**Requirements Analysis Document**

GoalManager System

Team #6

CSCI 4712 Senior Capstone Project

Spring 2017

Augusta University

Augusta, GA

Date: 04/21/2017

Version 3

Team Members

Andrew Keller

Matthew Tennis

Miguel Morell

**Abstract**

This document contains the requirements, analysis and the design artifacts for the GoalManager software system. GoalManager is a web-based application that allows users such as Employees, Supervisors, and Administrators to interact with the application to manage goals. Verified Employees can use the system view, update, and create goals. Verified Supervisors also have the same functionalities as Employees but with a few more capabilities such as being able to view a report, add a category, and approve a goal. Verified administrators handle user management such as adding and modifying an Employee, and creating and modifying a department.

The rest of this document is structured as follows. Chapter 1 contains the introduction. This chapter gives a brief description of the system and the scope of the system. Chapter 2 defines the functional requirements of the system. This chapter also goes over the use case of the system and the use case descriptions. Chapter 3 illustrates the user interface mockups for each form within the program. This chapter show the basic functionality and layout of the program.**Table of Contents**

1 INTRODUCTION……………………………………………………………………..4

* 1. SCOPE OF THE SYSTEM…………….……………………………4
  2. OVERVIEW OF THE DOCUMENT…………………………...…..4

1. REQUIREMENTS OF SYSTEM…….…………………………………………..….5
   1. FUNCTIONAL REQUIREMENTS………………….….…………..5
   2. USE CASES……………………………………….…….………......7
   3. USE CASE DESCRIPTION……………………….…….…………..8
2. **INTRODUCTION**

**1.1 SCOPE OF SYSTEM**

GoalManager is a web-based system for Taxslayer goals and *Employee* management. It includes three actors; an *Employee* that can login to view, create, or update goals then log out. A *Supervisor* can also login to view, create, or update goals along with being able to view a report, approve a goal, or add a goal category and then log out when finished. The *Administrator* can log in to add a new *Employee,* modify an *Employee,* add a category,modify a department, or create a department and then log out.

The system supports all three actors by keeping track of Taxslayer’s database records of all actors, goals, and departments. The *Employee*, *Supervisor*,and the *Administrator* all access the system through using a web-browser.

The functionality for CreateGoal, ViewGoal, UpdateGoal, Login, Logout, ViewReport, ApproveGoal, AddCategory, CreateEmployee, ModifyEmployee, CreateDepartment, and ModifyDepartment are completely within the scope of this system.

**1.2 OVERVIEW OF DOCUMENT**  
Chapter 2 Outlines the functional requirements of the system. Within this chapter is a list of functional requirements of GoalManager. It also includes a use case model of those functional requirements. A detailed description of each functional requirement then follows.

1. **REQUIREMENTS OF SYSTEM**

**2.1 FUNCTIONAL REQUIREMENTS**

* Login – The *Employee*, *Supervisor*, and *Administrator* all invoke this function and the system will allow the user to enter their credentials. Once verified, the system will then allow access to their respected MainView.
* CreateGoal – This is only accessible by the *Employee* and the *Supervisor*. Once verified after login, they can invoke this function on their respected MainView. They will be required to fill out a form to create a new goal for themselves. The *Supervisor* can also create a new goal for *Employees* under their department. Once submitted to a database queue, the goal will be awaiting approval.
* UpdateGoal - This is only accessible by the *Employee* and the *Supervisor*. Once verified after login, they can invoke this function on their respected MainView. They will be required to fill out a form relating to their goal progress. A *Supervisor* can also update a goal of an *Employee* under their department. Once submitted, the database updates with the new goal progress.
* ViewGoal – This is only accessible by the *Employee* and the *Supervisor*. Once verified after login, they can invoke this function on their respected MainView. They will be able to view their current goal or goals assigned to them. The *Supervisor* is also able to view the current goals of *Employees* under their department.
* ViewReport - This is only accessible by the *Supervisor*. Once verified after login, the *Supervisor* can invoke this function on SupervisorView. They will be able to view database records on themselves, their department, and the *Employees* under their department. These records will include Name, Goals Completed, Goals Failed, Role, and Department.
* ApproveGoal - This is only accessible by the *Supervisor*. Once verified after login, the *Supervisor* can invoke this function on SupervisorView. The *Supervisor* can select to “Approve” or “Deny” a goal. Once approved, the goal updates to the database as Approved.
* AddCategory - This is only accessible by the *Supervisor* and *Administrator*. Once verified after login, they can invoke this function on their respective HomeView. They will be able to fill a form out to create a new goal category that will update to the database.
* CreateEmployee – This is only accessible by the *Administrator*. Once verified after login, the *Administrator* can invoke this function on AdministratorView. They will be required to enter valid *Employee* information on a form. After submission, the new *Employee* updates to the database.
* ModifyEmployee - This is only accessible by the *Administrator*. Once verified after login, the *Administrator* can invoke this function on AdministratorView. They will be able to alter *Employee* record information and update the changes to the database.
* CreateDepartment - This is only accessible by the *Administrator*. Once verified after login, the *Administrator* can invoke this function on AdministratorView. They will be able to fill out a form to create a new department that will update to the database.
* ModifyDepartment - This is only accessible by the *Administrator*. Once verified after login, the *Administrator* can invoke this function on AdministratorView. They will be able to alter a department’s record information that will update to the database.
* Logout – Once logged in, the system will give access to logout, if invoked it will prompt the system to the LoginView and disallow access to the databases.

