Infra Audits Project Documentation: A MERN Stack Web Application

Name: RAGULRAM B
Seat No: 132
Project ID: 12
Project Title: Infra Audits

Technical Components:

Tech Stack	Components
Mongo DB	Data Base
Express JS	Backend
Node JS	Backend
React JS	Frontend

Problem Statement:

The current manual process of auditing infrastructure within the college is inefficient and lacks a centralized system for managing audits, complaints, and maintenance tasks. Auditors manually inspect areas, record findings, and report complaints, which are then communicated to the manager via disparate channels. This lack of organization leads to delays in addressing issues, miscommunication among stakeholders, and difficulties in tracking audit progress and task completion.

1. Introduction

1.1. Purpose:

The Infra Audits application is designed to automate and centralize the management of infrastructure audits within a college campus.

1.2. Scope of Project:

This project focuses on developing a web application with three user roles:

- **Editor (Manager):** Assigns audit tasks to auditors, manages complaints identified during audits, and assigns repair tasks to fixers.

- **Auditor:** Views assigned audit tasks, conducts audits based on predefined checklists, and reports identified complaints.
- **Fixer (Handyman):** Views assigned repair tasks, fixes reported complaints, and updates the status with completion photos.

2. System Overview

2.1. Users:

- Editor (Manager)
- Auditor
- Fixer (Handyman)

2.2. Features:

- **Editor Features:**
  - Assign Audits: Assign weekly audit tasks to specific auditors, including locations and deadlines.
  - Manage Complaints: View complaints reported by auditors, categorize them (civil, carpentry, electrical, plumbing), and assign them to fixers.
  - Track Repairs: View assigned repair tasks, track their progress, and receive completion updates with photos from fixers.
  - Dashboard: View an overview of assigned audits, reported complaints, and ongoing repairs.
- **Auditor Features:**
  - View Assigned Tasks: See upcoming and past audits assigned by the editor.
  - Conduct Audits: Use a checklist-based interface to record observations for assigned locations (cleanliness, furniture damage, door/handle condition).
  - Report Complaints: Identify and report specific complaints (civil, carpentry, electrical, plumbing) during audits.
  - Update Status: Mark audits as completed and submit reports to the editor.
- **Fixer Features:**
  - View Assigned Repairs: See assigned repair tasks categorized by complaint type (civil, carpentry, electrical, plumbing).

- Update Repair Status: Update the status of assigned repairs (in progress, completed).
- Upload Photos: Attach photos as proof of completion for fixed repairs.

2.3. Assumptions:

- Active Google Accounts: All users possess active Google accounts for authentication purposes.
- Device Availability: Users have regular access to internet-enabled devices to utilize the application.

3. Requirements

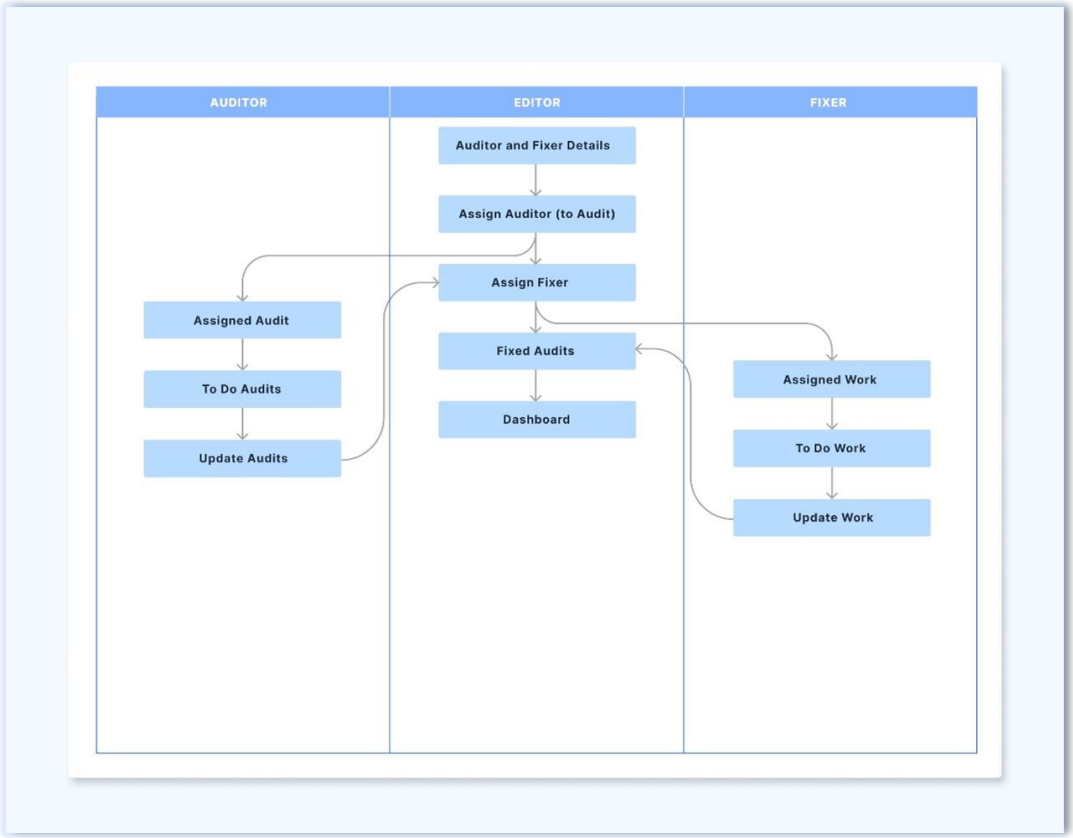
3.1. Functional Requirements:

- Role-Based Access Control: Define and enforce access permissions for each user role (editor, auditor, fixer).
- Audit Management:
  - Allow editors to assign weekly audits to specific auditors, including location and deadline.
  - Enable auditors to view assigned tasks and conduct audits using a checklist-based interface.
  - Facilitate auditors to report identified infrastructural complaints categorized by type (civil, carpentry, electrical, plumbing).
- Complaint Management:
  - Allow editors to view complaints reported by auditors.
  - Enable editors to categorize complaints (civil, carpentry, electrical, plumbing).
  - Facilitate editors to assign repair tasks for reported complaints to specific fixers.
- Repair Management:
  - Allow fixers to view assigned repair tasks categorized by complaint type.
  - Enable fixers to update the repair status (in progress, completed).
  - Facilitate fixers to upload photos as proof of completion for fixed repairs.
- Reporting:
  - Provide editors with a dashboard to view overall statistics on assigned audits, reported complaints, and ongoing repairs.

3.2. Non-Functional Requirements:

- Security: Implement secure user authentication and authorization mechanisms.
- Scalability: Design the system to accommodate a growing number of users and audits.
- Performance: Ensure the application responds quickly to user interactions.
- User Interface: Develop an intuitive and user-friendly interface for all user roles.
- Accessibility: Design the application to be accessible to users with disabilities.

4. Work Flow:



5.Flow Chart: