UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

CTRL+X Error BUG TRACKING SYSTEM

Final Project Report

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

Ctrl_X Bug Tracking: Bug Tracking System Development Report

1. Project Overview

Problem Statement: Software development teams struggle with unorganized bug tracking, leading to delays and inefficiencies. This system provides a centralized and automated approach to manage bug reports, assign responsibilities, and track resolution progress in real-time.

Solution: Ctrl_X Error Bug Tracker is a comprehensive System with Bug Tracking and Chat capabilities. The system provides different roles (admin, developer, tester) with tailored access to features. The primary goal of this project is to develop a fully functional and scalable Bug Tracking System that helps software teams track, manage and resolve bugs efficiently. The system ensures structured bug reporting, role-based assignment, resolution tracking and automated notifications to relevant stakeholders.

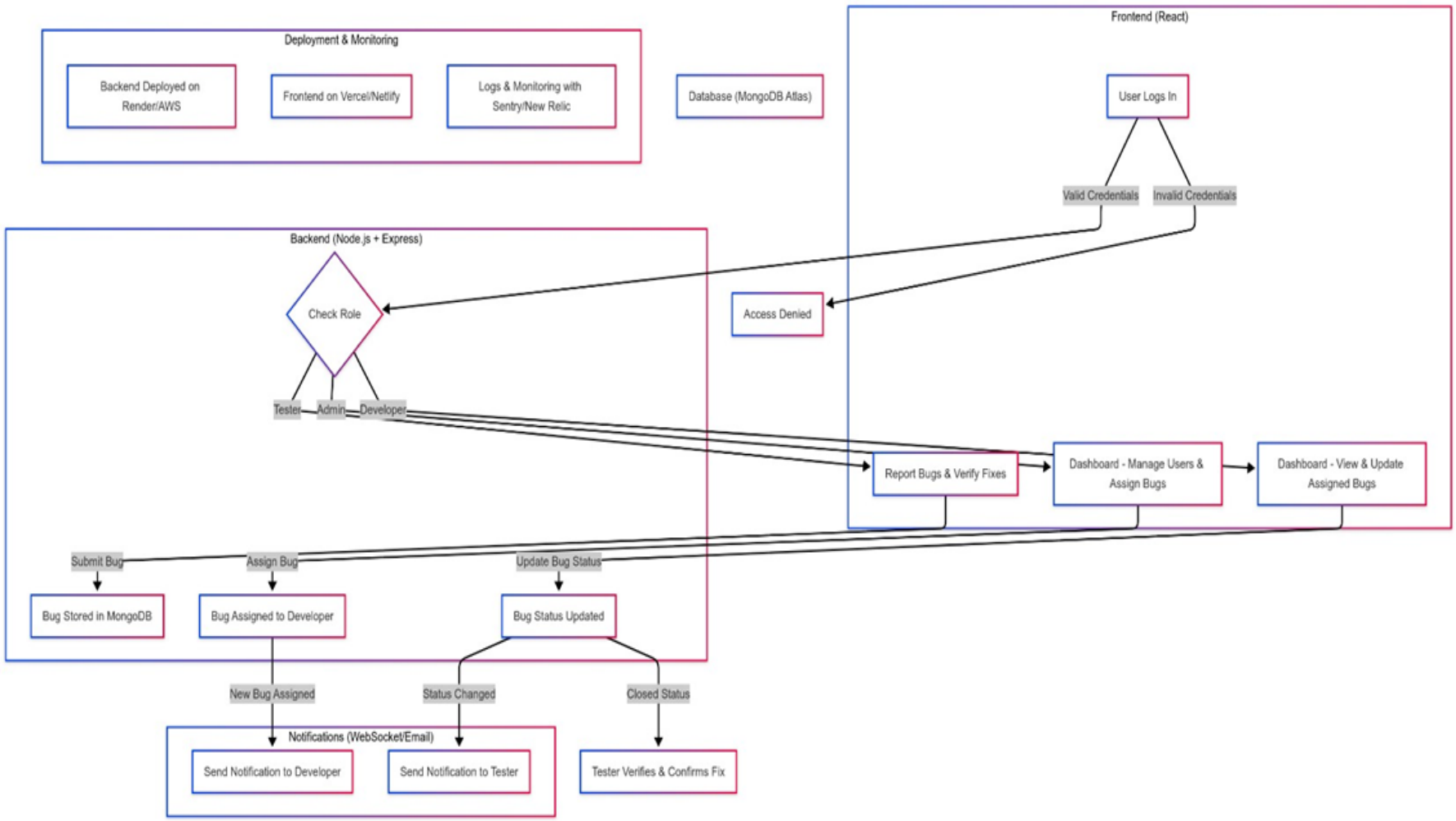
2. Software Development Lifecycle (SDLC)

2.1 Requirements Analysis

1. Gathered initial requirements through stakeholder interviews
2. Created user stories and defined acceptance criteria
3. Established project scope and identified key features
4. Prioritized requirements based on business value

2.2 Design

1. Created system architecture diagrams
2. Designed database schema and entity relationships
3. Developed wireframes and UI/UX mockups
4. Designed API endpoints and data flow


Deployment & Monitoring
Backend Deployed on Render/AWS
Frontend on Vercel/Netlify
Logs & Monitoring with Sentry/New Relic
Database (MongoDB Atlas)
Frontend (React)
User Logs In
Valid Credentials
Invalid Credentials
Backend (Node.js + Express)
Check Role
Access Denied
Tester
Admin
Developer
Report Bugs & Verify Fixes
Dashboard - Manage Users & Assign Bugs
Dashboard - View & Update Assigned Bugs
Submit Bug
Assign Bug
Update Bug Status
Bug Stored in MongoDB
Bug Assigned to Developer
Bug Status Updated
New Bug Assigned
Status Changed
Closed Status
Notifications (WebSocket/Email)
Send Notification to Developer
Send Notification to Tester
Tester Verifies & Confirms Fix

2.3 Implementation

1. Set up development environment and version control
2. Implemented frontend and backend components
3. Integrated third-party libraries
4. Created unit tests for critical components

2.4 Testing

1. Performed unit testing of individual components
2. Conducted integration testing of interconnected features
3. Carried out system testing of the entire application
4. Completed user acceptance testing with stakeholders

2.5 Deployment

1. Set up CI/CD pipeline for automated builds
2. Configured staging and production environments
3. Implemented database migration strategy
4. Created documentation for maintenance

2.6 Maintenance

1. Established bug tracking and resolution workflow
2. Created monitoring and logging systems
3. Planned for regular updates and feature enhancements
4. Documented procedures for troubleshooting common issues

3. Team Structure and Contributions

3.1 Team Members

Our team consisted of four members, each taking ownership of specific components while collaborating on system design and integration:

Team Member 1: Frontend Lead

1. Led the frontend architecture design
2. Implemented the React component structure
3. Developed the UI/UX following Tailwind CSS guidelines
4. Created responsive layouts for all screen sizes
5. Built the dashboard and visualization components

Team Member 2: Backend Lead

1. Designed the Express.js server architecture
2. Implemented RESTful API endpoints
3. Created MongoDB schemas and database logic
4. Set up authentication and authorization systems
5. Developed server-side validation

Team Member 3: Full Stack Developer (Features)

1. Implemented the bug tracker module
2. Built the bug tracking system
3. Created file upload and attachment functionality
4. Developed comment and notification systems
5. Handled cross-platform compatibility

Team Member 4: Full Stack Developer (Infrastructure)

1. Implemented real-time chat functionality
2. Set up CI/CD pipeline and deployment workflows
3. Created testing frameworks and test cases
4. Developed security features and data protection
5. Built reporting and analytics capabilities

3.2 Collaboration Methods

1. Alternate stand-up meetings to track progress
2. Weekly sprint planning and retrospectives
3. Shared code repositories with pull request reviews
4. Agile development methodology with 2-week sprints
5. Collaborative documentation using Markdown

Name	**Role**	**Contribution Details**
Naman Chanana	Backend & Full-Stack Lead	API development and testing, Auth & Notifications, Database design, API testing
Soumya Jain	Frontend Developer & UX	UI design implementation, React integration, UX
Smriti Walia	Testing, UI & Research	Research on application and use case, Bug testing, UI layout, documentation and Figma Design
Amulya Jain	Integration & Deployment	Connecting backend & frontend, middleware, Resolved and fixed dependencies and major issues along with Versioning of the software.

4. System Architecture

4.1 Frontend

1. **Technology**: React.js with Vite for fast development
2. **UI Framework**: Tailwind CSS for responsive design
3. **State Management**: React Context API for auth and user states
4. **Key Libraries**:
5. axios: API requests
6. react-router-dom: Routing
7. react-icons: UI icons
8. react-hot-toast: Notifications
9. date-fns: Date formatting

4.2 Backend

1. **Technology**: Node.js with Express.js
2. **Database**: MongoDB with Mongoose ODM
3. **Authentication**: JWT (JSON Web Tokens)
4. **File Storage**: Local file system with Multer

4.3 System Components

1. User Authentication System
2. Bug Tracking System
3. Real-time Chat
4. Dashboard & Reporting
5. Notification System

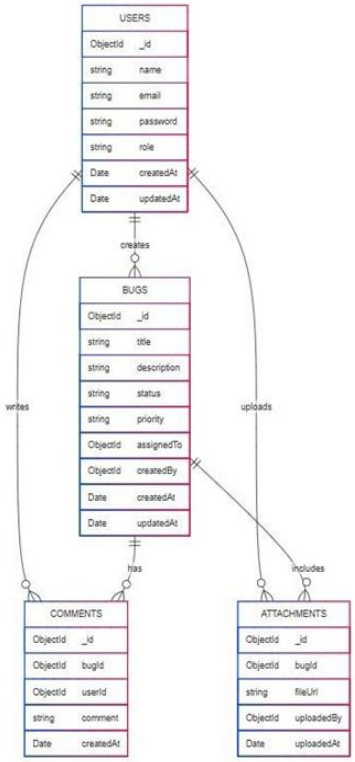
5. Database Design

5.1 Data Models

1. **User**: Stores user profiles, credentials, roles (admin, developer, tester)
2. **Task**: Work items with title, description, status, priority, assignments
3. **Bug**: Issues with severity, steps to reproduce, and tracking info
4. **Comment**: Feedback on tasks and bugs
5. **Chat/Message**: Communication between users
6. **Attachment**: File uploads for tasks and bugs

5.2 Relationships

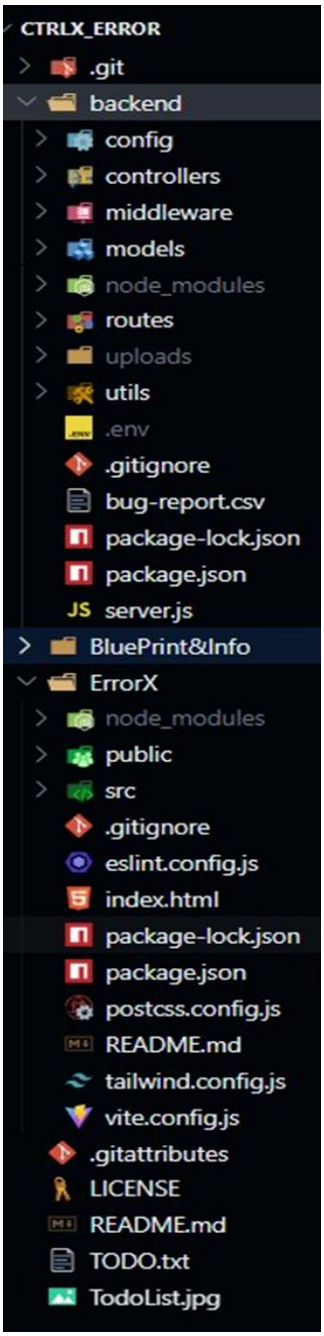
1. Tasks can have multiple assigned users
2. Bugs can be reported by users and assigned to others
3. Comments belong to tasks or bugs and are created by users
4. Messages belong to chat sessions between users



6. Development Process

6.1 Project Setup

1. Initialized project structure with separate frontend and backend and project description with README.md
2. Set up package.json with development scripts
3. Configured environment variables and. gitignore.



6.2 Backend Development

1. Created Express server
2. Implemented user authentication with JWT
3. Developed RESTful API endpoints for all resources
4. Added middleware for auth, file uploads, and error handling
5. Implemented controllers for business logic

6.3 Frontend Development

1. Set up React with Vite and routing
2. Created authentication flows (login, register)
3. Implemented dashboard layouts
4. Developed task and bug management interfaces
5. Added chat functionality
6. Created responsive UI with Tailwind CSS

7. Key Features

7.1 Bug Management

1. Create, assign, track and update Bug
2. Set priorities and deadlines
3. Track progress with checklists
4. Attach files to tasks
5. Add comments and updates

7.2 Bug Tracking

1. Log detailed bug reports with reproduction steps
2. Categorize by severity and type
3. Track bug status through resolution pipeline
4. Link related files and screenshots

7.3 Communication

1. Real-time chat between team members
2. File sharing in conversations
3. @mentions for user notifications
4. Dedicated channels for teams

7.4 User Experience

1. Responsive design for all devices
2. Dark/light theme switching
3. Dashboard with visual metrics
4. Role-based access control

8. Authentication & Authorization

8.1 User Authentication

1. JWT-based token system
2. Secure password storage with bcrypt
3. Login/logout functionality
4. Password reset capability

8.2 Role-Based Access

1. Admin: Full system access
2. Developer: bug tracker, bug fixes
3. Tester: Bug reporting, limited task access

9. Project Management

9.1 Agile Methodology

1. 2-week sprint cycles
2. User stories and story points for estimating complexity

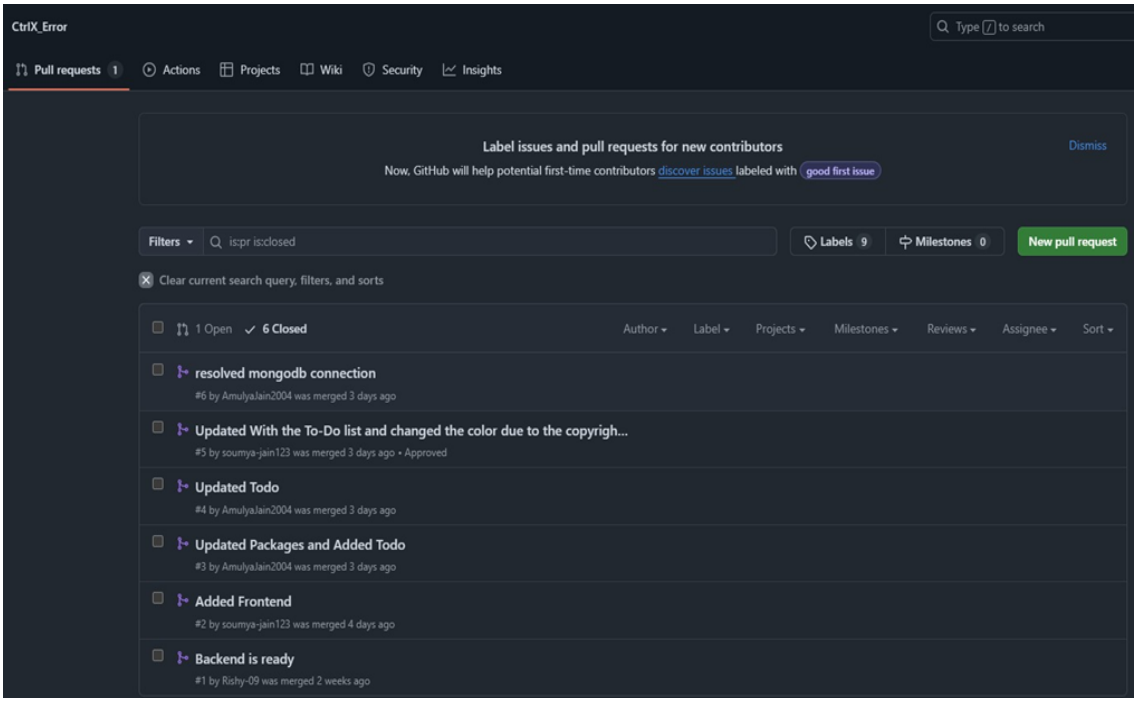
3. Kanban board for tracking task progress
4. Burndown charts for sprint performance monitoring

9.2 Version Control

1. Git for source code management
2. Feature branch workflow
3. Pull request reviews before merging
4. Semantic versioning for releases

9.3 Documentation

1. API documentation using Swagger
2. Code comments and JSDoc
3. User manuals and guides
4. Developer onboarding documentation



10. Implementation Challenges

10.1 Technical Challenges

1. Implementing real-time chat functionality
2. Managing file uploads and storage
3. Ensuring proper user permissions across features
4. Building responsive UI for all screen sizes

10.2 Solutions

1. Structured database models with clear relationships
2. Created middleware for authentication and file handling
3. Implemented thorough client-side validation
4. Used React Context for state management

11. Testing & Quality Assurance

11.1 Testing Approaches

1. Unit testing
2. Integration testing of API endpoints using Postman API software
3. End-to-end testing
4. Manual testing following test cases

11.2 Quality Metrics

1. Code coverage goals (>80%)
2. Lighthouse performance scores
3. Accessibility compliance
4. Cross-browser compatibility

12. Deployment

12.1 Infrastructure

1. Backend hosted on cloud server
2. Frontend deployed on static hosting
3. MongoDB Atlas for database
4. Environment variables for configuration

12.2 Deployment Process

1. Automated builds triggered by commits
2. Staging environment for pre-release testing
3. Blue-green deployment for zero downtime
4. Automated database backups

13. Future Improvements

1. Implement WebSockets for better real-time features
2. Add email notifications
3. Create mobile applications
4. Enhance reporting and analytics
5. Add CI/CD pipeline for automated testing and deployment

14. Conclusion

The Ctrl_X Error Bug Tracking system successfully delivers a complete task and bug management solution with integrated communication tools. Through effective teamwork and clear division of responsibilities, our four-member team created a robust application that maintains a clean architecture with separation of concerns between frontend and backend components.

Each team member contributed their expertise to specific areas while collaborating on the overall design and integration. This approach allowed us to develop a comprehensive system that addresses the complex requirements of bug tracker, bug tracking, and team communication.

The use of modern web technologies ensures a responsive and user-friendly experience, while the careful planning of data models and API endpoints provides a scalable foundation that can be extended with additional features in the future.