VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI

A Project Report on

WEB APPLICATION FOR FLAT RESIDENTS

Submitted in partial fulfillment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Under

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

By

SHREYA

4SO22MC094

Department of Computer Applications

St Joseph Engineering College, Mangaluru-575028

(An Autonomous Institution)

2024

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI

A Project Report on

WEB APPLICATION FOR FLAT RESIDENTS

Submitted in partial fulfillment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Under

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

By

SHREYA

4SO22MC094

Under the Guidance of

Internal Guide:	External Guide:
Dr. Anand R	Mr.Ramesh D
Professor	Director
Department of Computer Applications	Blueline Computers
St Joseph Engineering College	Carstreet, Mangalore.
Mangaluru	

Department of Computer Applications

St Joseph Engineering College, Mangaluru-575028

(An Autonomous Institution)

2024

ST JOSEPH ENGINEERING COLLEGE, MANGALURU

(An Autonomous Institution)

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project work titled

WEB APPLICATION FOR FLAT RESIDENTS

SUBMITTED BY

SHREYA

4SO22MC094

In partial fulfillment of the requirements for award of degree of Master of Computer Applications of Visvesvaraya Technological University, is a bonafide record of the work carried out at

BLUELINE COMPUTERS

Carstreet, Mangaluru

During the academic year
2023-2024.

Dr. Anand R Professor Department of Computer Applications St Joseph Engineering College Mangaluru-575 028.	**Mr. Murari B K** Project Coordinator Department of Computer Applications St Joseph Engineering College Mangaluru-575 028.
Dr Hareesh B HOD-MCA St Joseph Engineering College Mangaluru-575 028.	**Dr Rio D’Souza** Principal St Joseph Engineering College Mangaluru-575 028.

Examiner 1.

Examiner 2.

DECLARATION

I hereby declare that the entire work embodied in this dissertation has been carried out by me and no part of it has been submitted for any degree or diploma of any university previously.

Mangalore
29 July
2024

Shreya
4SO22MC094

ACKNOWLEDGEMENT

It is my profound gratitude to thank one and all who have helped in bringing this project entitled “Web Application for Flat Residents” up to the mark. Whatever I was able to put forward in terms of my work, is only due to few people from whom I learnt a lot.

Firstly, I would like to thank St Joseph Engineering College, for giving me the opportunity to pursue and complete my academic project. My special thanks to Dr Hareesh B, Technical Guide, for his help and valuable suggestions. It was a great learning experience doing the project under his guidance.

My sincere thanks to Rev.Fr Wilfred Prakash D’Souza, Director, SJEC and Dr Rio D’Souza for their never-ending support. I express my gratitude to, Dr Hareesh B HOD-MCA, for providing me the opportunity to take up this project. I would like to thank Dr Anand R Internal Guide & Professor-MCA, for his support and encouragement and I would also like to thank Project coordinator Mr Murari B K for his constant support.

I would also like to thank all the teaching and non-teaching staff of the Department of Computer Applications and also to the other staff members of the college. The unwavering support at every stage of this project and their expertise and encouragement have been instrumental in shaping my work.

Name: Shreya

Date: 7 August 2024

TABLE OF CONTENTS

1. Introduction.. 01
2. Literature Survey 02
3. Software Requirements Specification 04
4. System Design 15
5. Detailed Design... 20
6. Implementation... 33
7. Software Testing 38
8. Conclusion 43
9. Future Enhancements 44
Appendix A: References 45
Appendix B: User Manual.. 46

INTRODUCTION

1.1 Project Description 01

LITERATURE SURVEY

2.1 Existing and Proposed System 03

2.2 Feasibility Study 03

2.3 Tools and Technologies Used 04

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction 05

3.1.1 Purpose 05

3.1.2 Definitions and Abbreviations 05

3.1.3 Target Audience 05

3.1.4 Project Scope 06

3.1.5 Benefits 06

3.1.6 References 07

3.2 Overall Description 07

3.2.1 Identification of pre-existing work 07

3.2.2 Perspective on Product 07

3.2.3 Product Attributes 08

3.2.4 End user characteristics 09

3.2.5 Operating environment 09

3.2.6 Constraints in Design and Implementation 10

3.2.7 Assumptions and Dependencies 10

3.3 Product functionality 10

3.3.1 User module 10

3.3.2 Manager module 10

3.3.3 Owner module 11

3.4 External interface requirements 11

3.4.1 User interfaces 11

3.4.2 Hardware interfaces 12

3.4.3 Software interfaces 12

3.4.4 Communication interfaces 12

3.5 Other non functional requirements 12

3.5.1 Performance requirements 12

3.5.2 Safety requirements 12

3.5.3 Software quality attributes 12

3.5.3.1 Reliability 12

3.5.3.2 Maintainability 13

3.5.3.3 Portability 13
3.5.3.4 Usability 13
3.5.3.5 Flexibility 14
3.6 Specific Requirements 13
3.6.1 Operating Environment 13
3.6.1.1 Hardware 13
3.6.1.2 Software 14
3.7 Delivery plan 14

SYSTEM DESIGN

4.1 System design 15

4.1.1 Introduction 15

4.1.2 Scope 15

4.1.3 Audience 15

4.2 Software product architecture 16

4.2.1 Architectural design 16

4.2.1.1 View layer or presentation layer 16

4.2.1.2 Business layer 16

4.2.1.3 Access layer 16

4.3 Component Architecture 16

4.3.1 User Interfaces 16

4.4 Dataflow diagram 17

4.4.1 Context flow diagram 17

4.4.2 Level 1 DFD 18

DETAILED DESIGN

5.1 Use case diagram 19

5.2 Sequence diagram 20

5.2.1 Sequence diagram for admin 20

5.2.2 Sequence diagram for manager 21

5.2.3 Sequence diagram for user 21

5.3 Activity Diagram 22

5.3.1 User Operations 22

5.4 Database Design 22

5.4.1 ER Diagram 25

IMPLEMENTATION

6.1 Introduction 26
6.2 Pseudo codes.......... 26
6.2.1 Pseudo code for login page.......... 26
6.2.2 Pseudo code for register page 27
6.2.3 Pseudo code for Request Approval 27
6.2.4 Pseudo code for Adding Event.......... 28

SOFTWARE TESTING

7.1 Introduction 29
7.2 Testing objective 29
7.3 Unit testing 30
7.3.1 Unit test cases 30
7.4 Integration testing 31
7.4.1 Integration test cases 31

ABSTRACT

The project centers on the development of a comprehensive web-based application tailored for the efficient management of residential communities. Utilizing Django as the backend framework and employing HTML, CSS, and Bootstrap for the frontend, the application is designed to streamline communication, event management, and overall community administration. The system is structured to serve three primary user roles: administrators, community managers, and residents.

Administrators, typically the owners or main overseers of the community, are equipped with tools to manage event categories, maintain a list of registered users, and oversee the activities of community managers. The community managers, who are responsible for organizing and managing events, have the ability to create, approve, or cancel events, as well as track event participation and generate reports. Residents, as the end-users, can easily book events, view their event history, and receive notifications about upcoming community activities.

This application aims to replace traditional, often fragmented, methods of community management with a centralized platform that offers real-time data access, intuitive user interfaces, and robust security measures. The project emphasizes user-friendliness, ensuring that even non-technical users can navigate the system with ease. Additionally, the system is designed to be scalable, accommodating the growth of the community and the potential addition of new features in the future.

By providing a centralized hub for all community-related activities, the application fosters greater engagement among residents, improves operational efficiency for administrators and managers, and enhances the overall living experience within the community. This project not only addresses the immediate needs of residential community management but also lays the groundwork for future enhancements, such as mobile app integration, advanced analytics, and enhanced security features. The successful implementation of this project will contribute significantly to creating a more organized, connected, and efficiently managed residential environment.

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

The Web Application for Flat Residents is crafted to enhance community engagement among flat residents by providing an efficient platform for managing and participating in local events. Developed using Django, this application serves as a centralized hub where residents can stay informed about various community gatherings such as Diwali celebrations, Christmas parties, and Eid festivities.

The core functionality of the application revolves around a streamlined event management system. Admins can effortlessly create, update, and delete events, ensuring that residents receive timely information about upcoming activities. The application features an intuitive interface for admins to activate or deactivate events, thus allowing for dynamic event scheduling and management.

Residents benefit from an easy-to-use RSVP system that allows them to express their interest in attending events. Once they RSVP, their attendance is tracked in the system, providing valuable data for event planning and management. To incentivize active participation, the application includes a reward mechanism where residents who attend the most events are recognized with prizes.

This feature not only motivates residents to engage more but also fosters a greater sense of community involvement. In addition to event management and tracking, the application supports automated notifications. Residents receive timely event details ensuring they are well-informed and can plan their participation accordingly. Furthermore, the application includes functionality for admins to upload and display photos and videos from past events.

This feature helps residents relive past activities and enhances community bonding by showcasing shared experiences. Residents have personalized profiles where they can view their event participation history and any prizes they have earned. This not only provides a personal touch but also helps in tracking individual involvement over time.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING AND PROPOSED SYSTEM

AExisting community event management systems generally offer basic functionalities such as event creation, notifications, and RSVP management. While effective for listing events and managing attendance, these systems often lack advanced features that drive higher user engagement. They usually provide limited interaction opportunities and fail to incorporate personalized notifications or detailed attendance tracking.

The proposed system addresses these gaps by leveraging Django to create a more dynamic platform. It enhances traditional features with automated email and SMS reminders, ensuring residents are consistently updated about upcoming events. A notable advancement is the integrated attendance tracking and reward mechanism, where residents who participate in the most events receive prizes, thus encouraging increased involvement.

Additionally, the system supports uploading and displaying photos and videos from past events, which helps build a stronger community connection. Personalized user profiles allow residents to track their event participation and rewards. Overall, the proposed system offers a more engaging and comprehensive approach to community event management compared to existing solutions.

2.2 FEASIBILITY STUDY

The feasibility study evaluates the practicality of implementing the Web Application for flat residents, focusing on technical, economic, operational, and legal aspects.

Technical Feasibility

The project is technically feasible given the robust capabilities of Django, which provides a solid framework for developing a scalable and secure web application. The proposed features, are notified upcoming event, attendance tracking, and rewards, align well with Django’s strengths.

Economic Feasibility

The project’s costs are manageable within a typical budget for web development. Expenses will include development time, server hosting, and potentially. The anticipated benefits, such as improved community engagement and the potential for increased participation, outweigh these costs, making the project economically viable.

Operational Feasibility

The application is designed to be user-friendly for both administrators and residents. With intuitive interfaces and automated features, it should facilitate smooth operation and minimal training requirements. The reward system and notifications are expected to enhance user engagement, aligning with operational goals of increasing participation and community involvement.

Legal Feasibility

The project will adhere to relevant data protection to ensure that user information is handled securely and ethically. Legal compliance will be maintained through secure data storage practices and user consent mechanisms.

2.3 TOOLS AND TECHNOLOGIES USED

The Web application for Flat residents is being developed using Microsoft Visual Studio Code. The development of the website relies on Django, while the frontend is constructed using HTML, CSS, Bootstrap and JavaScript. The server being utilized is Xampp Server, and MySQL database is employed for this web page.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 INTRODUCTION

3.1.1 PURPOSE

The main goal of this Software Requirements Specification (SRS) is to transform the client's ideas into a structured document. Its purpose is to meet the client's needs and document user information. Since it is intended for developer use and will serve as the basis for validating the eventual system delivery, any future alterations to the requirements must follow the formal change approval process.

3.1.2 DOCUMENT ABBREVATIONS/ DEFINITIONS

- **CFD**: Context Flow Diagram
- **HTTP**: Hypertext Transfer Protocol
- **PHP**: PHP Hypertext Pre-processor
- **CSS**: Cascading Style Sheet
- **ER**: Entity Relationship Diagram
- **RAM**: Random Access Memory
- **DFD**: Data Flow Diagram
- **SRS**: Software Requirements Analysis and Specification
- **SQL**: Structured Query Language

3.1.3 TARGET AUDIENCE

3.1.3.1 Developer

The developer should refer to this document to make the necessary revisions before moving on to the next step of the development process if the system needs any changes or updates.

3.1.3.2 Administrator

Better security features are provided by them, and they are also in charge of adding any new users to the system.

3.1.3.3 User

The system's users would be fully aware of the conditions necessary for it to function.

3.1.4 PROJECT SCOPE

The Software Requirements Specification (SRS) document, the boundaries and deliverables of the project. It details what the project will achieve and the constraints within which it will operate. It includes the design, development, and deployment of a web-based platform aimed at enhancing community event management for residential flats. This application will serve as a central hub for administrators and residents to efficiently handle and participate in community events.

3.1.5 BENEFITS

It offers a range of benefits designed to enhance community engagement and streamline event management for residential flats. These benefits encompass improvements in user experience, operational efficiency, and community involvement.

- Enhanced Community Engagement
- Streamlined Event Management
- Increased Participation
- Efficient Communication
- Personalized User Experience

3.1.6 REFERENCES

1. Software Engineering, 5th Edition, Ian Somerville, Addison Wesley Longman Publishing

3.2 OVERALL DESCRIPTION

3.2.1 IDENTIFICATION OF PRE-EXISTING WORK

Web Application for Flat Residents project draws on several well-established concepts and technologies commonly used in web application development. Django, a high-level Python web framework, provides the foundation for the application, offering tools and libraries to build and maintain robust web applications efficiently. The application structure follows the Model-View-Template (MVT) architectural pattern, a core feature of Django that separates business logic from user interface concerns. It utilizes Django's built-in authentication system for user management, ensuring secure login and registration processes. The feedback system is inspired by common user feedback mechanisms found in many web platforms, allowing for continuous improvement based on user input. Additionally, the project incorporates standard web development practices such as form handling, file uploads, and session management to create a seamless user experience. These pre-existing elements are integrated and customized to meet the specific needs of community event management, resulting in a tailored solution for residential communities.

3.2.2 PERSPECTIVE ON PRODUCT

The web application is designed to enhance community engagement and streamline event management within residential complexes. By providing a centralized platform, it simplifies the process of organizing and participating in events, benefiting both administrators and residents. Admins can efficiently manage events, upload media, and send notifications, ensuring all residents are well-informed and engaged. The user-friendly interface allows residents to easily view upcoming events, register their attendance, and provide feedback. The application promotes active participation and a sense of community among residents by facilitating smooth interactions and transparent communication. The feedback system enables continuous improvement of events based on resident input, ensuring their needs and preferences are addressed. The use of Django ensures a secure, scalable, and maintainable system, capable of handling the dynamic requirements of community event management. This product ultimately aims to foster a more connected and engaged residential community, making event management hassle-free and interactive. It serves as a valuable tool for both administrators and residents, enhancing the overall living experience within the community.

3.2.3 PRODUCT ATTRIBUTES

This website's user interface is simple to use.

User Registration and Login:

- Secure authentication system for registering and logging in users.
- Ensures only authorized access to the application.

Admin Dashboard:

- Centralized interface for managing events and user registrations.
- Features for uploading images and videos related to events.
- Tools for sending notifications to users.

User Dashboard:

- User-friendly interface for viewing upcoming and past events.
- Allows users to register for events and update their attendance.

Event Management:

- Functionality for admins to create, update, and delete events.
- Includes options for setting event details such as name, date, and description.

Feedback System:

- Allows users to provide feedback on events they attended.
- Feedback is used for improving future events.

Notification System:

- Sends notifications to users about upcoming events and important announcements.
- Ensures users are well-informed and engaged.

Profile Management:

- Users can manage their profiles, update personal information, and view participation history.

Media Uploads:

- Admins can upload images and videos to enhance event descriptions.
- Supports engaging and informative event content.

3.2.4 END USER CHARACTERISTICS

The end users of this web application can be categorized into two primary groups: administrators and residents.

3.2.4.1 Admin

- Typically community managers or designated personnel responsible for organizing events and managing community activities.
- They are tech-savvy and comfortable using web-based platforms to handle administrative tasks.
- Require tools to create, update, and delete events, manage user registrations, and upload event-related media.
- Need access to a notification system to communicate effectively with residents.
- Value feedback from residents to improve event planning and execution.

3.2.4.3 Residents

- Members of the residential community who participate in events and activities.
- Have varying levels of technical proficiency, from basic to advanced.
- Seek a simple and intuitive interface to view event details, register for events, and provide feedback.
- Value timely notifications and updates about upcoming events and important community announcements.
- Appreciate a personalized experience, such as managing their profiles and viewing their event participation history.

3.2.5 OPERATING ENVIRONMENT

3.2.5.1 General Constraints

- All required fields in forms must be filled out with valid and accurate information to maintain data integrity.
- Validation checks should be implemented to ensure correct data entry by users.
- Only administrators have access to the event management and media upload functionalities.

- Residents can only access their profile, view events, register for events, and provide feedback.

3.2.6 CONSTRAINTS IN DESIGN AND IMPLEMENTATION

- Implement secure authentication mechanisms to ensure only registered users can access the application.
- Authorization controls must restrict access to admin functionalities, ensuring only authorized personnel can manage events and media uploads.
- Ensure all forms have necessary validation checks to maintain data integrity.
- Prevent the submission of incomplete or incorrect data to avoid issues in event management and user interactions.

3.2.7 ASSUMPTIONS AND DEPENDENCIES

- It is assumed that users, both admins and residents, have basic proficiency in using web applications and can navigate the user interface without extensive training. User should be knowledgeable with computer fundamentals.
- The application assumes reliable internet connectivity for accessing the platform, as it is web-based and requires an active connection for most functionalities.
- The application relies on a stable hosting environment capable of supporting Django applications, ensuring uptime, scalability, and performance.
- Various third-party libraries and packages are used for additional functionalities, such as form validation, media handling, and security enhancements

3.3 PRODUCT FUNCTIONALITY

3.3.1 ADMINISTRATOR MODULE

3.3.1.2 Event Management

- **Create Events:** Admins can create new events by providing details such as the event name, date, description, and any associated media files (images or videos).
- **Update Events:** Admins can edit existing events to modify details or update media content.
- **Delete Events:** Admins can remove events that are no longer relevant or needed.

3.3.1.2 User Management:

- **User Registration:** Admins can register new users by entering their personal details and assigning roles.
- **User Activation/Deactivation:** Admins can activate or deactivate user accounts to manage participation and access.

3.3.1.3 Notification System:

- **Send Notifications:** Admins can send notifications to all users or specific groups about upcoming events or important announcements.
- **Scheduled Notifications:** Admins can schedule notifications to be sent at a future date and time.

3.3.1.4 Feedback Management:

- **View Feedback:** Admins can view feedback provided by users for various events.
- **Analyze Feedback:** Admins can analyze feedback to make informed decisions about future events and improvements

3.3.2 USER MODULE

3.3.2.1 User Registration and Login

- **Register:** Residents can sign up by providing necessary details and creating an account.