**2.2 USE CASES**

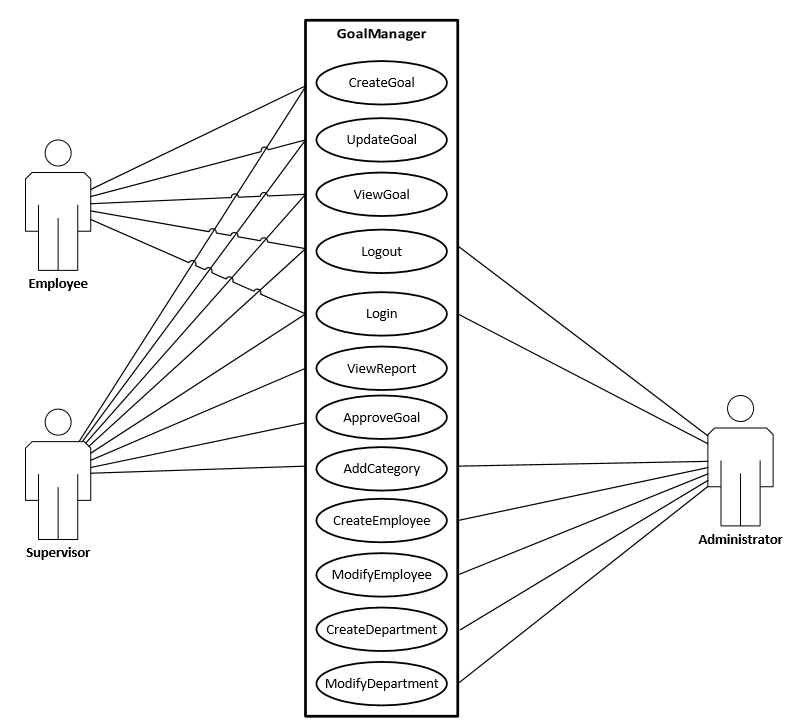


Figure 2.1: Use Case Diagram for GoalManager System

**2.3 USE CASE DESCRIPTIONS**

Figure 2.2: Login

|  |  |
| --- | --- |
| *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. |

Figure 2.3: Login (Failure)

|  |  |
| --- | --- |
| *Use Case name* | ViewGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “View” button attached to an individual Goal list element.      1. **GoalManager recognizes that the User is an Employee and then queries the database for the properties of the selected Goal, including its Title, Timeframe, Category, and Progress. GoalManager then populates the partial *ViewGoalView* Employee version 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 directed to the *EmployeeView.* |
| *Quality Requirements* |  |

Figure 2.4: ViewGoal

Figure 2.5: CreateGoal (Employee)

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “Create Goal” button. 2. **GoalManager recognizes that the User is an Employee and returns an Employee version of *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 directed 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. |

Figure 2.4: CreateGoal

Figure 2.6: CreateGoal (Employee Failure)

Figure 2.5: CreateGoal (Failure)

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “Create Goal” button. 2. **GoalManager recognizes that the User is an Employee and returns an Employee version of *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 directed to the *CreateGoalView* with errors or is directed 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. |

Figure 2.7: UpdateGoal (Employee)

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “Update Goal” button attached to a Goal list element. 2. **GoalManager recognizes that the User is an Employee and queries the database for the selected Goal’s properties, returning an Employee version of *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 directed 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. |

Figure 2.6: UpdateGoal

Figure 2.8: UpdateGoal (Employee Failure)

Figure 2.7: UpdateGoal (Failure)

