A

PROJECT REPORT ON

COLLEGE EVENT MANAGEMENT SYSTEM

SUBMITTED BY

Ms. Vaishnavi Jaywant Kurunde

SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN FULFILLMENT OF DEGREE

MASTER OF COMPUTER APPLICATION(SEM-I)

UNDER THE GUIDANCE OF

Ms. Samiksha Yeola

Through,

Sadhu Vaswani Institute of Management Studies for Girls,

Koregaon Park, Pune - 411001

2024 - 2025

DECLARATION BY STUDENT

To
The Director,
SVIMS, Koregaon Park, Pune

Subject: Declaration of Original Work

I. undersigned hereby declare that this project titled **"COLLEGE EVENT MANAGEMENT SYSTEM,"** written and submitted by me to SPPU, Pune, in partial fulfilment of the requirement of the award of the degree of **MASTER OF COMPUTER APPLICATION (MCA-I)** under the guidance of **Ms. Samiksha Yeola** , is my original work.

I further declare that, to the best of my knowledge and belief, this project has not been submitted to any other university or institution for the award of any degree or qualification.

Place : Pune
Date:

(Vaishnavi Jaywant Kurunde)

ACKNOWLEGEMENT

I extend my sincere gratitude to Dr. B. H. Nanwani, Dr. Neeta Raskar, Prof.Samiksha Yeola for allowing me to carry out the study and for their constant encouragement, valuable suggestions, and guidance during the research work.

I extend my special thanks to Dr.Shweti Chanda, for her kind co- operation and inspiration. I extend my special gratitude to my dearest family members and friends who encouraged and motivated me to complete the project report.

Place : Pune
Date: **(Vaishnavi Jaywant Kurunde)**

Chapter 1 : Introduction

1.1 Client/Organization Profile

The College Event Management System (CEMS)** is a web-based platform that simplifies and automates event management at educational institutions. It provides a centralized solution for administrators, event organizers, and participants, resulting in a more efficient approach to event coordination.

Name : Sadhu Vaswani Institute of Management Studies for Girls
Industry : Educational and event management

Managing multiple events across different departments in colleges can be difficult. Academic conferences, workshops, seminars, cultural festivals, and club meetings are all time-consuming and labor-intensive events. Manual management frequently results in mistakes, miscommunication, and inefficiencies.

Purpose:
The College Event Management System seeks to automate and centralize event management through:
1. Making event registration easier: simple sign-ups and real-time attendees.
2. Improving communication: making sure all stakeholders are aware of schedules and updates.
3. Improving event tracking by organizing event information and registrations.
4. Interactive calendar with real-time event editing.

Key Activities:
CEMS supports a diverse range of events, including:
1. Seminars & Workshops: Simplifies scheduling, registration, and feedback collection.
2. Competitions: Facilitates participant registration and event tracking.
3. Cultural & Academic Events: Manages registrations, schedules, and coordination.
4. Club & Departmental Events: Helps track participation and communication.

The system's ultimate goal is to reduce administrative workload, increase efficiency, and foster greater student engagement by providing a transparent and user-friendly event management platform.

1.2 Need for System

Managing events in educational institutions presents a number of challenges, such as scheduling, registration management, effective communication, and record keeping. Traditionally, these tasks are performed manually, which can lead to errors, inefficiencies, and miscommunication among stakeholders. As colleges expand and host a wide range of events, such as academic conferences, cultural festivals, workshops, and competitions, the management of these activities becomes increasingly complex.

- The College Event Management System (CEMS) addresses these issues by providing a centralized and automated event management solution. The limitations of manual processes lead to the need for such a system.
- Time-Consuming Processes: Manually managing registrations, tracking participant data, and creating schedules can be labor-intensive and prone to delays.
- Communication Gaps: Ineffective communication between organizers and participants often leads to missed deadlines or poorly attended events.
- Lack of transparency: Manual systems lack a streamlined approach to sharing updates, event details, and notifications.
- Data Management Issues: Storing and retrieving participant and event data manually is cumbersome and can lead to inaccuracies.
- Scaling Challenges: As the number of events increases, manual processes become unsustainable.
- By automating these processes, CEMS streamlines event management, enhances communication, reduces errors, and saves time. Features like real-time notifications, event registration tracking, and centralized data storage make it easier for institutions to handle events of any scale. This system not only simplifies administrative tasks but also ensures that students and faculty have a seamless experience when participating in or organizing events, making it a vital tool for modern educational institutions.

1.3 Scope & Feasibility of Work

College Event Management System (CEMS) is designed to modernize event management processes within educational institutions by providing a user-friendly, centralized platform. The system encompasses features aimed at addressing the needs of administrators, event organizers, and participants, ensuring a seamless experience for all stakeholders.

Scope of Work:

- Event Registration and Tracking:
- Facilitates online registration for events such as seminars, workshops, competitions, and cultural programs.
- Tracks participant details, attendance, and feedback for better insights and reporting.
- Automated Notifications and Updates
- Sends reminders and updates to participants through email or SMS, ensuring effective communication and timely information sharing.

Centralized Database:

- Stores event schedules, participant data, and feedback in a secure and easily accessible format.
- Allows administrators to generate reports for event analysis and future planning.
- Customizable Event Management:
- Supports a variety of event types, including academic, cultural, and sports events, with customizable options to meet specific institutional needs.

Feasibility of Work:

- The project is feasible in terms of:
- Technical Feasibility:
- Developed using widely used technologies like Flask, HTML, CSS, JavaScript, and SQLite, ensuring reliability and scalability.
- Compatible with commonly used operating systems and browsers for broad accessibility.

Operational Feasibility:

- Simplifies workflows for event organizers and participants, reducing administrative workload.
- Provides an intuitive interface, ensuring ease of use for users with basic technical knowledge.

Economic Feasibility:

- Minimal development costs due to the use of open-source technologies.
- Cost-effective maintenance and updates.
- This system is scalable and adaptable, capable of handling events for small institutions as well as larger universities, making it a practical and efficient solution for modern educational environments.

1.4 Operating Environment—Hardware & Software

The College Event Management System (CEMS) is a web-based application designed to operate in a user-friendly environment with minimal hardware and software requirements. It ensures compatibility and accessibility for diverse user groups, including administrators, organizers, and participants.

Hardware Requirements:
Client Side:

- A device with internet connectivity (PC, laptop, tablet, or smartphone).
- Recommended Specifications:
- Processor: Intel Core i3 or equivalent.
- RAM: 4 GB or higher.
- Storage: 10 GB free space for web browsers and local caching.
- Display: Resolution of 1280x720 or higher.

Server Side:

- A web server or cloud-based server for hosting the application.
- Recommended Specifications:
- Processor: Intel Xeon or equivalent.
- RAM: 8 GB or higher.
- Storage: 50 GB or more (for database and application files).
- Network: high-speed internet with a stable connection.

Software Requirements:
Client Side:

- A modern web browser (Google Chrome, Mozilla Firefox, Microsoft Edge) with JavaScript enabled.
- Operating Systems: Windows, macOS, Linux, Android, or iOS.

Server Side:

- Operating System: Linux or Windows Server.
- Web Framework: Flask (Python-based).

Database: SQLite for lightweight operations, with DBeaver CE for management.
Additional Tools: Flask-WTF, Flask-Login, Flask-Bcrypt, and other Python libraries for authentication, form handling, and security.

CEMS is designed to be lightweight and efficient, ensuring it can run smoothly on commonly available hardware and software configurations, making it both accessible and scalable for institutions of all sizes.

1.5 Architecture of the System

The College Event Management System (CEMS) is built using a 3-tier architecture to ensure modularity, scalability, and ease of maintenance. This architecture divides the system into three interconnected layers: the presentation layer, the application layer, and the data layer.

1. Presentation Layer

This is the frontend interface of the system, where users interact with the application. It focuses on providing a user-friendly experience for participants, event organizers, and administrators.
Tools and Technologies: HTML, CSS, JavaScript, Bootstrap.

- Features:
- User registration and login interface.
- Event listing, registration forms, and calendar views.
- Notifications, reminders, and responsive design for compatibility with multiple devices.

2. Application Layer

This layer is the backend logic of the system, managing data processing, business logic, and interactions between the frontend and database.

- Framework: Flask (Python-based).
- Features:
- Handling user authentication and role-based access control (e.g., admin, organizer, participant).
- Event management, such as creating, updating, and deleting events.
- Securing sensitive user data through encryption (e.g., Flask-Bcrypt).

3. Data Layer

The data storage and management layer handles all information related to users, events, and registrations.

- Database: SQLite, managed using DBeaver CE for development.
- Features:
- Storage of user profiles, event details, and participation data.
- Support for CRUD operations (Create, Read, Update, Delete).
- Ensures data integrity and prevents redundancy.
- System Flow:
- Users interact with the Presentation Layer via a web browser.
- Requests are processed in the Application Layer, where Flask routes them to the appropriate functionalities.

The Data Layer retrieves or updates data as required and sends responses back to the application logic, which then updates the user interface.
This modular architecture ensures the system is scalable, secure, and maintainable, enabling smooth operation and adaptability for future enhancements.

1.6 Detailed Description of Technology Used

The College Event Management System (CEMS) employs a combination of cutting-edge technologies to ensure an efficient, secure, and scalable platform for event management.

1. Frontend Technologies

The frontend provides the user interface (UI), ensuring a visually appealing and interactive experience for administrators, faculty, and students.

- **HTML (HyperText Markup Language)**

Purpose: HTML structures the content of web pages. It organizes elements like forms, buttons, navigation bars, and tables to display data and capture user inputs.
Features: semantic tags such as <header>, <footer>, and <main> for improved accessibility and SEO.
Forms and inputs for event creation, user registration, and login.
Advantages:
Simple to use and supported by all browsers.
Works seamlessly with CSS and JavaScript for enhanced functionality.

- **CSS (Cascading Style Sheets)**

Purpose: CSS defines the visual presentation of the application, including layout, colors, and typography.
Features: Enables custom styling for different user roles (admin, faculty, and students).
Media queries ensure responsive design across mobile, tablet, and desktop devices.
Advantages:
Enhances user experience by creating a professional look and feel.
Compatible with frameworks like Bootstrap for faster development.

- **Bootstrap (CSS Framework)**

Purpose: Simplifies the creation of responsive and mobile-first web designs.
Features: Predefined grid systems for consistent layouts.
Components like modals, carousels, and navigation bars that enhance usability.
Advantages:
Reduces development time by offering ready-to-use styles and components.
Ensures uniformity across devices with minimal customization.

- **JavaScript**

Purpose: Adds interactivity and dynamic behavior to the frontend.
Features: Form validation to prevent incorrect inputs during registration or event creation.
Event-based updates, such as displaying success messages or refreshing event lists without reloading the page.
Advantages:
Improves performance by reducing server-side calls.
Supported by a wide range of libraries (e.g., jQuery, AJAX) to simplify development.

2. Backend Technologies

The backend ensures the smooth execution of business logic, manages user requests, and interacts with the database to handle data securely.

- **Flask (Python Web Framework)**

Purpose:Flask serves as the backbone of the application, handling routing, user authentication, and API integration.

Features:

Lightweight Framework: Minimalistic and highly customizable, perfect for small to medium-scale projects.

Extensions Ecosystem: Includes robust libraries like Flask-WTF, Flask-Login, and SQLAlchemy for extended functionality.

Development Speed: Facilitates rapid prototyping and development.

Advantages:

Flexible and easy to learn, even for beginners.

Compatible with modern deployment solutions like Docker, AWS, and Heroku.

Python Libraries and Extensions

- **Flask-WTF:**

Purpose: Simplifies form handling and validation.

Features: Supports CSRF protection and provides reusable form templates.

Advantages: Enhances security and reduces development effort for complex forms.

- **Flask-Login:**

Purpose: Manages user authentication and session control.

Features: Implements role-based access, ensuring only authorized users can access specific features (e.g., event creation for admins).

Advantages: Streamlines login/logout processes and supports persistent sessions.

- **Flask-Bcrypt:**

Purpose: Encrypts user passwords for secure storage in the database.

Features: Uses hashing algorithms to protect sensitive data.

Advantages: Reduces vulnerability to common attacks like password theft and database breaches.

- **Werkzeug:**

Purpose: Assists with secure file uploads and URL routing.

Features: Handles HTTP requests and responses efficiently.

Advantages: Provides robust utilities for building secure and scalable applications.

- **WTForms:**

Purpose: Simplifies structured form creation and validation.

Features: Supports custom validators to ensure input accuracy.

Advantages: Improves form development speed and reliability.

3. Database and Tools

- **SQLite (Database Management System)**

Purpose:
SQLite is used for development due to its simplicity and lightweight nature.
Features:
Stores relational data, including user details, events, and messages.
No need for a separate server, as it operates within the application environment.
Advantages:
Ideal for small-scale projects due to minimal configuration requirements.
Highly portable, making it easy to move between environments.

- **DBeaver (Database Management Tool)**

Purpose:
DBeaver is used to manage and visualize the database during development.
Features:
Provides an intuitive interface for SQL queries, schema management, and debugging.
Supports various databases, making it versatile for future migrations to larger systems like MySQL or PostgreSQL.
Advantages:
Simplifies troubleshooting and improves efficiency.
Enables easy export/import of data for backups or migrations.

- **Advantages of the Technology Stack**

Scalability:
The modular design allows for easy integration of new features, such as feedback forms or analytics.
Security:
Extensions like Flask-Bcrypt and Flask-WTF enhance the security of user data.
Efficiency:
Lightweight tools like Flask and SQLite reduce resource consumption, ensuring faster performance.
User-Friendly:
Frontend technologies like Bootstrap and JavaScript provide a seamless and engaging user experience.

CHAPTER 2: PROPOSED SYSTEM

2.1 Proposed System

The College Event Management System (CEMS) is designed to streamline and automate the management of events within educational institutions. The system aims to simplify event scheduling, user management, communication, and resource allocation through an intuitive web-based interface. The core objective of the proposed system is to provide a seamless platform where faculty, students, and administrators can interact, manage events, and track event-related information efficiently.

Features of the Proposed System:

- User Management:

The system will support three user roles: Admin, Faculty, and Student.
Admin: Full access to the system. Can create, edit, or delete events, manage users, and generate reports.
Faculty: Can create and manage events, send messages, and view relevant event data.
Student: Can register for events, view event details, and communicate with faculty.

- **Event Creation and Management:**

Faculty and Admin users can create and manage events, including the ability to add event titles, descriptions, dates, and images.
Events can be categorized based on type, such as academic, cultural, sports, etc.
Events will be listed on a calendar interface, providing users with an easy-to-navigate event schedule.

- **User Registration and Authentication:**

The system will offer secure user authentication, including user login and registration with role-based access control.
Passwords will be securely stored using hashing algorithms to ensure data privacy and security.

- **Event Registration:**

Students can register for events they are interested in. They will receive notifications about upcoming events, changes, or cancellations.

- **Message System:**

A messaging feature will allow users to send and receive notifications related to events, such as reminders, changes, or event-related updates.
Faculty can send targeted messages to students or other faculty members based on event categories or individual user roles.

- **Database Management:**

The system will utilize a relational database to store and manage user and event data. The database will ensure fast and reliable data retrieval and secure storage of sensitive information.
The database will also allow easy scaling as the number of users and events increases.

- **Responsive User Interface:**

The frontend of the system will be built using HTML, CSS, and JavaScript to ensure a responsive and user-friendly interface.
The design will be mobile-friendly, providing an optimized experience across devices such as desktops, tablets, and smartphones.

- **Admin Dashboard:**

Admin users will have access to a dashboard for monitoring event activity, viewing statistics, and managing user accounts.
The dashboard will display real-time updates on event registrations, pending tasks, and overall system health.

- **System Flow:**

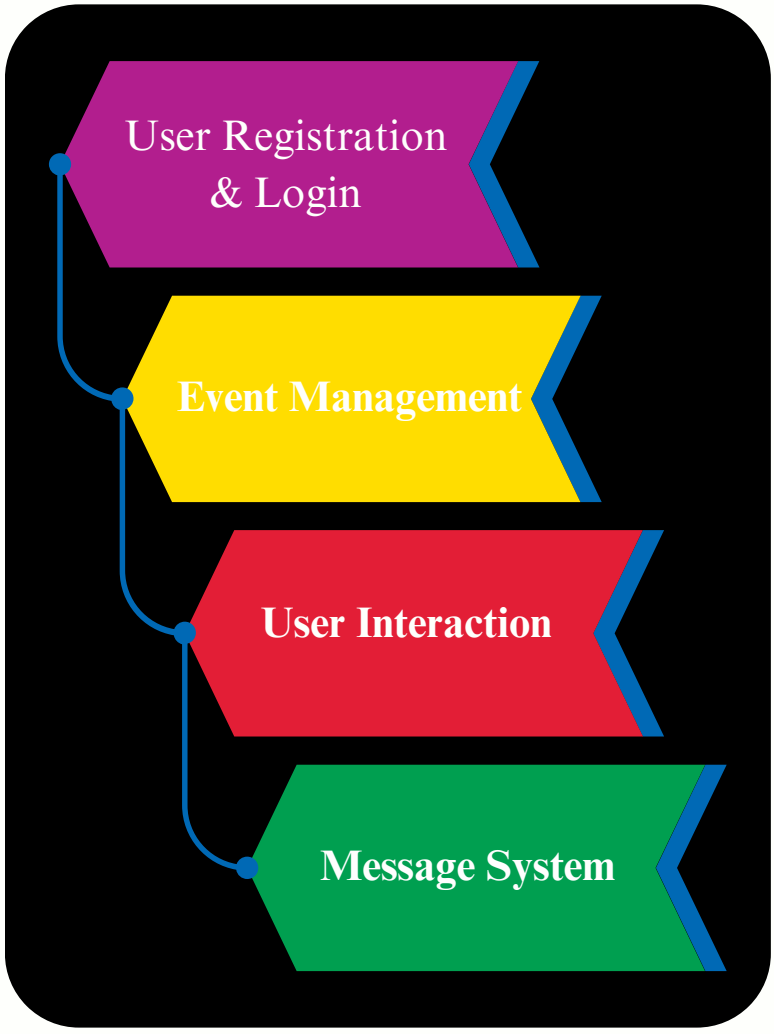


Fig 2.1.1

System Flow Fig2.1.1 in detail:

1. User Registration and Login:
   - Users will register via the web interface, selecting their role (Admin, Faculty, or Student).
   - After registration, they can log in using their credentials. Authenticated users are redirected to the appropriate dashboard based on their role.
2. Event Management:
   - Faculty or Admin users can create events using a form that includes the title, description, date, and other relevant information.
   - Events will be stored in the database and displayed on the event calendar.
   - Students can view event details and register directly through the event listing.
3. User Interaction:
   - Students will receive notifications about upcoming events and registration deadlines.
   - Faculty and Admin can send personalized messages to students and other faculty members regarding event updates, cancellations, or feedback.
4. Message System:
   - The messaging feature will allow users to interact directly with others for event-related inquiries. This will streamline communication between faculty, students, and administrators.

Advantages of the Proposed System:

- Efficiency and Automation:

By automating event management tasks such as registration, notifications, and reporting, the system saves time and reduces the chances of human error.

- Role-Based Access Control:

The system supports different user roles, ensuring that users have access only to the features relevant to their role. This maintains security and data integrity.

- Seamless Communication:

Integrated messaging functionality ensures continuous communication among users, which enhances collaboration and transparency across departments.

- Scalability:

As the institution grows, the system can easily handle an increasing number of users and events by expanding the database and adjusting the server infrastructure.

- User-Friendly Interface:

The responsive and intuitive design ensures that users can quickly learn how to interact with the system, leading to higher adoption rates among students, faculty, and administrators.

- Security:

Secure authentication mechanisms and password encryption ensure that user data is protected from unauthorized access.

2.2 Objectives of System

The College Event Management System (CEMS) is designed with the primary goal of automating and simplifying the entire process of event management within a college or university setting. The system seeks to provide an efficient, reliable, and user-friendly platform for organizing, registering, and tracking events. The following are the specific objectives of the system:

1. Efficient Event Creation and Management:

- Objective: To provide an intuitive and streamlined platform for faculty, administrators, and event organizers to create and manage events effortlessly.
- Details: Faculty and admin users can create events by providing necessary details such as event title, description, date, and venue. The system will automatically store and organize these events for easy access and modification.

