|  |  |  |
| --- | --- | --- |
|  | | |
| Department of Computer Engineering and Computer Science, CSULB  CECS 544 Software Test and Verification  Dr. Michael A. Hoffman | | |
| **Term Project: BugHound (Bug Report and tracking Web App)** | | |
|  | | |
| Project Team Members: | | |
| Loc Hinh | Phuc Dang | Hadi Sanjani |

Table of Contents

[1 Project Description 1](#_Toc35225254)

[1.1 Key Features 1](#_Toc35225255)

[1.2 Constraints 1](#_Toc35225256)

[2 Iteration I 2](#_Toc35225257)

[2.1 ER diagram for all DB elements related to programs, areas and employees 2](#_Toc35225258)

[2.2 Use cases for management functions for the DB 2](#_Toc35225259)

[2.2.1 Program: adding, updating, or deleting 2](#_Toc35225260)

[2.2.2 Employee: adding, updating, or deleting 2](#_Toc35225261)

[2.2.3 Functional Area: adding, updating, or deleting 2](#_Toc35225262)

[2.2.4 Bug: adding a bug, editing a bug, closing a bug, searching for a bug 4](#_Toc35225263)

Table of Figures

[Figure 1: Bug report form 1](#_Toc35225264)

[Figure 2: ER Diagram for Database 2](#_Toc35225265)

Table of Tables

[Table ‎2‑1: Adding a Program 2](#_Toc35225275)

[Table ‎2‑2: Updating an existing Program 2](#_Toc35225276)

[Table ‎2‑3: Deleting a Program 2](#_Toc35225277)

[Table ‎2‑4: Adding Employee 2](#_Toc35225278)

[Table ‎2‑5: Updating and existing Employee 2](#_Toc35225279)

[Table ‎2‑6: Deleting an existing Employee 2](#_Toc35225280)

[Table ‎2‑7: Adding Functional Area 2](#_Toc35225281)

[Table ‎2‑8: Updating an existing Functional Area 3](#_Toc35225282)

[Table ‎2‑9: Deleting a Functional Area 3](#_Toc35225283)

[Table ‎2‑10: adding a bug report 4](#_Toc35225284)

[Table ‎2‑11: editing a bug report 5](#_Toc35225285)

[Table ‎2‑12: closing a bug report 5](#_Toc35225286)

[Table ‎2‑13: search system for specific bug 6](#_Toc35225287)

# Project Description

This project is about writing a web-based bug recording and tracking software with a user-friendly Web UI with a specific set of features.

## Key Features

The key features of this software are as following:

* Using web browser, create, edit and update “bug” reports on multiple products
* Store error report content in relational tables
* Access error report content via SQL
* Search for bugs on multiple fields
* Facilities to add, delete or update information on program, releases, functional areas, employees

## Constraints

1. **Must be web browser based**
2. **An SQL database is required to be used for application data**
3. **The program must have a UI for all user accessible functions.**
4. **It will require server and client scripting such as php or asp and javascript**
5. **The Bug report form should be based on** Figure 1

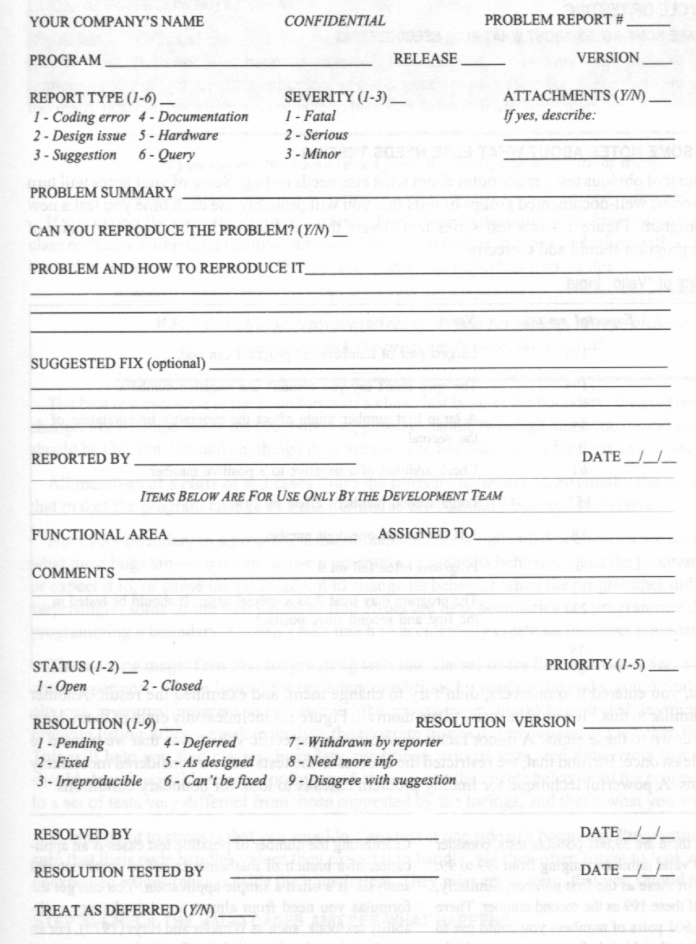
****

Figure : Bug report form

# Iteration I

## ER diagram for all DB elements related to programs, areas and employees

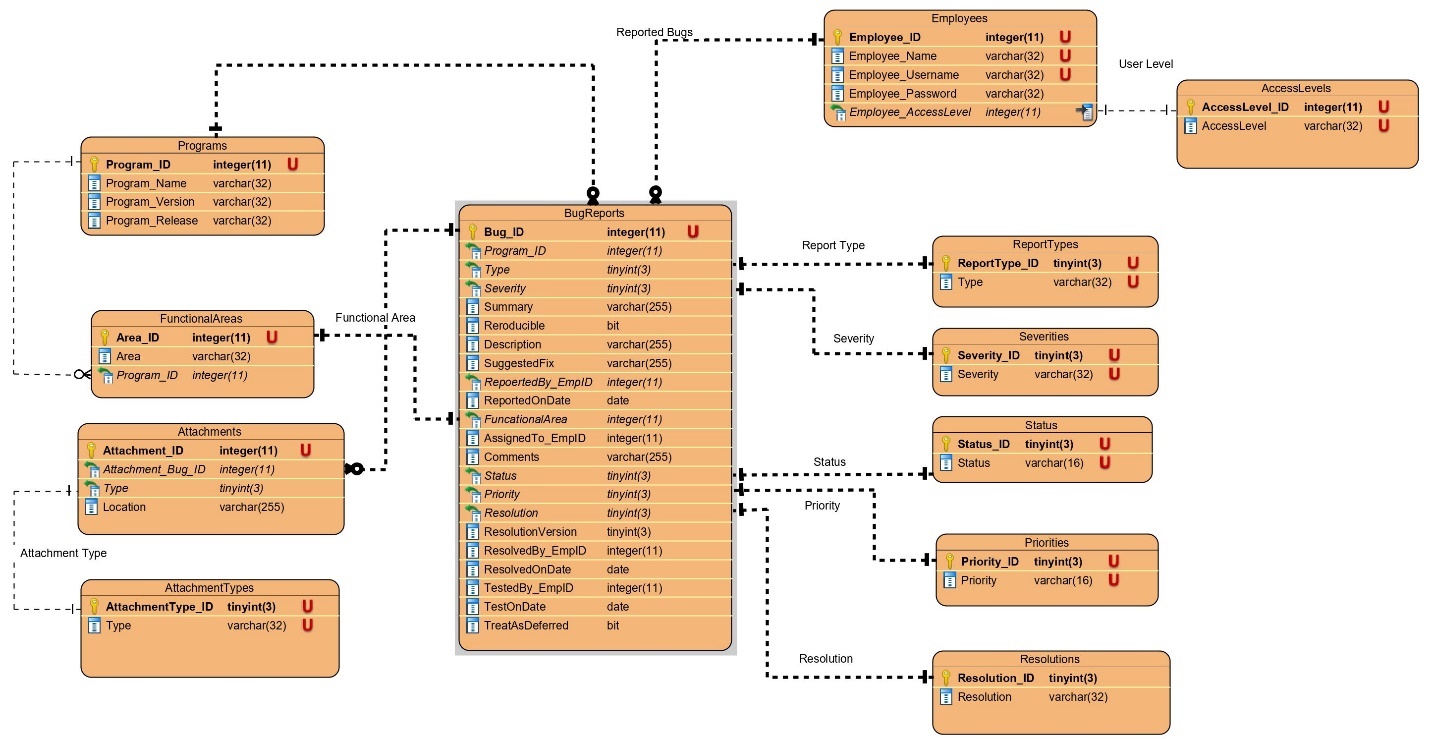


Figure : ER Diagram for Database

## Use cases for management functions for the DB

### Program: adding, updating, or deleting

Table ‑: Adding a Program

|  |  |
| --- | --- |
| Use Case Section | Use Case BH1 |
| 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 Web UI fully functional 4. All back-end functions are fully functional |
| Success Guarantee | User is able to add a program |
| Main Success Scenario | 1. Manager clicks on the Database maintenance in the main page 2. Manager complete the form for program information 3. Manager submit the program information by clicking on submit button 4. The program data is added to Bughound database. 5. A success message appears on the screen 6. Manager is redirected to the main page of the Bughound web app 7. Employee is able to choose the program from the bug report form |
| Extensions | 3a. Information are not entered correctly   1. User is notified by an error on the page 2. User is able correct the information   2,3a. 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 |