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “Update Goal” button attached to a Goal list element. 2. **GoalManager recognizes that the User is an Employee and queries the database for the selected Goal’s properties, returning an Employee version of *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 directed to the *UpdateGoalView* with errors or directed 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. |

Figure 2.9: Logout

|  |  |
| --- | --- |
| *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. |

|  |  |
| --- | --- |
| *Use Case name* | ViewGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then selects the “View” button on that list item.      1. **GoalManager recognizes that the User is a Supervisor and queries the database for the properties of the selected Goal, including its Title, Timeframe, Category, and Progress. GoalManager then populates the partial *ViewGoalView* Supervisor version 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 directed to *SupervisorView*. |
| *Quality Requirements* |  |

Figure 2.10: ViewGoal (Supervisor)

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then selects the “Create Goal” button.      1. **GoalManager recognizes that the User is a Supervisor and opens aSupervisor version of *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). |

Figure 2.11: CreateGoal (Supervisor)

|  |  |
| --- | --- |
| *Use Case name* | CreateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then selects the “Create Goal” button.      1. **GoalManager recognizes that the User is a Supervisor and opens aSupervisor version of *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 directed back to CreateGoalView with errors or is directed to *SupervisorView* if the “Cancel” button is pressed. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

Figure 2.12: CreateGoal (Supervisor Failure)

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then selects the “Update Goal” button on that list item.      1. **GoalManager recognizes that the User is a Supervisor and queries the database for the selected Goal’s properties, presenting a Supervisor version of *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 directed to the *SupervisorView* and the Goal is properly updated in the database. |
| *Quality Requirements* | Fields are hardened against SQL injection and personally identifying information is encrypted (when applicable). |

Figure 2.13: UpdateGoal (Supervisor)

|  |  |
| --- | --- |
| *Use Case name* | UpdateGoal (Failure) |
| *Participating actor*  *instances* | Initiated by Employee or Supervisor |
| *Flow of events* | 1. The User is logged in and on *MainView* with a list of his or her Goals. The User then clicks the “Update Goal” button on that list item.      1. **GoalManager recognizes that the User is a Supervisor and queries the database for the selected Goal’s properties, presenting a Supervisor version of *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 directed 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). |

Figure 2.14: UpdateGoal (Supervisor Failure)

|  |  |
| --- | --- |
| *Use Case name* | ViewReport |
| *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 a “View Report” button under the Employee list.      1. **GoalManager queries the database and presents the *ViewReportView*. *ViewReportView* populates a table of the Employee’s goals along with a Search form field for filtering data.** 2. The Supervisor is now able to view the Employee Report and selects the “Home” button when finished. 3. **GoalManager closes *ViewReportView* and returns Supervisor back to *SupervisorView.*** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is returned to *SupervisorView*. |
| *Quality Requirements* | Search field is hardened against SQL injection and personally identifying information is encrypted (when applicable). |

Figure 2.15: ViewReport (for Employees)

|  |  |
| --- | --- |
| *Use Case name* | ViewReport |
| *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 a “View Report” button under the Department list.      1. **GoalManager queries the database and presents the *ViewReportView*. *ViewReportView* populates a table of Employee’s goal history under the Department the Supervisor is in charge of, along with a Search form field for filtering data. *ViewReportView* also populates two graphs based on the number of goals the that are completed, active, denied, and failed for the entire department and for all Employees under that department.** 2. The Supervisor is now able to view the Department Report and selects the “Home” button when finished. 3. **GoalManager closes *ViewReportView* and returns Supervisor back to *SupervisorView.*** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is returned to *SupervisorView*. |
| *Quality Requirements* | Search field is hardened against SQL injection and personally identifying information is encrypted (when applicable). |

Figure 2.16: ViewReport (for Department)

|  |  |
| --- | --- |
| *Use Case name* | ApproveGoal |
| *Participating actor*  *instances* | Initiated by Supervisor |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView*, 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 directed to the *SupervisorView* and the Goal is approved. |
| *Quality Requirements* |  |

