****

**STUDENT NAME: SUVATHI S**

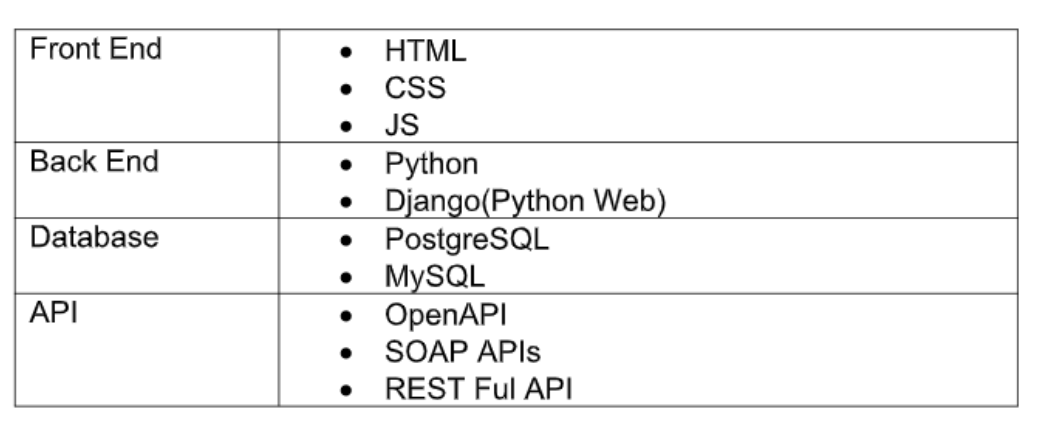
**SEAT NO: 369**

**PROJECT ID: 6**

**PROJECT TITLE: REPEAT TASK (TASK PORTAL)**

**STACK : PYTHON STACK(AI)**

**TECHNICAL COMPONENTS:**

****

**Module: Repeat Task**

This module will enable the creation of repeat tasks that need to be repeated on a daily, weekly, or monthly basis, similar to an alarm system.

**Domain: Task Portal**

Using the Python stack, we will create the Repeat Task module as part of the task portal system. This will be integrated with the centralized mailing system to ensure consistent and conflict-free communication within the institution.

**Problem Statement:**

Educational institutions face challenges due to decentralized email communication, such as inconsistent messaging, schedule conflicts, fragmented communication, and administrative burden.

**Purpose:**

Develop a centralized mailing system to manage communication efficiently, addressing schedule conflicts and messaging inconsistencies.

**Scope:**

* User Authentication via Google OAuth
* Mailer Request Form
* Conflict Checks
* Real-time Dashboard for schedule management
* Integration with existing email systems

**Business Context:**

Enhance communication clarity and timeliness across BIT, boosting efficiency by minimizing scheduling conflicts. Stakeholders include students, faculty, administrative staff, and IT department.

**Considerations:**

* All users have active Google accounts.
* Users have regular access to internet-enabled devices.

**Dependencies:**

* Integration with Google OAuth
* Availability of the existing email server

**User Personas:**

* **Student**: Needs a unified schedule to plan activities.
* **Faculty**: Requires efficient communication with students.
* **Admin Staff**: Manages system operations, resolves conflicts, and approves mail requests.

