button

"Cancel" Button

Onclick "Continue" button.

//Create User continue screen

Page.Load" Create User continue" screen

Display

"School Administrator" Heading

"Position" Label

School" Label

"Position" Combo box

FireSQL READ Position from AdminSchool table

"School" Combo box

FireSQL READ SchoolName from School table

"Create" Button

"Return" Button

Process 1.4.3 Capture User details and check that all fields are filled in and an option selected in Combo box

// Create User screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create User screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 1.4.4: Validate User details

//Create User screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create User screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 1.4.5: Verify User Details

//Create User screen

Verification

{

FireSQL READ UserName from User table

Compare UserNames with Inputted User Name

IF (User exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the User?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create User screen

Go back to Process 1.1.2

Alt Step 2

// Create User screen

Modal.Load "Error" modal

Display

"Error" Header

"The User already exists. Please try again." Text

"Okay" Button

Process 1.4.6: Generate UserID and save User details

FireSQL READ previous UserID from User table

FireSQL WRITE the following User details to the User table

UserID

Name

Surname

Email

Username

UserTypeID

Gender

DateOfBirthPassword

PhoneNumber

FireSQL WRITE the following Admin details to the AdminSchool table

AdminSchoolID

Position

SchoolID

Process 1.4.7: Display add User successful notification

// Create User screen

IF (Save = Successful)

FireSQL WRITE User added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The User was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create User screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 1.5: Search User

Process 1.5.1: Display User Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "User Management" navbar-dropdown.

Display "User Management" navbar-dropdown-items.

Onclick "User Management" navbar-dropdown-item.

// User Management Screen

Page.Load "User Management" screen.

Display

"User Management" Label

"Add" Button

"Search" Text box

"Search" Button

Display and Populate

"User Management" Table

"First Name" Table Header

FireSQL READ FirstName from User table

"Surname" Table Header

FireSQL READ Surname from User table

"Surname" Table Header

FireSQL READ UserType from UserType table

"Action" Table Header

"Pencil / View" Button

Process 1.5.2: Capture User search criteria details

System captures inputted search criteria in the "Search" Text box

Process 1.5.3: Display searched User results

// User Management screen

Search Criteria

{

FireSQL READ Name && Surname from User table

Compare Surname and Name with Inputted Name or Surname

IF (User exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "User Management" screen.

Display

"User Management" Label

"Add User" Button

"Search" Text box

"Search" Button

Display and Populate

"First Name" Table Header

FireSQL READ FirstName from User table

"Surname" Table Header

FireSQL READ Surname from User table

"Surname" Table Header

FireSQL READ UserType from UserType table

"Action" Table Header

"Pencil / View" Button

ELSE IF

Proceed to Alt step

Alt Step

// User Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"User not found. Please try again." Text

"Okay" Button

Process 1.5.4: Display searched User details screen

// User Management screen

Onclick "Pencil / View" button

// View User screen

Page.Load "View User" screen

Display

"User" Label

"Name" Label

"Surname" Label

"Email" Label

"Username" Label

"New Password" Label

"Confirm Password" Label

"Phone Number" Label

"Name" Text box

FireSQL READ Name from User table

"Surname" Text box

FireSQL READ Surname from User table

"Email" Text box

FireSQL READ Email from User table

"New Password" Text box

"Confirm Password" Text box

"User Type" Combo box

FireSQL READ UserType from UserType table

"Update" Button

"Delete" Button

FireSQL WRITE User searched details to AuditTrail table

Use Case 1.6: Update User

Process 1.6.1: Display User Management screen

// User Management Screen

Page.Load "User Management" screen.

Display

"User Management" Label

"Add User" Button

"Search" Text box

"Search" Button

Display and Populate

"First Name" Table Header

FireSQL READ FirstName from User table

"Surname" Table Header

FireSQL READ Surname from User table

"Surname" Table Header

FireSQL READ UserType from UserType table

"Action" Table Header

"Pencil / View" Button

Process 1.6.2: Display searched User screen

// User Management screen

Onclick "Pencil / View" button

// View User screen

Page.Load "View User" screen

Display

"User" Label

"Name" Label

"Surname" Label

"Email" Label

"Username" Label

"New Password" Label

"Confirm Password" Label

"Phone Number" Label

"Name" Text box

FireSQL READ Name from User table

"Surname" Text box

FireSQL READ Surname from User table

"Email" Text box

FireSQL READ Email from User table

"New Password" Text box

"Confirm Password" Text box

"User Type" Combo box

FireSQL READ UserType from UserType table

"Update" Button

"Delete" Button

INVOKE USE CASE 1.2: Search User

Process 1.6.3: Capture User details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View User screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 1.6.4: Validate updated User details

// View User screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the User?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View User screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 1.6.5: Save updated User details

Fire SQL UPDATE the following User details in User table:

UserID

Name

Surname

Email

Username

UserTypeID

Gender

DateOfBirthPassword

PhoneNumber

FireSQL UPDATE the following Admin details to the AdminSchool table

AdminSchoolID

Position

SchoolID

Process 1.6.6: Display update successful notification

// View User screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The User was successfully updated" Text

"Okay" Button

WRITE User updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 1.7: Delete User

Process 1.7.1: Display User Management screen

// User Management Screen

Page.Load "User Management" screen.

Display

"User Management" Label

"Add User" Button

"Search" Text box

"Search" Button

Display and Populate

"First Name" Table Header

FireSQL READ FirstName from User table

"Surname" Table Header

FireSQL READ Surname from User table

"Surname" Table Header

FireSQL READ UserType from UserType table

"Action" Table Header

"Pencil / View" Button

INVOKE USE CASE 1.2: Search User

Process 1.7.2: Display searched User screen

// View User screen

Page.Load "View User" screen

Display

"User" Label

"Name" Label

"Surname" Label

"Email" Label

"Username" Label

"New Password" Label

"Confirm Password" Label

"Phone Number" Label

"Name" Text box

FireSQL READ Name from User table

"Surname" Text box

FireSQL READ Surname from User table

"Email" Text box

FireSQL READ Email from User table

"New Password" Text box

"Confirm Password" Text box

"User Type" Combo box

FireSQL READ UserType from UserType table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 1.7.3: Delete searched User screen details

//View User screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the User?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from User table

UserID

Name

Surname

Email

Username

UserTypeID

Gender

DateOfBirthPassword

PhoneNumber

FireSQL DELETE the following Admin details to the AdminSchool table

AdminSchoolID

Position

SchoolID

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 1.7.2

Process 1.7.4: Display User deleted successfully notification

// View User screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The User was successfully added" Text

"Okay" Button

WRITE User deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View User screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 1.8: Add User Type

Process 1.8.1: Display User Type Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "User Management" navbar-dropdown.

Display "User Management" navbar-dropdown-items.

Onclick "User Type Management" navbar-dropdown-item.

// User Type Management Screen

Page.Load "User Type Management" screen.

Display

"User Type Management" Label

"Add User Type" Button

"Search" Text box

"Search" Button

Display and Populate

"User Type Management" Table

"User Type" Table Header

FireSQL READ UserType from UserType table

"Permisions" Table Header

FireSQL READ Permisions from UserType table

"Action" Table Header

"Pencil / Edit" Button

Process 1.8.2: Display add User Type screen

// User Type Management screen

Onclick "Add User Type" button.

// Create User Type screen

Page.Load "Create User Type" screen.

Display

"User Type" Heading

"User Type" Label

"Permissions" Label

"User Type" Text box

"Permissions" Combo box

FireSQL READ UserPermisions from UserType table

"Create" Button

"Cancel" Button

Process 1.8.3: Capture User Type details and check that all fields are filled in and an option selected in Combo box

// Create User Type screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create User Type screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 1.8.4: Validate User Type details

//Create User Type screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create User Type screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 1.8.5: Verify User Type Details

//Create User Type screen

Verification

{

FireSQL READ UserType from User Type table

Compare UserType with Inputted User Type Name

IF (User Type exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the User Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create User Type screen

Go back to Process 1.8.2

Alt Step 2

// Create User Type screen

Modal.Load "Error" modal

Display

"Error" Header

"The User Type already exists. Please try again." Text

"Okay" Button

Process 1.8.6: Generate UserTypeID and save User Type details

FireSQL READ previous UserTypeID from UserType table

FireSQL WRITE the following User Type details to the UserType table

UserTypeID

User Type

Permissions

Process 1.8.7: Display add User Type successful notification

// Create User Type screen

IF (Save = Successful)

FireSQL WRITE User Type added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The User Type was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create User Type screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 1.9: Search User Type

Process 1.9.1: Display User Type Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "User Management" navbar-dropdown.

Display "User Management" navbar-dropdown-items.

Onclick "User Type Management" navbar-dropdown-item.

// User Type Management Screen

Page.Load "User Type Management" screen.

Display

"User Type Management" Heading

"Add User Type" Button

"Search" Text box

"Search" Button

Display and Populate

"User Type Management" Table

"User Type" Table Header

FireSQL READ UserType from UserType table

"Permisions" Table Header

FireSQL READ Permisions from UserType table

"Action" Table Header

"Pencil / Edit" Button

Process 1.9.2: Capture User Type search criteria details

System captures inputted search criteria in the "Search" Text box

Process 1.1.10: Display searched User Type results

// User Type Management screen

Search Criteria

{

FireSQL READ UserType from UserType table

Compare UserType with Inputted User Type Name

IF (User Type exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "User Type Management" screen.

Display

"User Type Management" Heading

"Add User Type" Button

"Search" Text box

"Search" Button

Display and Populate

"User Type Management" Table

"User Type" Table Header

FireSQL READ UserType from UserType table

"Permisions" Table Header

FireSQL READ Permisions from UserType table

"Action" Table Header

"Pencil / Edit" Button

ELSE IF

Proceed to Alt step

Alt Step

// User Type Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"User Type not found. Please try again." Text

"Okay" Button

Process 1.1.11: Display searched User Type details screen

// User Type Management screen

Onclick "Pencil / Edit" button

// View User Type screen

Page.Load "View User Type" screen

Display

"User Type" Heading

"User Type" Label

"Permissions" Label

"User Type" Text box

FireSQL READ UserType from UserType table

"Permissions" Combo box

FireSQL READ UserPermisions from UserType table

"Update" Button

"Delete" Button

FireSQL WRITE User Type searched details to AuditTrail table

Use Case 1.10: Update User Type

Process 1.10.1: Display User Type Management screen

// User Type Management Screen

Page.Load "User Type Management" screen.

Display

"User Type Management" Label

"Add User Type" Button

"Search" Text box

"Search" Button

Display and Populate

"User Type Management" Table

"User Type" Table Header

FireSQL READ UserType from UserType table

"Permisions" Table Header

FireSQL READ Permisions from UserType table

"Action" Table Header

"Pencil / Edit" Button "Pencil / Edit" Button

Process 1.10.2: Display searched User Type screen

// User Type Management screen

Onclick "Pencil / Edit" button

// View User Type screen

Page.Load "View User Type" screen

Display

"User Type" Header

"User Type" Label

"User Permisions" Label

"User Type" Text box

FireSQL READ UserType from UserType table

"User Permissions" Combo box

FireSQL READ UserPermissions from UserType table

"Update" Button

"Delete" Button

INVOKE USE CASE 1.9: Search User Type

Process 1.10.3: Capture User Type details and check for empty fields and for unselected combo box

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View User Type screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 1.10.4: Validate updated User Type details

// View User Type screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the User Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View User Type screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 1.10.5: Save updated User Type details

Fire SQL UPDATE the following User Type details in UserType table:

UserType

UserPermissions

Process 1.10.6: Display update successful notification

// View User Type screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The User Type was successfully updated" Text

"Okay" Button

WRITE User Type updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 1.11: Delete User Type

Process 1.11.1: Display User Type Management screen

// User Type Management Screen

Page.Load "User Type Management" screen.

Display

"User Type Management" Label

"Add User Type" Button

"Search" Text box

"Search" Button

Display and Populate

"User Type Management" Table

"User Type" Table Header

FireSQL READ UserType from UserType table

"Permisions" Table Header

FireSQL READ Permisions from UserType table

"Action" Table Header

"Pencil / Edit" Button

INVOKE USE CASE 1.9: Search User Type

Process 1.11.2: Display searched User Type screen

// View User Type screen

Page.Load "View User Type" screen

Display

"User Type" Header

"User Type" Label

"User Permisions" Label

"User Type" Text box

FireSQL READ UserType from UserType table

"User Permissions" Combo box

FireSQL READ UserPermissions from UserType table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 1.11.3: Delete searched User Type screen details

//View User Type screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the User Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from User Type table

UserTypeID

UserType

UserPermissions

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 1.11.2

Process 1.11.4: Display User Type deleted successfully notification

// View User Type screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The User Type was successfully added" Text

"Okay" Button

WRITE User Type deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View User Type screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 1: User

Sub-system 2: Mechanic Schedule

Sub-system 3: Tracking Hardware

Use Case 3.1: Add Antenna

Process 3.1.1: Display Antenna Management screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Antenna Management" navbar-dropdown-item.

// Antenna Management Screen

Page.Load "Antenna Management" screen.

Display

"Antenna Management" Label

"Add Antenna" Button

"Search" Text box

"Search" Button

Display and Populate

"Antenna Management" Table

"Antenna Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Antenna table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Antenna table

"Antenna Status" Table Header

FireSQL READ AntennaStatus from AntennaStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.1.2: Display add Antenna screen

// Antenna Management screen

Onclick "Add Antenna" button.

// Create Antenna screen

Page.Load "Create Antenna" screen.

Display

"Antenna Serial Number" Label

"Co-ordinates" Label

"Date Installed" Label

"Suburb" Label

"Antenna Status" Label

"Antenna Serial Number" Text box

"Antenna Name" Text box

"Co-ordinates" Text box

"Suburb" Combo box

FireSQL READ SuburbName from Suburb table

"Date Installed" Date

"Antenna Status" Combo box

FireSQL READ AntennaStatus from AntennaStatus table

"Create" Button

"Cancel" Button

Process 3.1.3: Capture Antenna details and check for empty fields

// Create Antenna screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Antenna screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 3.1.4: Validate Antenna details

//Create Antenna screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Antenna screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 3.1.5: Verify Antenna Details

//Create Antenna screen

Verification

{

FireSQL READ Co-ordinates from Antenna table

Compare Co-ordinates with Inputted Co-ordinates

IF (Antenna exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Antenna?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Antenna screen

Go back to Process 3.1.2

Alt Step 2

// Create Antenna screen

Modal.Load "Error" modal

Display

"Error" Header

"The Antenna already exists. Please try again." Text

"Okay" Button

Process 3.1.6: Generate AntennaID and save Antenna details

FireSQL READ previous AntennaID from Antenna table

FireSQL WRITE the following Antenna details to the Antenna table

AntennaID

Co-ordinates

AntennaStatusID

DateInstalled

SuburbID

Process 3.1.7: Display add Antenna successful notification

// Create Antenna screen

IF (Save = Successful)

FireSQL WRITE Antenna added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Antenna was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Antenna screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 3.2: Search Antenna

Process 3.2.1: Display Antenna Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Antenna Management" navbar-dropdown-item.

// Antenna Management Screen

Page.Load "Antenna Management" screen.

Display

"Antenna Management" Label

"Add Antenna" Button

"Search" Text box

"Search" Button

Display and Populate

"Antenna Management" Table

"Antenna Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Antenna table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Antenna table

"Antenna Status" Table Header

FireSQL READ AntennaStatus from AntennaStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.2.2: Capture Antenna search criteria details

System captures inputted search criteria in the "Search" Text box

Process 3.2.3: Display searched Antenna results

// Antenna Management screen

Search Criteria

{

FireSQL READ AntennaName from Antenna table

Compare AntennaNames with Inputted Antenna Name

IF (Antenna exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Antenna Management" screen.

Display

"Antenna Management" Label

"Add Antenna" Button

"Search" Text box

"Search" Button

Display and Populate

"Antenna Management" Table

"Antenna Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Antenna table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Antenna table

"Antenna Status" Table Header

FireSQL READ AntennaStatus from AntennaStatus table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Antenna Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Antenna not found. Please try again." Text

"Okay" Button

Process 3.2.4: Display searched Antenna details screen

// Antenna Management screen

Onclick "Edit / Pencil" button

// View Antenna screen

Page.Load "View Antenna" screen

Display

"Antenna" Heading

"Antenna Serial Number" Label

"Co-ordinates" Label

"Date Installed" Label

"Suburb" Label

"Antenna Status" Label

"Antenna Serial Number" Text box

FireSQL READ AntennaSerialNumber from Antenna table

"Co-ordinates" Text box

FireSQL READ Co-ordinates from Antenna table

"Suburb" Combo box

FireSQL READ SuburbName from Suburb table

"Date Installed" Date

FireSQL READ DateInstalled from Antenna table

"Antenna Status" Combo box

FireSQL READ AntennaStatus from AntennaStatus table

"Update" Button

"Delete" Button

FireSQL WRITE Antenna searched details to AuditTrail table

Use Case 3.3: Update Antenna

Process 3.3.1: Display Antenna Management screen

// Antenna Management Screen

Page.Load "Antenna Management" screen.

Display

"Antenna Management" Label

"Add Antenna" Button

"Search" Text box

"Search" Button

Display and Populate

"Antenna Management" Table

"Antenna Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Antenna table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Antenna table

"Antenna Status" Table Header

FireSQL READ AntennaStatus from AntennaStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.3.2: Display searched Antenna screen

// Antenna Management screen

Onclick "Edit / Pencil" button

// View Antenna screen

Page.Load "View Antenna" screen

Display

"Antenna" Heading

"Antenna Serial Number" Label

"Co-ordinates" Label

"Date Installed" Label

"Suburb" Label

"Antenna Status" Label

"Antenna Serial Number" Text box

FireSQL READ AntennaSerialNumber from Antenna table

"Co-ordinates" Text box

FireSQL READ Co-ordinates from Antenna table

"Suburb" Combo box

FireSQL READ SuburbName from Suburb table

"Date Installed" Date

FireSQL READ DateInstalled from Antenna table

"Antenna Status" Combo box

FireSQL READ AntennaStatus from AntennaStatus table

"Update" Button

"Delete" Button

INVOKE USE CASE 3.2: Search Antenna

Process 3.3.3: Capture Antenna details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Antenna screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 3.3.4: Validate updated Antenna details

// View Antenna screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Antenna?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Antenna screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 3.3.5: Save updated Antenna details

Fire SQL UPDATE the following Antenna details in Antenna table:

Co-ordinates

AntennaStatusID

DateInstalled

SuburbID

Process 3.3.6: Display update successful notification

// View Antenna screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Antenna was successfully updated" Text

"Okay" Button

WRITE Antenna updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 3.4: Delete Antenna

Process 3.4.1: Display Antenna Management screen

// Antenna Management Screen

Page.Load "Antenna Management" screen.

Display

"Antenna Management" Label

"Add Antenna" Button

"Search" Text box

"Search" Button

Display and Populate

"Antenna Management" Table

"Antenna Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Antenna table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Antenna table

"Antenna Status" Table Header

FireSQL READ AntennaStatus from AntennaStatus table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 3.2: Search Antenna

Process 3.4.2: Display searched Antenna screen

// View Antenna screen

Page.Load "View Antenna" screen

Display

"Antenna" Heading

"Antenna Serial Number" Label

"Co-ordinates" Label

"Date Installed" Label

"Suburb" Label

"Antenna Status" Label

"Antenna Serial Number" Text box

FireSQL READ AntennaSerialNumber from Antenna table

"Co-ordinates" Text box

FireSQL READ Co-ordinates from Antenna table

"Suburb" Combo box

FireSQL READ SuburbName from Suburb table

"Date Installed" Date

FireSQL READ DateInstalled from Antenna table

"Antenna Status" Combo box

FireSQL READ AntennaStatus from AntennaStatus table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 3.4.3: Delete searched Antenna screen details

//View Antenna screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Antenna?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Antenna table

AntennaID

Co-ordinates

AntennaStatusID

DateInstalled

SuburbID

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 3.4.2

Process 3.4.4: Display Antenna deleted successfully notification

// View Antenna screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Antenna was successfully added" Text

"Okay" Button

WRITE Antenna deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Antenna screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 3.5: Add Tag

Process 3.5.1: Display Tag Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Tag Management" navbar-dropdown-item.

// Tag Management Screen

Page.Load "Tag Management" screen.

Display

"Tag Management" Label

"Add Tag" Button

"Search" Text box

"Search" Button

Display and Populate

"Tag Management" Table

"Tag Serial Number Name" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL Bicycle Code from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.5.2: Display add Tag screen

// Tag Management screen

Onclick "Add Tag" button.

// Create Tag screen

Page.Load "Create Tag" screen.

Display

"Tag Serial Number" Label

"Tag Serial Number" Text box

"Create" Button

"Cancel" Button

Process 3.5.3: Capture Tag details and check for empty fields

// Create Tag screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Tag screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 3.5.4: Validate Tag details

//Create Tag screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Tag screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 3.5.5: Verify Tag Details

//Create Tag screen

Verification

{

FireSQL READ TagSerialNumber from Tag table

Compare TagSerialNumber with Inputted Tag Serial Number

IF (Tag exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Tag?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Tag screen

Go back to Process 3.5.2

Alt Step 2

// Create Tag screen

Modal.Load "Error" modal

Display

"Error" Header

"The Tag already exists. Please try again." Text

"Okay" Button

Process 3.5.6: Generate TagID and save Tag details

FireSQL READ previous TagID from Tag table

FireSQL WRITE the following Tag details to the Tag table

TagID

TagStatusID

TagSerialNumber

Process 3.5.7: Display add Tag successful notification

// Create Tag screen

IF (Save = Successful)

FireSQL WRITE Tag added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Tag was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Tag screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 3.6: Search Tag

Process 3.6.1: Display Tag Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Tag Management" navbar-dropdown-item.

// Tag Management Screen

Page.Load "Tag Management" screen.

Display

"Tag Management" Label

"Add Tag" Button

"Search" Text box

"Search" Button

Display and Populate

"Tag Management" Table

"Tag Serial Number Name" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL Bicycle Code from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.6.2: Capture Tag search criteria details

System captures inputted search criteria in the "Search" Text box

Process 3.6.3: Display searched Tag results

// Tag Management screen

Search Criteria

{

FireSQL READ TagSerialNumber from Tag table

Compare TagSerialNumber with Inputted Tag Serial Number

IF (Tag exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Tag Management" screen.

Display

"Tag Management" Label

"Add Tag" Button

"Search" Text box

"Search" Button

Display and Populate

"Tag Management" Table

"Tag Serial Number Name" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL Bicycle Code from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Edit / Pencil" Button "Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Tag Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Tag not found. Please try again." Text

"Okay" Button

Process 3.6.4: Display searched Tag details screen

// Tag Management screen

Onclick "Edit / Pencil" button

// View Tag screen

Page.Load "View Tag" screen

Display

"Tag" Heading

"Tag Serial Number" Label

"Tag Status" Label

"Date Installed" Label

"Tag Serial Number" Text box

FireSQL READ TagSerialNumber from Tag table

"Tag Status" Text box

FireSQL READ TagStatus from TagStatus table

"Date Installed" Date

FireSQL READ DateInstalled from Tag table

"Update" Button

"Delete" Button

FireSQL WRITE Tag searched details to AuditTrail table

Use Case 3.7: Update Tag

Process 3.7.1: Display Tag Management screen

// Tag Management Screen

Page.Load "Tag Management" screen.

Display

"Tag Management" Label

"Add Tag" Button

"Search" Text box

"Search" Button

Display and Populate

"Tag Management" Table

"Tag Serial Number Name" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL Bicycle Code from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 3.7.6: Display searched Tag screen

// Tag Management screen

Onclick "Edit / Pencil" button

// View Tag screen

Page.Load "View Tag" screen

Display

"Tag" Heading

"Tag Serial Number" Label

"Tag Status" Label

"Date Installed" Label

"Tag Serial Number" Text box

FireSQL READ TagSerialNumber from Tag table

"Tag Status" Text box

FireSQL READ TagStatus from TagStatus table

"Date Installed" Date

FireSQL READ DateInstalled from Tag table

"Update" Button

"Delete" Button

INVOKE USE CASE 3.6: Search Tag

Process 3.7.3: Capture Tag details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Tag screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 3.7.4: Validate updated Tag details

// View Tag screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Tag?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Tag screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 3.7.5: Save updated Tag details

Fire SQL UPDATE the following Tag details in Tag table:

Co-ordinates

TagID

TagStatusID

TagSerialNumber

DateInstalled

Process 3.7.6: Display update successful notification

// View Tag screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Tag was successfully updated" Text

"Okay" Button

WRITE Tag updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 3.8: Delete Tag

Process 3.8.1: Display Tag Management screen

// Tag Management Screen

Page.Load "Tag Management" screen.

Display

"Tag Management" Label

"Add Tag" Button

"Search" Text box

"Search" Button

Display and Populate

"Tag Management" Table

"Tag Serial Number Name" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL Bicycle Code from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 3.6: Search Tag

Process 3.8.2: Display searched Tag screen

// View Tag screen

Page.Load "View Tag" screen

Display

"Tag" Heading

"Tag Serial Number" Label

"Tag Status" Label

"Date Installed" Label

"Tag Serial Number" Text box

FireSQL READ TagSerialNumber from Tag table

"Tag Status" Text box

FireSQL READ TagStatus from TagStatus table

"Date Installed" Date

FireSQL READ DateInstalled from Tag table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 3.8.3: Delete searched Tag screen details

//View Tag screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Tag?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Tag table

TagID

TagStatusID

TagSerialNumber

DateInstalled

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 3.8.2

Process 3.8.4: Display Tag deleted successfully notification

// View Tag screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Tag was successfully added" Text

"Okay" Button

WRITE Tag deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Tag screen

Modal.Load “Failed” modal

Display

“Failed” Header

“Update unsuccessful please try again” Text

“Okay” Button

Sub-system 4: Student

Use Case 4.1: Add Student

Process 4.1.1: Display Student Management screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Student Management" navbar-dropdown-item.

// Student Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

Process 4.1.2: Display add Student screen

// Student Management screen

Onclick "Add Student" button.

// Create Student screen

Page.Load "Create Student" screen.

