DESIGN DOCUMENT

Bug Tracking System is a tool that is helpful for tracking and solving bugs. The system provides a user-friendly interface, developed using HTML, CSS, and JavaScript, to allow employees to access the project they are working on and view the project details and the bugs related to it.
There are three types of users, Project Manager, Developer and Tester.
The backend is built using JDBC and MYSQL, manages all the core functionalities, including user access control rights, bug creation and closing, project details, team members working on each project.
Although the frontend and backend are currently not linked, this document outlines the design principles and architecture of the backend system, ensuring that it adheres to industrial coding standards and robust error handling practices.

Technology Stack

We used the following technologies:-

Frontend

1. HTML
2. CSS [Framework -Bootstrap]
3. JS

Backend

1. JAVA
2. JDBC
3. MYSQL
4. Junit

Project Scope

Key Functionalities Covered

User Management: Users can register, log in, and access the system based on their roles (Project Manager, Developer, Tester).

Project Management: Project Managers can create and manage projects, assign team members, and track the progress of the project.

Bug Reporting: Testers can report bugs, assign severity levels, and view bugs associated with projects they are working on.

Bug Tracking: Project Managers can view all bugs in the project, assign them to developers, and track the resolution progress.

Bug Resolution: Developers can update the status of bugs (e.g., mark them as resolved or close them).

Data Import: The system supports importing users and bug data from JSON or XML files.

Exclusions

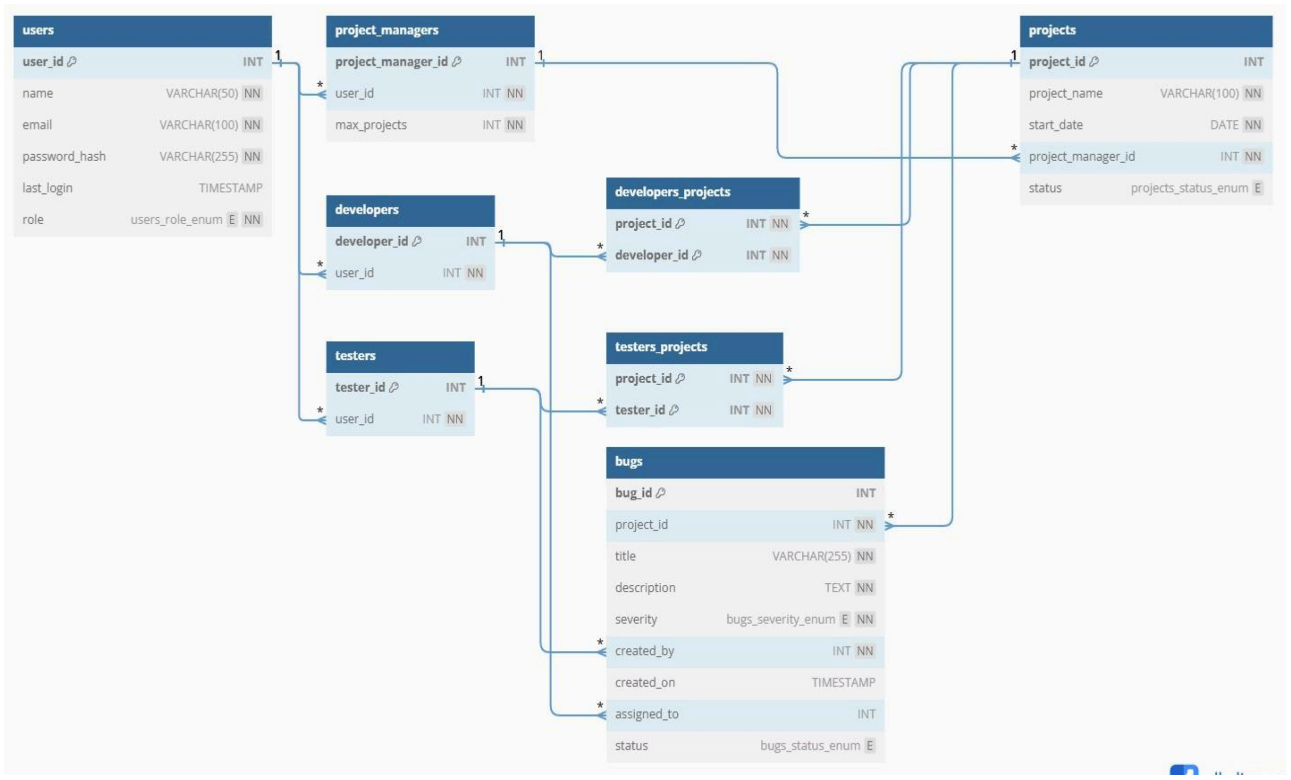
Integration with Version Control: The system does not include direct integration with version control systems like Git.

Real-Time Notifications: The system does not support real-time notifications or alerts for bug assignments or status changes.

Mobile Compatibility: The system is designed for desktop use and does not include a mobile-responsive design.

Advanced Reporting: The system does not provide detailed analytics or custom reporting features beyond basic bug tracking.

ER(ENTITY RELATIONSHIP) DIAGRAM



Users: Contains basic user information with fields for user_id, name, email, password_hash, last_login, and role.

Project Managers: Each project manager is a user (user_id) with an additional field for max_projects.

Testers: Each tester is a user (user_id) with a separate tester_id.

Developers: Each developer is a user (user_id) with a separate developer_id.

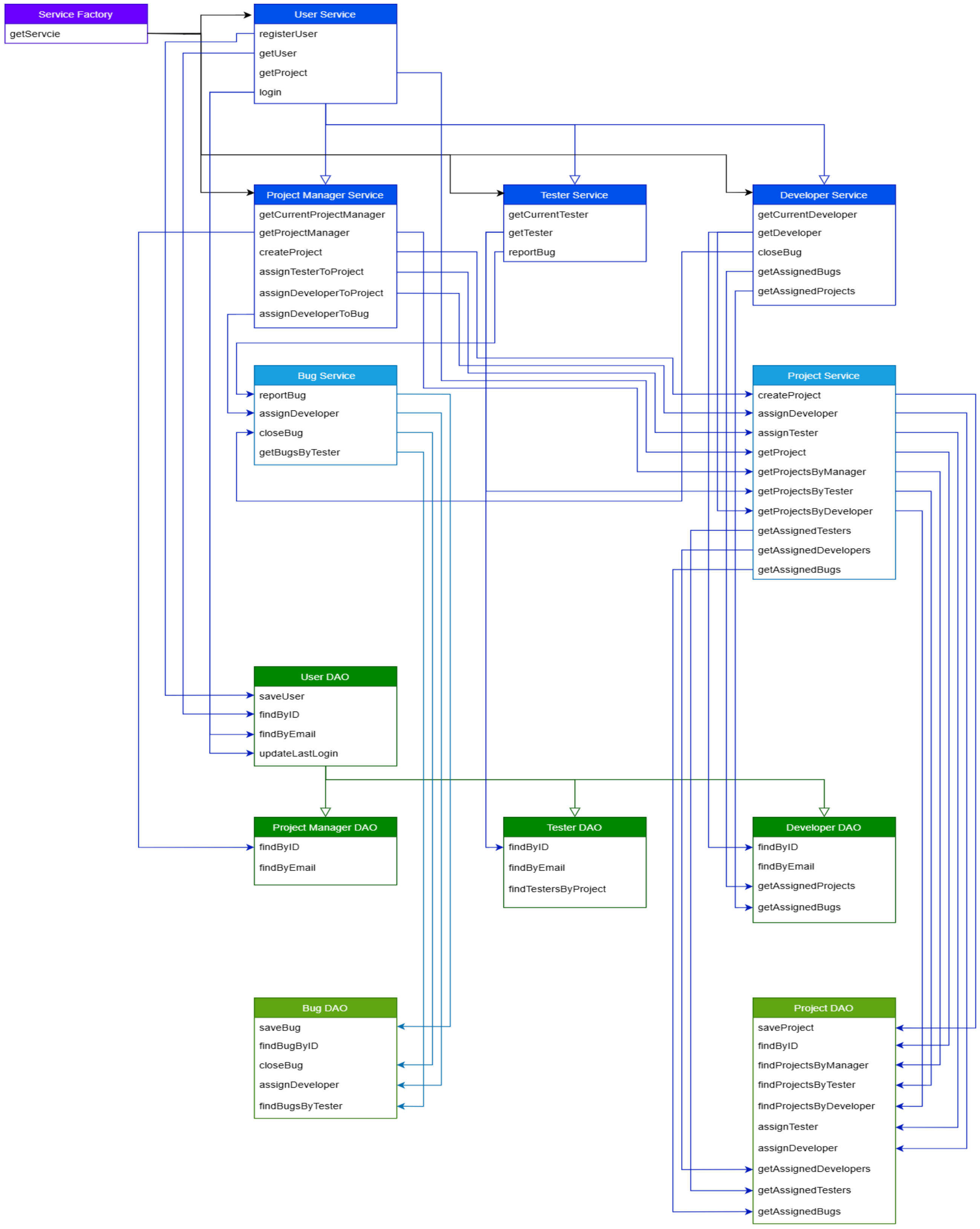
Projects: Contains details about projects such as project_id, project_name, start_date, project_manager_id, and status.

