

Chapter One

METHODOLOGY OF STUDY

3.1 Introduction

The existing methodology of a campus maintenance request and approval system includes the use of multiple communication channels like text alerts, emails, and notice messages for quick information dissemination.

3.2 OVERVIEW OF PROPOSED METHODOLOGY

The proposed automated campus maintenance request and approval system is designed to streamline the process of reporting, managing, and resolving maintenance issues on campus. This system aims to enhance efficiency, reduce response times, and ensure proper tracking and management of maintenance tasks.

Benefits

1. **Increased Efficiency:** Automating the process reduces the time spent on manual paperwork and approvals, allowing maintenance staff to focus on resolving issues.
2. **Better Communication:** Clear, real-time communication between users, maintenance teams, and administrators ensures that everyone is informed and issues are addressed promptly.
3. **Improved Accountability:** The system's tracking and reporting features create a clear record of all maintenance activities, promoting accountability and ensuring that tasks are completed.
4. **Enhanced User Experience:** Providing a simple, accessible way for users to report issues improves overall satisfaction and campus experience.

3.3. METHOD OF DATA COLLECTION

1. Observation:

- i. **Target Group:** Maintenance teams and administrative staff.
- ii. **Purpose:** Understand the current maintenance request and approval processes by directly observing day-to-day activities.
- iii. **Method:** Shadow maintenance personnel and administrative staff during their work to observe how requests are handled, from submission to completion.

2. Document Analysis:

- i. **Target Group:** Existing records and documentation related to maintenance requests and approvals.
- ii. **Purpose:** Review historical data on maintenance requests, approval timelines, and recurring issues to identify trends and bottlenecks.
- iii. **Method:** Analyze maintenance logs, request forms, and approval records to extract relevant data and insights.

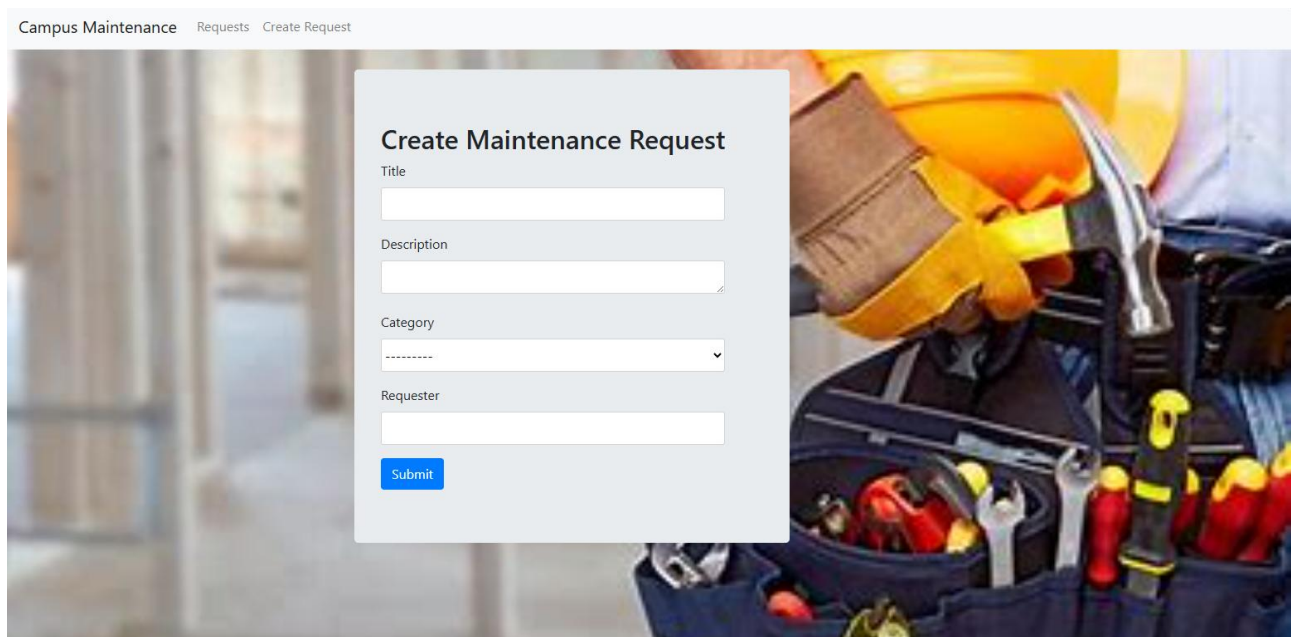


Fig 3.1 Campus maintenance request System

Source: From the Application design

3.4 SYSTEM DESIGN

The proposed automated campus maintenance request and approval system consists of a comprehensive, scalable, and secure architecture designed to facilitate efficient data collection, analysis, and reporting. The system comprises the following key components:

Front-End (User Interface):

Web and Mobile Interfaces: Provides user-friendly and responsive interfaces for survey participation and viewing results.

Back-End (Server-Side): Application Server: Manages business logic, user authentication, and survey distribution using Django Rest Framework.

Database Server: Stores survey data and user information securely using SQLite.

Security Layer: Authentication and Authorization: Ensures secure access through role-based controls and encryption.

Key Modules

1. Surveys Management: Instruments for designing, modifying, and sending surveys.
2. Data Collection: Real-time, automated data collection with privacy safeguards.
3. Data Analysis and Reporting: Tools for data interpretation and visualization, including analytics engines.
4. User Management: Admins' profile and role management features.
5. Efficiency Optimization: To improve responsiveness, cache and query optimization are used.
6. Maintenance and Support
7. User Support: Documentation and a help desk for users.

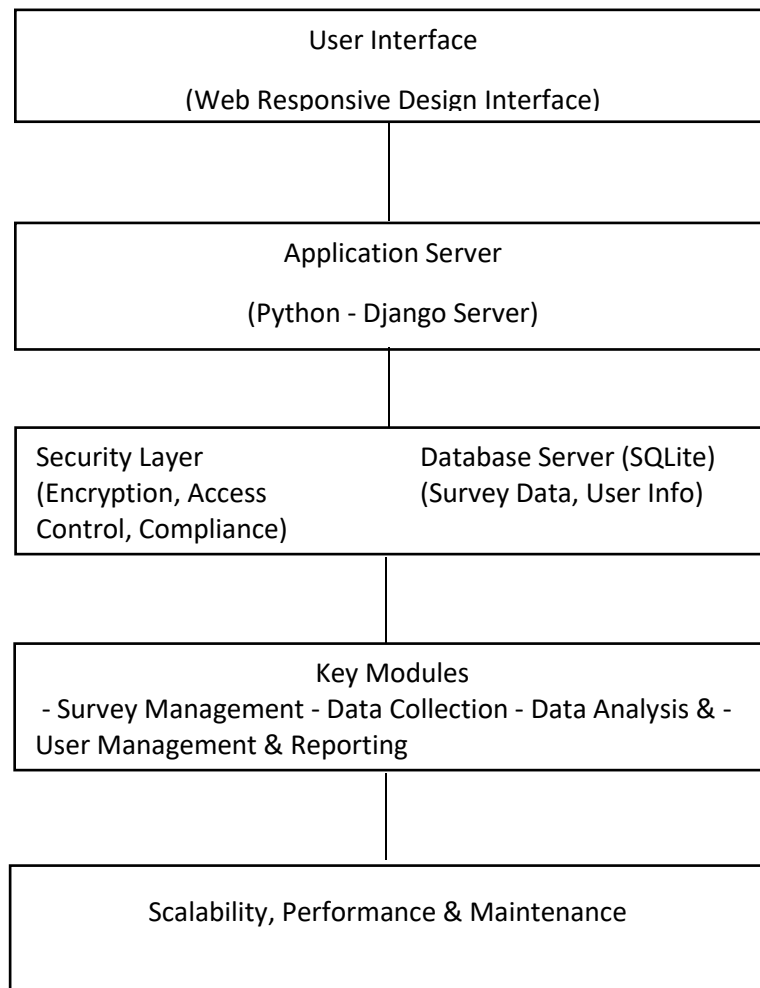


Fig 3.2 Showing System Design Use Case Diagram of the application.

This diagram illustrates the architecture's layers and key components, showing the flow from the user interface through the back-end processing and data management, with a focus on security, integration, and scalability.