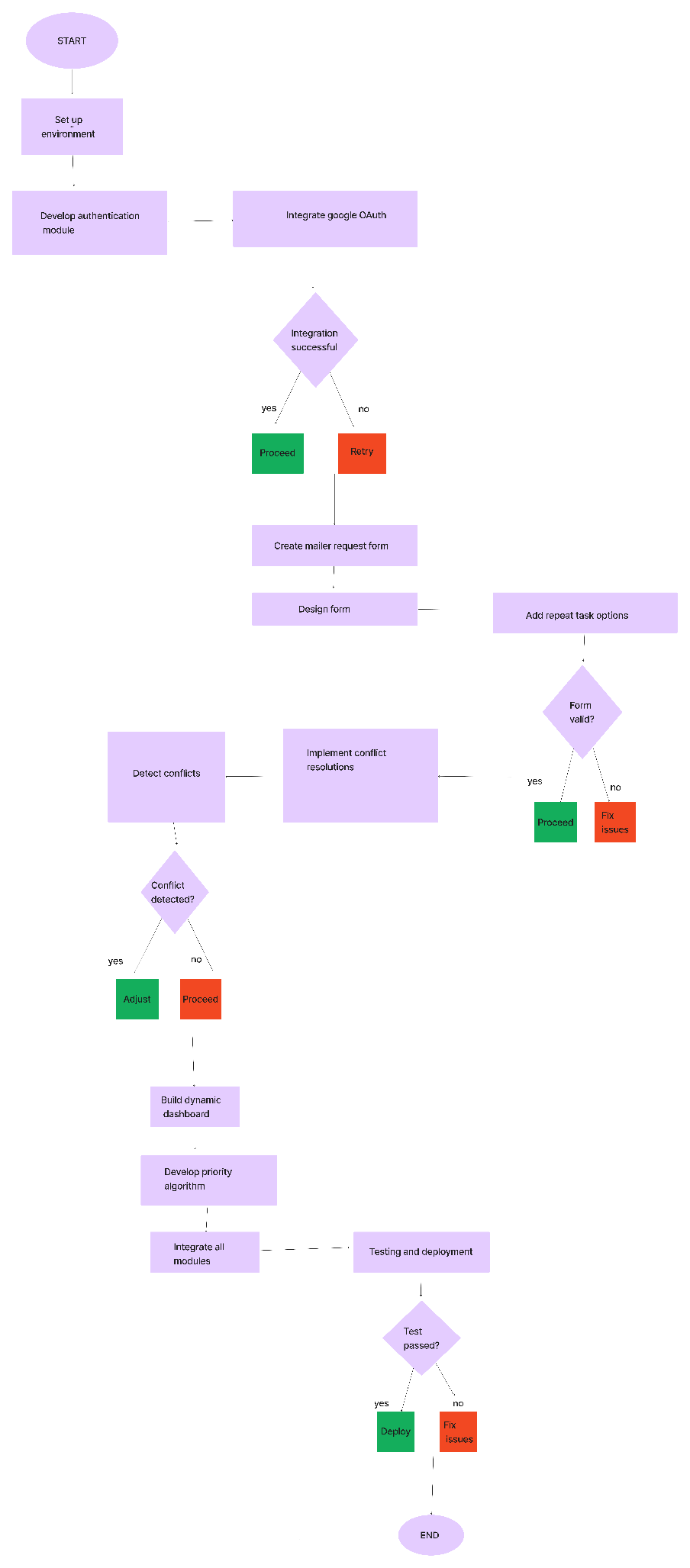
**User Stories:**

1. **Student**: View a unified schedule of classes and events.
2. **Faculty**: Ensure communications reach students without schedule conflicts.

**Functional Requirements:**

1. **User Authentication**: Secure login using Google OAuth.
2. **Mailer Request Form**: Users input mail content, scheduling time, category, and recipients.
3. **Conflict Resolution**: Automatic detection and adjustment of scheduling conflicts.
4. **Dynamic Dashboard**: Real-time schedule viewing and interaction.
5. **Priority Algorithm**: Automated prioritization of communications based on rules.

**FLOWCHART:**



**Repeat Task Module Implementation:**

**1. User Authentication**

* Integrate Google OAuth for secure user authentication.
* Ensure smooth login and access to the system.

**2. Mailer Request Form**

* Develop a form for users to input mail content, scheduling details, category, and recipients.
* Allow setting up repeat tasks on a daily, weekly, or monthly basis.

**3. Conflict Resolution**

* Implement automatic conflict detection for scheduling.
* Provide options for users to adjust conflicting schedules.

**4. Dynamic Dashboard**

* Design a real-time dashboard for viewing and managing schedules.
* Display repeat tasks and their occurrences clearly.

**5. Priority Algorithm**

* Create an algorithm to prioritize communications based on predefined rules.
* Ensure high-priority messages are delivered without conflicts.