Figure 2.17: ApproveGoal

|  |  |
| --- | --- |
| *Use Case name* | AddCategory (Success) |
| *Participating actor*  *instances* | Initiated by Supervisor or Administrator |
| *Flow of events* | 1. The Supervisoris on the *SupervisorView* which displays their Goals and department Goals and then selects the “Add Category” button.      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 *SupervisorView*.** |
| *Entry condition* | Supervisor is logged in. |
| *Exit condition* | Supervisor is directed to the *SupervisorView* and the Category is added to the Database. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.18: AddCategory

|  |  |
| --- | --- |
| *Use Case name* | AddCategory (Failure) |
| *Participating actor*  *instances* | Initiated by Supervisor or Administrator |
| *Flow of events* | 1. The Supervisor is on the *SupervisorView* which displays their Goals and department Goals and then selects the “Add Category” button.. 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 directed to the *AddCategoryView* with errors or to the *SupervisorView* if “Cancel” was pressed. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.19: AddCategory (Failure)

|  |  |
| --- | --- |
| *Use Case name* | CreateEmployee |
| *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 *CreateEmployeeView* with empty form fields for First Name, Last Name, Email, Role, Department, and Title.** 2. The Administrator fills out credentials related to form fields First Name, Last Name, Department, Title, Email, and 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 *AdministratorView* |
| *Exit condition* | Administrator is directed back to the *AdministratorView* and Employee is added to the database. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.20: CreateEmployee

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

Figure 2.21: CreateEmployee (Failure)

|  |  |
| --- | --- |
| *Use Case name* | ModifyEmployee |
| *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 the “Modify” button next to an Employee to modify. 3. **GoalManager returns the *ModifyEmployeeView* with editable form fields for First Name, Last Name, Email, Role, Department, Title, and Active Status.** 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 directed back to the *AdministratorView* and the Employee is properly updated to the data. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.22: ModifyEmployee

|  |  |
| --- | --- |
| *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 the “Modify” button next to an Employee to modify. 3. **GoalManager returns the *ModifyEmployeeView* with editable form fields for First Name, Last Name, Email, Role, Department, Title, and Active Status.** 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 directed back to the *ModifyEmployeeView* with errors or is directed 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. |

Figure 2.23: ModifyEmployee (Failure)

|  |  |
| --- | --- |
| *Use Case name* | ModifyDepartment |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* and selects “Modify Department”. 2. **GoalManager returns the *ModifyDepartmentView* with editable form fields for Name, Location, Description, and a dropdown field for Supervisor.** 3. Administrator manipulates any form in the field and clicks the “Submit” button. 4. **GoalManager successfully validates form data. GoalManager queries the database and makes any appropriate updates then prompts back to the *AdministratorView.*** |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator is directed to the *AdministratorView* and the proper update is made to the Department. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.24: ModifyDepartment

|  |  |
| --- | --- |
| *Use Case name* | ModifyDepartment (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* and selects “Modify Department”.      1. **GoalManager returns the *ModifyDepartmentView* with editable form fields for Name, Location, Description, and a dropdown field for Supervisor.** 2. Administrator manipulates any form in the field and clicks the “Submit” button. 3. **GoalManager unsuccessfully validates form data. GoalManager prompts back to the *ModifyDepartmentView*.** |
| *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. |

Figure 2.25: ModifyDepartment (Failure)

|  |  |
| --- | --- |
| *Use Case name* | CreateDepartment |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* and selects “Create Department”.      1. **GoalManager returns the *CreateDepartmentView* with fields for Department name, Location, Description, Supervisor, Goal Category and Quarter 1 – Quarter 4 Start and End dates.** 2. The Administrator enters valid information into the form fields and selects “Submit”. 3. **GoalManager successfully validates form data. GoalManager updates the database to make any appropriate changes and then returns the Administrator back to the *AdministratorView*.** |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator enters correct information and selects the “Submit” button on the Create Department View. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.26: CreateDepartment

|  |  |
| --- | --- |
| *Use Case name* | CreateDepartment (Failure) |
| *Participating actor*  *instances* | Initiated by Administrator |
| *Flow of events* | 1. The Administrator is on the *AdministratorView* and selects “Create Department”.      1. **GoalManager returns the *CreateDepartmentView* with fields for Department Name, Location, Description, Supervisor, Goal Category, and Quarter 1 – Quarter 4 Start and End dates.** 2. The Administrator enters invalid information into the form fields and selects “Submit”. 3. **GoalManager unsuccessfully validates form data. GoalManager prompts back to the *CreateDepartmentView* with a list of Model errors pertaining to the invalid form fields.** |
| *Entry condition* | Administrator is logged in. |
| *Exit condition* | Administrator enters incorrect information and selects the “Submit” button on the Create Department View to attempt to create the Department again. |
| *Quality Requirements* | Fields for credentials are hardened against SQL injection. |

Figure 2.27: CreateDepartment (Failure)

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

Figure 2.28: AddCategory

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

Figure 2.29: AddCategory (Failure)