3.5. DATABASE STRUCTURE

Data Structure for a Automated Campus Maintenance Request and Approval System.

The data structure for the automated campus maintenance request and approval system is designed to efficiently store and manage various types of information, including user data, survey details, responses, and analysis results. The structure is typically organized into several key entities, each with its attributes, ensuring data integrity and accessibility.

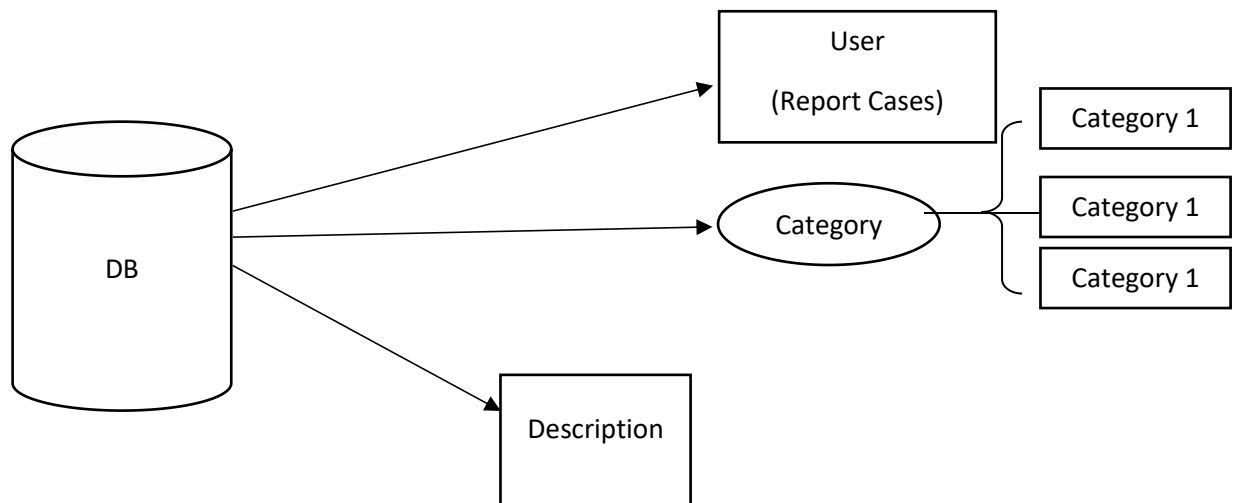


Fig 3.3 Database Structure for an Automated Campus Maintenance Request and Approval System

3.5.1. INPUT STRUCTURE

The input structure defines how data is formatted and received by the system, particularly during survey creation, response submission, and user management. This structure ensures data consistency, validation, and ease of processing. Here's an overview of the key input structures:

1. User Entity

Attributes:

- i. UserID: Unique identifier for each user (Primary Key).
- ii. Username: User's login name.
- iii. Password: Hashed password for secure authentication.

- iv. Email: User's email address.

2. Maintenance Request Entity

Attributes:

- i. id: Unique identifier for each survey (Primary Key).
- ii. title: type of the incidence.
- iii. requester: posted by.
- iv. description: Brief description of the incidence.
- v. category: category of the populace.
- vi. responses: Foreign key relationship with response table

3. Category

Attributes:

- i. id: Unique identifier for each response (Primary Key).
- ii. name: name of the category of maintenance

3.5.2 OUTPUT STRUCTURE

1. Response Entity

Attributes:

- i. id: Unique identifier for each response (Foreign Key).
- ii. responder: Identifier of the related survey (Foreign Key).
- iii. maintenance_request: category of maintenance request (Foreign Key)..

2. Admin User Entity

Attributes:

- i. id: The user id has a unique key called the primary key
- ii. name: Name of the admin user
- iii. email: email of the admin user (This is used in-terms of administering privileges and roles)

CHAPTER FOUR

SYSTEM TESTING AND IMPLEMENTATION

4.0 INTRODUCTION

Important stages in the automated campus maintenance request and approval system development lifecycle are system testing and implementation. The system's user appropriateness, planned functionality, and deployment readiness are examined at these stages. The aims, procedures, and techniques used in these stages are outlined in this introduction.