Table ‑: Updating an existing Program

|  |  |
| --- | --- |
| Use Case Section | Use Case BH2 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects program from existing programs list and is redirected to its edit page 3. Manager is redirected to the update page for the selected program 4. Manager modifies the fields available for edit 5. Manager clicks submit and the update is applied to the database 6. Success message will appear and the manager will be redirected to the programs’ list page |
| Extensions | 4a. 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   5a. 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 |
| Frequency of Occurrence | Each time the manager updates an existing program |

Table ‑: Deleting a Program

|  |  |
| --- | --- |
| Use Case Section | Use Case BH3 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects program from existing programs’ list and is redirected to its edit page 3. Manager clicks on ‘Delete’ button; a prompt for deletion confirmation will appear on the screen 4. Manager confirms delete operation 5. Success message appears on screen and manager is redirected to main page |
| Extensions | 3a. Manger decides not to delete the program.   1. Manger clicks cancel button and is redirected to the programs’ list page   4a. 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 |

### Employee: adding, updating, or deleting

Table ‑: Adding Employee

|  |  |
| --- | --- |
| Use Case Section | Use Case BH4 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager complete the form for employee information 3. Manager submit the employee information by clicking on submit button 4. The employee data is added to Bughound database. 5. A success message appears on the screen 6. Manager is redirected to the main page of the Bughound web app 7. User is able to choose added employee from the bug report form |
| Extensions | 2a. Manager enters incomplete information.   1. User is notified by an error on the page   3a. 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 |

Table ‑: Updating and existing Employee

|  |  |
| --- | --- |
| Use Case Section | Use Case BH5 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects an employee from existing employees list and is redirected to its edit page 3. Manager fills out the form for their update 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 main page |
| Extensions | 2a. Manger decides not to update employee information   1. Manger clicks cancel button and is directed to employees’ list page   4a. 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 |

Table ‑: Deleting an existing Employee

|  |  |
| --- | --- |
| Use Case Section | Use Case BH6 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects an employee from existing employees list and is redirected to its edit page 3. Manager clicks on ‘Delete’ button; a prompt for deletion will appear 4. Manager confirms delete operation 5. Success message appears on screen and manager is redirected to main page |
| Extensions | 3a. Manger decides not to delete employee information   1. Manger clicks cancel button and is directed to employees’ list page   4a. 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 |

### Functional Area: adding, updating, or deleting

Table ‑: Adding Functional Area

|  |  |
| --- | --- |
| Use Case Section | Use Case BH7 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager complete the form for functional area information after selecting an existing program 3. Manager submit the functional area information by clicking on submit button 4. The functional area data is added to Bughound database. 5. A success message appears on the screen 6. Manager is redirected to the main page of the Bughound web app 7. Employee is able to choose the functional area from the bug report form |
| Extensions | 3a. Manager enters incomplete information.   1. User is notified by an error on the page   3b. 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 |

Table ‑: Updating an existing Functional Area

|  |  |
| --- | --- |
| Use Case Section | Use Case BH8 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects functional area from existing areas list and is redirected to its edit page after he select an existing program 3. Manager fills out the form for their update 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 main page |
| Extensions | 4a. Manager enters incomplete information.   1. User is notified by an error on the page   4b. 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 |

Table ‑: Deleting a Functional Area

|  |  |
| --- | --- |
| Use Case Section | Use Case BH9 |
| 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:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI 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 clicks on the Database maintenance in the main page 2. Manager selects functional area from existing areas list and is redirected to its edit page after he select an existing program 3. Manager clicks on ‘Delete’ button; a prompt for deletion will appear 4. Manager confirms delete operation 5. Success message appears on screen and manager is redirected to main page |
| Extensions | 3a. Manger decides not to delete the area.   1. Manger clicks cancel button and is directed to the areas’ list page   4a. 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 |