Display

"Student" Heading

"Name" Label

"Surname" Label

"Phone Number" Label

"Date of Birth" Label

"Village" Label

"School" Label

"Name" Text box

"Surname" Text box

"Phone Number" Text box

"Date of Birth" Date

"Village" Combo box

FireSQL READ VillageName from Village table

"School" Combo box

FireSQL READ SchoolName from School table

"Guardian" Heading

"Guardian Name" Label

"Guardian Surname" Label

"Guardian Email" Label

"Guardian Phone Number" Label

"Guardian Name" Text box

"Guardian Surname" Text box

"Guardian Phone Number" Text box

"Create" Button

"Cancel" Button

Process 4.1.3: Capture Student details and check for empty fields

// Create Student screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Student screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 4.1.4: Validate Student details

//Create Student screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Student screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 4.1.5: Verify Student Details

//Create Student screen

Verification

{

FireSQL READ StudentName and StudentSurname from Student table

Compare StudentName and StudentSurname with Inputted Name and Surname

IF (Student exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Student?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Student screen

Go back to Process 4.1.2

Alt Step 2

// Create Student screen

Modal.Load "Error" modal

Display

"Error" Header

"The Student already exists. Please try again." Text

"Okay" Button

Process 4.1.6: Generate StudentID and save Student details

FireSQL READ previous StudentID from Student table

FireSQL WRITE the following Student details to the Student table

- StudentID
- StudentName
- StudentSurname
- PhoneNumber
- StudentDoB
- VillageID
- SchoolID
- GuardianID

FireSQL READ previous GuardianID from Guardian table

FireSQL WRITE the following Guardian details to the Guardian table

- GuardianID
- GuardianName
- GuardianEmail
- GuardianPhone

Process 4.1.7: Display add Student successful notification

// Create Student screen

IF (Save = Successful)

- FireSQL WRITE Student added details to AuditTrail table
- Modal.Load "Successful" modal
  - Display
    - "Successful" Header
    - "The Student was successfully added" Text
    - "Okay" Button

ELSE

- Proceed to Alt step

Alt Step

// Create Student screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 4.2: Search Student

Process 4.2.1: Display Student Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Student Management" navbar-dropdown-item.

// Student Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

Process 4.2.2: Capture Student search criteria details

System captures inputted search criteria in the "Search" Text box

Process 4.2.3: Display searched Student results

// Student Management screen

Search Criteria

{

FireSQL READ StudentName and StudentSurname from Student table

Compare StudentNames with Inputted Student Name

IF (Student exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Student Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Student not found. Please try again." Text

"Okay" Button

Process 4.2.4: Display searched Student details screen

// Student Management screen

Onclick "Edit / Pencil" button

// View Student screen

Page.Load "View Student" screen

Display

"Student" Heading

"Name" Label

"Surname" Label

"Phone Number" Label

"Date of Birth" Label

"Village" Label

"School" Label

"Name" Text box

FireSQL READ StudentName from Student table

"Surname" Text box

FireSQL READ StudentSurname from Student table

"Phone Number" Text box

FireSQL READ PhoneNumber from Student table

"Date of Birth" Date

FireSQL READ StudentDoB from Student table

"Village" Combo box

FireSQL READ VillageName from Village table

"School" Combo box

FireSQL READ SchoolName from School table

"Guardian" Heading

"Guardian Name" Label

"Guardian Surname" Label

"Guardian Email" Label

"Guardian Phone Number" Label

"Guardian Name" Text box

FireSQL READ GuardianName from Guardian table

"Guardian Email" Text box.

FireSQL READ GuardianEmail from Guardian table

"Guardian Phone Number" Text box

FireSQL READ GuardianPhone from Guardian table

"Update" Button

"Delete" Button

FireSQL WRITE Student searched details to AuditTrail table

Use Case 4.3: Update Student

Process 4.4.1: Display Student Management screen

// Student Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

Process 4.4.2: Display searched Student screen

// Student Management screen

Onclick "Edit / Pencil" button

// View Student screen

Page.Load "View Student" screen

Display

"Student" Heading

"Name" Label

"Surname" Label

"Phone Number" Label

"Date of Birth" Label

"Village" Label

"School" Label

"Name" Text box

FireSQL READ StudentName from Student table

"Surname" Text box

FireSQL READ StudentSurname from Student table

"Phone Number" Text box

FireSQL READ PhoneNumber from Student table

"Date of Birth" Date

FireSQL READ StudentDoB from Student table

"Village" Combo box

FireSQL READ VillageName from Village table

"School" Combo box

FireSQL READ SchoolName from School table

"Guardian" Heading

"Guardian Name" Label

"Guardian Surname" Label

"Guardian Email" Label

"Guardian Phone Number" Label

"Guardian Name" Text box

FireSQL READ GuardianName from Guardian table

"Guardian Email" Text box.

FireSQL READ GuardianEmail from Guardian table

"Guardian Phone Number" Text box

FireSQL READ GuardianPhone from Guardian table

"Update" Button

"Delete" Button

INVOKE USE CASE 4.2: Search Student

Process 4.3.3: Capture Student details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Student screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 4.4.4: Validate updated Student details

// View Student screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Student?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Student screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 4.3.5: Save updated Student details

FireSQL UPDATE the following Student details to the Student table

- StudentName
- StudentSurname
- PhoneNumber
- StudentDoB
- VillageID
- SchoolID
- GuardianID

FireSQL UPDATE the following Guardian details to the Guardian table

- GuardianName
- GuardianEmail
- GuardianPhone

Process 4.3.6: Display update successful notification

// View Student screen

IF (Update = Success)

- Modal.Load "Successful" modal
  - Display
    - "Successful" Header
    - "The Student was successfully updated" Text
    - "Okay" Button
- WRITE Student updated details to AuditTrail table

ELSE

- Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 4.4: Delete Student

Process 4.4.1: Display Student Management screen

// Student Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 4.2: Search Student

Process 4.4.2: Display searched Student screen

// View Student screen

Page.Load "View Student" screen

Display

"Student" Heading

"Name" Label

"Surname" Label

"Phone Number" Label

"Date of Birth" Label

"Village" Label

"School" Label

"Name" Text box

FireSQL READ StudentName from Student table

"Surname" Text box

FireSQL READ StudentSurname from Student table

"Phone Number" Text box

FireSQL READ PhoneNumber from Student table

"Date of Birth" Date

FireSQL READ StudentDoB from Student table

"Village" Combo box

FireSQL READ VillageName from Village table

"School" Combo box

FireSQL READ SchoolName from School table

"Guardian" Heading

"Guardian Name" Label

"Guardian Surname" Label

"Guardian Email" Label

"Guardian Phone Number" Label

"Guardian Name" Text box

FireSQL READ GuardianName from Guardian table

"Guardian Email" Text box.

FireSQL READ GuardianEmail from Guardian table

"Guardian Phone Number" Text box

FireSQL READ GuardianPhone from Guardian table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 4.4.3: Delete searched Student screen details

//View Student screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Student?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following Student details to the Student table

StudentID

StudentName

StudentSurname

PhoneNumber

StudentDoB

VillageID

SchoolID

GuardianID

FireSQL DELETE the following Guardian details to the Guardian table

GuardianID

GuardianName

GuardianEmail

GuardianPhone

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 4.4.2

Process 4.4.4: Display Student deleted successfully notification

// View Student screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Student was successfully added" Text

"Okay" Button

WRITE Student deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Student screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 4.5: Capture Student Mark

Process 4.5.1 Display Student Mark management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Student Management" navbar-dropdown.

Display "Student Mark Management" navbar-dropdown-items.

Onclick "Student Mark Management" navbar-dropdown-item.

// Student Mark Management Screen

Page.Load "Student Mark Management" screen.

Display

"Student Mark Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Student Mark Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL StudentSurname from Student table

"Current Mark" Table Header

FireSQL READ Mark from StudentMark table

"Action" Table Header

"Capture Mark" Button

Process 4.5.2 Display Capture Student Mark details

// Student Mark Management screen

Onclick "Capture Mark" button.

// Capture Student Mark screen

Page.Load "Capture Student Mark" screen.

Display

"Student Mark" Heading

"Mark" Label

"Mark Type" Label

"Date" Label

"Mark" Text box

"Mark Type" Combo box

FireSQL READ MarkType from MarkType table

"Date" DatePicker

"Capture" Button

"Cancel" Button

Process 4.5.3 Capture Student Mark Details and check for empty fields

// Capture Student Mark screen

IF (onclick = "Capture")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Capture Student Mark screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 4.5.4 Validate Student Mark details

//Caopture Student Mark screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Capture Student Mark screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 4.5.5 Verify Student Mark Details

//Capture Student Mark screen

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to capture this Student Mark?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Capture Student Mark screen

Go back to Process 4.5.2

Alt Step 2

// Capture Student Mark screen

Modal.Load "Error" modal

Display

"Error" Header

"The Student Mark could not be captured. Please try again." Text

"Okay" Button

Process 4.5.6: Save Student Mark details

FireSQL READ StudentID from Student table

FireSQL WRITE the following Student Mark details to the StudentMark table

StudentMarkID

MakrTypeID

Mark

Date

Process 4.5.7: Display capture Student Mark successful notification

// Capture Student Mark screen

IF (Save = Successful)

FireSQL WRITE Student Mark captured details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Student Mark was successfully captured" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Capture Student Mark screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Capture unsuccessful. Please try again" Text

"Okay" Button

Use Case 4.6: Search Student Mark

Process 4.6.1: Display Student Mark Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Student Management" navbar-dropdown.

Display "Student Management" navbar-dropdown-items.

Onclick "Student Mark Management" navbar-dropdown-item.

// Student Mark Management Screen

Page.Load "Student Mark Management" screen.

Display

"Student Mark Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Student Mark Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL StudentSurname from Student table

"Current Mark" Table Header

FireSQL READ Mark from StudentMark table

"Action" Table Header

"Capture Mark" Button

Process 4.6.2: Capture Student Mark search criteria details

System captures inputted search criteria in the "Search" Text box

Process 4.6.3: Display searched Student Mark results

// Student Mark Management screen

Search Criteria

{

FireSQL READ StudentName and StudentSurname from StudentMark table

Compare StudentName and StudentSurname with Inputted Student Name and Student Surname

IF (Student exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Student Mark Management" screen.

Display

"Student Mark Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Student Mark Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL StudentSurname from Student table

"Current Mark" Table Header

FireSQL READ Mark from StudentMark table

"Action" Table Header

"Capture Mark" Button

ELSE IF

Proceed to Alt step

Alt Step

// Student Mark Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Student not found. Please try again." Text

"Okay" Button

Use Case 4.7: Add Mark Type

Process 4.7.1: Display Mark Type Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Mark Type Management" navbar-dropdown-item.

// Student Management Screen

Page.Load "Mark Type Management" screen.

Display

"Mark Type Management" Label

"Add Mark Type" Button

"Search" Text box

"Search" Button

Display and Populate

"Mark Type Management" Table

"Name" Table Header

FireSQL READ MarkType from MarkType table

"Surname" Table Header

FireSQL READ Weighting from MarkType table

"Action" Table Header

"Edit / Pencil" Button

Process 4.7.2: Display add Mark Type screen

// Mark Type Management screen

Onclick "Add Mark Type" button.

// Create Mark Type screen

Page.Load "Create Mark Type" screen.

Display

"Mark Type" Heading

"Mark Type" Label

"Weighting" Label

"Mark Type" Text box

"Weighting" Text box

"Create" Button

"Cancel" Button

Process 4.7.3: Capture Mark Type details and check for empty fields

// Create Mark Type screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Mark Type screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 4.7.4: Validate Mark Type details

//Create Mark Type screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Mark Type screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 4.7.5: Verify Mark Type Details

//Create Mark Type screen

Verification

{

FireSQL READ MarkType and Weighting from Mark Type table

Compare MarkType and Weighting with Inputted Mark Type and Weighting

IF (Mark Type exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Mark Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Mark Type screen

Go back to Process 4.7.2

Alt Step 2

// Create Mark Type screen

Modal.Load "Error" modal

Display

"Error" Header

"The Mark Type already exists. Please try again." Text

"Okay" Button

Process 4.7.6: Generate MarkTypeID and save Mark Type details

FireSQL READ previous MarkTypeID from Mark Type table

FireSQL WRITE the following Mark Type details to the Mark Type table

MarkTypeID

MarkType

Weighting

Process 4.7.7: Display add Mark Type successful notification

// Create Mark Type screen

IF (Save = Successful)

FireSQL WRITE Mark Type added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Mark Type was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Mark Type screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 4.8: Search Mark Type

Process 4.8.1: Display Mark Type Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Mark Type Management" navbar-dropdown-item.

// Mark Type Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Mark Type Management" Table

"Name" Table Header

FireSQL READ MarkType from MarkType table

"Surname" Table Header

FireSQL READ Weighting from MarkType table

"Action" Table Header

"Edit / Pencil" Button

Process 4.8.2: Capture Mark Type search criteria details

System captures inputted search criteria in the "Search" Text box

Process 4.8.3: Display searched Mark Type results

// Mark Type Management screen

Search Criteria

{

FireSQL READ MarkTypeNam from MarkType table

Compare MarkType with Inputted Mark Type Name

IF (Mark Type exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Mark Type Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Mark Type Management" Table

"Name" Table Header

FireSQL READ MarkType from MarkType table

"Surname" Table Header

FireSQL READ Weighting from MarkType table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Mark Type Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Mark Type not found. Please try again." Text

"Okay" Button

Process 4.8.4: Display searched Mark Type details screen

// Mark Type Management screen

Onclick "Edit / Pencil" button

// View Mark Type screen

Page.Load "View Mark Type" screen

Display

"Mark Type" Heading

"Mark Type" Label

"Weighting" Label

"Mark Type" Text box

FireSQL READ MarkType from MarkType table

"Weighting" Text box

FireSQL READ Weighting from MarkType table

"Update" Button

"Delete" Button

FireSQL WRITE Mark Type searched details to AuditTrail table

Use Case 4.9: Update Mark Type

Process 4.4.1: Display Mark Type Management screen

// Mark Type Management Screen

Page.Load "Student Management" screen.

Display

"Student Management" Label

"Add Student" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

Process 4.4.9: Display searched Student screen

// Student Management screen

Onclick "Edit / Pencil" button

// View Student screen

Page.Load "View Student" screen

Display

"Mark Type" Heading

"Mark Type" Label

"Weighting" Label

"Mark Type" Text box

FireSQL READ MarkType from MarkType table

"Weighting" Text box

FireSQL READ Weighting from MarkType table

"Update" Button

"Delete" Button

INVOKE USE CASE 4.8: Search Mark Type

Process 4.9.3: Capture Mark Type details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Mark Type screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 4.4.4: Validate updated Mark Type details

// View Mark Type screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Mark Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Mark Type screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 4.9.5: Save updated Mark Type details

FireSQL UPDATE the following MarkType details to the Mark Type table

MarkType

Weighting

Process 4.9.6: Display update successful notification

// View Mark Type screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Mark Type was successfully updated" Text

"Okay" Button

WRITE Mark Type updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 4.10: Delete Mark Type

Process 4.10.1: Display Mark Type Management screen

// Mark Type Management Screen

Page.Load "Mark Type Management" screen.

Display

"Mark Type Management" Label

"Add Mark Type" Button

"Search" Text box

"Search" Button

Display and Populate

"Student Management" Table

"Name" Table Header

FireSQL READ StudentName from Student table

"Surname" Table Header

FireSQL READ StudentSurname from Student table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 4.8: Search Mark Type

Process 4.10.8: Display searched Mark Type screen

// View Mark Type screen

Page.Load "View Mark Type" screen

Display

"Mark Type" Heading

"Mark Type" Label

"Weighting" Label

"Mark Type" Text box

FireSQL READ MarkType from MarkType table

"Weighting" Text box

FireSQL READ Weighting from MarkType table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 4.10.9: Delete searched Mark Type screen details

//View Mark Type screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Mark Type?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following Mark Type details to the Mark Type table

MarkTypeID

MarkType

Weighting

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 4.10.8

Process 4.10.4: Display Mark Type deleted successfully notification

// View Mark Type screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Mark Type was successfully added" Text

"Okay" Button

WRITE Mark Type deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Mark Type screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 5: Bicycle

Use Case 5.1: Add Bicycle

Process 5.1.1: Display Bicycle Management screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "Bicycle Management" navbar-dropdown.

Display "Bicycle Management" navbar-dropdown-items.

Onclick "Bicycle Management" navbar-dropdown-item.

// Bicycle Management Screen

Page.Load "Bicycle Management" screen.

Display

"Bicycle Management" Label

"Add Bicycle" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Bicycle Make" Table Header

FireSQL READ BicycleMake from Bicycle table

"Bicycle Status" Table Header

FireSQL READ BicycleStatus from BicycleStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 5.1.2: Display add Bicycle screen

// Bicycle Management screen

Onclick "Add Bicycle" button.

// Create Bicycle screen

Page.Load "Create Bicycle" screen.

Display

"Bicycle" Heading

"Bicycle Code" Label

"Bicycle Make" Label

"Bicycle Code" Text box

"Bicycle Make" Text box

"Create" Button

"Cancel" Button

Process 5.1.3: Capture Bicycle details and check for empty fields

// Create Bicycle screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Bicycle screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 5.1.4: Validate Bicycle details

//Create Bicycle screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Bicycle screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 5.1.5: Verify Bicycle Details

//Create Bicycle screen

Verification

{

FireSQL READ BicycleCode from Bicycle table

Compare BicycleCode with Inputted Bicycle Code

IF (Bicycle exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Bicycle?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Bicycle screen

Go back to Process 5.1.2

Alt Step 2

// Create Bicycle screen

Modal.Load "Error" modal

Display

"Error" Header

"The Bicycle already exists. Please try again." Text

"Okay" Button

Process 5.1.6: Generate BicycleID and save Bicycle details

FireSQL READ previous BicycleID from Bicycle table

FireSQL WRITE the following Bicycle details to the Bicycle table

BicycleID

BicycleCode

BicycleMake

BicycleDateAdded

Process 5.1.7: Display add Bicycle successful notification

// Create Bicycle screen

IF (Save = Successful)

FireSQL WRITE Bicycle added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Bicycle screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 5.2: Search Bicycle

Process 5.2.1: Display Bicycle Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Bicycle Management" navbar-dropdown.

Display "Bicycle Management" navbar-dropdown-items.

Onclick "Bicycle Management" navbar-dropdown-item.

// Bicycle Management Screen

Page.Load "Bicycle Management" screen.

Display

"Bicycle Management" Label

"Add Bicycle" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Bicycle Make" Table Header

FireSQL READ BicycleMake from Bicycle table

"Bicycle Status" Table Header

FireSQL READ BicycleStatus from BicycleStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 5.2.2: Capture Bicycle search criteria details

System captures inputted search criteria in the "Search" Text box

Process 5.2.3: Display searched Bicycle results

// Bicycle Management screen

Search Criteria

{

FireSQL READ BicycleCode from Bicycle table

Compare BicycleCode with Inputted Bicycle Name

IF (Bicycle exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Bicycle Management" screen.

Display

"Bicycle Management" Label

"Add Bicycle" Button

"Search" Text box

"Search" Button

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Bicycle Make" Table Header

FireSQL READ BicycleMake from Bicycle table

"Bicycle Status" Table Header

FireSQL READ BicycleStatus from BicycleStatus table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Bicycle Management screen

Modal.Load "Error" modal

Display

"Error" Header

"Bicycle not found. Please try again." Text

"Okay" Button

Process 5.2.4: Display searched Bicycle details screen

// Bicycle Management screen

Onclick "Edit / Pencil" button

// View Bicycle screen

Page.Load "View Bicycle" screen

Display

"Bicycle" Heading

"Bicycle Code" Label

"Bicycle Make" Label

"Bicycle Date Added" Label

"Bicycle Code" Text box

READ BicycleCode from Bicycle table

"Bicycle Make" Text box

READ BicycleMake from Bicycle table

"Bicycle Telephone" Text box

READ DateAdded from Bicycle table

"Update" Button

"Delete" Button

FireSQL WRITE Bicycle searched details to AuditTrail table

Use Case 5.3: Update Bicycle

Process 5.3.1: Display Bicycle Management screen

// Bicycle Management Screen

Page.Load "Bicycle Management" screen.

Display

"Bicycle Management" Label

"Add Bicycle" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Bicycle Make" Table Header

FireSQL READ BicycleMake from Bicycle table

"Bicycle Status" Table Header

FireSQL READ BicycleStatus from BicycleStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 5.3.2: Display searched Bicycle screen

// Bicycle Management screen

Onclick "Edit / Pencil" button

// View Bicycle screen

Page.Load "View Bicycle" screen

Display

"Bicycle" Heading

"Bicycle Code" Label

"Bicycle Make" Label

"Bicycle Date Added" Label

"Bicycle Code" Text box

READ BicycleCode from Bicycle table

"Bicycle Make" Text box

READ BicycleMake from Bicycle table

"Bicycle Telephone" Text box

READ DateAdded from Bicycle table

"Update" Button

"Delete" Button

INVOKE USE CASE 5.2: Search Bicycle

Process 5.3.3: Capture Bicycle details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Bicycle screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 5.3.4: Validate updated Bicycle details

// View Bicycle screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Bicycle?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Bicycle screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 5.3.5: Save updated Bicycle details

Fire SQL UPDATE the following Bicycle details in Bicycle table:

BicycleMake

DateAdded

Process 5.3.6: Display update successful notification

// View Bicycle screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully updated" Text

"Okay" Button

WRITE Bicycle updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 5.4: Delete Bicycle

Process 5.4.1: Display Bicycle Management screen

// Bicycle Management Screen

Page.Load "Bicycle Management" screen.

Display

"Bicycle Management" Label

"Add Bicycle" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Bicycle Make" Table Header

FireSQL READ BicycleMake from Bicycle table

"Bicycle Status" Table Header

FireSQL READ BicycleStatus from BicycleStatus table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 5.2: Search Bicycle

Process 5.4.2: Display searched Bicycle screen

// View Bicycle screen

Page.Load "View Bicycle" screen

Display

"Bicycle" Heading

"Bicycle Code" Label

"Bicycle Make" Label

"Bicycle Date Added" Label

"Bicycle Code" Text box

READ BicycleCode from Bicycle table

"Bicycle Make" Text box

READ BicycleMake from Bicycle table

"Bicycle Telephone" Text box

READ DateAdded from Bicycle table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 5.4.3: Delete searched Bicycle screen details

//View Bicycle screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Bicycle?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Bicycle table

BicycleID

BicycleCode

BicycleMake

DateAdded

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 5.4.2

Process 5.4.4: Display Bicycle deleted successfully notification

// View Bicycle screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully added" Text

"Okay" Button

WRITE Bicycle deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Bicycle screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 5.5: Assign Bicycle to Student

Process 5.5.1: Display Bicycle Assignment screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Bicycle Management" navbar-dropdown.

Display "Bicycle Management" navbar-dropdown-items.

Onclick "Bicycle Assignment" navbar-dropdown-item.

// Bicycle Management Screen

Page.Load "Bicycle Assignment Management" screen.

Display

"Bicycle Assignment" Label

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Assignment" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Student Name" Table Header

FireSQL READ StudentName from Student table

"Student Surname" Table Header

FireSQL READ StudentSurname from Student table

"Action" Table Header

"Assign" Button

Process 5.5.2 Display Assign Bicycle to Student Screen

// Bicycle Management screen

Onclick "Assign" button.

// Assign Bicycle screen

Page.Load "Assign Bicycle" screen.

Display

"Bicycle" Heading

"Bicycle Make" Label

"Bicycle Code" Text box

"Select Student to Assign" Combo box

FireSQL READ StudentName from Student table

FireSQL READ StudentSurname from Student table

"Assign" Button

"Cancel" Button

Process 5.5.3 Capture and Validate Assign Bicycle to student details

//Assign Bicycle screen

Onclick "Assign" button

IF (Validation = False)

Proceed to Alt Step 1

ELSE

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to assign this Bicycle?" Text

"Yes" Button

"No" Button

IF (Onclick = "Yes")

Proceed to next Process

ELSE IF (Onclick = "No"

Proceed to Alt step 2

Alt Step 1

// Create Bicycle screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Alt Step 2

Go back to Process 5.5.2

Process 5.5.4 Update and Save inputted details

FireSQL UPDATE Bicycle table

BicycleStatusID

FireSQL UPDATE BicycleStudent table

BicycleID

StudentID

DateAssigned

Alt Step

// Create Bicycle screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 5.5.5 Display Assign Successful notification

// Assign Bicycle screen

IF (Save = Successful)

FireSQL WRITE Bicycle assigned details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully assigned" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Assign Bicycle screen

Modal.Load "Error" modal

Display

"Error" Header

"Bicycle assign unsuccessful. Please try again." Text

"Okay" Button

Use Case 5.6: Unassign Bicycle from Student

Process 5.6.1: Display Bicycle Assignment screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Bicycle Management" navbar-dropdown.

Display "Bicycle Management" navbar-dropdown-items.

Onclick "Bicycle Assignment" navbar-dropdown-item.

// Bicycle Management Screen

Page.Load "Bicycle Assignment Management" screen.

Display

"Bicycle Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Management" Table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Student Name" Table Header

FireSQL READ StudentName from Student table

"Student Surname" Table Header

FireSQL READ StudentSurname from Student table

"Action" Table Header

"Unassign" Button

Onclick "Assign" button.

Process 5.6.2 Update and Save Unassign Bicycle details

// Assign Bicycle screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to assign this Bicycle?" Text

"Yes" Button

"No" Button

IF (Onclick = "Yes")

FireSQL UPDATE BicycleStudent table

BicycleID

StudentID

DateAssigned

DateUnassigned

FireSQL UPDATE Bicycle table

BicycleStatusID

Alt Step

Go back to Process 5.6.1

Process 5.6.3 Display Unassign Successful notification

//Assign Bicycle screen

IF (Update = successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully unassigned" Text

"Okay" Button

ELSE IF

Proceed to Alt Step

Alt Step 1

// Assign Bicycle screen

Modal.Load "Failed" modal

Display

"Failed" Header

"The Bicycle unassign failed. Please try again." Text

"Okay" Button

Use Case 5.7: Assign Tag to Bicycle

Process 5.7.1: Display Tag Assignment screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Tag Management" navbar-dropdown.

Display "Tag Management" navbar-dropdown-items.

Onclick "Tag Assignment" navbar-dropdown-item.

// Tag Management Screen

Page.Load "Tag Assignment Management" screen.

Display

"Tag Assignment" Label

"Search" Text box

"Search" Button

Display and Populate

"Tag Assignment" Table

"Tag Serial Number" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Assign" Button

Process 5.7.2 Display Assign Tag to Student Screen

// Tag Management screen

Onclick "Assign" button.

// Assign Tag screen

Page.Load "Assign Tag" screen.