2. Role-Based User Management:

- Objective: To enable different types of users (Admin, Faculty, and Students) to interact with the system according to their roles and permissions.
- Details: Admins have full control over user accounts and event management, while faculty can create and manage events, and students can view and register for events. Role-based access control ensures security and prevents unauthorized access to sensitive data.

3. Streamlined Event Registration and Notifications:

- Objective: To simplify the event registration process for students and keep them informed about important event updates.
- Details: Students can easily register for events and receive automated notifications about event updates, registration deadlines, and changes. The system ensures timely reminders and notifications via email or on the dashboard.

4. Centralized Communication Platform:

- Objective: To provide a centralized messaging platform for communication between faculty, students, and administrators.
- Details: The system will allow faculty and administrators to send messages to individual users or groups. These messages may include event-related updates, reminders, or any other important notifications, enhancing communication and reducing delays in information dissemination.

5. Secure and Scalable System:

- Objective: To ensure that the system is secure, scalable, and capable of handling an increasing number of users and events.
- Details: The system will utilize secure authentication (e.g., hashed passwords) and be designed for easy scalability. This means that as the college grows, the system can handle more users and events without compromising performance.

6. Interactive Event Calendar:

- Objective: To provide an easy-to-use and interactive event calendar that displays all upcoming events and allows users to view and register for events conveniently.
- Details: Events will be displayed in a calendar format, making it easy for users to browse and register for events based on their dates and categories. The calendar will be updated in real-time to reflect changes or new events.

7. Data-driven Insights and Reports:

- Objective: To provide administrators with the ability to generate real-time reports on event attendance, registration numbers, and user activity.
- Details: Admin users can access reports that provide valuable insights into event registrations, user engagement, and event feedback, helping them make data-driven decisions for future events.

8. Enhanced User Experience:

- Objective: To deliver a smooth, responsive, and intuitive user interface that encourages widespread adoption and ease of use.
- Details: The system's frontend will be designed using HTML, CSS, JavaScript, and responsive design principles to ensure accessibility across all devices. Whether using a desktop, tablet, or mobile phone, users will be able to interact with the system seamlessly.

9. Simplified event updates and cancellations:

- Objective: To enable quick and efficient updates or cancellations of events, keeping users informed about any changes in real time.
- Details: If an event is modified or canceled, the system will automatically notify the affected users, ensuring that everyone is aware of the changes in time.

10. Improved Administrative Efficiency:

- Objective: To automate administrative tasks related to event management and user interactions, reducing manual effort and enhancing efficiency.
- Details: The system will automate tasks such as event creation, user management, and communication, freeing up administrative time and reducing the chances of human error.

The College Event Management System (CEMS) aims to simplify and automate key processes within event management in a college or university setting. With features like efficient event creation, role-based management, event registration, centralized communication, and data-driven reports, the system will greatly enhance the user experience for all stakeholders. The objectives outlined above ensure that the system is not only user-friendly but also secure, scalable, and capable of meeting the growing needs of educational institutions.

2.3 User Requirements

The College Event Management System (CEMS) is designed to meet the diverse needs of its users, including students, faculty, and administrators. Below is a detailed overview of the user requirements categorized by role.

1. Student Requirements:

Event Browsing and Registration:

- View a list of all available events categorized by type (e.g., academic, cultural, sports).
- Access detailed information about each event, including the title, description, date, venue, and organizer.
- Register for events easily through the platform.

User-Friendly Dashboard:

- Access a personalized dashboard displaying registered events and their statuses.

2. Faculty Requirements:

Event Creation and Management:

- Create events by providing details such as title, description, date, and images.
- Update or cancel events when necessary.
- Track registrations for events and view attendee lists.

Messaging:

- Send announcements or updates to registered participants.

Role-Specific Dashboard:

- Access a dashboard to monitor all events they have created or are associated with.

Secure Authentication:

- Use a secure login system to access faculty-specific features.

3. Administrator Requirements:

User and Role Management:

- Add, modify, or delete user accounts.
- Assign roles (admin, faculty, student) to users and manage access levels.

Event Oversight:

- Monitor all events created on the platform to ensure compliance with institutional policies.
- Approve or reject events before they are published, if needed.

System Reports:

- Generate reports on event registrations, user activity, and overall platform performance.

System Configuration:

- Manage system-wide settings such as notification preferences, default roles, and data backups.

Messaging System:

- Send important announcements to all users or specific groups (e.g., all students).

CHAPTER 3 : ANALYSIS & DESIGN

3.1 DFD-Data Flow Digram

The DFD shows the flow of data between different parts of the system:

- Actors:
  - Admin: Manages events and users.
  - Faculty: Sends messages to the admin.
  - Student: views events and registers for them.
- Processes:
  - User Authentication (Login/Registration)
  - Event Management (Create, Edit, View, Delete)
  - Message System (Faculty to Admin)

DFD Levels

Level 0: High-Level Overview
This represents the system as a single process.

- Inputs:
  - User credentials, event data, messages.
- Outputs:
  - Login success/failure, event details, admin responses.

- DFD Digrams

DFD Level 0

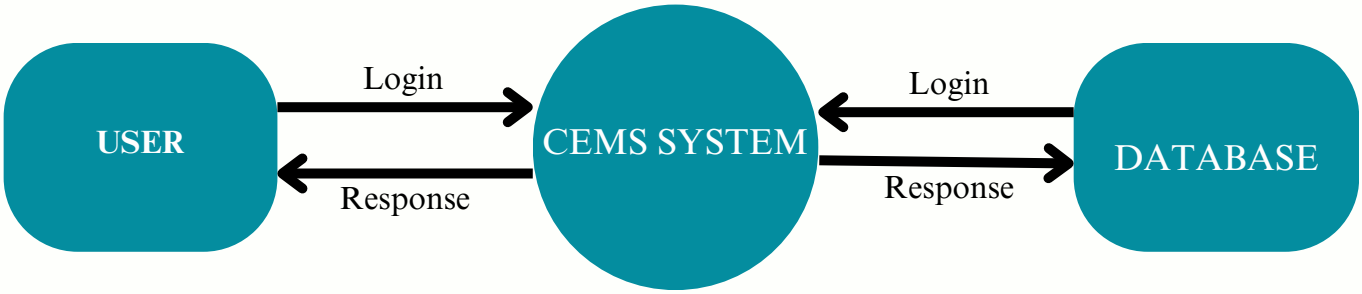


Fig 3.1.1

Level 1: Detailed View
Divides the main process into sub-processes:

1. Authentication:
   - Login/Registration routes process credentials and retrieve user data.
2. Event Management:
   - Admin creates or modifies events stored in the database.
   - Students fetch events.
3. Messaging:
   - Faculty sends messages, which are stored and retrieved by Admin.

- DFD Digrams

  DFD Level 1

Fig 3.1.2

Fig 3.1.3

Fig 3.1.4

3.2. Database Table Specifications

`User` **Table**

Column Name	Data Type	Constraints	Description
`id`	INTEGER	Primary Key, Auto-Inc	Unique ID for the user
`username`	VARCHAR(150)	NOT NULL	User's username
`email`	VARCHAR(150)	NOT NULL, UNIQUE	User's email
`password`	VARCHAR(255)	NOT NULL	Hashed password
`role`	VARCHAR(50)	NOT NULL	Role (admin, faculty, student)

Fig 3.2.1

`Event` **Table**

Column Name	Data Type	Constraints	Description
`id`	INTEGER	Primary Key, Auto-Inc	Unique ID for the event
`title`	VARCHAR(255)	NOT NULL	Event title
`description`	TEXT	NOT NULL	Event description
`date`	DATE	NOT NULL	Event date
`user_id`	INTEGER	Foreign Key	Creator's user ID

Fig 3.2.2

`Message` **Table**

Column Name	Data Type	Constraints	Description
`id`	INTEGER	Primary Key, Auto-Inc	Unique ID for the message
`sender_id`	INTEGER	Foreign Key	Sender's user ID
`content`	TEXT	NOT NULL	Message content
`is_read`	BOOLEAN	Default 0	Indicates if the message is read

Fig 3.2.3

3.3. Entity-Relationship Diagram (ERD)

The ERD represents relationships between entities (tables):

1. Entities:
   - User
   - Event
   - Message
2. Relationships:
   - User creates multiple Events (1-to-Many).
   - User sends multiple Messages (1-to-Many).

