SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

21CSC206P - Advanced Object Oriented Programming - Project Review 2

Bug Tracking System

Pratip Das RA2311056010188
Asutosh Ranjan RA2311056010190

Table of contents

- Abstract
- Objectives
- Architecture diagram
- Software Requirements
- Proposed System
- GUI Design
- Database Design
- Sample code
- Conclusion & Future Enhancement
- References

Abstract

- This project is aimed at developing a tracking system useful for application developed in an organization.
- The BugTracking System (BTS) is a application that can be accessed throughout the organization.
- This system can be used for logging defects against an application/module, assigning defects to individuals and tracking the defects to resolution.
- There are features like email notifications, user maintenance, user access control, report generators etc in this system.
- Bug tracking is the process of finding defects in a product (testing), and making new versions of the product that fix the defects.

Objectives

A bug tracking system aims to streamline the process of identifying, reporting, and resolving software defects. Here are the primary objectives:

1. **Efficient Defect Management:**

- **Centralized Repository:** Provide a centralized platform to store and track all reported bugs.
- **Clear Prioritization:** Allow for easy categorization and prioritization of bugs based on severity and impact.
- **Detailed Tracking:** Enable detailed tracking of each bug's lifecycle, from initial report to resolution and closure.

2. **Improved Communication and Collaboration:**

- **Effective Communication:** Facilitate seamless communication between developers, testers, and other stakeholders involved in the bug-fixing process.
- **Collaboration:** Promote collaboration among team members by providing a shared workspace for discussing and resolving issues.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be **University** *u/s 3 of UGC Act, 1956*

Objectives(Cont.)

3. Enhanced Productivity:

- **Time-Saving:** Streamline the bug-reporting and resolution process, reducing wasted time and effort.
- **Increased Efficiency:** Improve overall development efficiency by identifying and fixing bugs promptly.

4. Data Analysis and Reporting:

- **Data Collection:** Gather and analyze data related to bug trends, severity, and resolution times.
- **Reporting:** Generate comprehensive reports to help teams identify areas for improvement and make data-driven decisions.

Objectives(Cont.)

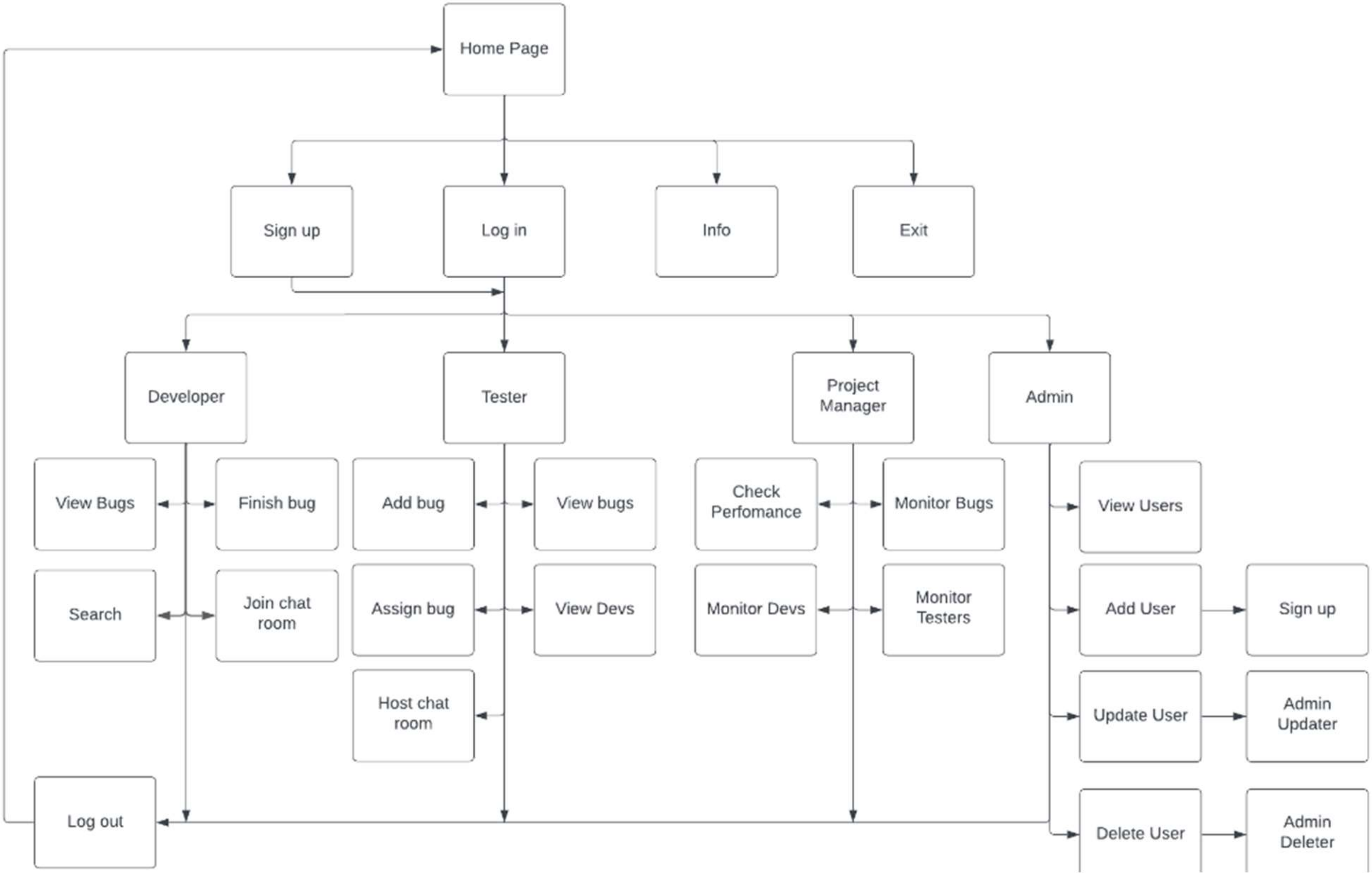
5. Quality Assurance:

- **Quality Improvement:** Contribute to the overall quality of the software product by ensuring that defects are addressed in a timely and effective manner.
- **Compliance:** Help organizations comply with industry standards and regulations related to software quality.

6. Scalability and Flexibility:

- **Scalability:** Accommodate the growth of the software project and the team over time.
- **Flexibility:** Adapt to changing requirements and workflows.

Architecture Diagram



Home Page

Sign up
Log in
Info
Exit

Developer
Tester
Project Manager
Admin

View Bugs
Finish bug
Search
Join chat room

Add bug
View bugs
Assign bug
View Devs
Host chat room

Check Perfomance
Monitor Bugs
Monitor Devs
Monitor Testers

View Users
Add User
Sign up
Update User
Admin Updater
Delete User
Admin Deleter

Log out

Software Requirements

1. **Presentation Layer (Client/Frontend):**

- **Technologies**: JSP/Servlets, HTML, CSS, JavaScript, Angular/React (optional)
- **Purpose**: Provides a user interface where users can interact with the system to report bugs, view bug statuses, and manage bug assignments.

2. **Business Logic Layer (Backend):**

- **Technologies**: Java, Spring Boot, Servlets, REST API, Hibernate/JPA (for ORM)
- **Purpose**: Implements core business logic, such as handling bug operations (add, update, delete, search) and processing user requests.

3. **Data Layer (Database):**

- **Technologies**: MySQL, PostgreSQL, or any relational database, Hibernate/JPA (for ORM)
- **Purpose**: Stores bug records, user information, and related data such as comments, status changes, and history logs.

Proposed System / Work

Overcoming the limitations of traditional bug tracking systems often involves incorporating advanced tools, methodologies, and processes that enhance efficiency, accuracy, and collaboration. Here's an approach to improving and modernizing a bug tracking system:

1. Adopt an Agile and Automated Testing Approach

- **Integration with CI/CD Pipelines**: Implement continuous integration/continuous deployment (CI/CD) to automate testing, detect bugs early, and address them faster. Automated tests run with each code commit can identify bugs before they reach production.
- **Test-Driven Development (TDD)**: Writing tests before code ensures that bugs are minimized from the start, leading to fewer issues in the bug tracking system.

