Name	NITHEESH KUMAR M
Roll Number	7376221SE133
Seat Number	217
Project ID	10
Problem Statement	BIT MAILER(FACULTY LOG)

TECHNICAL COMPONENTS:

Component	**Tech stack**
Frontend	Angular/React
Backend	Spring Boot Java
Database	MySQL
API	RESTful services

PROBLEM STATEMENT:

The decentralized nature of email communication within educational institutions leads to several challenges, including:

● Inconsistent messaging: Different departments and administrative units send emails independently, leading to duplication of information and inconsistent messaging.
● Schedule conflicts: Faculty receive multiple emails with overlapping schedules and events, leading to confusion and missed opportunities.
● Fragmented communication: Important announcements and updates get lost in the volume of emails, making it difficult for recipients to stay informed and engaged.
● Administrative burden: Managing email distribution lists, resolving conflicts, and ensuring timely delivery of critical information imposes a significant administrative burden on staff / faculty.

PROJECT FLOW:

PURPOSE:

The goal is to create a centralized mailing system that effectively handles communication about the activities and schedules of staff/faculties, resolving the current problems of conflicting schedules and inconsistent communication.

SCOPE:

A mailer request form, conflict checks, login authentication, and a real-time dashboard for organizing and viewing schedules are all included in this system. It combines with current email systems to guarantee scheduled and harmonious communication.

BUSINESS CONTEXT:

By reducing schedule conflicts, the centralized mailing system seeks to improve communication throughout BIT and increase organizational efficiency. The faculty members and administrative personnel are the main stakeholders.

CONSIDERATION:

- All users possess login credentials for authentication.
- Internet-enabled devices are regularly accessible to users.

DEPENDENCIES:

- Integration with login credentials for user authentication.
- reliability of the current email server's performance and accessibility.

USERS:

- Faculty members
- Admin staff

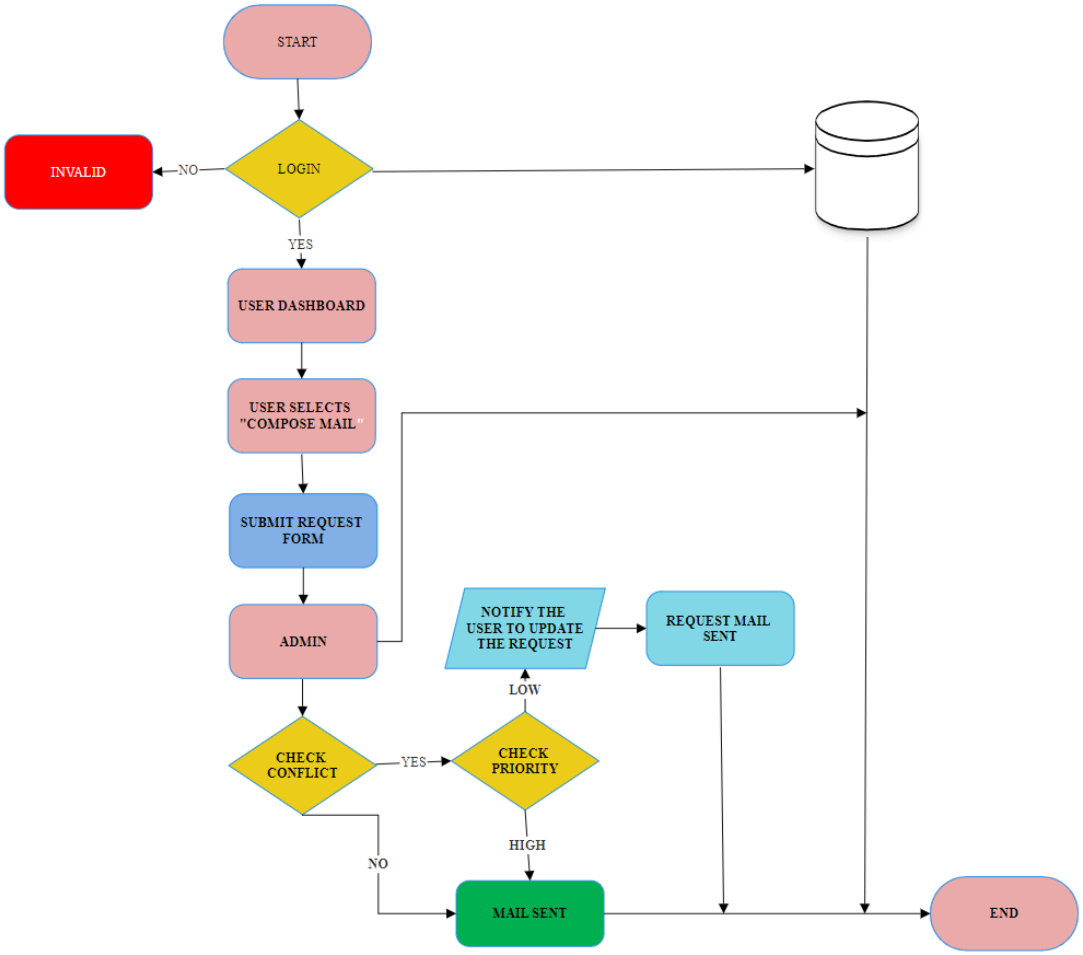
USER STORIES:

As a faculty member, he /she needs to ensure his/her duties without conflicting with their other scheduled activities.
As an admin staff,he/she needs to check if a faculty member has conflit meetings.If exists,then click on the reject button and return to a certain department to update the request.

