|  |  |
| --- | --- |
| Use Case Section | Use Case UC#1 |
| Use Case Name | Adding a Program |
| Scope | Management – Adding a Program to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to add a program to DB  Employee – Be able to select a program from the bug report. |
| Preconditions | Bughound Program up and running:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to add a program |
| Main Success Scenario | 1. Manager complete the form for program information  2. Manager submit the program information by clicking on submit button  3. The program data is added to Bughound database.  4. A success message appears on the screen  5. Manager is redirected to the main page of the Bughound web app  6. Employee is able to choose the program from the bug report form |
| Extensions | 1a. Information are not entered correctly   1. User is notified by an error on the page 2. User can correct the information   2a. Manager click on cancel button   1. Prompt for cancel confirmation; ‘YES’ to continue, ‘Do Not Discard” to cancel |
| Frequency of Occurrence | Each time a new program is needed to be added by manager |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#2 |
| Use Case Name | Updating an existing Program |
| Scope | Management – Updating an existing Program to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to update an existing program to DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to update an existing program |
| Main Success Scenario | 1. Manager selects program from existing programs list and is redirected to its edit page 2. Manager is redirected to the update page for the selected program 3. Manager modifies the fields available for edit 4. Manager clicks submit and the update is applied to the database 5. Success message will appear and the manager will be redirected to the programs’ list page |
| Extensions | 1a. Manger decides to discard the changes   1. Prompt for discard confirmation; ‘YES’ to continue, ‘Do Not Discard” to cancel 2. Manger clicks cancel button and is directed to the programs’ list page   1b. Manager decides to not to update program information before making any changes   1. Manager clicks the cancel button and is directed to the programs’ list page |
| Frequency of Occurrence | Each time the manager updates an existing program |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#3 |
| Use Case Name | Deleting a Program |
| Scope | Management – Deleting a Program from DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Ability to delete a program from DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to delete a program from DB |
| Main Success Scenario | 1. Manager selects program from existing programs’ list and is redirected to its edit page 2. Manager clicks on ‘Delete’ button; a prompt for deletion confirmation will appear 3. Manager confirms delete operation 4. Success message appears on screen and manager is redirected to main page |
| Extensions | 1a. Manger decides not to delete the program.   1. Manger clicks cancel button and is directed to the programs’ list page   2a. Manager decides to not to delete program information when asked for confirmation   1. Manager clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the manager deletes a program |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#4 |
| Use Case Name | Adding Functional Area |
| Scope | Management – Adding a Program to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to add a functional area to DB  Employee – Be able to select a functional area from the bug report. |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to add a functional area to the DB |
| Main Success Scenario | 1. Manager complete the form for functional area information after selecting an existing program  2. Manager submit the functional area information by clicking on submit button  3. The functional area data is added to Bughound database.  4. A success message appears on the screen  5. Manager is redirected to the main page of the Bughound web app  6. Employee is able to choose the functional area from the bug report form |
| Extensions | 1a. Manager enters incomplete information.   1. User is notified by an error on the page   2a. Manager click on cancel button   1. Prompt for cancel confirmation; ‘YES’ to continue, ‘Do Not Discard” to cancel |
| Frequency of Occurrence | Each time a new functional area is needed to be added by manager |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#5 |
| Use Case Name | Updating an existing Functional Area |
| Scope | Management – Updating an existing Functional Area to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to update an existing functional area to DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to update an existing functional area |
| Main Success Scenario | 1. Manager selects functional area from existing areas list and is redirected to its edit page after he select an existing program 2. Manager fills out the form for their update 3. Manager clicks submit and the update is applied to the database 4. Success message will appear and the manager will be redirected to the main page |
| Extensions | 1a. Manager enters incomplete information.   1. User is notified by an error on the page   2a. Manager click on cancel button   1. Prompt for cancel confirmation; ‘YES’ to continue, ‘Do Not Discard” to cancel |
| Frequency of Occurrence | Each time the manager updates an existing functional area |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#6 |
| Use Case Name | Deleting a Functional Area |
| Scope | Management – Deleting a Functional Area from DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Ability to delete a functional area from DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to delete a functional area from DB |
| Main Success Scenario | 1. Manager selects functional area from existing areas list and is redirected to its edit page after he select an existing program 2. Manager clicks on ‘Delete’ button; a prompt for deletion will appear 3. Manager confirms delete operation 4. Success message appears on screen and manager is redirected to main page |
| Extensions | 1a. Manger decides not to delete the area.   1. Manger clicks cancel button and is directed to the areas’ list page   2a. Manager decides to not to delete area when asked for confirmation   1. Manager clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the manager deletes a functional area |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#7 |
| Use Case Name | Adding Employee |
| Scope | Management – Adding an Employee to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to add an employee to DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to add an employee to the DB |
| Main Success Scenario | 1. Manager complete the form for employee information  2. Manager submit the employee information by clicking on submit button  3. The employee data is added to Bughound database.  4. A success message appears on the screen  5. Manager is redirected to the main page of the Bughound web app  6. Employee is able to choose employees from the bug report form |
| Extensions | 1a. Manager enters incomplete information.   1. User is notified by an error on the page   2a. Manager click on cancel button   1. Prompt for cancel confirmation; ‘YES’ to continue, ‘Do Not Discard” to cancel |
| Frequency of Occurrence | Each time a new employee is needed to be added by manager |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#8 |
| Use Case Name | Updating an existing Employee |
| Scope | Management – Updating an existing Employee to DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Be able to update an existing employee to DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to update an existing employee |
| Main Success Scenario | 1. Manager selects an employee from existing employees list and is redirected to its edit page 2. Manager fills out the form for their update 3. Manager clicks submit and the update is applied to the database 4. Success message will appear and the manager will be redirected to the main page |
| Extensions | 1a. Manger decides not to update employee information   1. Manger clicks cancel button and is directed to employees’ list page   2a. Manager decides not to apply changes to employee information   1. Manager clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the manager updates an existing employee |

|  |  |
| --- | --- |
| Use Case Section | Use Case UC#9 |
| Use Case Name | Deleting an Employee from DB |
| Scope | Management – Deleting an Employee from DB |
| Level | User-level goal |
| Primary Actor | DB User (Manager) |
| Stakeholders and Interests | Manager – Ability to delete an employee from DB |
| Preconditions | Bughound Program up and running:  Bughound:   1. Database is configured and running 2. Web server is configured and running 3. Bughound GUI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to delete a functional area from DB |
| Main Success Scenario | 1. Manager selects an employee from existing employees list and is redirected to its edit page 2. Manager clicks on ‘Delete’ button; a prompt for deletion will appear 3. Manager confirms delete operation 4. Success message appears on screen and manager is redirected to main page |
| Extensions | 1a. Manger decides not to delete employee information   1. Manger clicks cancel button and is directed to employees’ list page   2a. Manager decides not to delete employee   * + - 1. Manager clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the manager deletes an employee |