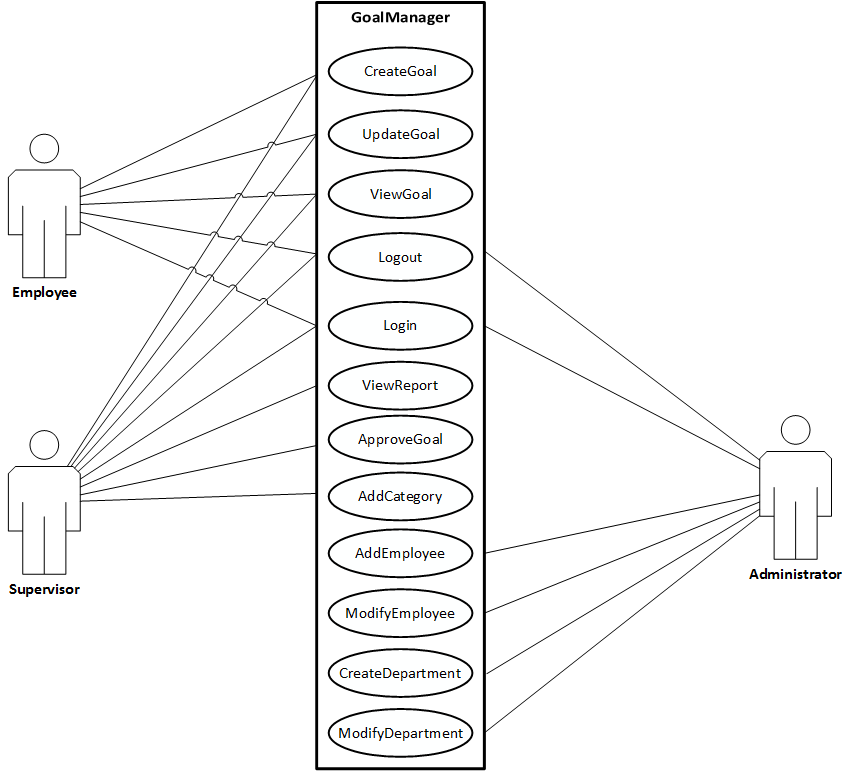
**Use Case Diagram**



Employee

|  |  |
| --- | --- |
| *Use Case name* | Login |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User clicks on the Login button.      1. GoalManager displays the Login View. 2. The User enters valid credentials into the Login View form. 3. GoalManager validates the Employee’s credentials and displays the default Employee View. 4. The User is logged into GoalManager |
| *Entry condition* | User navigates to the main View of GoalManager through a web browser. |
| *Exit condition* | User is authenticated and is successfully logged in at the directed to the default Employee View. |
| *Quality Requirements* | Fields for credentials (username and passwords) are salted and hashed, hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | Login (Failure) |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User clicks on the Login button.      1. GoalManager displays the Login View. 2. The User enters credentials into the Login View form. 3. GoalManager attempts to validate the User’s credentials, but the credentials are invalid. GoalManager displays the Login View with an error. 4. The User is not logged into GoalManager and remains on the Login View. |
| *Entry condition* | User navigates to the main View 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 and hashed, hardened against SQL injection. |

|  |  |
| --- | --- |
| *Use Case name* | ViewGoal |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is on the Main View with a list of his or her Goals and clicks the “View” button on that list item.      1. GoalManager queries the database for the selected Goal’s information and populates the View accordingly. 2. The Employee is able to see the selected Goal’s information until clicking the “Done” button. 3. GoalManager returns the Employee to the Main View. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee clicks “Done” on the Goal’s View page. |
| *Quality Requirements* |  |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Employee |
| *Flow of events* | 1. The Employee is on the Employee Main View with a list of his or her Goals and clicks the “Create Goal” button.      1. GoalManager presents the Create Goal View with its empty form fields. 2. The Employee enters valid information into the form on the Create Goal View and clicks “Submit”. 3. GoalManager successfully validates form data and writes the new Goal into the database’s pending approval queue. GoalManager returns Employee back to the Employee Main View. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee submits valid new Goal data. |
| *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 Employee |
| *Flow of events* | 1. The Employee is on the Employee Main View with a list of his or her Goals and clicks the “Create Goal” button.      1. GoalManager presents the Create Goal View with its empty form fields. 2. The Employee enters invalid information into the form on the Create Goal View and clicks “Submit”. 3. GoalManager attempts to validate form data, but invalid data fails validation. GoalManager returns Employee back to the Create Goal View with an error specific to the invalid field. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee submits invalid new Goal data. |
| *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 Employee |
| *Flow of events* | 1. The Employee is on the Employee Main View with a list of his or her Goals and clicks the “Update Goal” button on that list item.      1. GoalManager queries the database for the selected Goal’s fields and presents the Update Goal View. 2. The Employee enters valid information into the “Progress” form on the Update Goal View and clicks “Submit”. The Employee also has the option to Edit the body of a newly-created Goal for typing errors. 3. GoalManager successfully validates form data and writes the Goal’s updated progress to the database. GoalManager returns Employee back to the Employee Main View. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee submits updated Goal progress. |
| *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 Employee |
| *Flow of events* | 1. The Employee is on the Employee Main View with a list of his or her Goals and clicks the “Update Goal” button on that list item.      1. GoalManager queries the database for the selected Goal’s fields and presents the Update Goal View. 2. The Employee enters invalid information into the “Progress” form on the Update Goal View and clicks “Submit”. 3. GoalManager attempts to validate invalid form data. GoalManager returns Employee back to the Employee Update Goal View with an error. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee submits invalid Goal progress. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