2. Utilize AI and Machine Learning for Predictive Analysis

- **Prioritize Bugs Based on Impact and History**: Machine learning algorithms can analyze bug patterns, prioritize critical bugs, and predict potential issues based on past incidents. This can optimize resolution efforts by focusing on bugs that are most likely to impact functionality or user experience.
- **Intelligent Duplicate Detection**: AI models can help identify duplicate bug reports, saving time by preventing developers from addressing the same issue multiple times.

Proposed System / Work

3. Enhance Collaboration and Communication

- **Unified Communication Tools**: Integrate the bug tracking system with collaboration platforms like Slack, Microsoft Teams, or Jira to streamline communication between developers, testers, and product managers. Real-time notifications and discussions can enhance responsiveness and problem-solving.
- **Bug Assignment and Status Transparency**: Assign bugs based on developer expertise and workload. Maintain transparency in bug status, which encourages accountability and faster resolution.

4. Integrate User Feedback Directly into the System

- **In-app Reporting and Screen Recording**: Allow users to report bugs directly within the application, with the option to include screenshots, screen recordings, and logs. This can provide developers with valuable context, reducing the time spent reproducing bugs.
- **Collect User Behavior Data**: Monitoring user interactions can provide insights into where bugs occur and help developers identify issues that may not be easily reproducible.

Proposed System / Work

5. Implement Enhanced Data Analytics and Reporting

- **Real-time Analytics Dashboard**: Set up dashboards that visualize bug data, such as the frequency of specific issues, open/closed bug ratios, and time to resolution. This allows teams to quickly identify bottlenecks in the bug resolution process.
- **Root Cause Analysis (RCA) Tools**: Automate RCA tools to identify underlying issues that contribute to recurring bugs, which can help developers address the root of the problem instead of repeatedly fixing symptoms.

6. Encourage Cross-Functional Bug Reviews and Training

- **Cross-Team Bug Review Sessions**: Involve teams outside of development, such as product design and QA, in bug review meetings. This brings diverse perspectives to bug prioritization and resolution.
- **Skill Enhancement Workshops**: Regular training in new debugging tools, testing frameworks, and secure coding practices can empower developers to prevent bugs rather than rely solely on tracking them.

Modules

The Bug Tracking System project accommodates multiple user roles, including testers, developers, project managers, and admins. Each role is equipped with specific functionalities tailored to their responsibilities and permissions.

1. Tester:

- Define bugs with detailed information, including bug name, ID, type, priority, level of difficulty, project name, start date, and due date.
- View all bugs defined by them.
- Interact with developers, assign bugs to specific developers.
- Attach images to bug reports within the specified size limit (5MB).
- Send email notifications to assigned developers with comprehensive bug details.
- Host chat rooms for direct communication with developers.

Modules

2. Developer:

- View all bugs assigned to them.
- Update the status of assigned bugs as they progress in resolving them.
- Participate in chat rooms by entering the required IP address and port number for direct communication with testers.
- Search for relevant information and access useful links related to bug fixing.

3. Project Manager:

- Have a comprehensive view of testers, developers, and bugs in the system.
- Calculate performance metrics for testers and developers based on parameters like the number of issued and solved bugs, difficulty, and bug level.
- Generate reports on bug status, performance metrics, and other relevant data.

4. Admin:

- View, add, update, and delete user accounts.
- Have elevated privileges and complete control over user management in the system.

GUI Design

1. Dashboard:

- **Overview:** Displays a summary of the bug tracking system's status, including the number of open, closed, and in-progress bugs.
- **Charts and Graphs:** Visualizes data to provide insights into bug trends, severity levels, and resolution times.
- **Recent Activity:** Shows a feed of recent bug updates, such as new reports, status changes, or comments.

2. Bug Reporting Module:

- **Bug Details:** Fields for entering bug title, description, severity level, priority, assigned developer, and due date.
- **Attachments:** Allows users to upload screenshots, log files, or other relevant documentation.
- **Custom Fields:** Provides flexibility to add additional fields specific to the organization's needs.
- **Workflow:** Defines the bug lifecycle, such as "New," "Assigned," "In Progress," "Resolved," and "Closed."

GUI Design(Cont.)

3. Bug Search and Filtering:

- **Search Bar:** Allows users to search for bugs by keyword, title, description, or other criteria.
- **Filters:** Provides options to filter bugs by status, severity, priority, assigned developer, due date, and custom fields.
- **Saved Searches:** Enables users to save frequently used search criteria for quick access.

