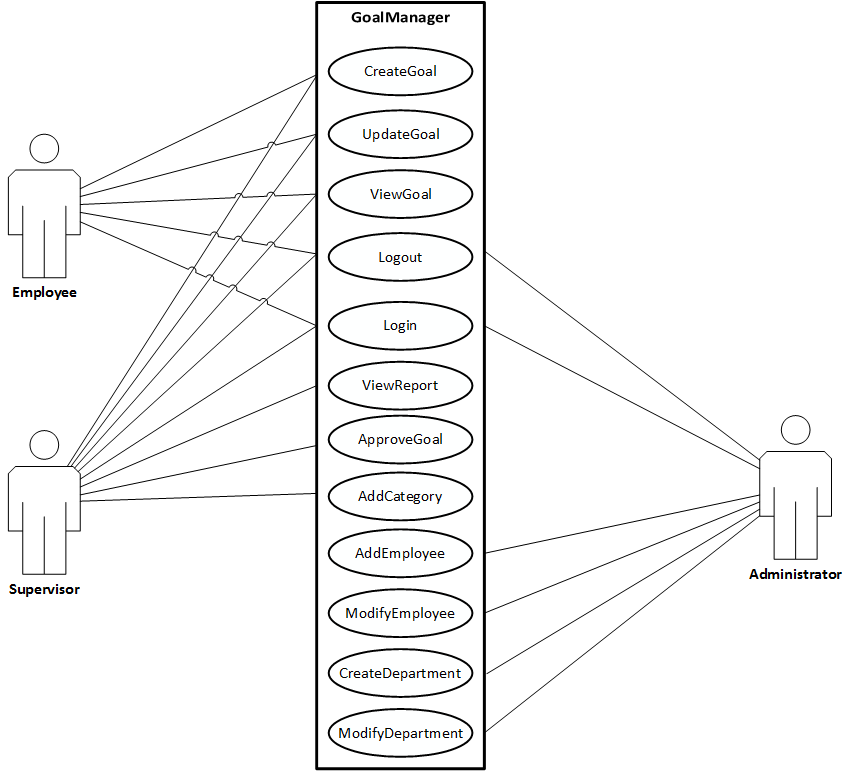
**Use Case Diagram**



Employee

|  |  |
| --- | --- |
| *Use Case name* | Login |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User navigates to the GoalManager web application and activates the “Login” button on the *HomeView*.      1. **GoalManager displays the *LoginView* with empty forms Username and Password.** 2. The User populates the forms by entering valid username and password credentials into the *LoginView* forms and presses “Submit”. 3. **GoalManager queries the database with provided credentials and validates the User. GoalManager then displays the appropriate default View for the actor *(EmployeeView, SuperView, or AdminView).*** 4. The User is logged into GoalManager on his or her default View. |
| *Entry condition* | User navigates to *HomeView* of GoalManager through a web browser. |
| *Exit condition* | User is authenticated and is successfully logged and directed to the appropriate default View (*EmployeeView, SuperView, or AdminView*). |
| *Quality Requirements* | Fields for credentials (username and passwords) are salted, hashed, and hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | Login (Failure) |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User navigates to the GoalManager web application and activates the “Login” button on the *HomeView*.      1. **GoalManager displays the *LoginView* with empty forms “Username” and “Password”.** 2. The User populates the forms by entering invalid username and password credentials into the *LoginView* forms and presses “Submit”. 3. **GoalManager queries the database with provided credentials and attempts to validate the User. However, the provided credentials are invalid. GoalManager then returns the User to *LoginView* with a login error message.** 4. The User is not logged into GoalManager and remains on LoginView. |
| *Entry condition* | User navigates to *HomeView* of GoalManager through a web browser. |
| *Exit condition* | User is returned to the Login View with an error. |
| *Quality Requirements* | Fields for credentials (username and passwords) are salted, hashed, and hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ViewGoal |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is logged in and on *EmployeeView* with a list of his or her Goals. The Employee then clicks the “View” button attached to an individual Goal list element.      1. **GoalManager queries the database for the properties of the selected Goal, including its Title, Timeframe, Category, and Progress. GoalManager then populates the partial *ViewGoalView* with the selected data.** 2. The Employee is able to view his or her Goal information, sort by Category, Timeframe, Title, and Progress. The Employee clicks the “Done” button when finished. 3. **GoalManager returns the Employee to *EmployeeView*.** |
| *Entry condition* | Employee is logged in to GoalManager and viewing *EmployeeView*. |
| *Exit condition* | Employee is prompted to the *EmployeeView.* |
| *Quality Requirements* |  |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is logged in and on *EmployeeView* with a list of his or her Goals. The Employee then clicks the “Create Goal” button. 2. **GoalManager returns *CreateGoalView* with empty form fields for Title, Timeframe, and Category.** 3. The Employee enters valid data representing the desired Title, Timeframe, and Category for the new Goal and clicks the “Submit” button. 4. **GoalManager successfully validates form data and writes the new Goal into the database’s pending approval queue. GoalManager returns the Employee back to *EmployeeView.*** |