- **Login:** Residents can log in using their credentials to access the platform.

3.3.2.2 Event Participation:

- **View Events:** Residents can view a list of upcoming and past events with details and media content.
- **Register for Events:** Residents can register their attendance for events and indicate the number of attendees.
- **Feedback Submission:** Residents can provide feedback on events they have attended, helping to improve future events.

3.3.2.3 Profile Management:

- **Edit Profile:** Residents can update their personal information and contact details.
- **View Participation History:** Residents can view a history of events they have registered for and attended.

3.3.2.4 Notification System:

- **Receive Notifications:** Residents receive notifications about upcoming events, changes to events, and important community announcements.

3.3 EXTERNAL INTERFACE REQUIREMENTS

3.3.1 USER INTERFACES

- A web browser can be used by the user to access the website.
- Links to other pages are on the home page.
- He/she can submit a request to attend the events and view the events that he/she can participate in.
- The login form is given a start validation. The system user is granted authorization after a successful validation.

3.3.2 HARDWARE INTERFACES

Any device with a web browser preinstalled.

3.3.3 SOFTWARE INTERFACES

Requires assistance from web browsers.

3.3.4 COMMUNICATION INTERFACES

The web application uses several communication interfaces to ensure efficient interaction between the client, server, and external systems. Primarily, it employs HTTP/HTTPS protocols for secure data exchange between the web browser and the server. HTTPS ensures that sensitive information, such as login credentials and personal details, is encrypted during transmission. The server provides RESTful API endpoints for operations like user registration, login, event management, feedback submission, and notification handling. These endpoints handle HTTP requests and return appropriate responses, facilitating smooth client-server communication

3.4 OTHER NON FUNCTIONAL REQUIREMENTS

3.4.1 PERFORMANCE REQUIREMENTS

The possibility of a system crash, which results in data loss, exists. There will be a backup to recover the data to prevent this.

The performance of the entire system should be more quick and error-free, with built-in

- Fast response times for user interactions such as logging in, viewing events, and submitting feedback, ideally within a few seconds.
- Efficient database operations with optimized queries for quick data retrieval and updates, ensuring minimal latency. A minimum 56 kbps bandwidth internet.
- We need a browser with IE6 or a higher version in order to access this page.

3.4.2 SAFETY REQUIREMENTS

A user can access the system only after logging in successfully. Only authorized users are able to use the application, preventing unauthorized access to the system.

3.4.3 SOFTWARE QUALITY ATTRIBUTES

3.4.3.4 Reliability

Verify the product's durability to ensure it can endure any challenges. Should continuously produce accurate findings. The project's capacity to function in various working environments and situations serves as a measure of product reliability.

3.4.3.5 Maintainability

Iterations of the product should be straightforward to maintain. It should be straightforward to add code to the existing system for development and to upgrade the system for new features and developing technologies. It need to be affordable and easy to maintain. Updating the program or fixing faults in the system is straightforward.

3.4.3.6 Portability

This can be evaluated in terms of the financial difficulties, challenges with technology, and behavioral difficulties associated with porting.

3.4.3.7 Usability

This can be evaluated using the ease of use. The software ought to be easy to use. Learning ought to be easy. It need to be simple to navigate. The system must be :

3.4.3.7.1 Easy to operate, prepare the input, and evaluate the result The application should have a user-friendly and intuitive interface that is easy to navigate for all users, including both administrators and residents..

3.4.3.7.2 Provide user interface guidelines or standards that are compatible with them for other frequently used systems.

3.4.3.7.3 Users should be able to customize their profiles and settings to suit their preferences.

3.4.3.8 Flexibility

Must be able to change and adapt. able to be altered to interact with various things. Connecting to other widely used third-party components ought to be simple.

3.5 SPECIFIC REQUIREMENTS

3.5.1 OPERATING ENVIRONMENT

3.5.1.4 Hardware

Processor	:	Minimum Intel Pentium 4 processor
RAM	:	Minimum 1GB
Hard Disk	:	Minimum 10GB

3.5.1.5 Software

Frontend	:	HTML, CSS, Bootstrap, JavaScript
Backend	:	Python
Database	:	MySQL

3.6 DELIVERY PLAN

Initially, the project kickoff and requirement gathering will take place over the first two weeks. The design phase, including UI/UX and database design, will follow for three weeks. Development will then proceed over six weeks, focusing on backend and frontend development and integration. Testing, including unit, integration, and user acceptance testing, will occur in the following three weeks, with bug fixes addressed promptly. Finally, deployment preparation and deployment to the staging environment will happen in the last two weeks, ensuring the application is ready for production.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM DESIGN

4.1.1 INTRODUCTION

System components, such as its architecture, modules, and data, are specified using system design.It's also referred to as creating a subapplication on occasion. System design places a focus on selecting the necessary segments, and it recommends that these modules' terms be consistent.

4.1.2 SCOPE

The depth of this text is crucial for helping end users understand the application in a methodical way. It goes on to describe the system's context flow in the areas where we can see it. Data flow of the system, where we may easily determine the data flow's direction. It also explains the connections between each module of the software and use-case realization to help the person who created this application adopt the best method while programming..

4.1.3 AUDIENCE

The system design audience includes developers who need technical specifications to build and integrate the application components. Project managers require an overview of the system design for planning, monitoring progress, and resource allocation. Administrators need to understand the system's capabilities for managing events and user interactions. While end users (residents) are not directly involved in the design, their usability needs are crucial. The quality assurance (QA) team needs to understand the architecture to create effective test plans and ensure the application meets all requirements.

4.2 SOFTWARE PRODUCT ARCHITECTURE

4.2.1 ARCHITECTURAL DESIGN

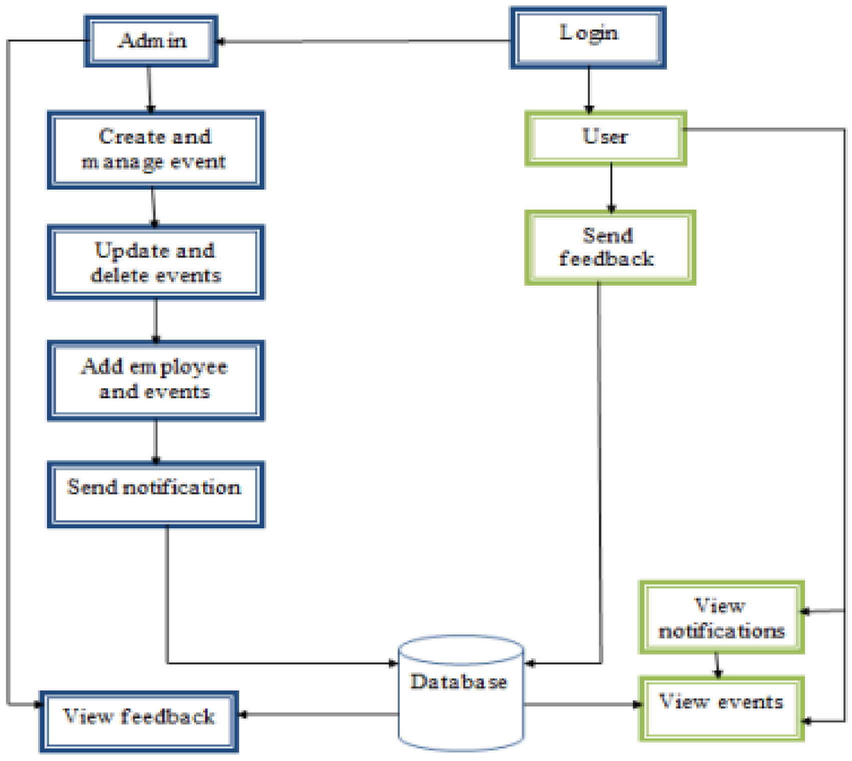


Figure 4.1: Web application for Flat residents Architectural Design

4.2.1.1 View Layer or Presentation Layer

This layer consists of the user interface and client-side logic, built using HTML, CSS, JavaScript, and Bootstrap for responsive design. Users interact with the application through a web browser, accessing various features such as event viewing, profile management, and notifications.

4.2.1.2 Application Layer

The application layer houses the core logic and handles communication between the presentation layer and the data layer. It is built using Django, a Python-based web framework.

4.2.1.3 Data Layer

The data layer is responsible for data storage, retrieval, and management. It uses a relational database, such as PostgreSQL or MySQL, managed through Django's ORM. The database schema includes tables for users, events, notifications, and feedback, ensuring efficient and secure data operations.

.

4.3 COMPONENT ARCHITECTURE

4.3.1 USER INTERFACE

The system's interaction with the user and how quickly they learn how it works are the main goals of the user interface.

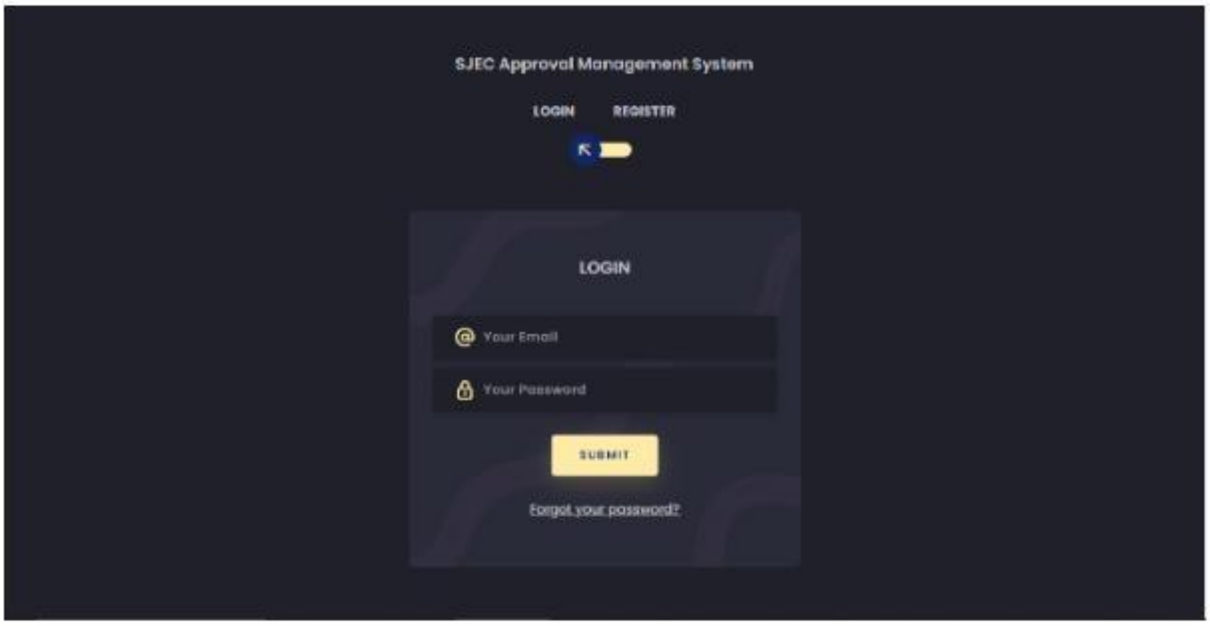


Fig 4.2 Login User Interface

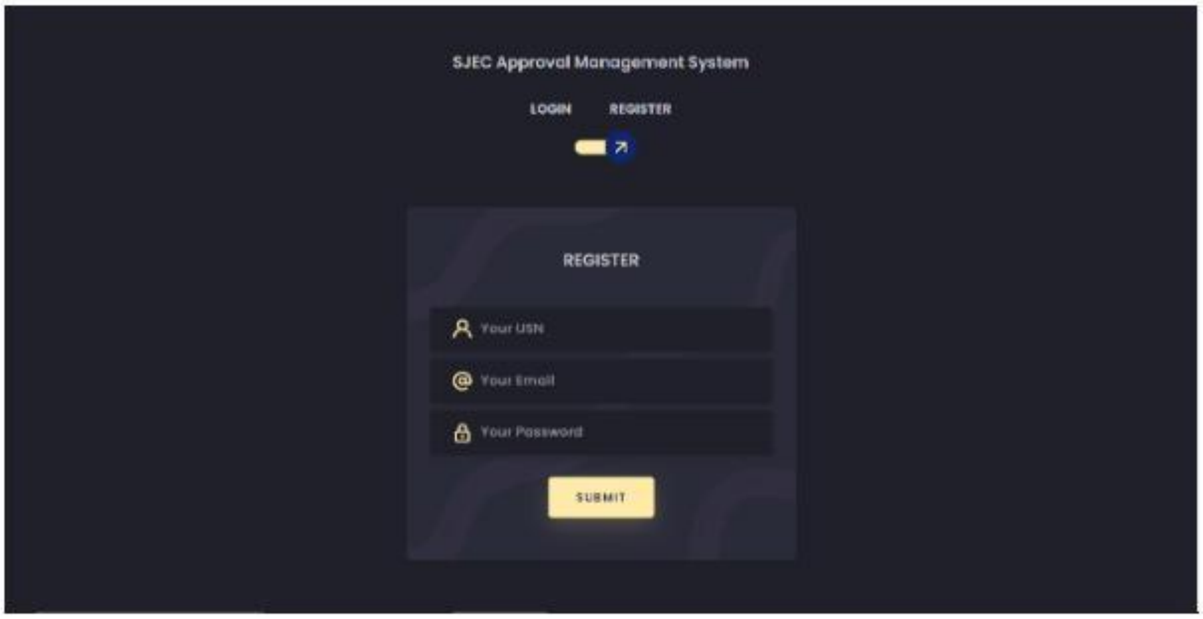


Fig 4.3 Register User Interface

4.4. DATA FLOW DIAGRAM

4.4.1 CONTEXT FLOW DIAGRAM

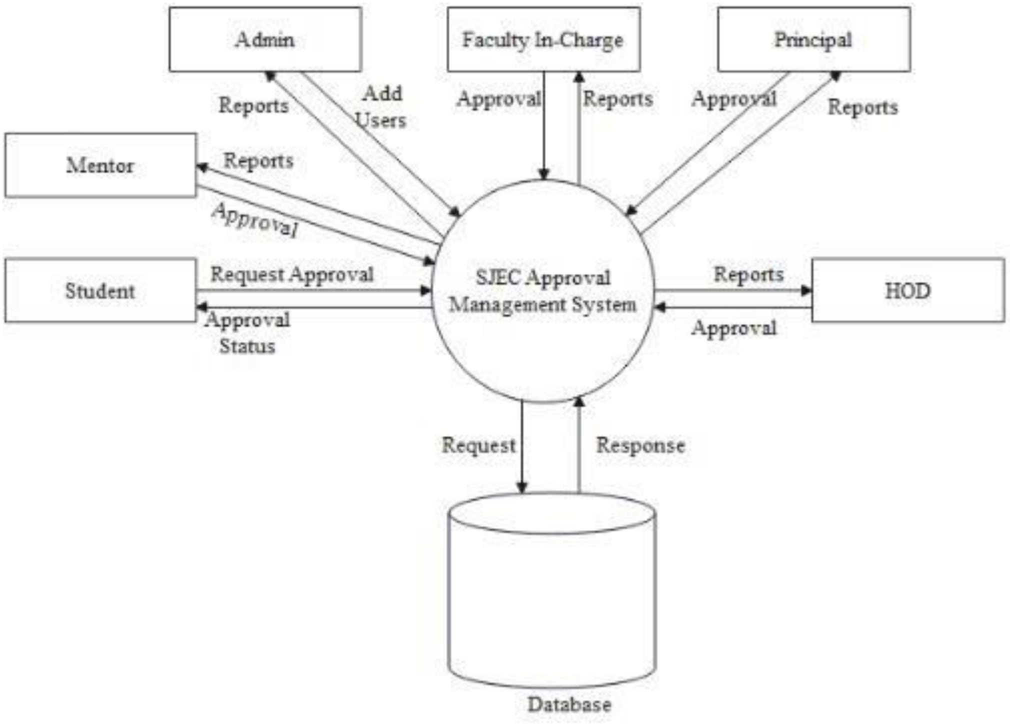
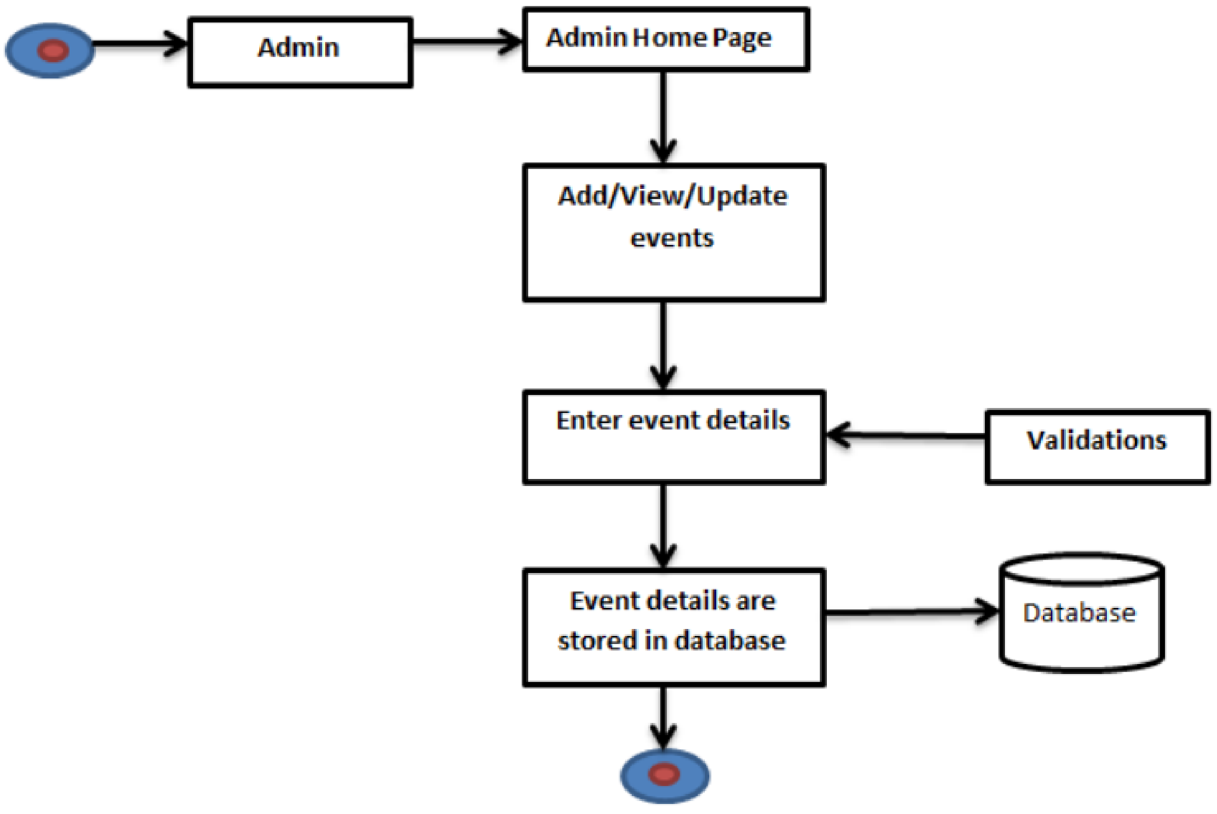
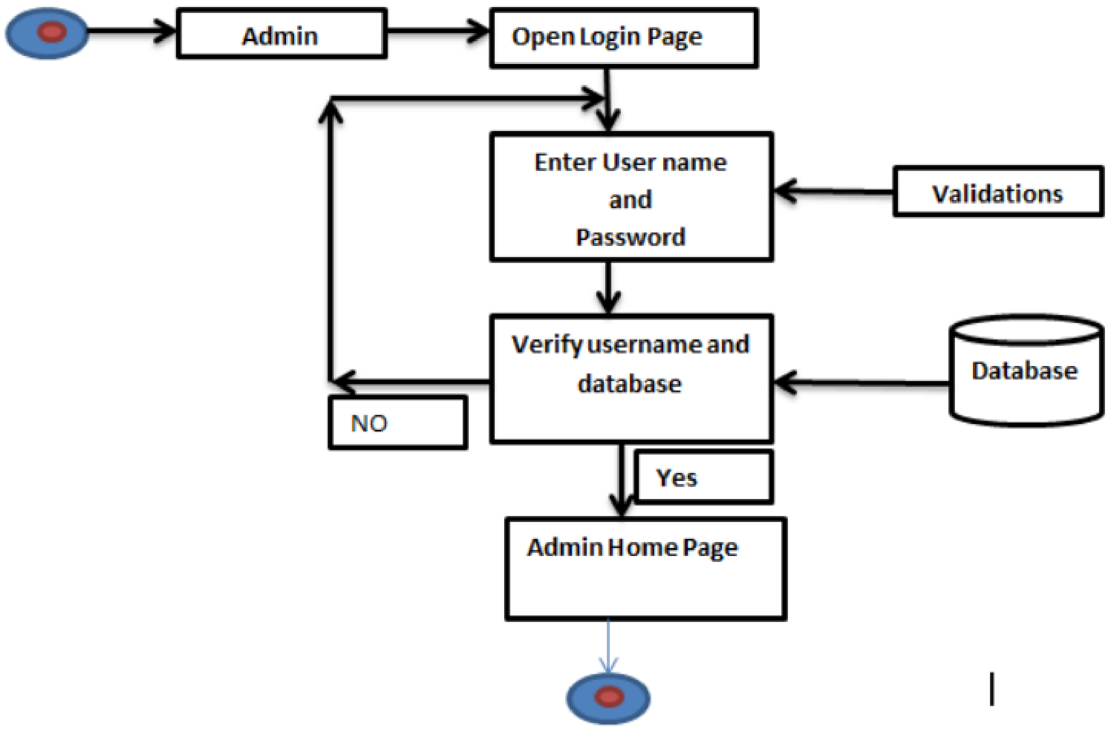