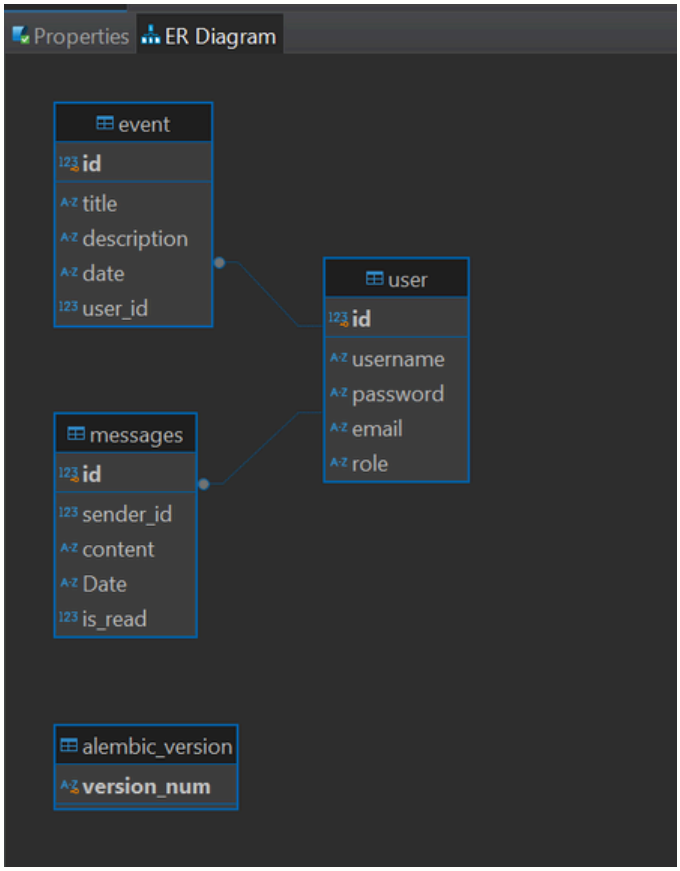


Fig 3.3.1
ER Digram Of College Event Management System

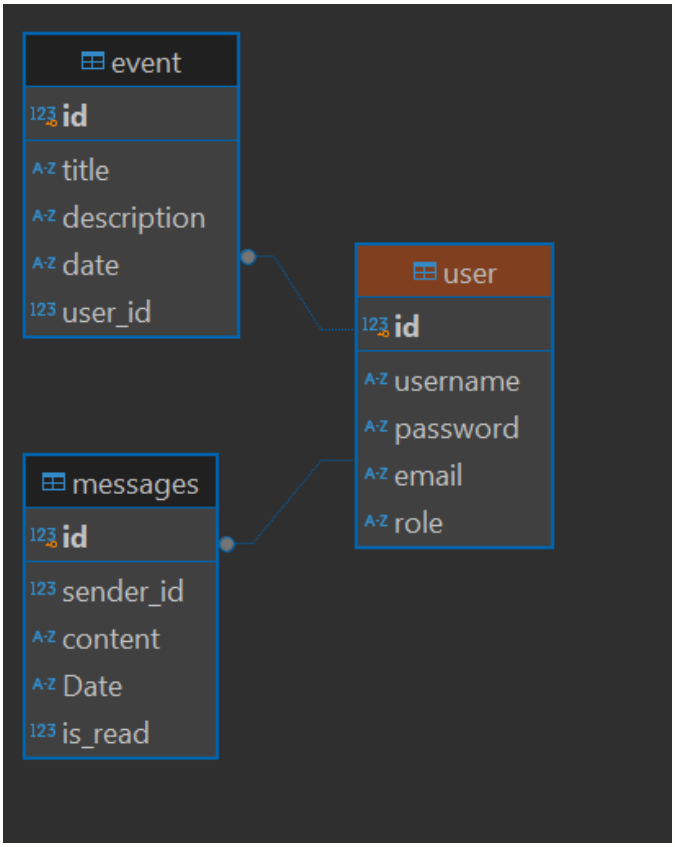


Fig 3.3.2
ER Digram User

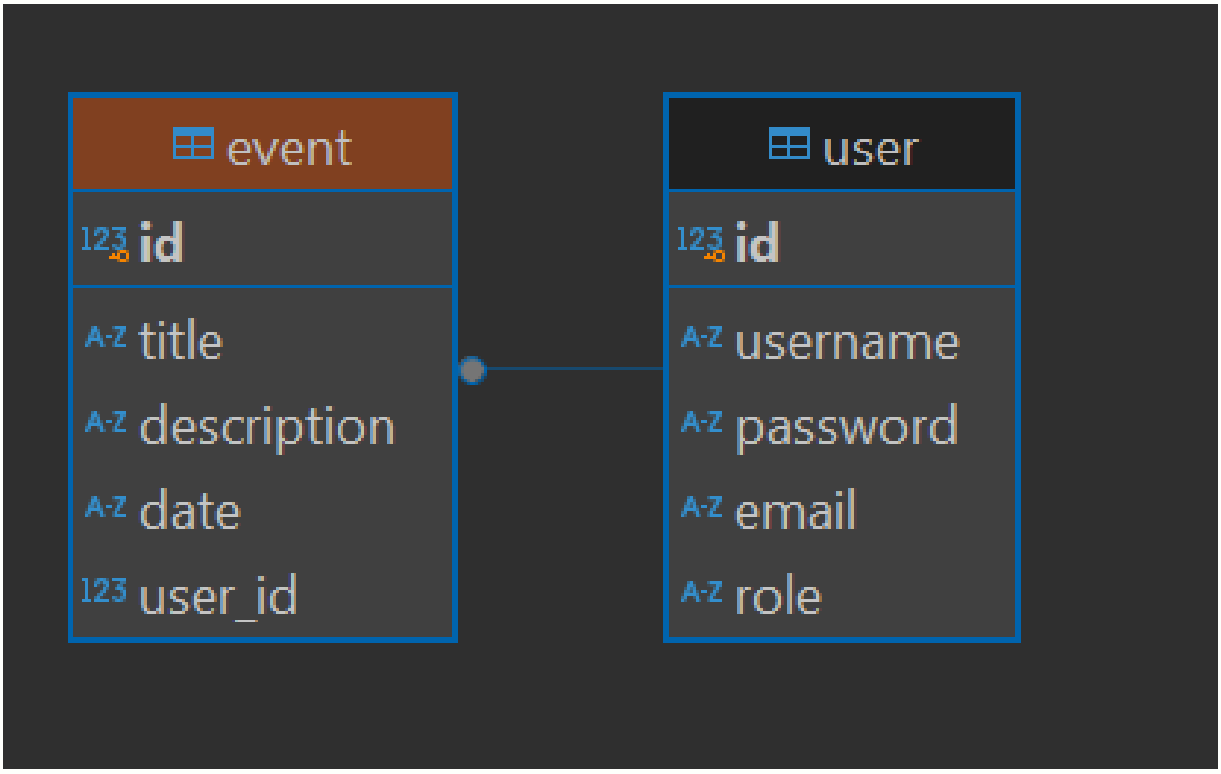


Fig 3.3.3 - ER Diagram Of Event Table

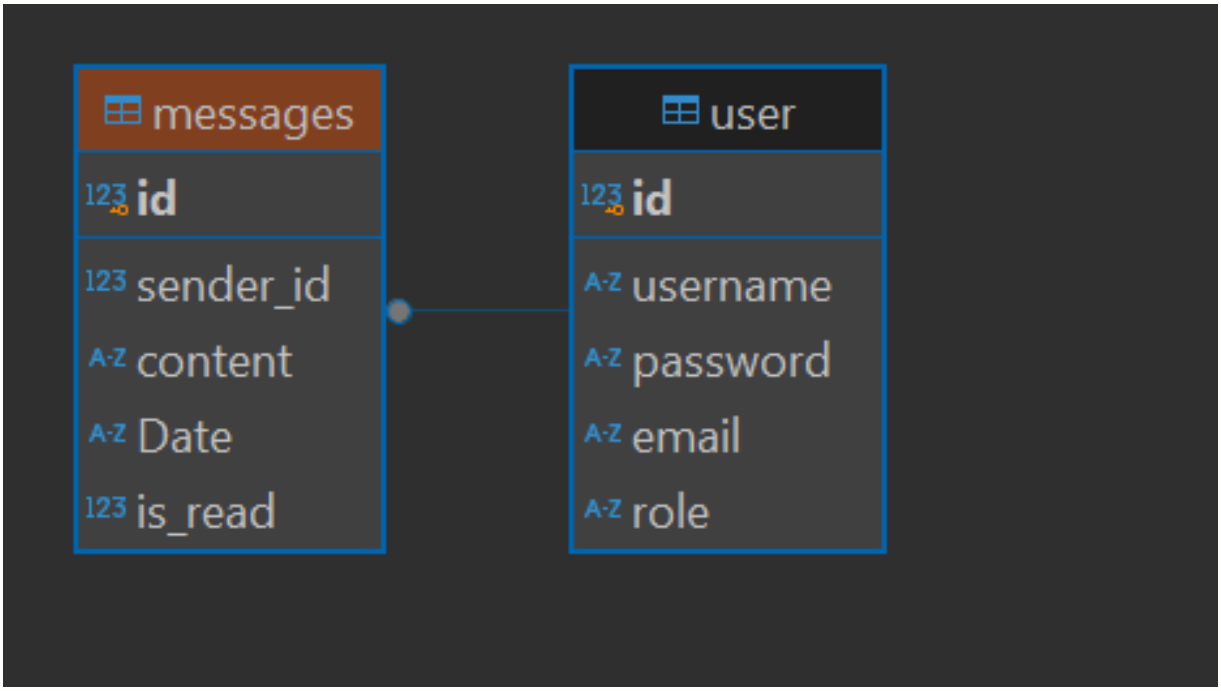


Fig 3.3.4 - ER Diagram Of MessageTable

3.4. Object Diagram

User Object
Attributes:

- id: Integer (Primary Key)
- username: String
- email: String
- password: String
- role: String (e.g., Admin, Faculty, Student)

Relationships:
One-to-many with Event (A user can create multiple events)
One-to-many with Message (A user can send multiple messages)
Event Object Attributes:

- id: Integer (Primary Key)
- title: String
- description: String
- date: Date
- user_id: Integer (Foreign Key, links to User)

Relationships:
Many-to-one with User (An event is created by one user)
Message Object Attributes:

- id: Integer (Primary Key)
- sender_id: Integer (Foreign Key, links to User)
- content: String
- is_read: Boolean (To indicate if the message is read or not)

Relationships:
Many-to-one with User A message is sent by one user.

Explanation of Diagram:

- User: This entity holds information about the users in the system, including their role, username, and email.
- Event: This entity contains the event details, such as the title, description, date, and the user_id which ties the event to the user who created it.
- Message: This entity holds messages sent by users (typically for admin communication), and each message is associated with a sender (sender_id links to User).

Relationships:

- A User can have many Events and many Messages (one-to-many relationships).
- An Event is associated with only one User (many-to-one).
- A Message is sent by one User, but a user can send multiple messages (many-to-one).