Display

"Tag" Heading

"Tag Serial Number" Label

"Tag Serial Number" Text box

"Select Bicycle to Assign" Combo box

FireSQL READ Bicycle Code from Bicycle table

"Assign" Button

"Cancel" Button

Process 5.7.3 Capture and Validate Assign Tag to student details

//Assign Tag screen

Onclick "Assign" button

IF (Validation = False)

Proceed to Alt Step 1

ELSE

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to assign this Tag?" Text

"Yes" Button

"No" Button

IF (Onclick = "Yes")

Proceed to next Process

ELSE IF (Onclick = "No"

Proceed to Alt step 2

Alt Step 1

// Create Tag screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Alt Step 2

Go back to Process 5.7.2

Process 5.7.4 Update and Save inputted details

FireSQL UPDATE Tag table

TagStatusID

DateInstalled

FireSQL UPDATE Bicycle table

TagID

Alt Step

// Create Tag screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 5.7.5 Display Assign Successful notification

// Assign Tag screen

IF (Save = Successful)

FireSQL WRITE Tag assigned details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Tag was successfully assigned" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Assign Tag screen

Modal.Load "Error" modal

Display

"Error" Header

"Tag assign unsuccessful. Please try again." Text

"Okay" Button

Use Case 5.8: Unassign Tag from Bicycle

Process 5.8.1: Display Tag Assignment screen

//Homepage

Page.Load "Homepage" screen.

Onclick "Tag Management" navbar-dropdown.

Display "Tag Management" navbar-dropdown-items.

Onclick "Tag Assignment" navbar-dropdown-item.

// Tag Management Screen

Page.Load "Tag Assignment Management" screen.

Display

"Tag Management" Label

"Search" Text box

"Search" Button

"Tag Assignment" Table

"Tag Serial Number" Table Header

FireSQL READ TagSerialNumber from Tag table

"Bicycle Code" Table Header

FireSQL READ BicycleCode from Bicycle table

"Tag Status" Table Header

FireSQL READ TagStatus from TagStatus table

"Action" Table Header

"Unassign" Button

Onclick "Unassign" button.

Process 5.8.2 Update and Save Unassign Tag details

// Assign Tag screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to assign this Tag?" Text

"Yes" Button

"No" Button

IF (Onclick = "Yes")

FireSQL UPDATE Bicycle table

TagID

FireSQL UPDATE Tag table

TagStatusID

DateInstalled

DateUninstalled

Alt Step

Go back to Process 5.8.1

Process 5.8.3 Display Unassign Successful notification

//Assign Bicycle screen

IF (Update = successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle was successfully unassigned" Text

"Okay" Button

ELSE IF

Proceed to Alt Step

Alt Step 1

// Assign Bicycle screen

Modal.Load "Failed" modal

Display

"Failed" Header

"The Bicycle unassign failed. Please try again." Text

"Okay" Button

Sub-system 6: Bicycle Part

Use Case 6.1: Add Bicycle Part

Process 6.1.1: Display Bicycle Part Management screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Bicycle Part Management" navbar-dropdown-item.

// Bicycle Part Management Screen

Page.Load "Bicycle Part Management" screen.

Display

"Bicycle Part Management" Label

"Add Bicycle Part" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Part Management" Table

"Part Name" Table Header

FireSQL READ PartName from Part table

"Part Description" Table Header

FireSQL READ PartDescription from Part table

"Section" Table Header

FireSQL READ SectionName from Section table

"Action" Table Header

"Edit / Pencil" Button

Process 6.1.2: Display add Bicycle Part screen

// Bicycle Part Management screen

Onclick "Add Bicycle Part" button.

// Create Bicycle Part screen

Page.Load "Create Bicycle Part" screen.

Display

"Bicycle Part" Heading

"Part Name" Label

"Part Picture" Picture Image

"Choose Picture" Button

"Part Description" Label

"Bicycle Section" Label

"Part Name" Text box

"Part Description" Text box

"Bicycle Section" Combo box

FireSQL READ BicycleSection from Section table

"Create" Button

"Cancel" Button

Process 6.1.3: Capture Bicycle Part details and check for empty fields

// Create Bicycle Part screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Bicycle Part screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 6.1.4: Validate Bicycle Part details

//Create Bicycle Part screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Bicycle Part screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 6.1.5: Verify Bicycle Part Details

//Create Bicycle Part screen

Verification

{

FireSQL READ PartName from Part table

Compare PartName with Inputted Part Name

IF (Part exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Bicycle Part?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Bicycle Part screen

Go back to Process 6.1.2

Alt Step 2

// Create Bicycle Part screen

Modal.Load "Error" modal

Display

"Error" Header

"The Bicycle Part already exists. Please try again." Text

"Okay" Button

Process 6.1.6: Generate Bicycle PartID and save Bicycle Part details

FireSQL READ previous PartID from Part table

FireSQL WRITE the following Bicycle Part details to the Part table

PartID

PartName

PartDescription

FireSQL WRITE the following Section Part details to the SectionPart table

SectionPartID

SectionID

PartID

Process 6.1.7: Display add Bicycle Part successful notification

// Create Bicycle Part screen

IF (Save = Successful)

FireSQL WRITE Bicycle Part added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Part was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Bicycle Part screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 6.2: Search Bicycle Part

Process 6.2.1: Display Bicycle Part Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Bicycle Part Management" navbar-dropdown-item.

// Bicycle Part Management Screen

Page.Load "Bicycle Part Management" screen.

Display

"Bicycle Part Management" Label

"Add Bicycle Part" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Part Management" Table

"Part Name" Table Header

FireSQL READ PartName from Part table

"Part Description" Table Header

FireSQL READ PartDescription from Part table

"Section" Table Header

FireSQL READ SectionName from Section table

"Action" Table Header

"Edit / Pencil" Button

Process 6.2.2: Capture Bicycle Part search criteria details

System captures inputted search criteria in the "Search" Text box

Process 6.2.3: Display searched Bicycle Part results

// Bicycle Part Management screen

Search Criteria

{

FireSQL READ PartName from Part table

Compare PartNames with Inputted Bicycle Part Name

IF (Bicycle Part exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Bicycle Part Management" screen.

Display

"Bicycle Part Management" Label

"Add Bicycle Part" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Part Management" Table

"Part Name" Table Header

FireSQL READ PartName from Part table

"Part Description" Table Header

FireSQL READ PartDescription from Part table

"Section" Table Header

FireSQL READ SectionName from Section table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Bicycle Part Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Bicycle Part not found. Please try again." Text

"Okay" Button

Process 6.2.4: Display searched Bicycle Part details screen

// Bicycle Part Management screen

Onclick "Edit / Pencil" button

// View Bicycle Part screen

Page.Load "View Bicycle Part" screen

Display

"Bicycle Part" Heading

"Part Name" Label

"Part Picture" Picture Image

"Choose Picture" Button

"Part Description" Label

"Bicycle Section" Label

"Part Name" Text box

FireSQL READ PartName from Part table

"Part Description" Text box

FireSQL READ PartDescription from Part table

"Bicycle Section" Combo box

FireSQL READ BicycleSection from Section table

"Update" Button

"Delete" Button

FireSQL WRITE Bicycle Part searched details to AuditTrail table

Use Case 6.3: Update Bicycle Part

Process 6.3.1: Display Bicycle Part Management screen

// Bicycle Part Management Screen

Page.Load "Bicycle Part Management" screen.

Display

"Bicycle Part Management" Label

"Add Bicycle Part" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Part Management" Table

"Bicycle Part Serial Number Name" Table Header

FireSQL READ AntenneaSerialNumber from Bicycle Part table

"Co-ordinates" Table Header

FireSQL READ Co-Ordinates from Bicycle Part table

"Bicycle Part Status" Table Header

FireSQL READ Bicycle PartStatus from Bicycle PartStatus table

"Action" Table Header

"Edit / Pencil" Button

Process 6.3.2: Display searched Bicycle Part screen

// Bicycle Part Management screen

Onclick "Edit / Pencil" button

// View Bicycle Part screen

Page.Load "View Bicycle Part" screen

Display

"Bicycle Part" Heading

"Part Name" Label

"Part Picture" Picture Image

"Choose Picture" Button

"Part Description" Label

"Bicycle Section" Label

"Part Name" Text box

FireSQL READ PartName from Part table

"Part Description" Text box

FireSQL READ PartDescription from Part table

"Bicycle Section" Combo box

FireSQL READ BicycleSection from Section table

"Update" Button

"Delete" Button

INVOKE USE CASE 6.2: Search Bicycle Part

Process 6.3.3: Capture Bicycle Part details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Bicycle Part screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 6.3.4: Validate updated Bicycle Part details

// View Bicycle Part screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Bicycle Part?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Bicycle Part screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 6.3.5: Save updated Bicycle Part details

Fire SQL UPDATE the following Bicycle Part details in Part table:

PartID

PartName

PartDescription

FireSQL UPDATE the following in the SectionPart table

SectionPartID

SectionID

PartID

Process 6.3.6: Display update successful notification

// View Bicycle Part screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Part was successfully updated" Text

"Okay" Button

WRITE Bicycle Part updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 6.4: Delete Bicycle Part

Process 6.4.1: Display Bicycle Part Management screen

// Bicycle Part Management Screen

Page.Load "Bicycle Part Management" screen.

Display

"Bicycle Part Management" Label

"Add Bicycle Part" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Part Management" Table

"Part Name" Table Header

FireSQL READ PartName from Part table

"Part Description" Table Header

FireSQL READ PartDescription from Part table

"Section" Table Header

FireSQL READ SectionName from Section table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 6.2: Search Bicycle Part

Process 6.4.2: Display searched Bicycle Part screen

// View Bicycle Part screen

Page.Load "View Bicycle Part" screen

Display

"Bicycle Part" Heading

"Part Name" Label

"Part Picture" Picture Image

"Choose Picture" Button

"Part Description" Label

"Bicycle Section" Label

"Part Name" Text box

FireSQL READ PartName from Part table

"Part Description" Text box

FireSQL READ PartDescription from Part table

"Bicycle Section" Combo box

FireSQL READ BicycleSection from Section table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 6.4.3: Delete searched Bicycle Part screen details

//View Bicycle Part screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Bicycle Part?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Part table

PartID

PartName

PartDescription

FireSQL DELETE the following details from SectionPart table

SectionPartID

SectionID

PartID

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 6.4.2

Process 6.4.4: Display Bicycle Part deleted successfully notification

// View Bicycle Part screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Part was successfully added" Text

"Okay" Button

WRITE Bicycle Part deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Bicycle Part screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 6.5: Add Bicycle Section

Process 6.5.1: Display Bicycle Section Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Bicycle Section Management" navbar-dropdown-item.

// Bicycle Section Management Screen

Page.Load "Bicycle Section Management" screen.

Display

"Bicycle Section Management" Label

"Add Bicycle Section" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Section Management" Table

"Section Name" Table Header

FireSQL READ Bicycle SectionName from Section table

"Section Description Code" Table Header

FireSQL READ SectionDescription from Section table

"Action" Table Header

"Edit / Pencil" Button

Process 6.5.2: Display add Bicycle Section screen

// Bicycle Section Management screen

Onclick "Add Bicycle Section" button.

// Create Bicycle Section screen

Page.Load "Create Bicycle Section" screen.

Display

"Bicycle Section" Heading

"Choose Picture" Button

"Section Picture" Picture Image

"Section Name" Label

"Section Description" Label

"Section Name" Text box

"Section Description" Text box

"Create" Button

"Cancel" Button

Process 6.5.3: Capture Bicycle Section details and check for empty fields

// Create Bicycle Section screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Bicycle Section screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 6.5.4: Validate Bicycle Section details

//Create Bicycle Section screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Bicycle Section screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 6.5.5: Verify Bicycle Section Details

//Create Bicycle Section screen

Verification

{

FireSQL READ Bicycle SectionName from Section table

Compare Bicycle SectionName with Inputted Section Name

IF (Section exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Bicycle Section?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Bicycle Section screen

Go back to Process 6.5.2

Alt Step 2

// Create Bicycle Section screen

Modal.Load "Error" modal

Display

"Error" Header

"The Bicycle Section already exists. Please try again." Text

"Okay" Button

Process 6.5.6: Generate Bicycle SectionID and save Bicycle Section details

FireSQL READ previous SectionID from Bicycle Section table

FireSQL WRITE the following Bicycle Section details to the Section table

SectionID

SectionName

SectionDescription

Process 6.5.7: Display add Bicycle Section successful notification

// Create Bicycle Section screen

IF (Save = Successful)

FireSQL WRITE Bicycle Section added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Section was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Bicycle Section screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 6.6: Search Bicycle Section

Process 6.6.1: Display Bicycle Section Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Bicycle Section Management" navbar-dropdown-item.

// Bicycle Section Management Screen

Page.Load "Bicycle Section Management" screen.

Display

"Bicycle Section Management" Label

"Add Bicycle Section" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Section Management" Table

"Section Name" Table Header

FireSQL READ Bicycle SectionName from Section table

"Section Description Code" Table Header

FireSQL READ SectionDescription from Section table

"Action" Table Header

"Edit / Pencil" Button

Process 6.6.2: Capture Bicycle Section search criteria details

System captures inputted search criteria in the "Search" Text box

Process 6.6.3: Display searched Bicycle Section results

// Bicycle Section Management screen

Search Criteria

{

FireSQL READ SectionName from Section table

Compare SectionName with Inputted Section Name

IF (Bicycle Section exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Bicycle Section Management" screen.

Display

"Bicycle Section Management" Label

"Add Bicycle Section" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Section Management" Table

"Section Name" Table Header

FireSQL READ Bicycle SectionName from Section table

"Section Description Code" Table Header

FireSQL READ SectionDescription from Section table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Bicycle Section Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Bicycle Section not found. Please try again." Text

"Okay" Button

Process 6.6.4: Display searched Bicycle Section details screen

// Bicycle Section Management screen

Onclick "Edit / Pencil" button

// View Bicycle Section screen

Page.Load "View Bicycle Section" screen

Display

"Bicycle Section" Heading

"Choose Picture" Button

"Section Picture" Picture Image

"Section Name" Label

"Section Description" Label

"Section Name" Text box

FireSQL READ SectionName from Section table

"Section Description" Text box

FireSQL READ SectionDescription from Section table

"Update" Button

"Delete" Button

FireSQL WRITE Bicycle Section searched details to AuditTrail table

Use Case 6.7: Update Bicycle Section

Process 6.7.1: Display Bicycle Section Management screen

// Bicycle Section Management Screen

Page.Load "Bicycle Section Management" screen.

Display

"Bicycle Section Management" Label

"Add Bicycle Section" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Section Management" Table

"Section Name" Table Header

FireSQL READ Bicycle SectionName from Section table

"Section Description Code" Table Header

FireSQL READ SectionDescription from Section table

"Action" Table Header

"Edit / Pencil" Button

Process 6.7.6: Display searched Bicycle Section screen

// Bicycle Section Management screen

Onclick "Edit / Pencil" button

// View Bicycle Section screen

Page.Load "View Bicycle Section" screen

Display

"Bicycle Section" Heading

"Choose Picture" Button

"Section Picture" Picture Image

"Section Name" Label

"Section Description" Label

"Section Name" Text box

FireSQL READ SectionName from Section table

"Section Description" Text box

FireSQL READ SectionDescription from Section table

"Update" Button

"Delete" Button

INVOKE USE CASE 6.6: Search Bicycle Section

Process 6.7.3: Capture Bicycle Section details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Bicycle Section screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 6.7.4: Validate updated Bicycle Section details

// View Bicycle Section screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Bicycle Section?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Bicycle Section screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 6.7.5: Save updated Bicycle Section details

Fire SQL UPDATE the following Bicycle Section details in Bicycle Section table:

SectionID

SectionDescription

SectionName

Process 6.7.6: Display update successful notification

// View Bicycle Section screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Section was successfully updated" Text

"Okay" Button

WRITE Bicycle Section updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 6.8: Delete Bicycle Section

Process 6.8.1: Display Bicycle Section Management screen

// Bicycle Section Management Screen

Page.Load "Bicycle Section Management" screen.

Display

"Bicycle Section Management" Label

"Add Bicycle Section" Button

"Search" Text box

"Search" Button

Display and Populate

"Bicycle Section Management" Table

"Section Name" Table Header

FireSQL READ Bicycle SectionName from Section table

"Section Description Code" Table Header

FireSQL READ SectionDescription from Section table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 6.6: Search Bicycle Section

Process 6.8.2: Display searched Bicycle Section screen

// View Bicycle Section screen

Page.Load "View Bicycle Section" screen

Display

"Bicycle Section" Heading

"Choose Picture" Button

"Section Picture" Picture Image

"Section Name" Label

"Section Description" Label

"Section Name" Text box

FireSQL READ SectionName from Section table

"Section Description" Text box

FireSQL READ SectionDescription from Section table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 6.8.3: Delete searched Bicycle Section screen details

//View Bicycle Section screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Bicycle Section?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Bicycle Section table

SectionID

SectionName

SectionDescription

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 6.8.2

Process 6.8.4: Display Bicycle Section deleted successfully notification

// View Bicycle Section screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Bicycle Section was successfully added" Text

"Okay" Button

WRITE Bicycle Section deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Bicycle Section screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 7: Tracking

Use Case 7.1: Add Log Entry

Process 7.1.1 Capture tag details

The system will capture the following tag details:

TagID

DateTime

Process 7.1.2 Create and Save Log entry details

FireSQL INSERT the following Tag details in the Tag table

TagID

AntennaID

DateTime

7.1.2 [ALT] Tag read error termination

Terminate Use Case

Use Case 7.2: Search Log Entry

Process 7.2.1: Display Log Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Log Management" navbar-dropdown-item.

// Log Management Screen

Page.Load "Log Management" screen.

Display

"Log Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Log Management" Table

"Tag ID" Table Header

FireSQL READ TagID from Log table

"Bicycle ID" Table Header

FireSQL READ BicycleID from Log table

"Student ID" Table Header

FireSQL READ StudentID from Log table

"Time" Table Header

FireSQL READ DateTime from Log table

Process 7.2.2 Capture Log entry Search criteria details

The system captures the details entered by the Administrator

Process 7.2.4 Display Searched Log Entry Screen

// Log Management Screen

Page.Load "Log Management" screen.

Display

"Log Management" Label

"Search" Text box

"Search" Button

Display and Populate

"Log Management" Table

"Tag ID" Table Header

FireSQL READ TagID from Log table

"Bicycle ID" Table Header

FireSQL READ BicycleID from Log table

"Student ID" Table Header

FireSQL READ StudentID from Log table

"Time" Table Header

FireSQL READ DateTime from Log table

Alt Step

// Route Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Log entry not found. Please try again." Text

"Okay" Button

Use Case 7.3: Add Route

Process 7.3.1: Display Route Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Route Management" navbar-dropdown-item.

// Route Management Screen

Page.Load "Route Management" screen.

Display

"Route Management" Label

"Add Route" Button

"Search" Text box

"Search" Button

Display and Populate

"Route Management" Table

"Route ID" Table Header

FireSQL READ RouteID from Route table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Action" Table Header

"Edit / Pencil" Button

Process 7.3.2: Display add Route screen

// Route Management screen

Onclick "Add Route" button.

// Create Route screen

Page.Load "Create Route" screen.

Display

"Route" Heading

"School" Label

"Antenna" Label

"Antenna Position" Label

"Antenna ID" Label

"School" Combo box

FireSQL READ SchoolName from School table

"Antenna Position" Text box

"Antenna ID" Combo box

FireSQL READ AntennaID from Antenna table

"Create" Button

"Cancel" Button

Process 7.3.3: Capture Route details and check for empty fields

// Create Route screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Route screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 7.3.4: Validate Route details

//Create Route screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Route screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 7.3.5: Verify Route Details

//Create Route screen

Verification

{

FireSQL READ RouteID from Antenna table

Compare RouteID with Inputted Route Name

IF (Route exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Route?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Route screen

Go back to Process 7.3.2

Alt Step 2

// Create Route screen

Modal.Load "Error" modal

Display

"Error" Header

"The Route already exists. Please try again." Text

"Okay" Button

Process 7.3.6: Generate RouteID and save Route details

FireSQL READ previous RouteID from Route table

FireSQL WRITE the following Route details to the Route table

RouteID

SchoolID

FireSQL WRITE the following Route details to the RouteAntenna table

RouteAntennaID

RouteID

AntennaID

Position

Process 7.3.7: Display add Route successful notification

// Create Route screen

IF (Save = Successful)

FireSQL WRITE Route added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Route was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Route screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 7.4: Search Route

Process 7.4.1: Display Route Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display “System Management” navbar-dropdown-items.

Onclick “Route Management” navbar-dropdown-item.

// Route Management Screen

Page.Load “Route Management” screen.

Display

“Route Management” Label

“Add Route” Button

“Search” Text box

“Search” Button

Display and Populate

“Route Management” Table

“Route ID” Table Header

FireSQL READ RouteID from Route table

“School Name” Table Header

FireSQL READ SchoolName from School table

“Action” Table Header

“Edit / Pencil” Button

Process 7.4.2: Capture Route search criteria details

System captures inputted search criteria in the “Search” Text box

Process 7.4.3: Display searched Route results

// Route Management screen

Search Criteria

{

FireSQL READ RouteID from Route table

Compare RouteID with Inputted Route ID Name

IF (Route exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Route Management" screen.

Display

"Route Management" Label

"Add Route" Button

"Search" Text box

"Search" Button

Display and Populate

"Route Management" Table

"Route ID" Table Header

FireSQL READ RouteID from Route table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// Route Management screen

Modal.Load "Error" modal

Display

"Failed" Header

"Route not found. Please try again." Text

"Okay" Button

Process 7.4.4: Display searched Route details screen

// Route Management screen

Onclick "Edit / Pencil" button

// View Route screen

Page.Load "View Route" screen

Display

"Route" Heading

"School" Label

"Antenna" Label

"Antenna Position" Label

"Antenna ID" Label

"School" Combo box

FireSQL READ SchoolName from School table

"Antenna Position" Text box

FireSQl READ Position from AntennaRoute table

"Antenna ID" Combo box

FireSQL READ AntennaID from Antenna table

"Update" Button

"Delete" Button

FireSQL WRITE Route searched details to AuditTrail table

Use Case 7.5: Update Route

Process 7.5.1: Display Route Management screen

// Route Management Screen

Page.Load "Route Management" screen.

Display

"Route Management" Label

"Add Route" Button

"Search" Text box

"Search" Button

Display and Populate

"Route Management" Table

"Route ID" Table Header

FireSQL READ RouteID from Route table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Action" Table Header

"Edit / Pencil" Button

Process 7.5.2: Display searched Route screen

// Route Management screen

Onclick "Edit / Pencil" button

// View Route screen

Page.Load "View Route" screen

Display

"Route" Heading

"School" Label

"Antenna" Label

"Antenna Position" Label

"Antenna ID" Label

"School" Combo box

FireSQL READ SchoolName from School table

"Antenna Position" Text box

FireSQl READ Position from AntennaRoute table

"Antenna ID" Combo box

FireSQL READ AntennaID from Antenna table

"Update" Button

"Delete" Button

INVOKE USE CASE 7.4: Search Route

Process 7.5.3: Capture Route details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Route screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 7.5.4: Validate updated Route details

// View Route screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Route?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Route screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 7.5.5: Save updated Route details

FireSQL UPDATE the following Route details to the Route table

RouteID

SchoolID

FireSQL UPDATE the following Route details to the RouteAntenna table

RouteAntennaID

RouteID

AntennaID

Position

Process 7.5.6: Display update successful notification

// View Route screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Route was successfully updated" Text

"Okay" Button

WRITE Route updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 7.6: Delete Route

Process 7.6.1: Display Route Management screen

// Route Management Screen

Page.Load "Route Management" screen.

Display

"Route Management" Label

"Add Route" Button

"Search" Text box

"Search" Button

Display and Populate

"Route Management" Table

"Route ID" Table Header

FireSQL READ RouteID from Route table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 7.4: Search Route

Process 7.6.2: Display searched Route screen

// View Route screen

Page.Load "View Route" screen

Display

"Route" Heading

"School" Label

"Antenna" Label

"Antenna Position" Label

"Antenna ID" Label

"School" Combo box

FireSQL READ SchoolName from School table

"Antenna Position" Text box

FireSQl READ Position from AntennaRoute table

"Antenna ID" Combo box

FireSQL READ AntennaID from Antenna table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 7.6.3: Delete searched Route screen details

//View Route screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Route?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following Route details to the Route table

RouteID

SchoolID

FireSQL DELETE the following Route details to the RouteAntenna table

RouteAntennaID

RouteID

AntennaID

Position

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 7.6.2

Process 7.6.4: Display Route deleted successfully notification

// View Route screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Route was successfully added" Text

"Okay" Button

WRITE Route deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Route screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 8: Job Tasks

Use Case 8.1: Add Fault Task

Process 8.1.1: Display Fault Task Management screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Fault Task Management" navbar-dropdown-item.

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.1.2: Display add Fault Task screen

// Fault Task Management screen

Onclick "Add Fault Task" button.

// Create Fault Task screen

Page.Load "Create Fault Task" screen.

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

"Fault Task Description" Text box

"Create" Button

"Cancel" Button

Process 8.1.3: Capture Fault Task details and check for empty fields

// Create Fault Task screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Fault Task screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 8.1.4: Validate Fault Task details

//Create Fault Task screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Fault Task screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 8.1.5: Verify Fault Task details

//Create FaultTask screen

Verification

{

FireSQL READ FaultTask from FaultTask table

Compare FaultTask with Inputted Fault Task

IF (Fault Task exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Fault Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create Fault Task screen

Go back to Process 8.1.2

Alt Step 2

// Create Fault Task screen

Modal.Load "Error" modal

Display

"Error" Header

"The Fault Task already exists. Please try again." Text

"Okay" Button

Process 8.1.6: Generate FaultID and save FaultTask details

FireSQL READ previous FaultTaskID from Fault Task table

FireSQL WRITE the following Fault Task details to the FaultTask table

FaultTaskID

FaultTask

FaultTaskDescription

Process 8.1.7: Display add Fault Task successful notification

// Create Fault Task screen

IF (Save = Successful)

FireSQL WRITE Fault Task added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Fault Task screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 9.2: Search Fault Task

Process 8.2.1: Display Fault Task Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Fault Task Management" navbar-dropdown-item.

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.2.2: Capture Fault Task search criteria details

System captures inputted search criteria in the "Search" Text box

Process 8.2.3: Display searched Fault Task results

// Fault Task Management screen

Search Criteria

{

FireSQL READ FaultTask from FaultTask table

Compare FaultTask with Inputted Fault Task

IF (Fault Task exists)

    Return True

ELSE

    Return False

}

IF (search criteria = True)

    Page.Load "Fault Task Management" screen.

