INTRODUCTION

Incident-Hub is a fully featured **enterprise incident management system** designed to streamline how organizations report, track, manage, and resolve technical issues across infrastructure and applications. The platform provides a centralized portal where authorized users can create incidents, monitor progress, communicate with assigned engineers, and generate reports for audits and analysis.

This system incorporates **role-based access**, **real-time email notifications**, **file attachments**, **audit logs**, and a clean, responsive user interface, ensuring operational efficiency and transparency within enterprise technology environments.

Abstract

This project is a **minimal yet functional incident management web application** developed using open-source technologies such as **Python (Flask/Django)**, **SQLite**, and **Bootstrap**. The system enables users to create incidents, update their progress, assign them to responsible engineers, and close them upon resolution.

To enhance usability, the system integrates **SMTP-based email notifications**, which keep stakeholders updated about incident status changes. The backend provides a clean set of **REST APIs** that handle core operations such as creation, assignment, and resolution of incidents.

With the help of **Docker**, the application is fully containerized, ensuring environment consistency and ease of deployment. All code and revisions are maintained using **Git**, enabling better collaboration and version tracking.

Tools Used

Backend Technologies

- **Python (Flask/Django)** – Framework for API handling and server logic
- **SQLite** – Lightweight relational database
- **REST APIs** – For incident operations (create, update, assign, resolve)

Frontend Technologies

- **HTML5 / CSS3 / JavaScript** – For UI structure and interactivity

Additional Tools

- **SMTP Email (Gmail or custom)** – For incident notifications **to the admin and assigned user**
- **Git** – Version control and collaboration

Steps Involved in Building The Project

1. Project Setup & Architecture

- Initialized Flask app with modular folder structure.
- Configured environment variables for DB, SMTP, and app settings.

2. Database Design & Models

- Created SQLite database with models for User, Incident, Comment, etc.
- Added foreign key relations for structured incident tracking.

3. Authentication & Authorization

- Implemented login, registration, and session-based auth.
- Applied role-based permissions for Admin, Engineer, and Reporter.

4. Core Incident Management

- Built REST APIs for create, update, assign, and resolve incidents.
- Added status workflow (Open → In Progress → Resolved → Closed).

5. User Interface Development

- Designed responsive UI using Bootstrap.
- Created role-specific dashboards and incident detail pages.

6. Email Notification System

- Integrated SMTP for incident alerts and updates.
- Added triggers for new, assigned, and resolved incidents.

7. File Management

- Implemented secure file uploads with UUID naming.
- Added attachment access controls and download support.

8. Deployment & Version Control

- Containerized the app using Docker for consistent deployment.
- Used Git for version control and maintaining project history.

Conclusion

The Open-Source Incident Management System successfully delivers a lightweight yet effective solution for logging, tracking, and resolving infrastructure or application issues. By combining **Flask**, **SQLite**, **Bootstrap**, **SMTP email notifications**, **Docker**, and **Git**, the project demonstrates how a simple open-source stack can create a functional and scalable workflow tool.

The system provides essential features such as role-based access, REST APIs, incident lifecycle management, file attachments, and real-time notifications—making it suitable for small teams, learning environments, or community-driven projects. Overall, the project meets its objectives by offering a structured, user-friendly platform that improves issue visibility, communication, and resolution efficiency