3.4.1 Object Diagram

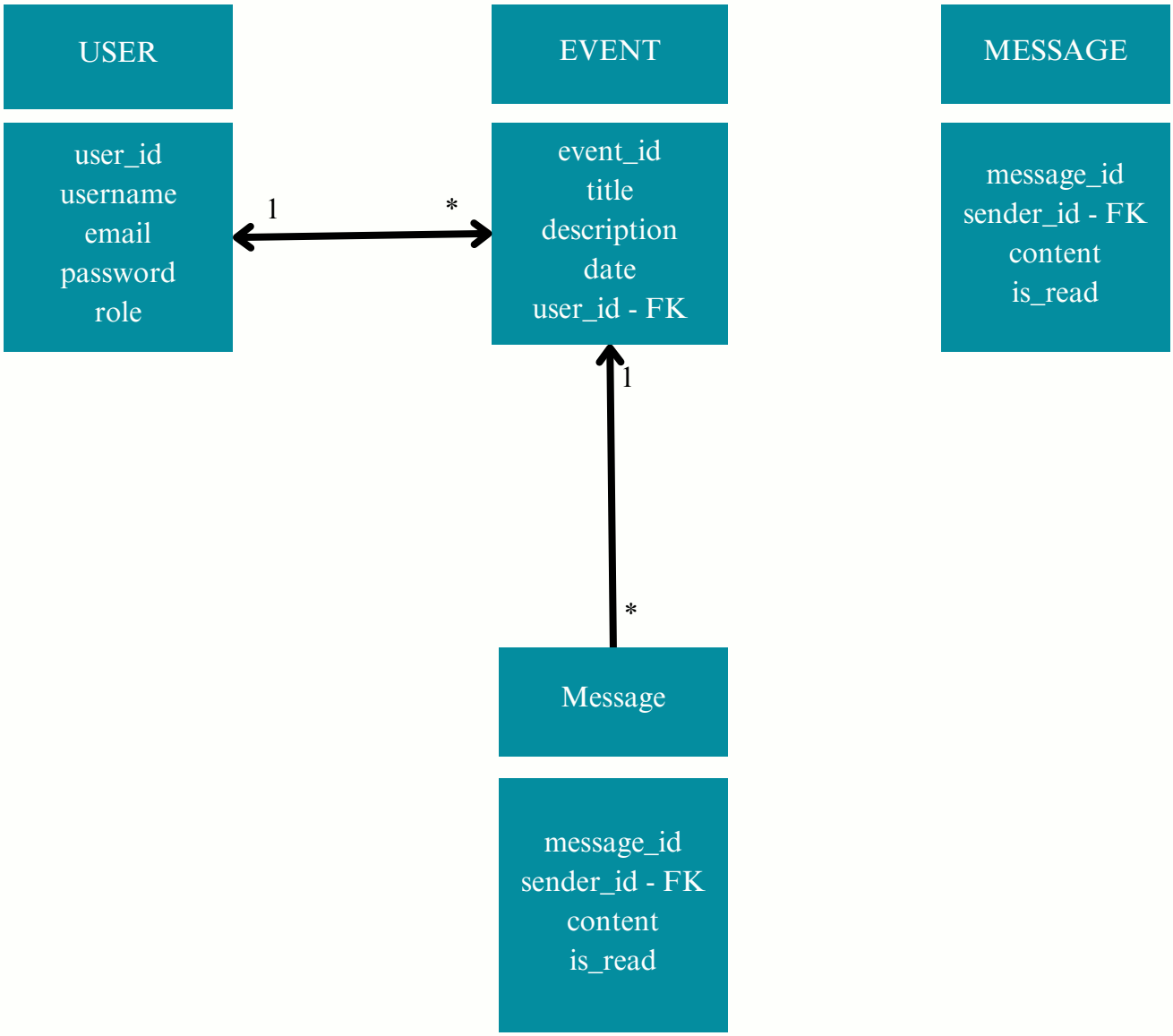


Fig 3.4.1 - Object Diagram

3.5 Class Diagram

A class diagram is used to show the structure of a system by defining its classes, attributes, methods, and the relationships among the objects. Below is a textual explanation for the class diagram for your college event management system (CEMS), followed by a description of its components and relationships.

- User Class represents a user in the system. It contains attributes like user_id, username, email, password, and role. It also has methods like register(), login(), and logout().
- Event Class contains attributes for event details such as event_id, title, description, and date. It has methods like create_event(), update_event(), delete_event(), and view_event().
- Message Class represents the messages exchanged in the system. It contains attributes like message_id, sender_id, content, and is_read. It also has methods like send_message() and mark_as_read().
- EventForm Class is used to handle event form data and validation. It has attributes like title, description, and date. Its methods include validate_on_submit() and populate_event_data().
- LoginForm Class represents the login form with username and password attributes, and a method validate_on_submit().

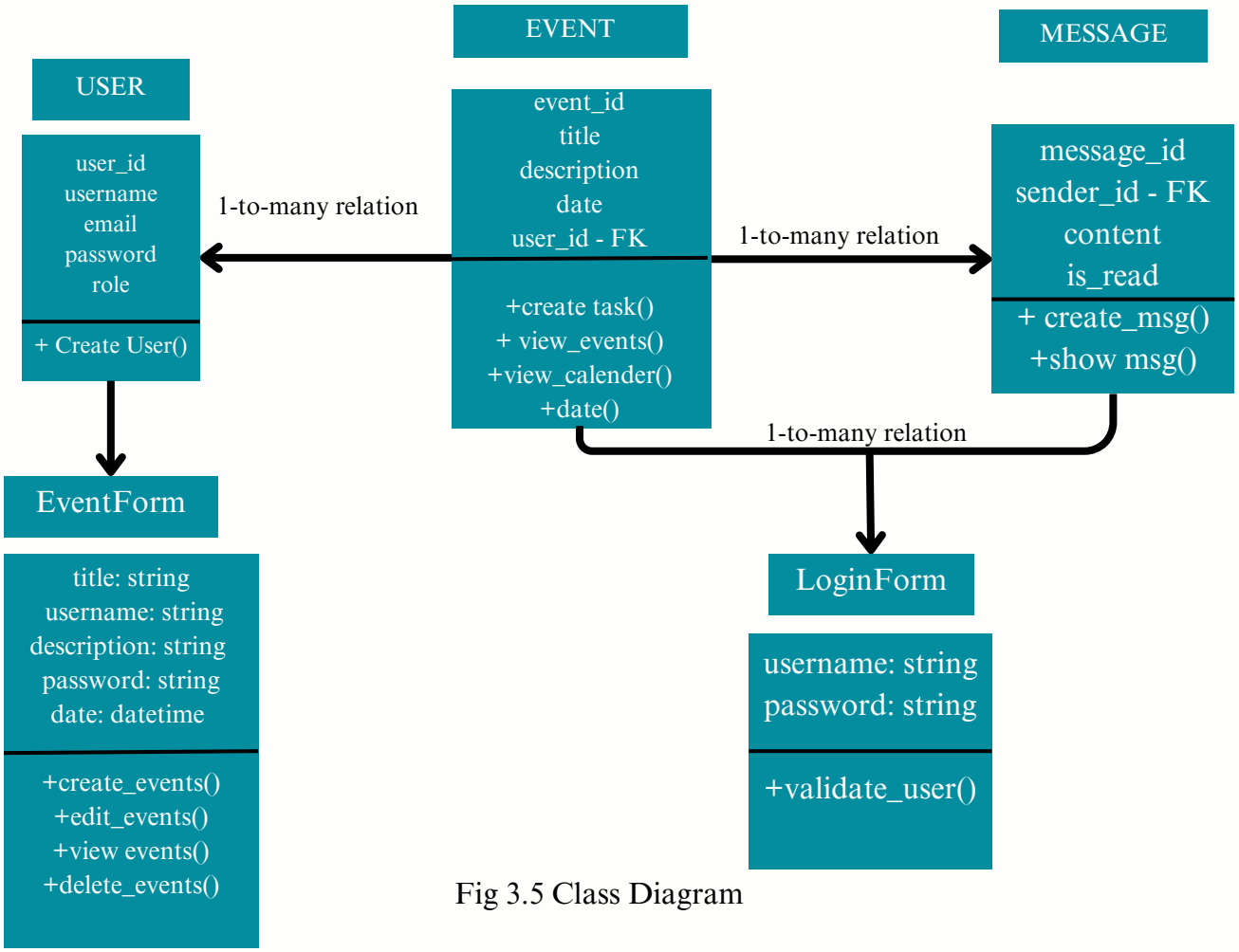


Fig 3.5 Class Diagram

3.6 Use Case Diagram

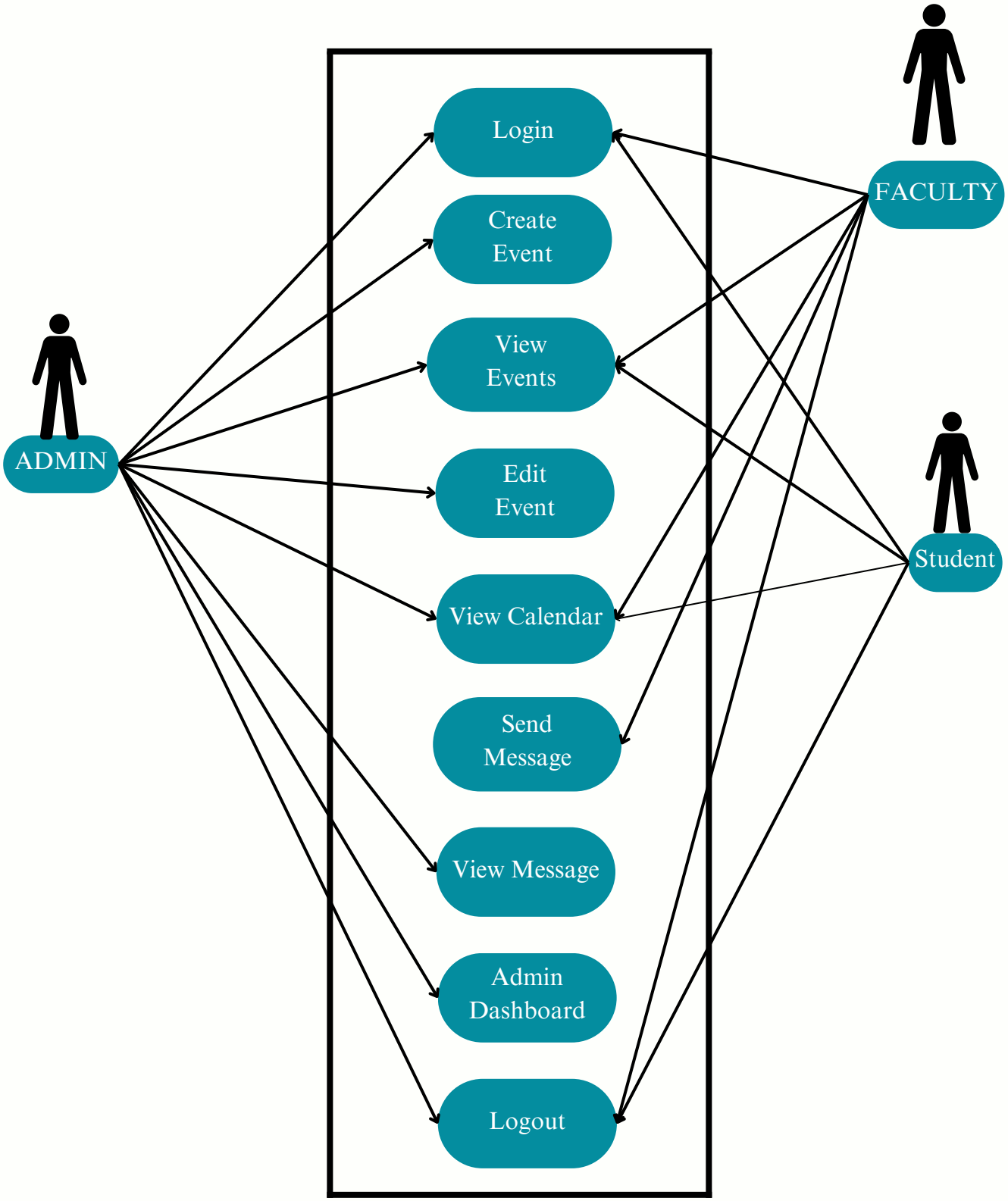
A use case diagram represents the functional requirements of the system from the user's perspective. Below is a textual description and explanation of the use case diagram for your College Event Management System (CEMS).

