*International Journal of Research in Engineering and Science (IJRES)*
*ISSN (Online): 2320-9364, ISSN (Print): 2320-9356*
*www.ijres.org Volume 13 Issue 11 ǁ 2025.*

# Intelligent Bug Prioritization & Assignment Tool

[*1]Ms. Priyadarshini Badgujar (Project Guide),
[2]Krishna Bhattu, [3]Harshwardhan Jadhav,
[4]Harsh Josolkar, [5]Yash Kapure.

*Department of Computer Science & Design Engineering, University of Mumbai, NHITM, Thane (W) – 400615*

***Abstract***
*Efficient bug management is essential for maintaining software quality and timely project delivery. This research presents an Intelligent Bug Prioritization and Assignment Tool that leverages artificial intelligence to automate the analysis and assignment of reported bugs. The system predicts bug severity, estimates resolution time, and recommends suitable developers based on domain relevance. Supa base, integrated as a secure cloud database, supports real-time data synchronization and role-based access. The proposed model aims to minimize manual intervention, improve accuracy in prioritization, and enhance coordination among testers, developers, and managers, thereby optimizing the overall software maintenance process.*

***Keywords:*** *Artificial Intelligence, Bug Prioritization, Automated Assignment, Software Maintenance*

---

Date of Submission: xx-xx-xxxx Date of acceptance: xx-xx-xxxx

---

## I. INTRODUCTION

In the modern software development landscape, the continuous growth in application complexity and the rising expectations of users have made efficient bug management an essential aspect of ensuring software quality and reliability. Regardless of the development approach or methodology, every software product experiences bugs at different stages of its lifecycle. Traditionally, these bug reports are reviewed and prioritized manually by developers or project managers—a process that is often slow, inconsistent.

As software projects scale and development teams become more distributed, the number of daily bug reports has increased exponentially. Manual triaging under such conditions can result in redundant efforts, uneven workload distribution, and occasional misclassification of bug severity. Furthermore, determining the criticality of a bug requires a comprehensive understanding of the system and its impact on users, which may not always be available to all team members.

To address these limitations, the Intelligent Bug Prioritization and Assignment Tool **leverages** Artificial Intelligence and Natural Language Processing to automate and optimize the bug triage process. The system analyses bug descriptions, extracts key contextual features, predicts severity levels such as *Critical, High, Medium,* or *Low*, and assigns the most suitable developer based on their expertise and past experience. Additionally, it estimates the resolution time for each issue, supporting more accurate project planning and resource allocation.

By automating bug triage, the proposed model minimizes manual intervention, eliminates bias, and enhances the precision and consistency of bug prioritization. Through the integration of predictive analytics and a user-friendly interface, the system enables faster, data-driven decision-making—leading to improved team productivity, reduced turnaround time, and enhanced overall software reliability.

### 1.1.1 Limitation / Existing System / Research Gap

Traditional bug triage in software development is a manual and time-consuming process dependent on human judgment, often leading to inconsistencies and delays in issue resolution. With the growing complexity of projects and increasing bug reports, manual prioritization struggles to remain efficient. Existing automated methods usually focus on isolated tasks like severity prediction or developer assignment and fail to provide an integrated solution.

The proposed Intelligent Bug Prioritization and Assignment Tool uses Artificial Intelligence (AI) and Natural Language Processing (NLP) to automate bug analysis, predict severity, estimate resolution time, and assign

developers effectively. This approach ensures higher accuracy, faster triage, and improved software maintenance efficiency.

### 1.1.2 Objectives

The main goal of this project is to develop an **Intelligent Bug Prioritization and Assignment Tool** that automates bug triage using AI and NLP techniques.

Specific objectives include:

1. To automatically analyze bug reports using NLP.
2. To predict bug severity levels (*Critical, High, Medium, Low*).
3. To recommend suitable developers based on expertise and history.
4. To estimate bug resolution time for better planning.
5. To minimize manual effort and improve accuracy in bug management.
6. To design a user-friendly, scalable, and integrated platform.

## I.2 Proposed System

### 1.2.1 Analysis/ Framework/ Algorithm

1. Tester logs a bug report (title, description, file).
2. AI module analyses content → returns Severity, ETA, Fix Suggestion, and Suggested Domain.
3. **Manager** reviews AI suggestions and assigns the bug to a developer.
4. **Developer** receives assignment, updates status, and resolves the issue.
5. **Supa base** manages roles, authentication, and storage.

### 1.2.2 Tools and Technologies

| Layer | Technology |
|---|---|
| **Frontend** | React 18 + Vite + Tailwind CSS |
| **Backend** | Flask (Python) |
| **AI** | OpenAI GPT API, Google Gemini |
| **Database** | Supabase (PostgreSQL) |
| **Authentication** | Supabase Auth |
| **Hosting** | Render / Vercel / Supabase Cloud |

**Table 1.1 - Tools and Technologies**

### 1.2.3 Methodology

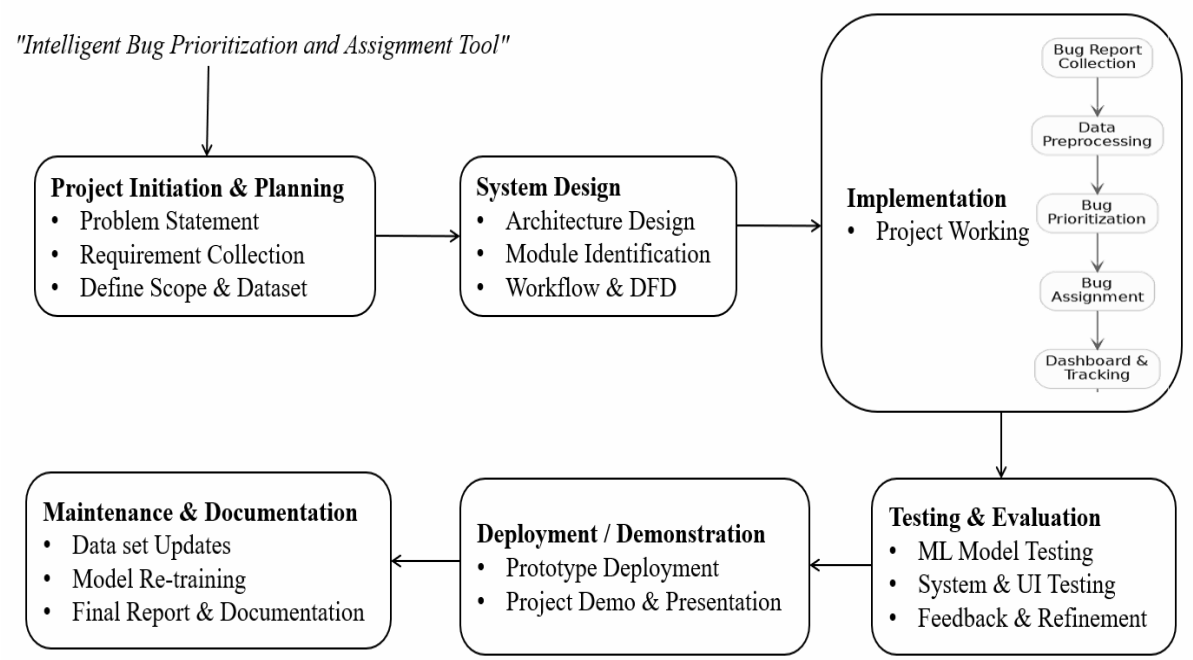

**Figure 1.1 – Methodology/ Project Lifecycle**

- **Project Initiation & Planning:**
  Define the problem statement, gather requirements, and decide the project's scope and dataset.
- **System Design:**
  Create the system architecture, identify modules, and design workflow diagrams and DFDs.
- **Implementation:**
  The main development stage — where modules like *Bug Report Collection, Preprocessing, Prioritization, Assignment,* and *Dashboard Tracking* are built and integrated.
- **Testing & Evaluation:**
  Perform ML model testing, UI testing, and gather feedback for improvement.
- **Deployment / Demonstration:**
  Deploy the prototype, and present or demonstrate the project to stakeholders.
- **Maintenance & Documentation:**
  Update datasets, retrain AI models if needed, and prepare the final documentation and report.

### 1.2.4 Design Details

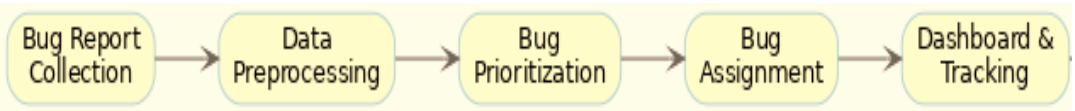


**Figure 1.2 – System Design Workflow**

**1. Bug Report Collection**
- Collect bug reports from issue trackers. (Pre - defined bug set)
- Extract details: bug ID, severity, description, reporter info, etc.

**2. Data Preprocessing**
- Clean raw bug reports (remove duplicates, irrelevant data).
- Apply NLP techniques to process text (tokenization, stop-word removal).
- Feature extraction for ML model input.

**3. Bug Prioritization**
- ML/AI model predicts **bug severity and priority level**.
- Helps in reducing resolution time and focusing on critical issues.

**4. Bug Assignment**
- Assign bugs automatically to the **most suitable developer**.
- Based on expertise, workload, and past bug-fix history.

**5. Dashboard & Tracking**
- Provide a **visual dashboard** for bug status.
- Track: Open, In-progress, Resolved, Closed bugs.

### 1.2.5 Hardware and Software Setup

**Software Requirements:**
- Operating System: Windows 10, macOS, or Linux.
- Programming Languages: Python 3.11 and JavaScript (ES6+).
- Frameworks: Flask (backend) and React + Vite (frontend).
- Database: Supa base (PostgreSQL).
- AI API: OpenAI GPT-3.5-Turbo, Google Gemini.
- Tools and Libraries: Pandas, NumPy, Tailwind CSS, Axios.
- IDEs and Testing Tools: Visual Studio Code, Postman.
- Browser: Google Chrome (latest version).

**Hardware Requirements:**
- Minimum: Intel Core i3 processor, 4 GB RAM, 128 GB SSD, 5 Mbps internet connection.

- Recommended: Intel Core i5 or higher, 8 GB RAM, 256 GB SSD, 10 Mbps broadband connection.

## II. RESULT AND DISCUSSION

### 2.1.1 Implementation Plan

**Phase 1: System Design and Backend Development**

- *Requirement Analysis:*
  – Collected system requirements from testers, developers, and project managers.
  – Defined the complete workflow for bug reporting, prioritization, and automated assignment.
- *Database Design:*
  – Structured the database schema in Supa base (PostgreSQL), establishing relationships between users, bugs, and task assignments.
  – Implemented secure, role-based access control and real-time data synchronization.
- *Backend Development:*
  – Built Flask-based APIs for managing bug submission, retrieval, and assignment processes.
  – Integrated an AI-driven module utilizing the GPT model for severity prediction and domain specific task allocation.
  – Added user authentication and role-based permissions for secure system access.

**Phase 2: Frontend Development and AI Integration**

- *Frontend Interface Design:*
  – Developed a responsive web dashboard using React with Vite and Tailwind CSS.
  – Designed dedicated panels for Testers (reporting bugs), Developers (tracking assigned tasks), and Managers (monitoring overall progress).
- *AI Module Integration:*
  – Connected the OpenAI API to generate severity levels and recommend appropriate developers.
  – Automated prioritization based on predicted severity scores and estimated resolution times.
- *Testing and Validation:*
  – Conducted unit and integration testing through Postman and manual verification.
  – Evaluated AI model accuracy and measured system performance metrics.

**Phase 3: Deployment and Evaluation**

- *Cloud Deployment:*
  – Deployed the backend on a cloud environment with Supa base as the database service.
  – Hosted the frontend for browser-based access, ensuring smooth user interaction.
- *Performance Evaluation:*
  – Monitored throughput, latency, and user feedback to assess overall performance.
  – Gathered insights from testers and managers to refine AI-based recommendations.
- *Final Optimization:*
  – Enhanced API response time and minimized database delays.
  – Improved UI components and implemented audit logs to ensure system transparency.

### II.2 Result

The developed system successfully automated the bug triage process by accurately predicting severity levels and assigning issues to suitable developers using AI and NLP techniques. The integration of Flask, Supa base, and React provided a seamless and responsive platform for testers, developers, and managers. Performance evaluation showed improved accuracy, faster response times, and reduced manual effort in bug management. User feedback confirmed enhanced efficiency and transparency in tracking and resolving software issues. Overall, the tool proved effective in streamlining bug prioritization and assignment in real-world scenarios.

### II.2.1 Results/ Outputs

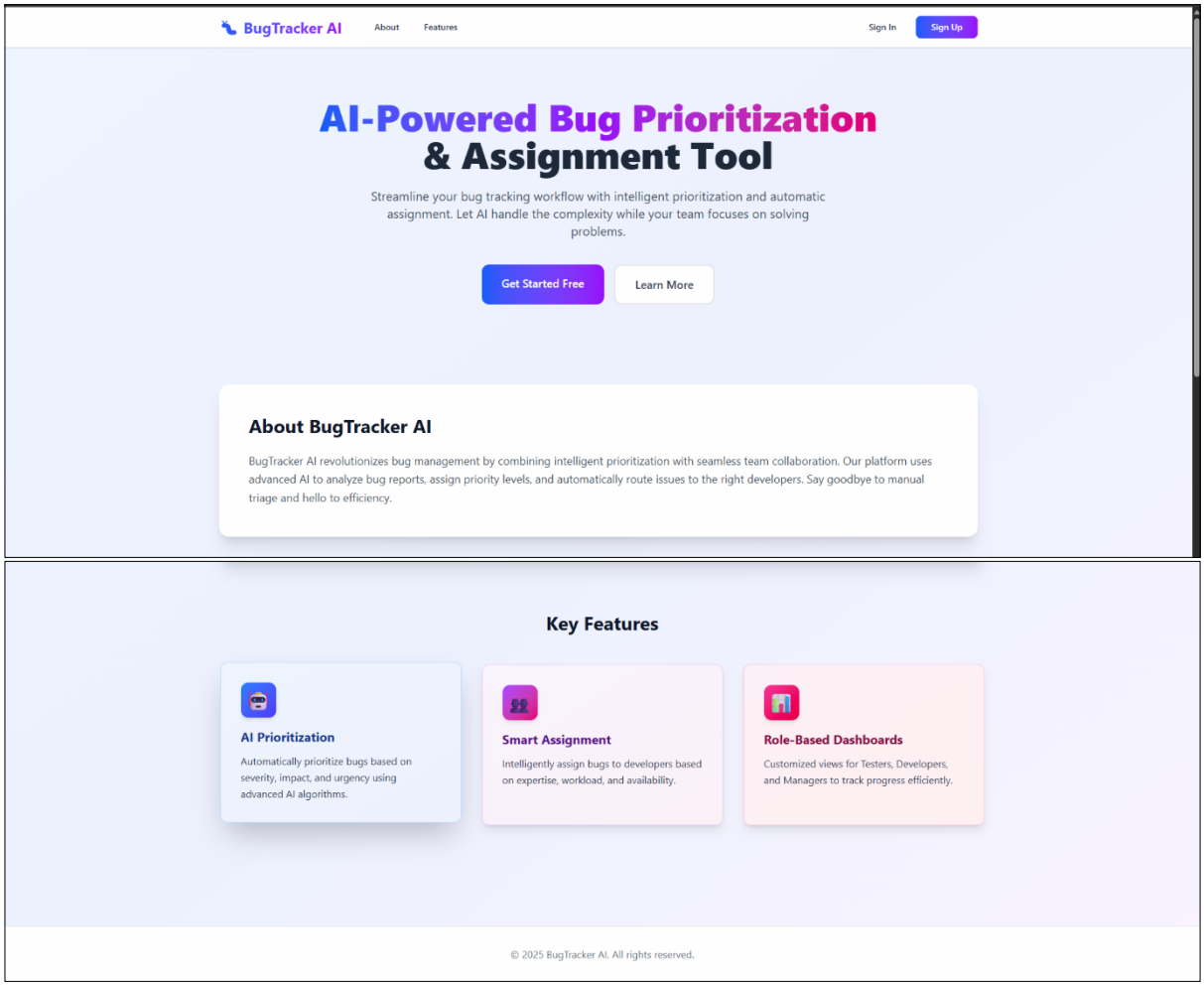


**Figure. 2.1 – Landing Page**

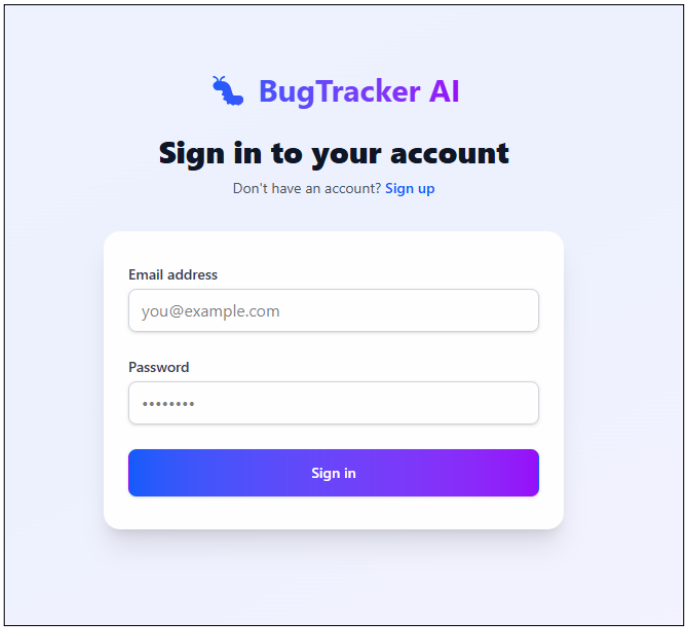


**Figure. 2.2 – Sign in Page**

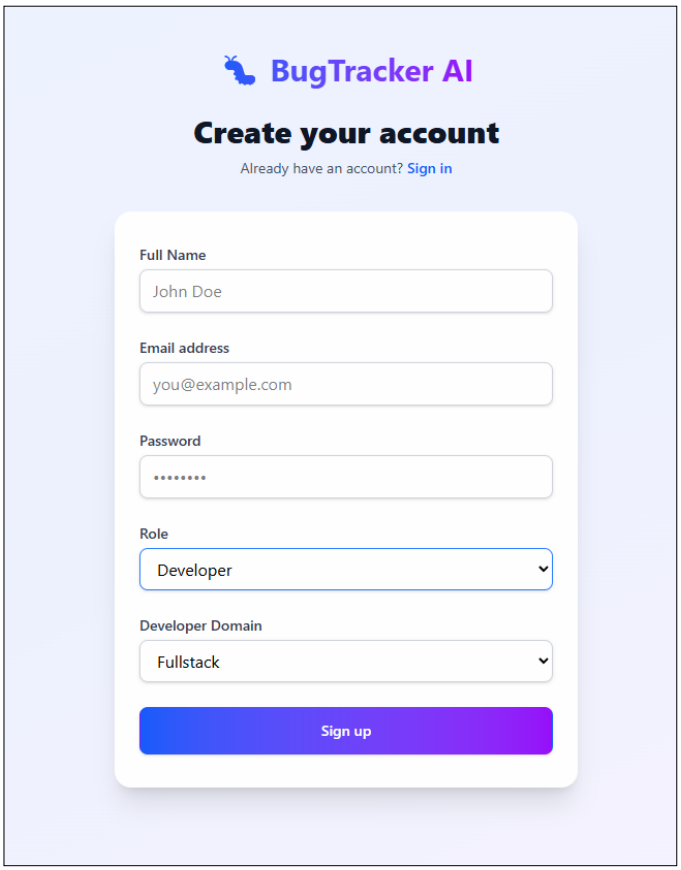


**Figure. 2.3 – Sign Up Page**

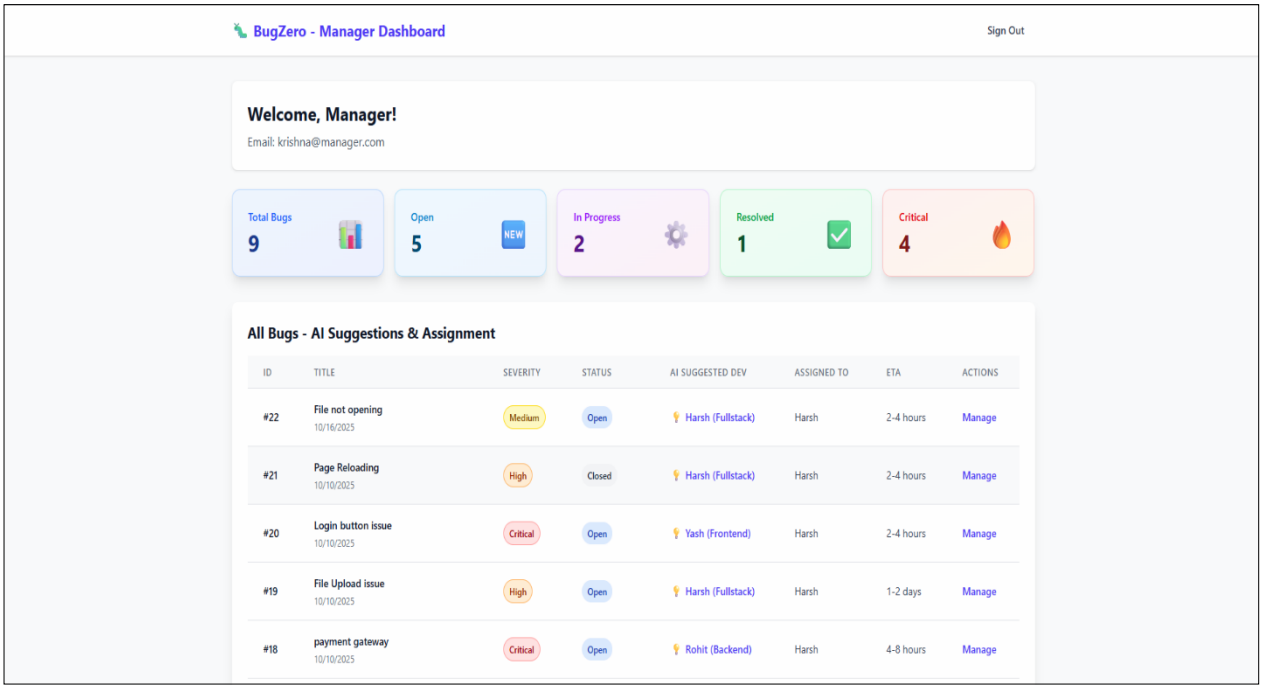


**Figure. 2.4 – Manager Dashboard**

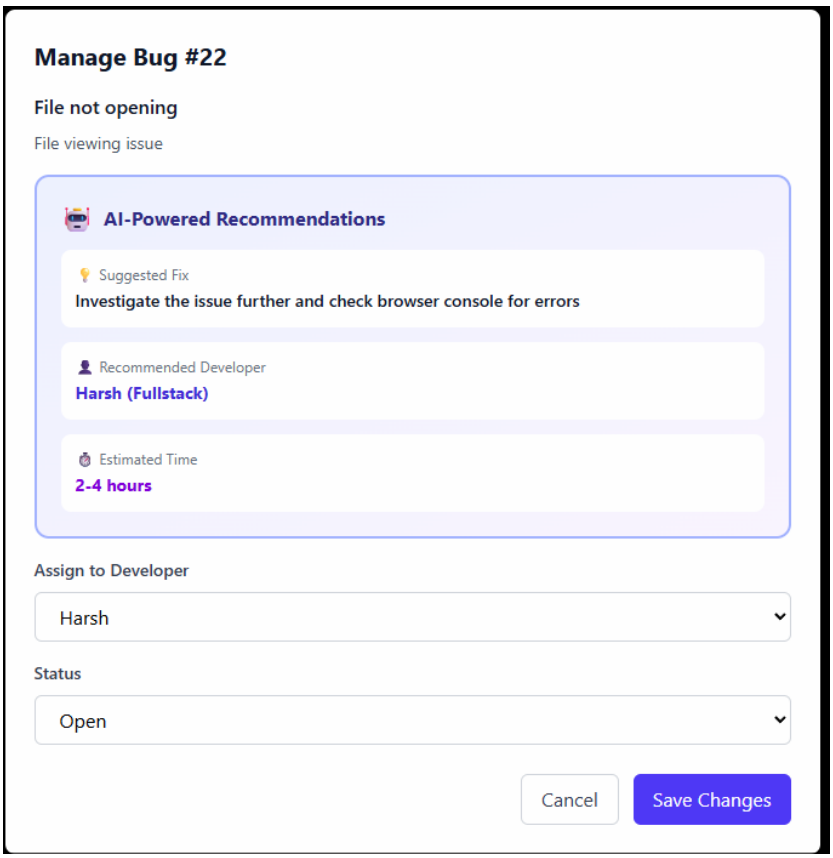


**Figure. 2.5 – Manager Action**

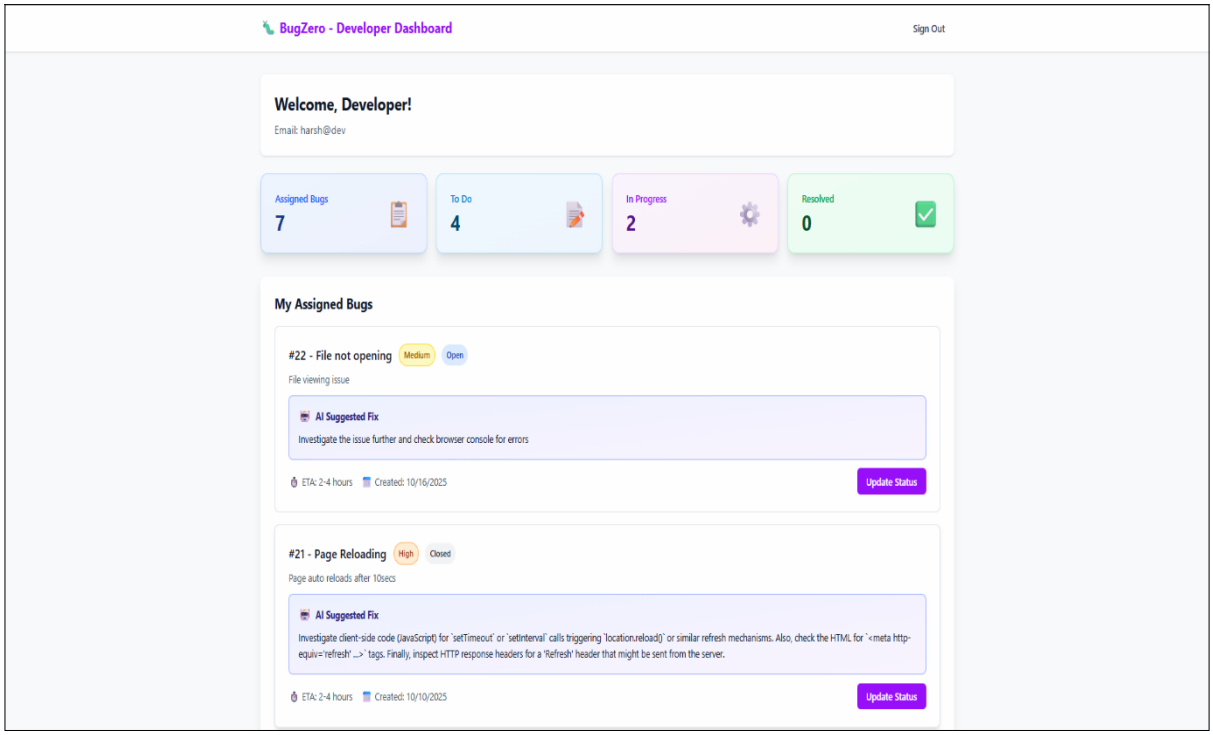


**Figure. 2.6 – Developer Dashboard**

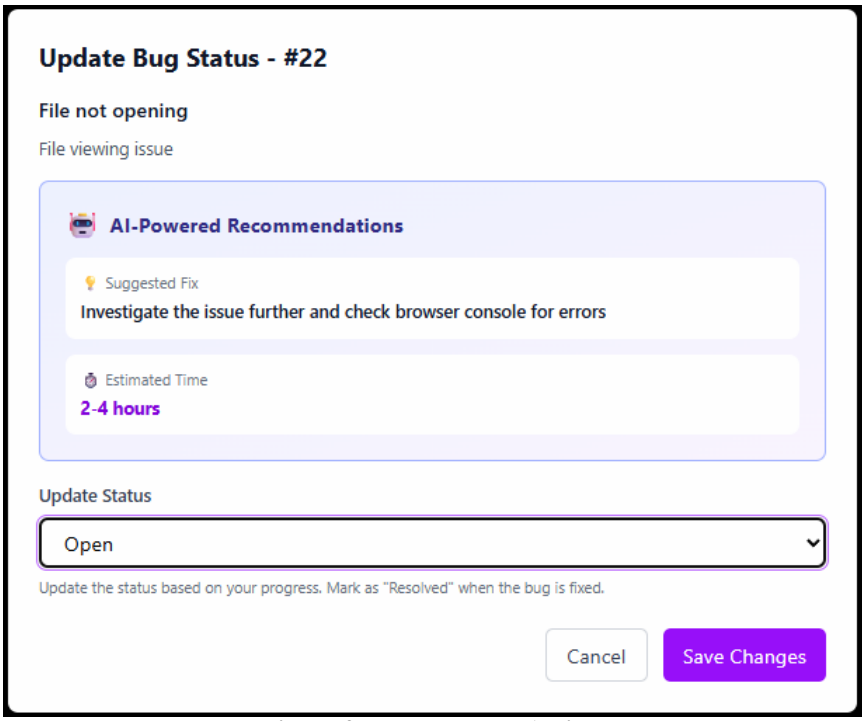


**Figure 2.7 – Developer Action**

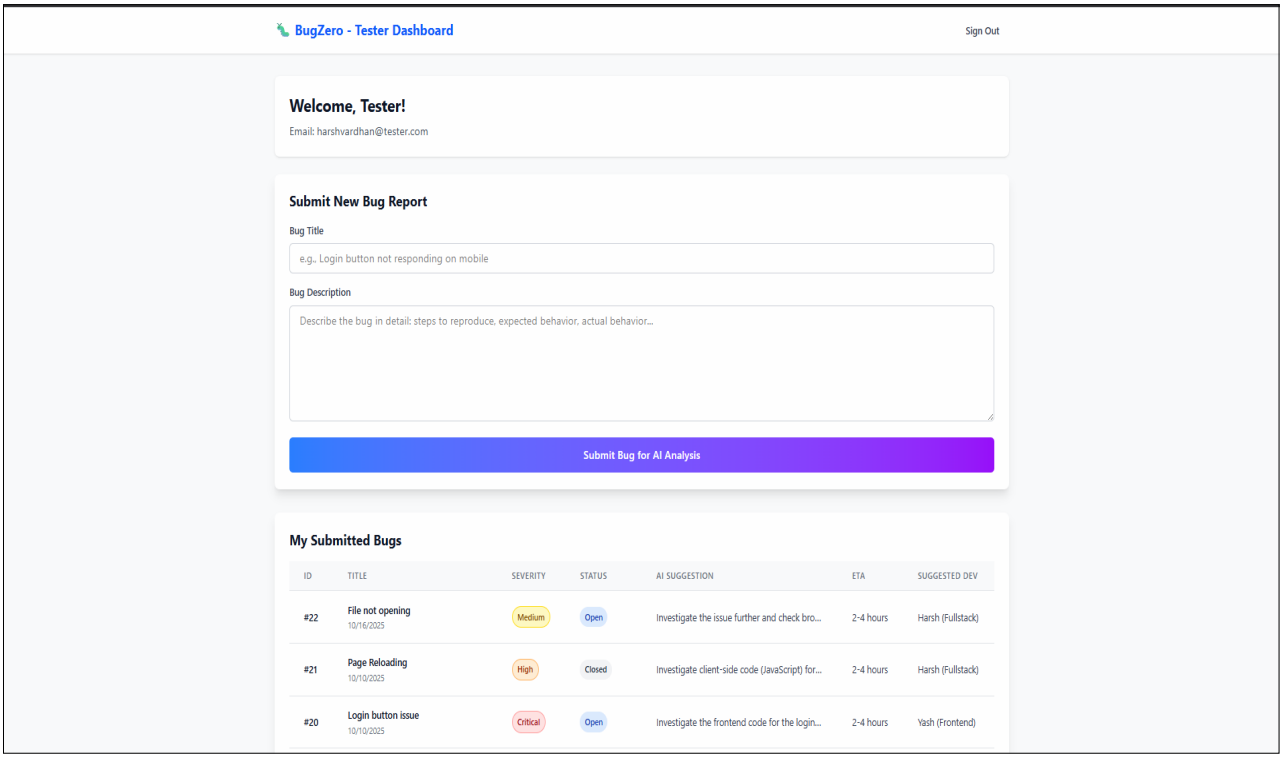


**Figure 2.8 – Tester Dashboard**

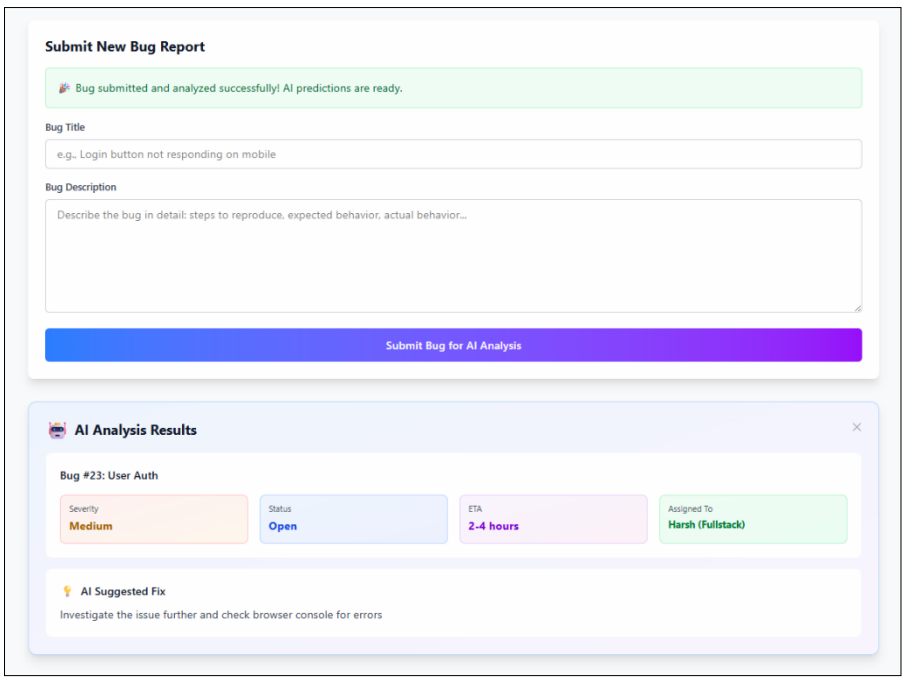


**Figure 2.9 – Tester Bug Submission**

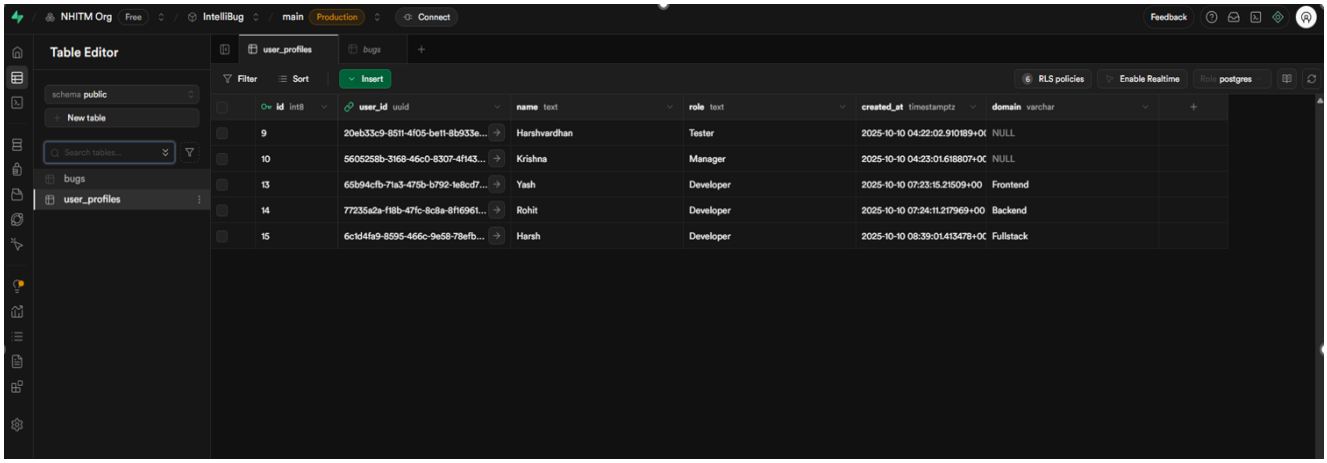


**Figure 2.10 – Supa base *User Profile Database***

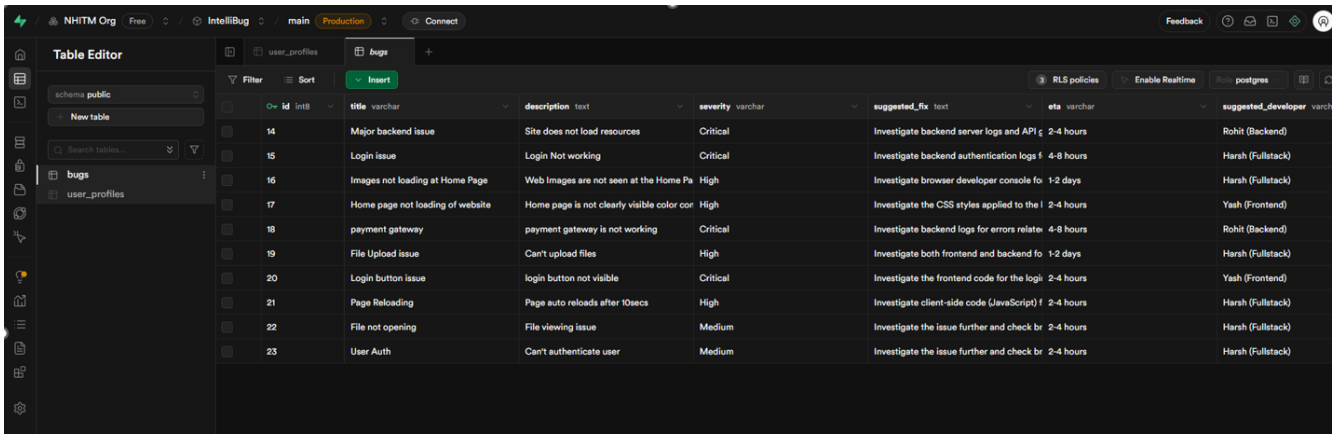


**Figure 2.11 - *Supa base Bugs Database***

## III. CONCLUSION

The Intelligent Bug Prioritization and Assignment Tool effectively automate the bug management workflow through AI-driven analysis. It reduces manual intervention by predicting bug severity, estimating resolution duration, and suggesting the most appropriate developer for each task. The system improves coordination among testers, developers, and managers, enabling faster and more efficient issue resolution. This approach highlights the potential of AI to enhance software maintenance, optimize team productivity, and elevate overall software quality in practical development environments.

## IV. REFERENCES

[1]. M. Borg, L. Jonsson, E. Engström, B. Bartalos, and A. Szabó, "Adopting automated bug assignment in practice - A longitudinal case study at Ericsson," *Empirical Software Engineering*, vol. 29, no. 5, 2024.

[2]. S. Kumar, R. Kaur, and V. Gupta, "AI-driven bug prioritization using NLP and deep learning models," *IEEE Access*, vol. 12, pp. 14567–14579, 2023.

[3]. H. Lin, T. Li, and J. Chen, "Automated issue classification and developer assignment with transformer - based models," *Journal of Systems and Software*, vol. 195, pp. 111584, 2023.

[4]. A. Sharma and P. Patel, "Machine learning-based intelligent bug triaging system for large-scale software projects," *Procedia Computer Science*, vol. 218, pp. 1030–1040, 2023.

[5]. M. J. Lee, D. H. Kim, and S. Park, "Enhanced bug severity prediction using hybrid ensemble techniques," IEEE Transactions on Software Engineering, vol. 49, no. 2, pp. 482–495, 2022.

[6]. Y. Zhang, X. Wu, and L. Wang, "Improving bug report classification with attention-based neural networks," Information and Software Technology, vol. 139, pp. 106690, 2021.

[7]. P. Thung, D. Lo, and L. Jiang, "Automatic bug triaging using text mining and topic modeling," Empirical Software Engineering, vol. 26, no. 4, pp. 2459–2483, 2020.
[8]. A. Sureka, "Mining bug repositories for developer recommendation and assignment," Journal of Software: Evolution and Process, vol. 29, no. 6, 2017.
[9]. C. Tian, M. Reformat, and Y. Zou, "An intelligent approach for developer recommendation in bug fixing," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, no. 10, pp. 1423 – 1436, 2016.
[10]. G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 111–120, 2011.