|  |  |
| --- | --- |
| *Use Case name* | Logout |
| *Participating actor*  *instances* | Initiated by Employee, Supervisor, or Administrator |
| *Flow of events* | 1. The User clicks on the “Logout” button provided on each User Main View.      1. GoalManager terminates the User’s session and returns the Successful Logout View. |
| *Entry condition* | User is logged in to the system. |
| *Exit condition* | User activates the Logout function and is gracefully logged out of the web application. |
| *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 Main View with a list of thier Goals and department Goals and then clicks the “View” button on that list item.      1. GoalManager queries the database for the selected Goal’s information and populates the View accordingly. 2. The Supervisor is able to see the selected Goal’s information until clicking the “Done” button. 3. GoalManager returns the Supervisor to the Main View. |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor clicks “Done” on the Goal’s View page. |
| *Quality Requirements* |  |

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the Supervisor Main View with a list of their Goals and department Goals and then clicks the “Create Goal” button.      1. GoalManager presents the Create Goal View with its empty form fields. 2. The Supervisor enters valid information into the form on the Create Goal View and clicks “Submit”. 3. GoalManager successfully validates form data and writes the new Goal into the database’s pending approval queue. GoalManager returns Supervisor back to the Supervisor Main View. |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor submits valid new Goal data. |
| *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 Supervisor Main View with a list of their Goals and department Goals and then clicks the “Create Goal” button.      1. GoalManager presents the Create Goal View with its empty form fields. 2. The Supervisor enters invalid information into the form on the Create Goal View and clicks “Submit”. 3. GoalManager attempts to validate form data, but invalid data fails validation. GoalManager returns Supervisor back to the Create Goal View with an error specific to the invalid field. |
| *Entry condition* | Employee is logged in to the system. |
| *Exit condition* | Employee submits invalid new Goal data. |
| *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 Supervisor Main View with a list of their Goals and clicks the “Update Goal” button on that list item.      1. GoalManager queries the database for the selected Goal’s fields and presents the Update Goal View. 2. The Supervisor enters valid information into the “Progress” form on the Update Goal View and clicks “Submit”. 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 writes the Goal’s updated progress to the database. GoalManager returns Supervisor back to the Supervisor Main View. |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor submits updated Goal progress. |
| *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 Supervisor Main View 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 Update Goal View. 2. The Supervisor enters invalid information into the “Progress” form on the Update Goal View and clicks “Submit”. 3. GoalManager attempts to validate invalid form data. GoalManager returns Supervisor back to the Supervisor Update Goal View with an error. |
| *Entry condition* | Supervisor is logged in to the system. |
| *Exit condition* | Supervisor submits invalid Goal progress. |
| *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 Supervisor Main View 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 populates a form with a list of Employees under the Supervisor’s department along with a list of departments the Supervisor is in charge of. 2. The Supervisor selects an Employee or department by selecting the “View” button next to the name. 3. GoalManager validates the Supervisor’s request and displays the default Employee/Department View Report form. 4. The Supervisor is now able to view the Employee/department report. |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor selects the “Done” button on the View Report form. |
| *Quality Requirements* |  |

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

|  |  |
| --- | --- |
| *Use Case name* | AddCategory (Success) |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the Main View which displays their Goals and department Goals and then clicks the “Add Category” button on the Goal that needs to be approved.      1. GoalManager displays a form for entering Category information. 2. The Supervisor fills out the form with the correct information in each field and selects the “Confirm” button. 3. GoalManager validates the Supervisor’s form and adds the new Category to the Goal database. 4. The Supervisor is prompted back to the Main View page. |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor submits valid information in the Category form. |
| *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 Main View which displays their Goals and department Goals and then clicks the “Add Category” button on the Goal that needs to be approved. 2. GoalManager displays a form for entering Category information. 3. The Supervisor fills out the form with the incorrect information in one or more field and selects the “Confirm” button. 4. GoalManager attempts to validate the Supervisor’s form but the information is invalid and notifies Supervisor that one or more field is entered incorrectly. 5. The Supervisor is prompted back to the Add Category form to re-edit the incorrect information. |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor enters invalid information to the Category form. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |