UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

CTRL-X: BUG TRACKING SYSTEM

Software Requirement Specification

Software Engineering

Submitted to:
Dr. Shantanu Agnihotri sir

Submitted by:
Naman Chanana SAP Id: 500119485
Soumya Jain SAP Id: 500119436
Smriti Walia SAP Id: 500124833
Amulya Jain SAP Id: 500122439

Software Requirements Specification (SRS) for Bug Tracking System

1. Abstract

This document defines the requirements for the **Bug Tracking System**, which we are developing to enable software teams to **report, track, assign, and resolve bugs** efficiently. This system serves as a centralized platform for managing software defects, ensuring transparency in issue resolution and streamlining development workflows. We have structured this document according to the IEEE standard for Software Requirements Specification while maintaining a real-world approach.

---

2. Introduction

2.1 Purpose

The purpose of this document is to define the **external requirements** for the Bug Tracking System. We are building this system to enhance software quality assurance by allowing users to report bugs, assign them to developers, track resolution status and generate reports. It will serve as a **collaborative tool** for development teams to efficiently handle and resolve software issues.

2.2 Scope

This document outlines all the requirements for the Bug Tracking System. The system will allow:

- **Bug reporting with file attachments (logs, screenshots, and other supporting documents).**
- **Role-based access (Admin, Developer, Tester) with clearly defined permissions.**
- **Bug status tracking and discussions through comments.**
- **Priority tagging and filtering by severity, assignee, and status.**
- **Real-time notifications for bug assignment, comments, and status updates.**
- **Exporting reports in PDF/CSV formats for tracking and auditing purposes.**
- **Security measures such as authentication and authorization (JWT-based authentication).**
- **Integration with email notifications for critical bug alerts.**

2.3 Definitions, Acronyms, Abbreviations

- **Bug**: A software defect or issue that needs to be fixed.
- **Reporter**: A user (typically a tester) who logs a bug.
- **Assignee**: A developer responsible for fixing a bug.

- **Priority**: The importance of the bug (Low, Medium, High, Critical).
- **Status**: The state of a bug (Open, In Progress, Resolved, Closed).
- **JWT**: JSON Web Token (Used for authentication).

2.4 References

- IEEE Standard for Software Requirements Specifications.

2.5 Developer's Responsibilities

We are responsible for:

- Designing and implementing the system.
- Deploying and maintaining the system.
- Providing training and documentation for users.
- Ensuring security, scalability and usability of the system.
- Addressing user feedback and future enhancements.

---

3. General Description

3.1 Product Functions Overview

The Bug Tracking System includes the following core functionalities:

1. **User Authentication & Role Management**: Secure login, registration and role-based access control.
2. **Bug Reporting & Management**: Users can report bugs with relevant details and file attachments.
3. **Bug Assignment & Status Updates**: Admins can assign bugs and developers can update their status.
4. **Dashboard & Filtering**: View bug reports based on different parameters (status, severity, priority).
5. **Notifications**: Users receive real-time alerts for bug assignments, comments, and status changes.
6. **Comments & Discussions**: Developers and testers can discuss bug resolutions in a thread.
7. **Report Generation**: Generate and download reports in PDF or CSV format.
8. **Multiple Assignees**: Bugs can be assigned to multiple developers or testers.
9. **Tagging System**: Bugs can be categorized for better tracking.

3.2 User Characteristics

- **Testers**: Report and track bugs, verify fixes and add comments.
- **Developers**: View assigned bugs, update status and discuss issues.
- **Admins**: Manage users, assign bugs, monitor system activity, and generate reports.

3.3 General Constraints

- The system should be accessible via **web browsers** (Chrome, Firefox, Edge).
- Backend services must be hosted on **Render/AWS**, with **MongoDB Atlas** as the database.
- Frontend should be deployed on **Vercel/Netlify**.
- The system must support **file attachments up to 10MB per bug**.

3.4 Assumptions & Dependencies

- The system requires a **stable internet connection**.
- Users must have valid credentials to access the system.
- Bugs must be manually reported; **automated bug detection is not in scope**.

---

4. Specific Requirements

4.1 Inputs and Outputs

Inputs:

- **Bug reports** submitted by testers.
- **File attachments** (logs, screenshots) for better debugging.
- **User authentication credentials** (JWT-based login system).
- **Bug updates** such as status change and comments.

Outputs:

- **Dashboard displaying bug statuses**.
- **Real-time notifications for updates**.
- **PDF/CSV reports of bug data**.
- **History logs of bug changes.**

4.2 Functional Requirements

1. **User Authentication & Role-Based Access**
    - Secure login system using JWT authentication.
    - Role-based access control (Admin, Developer, Tester).
2. **Bug Reporting System**
    - Users can submit bug reports with details and attachments.
    - Each bug is assigned a unique **Bug ID**.
3. **Bug Assignment & Tracking**
    - Admins assign bugs to developers.
    - Developers update status: **(Open → In Progress → Resolved → Closed)**.
    - A bug history log tracks all status updates.
4. **Comment System for Bugs**
    - Users can discuss issues in **comment threads**.
    - Comments are time-stamped and linked to specific users.
5. **Bug Filtering & Search**
    - Search bugs by **title, severity, status or priority**.
    - Filter bugs by **assignee, date reported and category**.
6. **File Attachments**
    - Users can upload **up to 3 files per bug**.
    - Supported formats: .png, .jpg, .pdf, .log.
7. **Report Generation**
    - Users can download **PDF or CSV reports** for analysis.
8. **Real-time Notifications**
    - WebSockets notify users when bugs are assigned or resolved.
    - Email alerts are sent for critical issues.
9. **Multiple Assignees & Tagging**
    - Bugs can be assigned to multiple users for collaborative debugging.
    - Tags help categorize bugs for better tracking.

4.3 External Interface Requirements

- **User Interface**: The system will have a web-based UI built with **React.js and TailwindCSS**.

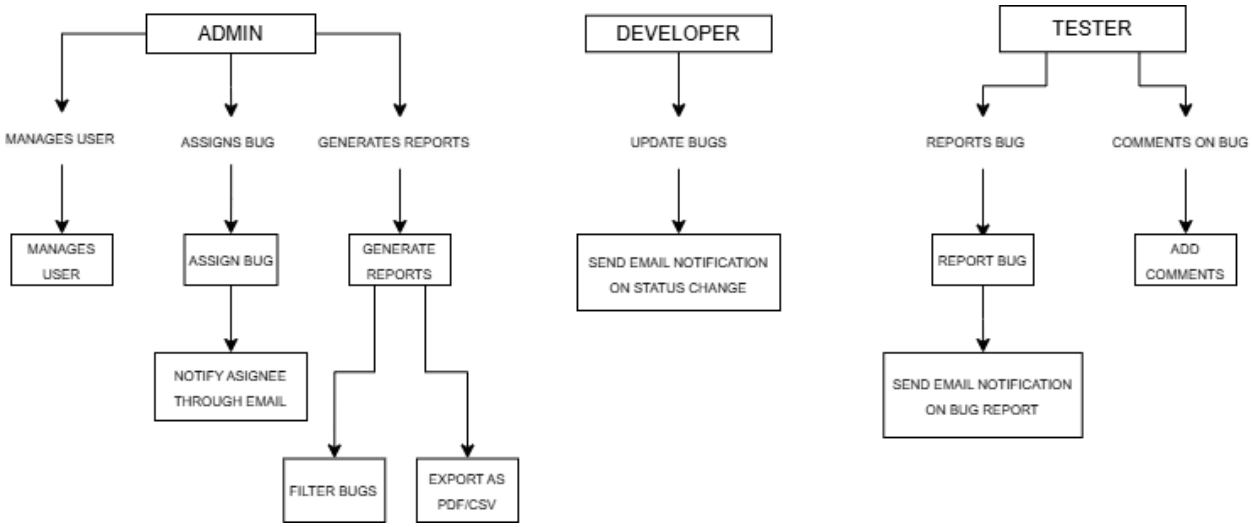
- **API Interface**: The backend will be a **RESTful API** built using **Express.js and MongoDB**.
- **Security Interface**: JWT authentication and role-based access will be implemented.

4.3.1 Use Case Diagram

The **Use Case Diagram** below represents how different actors interact with the **Bug Tracking System**.

- **Actors:** Admin, Developer, Tester
- **Use Cases:** Reporting Bugs, Assigning Bugs, Updating Status, Generating Reports, etc.
- **Relationships:**
    - <<include>> → Used when one use case **always** calls another (e.g., Assign Bug <<include>> Notify Assignee).
    - <<extend>> → Used when one use case **optionally extends** another (e.g., Generate Report <<extend>> Export as PDF/CSV).

Diagram:



4.4 Performance Constraints

- The system must handle **up to 1,000 bug reports**.
- Dashboard must **load within 3 seconds** for up to **1,000 bugs**.
- PDF & CSV report generation should **complete in 10 seconds**.

4.5 Design Constraints

Software Constraints

- The system is designed to be **platform-independent** and will run on **cloud-based infrastructure** (AWS/Render) rather than a specific OS like UNIX.
- It will be built using **Node.js (Express.js) for the backend** and **React.js for the frontend**.

Hardware Constraints

- The system will be hosted on cloud-based virtual servers instead of dedicated hardware.
- It will support scalable architecture to handle high bug-reporting volumes.

Acceptance Criteria

- The system must correctly store and retrieve bug reports.
- Users should be able to **assign, update, and close** bug reports.
- The dashboard should correctly display **bug statuses and history**.
- PDF/CSV export should work without errors.
- Security testing should confirm **secure authentication and authorization**.

4.6 Use Case Diagram: -