

IQAC MAILER

Seat No.: 9

Name: Kesavarshini T

Project ID: 9

Problem Statement:

Design a dashboard for scheduling overlaps and communication gaps among academic teams hinder task coordination and productivity, necessitating a centralized system for streamlined management and enhanced efficiency.

1.Introduction:

1.1 Purpose of this document:

The Software Requirements Specification (SRS) document for the IQAC mailer project aims to outline the functional and non-functional requirements of the system. It serves as a comprehensive guide for the development team, stakeholders, and users, detailing the project scope, objectives, features, and constraints.

1.2 Project Proposal:

1.2.1 Scope:

The system will include features such as:

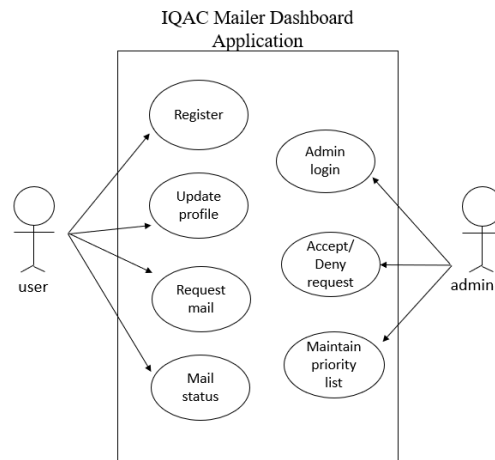
- User authentication and authorization.
- Customizable dashboard for task management.
- Mail format customization (TO, SUB, BODY).
- Admin approval/denial process based on task priority.
- Automatic mail sending functionality.
- Integration of AI for content generation (optional)

1.2.2 Objective:

- Improve task management efficiency.
- Reduce scheduling conflicts and overlaps.
- Automate mail generation and sending processes.
- Enhance user experience with an intuitive interface.
- Implement AI-driven content generation for mails.

2.Overall Description:

2.1 System Environment:

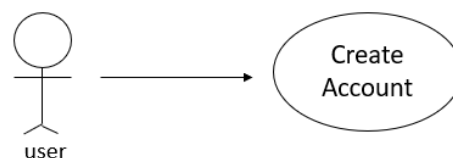


3. Functional Requirements:

- User Management:
Admin, Team Members, and System Users roles.
Authentication and authorization mechanisms.
- Dashboard:
Customizable dashboard for task viewing and management.
Task assignment and scheduling features.
- Mail Generation:
Customizable mail formats (TO, SUB, BODY).
Automatic mail sending based on admin approval.
- AI Integration:
Content generation using AI for mail bodies.

3.1 Registered Users Use cases:

3.1.1 Use Case: Account creation

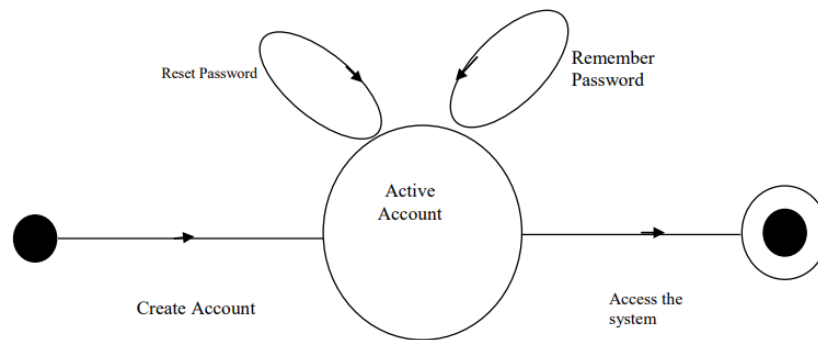


Description: Enables users to create new accounts by providing necessary information and verifying their identity for system access.

Step-by-Step Process:

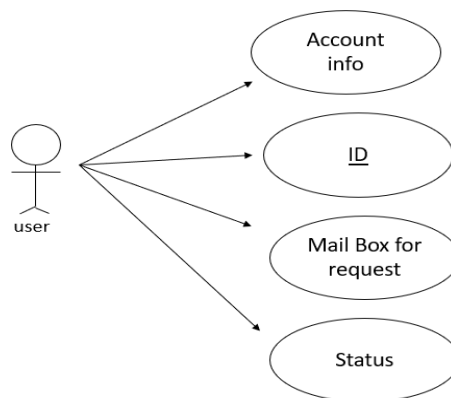
1. Account will be created by the user.
2. The system will activate the account

3. User can Reset the password
4. Finally, User now can access the system.



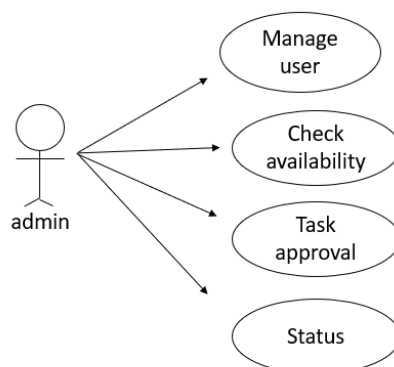
Account Creation Process

3.1.2 Use Case: User use case (After Registration)



Description: Allows registered users to log in, access system features, manage tasks, and communicate with teams efficiently.

3.1.3: Use Case: for Admin

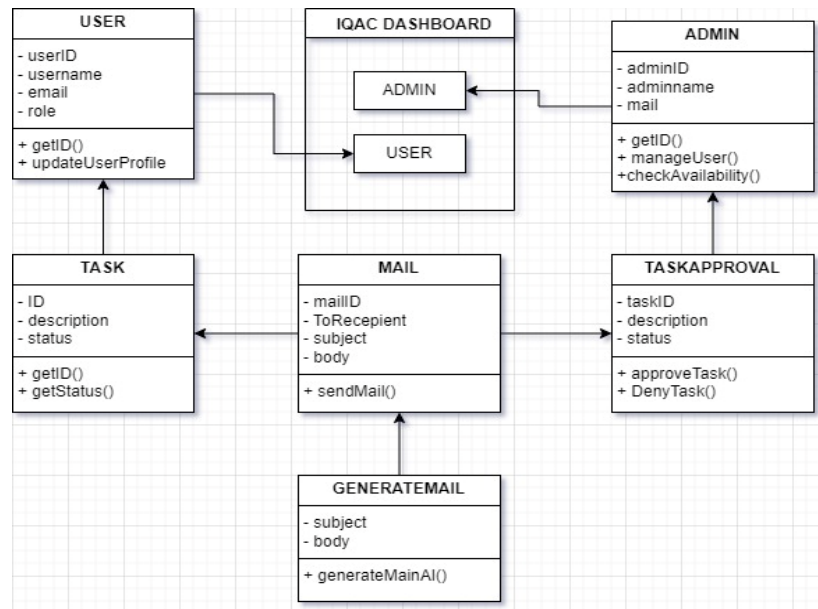


Description: Allows admin to log in, access features like availability check of the student, and approving the Task efficiently.

3.1.3 Class Diagram:

Describes the structure of the software system with classes like Software System, Requirement, User, and Admin, illustrating their attributes and methods for requirement management and user interaction.

Made this class diagram using **Draw.io**



Class diagram for the IQAC Mailer

4. System Architecture:

4.1 Frontend (React.js):

- Utilizes React.js for the user interface (UI) layer.
- Implements components, state management, and UI rendering.
- Enables dynamic, interactive, and responsive user experiences.

4.2 Backend (Node.js with Express.js):

- Node.js serves as the server-side runtime environment.
- Express.js facilitates backend development with its lightweight and flexible framework.
- Handles HTTP requests, business logic, and data processing.

4.3 Database (MongoDB):

- MongoDB is used as the NoSQL database for storing and managing data.
- Offers scalability, flexibility, and easy integration with Node.js.
- Supports efficient data retrieval, storage, and manipulation.

4.4 Open API Specifications:

- Defines API endpoints, methods (GET, POST, PUT, DELETE), request/response payloads, and authentication mechanisms in the OpenAPI specification (usually a YAML or JSON file).
- Includes details such as path parameters, query parameters, request bodies, and response codes.

5. Implementation and Testing:

5.1 Introduction:

The application was implemented using the VS Code development environment, leveraging its powerful features for code editing, debugging, and version control. MongoDB, a NoSQL database, was utilized for data storage and management, providing flexibility and scalability.

It contains Three interfaces:

- User Registration interface
- Login interface
- Main Application interface

It contains Three Dialogue Boxes:

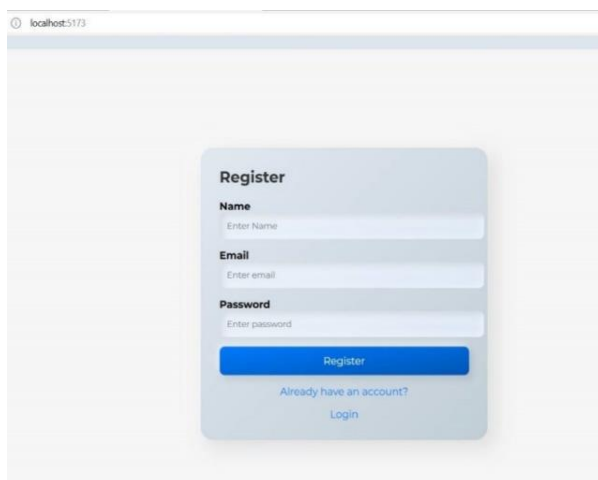
- Welcome Dialogue box
- Task Creation dialogue box
- Comment interface dialogue box