| *Entry condition* | Employee is logged in to GoalManager and viewing *EmployeeView*. |
| *Exit condition* | Employee is prompted to the *EmployeeView* and the Goal is added to the database |
| *Quality Requirements* | Fields (Title, Timeframe, Category) are scrubbed for SQL injection prevention and passed through form validation before being written to database. |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is logged in and on *EmployeeView* with a list of his or her Goals. The Employee then clicks the “Create Goal” button. 2. **GoalManager returns *CreateGoalView* with empty form fields for Title, Timeframe, and Category.** 3. The Employee enters invalid data representing the desired Title, Timeframe, and Category for the new Goal and clicks the “Submit” button. 4. **GoalManager attempts to validate form data, but one or more fields filled with invalid data fail validation. GoalManager returns the Employee back to *CreateGoalView* with an error.** |
| *Entry condition* | Employee is logged in to GoalManager and viewing *EmployeeView*. |
| *Exit condition* | Employee is prompted to the *CreateGoalView* with errors or is prompted to the *EmployeeView* if the “Cancel” button is pressed. |
| *Quality Requirements* | Fields (Title, Timeframe, Category) are scrubbed for SQL injection prevention and passed through form validation before attempting to write to database. |

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is logged in and on *EmployeeView* with a list of his or her Goals. The Employee then clicks the “Update Goal” button attached to a Goal list element. 2. **GoalManager queries the database for the selected Goal’s properties, returning *UpdateGoalView* populated with the selected Goal’s fields in addition to empty forms for Progress and Notes.** 3. The Employee enters valid data representing the desired Progress and Notes information for the Goal to be updated and clicks the “Submit” button. 4. **GoalManager successfully validates form data and writes the updated Goal progress to the database. GoalManager returns the Employee back to *EmployeeView.*** |
| *Entry condition* | Employee is logged in to GoalManager and viewing *EmployeeView*. |
| *Exit condition* | Employee is prompted to the *EmployeeView* and the Goal is properly Updated. |
| *Quality Requirements* | Fields (Progress and Notes) are scrubbed for SQL injection prevention and passed through form validation before being written to database. |

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is logged in and on *EmployeeView* with a list of his or her Goals. The Employee then clicks the “Update Goal” button attached to a Goal list element. 2. **GoalManager queries the database for the selected Goal’s properties, returning *UpdateGoalView* populated with the selected Goal’s fields in addition to empty forms for Progress and Notes.** 3. The Employee enters invalid data representing the desired Progress and Notes information for the Goal to be updated and clicks the “Submit” button. 4. **GoalManager attempts to validate form data fails due to invalid data. GoalManager returns the Employee back to *UpdateGoalView* with an error*.*** |
| *Entry condition* | Employee is logged in to GoalManager and viewing *EmployeeView*. |
| *Exit condition* | Employee is prompted to the *UpdateGoalView* with errors or prompted to the *EmployeeView* if the “Cancel” button. |
| *Quality Requirements* | Fields (Progress and Notes) are scrubbed for SQL injection prevention and passed through form validation before being written to database. |