4.1 CHOICE OF PROGRAMMING, LANGUAGES AND DATABASE

Database and Programming Language Selection A dependable, scalable, and efficient digital campus emergency and response system must be built using the appropriate database systems and programming languages. Some of the factors influencing the choice are developer experience, integration abilities, security, scalability, and system requirements. The following is a summary of the recommended database systems and programming languages for this project:

1. Programming Languages

Front-End Development:

i. HTML5 and CSS3 (Bootstrap):

- a) The user interface must be structured and styled using HTML5 and CSS3. They serve as the basis for creating designs that are responsive and accessible to a wide range of devices and screen sizes.
- b) Advantages: Easy to use, versatile in design, and supported by a wide range of browsers.

ii. JavaScript:

- a) Rationale: JavaScript is the standard language for web development, providing rich interactivity and responsive user interfaces integrated with bootstrap for.
- b) Benefits: High performance, large community support, extensive libraries and tools, and easy integration with other technologies.

Back-End Development:

i. Python (with Django Framework):

- a) Rationale: Python is known for its simplicity, readability, and versatility. Django, a high-level Python web framework, is well-suited for developing secure and maintainable web applications. It provides built-in functionalities for authentication, URL routing, ORM (Object-Relational Mapping), and more.
- b) Benefits: Rapid development, strong security features, comprehensive documentation, and a large ecosystem of packages and tools.

2. Database Systems

Relational Database:

i. SQLite:

- a) Rationale: SQL is a widely-used open-source relational database, known for its reliability, ease of use, and performance. It is suitable for applications with a large volume of read-heavy workloads.
- b) Benefits: Extensive documentation, wide community support, and integration with various platforms and tools.

The digital campus emergency and response system's database system and programming language selection is influenced by factors including security, ease of development, scalability, and the capacity to manage intricate data exchanges. Utilizing technologies such as CSS, HTML, JavaScript, Python, and SQLite, the system can offer a reliable, easy-to-use, and effective platform for gathering and evaluating ideas on campus.

4.2 SYSTEM REQUIREMENTS

The system requires a scalable design with secure authentication and data encryption to manage enormous volumes of survey responses and real-time data analytics. Supporting online and mobile interfaces as well as multi-platform accessibility should guarantee a consistent user experience across devices.

4.2.1 HARDWARE REQUIREMENT AND SPECIFICATIONS

The system requires a cloud-based server infrastructure with at least 4 CPU cores, 8 GB of RAM, and 100 GB of SSD storage to enable scalability and performance. For development and testing, a local PC with a multi-core processor, 256 GB SSD, and 16 GB RAM is recommended. High-speed internet connectivity and backup solutions are necessary to maintain data availability and integrity.

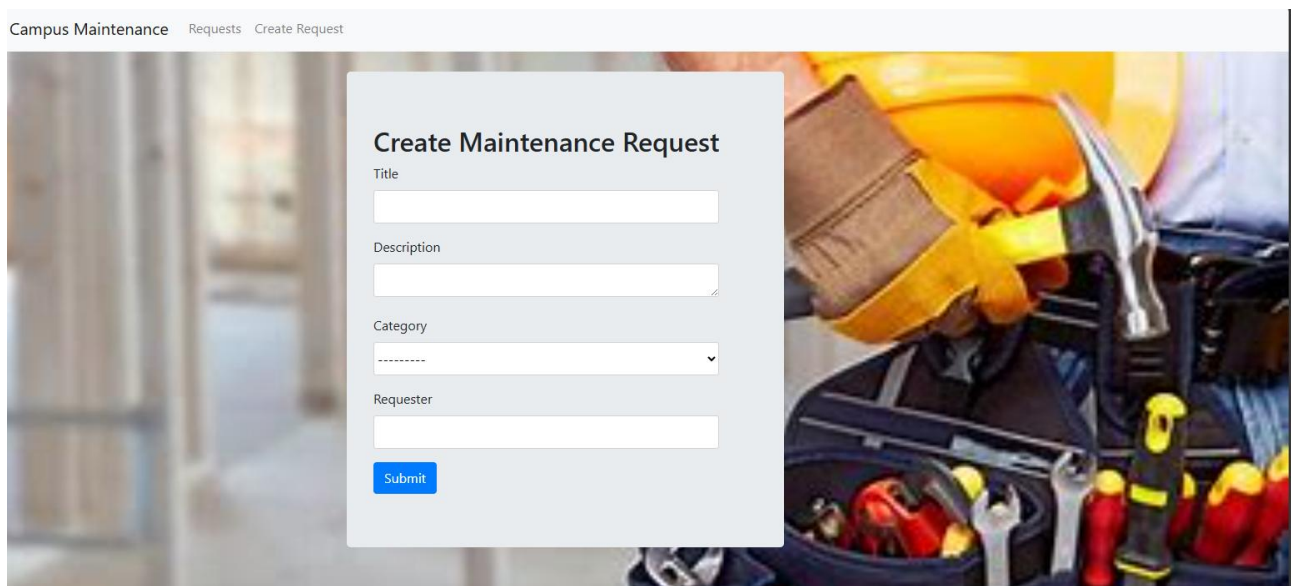
4.2.2 SOFTWARE REQUIREMENTS AND SPECIFICATIONS