Actors in the Use Case Diagram:

1. Admin: Manages the creation, editing, and deletion of events.
2. Faculty: Can send messages to the admin and view events.
3. Student: Can view events.

Use Cases:

- Admin:
    - Create Event: The admin can create new events.
    - Edit Event: The admin can edit existing events.
    - Delete Event: The admin can delete events.
    - View All Events: The admin can view all the events created by themselves or others.
- Faculty:
    - Send Message: The faculty can send messages to the admin.
    - View Events: The faculty can view events.
- Student
    - View Events: The student can view the events.



3.6 Use Case Diagram

CHAPTER 4: USER MANUAL

4.1 Limitations

The College Event Management System (CEMS) is designed to streamline event management processes for users such as students, faculty, and administrators. However, as with any system, there are certain limitations that users should be aware of:

1. Limited User Roles:
   - The system currently supports only three primary roles: admin, faculty, and student. There may be a need for additional roles in larger or more diverse institutions (e.g., event coordinators, department heads).
2. No Mobile Application:
   - The system is currently designed only for web access. There is no dedicated mobile application, which limits the convenience for users who prefer accessing event details via smartphones or tablets.
3. Limited Communication Features:
   - The messaging system is rudimentary, requiring manual handling of messages by faculty or administrators. The system does not support advanced features such as chatbots or automated responses for common queries.
4. No Calendar Integration:
   - Events are displayed within the system, but there is no integration with external calendars (such as Google Calendar or Outlook). This limits users' ability to sync events with their personal calendars.
5. Basic Event Management Features:
   - While events can be created, edited, and viewed, the system lacks more advanced event features like recurring events, event categorization, or multi-day event support. Additionally, there is no notification system for event changes.

4.2 Future Enhancements

There are several ways the College Event Management System can be improved in the future to better serve its users and provide more robust features:

Introduction of Additional User Roles:

- Future versions can introduce more user roles such as event coordinators, volunteers, and department heads, allowing for more granular permissions and better role-based access to system features.

Mobile Application Development:

- Developing a mobile app for iOS and Android would significantly enhance the accessibility of the system, allowing users to manage and register for events on the go.

Advanced Communication Features:

- Implementing a real-time chat system or integrating with external messaging platforms (e.g., WhatsApp or Slack) for more efficient communication between event organizers and participants.

External Calendar Integration:

- Integrating the system with popular external calendar applications such as Google Calendar and Outlook will allow users to easily sync their event schedules with their personal calendars.

Event notifications and reminders:

- The future version of the system could include a notification and reminder system, which would alert users about event updates, cancellations, and reminders of upcoming events via email or push notifications.

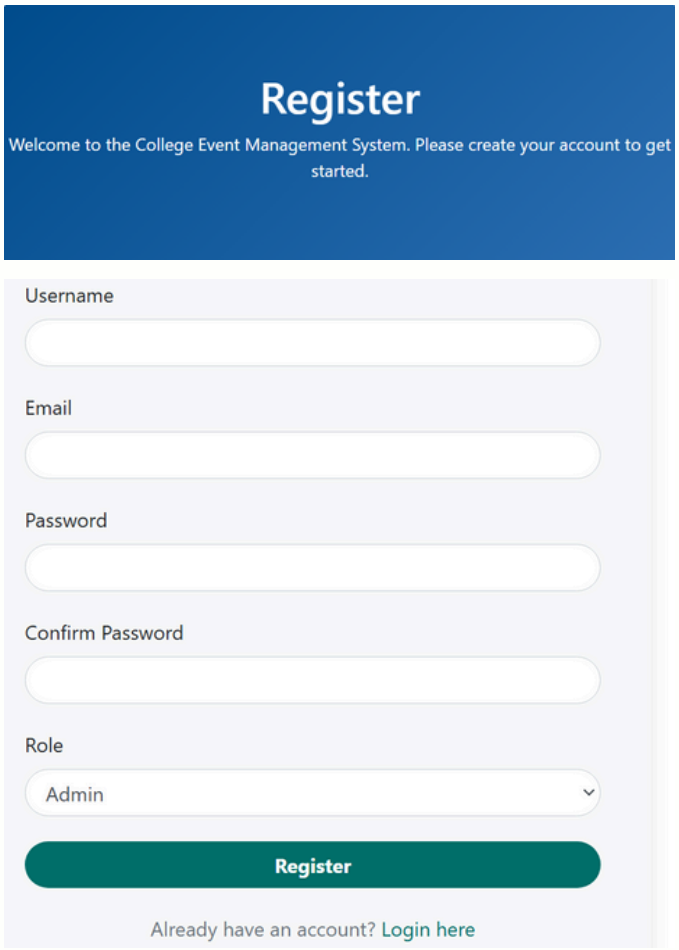
4.3 BIBLIOGRAPHY

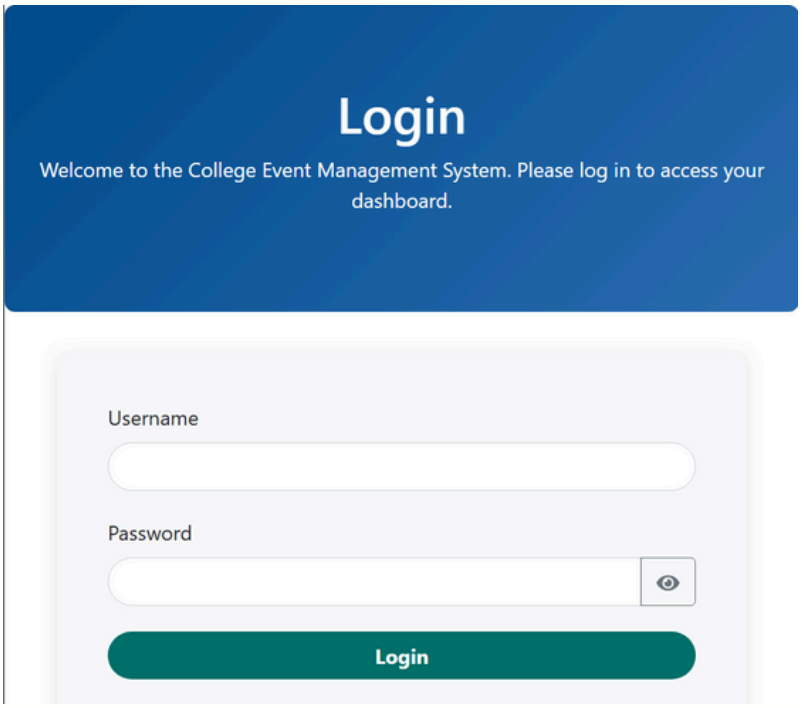
For the development of the College Event Management System, various resources were consulted to ensure that the system meets both user needs and technical standards. Below are some key references used in the development of the project:

1. Flask Documentation:
   - Flask is a web framework used to develop the back-end of the College Event Management System. The official documentation provided the necessary guidelines and functions to effectively build routes, manage templates, and handle user sessions.
   - Source: Flask Documentation
2. SQLAlchemy Documentation:
   - SQLAlchemy is used as the Object Relational Mapper (ORM) in the system to manage database interactions. The SQLAlchemy documentation helped in understanding how to model tables and relationships in the database.
   - Source: SQLAlchemy Documentation
3. Flask-Login Documentation:
   - Flask-Login is used for managing user authentication. The official documentation was crucial for understanding how to implement user login, session management, and role-based access control.
   - Source: Flask-Login Documentation
4. Python Documentation:
   - The official Python documentation was referenced for understanding various concepts such as decorators, error handling, and data manipulation that were used throughout the system.
   - Source: Python Documentation
5. Django documentation (for future enhancements):
   - Although this project is based on Flask, the Django framework was studied for potential future features such as improved admin interfaces and ORM features that could be considered in further system upgrades.
   - Source: Django Documentation

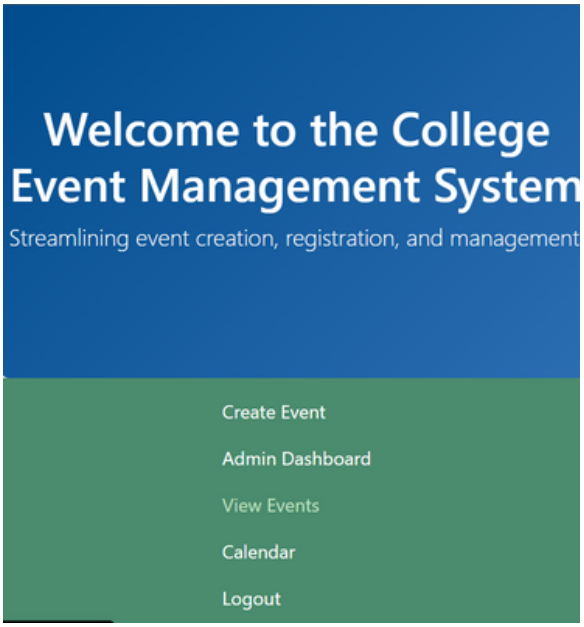
4.4 User Interface Design (Screens etc.)

