# Bug Tracking System (Mini Jira)

**Overview**

The Bug Tracking System is a collaborative web application designed to help teams report, track, and resolve software defects effectively.  
It simulates how professional teams use tools like Jira to manage software quality, prioritize issues, and improve overall development workflow.

This project provides hands-on experience in managing issue lifecycles, applying Agile methodologies, and practicing QA/QE testing and automation techniques.

**Team Setup and Methodology**

• Team size: 3–4 members  
• Methodology: Agile Scrum  
• Tool: Jira (for backlog, sprint, and ticket management)  
• Deliverables per sprint: Defined user stories, implemented features, and test outcomes  
• **Roles**:  
o *Product Owner* – manages backlog and prioritization  
o *Scrum Master* – facilitates sprints and resolves blockers  
o *Developers / QA Engineers* – implement features and perform testing

Each team will maintain its own **Scrum board** on Jira, use **GitHub** for version control, and **document API** progress using Postman.

**System Architecture**

• Backend: Node.js + Express.js  
• Architecture: Layered Modular Architecture (Controller, Service, Repository)  
• Database: Microsoft SQL Server  
• Frontend: React.js + TypeScript  
• **Testing**:  
o Manual Tests  
o Unit & Integration: Jest  
o Performance: k6  
o End-to-End (E2E): Cypress  
• API Documentation: Postman  
• Version Control: Git & GitHub  
• Deployment (Optional): Render / Azure / AWS / Vercel

**Core Features**

**Authentication & Roles**

• User registration and login (JWT authentication)  
• Roles: Admin, Tester, Developer  
• Password encryption and role-based access control

**Bug Management**

• Report new bugs with details (title, description, severity, steps to reproduce)  
• Assign bugs to specific developers  
• Update bug status (Open, In Progress, Resolved, Closed)  
• Comment on bugs for discussion and clarification  
• Attachments (optional) for screenshots or supporting files

**Project Management**

• Create and manage multiple projects  
• View bugs by project or assigned developer  
• Filter bugs by status, priority, or reporter

**Admin Module**

• Manage users and roles  
• View system analytics (total bugs, open vs resolved, user activity)  
• Generate bug summary reports

**Database Design**

**Tables Overview**

1. Users Table
2. Projects Table
3. Bugs Table
4. Comments Table

**Backend Deliverables**

**Architecture**  
Follows the **Layered Modular Architecture** separating logic into controllers, services, and repositories for scalability and maintainability.

**API Endpoints**  
• User Authentication and Role Management  
• Project Management APIs  
• Bug CRUD Operations (Create, Read, Update, Delete)  
• Comment Management APIs  
• Reporting and Analytics APIs

**Frontend Deliverables**

• Pages:  
o Login / Register  
o Dashboard (bug overview and charts)  
o Project List / Details  
o Bug List / Create / Edit  
o User Profile  
o Admin Dashboard

• State Management: React Context API or Redux  
• UI: TailwindCSS / DaisyUI  
• Routing: React Router DOM  
• Testing: Cypress E2E scenarios covering bug reporting, assignment, and status updates

**Expected QA/QE Practice Areas**

• Writing and executing test cases and scenarios  
• Validating APIs manually using Postman  
• Implementing Jest unit and integration tests  
• Running performance benchmarks using k6  
• Conducting end-to-end testing with Cypress  
• Managing sprint testing cycles in Jira  
• Writing bug reports, test summaries, and status updates