Fig 4.4 Context Flow Diagram for SJEC Approval Managemnt System

4.4.2 DFD


Admin
Admin Home Page
Add/View/Update events
Enter event details
Validations
Event details are stored in database
Database


Admin
Open Login Page
Enter User name and Password
Validations
Verify username and database
Database
NO
Yes
Admin Home Page

CHAPTER 5

DETAILED DESIGN

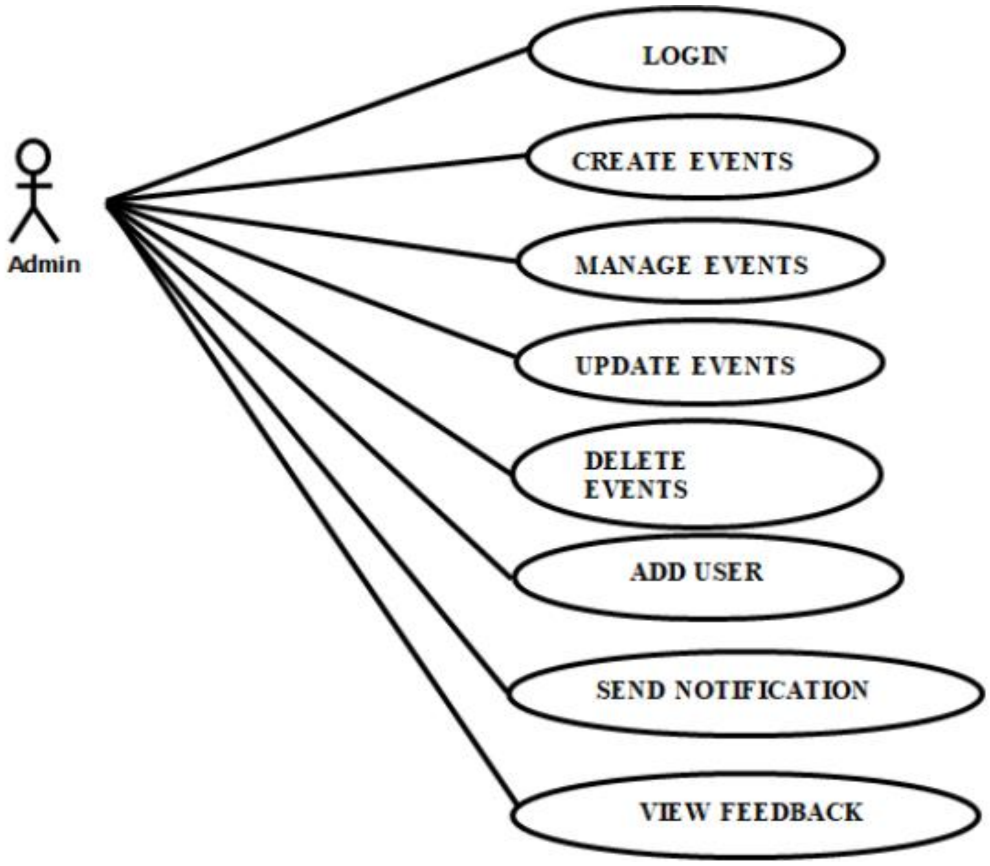
5.1 USE CASE DIAGRAM

A use case illustration for the SJEC Approval Management System illustrates how administrators and users interact with the system.

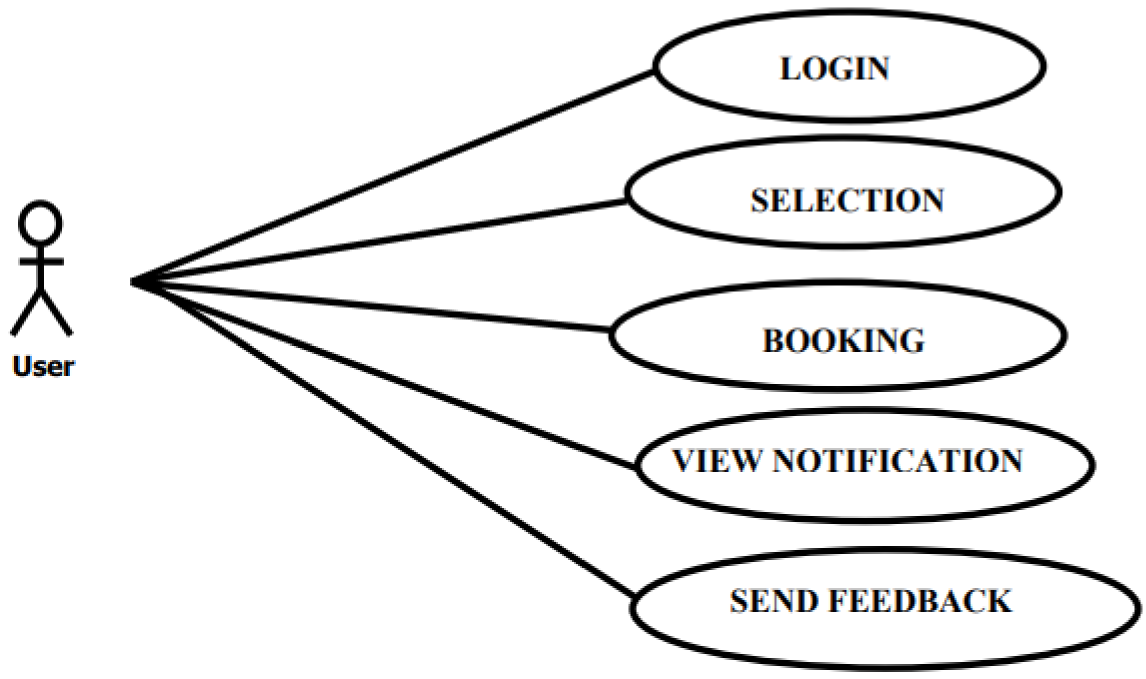
System: The system, which is simply the SJEC Approval Management website, is shown as a rectangle.

Actor: Each actor is depicted as a stick person; they are both Users and Admin..

Use case: The operations carried out on the SJEC Approval Management system are represented by an ellipse in the Use Case..



Admin side Use case diagram

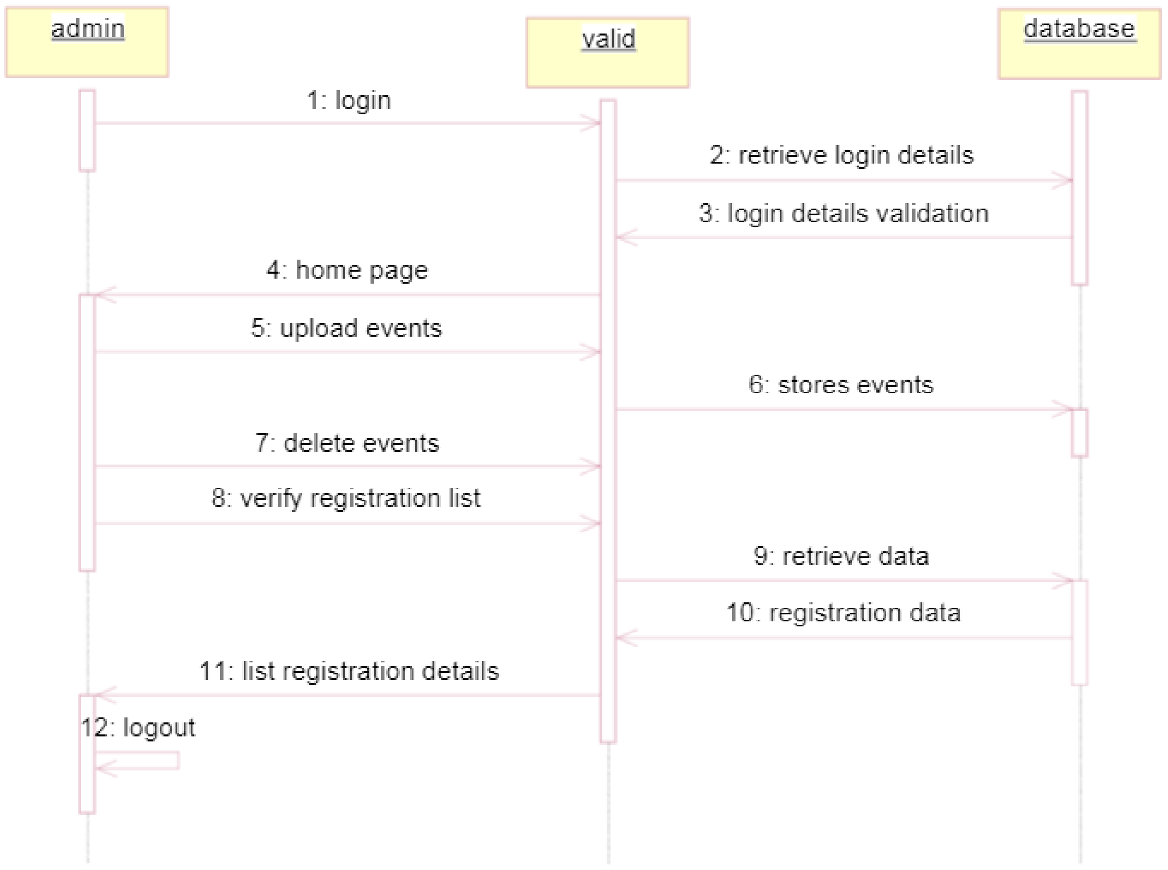


User side use case diagram

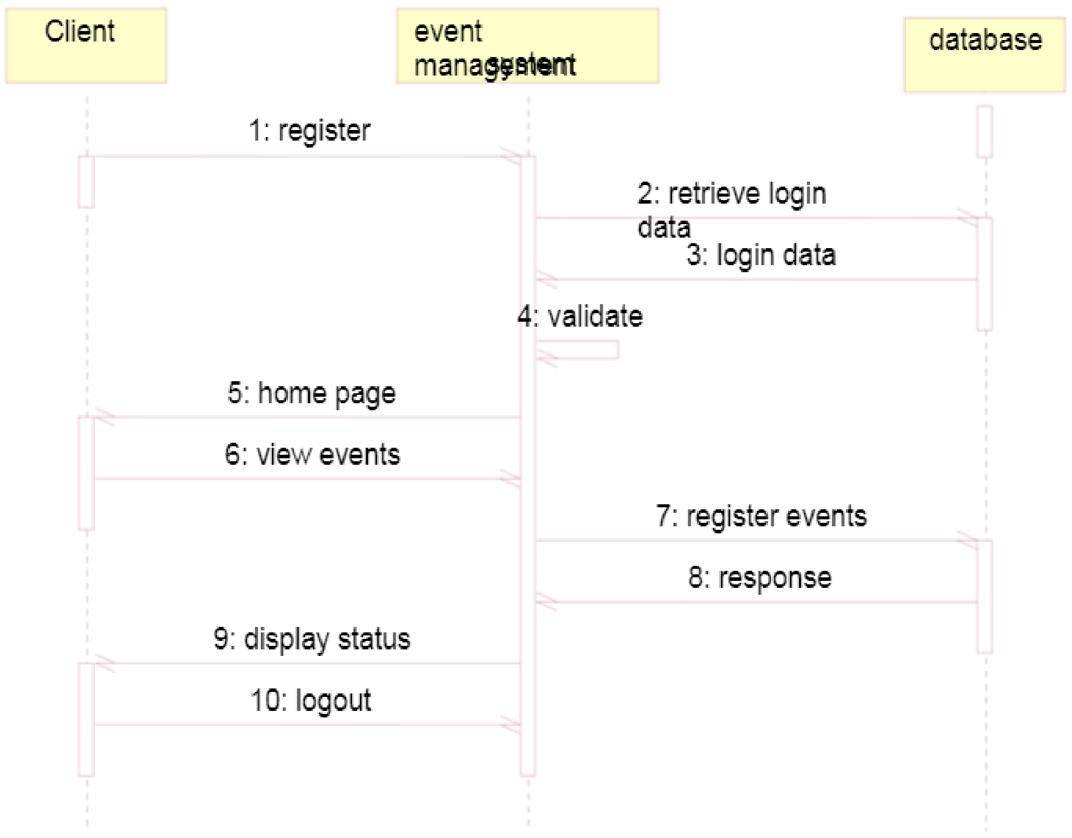
5.2 SEQUENCE DIAGRAM

The sequence diagram illustrates how the objects interact with one another over time. It is used to show how the objects can communicate and exchange messages. A sequence diagram is also known as an event diagram.

5.2.1 SEQUENCE DIAGRAM FOR ADMIN



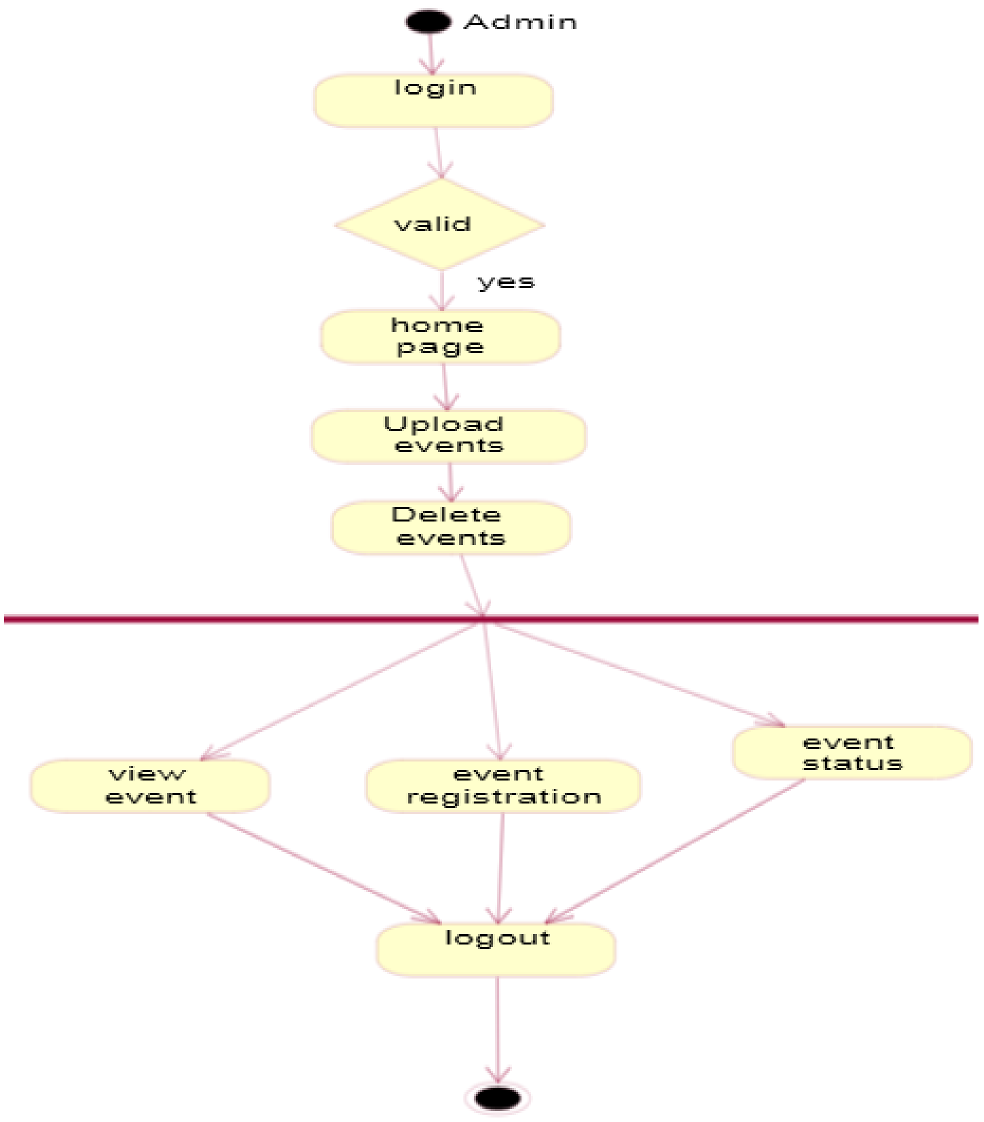
5.2.2 **SEQUENCE DIAGRAM FOR USER**



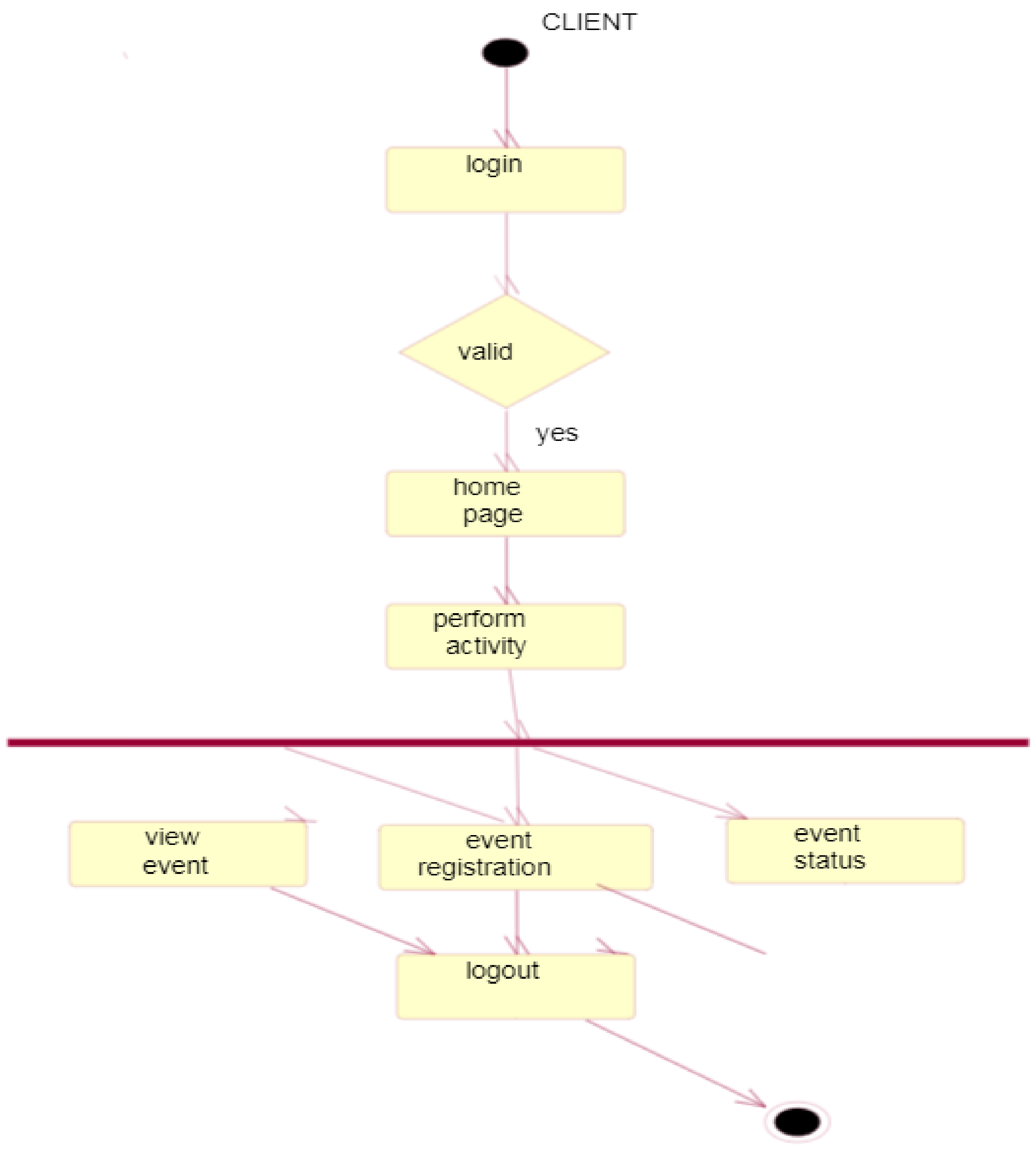
5.3 ACTIVITY DIAGRAMS

An activity diagram is like a map that shows how one action connects to the next in a project. It's a way to visualize the steps in a process, like a recipe or a set of instructions. This helps us understand how a system or project functions, like putting together the pieces of a puzzle to see the big picture.

5.3.1 Activity Diagram for Admin



5.3.2 **Activity Diagram for User**



5.4 DATABASE DESIGN

In the web application for Flat residents database, there are many tables, but the ones that matter the most are shown here.

Table 5.1:User Table

Attribute	Data Type	Constraints	Description
user_id	int	Primary Key	User id
username	varchar(10)	Not Null	User Name

password	varchar(10)	Not Null	Password
email	varchar(20)	Not Null	Email
full_name	varchar(20)	Not Null	Full Name
phone_number	varchar(20)	Null	Phone Number
is_active	boolean	Default false	Is Active/Deactive

Table 5.2:Events Table			
Attribute	**Data Type**	**Constraints**	**Description**
event_id	Int	Primary Key	Event id
event_name	varchar(100)	Not Null	Event name
event_date	date	Not Null	Event date
event_time	time	Not Null	Event time
event_location	varchar(255)	Not Null	Event location
event_description	Text	Not Null	Event description
event_media	varchar(255)	Null	Media
is_active	boolean	Defult false	Is Active/Deactive

Table 5.3:User response Table			
Attribute	**Data Type**	**Constraints**	**Description**
response_id	int	Primary Key	Response ID
event_id	int	Foreign Key	Event Name
user_id	int	Foreign Key	User ID
attendance_status	varchar(20)	Not null	Attendence Status
num_attendees	int	null	Attendees

Table 5.4:ADMIN Table

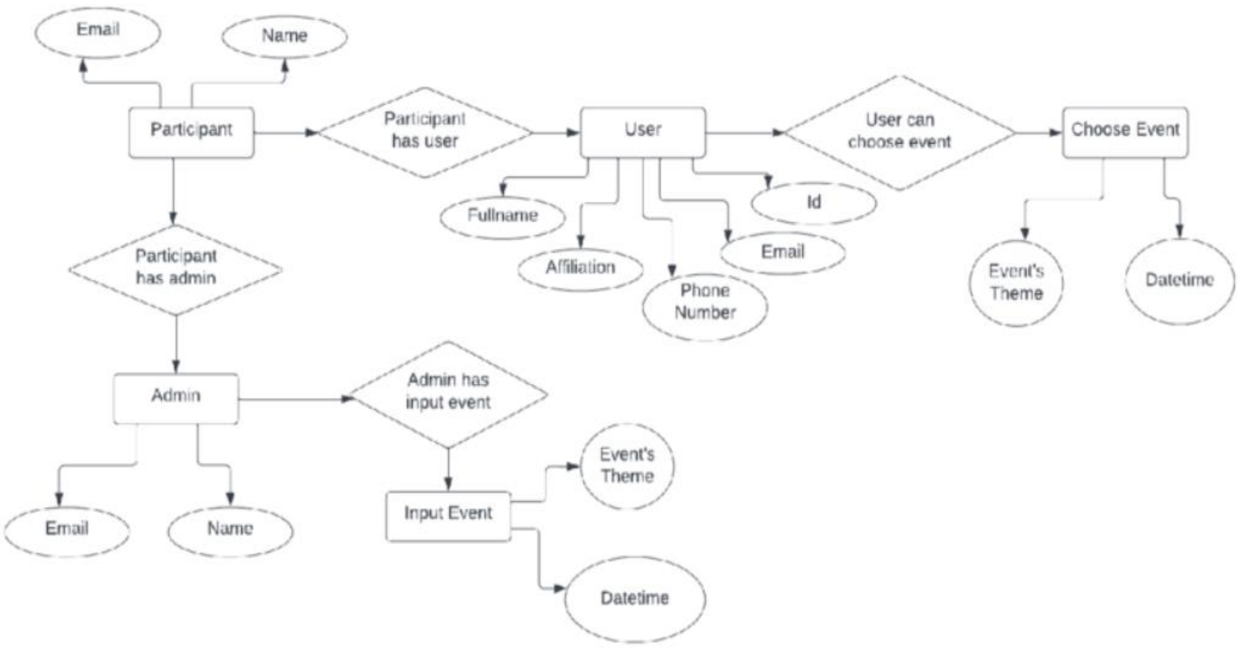
Attribute	Data Type	Constraints	Description
admin_id	Varchar(10)	Primary Key	Admin ID
admin_name	text	Not Null	Admin Name
admin_email	varchar(30)	varchar(30)	Admin Email
admin_password	varchar(100)	Foreign Key	Admin Password

Table 5.5:EVENTS Table

Attribute	Data Type	Constraints	Description
approval_id	int	Primary Key	Approval ID
user_id	int	Foreign Key	User ID
aproval_status	varchar(30)	Not Null	Approval status
aproval_date	date	Not Null	Approval date

5.4.1 ER DIAGRAM

Here, I've just included the key components of my project and have illustrated how they relate to one another.


Email
Name
Participant
Participant has user
User
User can choose event
Choose Event
Fullname
Affiliation
Phone Number
Email
Id
Event's Theme
Datetime
Participant has admin
Admin
Admin has input event
Email
Name
Input Event
Event's Theme
Datetime

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

Implementation means turning a plan into reality. The implementation plan provides a quick summary, explains the steps, and lists the tasks involved. When we test many different cases during implementation, we're making sure the system is ready to use.
Coding standards are important for a system because they help people who aren't programmers understand how the system works. These standards also make it easier for others to make changes and updates to the system, even if they weren't the original developers. Think of it like using a common set of rules for naming things, similar to how window classes are named.

6.2 PSEUDO CODES

6.2.1 PSEUDO CODE FOR REGISTRATION PAGE

The registration page for the web application is designed to securely collect user information such as username, password, email, and other necessary details. Upon submission, the page performs several validation checks to ensure that all required fields are filled, the email format is valid, and the password fields match. It also checks if the username or email already exists in the database to prevent duplicate accounts. If all validations pass, the password is encrypted for security, and the user information is saved to the database. A success message is displayed, and the user is redirected to the login page. If any validation fails, an appropriate error message is shown to guide the user in correcting the input. This ensures a secure and smooth registration process for new users.

```
BEGIN RegistrationPage

    DISPLAY Registration Form
        - Input: Username
        - Input: Password
        - Input: Confirm Password
        - Input: Email
    - Input: Other Details (optional)

ON User Clicks "Submit" Button
    GET input values: username, password, confirm_password, email, other_details

    VALIDATE input values
        IF username IS empty
            DISPLAY error message "Username is required"
            RETURN
        ENDIF

        IF email IS empty OR email IS NOT valid format
            DISPLAY error message "Valid email is required"
            RETURN
        ENDIF

        IF password IS empty OR confirm_password IS empty
            DISPLAY error message "Password is required"
            RETURN
        ENDIF

        IF password != confirm_password
            DISPLAY error message "Passwords do not match"
            RETURN
        ENDIF

        CHECK IF username OR email ALREADY EXISTS in the database
            IF exists
                DISPLAY error message "Username or Email already exists"
                RETURN
            ENDIF

    IF all validations pass
        ENCRYPT password
        CREATE new user object with username, encrypted password, email, other_details
        SAVE user object to the database
        DISPLAY success message "Registration successful"
        REDIRECT to login page
    ELSE
        DISPLAY error message "Registration failed. Please try again."
    ENDIF
    END ON
END RegistrationPage
```

6.2.2 PSEUDO CODE FOR LOGIN PAGE

To access the system, the user needs to log in using their email address and password. First, we check if the email is correct. If it is, we then make sure that the password matches. If the password is correct, the login is successful, and they can access the system. If not, an error message pops up.

```
BEGIN LoginPage

    DISPLAY Login Form
        - Input: Username
        - Input: Password

    ON User Clicks "Login" Button
        GET input values: username, password

        VALIDATE input values
            IF username IS empty
                DISPLAY error message "Username is required"
                RETURN
            ENDIF

            IF password IS empty
                DISPLAY error message "Password is required"
                RETURN
            ENDIF

        IF validations pass
            FETCH user object FROM database WHERE username = input_username

            IF user object EXISTS
                ENCRYPT input_password
                COMPARE encrypted input_password WITH stored encrypted password

                IF passwords MATCH
                    SET session user_id = user object id
                    DISPLAY success message "Login successful"
                    REDIRECT to user dashboard
                ELSE
                    DISPLAY error message "Invalid username or password"
                ENDIF
            ELSE
                DISPLAY error message "Invalid username or password"
            ENDIF
        ELSE
            DISPLAY error message "Login failed. Please try again."
        ENDIF

    END ON

END LoginPage
```

6.2.3 PSEUDO CODE FOR FORGOT PASSWORD

A link to reset the password will be sent to the user's email address if they enter their email address in the case of password forgetfulness. A message will be presented in error if the email address is incorrect.

```
BEGIN ForgotPasswordPage

    DISPLAY Forgot Password Form
        - Input: Email Address

    ON User Clicks "Submit" Button
        GET input value: email

        VALIDATE input value
            IF email IS empty OR email IS NOT valid format
                DISPLAY error message "Valid email is required"
                RETURN
            ENDIF

        IF validation passes
            FETCH user object FROM database WHERE email = input_email

            IF user object EXISTS
                GENERATE unique password reset token
                STORE token IN database associated with user object
                CREATE password reset link WITH token
                SEND password reset link TO user email
                DISPLAY success message "Password reset link has been sent to your
email"
            ELSE
                DISPLAY error message "Email is not registered"
            ENDIF
        ELSE
        DISPLAY error message "Submission failed. Please try again."
    ENDIF

  END ON

END ForgotPasswordPage
```

6.2.4 PSEUDO CODE FOR USER APPROVAL

```
BEGIN UserApproval

    DISPLAY Pending User Approvals List
        FOR each pending user
            DISPLAY user details (username, email, registration date, etc.)
            DISPLAY "Approve" and "Reject" buttons

    ON Admin Clicks "Approve" Button for a User
        GET user_id FROM selected user
        VALIDATE admin authorization
            IF admin IS NOT authorized
                DISPLAY error message "You are not authorized to approve users"
                RETURN
            ENDIF

        IF admin IS authorized
            FETCH user object FROM database WHERE user_id = input_user_id
            UPDATE user object status TO "Approved"
            SAVE user object TO database
            DISPLAY success message "User approved successfully"
        ELSE
            DISPLAY error message "Approval failed. Please try again."
        ENDIF

    END ON

    ON Admin Clicks "Reject" Button for a User
        GET user_id FROM selected user
        VALIDATE admin authorization
            IF admin IS NOT authorized
                DISPLAY error message "You are not authorized to reject users"
                RETURN
            ENDIF

        IF admin IS authorized
            FETCH user object FROM database WHERE user_id = input_user_id
            UPDATE user object status TO "Rejected"
            SAVE user object TO database
            DISPLAY success message "User rejected successfully"
        ELSE
        DISPLAY error message "Rejection failed. Please try again."
    ENDIF

  END ON

END UserApproval
```

6.2.5 PSEUDO CODE FOR ADDING EVENT

```
BEGIN AddEvent

  DISPLAY Add Event Form
    - Input: Event Name
    - Input: Event Date
    - Input: Event Time
    - Input: Event Location
    - Input: Event Description
    - Input: Upload Event Image/Video (optional)

  ON Admin Clicks "Submit" Button
    GET input values: event_name, event_date, event_time, event_location,
event_description, event_media

    VALIDATE input values
      IF event_name IS empty
        DISPLAY error message "Event name is required"
        RETURN
      ENDIF

      IF event_date IS empty OR event_date IS NOT valid date format
        DISPLAY error message "Valid event date is required"
        RETURN
      ENDIF

      IF event_time IS empty OR event_time IS NOT valid time format
        DISPLAY error message "Valid event time is required"
        RETURN
      ENDIF

      IF event_location IS empty
        DISPLAY error message "Event location is required"
        RETURN
      ENDIF

      IF event_description IS empty
        DISPLAY error message "Event description is required"
        RETURN
      ENDIF

    IF all validations pass
```

```
        CREATE new event object WITH input values
        IF event_media IS NOT empty
            ATTACH event_media TO event object
        ENDIF
        SAVE event object TO database
        DISPLAY success message "Event added successfully"
        REDIRECT to events list page
    ELSE
        DISPLAY error message "Adding event failed. Please try again."
    ENDIF

  END ON

END AddEvent
```

CHAPTER 7

SOFTWARE TESTING

7.1 INTRODUCTION

Testing is like checking a computer program to make sure it works correctly. It's super important because it determines if the system will work well or not. The main goal of testing is to find any mistakes or problems in the software so we can fix them. We do lots of checks and tests on this project to make sure it meets what the user wants. This is especially useful for people who are new to the software, so they know what to expect. We've already done two types of testing: one where we look at individual parts, and another where we see how the whole thing works together. During testing, we set up different situations, create data to use, and make a plan for what we expect to happen. This helps us figure out if everything is working the way it should.

7.2 Testing Objective

- To ensure that the software is created to fulfill the needs of the user, testing is done.
- Testing enhances the program's quality and guarantees that the final product is error-free. to make certain that all validation and verification are finished.
- Make sure that the specifications are followed in the implementation of each module.

In our software testing process for this project, we include Unit Testing, Integration Testing, Functional Testing, and Regression Testing. We'll break down the project into modules and test each one individually.

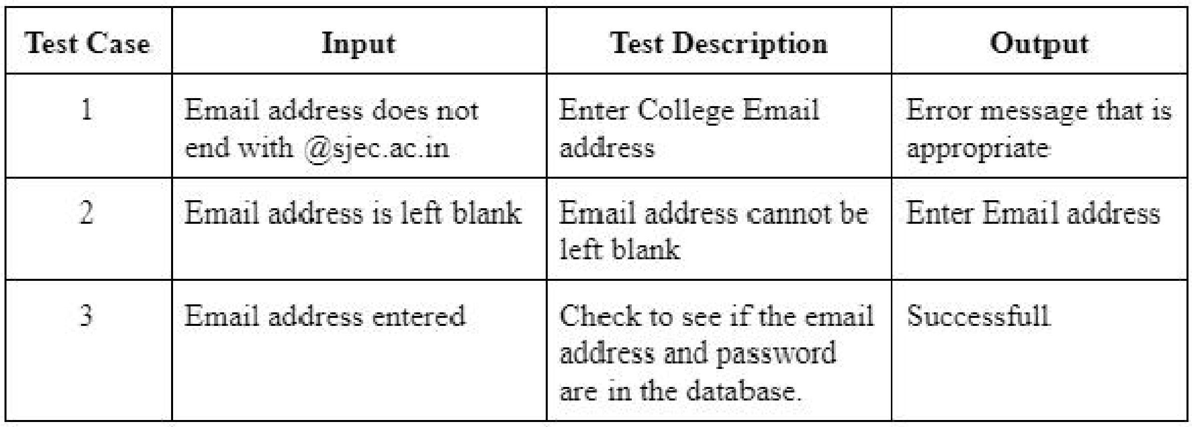
7.3 UNIT TESTING

A test that confirms the functionality of a particular area of code, often at the functional level, is known as a unit test, also referred to as a component test. When writing code, developers or programmers frequently run these tests to ensure that a certain function is operating as intended. To cut risks, costs, and time associated with software development, approaches for defect prevention and detection are used. Unit testing is used to make sure that each piece of software operates as intended. The smallest piece of software that can be tested is a unit. It normally has a single output and one or more inputs.

7.3.1 UNIT TEST CASES

Unit testing entails testing an application's smallest possible unit. Detecting and repairing problems at this point is straightforward, takes less time, and in less cost. Each module in the SJEC Approval Management System is tested separately with the proper inputs and yields the desired outcomes.

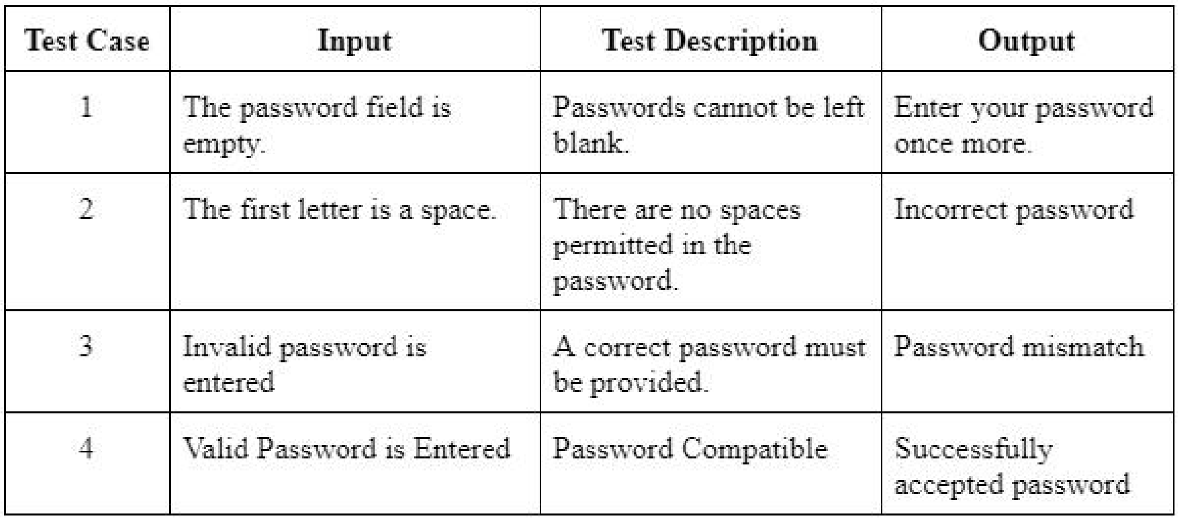
7.3.1.1 Testing for Valid Email Address



Test Case	Input	Test Description	Output
1	Email address does not end with @sjec.ac.in	Enter College Email address	Error message that is appropriate
2	Email address is left blank	Email address cannot be left blank	Enter Email address
3	Email address entered	Check to see if the email address and password are in the database.	Successfull

Table 7.1 Testing for valid Email Address

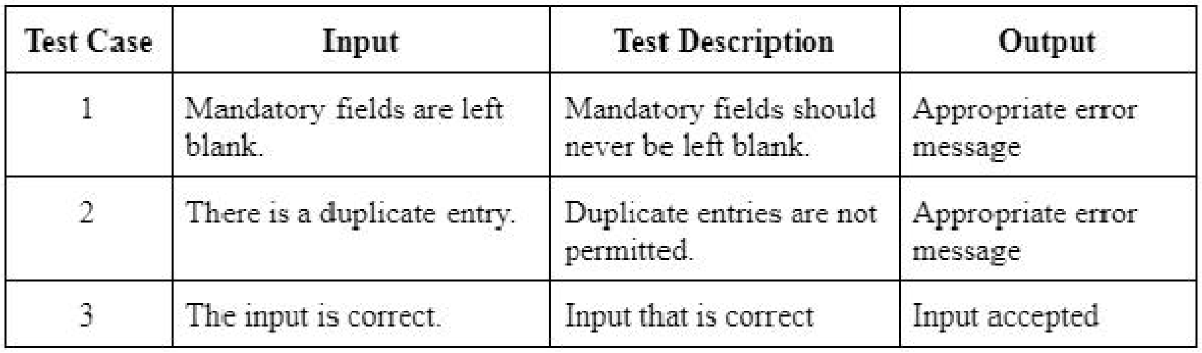
7.3.1.2 Testing for Valid Password



Test Case	Input	Test Description	Output
1	The password field is empty.	Passwords cannot be left blank.	Enter your password once more.
2	The first letter is a space.	There are no spaces permitted in the password.	Incorrect password
3	Invalid password is entered	A correct password must be provided.	Password mismatch
4	Valid Password is Entered	Password Compatible	Successfully accepted password

Table 7.2 Testing for valid password

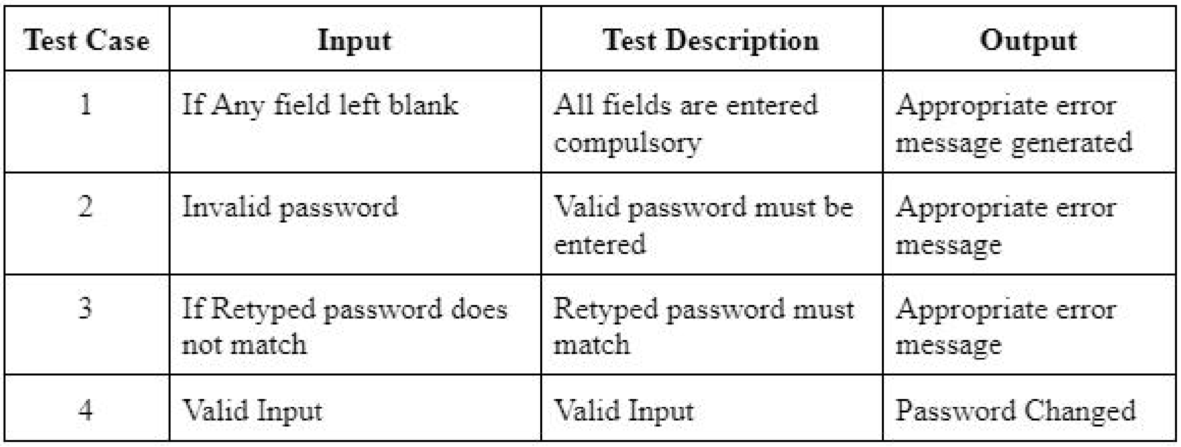
7.3.1.3 Testing for Data Insertion



Test Case	Input	Test Description	Output
1	Mandatory fields are left blank.	Mandatory fields should never be left blank.	Appropriate error message
2	There is a duplicate entry.	Duplicate entries are not permitted.	Appropriate error message
3	The input is correct.	Input that is correct	Input accepted

Table 7.3 Testing for Data Insertion

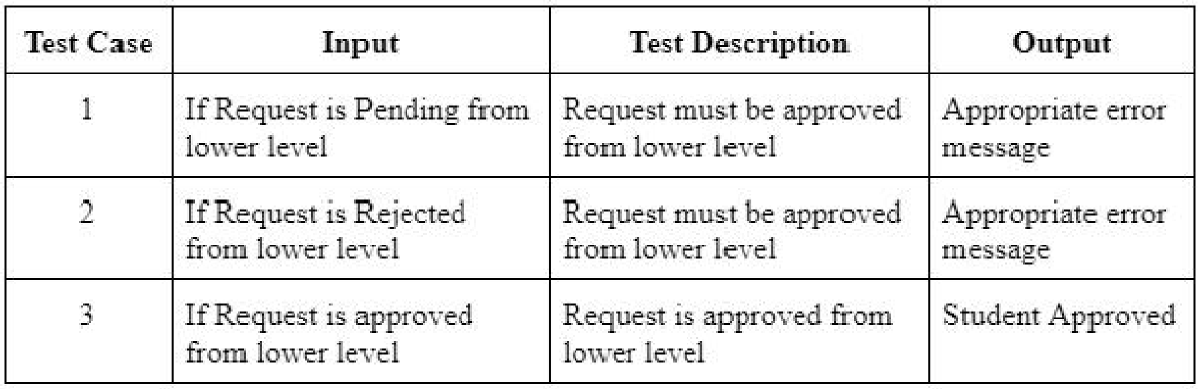
7.3.1.4 Testing for Change Password



Test Case	Input	Test Description	Output
1	If Any field left blank	All fields are entered compulsory	Appropriate error message generated
2	Invalid password	Valid password must be entered	Appropriate error message
3	If Retyped password does not match	Retyped password must match	Appropriate error message
4	Valid Input	Valid Input	Password Changed

Table 7.4 Testing for Change Password

7.3.1.5 Testing for Approve Requested Approval



Test Case	Input	Test Description	Output
1	If Request is Pending from lower level	Request must be approved from lower level	Appropriate error message
2	If Request is Rejected from lower level	Request must be approved from lower level	Appropriate error message
3	If Request is approved from lower level	Request is approved from lower level	Student Approved

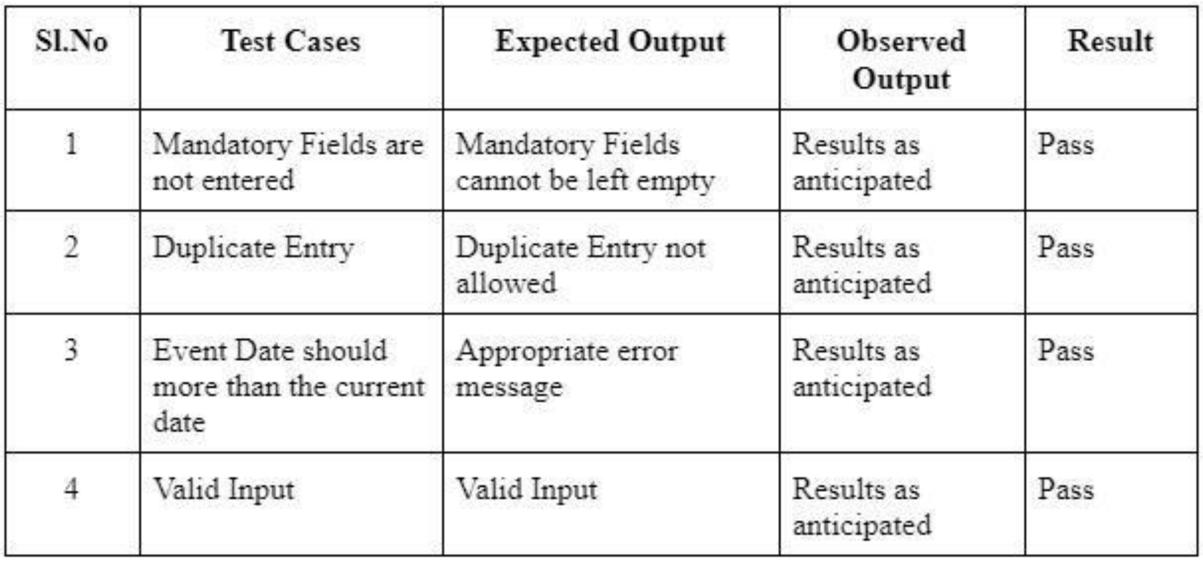
Table 7.4 Testing for Approve Requested Approval

7.4 INTEGRATION TESTING

Integration testing is a way to check how different parts of the software work together, just like how they were planned. It's really helpful in our project because it helps us find and fix problems where these parts connect. When we do integration testing, we look for issues in the connections and how different pieces of the software talk to each other. We start by testing smaller groups of connected parts and gradually bring them all together to make sure they work as a complete system.

7.4.1 INTEGRATION TEST CASES

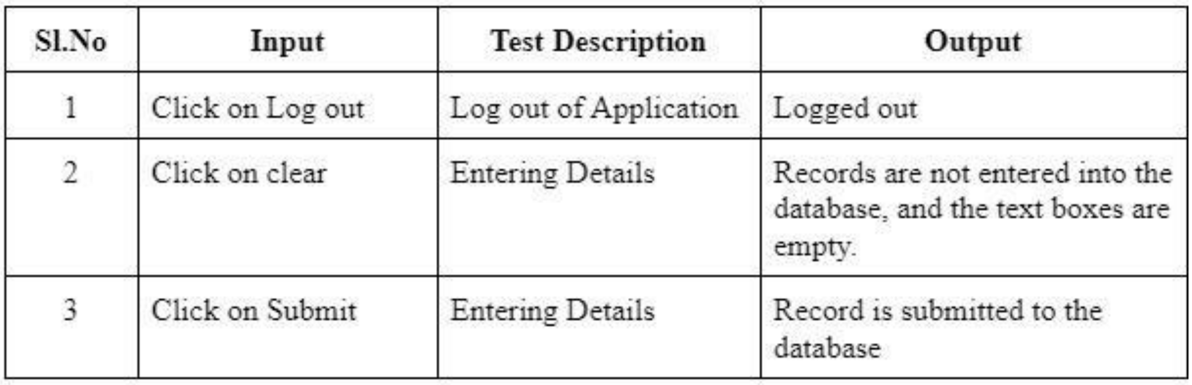
We're like detectives checking how different parts of the application work together. Sometimes, when we put two parts together, they can cause problems and not give us the results we want. That's why we make sure to test each part of our project when it's all put together as a whole.



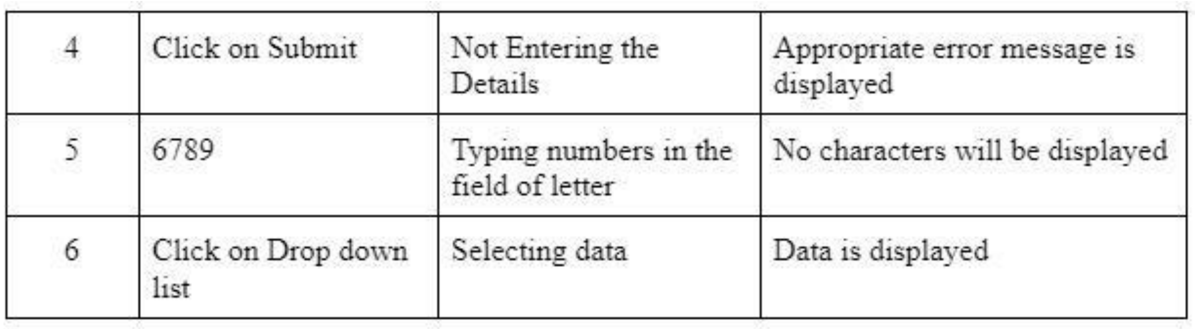
Sl.No	Test Cases	Expected Output	Observed Output	Result
1	Mandatory Fields are not entered	Mandatory Fields cannot be left empty	Results as anticipated	Pass
2	Duplicate Entry	Duplicate Entry not allowed	Results as anticipated	Pass
3	Event Date should more than the current date	Appropriate error message	Results as anticipated	Pass
4	Valid Input	Valid Input	Results as anticipated	Pass

Table 7.5 Integration Test Cases for Data Insertion

7.4.2 OTHER TEST CASES



Sl.No	Input	Test Description	Output
1	Click on Log out	Log out of Application	Logged out
2	Click on clear	Entering Details	Records are not entered into the database, and the text boxes are empty.
3	Click on Submit	Entering Details	Record is submitted to the database



4	Click on Submit	Not Entering the Details	Appropriate error message is displayed
5	6789	Typing numbers in the field of letter	No characters will be displayed
6	Click on Drop down list	Selecting data	Data is displayed

CHAPTER 8

CONCLUSION

The web application for flat residents offers an efficient and user-friendly platform for managing community events and fostering resident engagement. It streamlines the process of event creation and participation, allowing administrators to easily organize events and send notifications while enabling residents to view events and provide feedback. With its secure authentication and authorization mechanisms, the application ensures that user data is protected and access is appropriately controlled.

The real-time update capabilities, facilitated by Web Sockets, enhance the user experience by providing instant notifications and updates on event statuses. The integration of a scalable architecture, incorporating load balancing and caching, ensures that the application can handle high user loads and maintain performance during peak times.

Furthermore, the system's intuitive design and accessibility features make it easy for residents of all technical skill levels to use. The use of Django for the backend provides a robust framework for managing data and executing business logic, while the responsive frontend ensures a consistent experience across different devices.

In summary, this web application not only improves the organization and management of community events but also enhances communication and collaboration among residents. By providing a centralized platform for event management, notifications, and feedback, the application contributes to building a more connected and engaged residential community.

CHAPTER 9

FUTURE ENHANCEMENTS

Future enhancements for the web application could include additional features and improvements to further enhance user experience and functionality. Integrating a chat system would allow real-time communication among residents and administrators, fostering a stronger community bond. Implementing AI-driven analytics could provide valuable insights into event participation trends and user preferences, enabling more personalized event recommendations. Expanding the notification system to include SMS and mobile push notifications would ensure that residents receive important updates promptly, regardless of their preferred communication method.

Enhancing the application’s mobile responsiveness and developing dedicated mobile apps for iOS and Android would provide a more seamless experience for users accessing the platform on their smartphones. Integrating payment gateways could facilitate the management of event fees and other community-related payments directly through the application. Adding a comprehensive feedback and suggestion system would enable residents to share their ideas and opinions, helping administrators to continuously improve the community experience.

Finally, incorporating smart home integration features could connect the application with residents’ home automation systems, providing additional convenience and enhancing the overall living experience. By continually evolving and adding new capabilities, the application can remain a valuable tool for managing and enhancing community life in residential complexes.

APPENDIX A

REFERENCES

[1] William S. Vincent, Django for Professionals: Production websites with Python & Django, 1st Edition, WelcomeToCodePublication Narosa Publishing House.

[2] Jacob Lett), Bootstrap 4 Quick Start: Responsive Web Design, Publication: Bootstrap Creative

[3] www.Google.com

[4] www.wikipedia.com

[5] www.codeproject.com

APPENDIX B

USER MANUAL

1. LOGIN PAGE FOR ADMIN

Admin can login using Email address and Password.

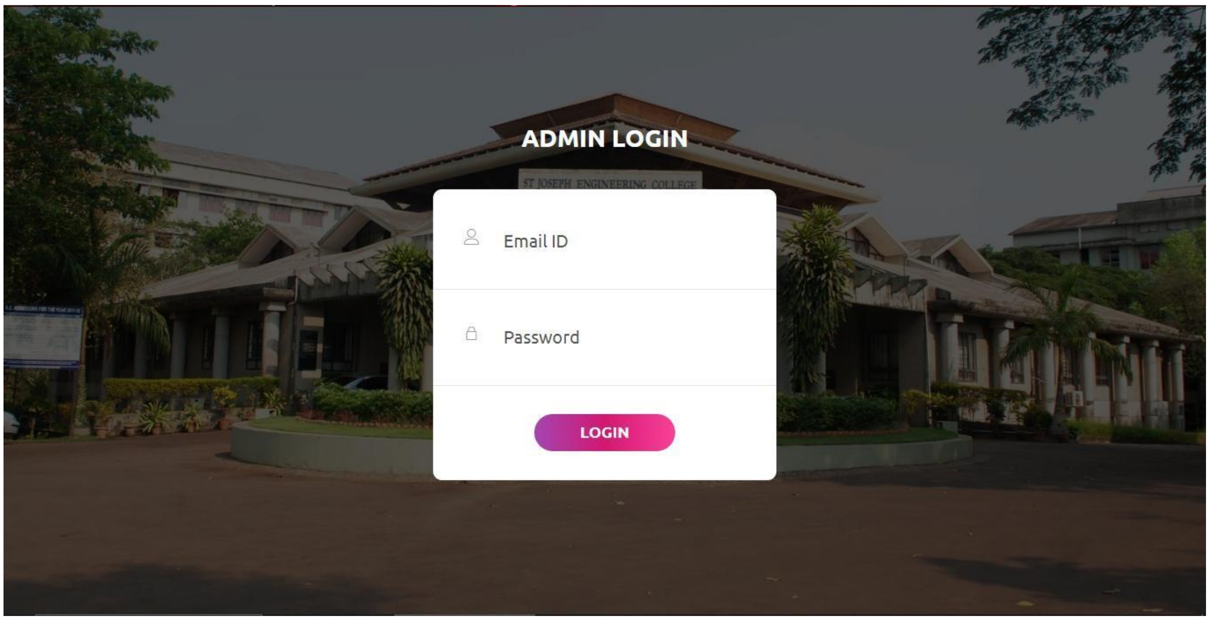
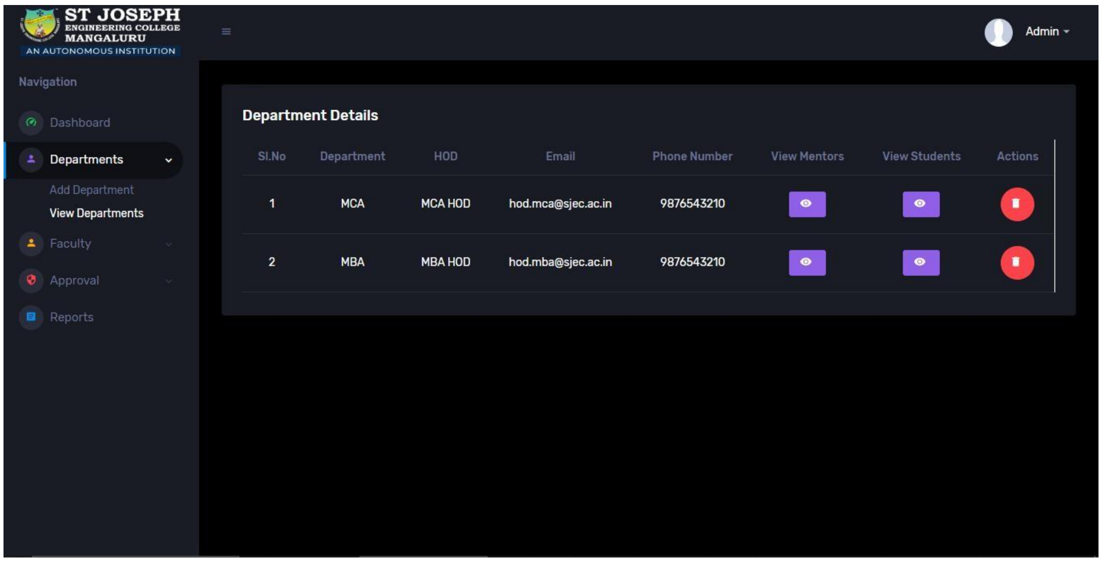


Fig 1:Admin Login Page

2. MANAGE DEPARTMENTS AND HOD

Here admin can view and manage the various departments and HOD’s.



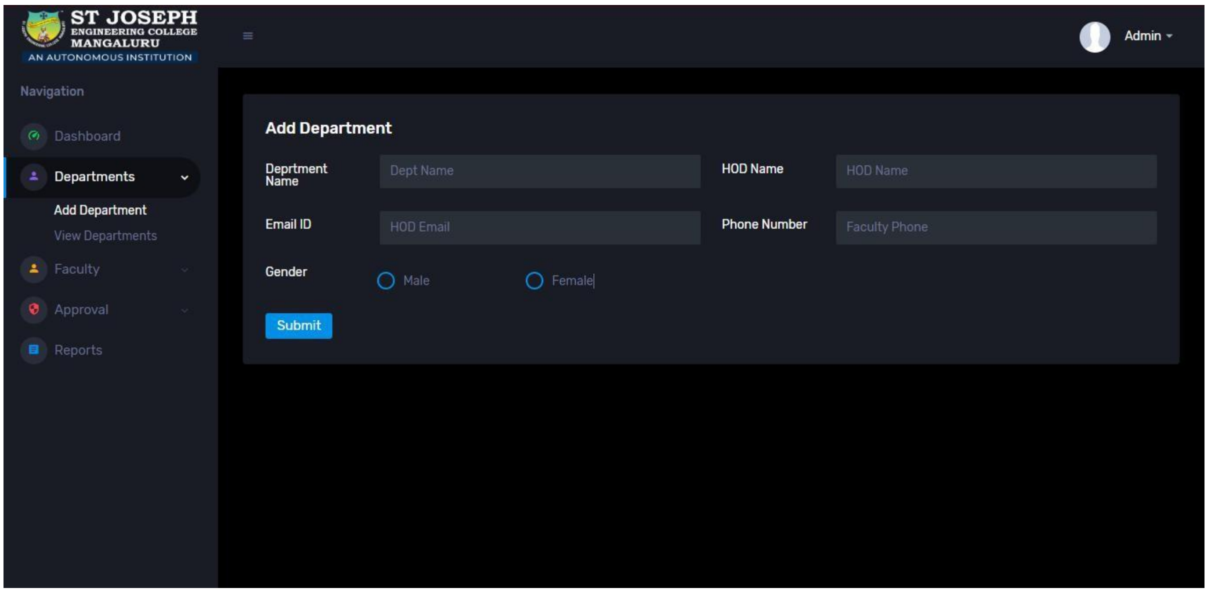
Department Details

Sl.No	Department	HOD	Email	Phone Number	View Mentors	View Students	Actions
1	MCA	MCA HOD	hod.mca@sjec.ac.in	9876543210			
2	MBA	MBA HOD	hod.mba@sjec.ac.in	9876543210			

Fig 2: Manage Departments and HOD

3. ADD DEPARTMENT AND HOD

Here the Admin can add department and HOD's by filling details such as Department Name, HOD Name and Email ID.



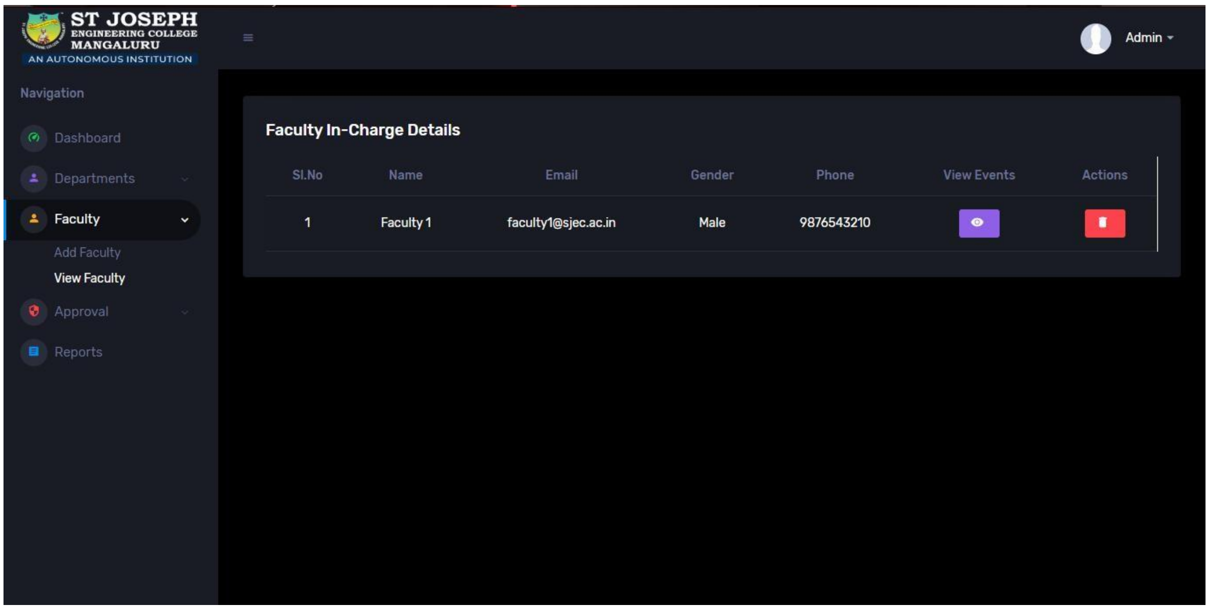
Add Department

Deprtment Name	Dept Name	HOD Name	HOD Name
Email ID	HOD Email	Phone Number	Faculty Phone
Gender	Male	Female	

Fig 3: Add Department and HOD

4. MANAGE FACULTY IN-CHARGE

The admin can manage the faculty in-charge and can also view the details of the faculties.



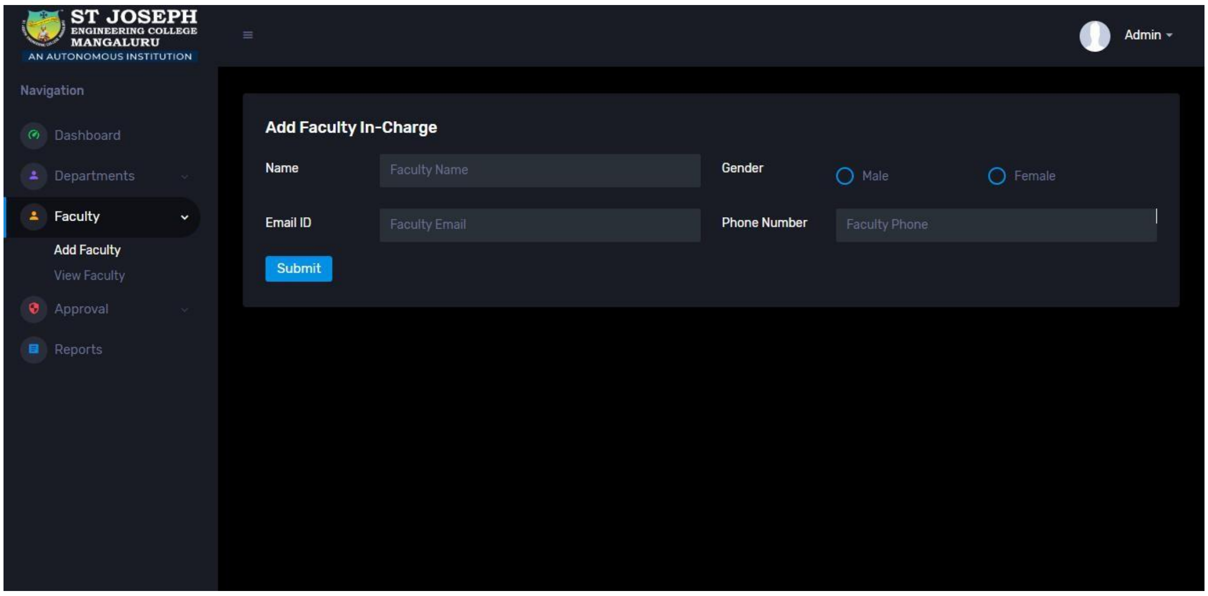
Faculty In-Charge Details

Sl.No	Name	Email	Gender	Phone	View Events	Actions
1	Faculty 1	faculty1@sjec.ac.in	Male	9876543210		

Fig 4: Manage Faculty In-charge

5. ADD FACULTY IN-CHARGE

Admin can add Faculty in-charge by updating their details such as name, email and phone number



Add Faculty In-Charge

Name	Faculty Name	Gender	Male Female
Email ID	Faculty Email	Phone Number	Faculty Phone

Fig 5: Add Faculty In-charge

6. STUDENT LOGIN PAGE

Student can login using their email address and password

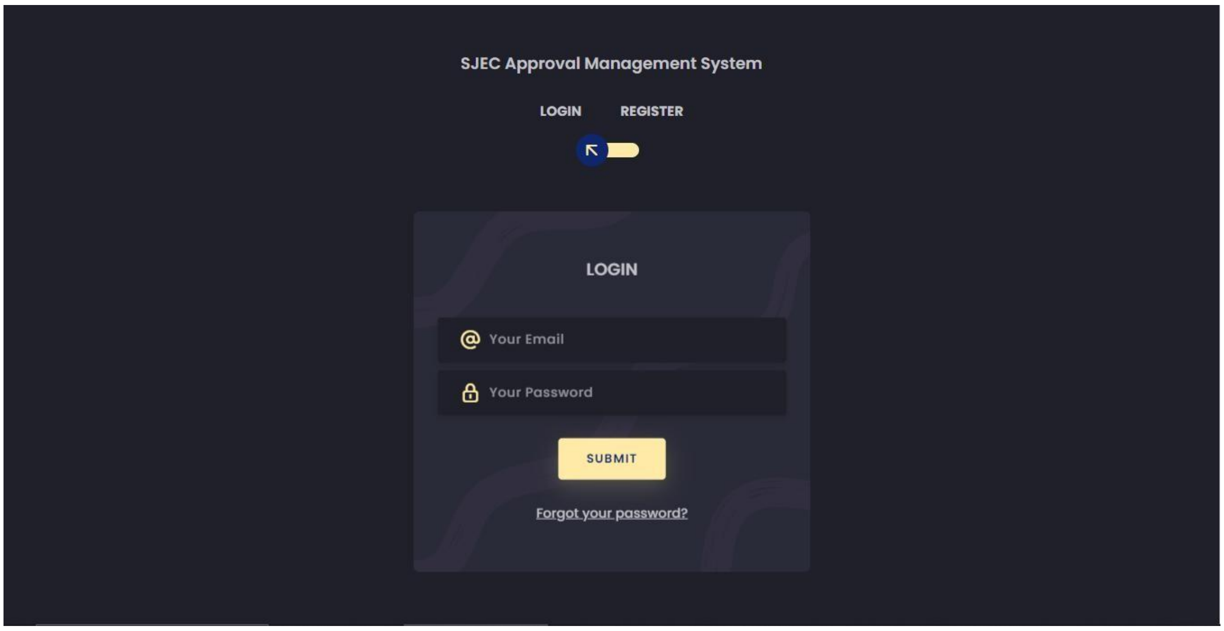


Fig 6: Student Login Page

7. STUDENT REGISTRATTION PAGE

Students of SJEC can register to the website using their USN, Email ID and password.

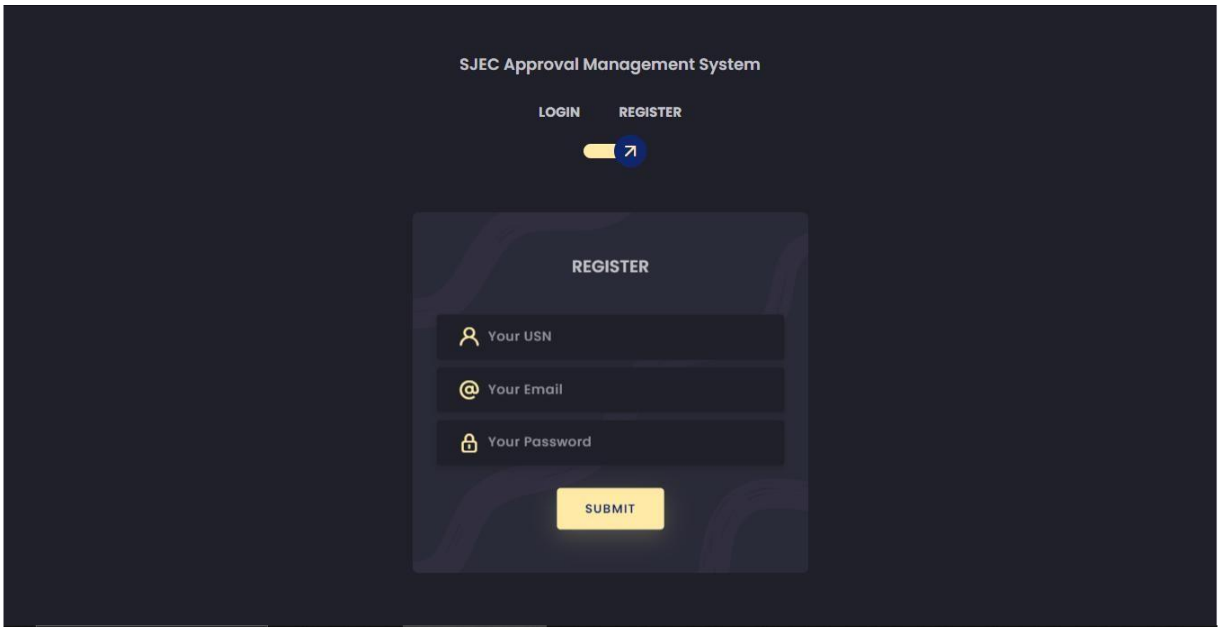
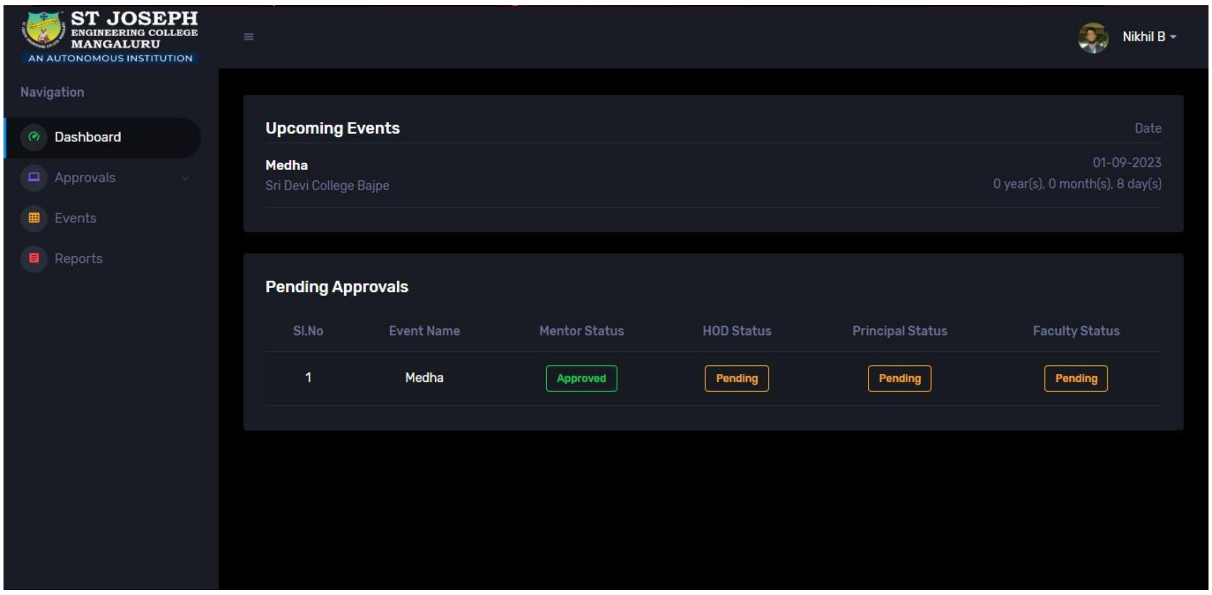


Fig 7: Student Registration Page

8. STUDENT HOME PAGE

Student Home Page contains details for the students such as events and approvals.



Upcoming Events

	Date
Medha Sri Devi College Bajpe	01-09-2023 0 year(s), 0 month(s), 8 day(s)

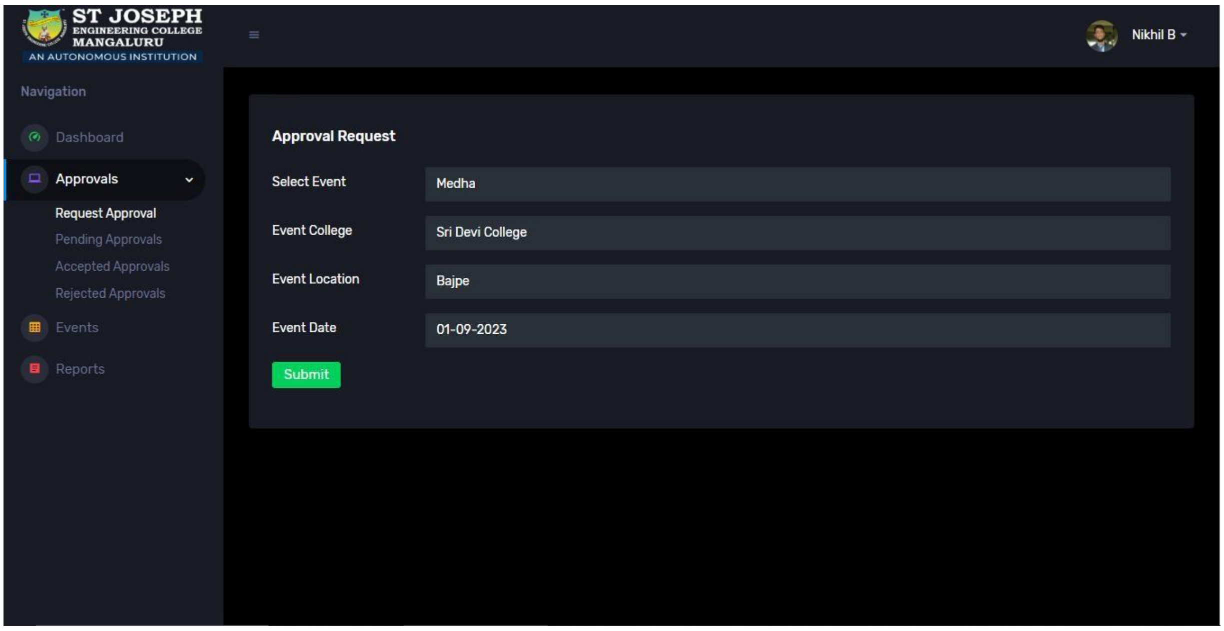
Pending Approvals

Sl.No	Event Name	Mentor Status	HOD Status	Principal Status	Faculty Status
1	Medha	Approved	Pending	Pending	Pending

Fig 8: Student Home Page

9. REQUEST APPROVAL PAGE

Student will have to select from the list of events available to view the event that he or she will participate.



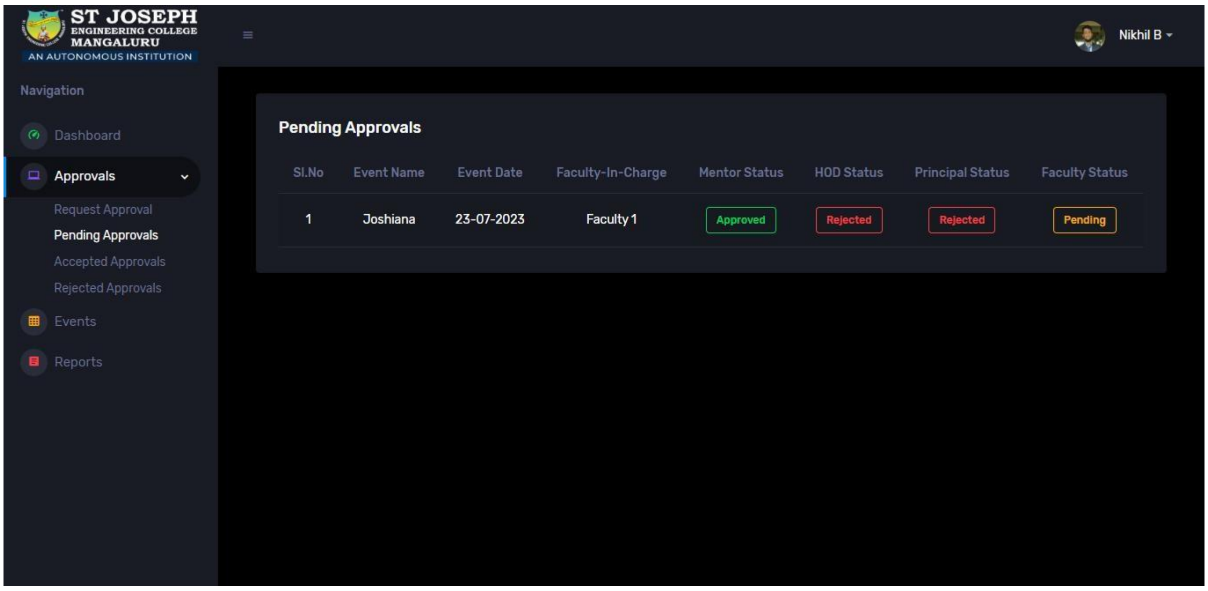
Approval Request

Select Event	Medha
Event College	Sri Devi College
Event Location	Bajpe
Event Date	01-09-2023

Fig 9: Request Approval Page

10. PENDING APPROVAL PAGE

Students can view the approvals that are pending.



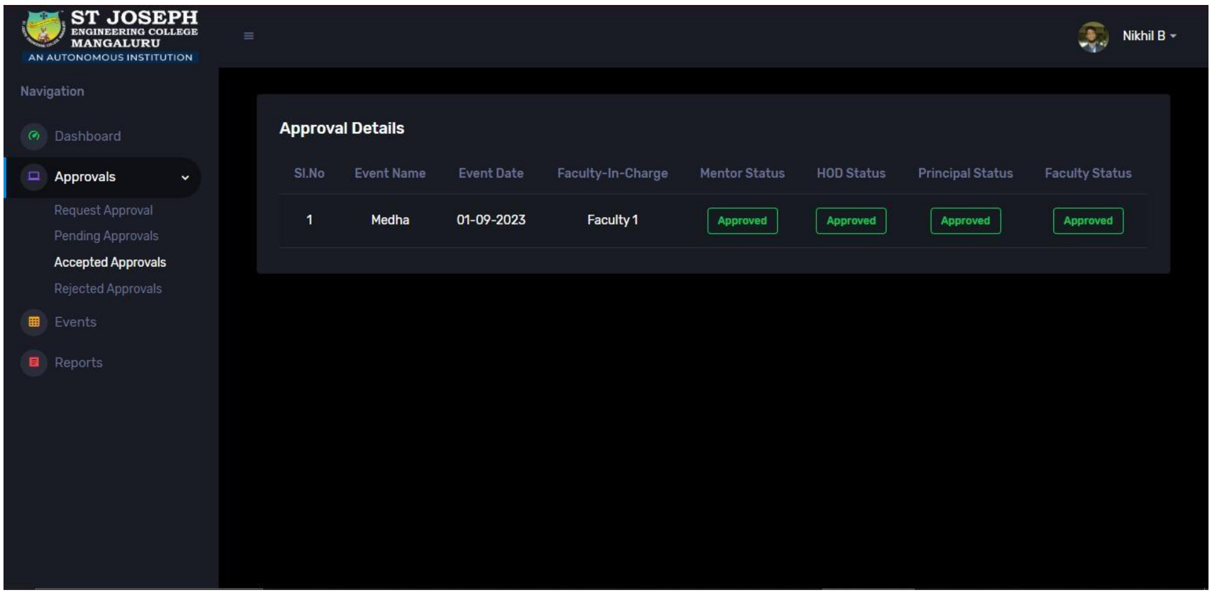
Pending Approvals

Sl.No	Event Name	Event Date	Faculty-In-Charge	Mentor Status	HOD Status	Principal Status	Faculty Status
1	Joshiana	23-07-2023	Faculty 1	Approved	Rejected	Rejected	Pending

Fig 10: Pending Approval Page

11. ACCEPTED APPROVAL PAGE

Students can view the approvals that have been accepted.



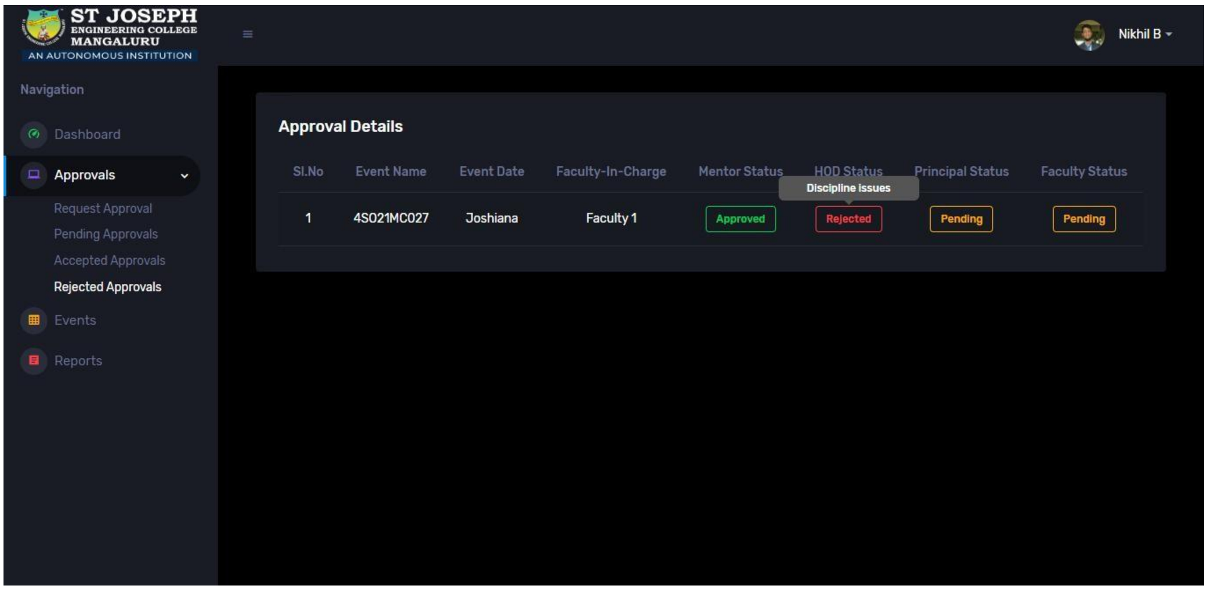
Approval Details

Sl.No	Event Name	Event Date	Faculty-In-Charge	Mentor Status	HOD Status	Principal Status	Faculty Status
1	Medha	01-09-2023	Faculty 1	Approved	Approved	Approved	Approved

Fig 11: Accepted Approval Page

12. REJECTED APPROVAL PAGE

Students can view the approvals that have been rejected along with the reason.



Approval Details

Sl.No	Event Name	Event Date	Faculty-In-Charge	Mentor Status	HOD Status	Principal Status	Faculty Status
1	4SO21MC027	Joshiana	Faculty 1	Approved	Rejected	Pending	Pending

Fig 12: Rejected Approval Page

13. MENTOR LOGIN

Mentors can login using their email address and password.

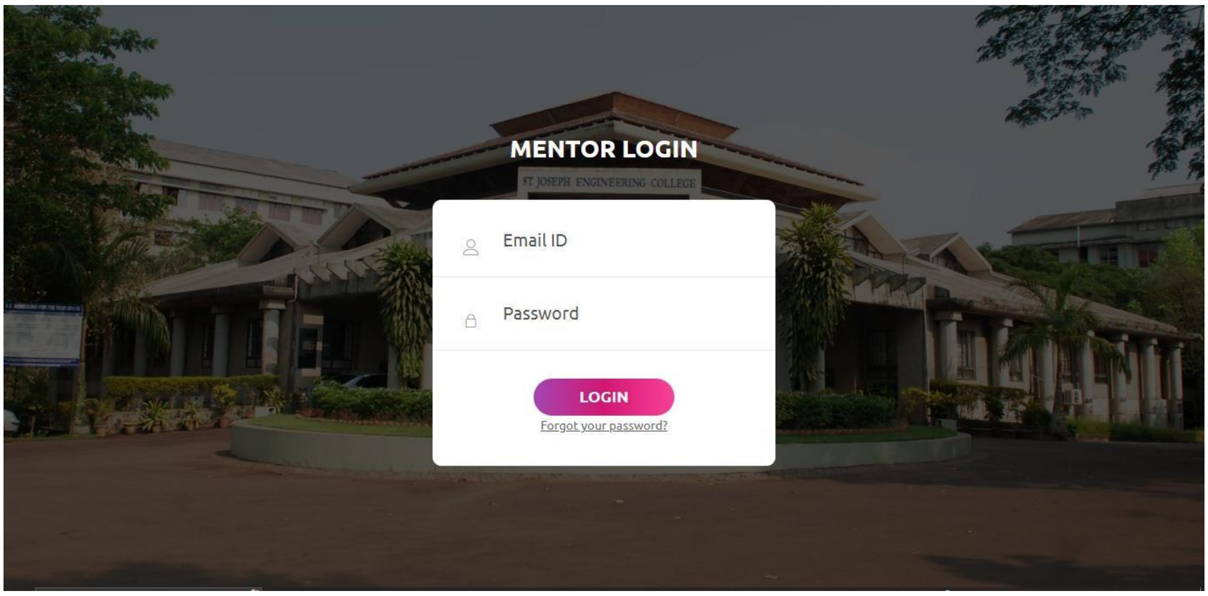
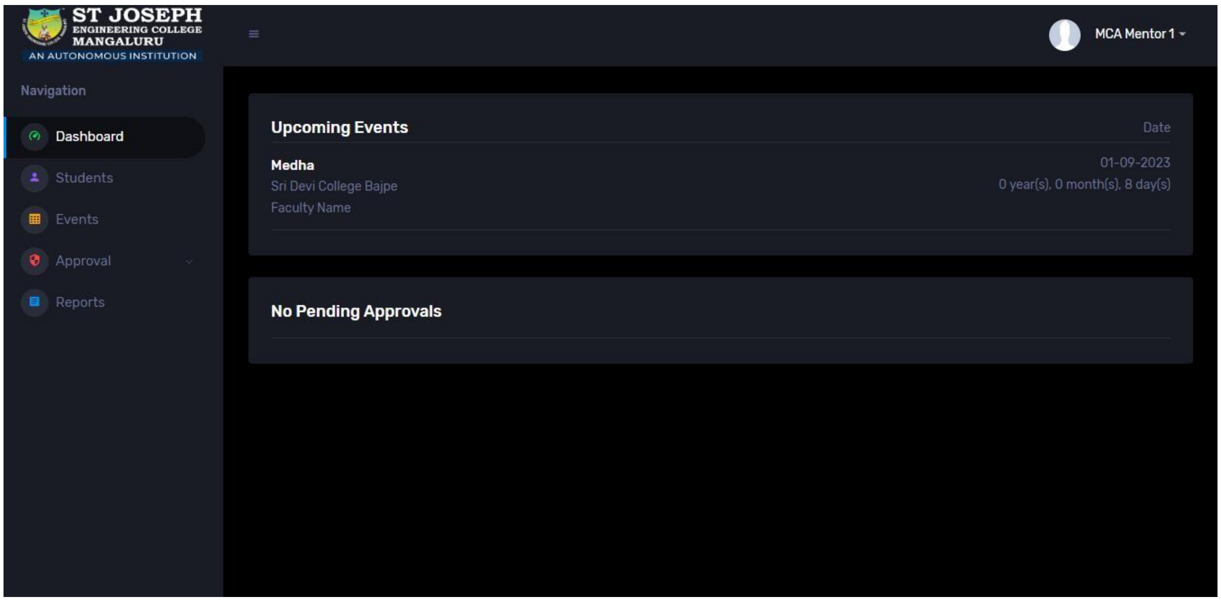


Fig 13: Mentor Login

14. MENTOR HOME PAGE

Mentor home page contains details such as upcoming events and pending approvals.