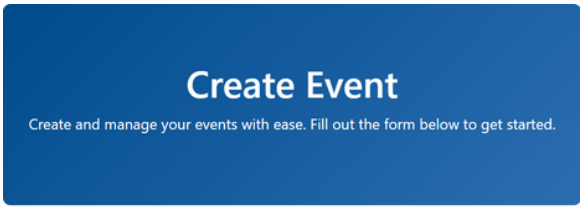
Index Page

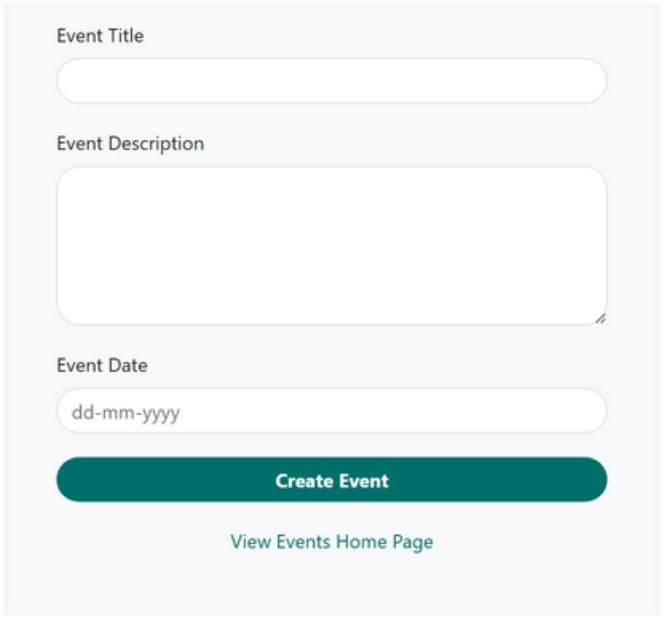


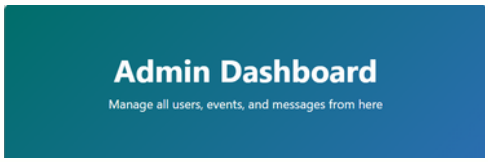


User - Admin









All Users

ID	Username	Email	Role
1	Vaishnavi	Vaishnavi.Kurunde24@gmail.com	admin
2	Jaywant	vkurundecoder001@gmail.com	faculty

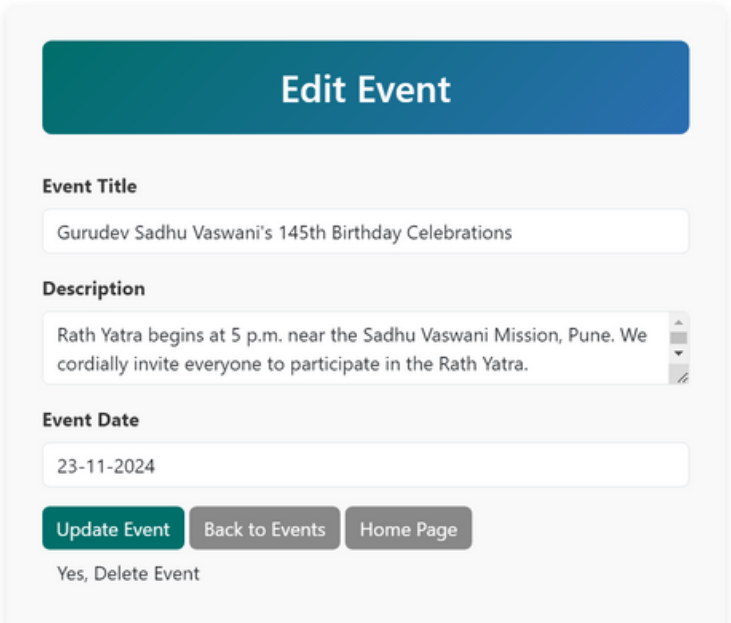
All Events

ID	Title	Description	Date	Creator
2	Gurudev Sadhu Vaswani's 145th Birthday Celebrations	Rath Yatra begins at 5 p.m. near the Sadhu Vaswani Mission, Pune. We cordially invite everyone to participate in the Rath Yatra. Reporting Time: 4.45pm	2024-11-23 00:00	Vaishnavi

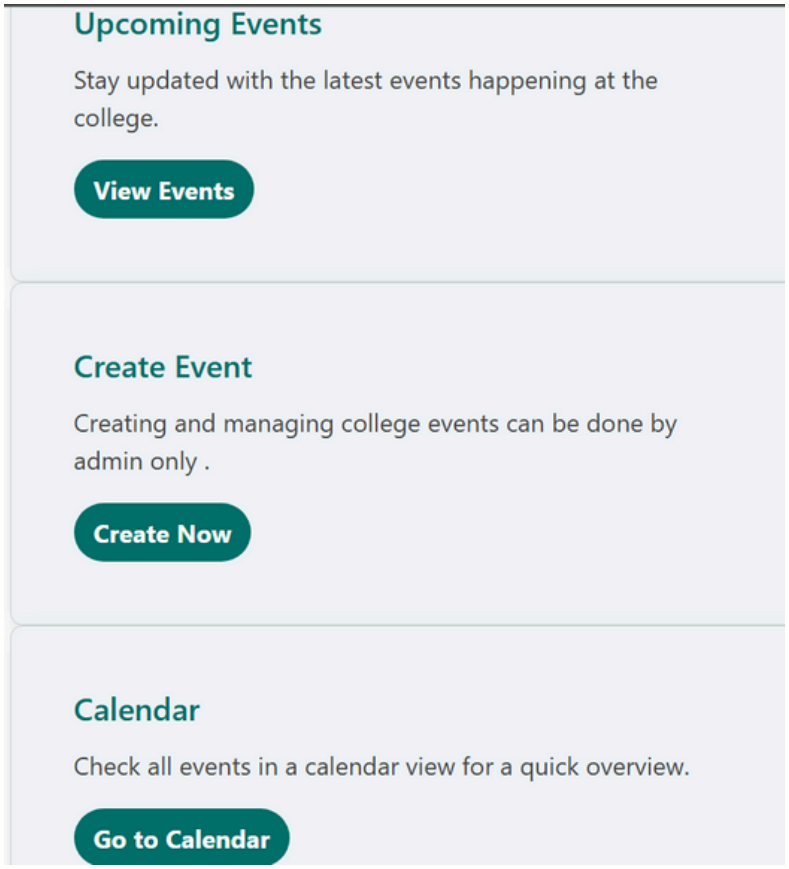
Messages from Faculty

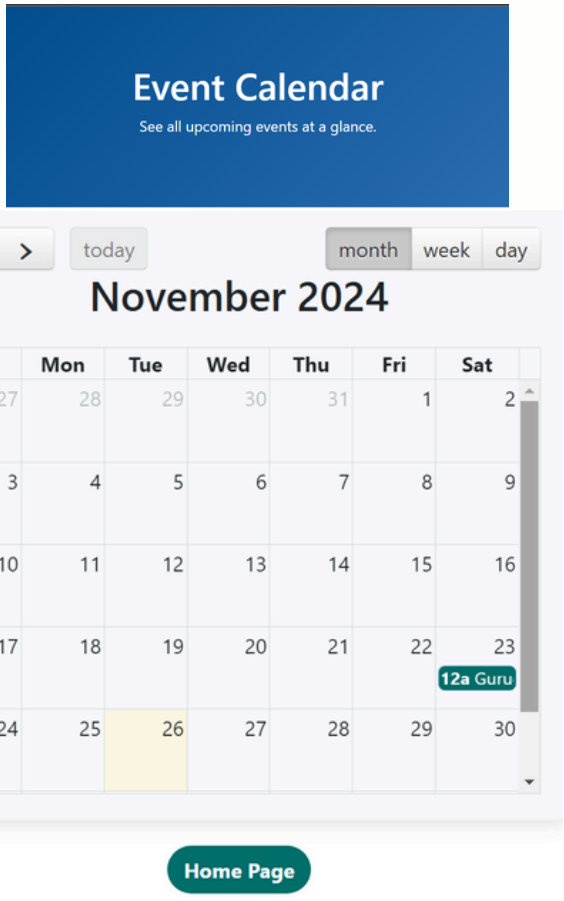
ID	Sender ID	Content	Date
1	2	Make some changes in Rangoli event.	2024-11-13 08:09
2	2	Please include students too in event. For 23 Nov of Rath Yatra.	Not available