To hosting the application, the system needs a Windows or Linux-based server environment

(such as Windows 10, Ubuntu), with Python and Django for backend development. It is advised to use a front-end stack with HTML5, CSS3, and JavaScript. PostgreSQL, SQLite, or MySQL are also required by the system for relational data storage. Any local server, such as Xampp or Wampp, can execute the system. It can also be connected into the cloud using PythonAnywhere, a customizable cloud hosting service for Python.

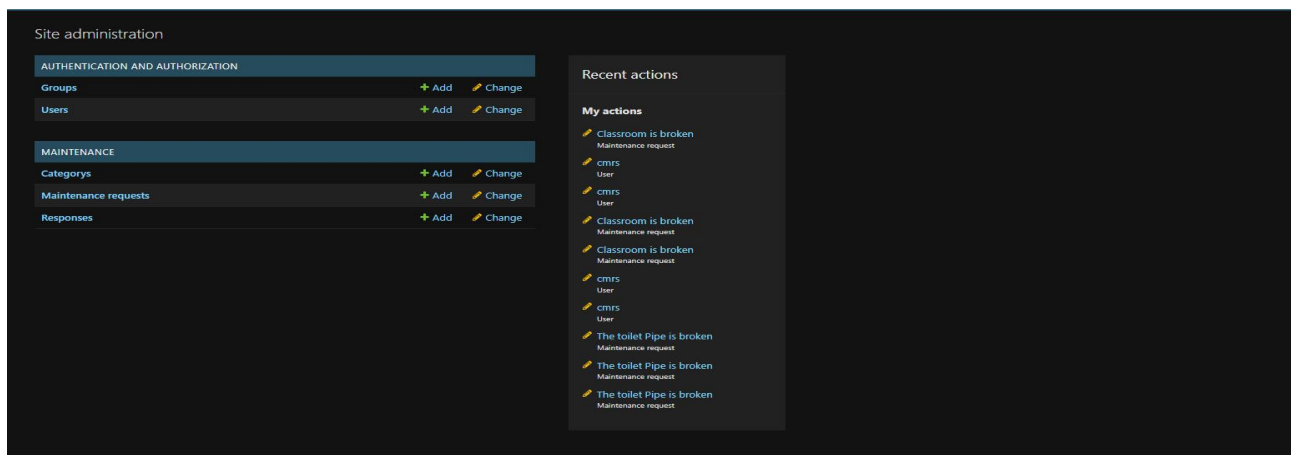
4.3 RESULT INTERFACE

The result interface uses interactive dashboards with cards and tables to display survey data for simple examination. All request and actions within the application are visible to the administrator. The interface further improves the user experience in data analysis by including visual tools such as word clouds for qualitative data.



The screenshot shows a web application interface for creating a maintenance request. At the top, there is a navigation bar with the text "Campus Maintenance" and two links: "Requests" and "Create Request". The main content area features a light gray modal box titled "Create Maintenance Request". Inside this modal, there are four input fields: "Title", "Description", "Category" (a dropdown menu), and "Requester". Below these fields is a blue "Submit" button. The background of the page is a blurred image of a person wearing a yellow hard hat and safety glasses, working with tools in a toolbox.

Fig 4.1 Maintenance Request



The screenshot displays an administrator dashboard with a dark theme. On the left side, under the heading "Site administration", there are two sections: "AUTHENTICATION AND AUTHORIZATION" and "MAINTENANCE". The "AUTHENTICATION AND AUTHORIZATION" section includes links for "Groups" and "Users", each with "+ Add" and "Change" options. The "MAINTENANCE" section includes links for "Categorys", "Maintenance requests", and "Responses", each with "+ Add" and "Change" options. On the right side, there is a "Recent actions" section titled "My actions". This section lists several actions, each preceded by a yellow checkmark icon. The actions include "Classroom is broken Maintenance request", "cms User", and "The toilet Pipe is broken Maintenance request".

Fig 4.2 Admin Dashboard View

4.4 SYSTEM TESTING AND IMPLEMENTATION

User acceptability testing and integration testing are two of the many tests the system undergoes to ensure functioning, security, and usability. The implementation strategy is a phased rollout, where a pilot group is included before the entire group is distributed. After launch, continuous monitoring and support are provided to address issues and enhance performance.

4.5 DESCRIPTION/DISCUSSION OF FINDINGS

The installation of the automated campus maintenance request and approval system significantly improved the effectiveness of data collection and user interaction. The system's user-friendly interface and real-time analytics capabilities allowed for deeper insights on sentiment on campus. Additionally, the transition from manual to digital processes reduced errors and administrative load, resulting in more reliable and valuable data for decision-making.

4.6 SYSTEM DOCUMENTATION

Comprehensive installation, configuration, and operation instructions as well as troubleshooting advice are included in the system documentation. To help developers and end users alike, it offers thorough knowledge on system architecture, data structures, and user interfaces. Consistent updates guarantee that documentation stays up to date and accurately reflects system modifications.

4.6.1 HOW TO LOAD THE SOFTWARE

To load the software, use the following steps:

1. Create a folder in a desktop setting.
2. Transfer the program from its original place to the newly created folder.
3. Install Python on your computer and include it in an environment variable.
4. Add an OS path to the configuration settings in the emergency folder.

4.6.2 HOW TO RUN THE SOFTWARE

Apply the following steps to run the software:

Locally

1. Open your command prompt and CD into the folder location of the maintenance folder inside the parent folder

2. Located the maintenance sub-folder and run *pip install django* to install Django rest frame-work
3. Now, run *python manage.py makemigrations* to load the admin, make current data migrations
4. Now, run *python manage.py migrate* to migrate the data into the database
5. Configure your static file and templates.
6. After all configurations, run *python manage.py runserver* to run the application
7. Using the inbuilt SQLite for Database and Django Server for local hosting

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

The automated method for requesting and approving campus maintenance is intended to increase the efficacy and efficiency of addressing maintenance concerns on campus. Through an intuitive interface, this system enables staff, instructors, and students to submit maintenance requests, which are then automatically forwarded to the relevant maintenance teams. The technology shortens response times, lessens the administrative load, and improves communication between users and maintenance staff by automating the process.

5.2 CONCLUSION

To improve campus facility management, an automated system for repair requests and approvals must be put in place. It expedites the reporting and resolution of maintenance issues, increasing user happiness and speeding up the process. By following requests from submission to completion, the system also promotes accountability and transparency. Additionally, reporting capabilities give campus administrators the flexibility to make defensible decisions based on insights gleaned from data.

5.3 RECOMMENDATION

To optimize the advantages of the automated campus maintenance request and approval system, it is imperative to carry out comprehensive training and onboarding for all users, comprising maintenance staff, faculty, and students, to guarantee they are proficient in operating the system. To track performance and collect user input, regular system evaluations should be conducted. This will enable ongoing enhancement and flexibility in response to evolving requirements. Coordination and resource management will be improved by integrating the system with currently used campus management technologies, such as inventory and facility management systems. Encouraging wider adoption also requires encouraging user interaction, which makes sure that all maintenance concerns are recorded and reported through the system. Finally, a more effective and efficient maintenance management process may be achieved by routine data analysis utilizing the system's reporting and analytics functions to spot trends, enhance maintenance operations, and deal with reoccurring problems.