It contains Three Databases:

- User database
- Task database
- Comment database

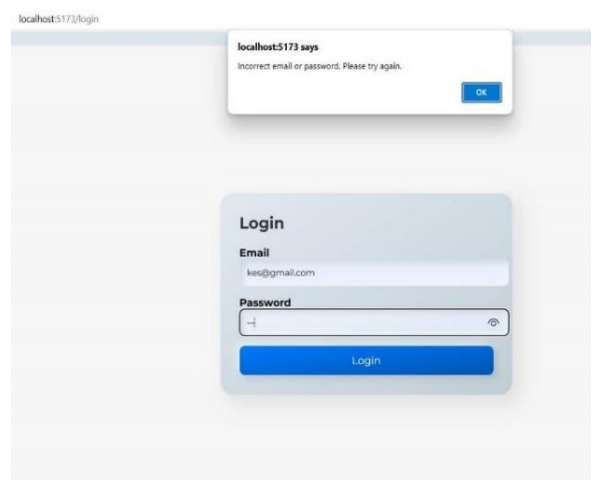
5.2 Layout:

Registration page:



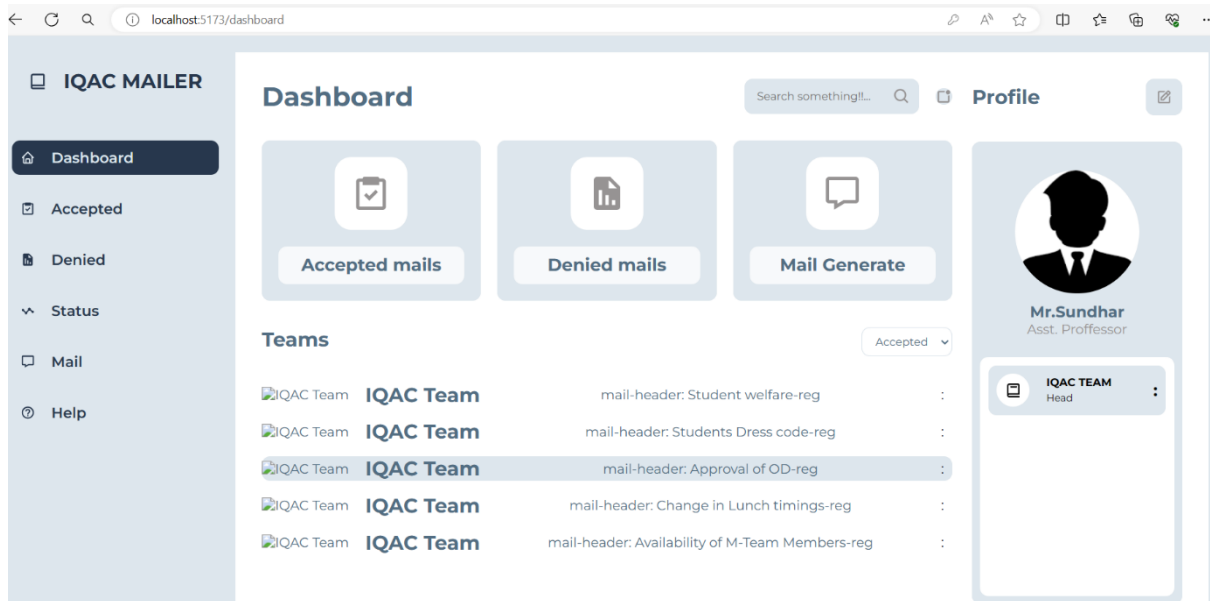
The screenshot shows a web browser window with the address bar displaying 'localhost:5173'. The main content area features a 'Register' form. The form has three input fields: 'Name' (placeholder 'Enter Name'), 'Email' (placeholder 'Enter email'), and 'Password' (placeholder 'Enter password'). Below these fields is a blue 'Register' button. At the bottom of the form, there is a link that says 'Already have an account? Login'.

Login page:



The screenshot shows a web browser window with the address bar displaying 'localhost:5173/login'. The main content area features a 'Login' form. The form has two input fields: 'Email' (placeholder 'kes@gmail.com') and 'Password' (placeholder with a password icon). Below these fields is a blue 'Login' button. At the top of the page, there is a white notification box with the text 'localhost:5173 says' and 'Incorrect email or password. Please try again.' with an 'OK' button.

Dashboard Layout:



Timeline for the Project:



Challenges faced by the Client:

- Priorities of the mail should be validated.
- What if the timeline varied for the higher priority mail?
- When both are higher priority mails, what to do?