**Implementation Steps:**

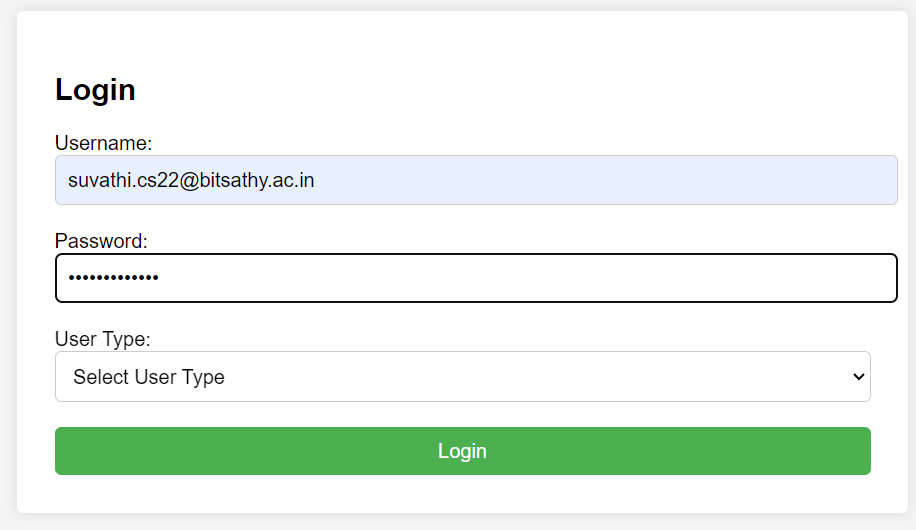
1. **Setup Project Environment**
   * Install necessary libraries and frameworks (Flask, OAuthlib, etc.).
   * Configure Google OAuth for user authentication.
2. **Develop User Authentication Module**
   * Implement Google OAuth integration.
   * Create user login and session management features.
3. **Create Mailer Request Form**
   * Design the form for mail content, scheduling details, and recipient input.
   * Add options for setting repeat tasks.
4. **Implement Conflict Resolution**
   * Develop logic for detecting scheduling conflicts.
   * Provide user interface for conflict adjustments.
5. **Build Dynamic Dashboard**
   * Design the real-time dashboard interface.
   * Integrate schedule and repeat task viewing.
6. **Develop Priority Algorithm**
   * Create rules for prioritizing communications.
   * Implement the algorithm to manage mail delivery priorities.
7. **Integrate All Modules**
   * Ensure seamless integration of user authentication, mailer request form, conflict resolution, dashboard, and priority algorithm.
8. **Testing and Deployment**
   * Conduct thorough testing of all modules.
   * Deploy the system and monitor performance.

### Conclusion:

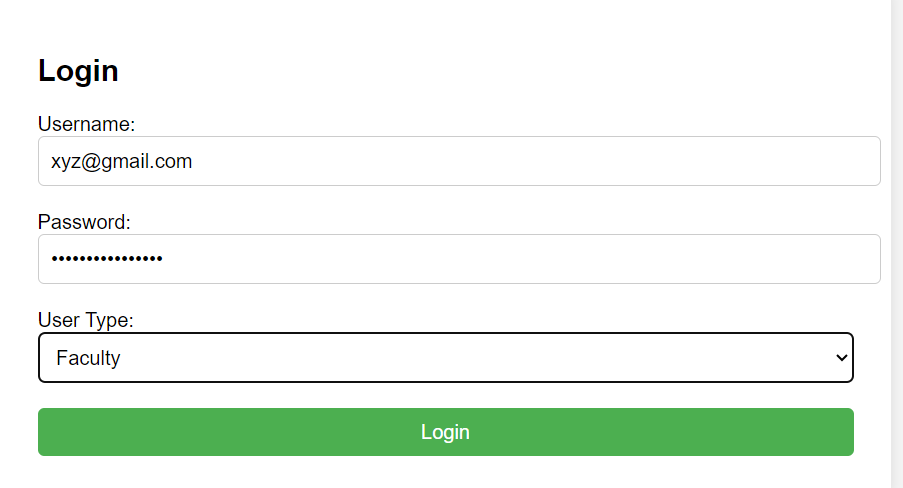
By following this project plan, we will develop a centralized mailing system with a Repeat Task module, enhancing communication within educational institutions. This system will address the current challenges of inconsistent messaging, schedule conflicts, fragmented communication, and administrative burden, ultimately improving organizational efficiency.

**PROTOTYPE OF THE PROJECT:**

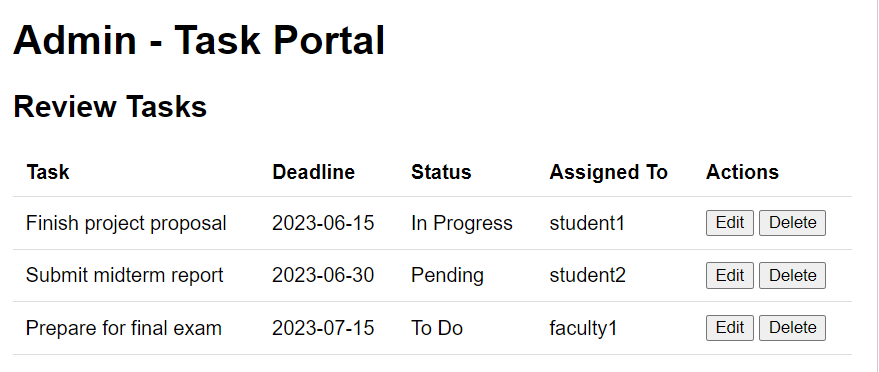
* 1. LOGIN PAGE







* 1. ADMIN’S VIEW



* 1. VIEWING ASSIGNED TASKS