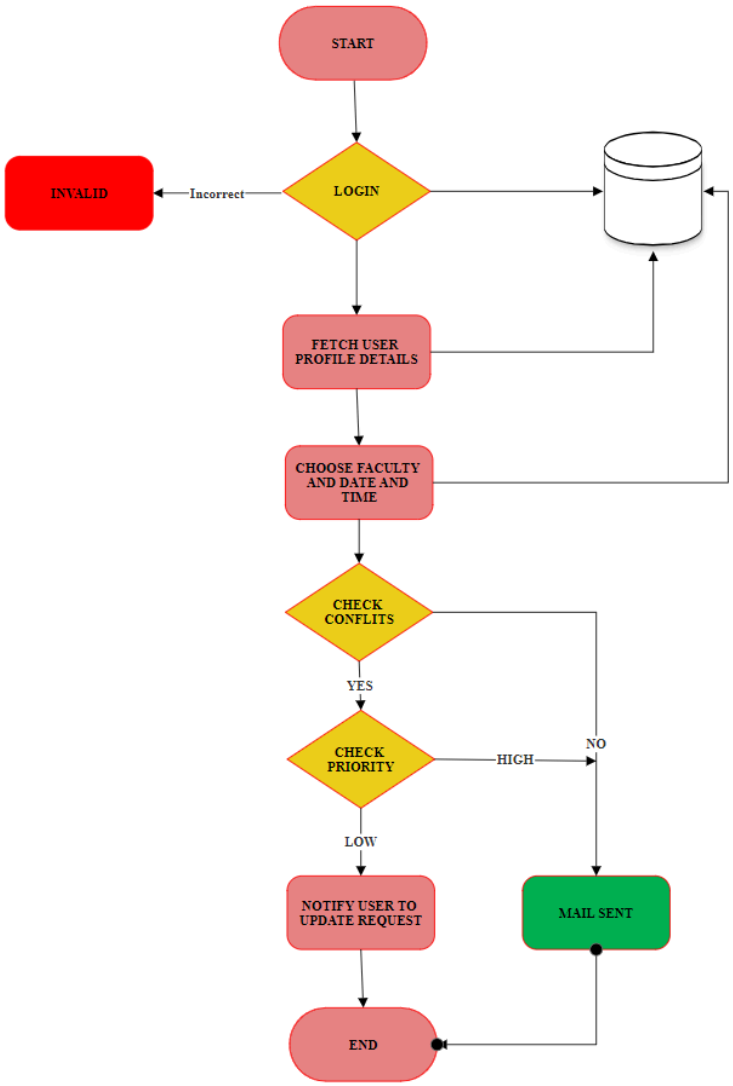
FUNCTIONAL REQUIREMENTS:

1. Centralized Messaging
2. Conflict Resolution
3. Unified Announcements
4. Schedule Management
5. Distribution Automation

FLOW CHART:


START
LOGIN
INVALID
NO
YES
USER DASHBOARD
USER SELECTS "COMPOSE MAIL"
SUBMIT REQUEST FORM
ADMIN
NOTIFY THE USER TO UPDATE THE REQUEST
REQUEST MAIL SENT
LOW
CHECK CONFLICT
YES
CHECK PRIORITY
HIGH
NO
MAIL SENT
END

ADMIN PROCESS:



Functional Dependencies for the Overall Project:

1. User Interface (Frontend - React)

- **Depends on:** Proper connection to the backend RESTful APIs.
- **Example:** The email approval dashboard depends on API endpoints to fetch and update email statuses.
- **Technologies:** React, Axios/Fetch for API calls.

2. API Layer (Backend - Spring Boot)

- **Depends on:** Properly defined endpoints, request/response handling, and business logic implementation.
- **Example:** The email-related endpoints (`/emails`, `/emails/{id}/approve`, `/emails/{id}/decline`) depend on the corresponding services to perform actions.
- **Technologies:** Spring Boot, RESTful APIs.

3. Business Logic (Service Layer - Spring Boot)

- **Depends on:** Repository layer for data access and entities for data representation.
- **Example:** The `EmailService` depends on `EmailRepository` to fetch emails from the database and manage approval/decline operations.
- **Technologies:** Spring Boot.

4. Data Access (Repository Layer - Spring Boot)

- **Depends on:** Entity classes and database tables for CRUD operations.
- **Example:** `EmailRepository` depends on the `Email` entity and corresponding MySQL table to perform data operations.
- **Technologies:** Spring Data JPA, Hibernate.

5. Database (MySQL)

- **Depends on:** Proper configuration and schema design to store and manage application data.
- **Example:** The `emails` table stores all email records and their statuses.
- Technologies: MySQL, SQL.

These dependencies ensure seamless interaction between the frontend and backend, enabling efficient data handling and user interactions for the admin dashboard project.