Upcoming Events Date

Medha 01-09-2023

Sri Devi College Bajpe 0 year(s), 0 month(s), 8 day(s)

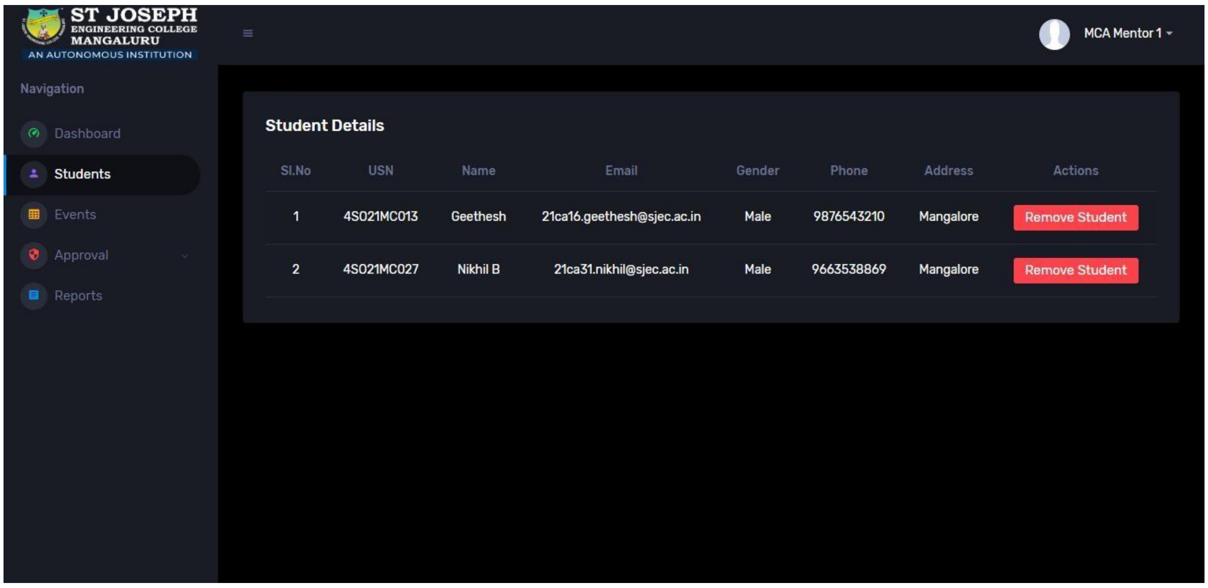
Faculty Name

No Pending Approvals

Fig 14: Mentor Home Page

15. MANAGE STUDENTS

Mentors can view the students under them.



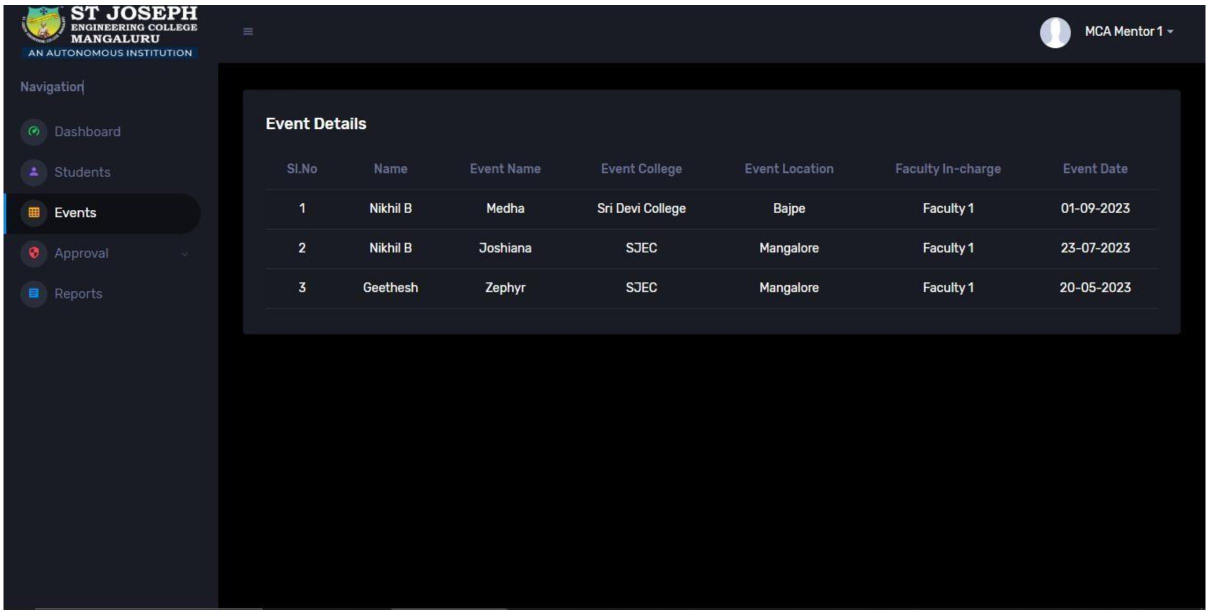
Student Details

Sl.No	USN	Name	Email	Gender	Phone	Address	Actions
1	4SO21MC013	Geethesh	21ca16.geethesh@sjec.ac.in	Male	9876543210	Mangalore	Remove Student
2	4SO21MC027	Nikhil B	21ca31.nikhil@sjec.ac.in	Male	9663538869	Mangalore	Remove Student

Fig 15: Manage Students

16. VIEW EVENTS

Mentors can view the events that the students are participating in.



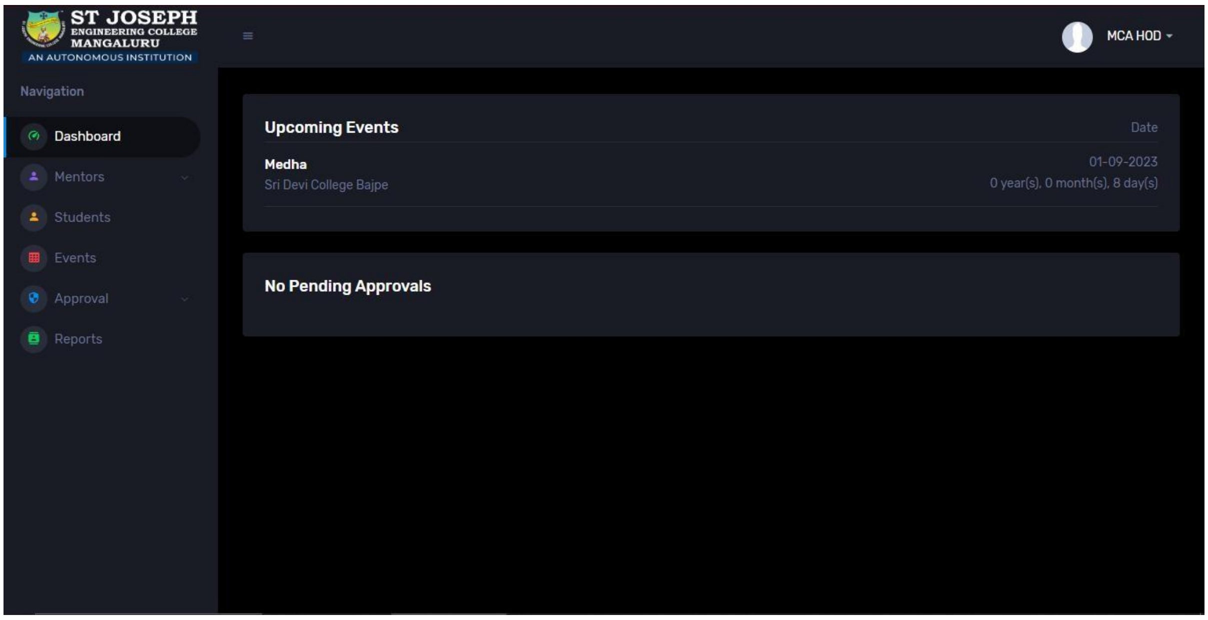
Event Details

Sl.No	Name	Event Name	Event College	Event Location	Faculty In-charge	Event Date
1	Nikhil B	Medha	Sri Devi College	Bajpe	Faculty 1	01-09-2023
2	Nikhil B	Joshiana	SJEC	Mangalore	Faculty 1	23-07-2023
3	Geethesh	Zephyr	SJEC	Mangalore	Faculty 1	20-05-2023

Fig 16: Manage Students

17.HOD HOME PAGE

HOD home page contains details such as upcoming events and pending approvals.



Upcoming Events Date

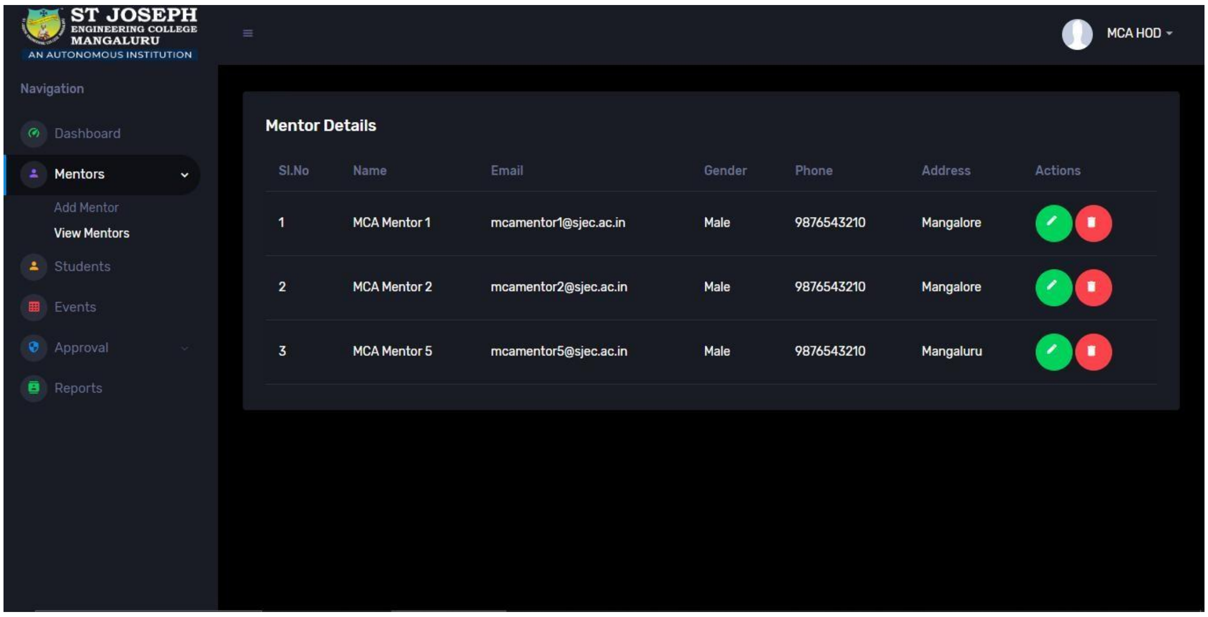
Medha
Sri Devi College Bajpe
01-09-2023
0 year(s), 0 month(s), 8 day(s)

No Pending Approvals

Fig 17: HOD Home Page

18. MANAGE MENTORS

HOD can manage the mentors that are present in their department.



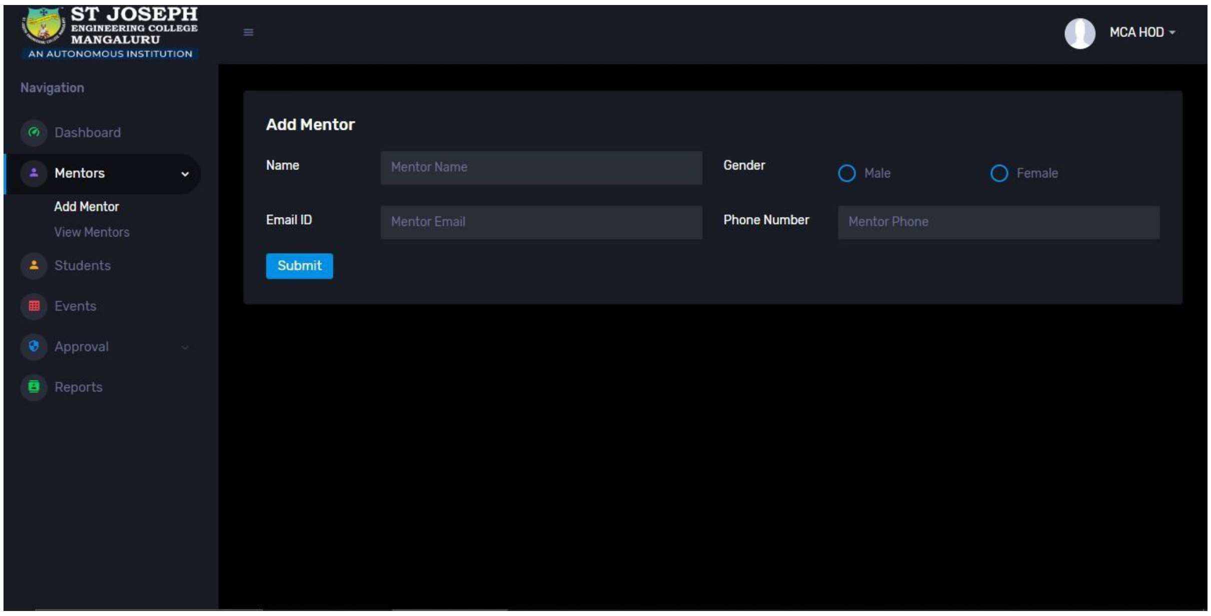
Mentor Details

Sl.No	Name	Email	Gender	Phone	Address	Actions
1	MCA Mentor 1	mcamentor1@sjec.ac.in	Male	9876543210	Mangalore	
2	MCA Mentor 2	mcamentor2@sjec.ac.in	Male	9876543210	Mangalore	
3	MCA Mentor 5	mcamentor5@sjec.ac.in	Male	9876543210	Mangaluru	

Fig 18: Manage Mentors

19. ADD MENTORS

HOD can add the mentors for their department.



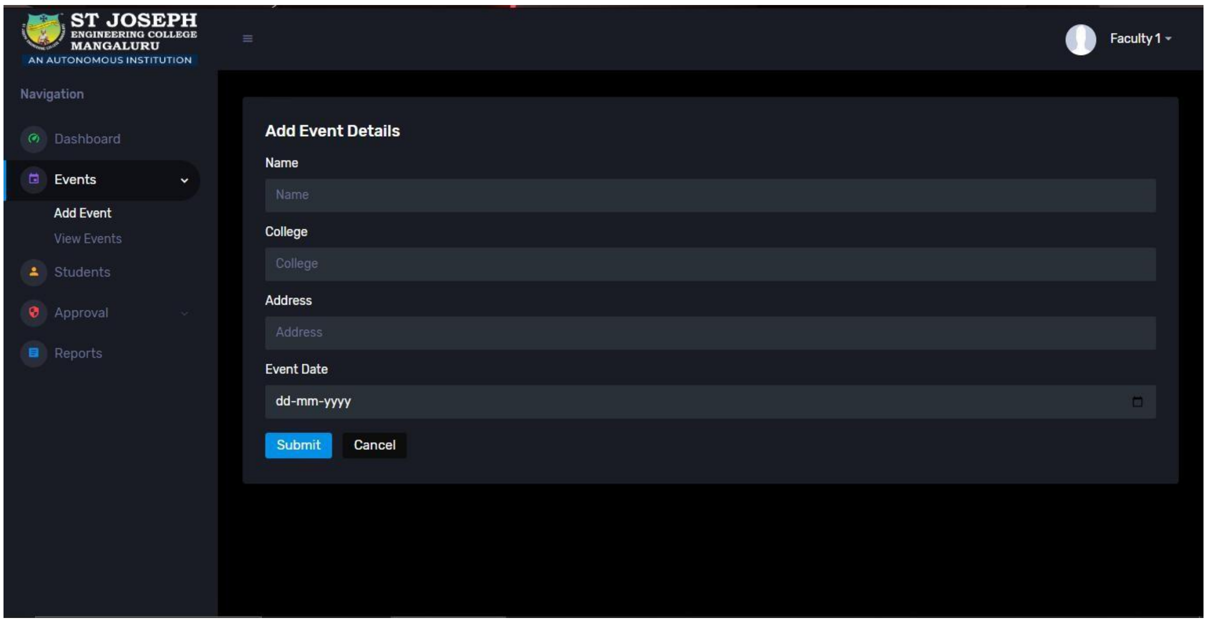
Add Mentor

Name	Mentor Name	Gender	Male Female
Email ID	Mentor Email	Phone Number	Mentor Phone

Fig 19: Add Mentors

20. ADD EVENT

Events can be added for the students to request the approvals by the faculty.



Add Event Details

Name: Name

College: College

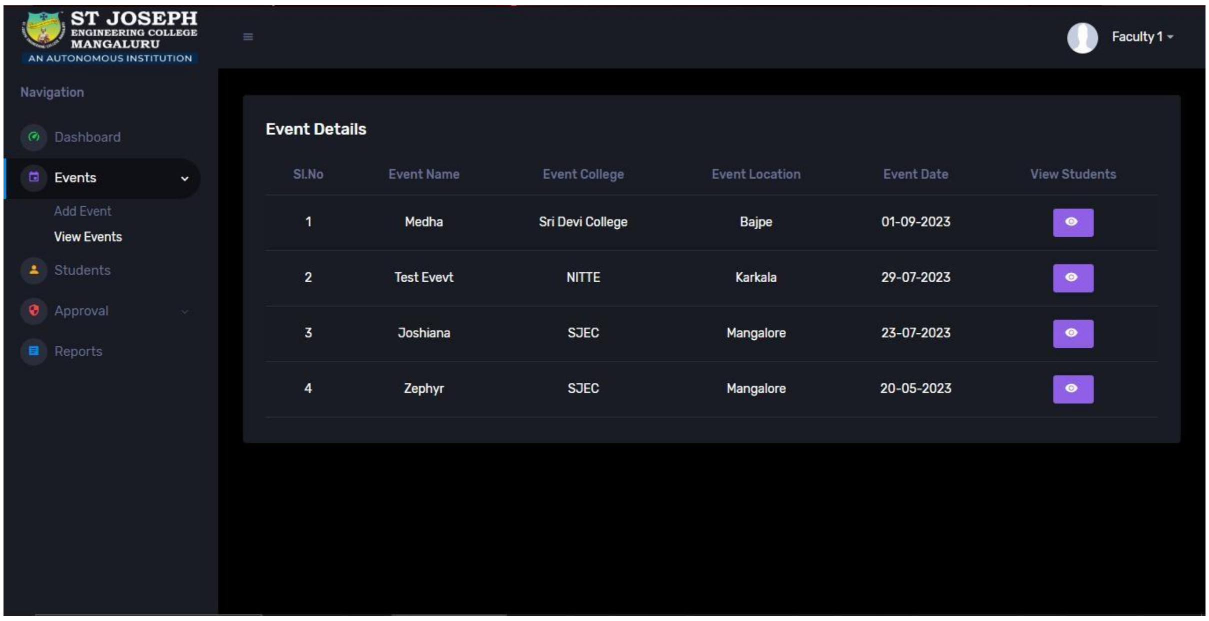
Address: Address

Event Date: dd-mm-yyyy

Fig 20: Add Event

21. MANAGE EVENTS

The faculty in-charge can manage the events that they have updated.



Event Details

Sl.No	Event Name	Event College	Event Location	Event Date	View Students
1	Medha	Sri Devi College	Bajpe	01-09-2023	
2	Test Evevt	NITTE	Karkala	29-07-2023	
3	Joshiana	SJEC	Mangalore	23-07-2023	
4	Zephyr	SJEC	Mangalore	20-05-2023	

Fig 21: Manage Events