|  |  |
| --- | --- |
| *Use Case name* | Logout |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User clicks the “Logout” button provided on each default View *(EmployeeView, SuperView, AdminView)*. 2. **GoalManager terminates the User’s session and returns *LogoutView.*** |
| *Entry condition* | Employee is logged in to GoalManager and viewing a default View *(EmployeeView, SuperView, AdminView)*. |
| *Exit condition* | Employee is logged out. |
| *Quality Requirements* | Logout is graceful and terminates the session in compliance with OWASP best practices. |

Supervisor

|  |  |
| --- | --- |
| *Use Case name* | ViewGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SuperView* with a list of their Goals and department Goals and then selects the “View” button on that list item.      1. **GoalManager queries the database for the properties of the selected Goal, including its Title, Timeframe, Category, and Progress. GoalManager then populates the partial *ViewGoalView* with the selected data.** 2. The Supervisor is prompted with the selected Goal’s information allowing them to sort by Category, Timeframe, Progress, and Title. When the Supervisor is finished they will select the “Done” button. 3. **GoalManager closes ViewGoalView and returns the** **Supervisor** **to the *SupervisorView*.** |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor is returned to *SupervisorView*. |
| *Quality Requirements* |  |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView* with a list of their Goals and department Goals and selects the “Create Goal” button.      1. **GoalManager opens a *CreateGoalView* with its empty form fields of Title, Timeframe, Description, Scope, and Category.** 2. The Supervisor enters valid information on the *CreateGoalView* and selects the “Submit” button. 3. **GoalManager successfully validates *CreateGoalView* data and adds the new Goal into the database’s pending approval queue. GoalManager returns Supervisor back to the *SupervisorView*.** |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor is prompted to the *SupervisorView* and the Goal is added to the database. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView* with a list of their Goals and department Goals and selects the “Create Goal” button.      1. **GoalManager presents the *CreateGoalView* with its empty form fields of Title, Timeframe, Description, Scope, and Category.** 2. The Supervisor enters invalid information in the *CreateGoalView* and selects the “Submit” button. 3. **GoalManager attempts to validate form data, but data fails validation. GoalManager returns Supervisor back to the *CreateGoalView* with an error specific to the invalid field.** |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor is prompted back to CreateGoalView with errors or is prompted to *SupervisorView* if the “Cancel” button is pressed. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView* with a list of their Goals and department Goals and selects the “Update Goal” button on that list item.      1. **GoalManager queries the database for the selected Goal’s fields and presents the *UpdateGoalView* with an empty form field for Progress.** 2. The Supervisor enters valid information into the Progress form on the *UpdateGoalView* and selects the “Submit” button. The Supervisor also has the option to Edit the body of a newly-created Goal for typing errors. 3. **GoalManager successfully validates form data and updates the Goal’s progress to the database. GoalManager returns Supervisor back to the *SuperView*.** |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor is prompted to the *SupervisorView* and the Goal is properly updated in the databse. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView* with a list of their Goals and department Goals and then clicks the “Update Goal” button on that list item.      1. **GoalManager queries the database for the selected Goal’s fields and presents the *UpdateGoalView* with an empty form field for Progress.** 2. The Supervisor enters invalid information into the Progress form on the *UpdateGoalView* and selects the “Submit” button. 3. **GoalManager attempts to validate invalid form data. GoalManager returns Supervisor back to the *UpdateGoalView* with an error.** |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor is prompted to the *UpdateGoalView* with errors or to the *SupervisorView* if the “Cancel” button. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | ViewReport |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SuperView* with a list of their Goals and department Goals and then clicks the “View Report” button on that list item.      1. **GoalManager queries the database and presents the *ViewReportView*. *ViewReportView* populates a list of Employees under the Supervisor’s department and a list of departments the Supervisor is in charge of, along with a Search form field for filtering data.** 2. The Supervisor selects an Employee or department by selecting the “View” button next to their name. Supervisor also has the option to fill the search field and select the “Submit” button to filter through data. 3. **GoalManager validates the Supervisor’s request and displays the Report form with fields Employee/Department name, Department, Goal Progress, Goals Completed, and Goals Failed.** 4. The Supervisor is now able to view the Employee or Department Report and selects the “Done” button when finished. 5. **GoalManager closes *ViewReportView* and returns Supervisor back to *SuperView.*** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is returned to *SuperView*. |
| *Quality Requirements* | Search field is hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | ApproveGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SuperView* which displays their Goals and department Goals and then selects the “Approve Goal” button on the Goal that needs to be approved.      1. **GoalManager updates the Goal status to “Approved” in the database.** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is prompted to the *SupervisorView* and the Goal is approved. |
| *Quality Requirements* |  |