Testers_Projects: A junction table linking testers to projects (project_id, tester_id).

Developers_Projects: A junction table linking developers to projects (project_id, developer_id).

Bugs: Stores bug reports with fields like bug_id, project_id, title, description, severity, created_by, created_on, assigned_to, and status.

UML CLASS DIAGRAM



Service Factory:
Provides access to different services through the getService() method.

User Service:
Handles user-related operations like registerUser, getUser, and login.

Project Manager Service:
Manages project manager-specific functions like creating a project, assigning testers and developers to projects, and getting project-related information.

Tester Service:
Manages tester-related operations such as retrieving tester information and reporting bugs.

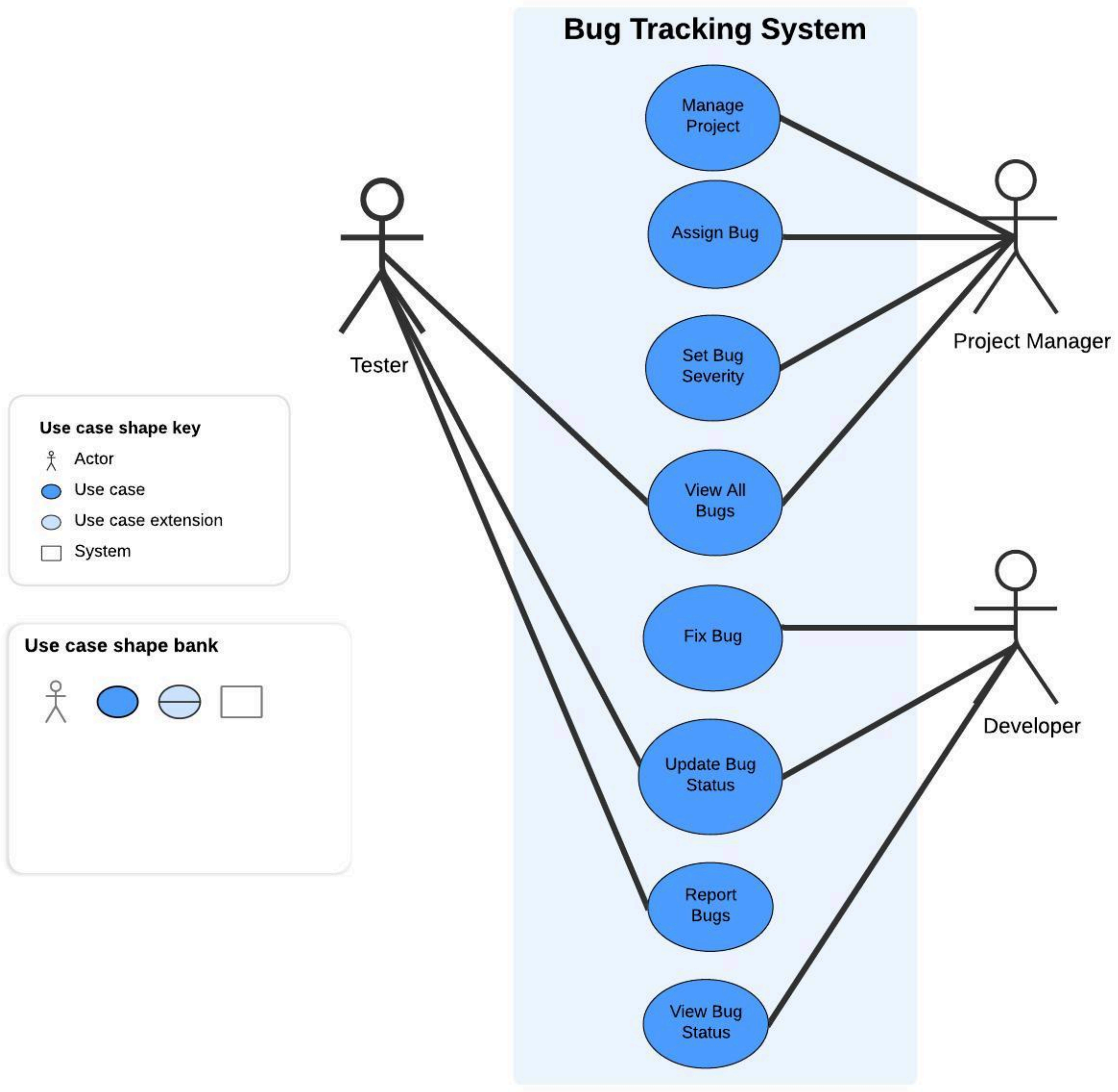
Developer Service:
Manages developer-related tasks like closing bugs, viewing assigned bugs, and retrieving developer information.

Bug Service:
Handles operations related to bugs, including reporting, assigning developers, closing bugs, and retrieving bugs by tester.

Project Service:
Manages project-related operations, such as creating projects, assigning users (developers/testers), and retrieving projects based on different roles.

DAOs (Data Access Objects):
User DAO: Handles database operations for users, like saving a user, finding by ID, and updating the last login.
Project Manager DAO, Tester DAO, Developer DAO: Each handles database operations related to their respective roles.
Bug DAO: Manages database operations related to bugs, like saving, finding, assigning developers, and closing bugs.
Project DAO: Handles database operations for projects, such as saving a project, finding by ID, and retrieving projects based on different criteria.

USE CASE DIAGRAM



Each actor is associated with specific use cases:

Tester can:
- Report Bugs
- Set Bug Severity

- View Bug Status

Project Manager can:
- Manage Project
- Assign Bug
- View All Bugs
- Set Bug Severity
- View Bug Status

Developer can:
- Fix Bug
- Update Bug Status
- View Bug Status

Error Handling and Validations

-User Registration and Login

Validation: Ensure all required fields (e.g., username, email, password) are filled. Validate email format and password strength.

Error Handling: Display appropriate error messages for invalid inputs or if the user already exists in the system.

-Project Management

Validation: Check that the project start date is at least 2 days later than the current date. Ensure no developer is assigned to more than one project, and no tester is assigned to more than two projects.

Error Handling: Handle errors like project creation failure, duplicate project names, and invalid team assignments.

-Bug Reporting

Validation: Ensure that bugs are reported only by testers assigned to the project and that all required fields (e.g., bug title, description, severity level) are filled.

Error Handling: Provide feedback if the user tries to report a bug for a project not in progress or if the bug data is incomplete.

-Bug Assignment and Resolution

Validation: Ensure that bugs are assigned to developers and that developers mark bugs as resolved or closed only after resolving them.

Error Handling: Handle cases where a developer tries to close a bug that is not assigned to them or is already closed.

Future Scope

- *Integration with CI/CD Pipelines:* Integrate the bug tracking system with CI/CD pipelines to automatically create bug reports for failed builds or tests.

- *Real-Time Notifications:* Implement real-time notifications to alert users when bugs are assigned, updated, or resolved.

- *Advanced Reporting:* Add advanced reporting features to generate custom reports based on various criteria like severity, status, and user assignments.

- *Mobile App Development:* Develop a mobile application to provide access to the bug tracking system on the go.

- *AI-Powered Bug Detection:* Incorporate AI/ML algorithms to predict potential bugs based on code patterns and historical data.

Conclusion

The Bug Tracking System is a comprehensive tool designed to manage software development projects by tracking and resolving bugs efficiently. It provides a structured way to report, assign, and resolve bugs, ensuring that the project progresses smoothly. While the current implementation covers the essential functionalities, there is scope for further enhancement, including integration with other development tools and platforms. With its modular design and robust error handling, the system is well-equipped to handle various scenarios in a software development lifecycle.