Bug Tracker Application- Use Case Document

Created @May 26, 2023

Table of Contents

1. Introduction

The Bug Tracker is a software application designed to manage and track bugs reported for different projects. It allows users to create bugs, update bug status, search bugs based on various criteria, view bugs associated with specific users or projects and performing administrative actions like deleting bugs. The application provides administrative capabilities to manage bugs efficiently.

This use case document outlines the key use cases and interactions for the Bug Tracker application, considering the roles of Admin and User.

2. Actors

The following actors interact with the Bug Tracker application:

1. Admin: An administrative user responsible for managing bugs and performing CRUD operations.
2. User: A user who reports bugs and interacts with the Bug Tracker application.

3. Use Cases

3.1. Create Bug

Brief Description: This use case describes the process of creating a bug.

Primary Actor: Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.
2. The application presents a bug creation form.
3. The User enters the necessary information about the bug, such as name, project, description, and status.
4. The User submits the bug creation form.
5. The application validates the form inputs and creates a new bug entry in the system.

Alternative Flows:

- If the form inputs are invalid or incomplete (Step 5), the application displays error messages and prompts the User to correct the information.
- If there is an error during the bug creation process (e.g., connectivity issues, system errors), the application displays an error message and allows the User to retry the bug creation.

3.2. Read Bug

Brief Description: This use case describes the process of viewing bug details.

Primary Actor: User/Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.
2. The application displays a list of bugs.
3. The User selects a specific bug from the list.
4. The application retrieves and displays the detailed information of the selected bug, including name, project, description, and status.

3.3. Update Bug Status

Brief Description: This use case describes the process of updating the status of a bug.

Primary Actor: User/Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.
2. The User selects a bug to update the status.
3. The application presents an update form for the selected bug.
4. The User updates the status of the bug.
5. The User submits the update form.
6. The application validates the update and updates the bug status in the system.

Alternative Flows:

- If the update form inputs are invalid or incomplete (Step 6), the application displays error messages and prompts the User to correct the information.
- If there is an error during the status update process (e.g., connectivity issues, system errors), the application displays an error message and allows the User to retry the update.

3.4. Search Bugs

Brief Description: This use case describes the process of searching for bugs based on name, project, and status.

Primary Actor: User/Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.
2. The application presents a search form.
3. The User enters the search criteria, such as bug name, project, and status.
4. The User submits the search form.
5. The application retrieves and displays a list of bugs that match the search criteria.

Alternative Flows:

- If no bugs match the search criteria (Step 5), the application displays a message indicating no bugs found.

3.5. View Bugs by User

Brief Description: This use case describes the process of viewing bugs raised under a specific User's name.

Primary Actor: User/Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.
2. The application presents a list of bugs associated with the User's account.
3. The User selects a specific bug from the list.
4. The application retrieves and displays the detailed information of the selected bug, including name, project, description, and status.

3.6. View Bugs by Project

Brief Description: This use case describes the process of viewing bugs associated with a specific project.

Primary Actor: User/Admin

Preconditions: The User is authenticated and has access to the Bug Tracker application.

Basic Flow:

1. The User opens the Bug Tracker application.

2. The application presents a list of projects.
3. The User selects a specific project.
4. The application retrieves and displays a list of bugs associated with the selected project.
5. The User selects a specific bug from the list.
6. The application retrieves and displays the detailed information of the selected bug, including name, project, description, and status.

3.7. Delete Bug

Brief Description: This use case describes the process of deleting a bug by an Admin.

Primary Actor: Admin

Preconditions: The Admin is authenticated and has administrative privileges in the Bug Tracker application.

Basic Flow:

1. The Admin opens the Bug Tracker application.
2. The application displays a list of bugs.
3. The Admin selects a specific bug to delete.
4. The application prompts for confirmation.
5. The Admin confirms the deletion.
6. The application removes the bug from the system.

Alternative Flows:

- If the Admin cancels the deletion (Step 5), the application returns to the bug list without deleting the bug.

4. System Architecture

The Bug Tracker application follows a client-server architecture. The client-side is developed using a frontend framework (Angular) for the user interface, while the server-side involves a backend application developed using a backend framework (Java-Spring boot) and a database (MySQL) to store bug and user information.

The client-side communicates with the server-side via RESTful APIs to perform various operations such as bug creation, updates, searches, retrieving and deleting

bug details. The server-side handles the business logic, validation, and interacts with the database to store and retrieve data.

5. Assumptions and Dependencies

- Users have valid accounts and proper authentication measures are in place.
- The application integrates with a database to store bug and user information.
- Users have access to a stable internet connection for using the application.
- The application is compatible with modern web browsers and devices.