|  |  |
| --- | --- |
| *Use Case name* | AddCategory (Success) |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisoris on the *SuperView* which displays their Goals and department Goals and then selects the “Add Category” button on the Goal that needs to be approved.      1. **GoalManager presents an *AddCategoryView* which displays a form for entering a Category Name.** 2. The Supervisor fills out the form with the correct information in the field and selects the “Confirm” button. 3. **GoalManager validates the Supervisor’s form and updates the new Category to the Goal database. GoalManager closes *AddCategoryView* and returns the Supervisor back to the *SuperView*.** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is prompted to the *SupervisorView*  and the Category is added to the Database. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | AddCategory (Failure) |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SuperView* which displays their Goals and department Goals and then selects the “Add Category” button on the Goal that needs to be approved. 2. **GoalManager presents *AddCategoryView* which displays a form for entering a Category Name.** 3. The Supervisor fills out the form with the incorrect information and selects the “Confirm” button. 4. **GoalManager attempts to validate the Supervisor’s form but the information is invalid and notifies Supervisor that it is entered incorrectly.** 5. The Supervisor is prompted back to the *AddCategoryView* to re-edit the incorrect information. |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is prompted to the *AddCategoryView*  with errors or to the *SupervisorView*  if “Cancel” was pressed. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Administrator

|  |  |
| --- | --- |
| *Use Case name* | AddEmployee |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is logged in and is on the *AdministratorView* and selects “Add Employee” button.      1. **GoalManager returns the *AddEmployeeView* with empty form fields for First Name, Last Name, Role, DepartmentId, Title.** 2. The Administrator fills out credentials related to form fields First Name, Last Name, DepartmentID, Title, Role. The Administrator presses the Submit Button. 3. **GoalManager successfully validates form data and writes the new Employee into the database. GoalManager returns the Administrator back to *AdministratorView*** |
| *Entry condition* | Administrator is logged in and is viewing the *AdminstratorView* |
| *Exit condition* | Administrator is prompted back to the *AdminstratorView* and Employee is added to the database. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | AddEmployee (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is logged in and is on the *AdministratorView* and selects “Add Employee” button.      1. **GoalManager returns the *AddEmployeeView* with empty form fields for First Name, Last Name, Role, DepartmentId, Title.** 2. The Administrator fills out credentials related to form fields First Name, Last Name, DepartmentID, Title, Role. The Administrator presses the Submit Button or presses the “Cancel” button. 3. **GoalManager unsuccesfully validates form data and prompts back to the *AddEmployeeView* with errors or if the Administrator pressed the “Cancel” button GoalManager returns the Administrator back to *AdministratorView,* resepectively.** |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator enters incorrect information and is directed to the *AddEmployeeView* to attempt to create an Employee again or the *AdminstratorView* if “Cancel” button was pressed. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ModifyEmployee |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministorView* where he/she is able to search for an Employee. The Administrator performs a search using the search bar.      1. **GoalManager queries a list of Employee’s related to the search and populates the list of Employee’s to select from.** 2. The Administrator clicks an Employee to modify. 3. **GoalManager returns the *ModifyEmployeeView* with editable form fields for First Name, Last Name, Role, DepartmentId, Title.** 4. Administrator manipulates any form in the field and clicks the “Submit” button. 5. GoalManager successfully validates form data. Goal Manager queries the database and make any appropriate updates then prompts back to the *AdministratorView.* |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is prompted back to the *AdministratorView* and the Employee is properly updated to the data. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ModifyEmployee (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* where he/she is able to search for an Employee. The Administrator performs a search using the search bar.      1. GoalManager queries a list of Employee’s related to the search and populates the list of Employee’s to select from. 2. The Administrator clicks an Employee to modify. 3. GoalManager returns the *ModifyEmployeeView* with editable form fields for First Name, Last Name, Role, DepartmentId, Title. 4. Administrator manipulates any form in the field and clicks the “Submit” button or presses the “Cancel” button. 5. GoalManager unsuccessfully validates form data. Goal Manager prompts back to the *ModifyEmployeeView* or the *AdministratorView* if the Administrator pressed the “Cancel”. |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is prompted back to the *ModifyEmployeeView* with errors or is prompted to the *AdministratorView* if the “Cancel” button was pressed.  Administrator clicks the “Cancel” button on the Modify Employee View. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | AddDepartment |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdminstratorView* and selects “Add Department” button.      1. GoalManager returns the *AddDepartmentView* with empty form fields for Location, Title, Description, Supervisor. 2. The Administrator fills out the form field for Name, Location, Description, Supervisor then selects the “Submit” button. 3. GoalManager validates the form data. GoalManager then adds the Department to the database then prompts to the *AdministratorView.* |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is prompted to the *AdminstratorView* and the Department is added to the database. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | AddDepartment (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdminstratorView* and selects “Add Department” button.      1. GoalManager returns the *AddDepartmentView* with empty form fields for Location, Title, Description, Supervisor. 2. The Administrator fills out the form field for Name, Location, Description, Supervisor then selects the “Submit” button or presses the “Cancel” button. 3. GoalManager unsuccessfully validates the form data. GoalManager then prompts back to the *AddDepartmentView* with errors then prompts to the *AdministratorView,* or prompts to the *AdministratorView.* |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is prompted to the *AddDepartmentView* or selects the “Cancel”button. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ModifyDepartment |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministorView* where he/she is able to search for a Department. The Administrator performs a search using the search bar.      1. GoalManager queries a list of Department’s related to the search and populates the list of Department’s to select from. 2. The Administrator clicks a Department to modify. 3. GoalManager returns the *ModifyDepartmentView* with editable form fields for Name, Location, Description, Supervisor. 4. Administrator manipulates any form in the field and clicks the “Submit” button. 5. GoalManager successfully validates form data. Goal Manager queries the database and make any appropriate updates then prompts back to the *AdministratorView.* |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is prompted to the *AdministratorView* and the proper update is made to the Department. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ModifyDepartment (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* where he/she is able to search for an Department. The Administrator performs a search using the search bar.      1. GoalManager queries a list of Department’s related to the search and populates the list of Employee’s to select from. 2. The Administrator clicks an Employee to modify. 3. GoalManager returns the *ModifyDepartmentView* with editable form fields for Name, Location, Description, Supervisor. 4. Administrator manipulates any form in the field and clicks the “Submit” button or presses the “Cancel” button. 5. GoalManager unsuccessfully validates form data. Goal Manager prompts back to the *ModifyDepartmentView* or the *AdministratorView* if the Administrator pressed the “Cancel”. |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator enters incorrect information and selects the “Submit” button on the Modify Department View to attempt to modify the Department again. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |