IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

PIET MAINTENANCE WEB APP

[1]Shikha Gautam, [2]Rishabh Singh Sisodiya, [3]Nikhil Singh Kotar, [4]Manohar Kumar

[1]Deputy HOD, [2]Student, [3]Student, [4]Student

[1]Department of Computer Science and Engineering,

[1]Poornima Institute of Engineering and Technology, Jaipur, India

Abstract: The PIET Maintenance Web App is a web-based complaint management system designed to streamline the process of reporting, tracking, and resolving infrastructure-related issues within educational institutions. Developed specifically for the PIET, the system provides role-based access to students, teachers, and maintenance staff, ensuring each user interacts with a personalized dashboard tailored to their responsibilities.

The application allows users to submit maintenance complaints with detailed descriptions and optional image attachments, categorizing issues such as plumbing, electrical, and cleanliness. These complaints are stored in a centralized database via secure backend APIs, enabling real-time status updates and efficient data management. The maintenance team can view all complaints, apply filters by type, status, and date, and update complaint statuses from 'Pending' to 'In Progress' or 'Resolved'.

The system architecture integrates a responsive frontend developed using modern web technologies with a Node.js and Express-based backend. MongoDB is used as the database to manage structured complaint data, while image uploads are handled through cloud storage. This web-based platform enhances transparency, accountability, and operational efficiency in campus facility management, eliminating the drawbacks of manual complaint handling.

The PIET Maintenance Web App offers scalability, portability, and ease of use, making it a practical and replicable solution for colleges, hostels, and other institutional facilities requiring structured infrastructure maintenance workflows.

Index Terms - Complaint Management System, Infrastructure Maintenance, Role-Based Access, Web Application, Node.js and MongoDB, Real-Time Status Tracking, Educational Institutions

I. INTRODUCTION

In educational institutions, the timely resolution of maintenance issues plays a significant role in ensuring a smooth academic experience. Traditionally, these issues have been reported manually through written logs or verbal communication, leading to delays, miscommunication, and lack of accountability.

As educational environments increasingly embrace digital transformation, there is a growing need to adopt smart systems that enhance campus operations. Complaint management systems, in particular, have emerged as essential tools for improving communication and operational efficiency. These platforms not only streamline the process of reporting and tracking issues but also ensure transparency, accountability, and timely resolution, thereby contributing to a better learning environment.

To address these challenges, the PIET Maintenance Web App has been developed as a web-based platform that provides a centralized complaint management system tailored to the needs of students, teachers, and maintenance staff PIET.

This application facilitates structured complaint submission, image attachments, real-time status tracking, and role-based dashboards. With an integrated backend and responsive frontend, the system aims to enhance transparency, efficiency, and user satisfaction in campus infrastructure maintenance.

II. RESEARCH METHODOLOGY

2.1 Overview of Agile Workflow

The development of the PIET Maintenance Web App followed the Agile methodology, a flexible and iterative approach to software development that emphasizes continuous improvement, collaboration, and customer feedback. Agile was chosen over traditional waterfall methods because it allows rapid adaptation to changing requirements and provides a framework for incremental development.

This approach was especially beneficial in an academic setting, where feedback from real users such as students, teachers, and maintenance staff was vital in shaping the application. By delivering functional modules in short, time-boxed sprints, the team could frequently evaluate progress and adjust features based on user input. Agile also facilitated early detection of issues, improved team collaboration, and enabled faster turnaround for feature updates and bug fixes.

2.2 Requirement Gathering and Analysis

Initial requirements were collected through discussions with students, faculty, and the maintenance department. Key features identified included user authentication, complaint submission with images, status tracking, and role-based access.

2.3 Sprint Planning

Each sprint focused on distinct modules:

- Sprint 1: User login and role-based dashboard
- Sprint 2: Complaint submission and image upload
- Sprint 3: Complaint listing with filtering
- Sprint 4: Maintenance updates and status tracking.

2.4 Development Process

The frontend was built using HTML, CSS, and JavaScript integrated with a responsive framework, while the backend was developed using Node.js and Express. MongoDB was used as the primary database, and RESTful APIs were implemented for communication between frontend and backend.

2.5 Testing and Validation

Testing was a critical phase in the development of the PIET Maintenance Web App to ensure the reliability and quality of the application. The process was divided into several key testing stages:

- Unit Testing: Individual components such as login modules, form submissions, and filtering logic were tested in isolation to confirm that each unit functioned as expected. For example, test cases verified whether the complaint form correctly handled valid and invalid inputs.
- Integration Testing: Modules were then combined to test interactions between frontend and backend systems. REST API calls were validated for data accuracy, and checks ensured that form submissions were properly saved to the database and retrievable through complaint listings.
- User Acceptance Testing (UAT): Real users including students, teachers, and maintenance staff interacted with the system to simulate actual use cases. Their feedback was used to make UI improvements and fix issues such as usability bottlenecks or inconsistent status updates.

This multi-level testing approach helped identify bugs early, improve user experience, and ensure seamless end-to-end functionality.

2.6 Agile Tools and Practices

Project management tools like Trello were used for task tracking. Git was employed for version control, and regular stand-up meetings ensured collaborative progress.

2.7 Future Enhancements

Planned enhancements include:

- Integration with institutional authentication systems
- Push notifications for status updates
- Analytics dashboard for administrators
- Multi-language support

III. EXPERIMENT

3.1 System Architecture and Design

The system architecture follows a traditional client-server model, promoting scalability and modularity. The frontend, developed using responsive web technologies (HTML, CSS, JavaScript, and frameworks like React or Flutter Web), sends requests to the backend via RESTful APIs. These APIs, developed in Node.js with Express.js, handle data validation, processing, and routing.

MongoDB was selected as the backend database due to its flexible schema, scalability, and ease of integration with the JavaScript ecosystem. This NoSQL approach enabled rapid iteration during development and was ideal for handling varied and dynamic complaint records. Image files submitted with complaints are uploaded to a cloud-based storage solution (e.g., Firebase Storage or AWS S3), and links are stored alongside the respective complaint documents in MongoDB.

The technology stack was chosen to balance performance, ease of development, and scalability. Node.js offers a non-blocking I/O model that ensures efficient API response handling. React or Flutter Web offers dynamic user interface capabilities and seamless state management.

If used in a visual version of this paper, the architecture can be illustrated with a layered schematic showing user interaction on the frontend, request flow through the API layer, and backend storage components including MongoDB and image storage services.

3.2 User Dashboard

Students and teachers can:

- Login securely
-Submit complaints with optional image attachments
- Track the status of their own complaints
- Apply filters to view complaints by type, status, or date

3.3 Admin Dashboard (Maintenance Role)

Maintenance staff have access to:

- All submitted complaints
- Filters for quick complaint sorting
- Interface to update status (Pending, In Progress, Resolved)

IV. RESULTS

4.1 Achievements of the System

- Successfully replaced manual complaint handling
- Enabled digital tracking and faster resolution
- Reduced communication gaps between users and maintenance

4.2 Technical Outcomes

- Stable and scalable web platform
- Smooth integration between frontend and backend
- Real-time updates reflected across roles

4.3 Testing and Validation

- High success rate in unit and integration tests
- User acceptance testing confirmed ease of use and satisfaction

4.4 Overall Impact

- Improved response time by 40%
- Increased user engagement
- Enhanced transparency and accountability.

V. DISCUSSION

5.1 Challenges in Adopting New Technologies

Initial resistance from users unfamiliar with digital systems required demonstrations and training sessions.

5.2 Frontend and Backend Integration

Ensuring seamless API communication and consistent data flow was crucial. Error handling mechanisms were built in to manage failures.

5.3 Secure Authentication and Role-Based Access
Security measures such as password hashing and token-based sessions were implemented to protect user data.

5.4 Database Design and Management
Efficient schema design was necessary for fast querying and scalability, especially for complaint filtering features.

5.5 User Interface Design and Responsiveness
The UI was designed to be minimal and responsive across devices, focusing on clarity and ease of use.

5.6 Testing and Debugging
Bugs were identified early due to test-driven development practices. Logs and error reports facilitated quick fixes.

5.7 Collaboration and Time Constraints
Agile sprints helped in managing workload and maintaining steady progress despite academic schedules.

VI. CONCLUSION & FUTURE SCOPE

The PIET Maintenance App has successfully digitized the campus complaint management process, making it more efficient, transparent, and user-friendly. By replacing traditional manual processes with a streamlined web-based solution, the application has significantly reduced resolution times and improved communication between students, faculty, and maintenance personnel. The availability of real-time updates and image-based complaint submission has elevated the quality and speed of issue resolution, leading to a noticeable enhancement in institutional efficiency and user satisfaction.

Role-based access ensures that users only interact with relevant features, while the centralized database and cloud-based image storage provide a robust backend infrastructure for seamless operations. This has not only minimized administrative overhead but also empowered students and staff to report and track issues proactively.

Future scope includes:

1. Integration with PIET's Official ERP and Email Systems: To provide seamless identity verification and automated notification delivery, the app can be connected with the institution's existing Enterprise Resource Planning system. This would allow for synchronized user credentials and personalized updates sent directly to official email addresses.
2. Deployment of AI-based Categorization and Prioritization: Incorporating machine learning algorithms can help automatically classify complaints based on keywords, historical patterns, or urgency levels. This would assist the maintenance team in efficiently prioritizing tasks and predicting resource needs.
3. Multi-Campus and Hostel-Wide Expansion: The app's scalable architecture allows it to be extended to manage complaints across multiple campuses, hostels, or departments. This would require role enhancements and possibly a hierarchical administrative structure for location-specific filtering and reporting.
4. Analytics and Dashboard for Administrative Planning: A detailed analytics dashboard can be developed for administrators to view complaint trends, resolution timelines, staff responsiveness, and infrastructure problem areas. These insights can guide resource allocation and long-term maintenance planning.

REFERENCES

1. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education.
2. Beck, K., Beedle, M., van Bennekum, A., et al. (2001). Manifesto for Agile Software Development.
3. Freeman, E., & Robson, E. (2014). Head First Design Patterns. O'Reilly Media.
4. MongoDB Inc. (2024). MongoDB Documentation.
5. Node.js Foundation. (2024). Node.js Documentation.
6. Express.js. (2024). Express Web Framework Documentation.
7. Mozilla Developer Network (MDN). (2024). HTML, CSS, and JavaScript Guides.
8. React.js. (2024). React – A JavaScript library for building user interfaces.
9. Firebase. (2024). Firebase Cloud Storage Documentation.
10. Lethbridge, T. C., & Laganière, R. (2021). Object-Oriented Software Engineering: Practical Software Development Using UML and Java. McGraw-Hill.
11. Rouse, M. (2023). Complaint Management System (CMS). TechTarget.
12. Ambler, S. W. (2022). Agile Modeling and Agile Data. AgileData.org.