B.TECH PROJECT PRESENTATION

BUGPULSE

Real-Time Bug Reporting and Tracking

Students: Ujjwal Srivastava (2201220130123) & Umesh Mishra (2201220130124)

Guide: Er. Akshita

Information Technology (2026) | SRMCEM, Lucknow

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

Table of Contents

1 Introduction & Background

1.1 Background

In software engineering, effective bug tracking ensures reliability, stability, and usability of applications. Modern systems involve multiple contributors, diverse environments, and large user bases, making defect management increasingly challenging.

> Traditional platforms like Bugzilla, Jira, and GitHub Issues provide structured workflows but operate statically, with fragmented communication between stakeholders.

These systems lack real-time collaboration and feedback mechanisms critical for agile development. Recent advancements in bug triaging and automated classification highlight the potential of intelligent tools, yet a gap remains for a real-time, analytics-driven platform.

1.2 Relevance of BUGPULSE

BUGPULSE addresses these challenges through a web-based, real-time bug reporting solution. Users submit detailed reports with descriptions, logs, and screenshots, while developers receive instant notifications and status updates.

Key Features:

- Real-time interaction and transparency
- Collaborative features for teams
- Analytical insights for quality improvement
- Integration of modern web technologies

1.3 Role in Software Development

Real-time tracking accelerates development lifecycle by ensuring instant communication of bug reports and resolutions. It supports agile methodologies and CI/CD pipelines, enabling early detection of recurring problems and data-driven decision-making.

References: [3][4][5][6][7][8][9]

2 Literature Review & Research Gap

2.1 Bug Tracking Systems

Platforms like Bugzilla [9], Jira [8], and GitHub Issues [10] are industry standards for issue reporting and resolution. They improve transparency but operate as static repositories lacking real-time collaboration.

2.2 Bug Triaging & Assignment

Anvik et al. [3] demonstrated ML-based bug assignment reduces manual effort. Zimmermann et al. [4] showed prediction models identify defect-prone modules, improving efficiency.

2.3 Duplicate Detection

Sun et al. [5] introduced discriminative models for duplicate detection. Tian et al. [6] explored information retrieval-based approaches to identify semantically similar reports.

2.4 Deep Learning Applications

Chen et al. [7] demonstrated deep neural networks achieve higher accuracy in automated bug triaging compared to traditional ML methods. However, adoption in practical systems remains minimal.

2.5 Collaborative & Real-Time Tracking

Industry practices stress immediate communication between users, testers, and developers [8][9][10]. Most platforms lack real-time notifications, engagement analytics, or integrated feedback loops.

2.6 Identified Research Gap

While significant work exists in bug classification, duplicate detection, and triaging [3][5][6][7], most solutions remain research-oriented or limited to algorithmic improvements. Industry tools lack adaptability, transparency, and collaborative features. No existing solution fully integrates real-time bug tracking, intelligent insights, and user-centric collaboration, motivating the development of BUGPULSE.

References: [3][4][5][6][7][8][9][10]

3 Problem Definition & Objectives

3.1 Problem Statement

Modern software development relies on efficient bug tracking to maintain quality and user trust. However, existing platforms function as static repositories with:

1. **Limited Real-Time Interaction**
   Delays in status updates reduce responsiveness
2. **Fragmented Communication**
   Lack of seamless collaboration leads to misinterpretation
3. **Duplicate Bug Reports**
   Repeated submissions waste developer effort
4. **Inefficient Bug Triaging**
   Manual assignment slows resolution process
5. **Lack of Analytics**
   Minimal data-driven insights on critical bugs
6. **Poor User Experience**
   Platforms focus on developers, neglecting non-technical users

3.2 Project Objectives

- Objective 1: Develop intuitive web-based interface for bug reporting with descriptions, screenshots, and logs
- Objective 2: Provide real-time notifications and status updates for users and developers
- Objective 3: Integrate efficient bug triaging mechanisms for appropriate developer assignment
- Objective 4: Implement duplicate bug detection techniques to minimize redundancy
- Objective 5: Enhance collaboration among stakeholders through transparent communication workflow
- Objective 6: Generate analytics and insights on frequently reported or critical issues

Consolidated Goal: Design a real-time, collaborative, and analytics-driven bug reporting system that bridges the gap between traditional static trackers and modern agile development needs.

References: [3][4][5][6][7][8][9][10]

4 Proposed Methodology

4.1 Requirement Analysis

Analyze existing systems (Bugzilla, Jira, GitHub Issues) to identify gaps in interactivity, collaboration, and analytics.

Focus Areas: Ease of reporting, real-time updates, duplicate detection, usability for non-technical users

4.2 System Design

BUGPULSE follows a modular, layered architecture:

- Frontend Layer: Responsive web interface for bug submission
- Backend Layer: Service-oriented architecture for bug storage, notifications, triaging
- Database Layer: Structured repository for reports, users, analytics
- Real-Time Layer: Instant notifications and status updates
- Analytics Layer: Insights on issues, resolution time, severity trends

4.3 Module Implementation

- Bug Reporting Module: Submit reports with logs/screenshots
- Bug Management & Triaging: Assign, prioritize, resolve issues
- Duplicate Detection: IR and similarity detection techniques
- Notification & Collaboration: Real-time alerts and communication
- Analytics & Insights: Dashboards with metrics and trends

4.4 Testing & Evaluation

1. Unit Testing: Verify individual modules
2. Integration Testing: Ensure smooth data flow
3. User Testing: Pilot deployment evaluation
4. Performance Evaluation: Response time, notification latency
5. Comparative Evaluation: Benchmark against Jira/Bugzilla

Scalability: Designed for both small-scale academic projects and enterprise-level systems

References: [3][4][5][6][7][8][9][10]

5 Technologies & System Architecture

5.1 AI/ML Models

- Duplicate Detection: Similarity-based models with TensorFlow.js, Sentence Transformers
- Bug Triaging: Classification models via Brain.js, ml5.js
- Sentiment Analysis: Lexicon-based + TensorFlow.js models
- Recommendation: Content-based techniques for fix suggestions

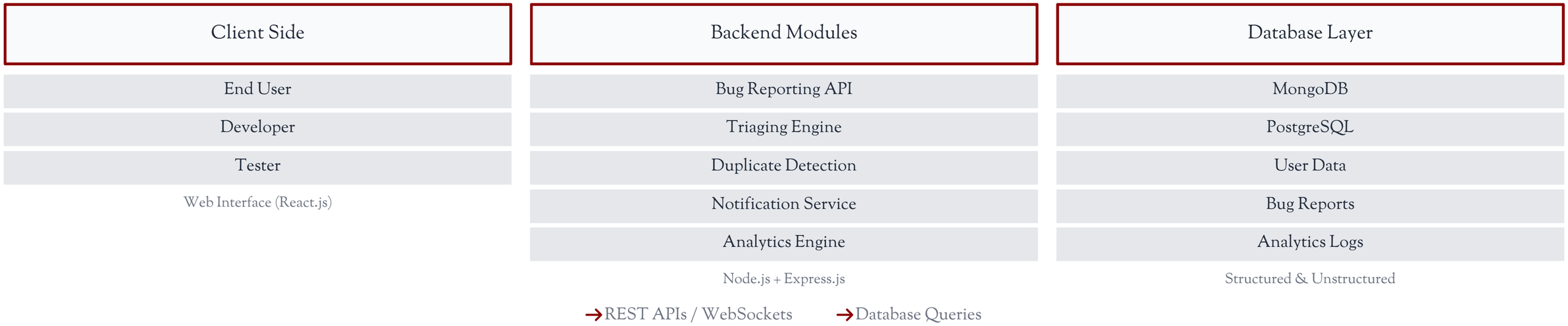
5.2 Real-Time Processing

- NLP: Parsing bug reports, extracting key details, contextual search
- Collaboration: Slack/MS Teams bot integration
- WebSockets/Socket.IO: Real-time communication
- WebRTC: Collaborative features

5.3 Backend & Frontend

- Backend: Node.js, Express.js REST APIs
- Databases: MongoDB, PostgreSQL/MySQL
- Frontend: React.js, Bootstrap/Tailwind CSS
- Visualization: D3.js, Chart.js for dashboards

System Architecture



Key Integration: AI modules (TensorFlow.js, Brain.js) connect via REST APIs to frontend and databases, enabling intelligent bug triaging and duplicate detection in real-time.

Technologies: TensorFlow.js, Brain.js, Node.js, React.js, MongoDB, WebSockets

6 Module Description

1 Bug Reporting Module

User-friendly interface for submitting bug reports with detailed descriptions, screenshots, and system logs. Real-time form validation ensures completeness.

Features: Drag-drop uploads, auto-save drafts, severity selection

2 Bug Management & Triaging

Classify, prioritize, and assign bugs using automated triaging. Supports severity levels (critical, high, medium, low) and tracks status (open, in progress, resolved, closed).

Features: Auto-assignment, priority scoring, status workflow

3 Duplicate Detection Module

ML-based similarity detection identifies semantically similar reports at submission time. Links or merges duplicates to eliminate redundancy.

Features: Vector embeddings, similarity scoring, auto-suggestions

4 Notification & Collaboration

Real-time communication with instant notifications on status changes. Collaboration features include commenting, tagging, and team mentions.

Features: WebSocket alerts, email notifications, in-app chat

5 Analytics & Insights

Dashboards displaying metrics: bugs reported/resolved/pending, average resolution time, frequently reported issues, trend analysis on severity and categories.

Features: Interactive charts, export reports, trend forecasting

6 User Management & Security

Role-based access control (user, tester, developer, admin) with secure JWT/OAuth2 authentication. Ensures data privacy and prevents unauthorized modifications.

Features: SSO integration, audit logs, permission matrix

Module Integration: All six modules communicate through REST APIs and WebSockets, ensuring seamless data flow and real-time synchronization across the platform.

References: [3][5][6][7][8][9]

7 Progress Report & Current Status

65%

Work Completed

Project Monitoring Report II

Project Title: BUGPULSE – Real-Time Bug Reporting and Tracking

Area: Artificial Intelligence and Web-Based Software Engineering

Type: Software Based (Web Application)

Team: Ujjwal Srivastava & Umesh Mishra

Guide: Er. Akshita

Coordinator: Er. Pooja Yadav

Completed Work

- **Priority Categorization**
  Implemented using TensorFlow library for automated bug prioritization
- **Basic UI Screens**
  Bug reporting interface and dashboard modules implemented
- **Backend Architecture**
  Established using Node.js and Express.js framework
- **Database Schema**
  Created for MongoDB and PostgreSQL integration
- **Core Modules**
  Initial implementation of bug reporting and triaging modules

Current Difficulties

Latency Issues
Observed during concurrent updates and real-time notifications. Requires optimization of WebSocket connections and database queries.

AI Module Performance
Performance optimization required for TensorFlow.js models across different browsers. Model loading and inference times need improvement.

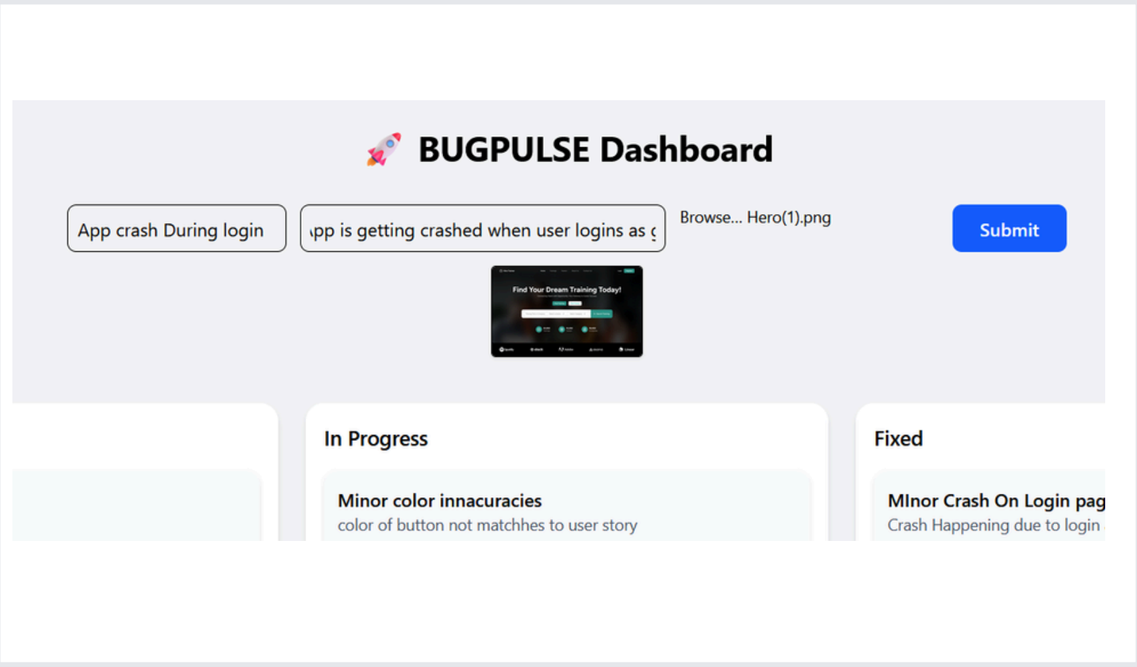
Next Steps: Optimize real-time communication layer, refine AI model performance, complete notification module, begin integration testing

Progress Report Date: February 2026 | Project Timeline: September 2025 – April 2026

8 System Demonstration

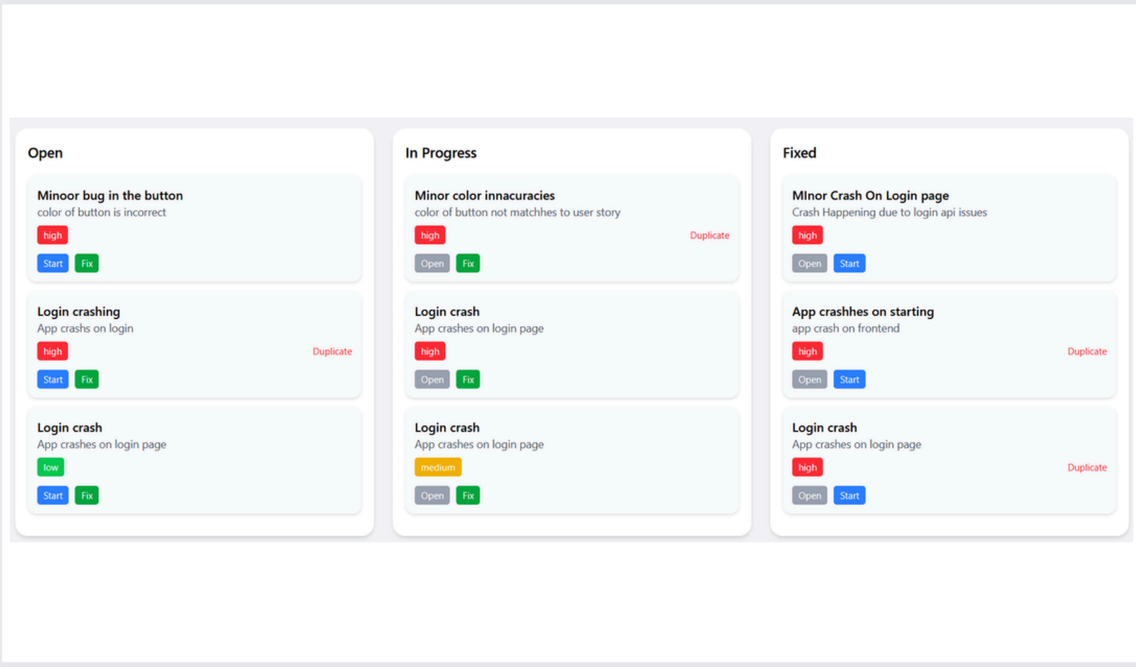
Interface Screenshots and Feature Previews

1 Bug Reporting Interface



Features: Intuitive form design, drag-drop screenshot upload, auto-validation, severity selection

2 Developer Dashboard



Features: Status tracking, priority view, assignment management, filter options

Note: Screenshots will be captured from the deployed system during the final presentation. Current UI implementation is 65% complete with core functionality operational.

Demo Screenshots: To be updated with actual system captures

9 Applications, Advantages & Limitations

9.1 Key Advantages

1. Real-Time Collaboration: Instant communication among stakeholders ensures faster resolution
2. Improved Transparency: Visibility into bug status keeps all parties informed
3. Reduced Redundancy: Duplicate detection prevents multiple submissions
4. Efficient Triaging: Automated assignment streamlines workflow
5. Enhanced UX: Intuitive interface for technical and non-technical users
6. Actionable Insights: Analytics help identify recurring issues
7. Scalability: Modular design for small to enterprise projects
8. Agile Support: Complements CI/CD pipelines with quick feedback

9.2 Application Domains

- Software Companies
- Open-Source Projects
- Academic Projects
- Enterprise IT Teams
- Agile Environments
- QA Teams
- Customer-Facing Platforms

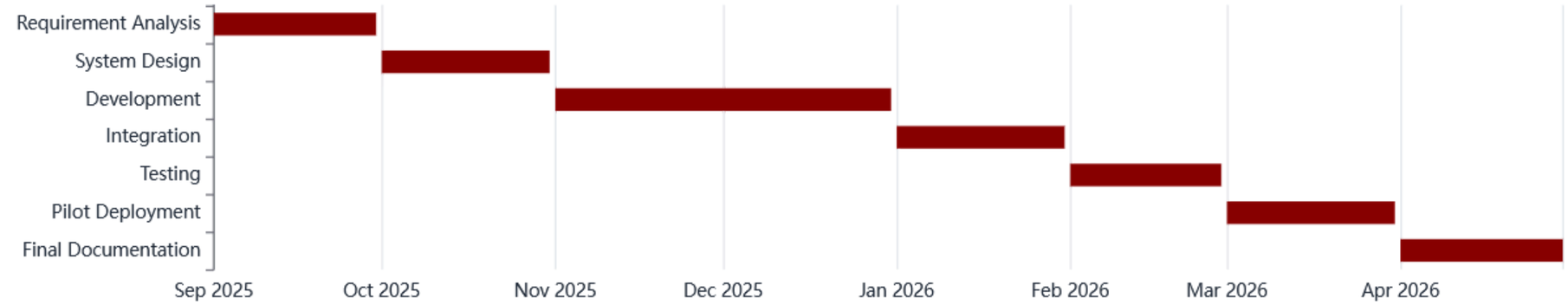
9.3 Limitations

- Internet Dependency: Real-time features require stable connection
- Training Requirement: Users need orientation for advanced features
- Resource Intensive: Large projects may need additional server resources
- Integration Complexity: Legacy system linkage may require customization
- Algorithm Accuracy: Duplicate detection depends on model precision

References: [3][5][6][7][8][9]

10 Project Timeline & Conclusion

8-Month Project Timeline (Sep 2025 – Apr 2026)



Phase-wise Work Plan

Sep 2025: Requirement Analysis & Literature Review

Oct 2025: System Design & Architecture

Nov–Dec 2025: Module Development (Core System)

Jan 2026: System Integration

Feb 2026: Testing & Refinement

Mar 2026: Pilot Deployment & User Testing

Apr 2026: Final Documentation & Completion

Key Takeaways

- BUGPULSE addresses critical gaps in existing bug tracking systems by integrating real-time collaboration, AI-powered triaging, and analytics-driven insights.
- The system accelerates bug resolution, fosters transparency, and enhances software quality in modern development environments.
- Current progress at 65% completion with core modules operational and ongoing optimization.
- Designed for scalability across small academic projects to enterprise-level systems.

Thank You

Questions & Discussion

Presented by: Ujjwal Srivastava & Umesh Mishra | Guide: Er. Akshita | SRMCEM, Lucknow