    Display

        "Fault Task Management" Label

        "Add Fault Task" Button

        "Search" Text box

        "Search" Button

    Display and Populate

        "Fault Task Management" Table

            "Fault Task" Table Header

                FireSQL READ FaultTask from FaultTask table

            "Fault Task Description" Table Header

                FireSQL READ FaultTaskDescription from FaultTask table

            "Action" Table Header

                "Edit / Pen" Button

ELSE IF

Proceed to Alt step

Alt Step

// Fault Task Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Fault Task not found. Please try again." Text

"Okay" Button

Process 8.2.4: Display searched Fault Task details screen

// Fault Task Management screen

Onclick "Edit / Pen" button

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

FireSQL WRITE Fault Task searched details to AuditTrail table

Use Case 8.3: Update Fault Task

Process 8.3.1: Display Fault Task Management screen

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.3.2: Display searched Fault Task screen

// Fault Task Management screen

Onclick "Edit / Pen" button

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

INVOKE USE CASE 8.2: Search Fault Task

Process 8.3.3: Capture Fault Task details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Fault Task screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 8.3.4: Validate updated Fault Task details

// View Fault Task screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Fault Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Fault Task screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 8.3.5: Save updated Fault Task details

Fire SQL UPDATE the following Fault Task details in FaultTask table:

FaultTask

FaultTaskDescription

Process 8.3.6: Display update successful notification

// View Fault Task screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully updated" Text

"Okay" Button

WRITE Fault Task updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 8.4: Delete Fault Task

Process 8.4.1: Display Fault Task Management screen

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

INVOKE USE CASE 8.2: Search Fault Task

Process 8.4.2: Display searched Fault Task screen

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 8.4.3: Delete searched Fault Task screen details

//View Fault Task screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Fault Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Fault Task table

FaultTaskID

FaultTask

FaultTaskDescription

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 8.4.2

Process 8.4.4: Display Fault Task deleted successfully notification

// View Fault Task screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully added" Text

"Okay" Button

WRITE Fault Task deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Fault Task screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Use Case 8.5: Add Maintenance Task

Process 8.5.1: Display Maintenance Task Management screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Maintenance Task Management" navbar-dropdown-item.

// Maintenance Task Management Screen

Page.Load "Maintenance Task Management" screen.

Display

"Maintenance Task Management" Label

"Add Maintenance Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Maintenance Task Management" Table

"Maintenance Task" Table Header

FireSQL READ MaintenanceTask from MaintenanceTask table

"Maintenance Task Description" Table Header

FireSQL READ MaintenanceTaskDescription from MaintenanceTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.5.2: Display add Maintenance Task screen

// Maintenance Task Management screen

Onclick "Add Maintenance Task" button.

// Create Maintenance Task screen

Page.Load "Create Maintenance Task" screen.

Display

"Maintenance Task" Label

"Maintenance Task Description" Label

"Maintenance Task" Text box

"Maintenance Task Description" Text box

"Create" Button

"Cancel" Button

Process 8.5.3: Capture Maintenance Task details and check for empty fields

// Create Maintenance Task screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create Maintenance Task screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 8.5.4: Validate Maintenance Task details

//Create Maintenance Task screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create Maintenance Task screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 8.5.5: Verify Maintenance Task details

//Create MaintenanceTask screen

Verification

{

FireSQL READ MaintenanceTaskDescription from MaintenanceTask table

Compare MaintenanceTaskDescription with Inputted Maintenance Task Description

IF (Maintenance Task Description exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the Maintenance Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

Go back to Process 8.5.2

Alt Step 2

// Create Maintenance Task screen

Modal.Load "Error" modal

Display

"Error" Header

"The Maintenance Task already exists. Please try again." Text

"Okay" Button

Process 8.5.6: Generate MaintenanceTaskID and save Maintenance Task details

FireSQL READ previous MaintenanceTaskID from MaintenanceTask table

FireSQL WRITE the following Maintenance Task details to the MaintenanceTask table

MaintenanceTaskID

FaultTask

FaultTaskDescription

Process 8.5.7: Display add Fault Task successful notification

// Create Fault Task screen

IF (Save = Successful)

FireSQL WRITE Fault Task added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create Fault Task screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 8.6: Search Fault Task

Process 8.6.1: Display Fault Task Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Fault Task Management" navbar-dropdown-item.

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.6.2: Capture Fault Task search criteria details

System captures inputted search criteria in the "Search" Text box

Process 8.6.3: Display searched Fault Task results

// Fault Task Management screen

Search Criteria

{

FireSQL READ FaultTask from FaultTask table

Compare FaultTask with Inputted Fault Task

IF (Fault Task exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

ELSE IF

Proceed to Alt step

Alt Step

// Fault Task Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Fault Task not found. Please try again." Text

"Okay" Button

Process 8.6.4: Display searched Fault Task details screen

// Fault Task Management screen

Onclick "Edit / Pen" button

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

FireSQL WRITE Fault Task searched details to AuditTrail table

Use Case 8.7: Update Fault Task

Process 8.7.1: Display Fault Task Management screen

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

Process 8.7.2: Display searched Fault Task screen

// Fault Task Management screen

Onclick "Edit / Pen" button

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

INVOKE USE CASE 8.2: Search Fault Task

Process 8.7.3: Capture Fault Task details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View Fault Task screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 8.7.4: Validate updated Fault Task details

// View Fault Task screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the Fault Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View Fault Task screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 8.7.5: Save updated Fault Task details

Fire SQL UPDATE the following Fault Task details in FaultTask table:

FaultTask

FaultTaskDescription

Process 8.7.6: Display update successful notification

// View Fault Task screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully updated" Text

"Okay" Button

WRITE Fault Task updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 8.8: Delete Fault Task

Process 8.8.1: Display Fault Task Management screen

// Fault Task Management Screen

Page.Load "Fault Task Management" screen.

Display

"Fault Task Management" Label

"Add Fault Task" Button

"Search" Text box

"Search" Button

Display and Populate

"Fault Task Management" Table

"Fault Task" Table Header

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Table Header

FireSQL READ FaultTaskDescription from FaultTask table

"Action" Table Header

"Edit / Pen" Button

INVOKE USE CASE 8.2: Search Fault Task

Process 8.8.2: Display searched Fault Task screen

// View Fault Task screen

Page.Load "View Fault Task" screen

Display

"Fault Task" Label

"Fault Task Description" Label

"Fault Task" Text box

FireSQL READ FaultTask from FaultTask table

"Fault Task Description" Text box

FireSQL READ FaultTaskDescription from FaultTask table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 8.8.3: Delete searched Fault Task screen details

//View Fault Task screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the Fault Task?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from Fault Task table

FaultTaskID

FaultTask

FaultTaskDescription

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 8.4.2

Process 8.8.4: Display Fault Task deleted successfully notification

// View Fault Task screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Fault Task was successfully added" Text

"Okay" Button

WRITE Fault Task deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View Fault Task screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 9: School

Use Case 9.1: Add School

Process 9.1.1: Display School Management screen

//Homepage #Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "School Management" navbar-dropdown-item.

// School Management Screen

Page.Load "School Management" screen.

Display

"School Management" Label

"Add School" Button

"Search" Text box

"Search" Button

Display and Populate

"School Management" Table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Village" Table Header

FireSQL READ VillageName from Village table

"Action" Table Header

"Edit / Pencil" Button

Process 9.1.2: Display add School screen

// School Management screen

Onclick "Add School" button.

// Create School screen

Page.Load "Create School" screen.

Display

"School" Heading

"School Name" Label

"School Email" Label

"School Telephone" Label

"Country" Label

"Province" Label

"City" Label

"Suburb" Label

"School" Text box

"School Name" Text box

"School Email" Text box

"School Telephone" Text box

"Country" Combo box

FireSQL READ CountryName from Country table

"Province" Combo box

FireSQL READ ProvinceNumber from Province table

"City" Combo box

FireSQL READ CityName from City table

"Suburb" Combo box

FireSQL READ SuburbName from Suburb table

"Create" Button

"Cancel" Button

Process 9.1.3: Capture School details and check for empty fields

// Create School screen

IF (onclick = "Create")

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

END IF (onclick = "Cancel")

Alt Step

// Create School screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field." Text

"X" Button

Process 9.1.4: Validate School details

//Create School screen

IF (Validation = False)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// Create School screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 9.1.5: Verify School Details

//Create School screen

Verification

{

FireSQL READ SchoolName from School table

Compare SchoolNames with Inputted School Name

IF (School exists)

Return False

EISE

Return True

}

IF (Verification = True)

Modal.Load "Successful" modal

Display

"Confirmation" Header

"Are you sure you want to add the School?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

IF onclick = "No"

Proceed to Alt step 1

IF (verification = False)

Proceed to Alt Step 2

Alt Step 1

// Create School screen

Go back to Process 9.1.2

Alt Step 2

// Create School screen

Modal.Load "Error" modal

Display

"Error" Header

"The School already exists. Please try again." Text

"Okay" Button

Process 9.1.6: Generate SchoolID and save School details

FireSQL READ previous SchoolID from School table

FireSQL WRITE the following School details to the School table

SchoolID

SchoolName

SchoolEmail

SchoolTelephone

Process 9.1.7: Display add School successful notification

// Create School screen

IF (Save = Successful)

FireSQL WRITE School added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The School was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step

Alt Step

// Create School screen

Modal.Load "Failed" modal

Display

"Successful" Header

"Create unsuccessful. Please try again" Text

"Okay" Button

Use Case 9.2: Search School

Process 9.2.1: Display School Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "School Management" navbar-dropdown-item.

// School Management Screen

Page.Load "School Management" screen.

Display

"School Management" Label

"Add School" Button

"Search" Text box

"Search" Button

Display and Populate

"School Management" Table

"School Name" Table Header

FireSQL READ SchoolName from School table

"Village" Table Header

FireSQL READ VillageName from Village table

"Action" Table Header

"Edit / Pencil" Button

Process 9.2.2: Capture School search criteria details

System captures inputted search criteria in the "Search" Text box

Process 9.2.3: Display searched School results

// School Management screen

Search Criteria

{

FireSQL READ SchoolName from School table

Compare SchoolNames with Inputted School Name

IF (School exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

Page.Load "School Management" screen.

Display

"School Management" Label

"Add School" Button

"Search" Text box

"Search" Button

Display and Populate

"School Management" Table

"School Name" Table Header

READ SchoolName from School table

"Village" Table Header

READ VillageName from Village table

"Action" Table Header

"Edit / Pencil" Button

ELSE IF

Proceed to Alt step

Alt Step

// School Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"School not found. Please try again." Text

"Okay" Button

Process 9.2.4: Display searched School details screen

// School Management screen

Onclick "Edit / Pencil" button

// View School screen

Page.Load "View School" screen

Display

"School" Label

"School Name" Label

"School Email" Label

"School Telephone" Label

"Country" Label

"Province" Label

"City" Label

"Suburb" Label

"School" Text box

"School Name" Text box

READ SchoolName from School table

"School Email" Text box

READ SchoolEmail from School table

"School Telephone" Text box

READ SchoolTelephone from School table

"Country" Combo box

READ CountryName from Country table

"Province" Combo box

READ ProvinceNumber from Province table

"City" Combo box

READ CityName from City table

"Suburb" Combo box

READ SuburbName from Suburb table

"Update" Button

"Delete" Button

FireSQL WRITE School searched details to AuditTrail table

Use Case 9.3: Update School

Process 9.3.1: Display School Management screen

// School Management Screen

Page.Load "School Management" screen.

Display

"School Management" Label

"Add School" Button

"Search" Text box

"Search" Button

Display and Populate

"School Management" Table

"School Name" Table Header

READ SchoolName from School table

"Village" Table Header

READ VillageName from Village table

"Action" Table Header

"Edit / Pencil" Button

Process 9.3.2: Display searched School screen

// School Management screen

Onclick "Edit / Pencil" button

// View School screen

Page.Load "View School" screen

Display

"School" Label

"School Name" Label

"School Email" Label

"School Telephone" Label

"Country" Label

"Province" Label

"City" Label

"Suburb" Label

"School" Text box

"School Name" Text box

READ SchoolName from School table

"School Email" Text box

READ SchoolEmail from School table

"School Telephone" Text box

READ SchoolTelephone from School table

"Country" Combo box

READ CountryName from Country table

"Province" Combo box

READ ProvinceNumber from Province table

"City" Combo box

READ CityName from City table

"Suburb" Combo box

READ SuburbName from Suburb table

"Update" Button

"Delete" Button

INVOKE USE CASE 9.2: Search School

Process 9.3.3: Capture School details and check for empty fields

Onclick "Update" button

IF (Empty fields = True)

Proceed to Alt Step

ELSE

Proceed to next Process

Alt Step

// View School screen

Alert.Show "Input Error"

Display below input field of error

"Please fill this field" Text

"X" Button

Process 9.3.4: Validate updated School details

// View School screen

IF (validation = true)

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to update the School?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes")

Proceed to next Process

END IF (onclick = "No")

ELSE

Proceed to Alt step

Alt Step

// View School screen

Alert.Show "Validation Error"

Display below input field of error

"Invalid Input." Text

"X" Button

Process 9.3.5: Save updated School details

Fire SQL UPDATE the following School details in School table:

SchoolName

SchoolID

SchoolEmail

SchoolTelephone

Process 9.3.6: Display update successful notification

// View School screen

IF (Update = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"The School was successfully updated" Text

"Okay" Button

WRITE School updated details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful. Please try again." Text

"Okay" Button

Use Case 9.4: Delete School

Process 9.4.1: Display School Management screen

// School Management Screen

Page.Load "School Management" screen.