4. Bug Detail View:

- **Comprehensive Information:** Displays all relevant details about a specific bug, including its status, history, comments, and attachments.
- **Comments:** Allows users to add comments, questions, or updates to a bug.
- **Attachments:** Provides a list of all attached files.
- **Workflow Actions:** Enables users to change the bug's status, assign it to a different developer, or update other details.

GUI Design(Cont.)

5. Developer Portal:

- **Assigned Bugs:** Displays a list of bugs assigned to the developer.
- **Bug Detail View:** Provides the same functionality as the bug detail view for general users.
- **Time Tracking:** Allows developers to record time spent on each bug.
- **Code Repository Integration:** Links to the organization's code repository to facilitate bug analysis and resolution.

Database Design

Entities:

- **Bug:** Represents a reported issue or defect.
- **User:** Represents a person who interacts with the system (e.g., reporter, developer, tester).
- **Project:** Represents a software project or product.
- **Priority:** Represents the urgency or importance of a bug.
- **Severity:** Represents the impact of a bug on the system.
- **Status:** Represents the current state of a bug (e.g., New, Assigned, In Progress, Resolved, Closed).

Attributes:

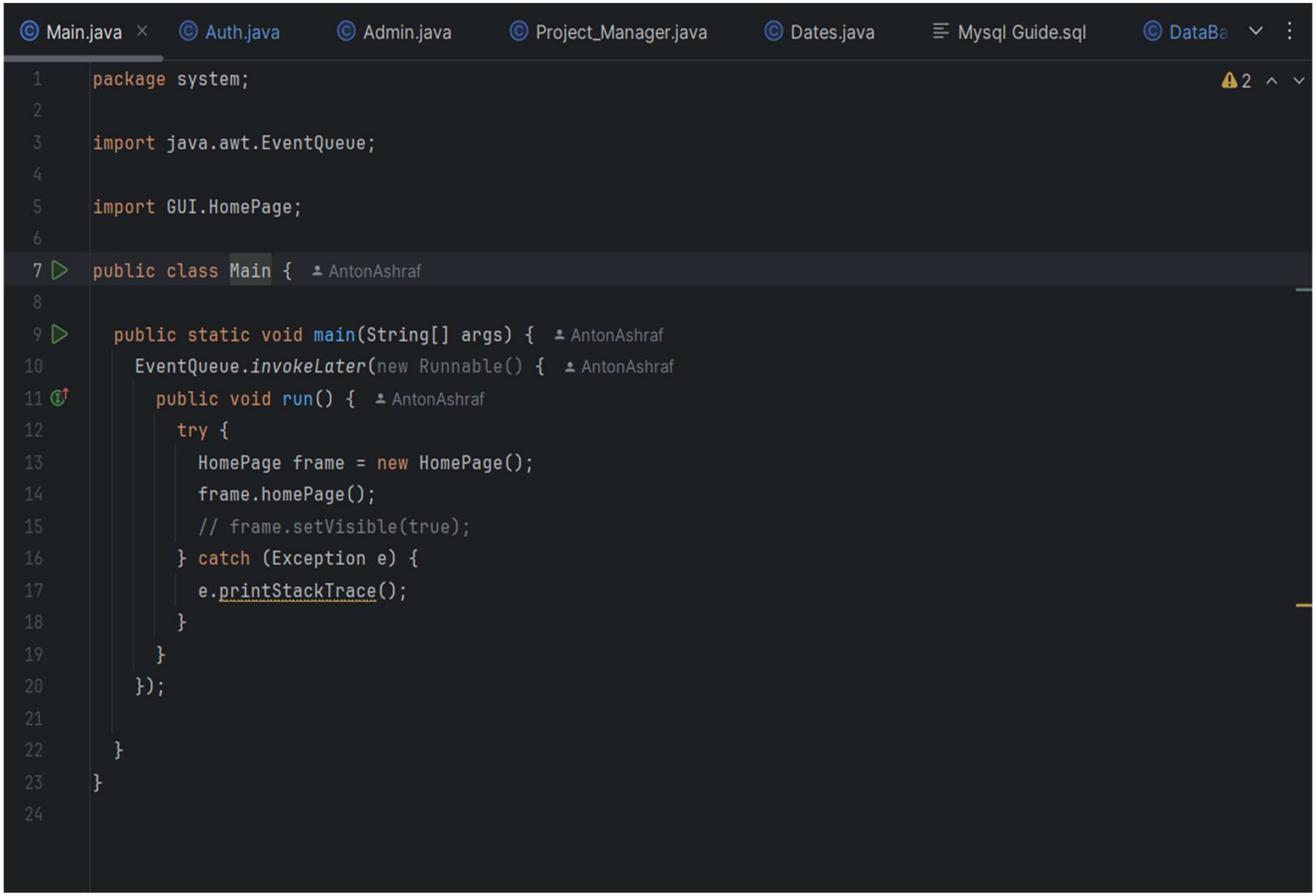
- **Bug:** bug_id, title, description, created_date, due_date, priority_id, severity_id, status_id, project_id.
- **User:** user_id, name, email, role (e.g., Reporter, Developer, Tester).
- **Project:** project_id, name, description.
- **Priority:** priority_id, name, description.
- **Severity:** severity_id, name, description.
- **Status:** status_id, name, description.

Database Design

Relationships:

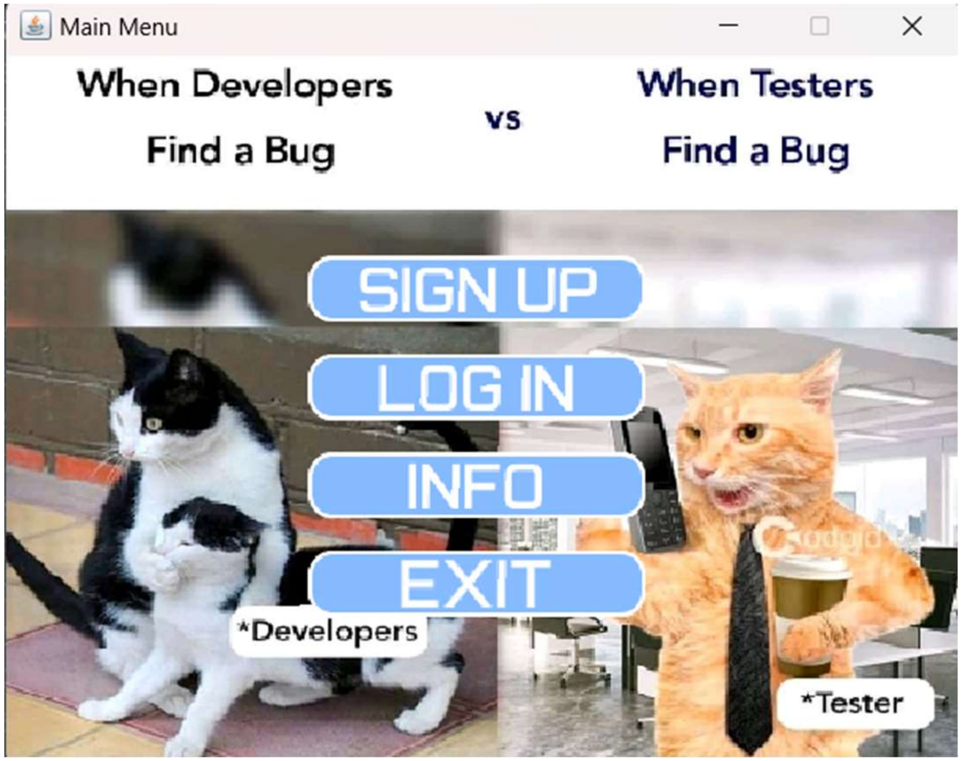
- **Many-to-many:** A bug can be assigned to multiple users (e.g., developers, testers), and a user can be assigned to multiple bugs.
- **One-to-many:** A bug belongs to one project, and a project can have many bugs.
- **One-to-many:** A bug has one priority, one severity, and one status.

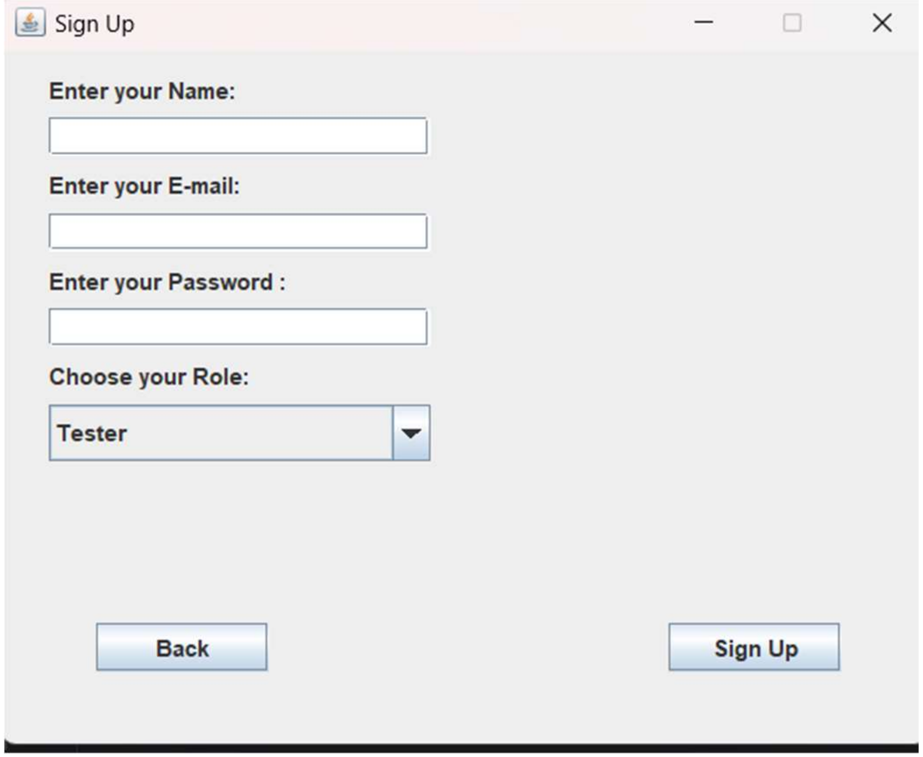
Sample Code



```
package system;

import java.awt.EventQueue;

import GUI.HomePage;

public class Main {

  public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
      public void run() {
        try {
          HomePage frame = new HomePage();
          frame.homePage();
          // frame.setVisible(true);
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });

  }
}
```

Sample Code





Enter your Name:

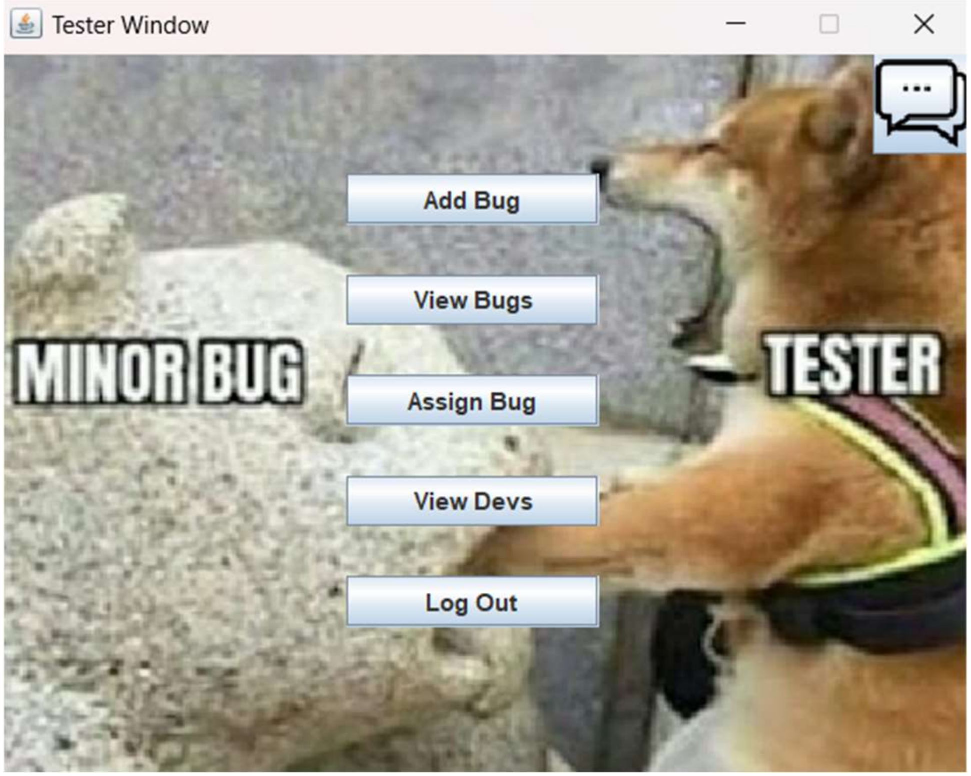
Enter your E-mail:

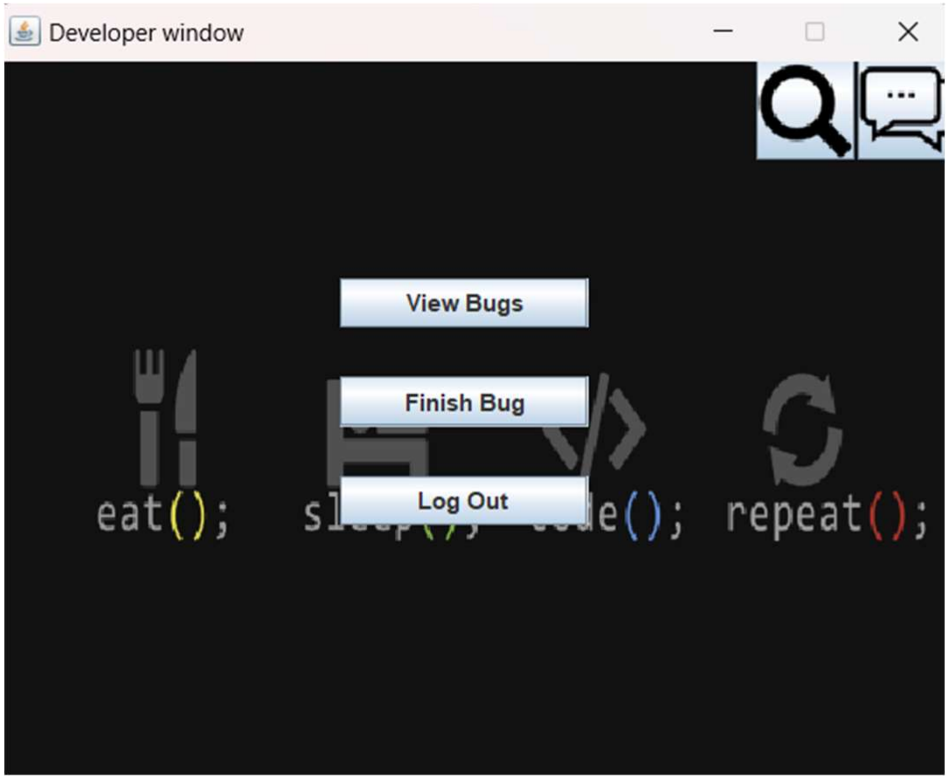
Enter your Password :

Choose your Role:

Tester

Sample Code


Tester Window
Add Bug
View Bugs
Assign Bug
View Devs
Log Out
MINOR BUG
TESTER


Developer window
View Bugs
Finish Bug
Log Out
eat();
repeat();

Conclusion & Future Enhancements

Conclusion:

- By combining these features and technologies, the Bug Tracking System project offers a robust, user-friendly, and efficient solution for bug tracking and management. It enhances software quality, productivity, and communication among stakeholders, making it an ideal choice for an object-oriented programming course project.

Future Enhancements:

AI and Machine Learning Integration

- **Smart Bug Triage**: Use machine learning algorithms to prioritize and assign bugs automatically based on historical data, such as bug severity, team member expertise, and resolution time.
- **Automated Root Cause Analysis (RCA)**: Implement AI tools to analyze bug reports and logs, identifying possible root causes and suggesting potential fixes.
- **Predictive Analysis for Bug Occurrence**: Based on code changes and historical patterns, AI could predict the likelihood of bugs occurring in specific areas, enabling proactive measures.

Reference

- Github
- Stack Overflow
- GeeksforGeeks
- YouTube

Thank You