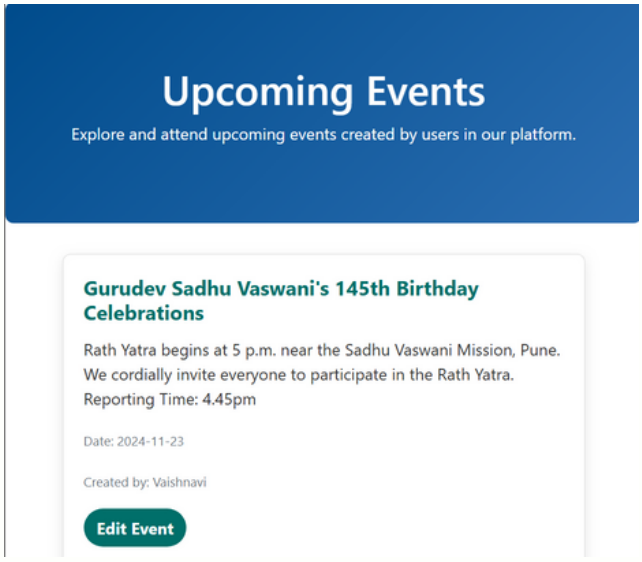
Home Page | View Events | Create Events



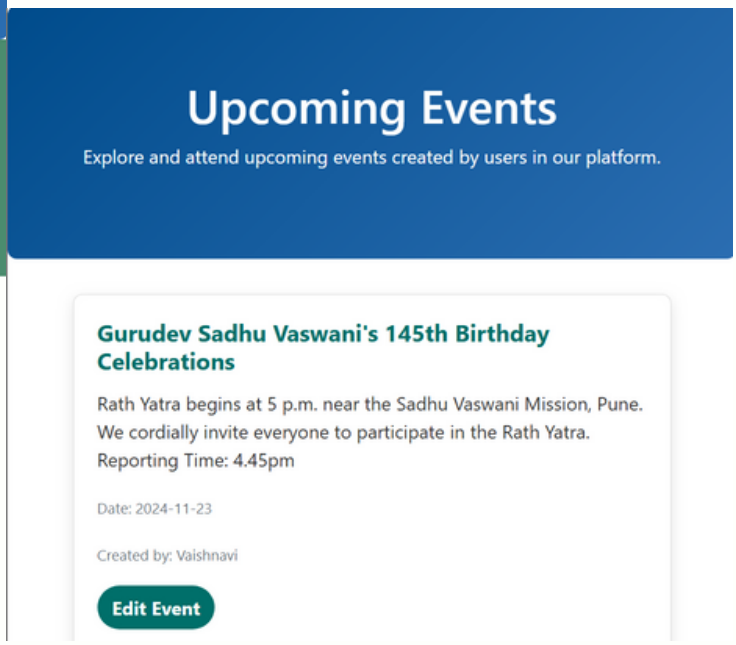
Common Pages for All users

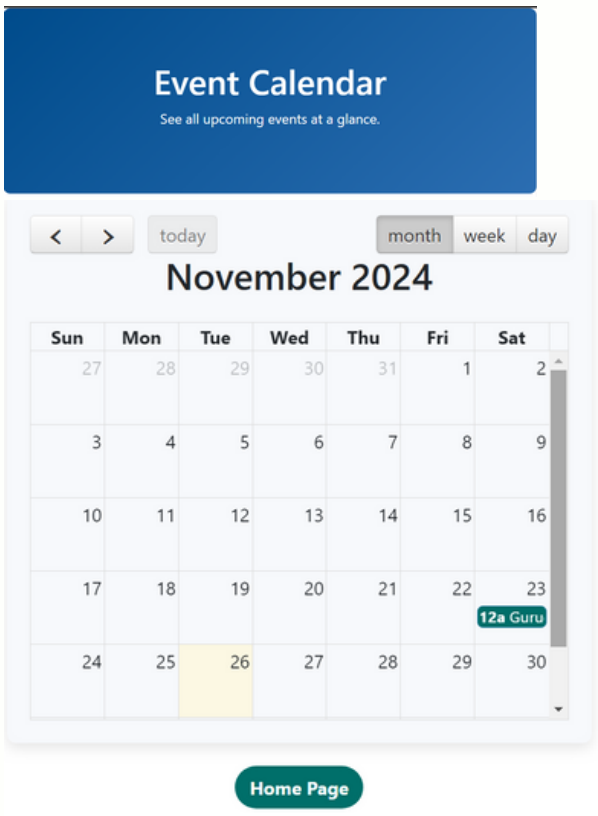


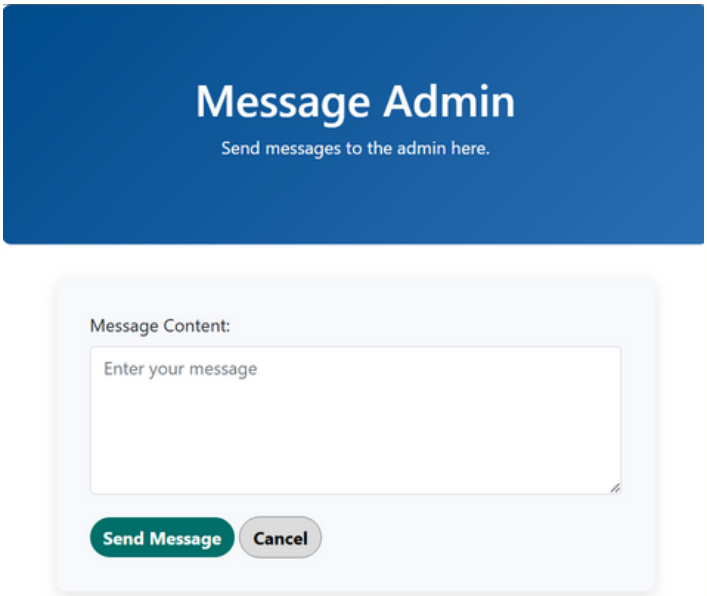




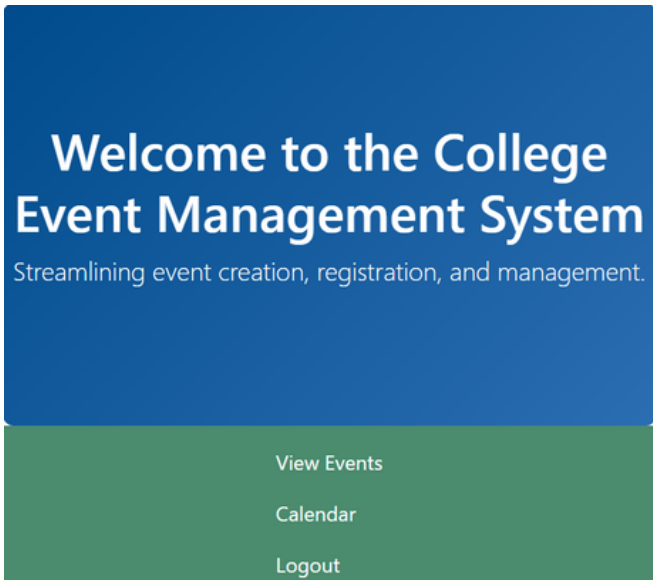
User- Faculty

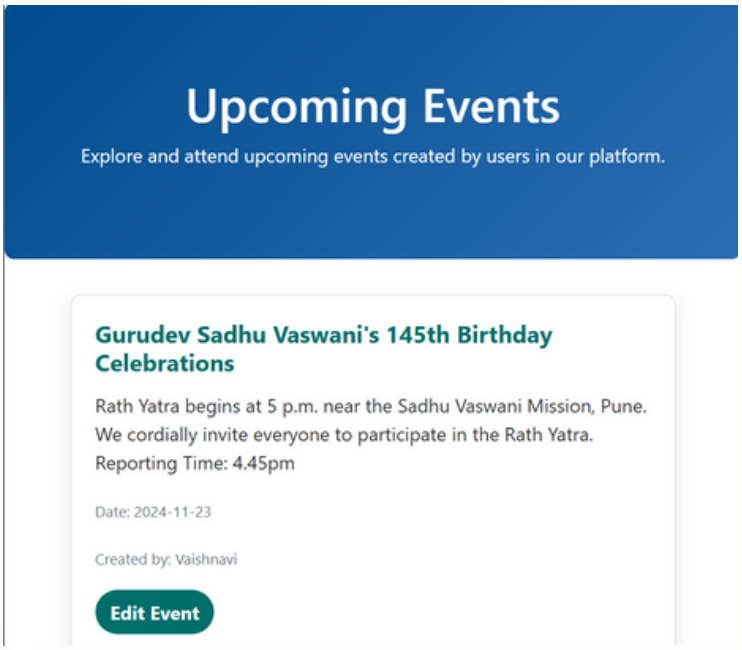


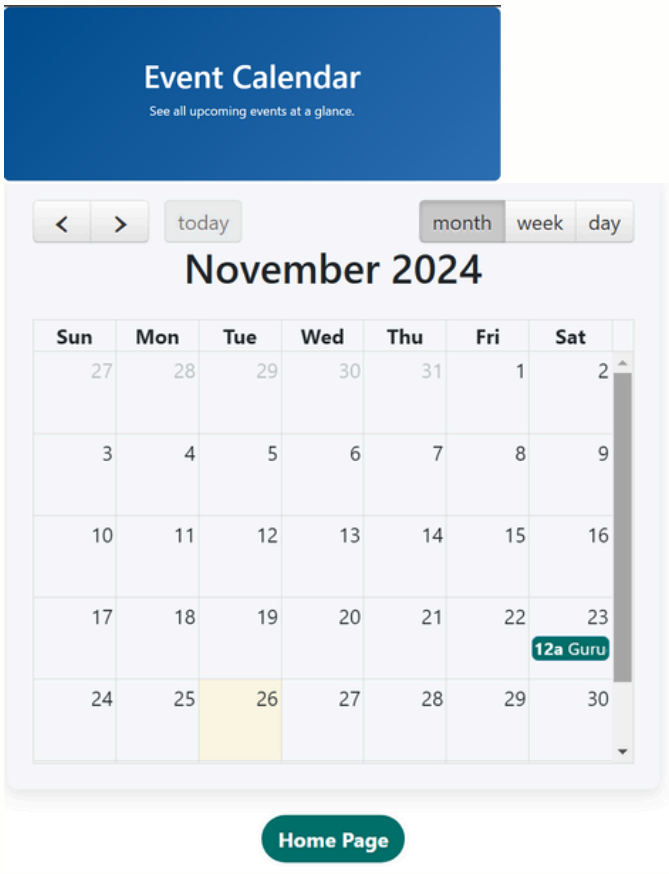




User- Student







ANNEXURE:

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import datetime
from werkzeug.security import generate_password_hash, check_password_hash

db = SQLAlchemy()

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), nullable=False, unique=True)
    password = db.Column(db.String(150), nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    role = db.Column(db.String(50), nullable=False)  # 'admin', 'faculty', or 'student'

    def set_password(self, password):
        """Hash and set the user's password."""
        self.password = generate_password_hash(password)

    def check_password(self, password):
        """Check hashed password."""
        return check_password_hash(self.password, password)

    def __repr__(self):
        return f'<User {self.username}>'

class Event(db.Model):

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.String(200), nullable=False)
    date = db.Column(db.DateTime, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    user = db.relationship('User', backref='events')  # This sets up the relationship
```

ANNEXURE:

```
 def __repr__(self):
   return f'<Event {self.title} - {self.date}>'

def new_event (title, description, date, creator_id):
  """Function to save a new event with user association."""
  try:
    new_event = Event(title=title, description=description, date=date, creator_id=creator_id)
    db.session.add(new_event)
    db.session.commit() # Ensure this is successfully reached
    print(f"Event saved: {new_event}") # Debugging print statement
  except Exception as e:
    db.session.rollback() # Rollback in case of error
    print(f"Error saving event: {e}") # Log the error
    raise

class Message(db.Model):
  __tablename__ = 'messages'

  id = db.Column(db.Integer, primary_key=True)
  sender_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False) # ID of the user
who sent the message
  content = db.Column(db.Text, nullable=False) # Message content
  date = db.Column(db.DateTime, nullable=False) # Date and time when the message was sent
  sender = db.relationship('User', foreign_keys=[sender_id], backref='sent_messages')
  is_read = db.Column(db.Boolean, default=False)

  def __repr__(self):
    return f"<Message {self.id} - From: {self.sender_id} To: {self.recipient_id}>"
```