Display

"School Management" Label

"Add School" Button

"Search" Text box

"Search" Button

Display and Populate

"School Management" Table

"School Name" Table Header

FIRESQL READ SchoolName from School table

"Village" Table Header

FIRESQL READ VillageName from Village table

"Action" Table Header

"Edit / Pencil" Button

INVOKE USE CASE 9.2: Search School

Process 9.4.2: Display searched School screen

// View School screen

Page.Load "View School" screen

Display

"School" Label

"School Name" Label

"School Email" Label

"School Telephone" Label

"Country" Label

"Province" Label

"City" Label

"Suburb" Label

"School" Text box

"School Name" Text box

FIRESQL READ SchoolName from School table

"School Email" Text box

FIRESQL READ SchoolEmail from School table

"School Telephone" Text box

FIRESQL READ SchoolTelephone from School table

"Country" Combo box

FIRESQL READ CountryName from Country table

"Province" Combo box

FIRESQL READ ProvinceNumber from Province table

"City" Combo box

FIRESQL READ CityName from City table

"Suburb" Combo box

FIRESQL READ SuburbName from Suburb table

"Update" Button

"Delete" Button

Onclick = "Delete"

Process 9.4.3: Delete searched school screen details

//View School screen

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to delete the School?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQL DELETE the following details from School table

SchoolID

SchoolName

SchoolEmail

SchoolTelephone

Proceed to next Process

ELSE IF (onclick = "No")

Proceed to Alt step

Alt Step

Go to Process 9.4.2

Process 9.4.4: Display school deleted successfully notification

// View School screen

IF (Delete = Successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The School was successfully added" Text

"Okay" Button

WRITE School deleted details to AuditTrail table

ELSE

Proceed to Alt step

Alt Step

//View School screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Update unsuccessful please try again" Text

"Okay" Button

Sub-system 10: Job

Use Case 10.1: Report Repair Job

Process 10.1.1 Display Student QR Scan Screen

//Homepage

#Bikes4ERP_VVS

Page.Load "Homepage" screen.

Onclick "Report Fault" Button

// Student QR Scan Screen

Page.Load "Student QR Scan Screen" screen.

Display

"Please scan your bicycle QR code to continue" Heading

"qr-scan.gif" Image

Onscan of "QR code"

Proceed to next process

Process 10.1.2 Generate RepairJobID and Save Repair Job details

FireSQL READ previous RepairJobID in RepairJob table

FireSQL INSERT the following RepairJob details in the ReapirJob table

RepairJobID

BicycleID

StudentID

RepairJobStatus

DateReported

Assigned

DateScheduled

Process 10.1.3 Display Full Bicycle Blueprint Screen

// Bicycle Blueprint screen

Page.Load "Bicycle Blueprint" screen

Display

"Please select the section of the bicycle where your fault appears" Heading

"Bicycle Blueprint" Picture Image

Bicycle Blueprint options populated using FireSQL READ SectionNames from Section table

"Unknown" Button

IF (Onclick = "Bicycle Blueprint Section")

Proceed to Process 10.1.4

ELSE IF (Onclick = "Unknown"

Proceed to ALT Process 10.1.4

Process 10.1.4 [ALT] Display Unknown Fault Screen

Modal.Load "Unknown Fault" modal

Display

"Please enter any additional information." Header

"unknownDetails" Text Field

"Submit" Button

Onclick "Submit"

Proceed to Process 10.1.6

Process 10.1.4 Display Section Blueprint Screen

// Section Blueprint screen

Page.Load "Section Blueprint" screen

Display

"Please select the part from "INSERT SectionName selected in Process 10.1.3" where your fault appears" Heading

"Part Picture" Picture Image

"Part Name" Label

"Unknown" Button

The Section Blueprint options are populated using FireSQL READ query PartName from the Part table

IF (Onclick = "Part Picture")

Proceed to next Process

ELSE IF (Onclick = "Unknown")

Proceed to ALT Process 10.1.5

Process 10.1.5 [ALT] Display Unknown Fault Screen

Modal.Load "Unknown Fault" modal

Display

"Please enter any additional information." Header

"unknownDetails" Text Field

"Submit" Button

Onclick "Submit"

Proceed to Process 10.1.6

Process 10.1.5 Display Part Faults Screen

// Section Blueprint screen

Page.Load "Part Faults" screen

Display

"Please select the fault that appears on the "INSERT PartName selected in Process 10.1.4" where your fault appears" Heading

"Fault Picture" Picture Image

"Fault Name" Label

"Unknown" Button

The Part Fault options are populated using FireSQL READ query FaultType from the FaultType table

IF (Onclick = "Part Picture")

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to select this fault?" Text

"Yes" Button

"No" Button

IF (Onclick = "Yes")

Proceed to next Process

ELSE IF (Onclick = "No")

Proceed to Process 10.1.3

ELSE IF (Onclick = "Unknown")

Proceed to ALT Process 10.1.5

Process 10.1.6 [ALT] Display Unknown Fault Screen

Modal.Load "Unknown Fault" modal

Display

"Please enter any additional information." Header

"unknownDetails" Text Field

"Submit" Button

Onclick "Submit"

Proceed to Process 10.1.6

Process 10.1.6 Generate ReportLineID and Save ReportLine details

FireSQL READ previous SchoolID from School table

FireSQL WRITE the following School details to the School table

SchoolID

SchoolName

SchoolEmail

SchoolTelephone

Process 10.1.7 Display Fault List Screen details

// Section Blueprint screen

IF (Save = Successful)

FireSQL WRITE School added details to AuditTrail table

Modal.Load "Successful" modal

Display

"Successful" Header

"The School was successfully added" Text

"Okay" Button

ELSE

Proceed to Alt step 1

OnModalDismiss

// Report List Screen

Page.Load "Report List" screen

Display

"Report Job – Fault List" Heading

"Add new Fault" Button

"Report Job" Button

Display and Populate

"Job Faults" Table

"Section" Table Header

FireSQL READ SectionName from Section table

"Part" Table Header

FireSQL READ PartName from Part table

"Fault Type" Table Header

FireSQL READ FaultType from FaultType table

"Action" Table Header

"Delete Fault" Icon Link

IF (onclick = "Report Job")

Proceed to next Process

ELSE IF (onclick = "Add new Fault")

Proceed to Alt Step 2

Alt Step 1

Modal.Load "Failed" modal

Display

"Failed" Header

"Fault Reported unsuccessful. Please try again." Text

"Okay" Button

Alt Step 2

Proceed to process 10.1.3

Process 10.1.8 Assign Job and Create Check-In entry

Fire SQL INSERT the following Check-in details in Check-in table

Check-InID

Check-InDate

Check-InStatusID

JobTypeID

MechanicSchoolID

Process 10.1.9 Display Fault Reported Successfully notification

// Report List Screen

IF (save = successful)

Modal.Load "Successful" modal

Display

"Successful" Header

"The Faults have been reported successfully" Text

"Okay" Button

FireSQl WRITE Report Fault added details to AuditTrail table

ELSE IF

Proceed to Alt Step

Alt Step

// Report List Screen

Modal.Load "Failed" modal

Display

"Failed" Header

"Job reported unsuccessful. Please try again." Text

"Okay" Button

Use Case 10.2: Complete Job

Process 10.2.1 Display Fix Fault Screen

Process 10.2.2 Save and Update ReportFaultTask and ReportLine details

Process 10.2.3 Check if all Faults are completed

Process 10.2.4 Display Job Completed successfully notification

Use Case 10.3: Check-in Bicycle

Process 10.3.1 Receive Student Bicycle

Process 10.3.2 Display QR Scan Screen

Process 10.3.3 Capture and Retrieve Check-In details

Process 10.3.4 Update and Save Check-In details

Process 10.3.5 Display Check-in Successful Notification

Use Case 10.4: Scheduled Maintenance

Process 10.4.1 Generate Next Maintenance date details

Process 10.4.2 Generate MaintenanceListID and Save MaintenanceList details

Process 10.4.3 Update Past Maintenance List Status

Process 10.4.4 Assign Job to Mechanic

Sub-system 11: Reporting
Sub-system 12: Notification
Sub-system 13: Administration

Use Case 13.1: Backup Data

Process 13.1.1: Display Backup and Restore screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "System Data Management" navbar-dropdown-item.

// System Backup and Restore Screen

Page.Load "System Backup and Restore" screen.

Display

"Backup System" Button

"Search" Text box

"Search" Button

Display and Populate

"Restore Management" Table

"Backup ID" Table Header

FireSQL READ BackupID from Backup (External backup database) table

"DateTime" Table Header

FireSQL READ DateTime from Backup (External backup database) table

"Action" Table Header

"Restore" Button

Process 13.1.2 Perform system backup

// System Backup and Restore screen

Onclick "Backup System" Button

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to Backup the system?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQl INSERT Backup Data into the External Backup database

BackupID

DateTime

Proceed to next Process

ELSE IF (onclick = "No)

Proceed to Alt Step

Alt Step

Proceed to Process 13.1.1

Process 13.1.3 Display Backup successful notification

// System Backup and Restore screen

IF (Backup = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"System has been backed up successfully." Text

"Okay" Button

WRITE System backed up details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Backup unsuccessful please try again" Text

"Okay" Button

Use Case 13.2: Restore System

Process 13.2.1 Display System Backup and Restore Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "System Data Management" navbar-dropdown-item.

// System Backup and Restore Screen

Page.Load "System Backup and Restore" screen.

Display

"Backup System" Button

"Search" Text box

"Search" Button

Display and Populate

"Restore Management" Table

"Backup ID" Table Header

FireSQL READ BackupID from Backup (External backup database) table

"DateTime" Table Header

FireSQL READ DateTime from Backup (External backup database) table

"Action" Table Header

"Restore" Button "Edit / Pencil" Button

Process 13.2.2 Perform System Restore

// System Backup and Restore screen

Onclick "Restore System" Button

Modal.Load "Confirmation" modal

Display

"Confirmation" Header

"Are you sure you want to Restore the system?" Text

"Yes" Button

"No" Button

IF (onclick = "Yes)

FireSQl RESTORE Backup Data from the External Backup database

BackupID

DateTime

Proceed to next Process

ELSE IF (onclick = "No)

Proceed to Alt Step

Alt Step

Proceed to Process 13.2.1

Process 13.2.3 Display Restore Successful notification

// System Backup and Restore screen

IF (Restore = Success)

Modal.Load "Successful" modal

Display

"Successful" Header

"System has been restored successfully." Text

"Okay" Button

WRITE System restored details to AuditTrail table

ELSE

Proceed to Alt Step

Alt Step

Modal.Load "Failed" modal

Display

"Failed" Header

"Restore unsuccessful please try again" Text

"Okay" Button

Use Case 13.3: View Audit Trail Log

Process 13.3.1 Display Audit Trail Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Audit Trail Management" navbar-dropdown-item.

// Audit Trail Management Screen

Page.Load "Audit Trail Management" screen.

Display

"View Audit Trail Log" Button

"Search" Text box

"Search" Button

Display and Populate

"Audit Trail Management" Table

"Audit Trail ID" Table Header

FireSQL READ AuditTrailID from AuditTrail table

"DateTime" Table Header

FireSQL READ DateTime from AuditTrail table

"Transaction Type" Table Header

FireSQL READ TransactionType from AuditTrail table

"Action" Table Header

"View" Button

Onclick "View Audit Trail Log"

Proceed to next Process

Process 13.3.2 Display Audit trail Log Screen

// Audit Trail Log screen

Page.Load "Audit Trail Log" screen

"Audit Trail Log" Table

"Audit Trail ID" Table Header

FireSQL READ AuditTrailID from AuditTrail table

"DateTime" Table Header

FireSQL READ DateTime from AuditTrail table

"Transaction Type" Table Header

FireSQL READ TransactionType from AuditTrail table

Use Case 13.4: Search Audit Trail Entry

Process 13.4.1 Display Audit Trail Management Screen

//Homepage

Page.Load "Homepage" screen.

Onclick "System Management" navbar-dropdown.

Display "System Management" navbar-dropdown-items.

Onclick "Audit Trail Management" navbar-dropdown-item.

// Audit Trail Management Screen

Page.Load "Audit Trail Management" screen.

Display

"View Audit Trail Log" Button

"Search" Text box

"Search" Button

Display and Populate

"Audit Trail Management" Table

"Audit Trail ID" Table Header

FireSQL READ AuditTrailID from AuditTrail table

"DateTime" Table Header

FireSQL READ DateTime from AuditTrail table

"Transaction Type" Table Header

FireSQL READ TransactionType from AuditTrail table

"Action" Table Header

"View" Button

Process 13.4.2 Capture Audit Trail Search criteria details

// Audit Trail Management screen

Onlclick "Search" the system captures the search details in the search text box

Process 13.4.3 Display Audit Trail Results

Search Criteria

{

FireSQL READ TransactionType from AuditTrail table

Compare TransactionType with Inputted Transaction Type

IF (Audit Trail entry exists)

Return True

ELSE

Return False

}

IF (search criteria = True)

// Audit Trail Management Screen

Page.Load "Audit Trail Management" screen.

Display

"View Audit Trail Log" Button

"Search" Text box

"Search" Button

Display and Populate

"Audit Trail Management" Table

"Audit Trail ID" Table Header

FireSQL READ AuditTrailID from AuditTrail table

"DateTime" Table Header

FireSQL READ DateTime from AuditTrail table

"Transaction Type" Table Header

FireSQL READ TransactionType from AuditTrail table

"Action" Table Header

"View" Button

ELSE IF

Proceed to Alt step

Alt Step

// Audit Trail Management screen

Modal.Load "Error" modal

Display

"Successful" Header

"Audit Trail entry not found. Please try again." Text

"Okay" Button

Process 13.4.4 Display Audit Trail Screen

// Audit Trail Management screen

Onlclick "View" button

// View Audit Trail Entry screen

Page.Load "View Audit Trail Entry" screen

Display

"Audit Trail Entry" Heading

"Audit Trail ID" Label

"User" Label

"Date Time" Label

"Transaction Type" Label

"Audit Trail Entry" Heading

"Audit Trail ID" Textbox read-only

"User" Textbox read-only

"Date Time" Textbox read-only

"Transaction Type" Textbox read-only

"Back to search results" Button