### Bug: adding a bug, editing a bug, closing a bug, searching for a bug

Table ‑: adding a bug report

|  |  |
| --- | --- |
| Use Case Section | Use Case BH10 |
| Use Case Name | Adding a bug report |
| Scope | User – adding a bug report |
| Level | User-level goal |
| Primary Actor | Employee |
| Stakeholders and Interests | Manager – Ability to monitor and track reported bugs of a program  User (Employee) – Ability to submit bug reports |
| Preconditions | Bughound Program up and running:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI fully functional 4. All back-end functions are fully functional 5. User is logged in |
| Success Guarantee | User is able to add a bug report |
| Main Success Scenario | 1. User clicks on the add a bug report in the main page 2. User enters the bug report form page 3. User fills out all the fields of the bug report form 4. User click save (submit) button 5. User is asked for the confirmation of adding the bug report 6. Success message appears on screen and user is redirected to main page |
| Extensions | 2a. User decides not to add bug report   1. User clicks cancel button and is directed to main page   4.a. User did not fill the minimum required form’s fields   1. Notify the user about the error   5a. User decides to not to add the bug report when asked for confirmation   1. User clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the user adds a new bug report |

Table ‑: editing a bug report

|  |  |
| --- | --- |
| Use Case Section | Use Case BH11 |
| Use Case Name | Editing a bug report |
| Scope | User – editing a bug report |
| Level | User-level goal |
| Primary Actor | Employee |
| Stakeholders and Interests | Manager – Ability to monitor and track updates of reported bugs of a program  User (Employee) – Ability to edit an already submitted bug reports by himself and also be able to see whether the submitted bug is resolved or not |
| Preconditions | Bughound Program up and running:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI fully functional 4. All back-end functions are fully functional 5. User is logged in |
| Success Guarantee | User is able to edit a bug report |
| Main Success Scenario | 1. User clicks on the update an existing bug in the main page 2. User select a bug report form the listed bugs 3. User modifies any editable field of the bug report 4. User click save (submit) button 5. User is asked for the confirmation of finalizing the bug report 6. Success message appears on screen and user is redirected to main page |
| Extensions | 2a,4a. User decides not to edit the bug report   1. User clicks cancel button and is directed to bug list page   4.b. User did not fill the minimum required form’s fields   1. Notify the user about the error   5a. User decides to not to save the modified bug report when asked for confirmation   1. User clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the user edits an existing bug report |

Table ‑: closing a bug report

|  |  |
| --- | --- |
| Use Case Section | Use Case BH12 |
| Use Case Name | Closing a bug report |
| Scope | User – Closing a bug report |
| Level | User-level goal |
| Primary Actor | Employee |
| Stakeholders and Interests | Manager – Ability to monitor and track closed bugs’ report of a program  User (Employee) – Ability to close an existing bug reports |
| Preconditions | Bughound Program up and running:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI fully functional 4. All back-end functions are fully functional 5. User is logged in |
| Success Guarantee | User is able to close a bug report |
| Main Success Scenario | 1. User clicks on the update an existing bug in the main page 2. User select a bug report form the listed bugs 3. User modifies any editable field of the bug report 4. User click save (submit) button 5. User is asked for the confirmation of finalizing the bug report 6. Success message appears on screen and user is redirected to main page |
| Extensions | 2a,4a. User decides not to edit the bug report   1. User clicks cancel button and is directed to bug list page   5a. User decides to not to save the modified bug report when asked for confirmation   1. User clicks the cancel button when asked for confirmation |
| Frequency of Occurrence | Each time the user wants to close an existing bug report |

Table ‑: search system for specific bug

|  |  |
| --- | --- |
| Use Case Section | Use Case BH13 |
| Use Case Name | Search system for a specific bug report |
| Scope | User – search for a bug report |
| Level | User-level goal |
| Primary Actor | Employee |
| Stakeholders and Interests | Manager – Ability to monitor and track bugs’ report of a program  User (Employee) – Ability to search for an existing bug reports |
| Preconditions | Bughound Program up and running:   1. Database is configured and running 2. Web server is configured and running 3. Bughound Web UI fully functional 4. All back-end functions are fully functional 5. User is logged in |
| Success Guarantee | User is able to search for a bug report |
| Main Success Scenario | 1. User clicks on the search for an existing bug in the main page 2. User fills searchable fields in search form 3. User click search button 4. Search results appears on screen and user is able to click on the each found results and redirected to the selected bug report |
| Extensions | 2a. User decides not to search for a bug report   1. User clicks cancel button and is directed main page   4a. User decides to not to go to any bug report   1. User clicks the cancel button and redirected to main page |
| Frequency of Occurrence | Each time the user wants to search for an existing bug report |