

Software Design Specification

CAMS (Campus Activity Management System)

Name & Reg No of Group Members

- | | |
|---------------------------|--------------------|
| 1. Gabriel M S | NIE22CS023 |
| 2. Joseph Mohan | NIE22CS029 |
| 3. Akshaykumar V S | LNIE22CS054 |

Table of Contents:

1.Abstract.....	3
2.Proposed tool & brief description about their selection.....	3
3.Introduction about the proposed project.....	5
4.System Overview.....	6
5.Design Considerations.....	8
a) Assumptions & Dependencies.....	8
b) General constraints.....	9
6.Development Methods.....	11
7.System Architecture.....	12
a) High level design.....	12
b) Low level design.....	16
8.Detailed system design.....	29
9.Interfaces [UI].....	32
10.User Roles.....	41
11.Glossary.....	41

1. Abstract of the Project

The Campus Activity Management System (CAMS) is an online platform designed to streamline the management and participation of various activities within a campus, including sports, arts, and other events. The system facilitates the smooth operation of campus activities by involving three distinct user roles: Admin, Faculty, and Student. The Admin is responsible for managing the overall system, including adding or removing faculty members, restricting or removing students, and overseeing all activities. Admins can also create, modify, or delete activities and access detailed student information. Faculty members can add, modify, or remove activities, monitor student participation, and issue participation certificates upon verification. Students can view and apply for activities, cancel participation if needed, download their participation certificates, and track activity points and semester-wise reports. The system aims to enhance the management efficiency and participation experience for all campus stakeholders by providing a user-friendly and organized platform for campus activities.

2. Proposed Tools and brief description about their selection

Here's a list of proposed tools and technologies for your Campus Activity Management System project, based on the Python Django framework and MySQL as the database:

2.1 Backend Framework:

- **Django (Python):** A high-level Python web framework that promotes rapid development and clean, pragmatic design. Django will be used for building the backend of the application, handling user authentication, session management, routing, and overall application logic.

2.2 Database:

- **MySQL:** An open-source relational database management system. MySQL will store all the data related to activities, users, participation, certificates, and reports. It offers reliability, scalability, and flexibility for the project.

2.3 Frontend Technologies:

- **HTML5/CSS3:** For structuring and styling the web pages. HTML5 is the standard for building modern web pages, while CSS3 helps create a responsive and visually appealing design.

- **JavaScript:** For implementing dynamic features like real-time interaction, form validation, and user interface improvements.

2.4 Templating Engine:

- **Django Templates:** A built-in templating engine provided by Django for rendering HTML pages dynamically, combining data from the backend with the front-end structure.

2.5 User Authentication & Authorization:

- **Django Authentication System:** Django comes with a robust built-in authentication system to manage user login, registration, permissions, and access control for Admin, Faculty, and Students.

2.6 File Storage and Management:

- **Django File Storage:** For handling the upload and download of participation certificates, activity reports, and other documents within the system.

2.7 Data Visualization and Reports:

- **Chart.js:** JavaScript libraries that can be used to generate visualizations, such as graphs and charts, for displaying activity participation points and semester-wise reports for students.

2.8 Version Control:

- **Git:** For source code management, tracking changes, and collaboration among team members during the development process.
- **GitHub:** Online Git repository hosting services for code collaboration and version control.

2.9 Deployment:

- **Local Machine:** Initially the project is hosted in the local machine.

2.10 Development Tools:

VS Code: Integrated Development Environments (IDEs) that help with code editing, debugging, and managing the Django project.

2.11 Testing:

- **Django Test Framework:** For unit testing backend code, views, and models to ensure functionality and prevent issues during development.

3.Introduction about the proposed project

The **Campus Activity Management System (CAMS)** is an innovative web-based platform designed to streamline the management, organization, and participation of various activities within a campus. The system aims to provide a centralized, easy-to-use interface for both administrators and users (students and faculty) to efficiently manage, track, and engage in a wide range of campus events, including sports, arts, and other extracurricular activities.

Traditionally, managing campus activities has been a complex and time-consuming task involving manual tracking, paperwork, and limited visibility. With CAMS, the entire process is digitized and simplified, allowing real-time updates, easy access, and seamless communication between all users. The system is built using the **Django framework** for the backend, coupled with a **MySQL database** for robust data storage and management.

The platform accommodates three primary user roles: **Admin**, **Faculty**, and **Student**.

- **Admins** have full control over the system, including the ability to create, modify, or delete activities, manage user access, and oversee all student participation.
- **Faculty members** can add and manage activities, view and verify student participation, and generate participation certificates.
- **Students** can explore available activities, apply for participation, track their progress, and download certificates.

Through this project, we aim to enhance student engagement, streamline event management, and improve the overall campus experience by reducing administrative workload, increasing visibility of activities, and enabling a more interactive and organized approach to campus events.

By implementing this system, the campus will benefit from a more efficient, paperless, and transparent process for managing activities, while also offering an enhanced experience for students and faculty alike.

4. System Overview

The **Campus Activity Management System (CAMS)** is an online application designed to automate the process of managing and organizing campus activities such as sports, arts, and other extracurricular events. The system provides a streamlined, user-friendly platform for administrators, faculty members, and students to interact with each other and efficiently manage various activities within the campus environment.

The system is divided into three distinct user roles, each with its own set of privileges and functionalities:

1. **Admin:** The Admin is the highest authority within the system. Admin users have complete control over the platform and can manage faculty and student users, create or modify activities, and monitor overall system usage. Key functionalities available to the Admin include:
 - User management (adding, removing, and restricting faculty and students)
 - Activity management (creating, modifying, or deleting activities)
 - Viewing and managing detailed student participation data
 - Generating reports and overseeing overall system performance.
2. **Faculty:** Faculty users have the responsibility of managing specific activities. They can create, modify, or remove activities, as well as verify student participation. Faculty members also have access to detailed reports on students who have applied for activities and can issue participation certificates. Key features available to faculty include:
 - Adding and modifying activities related to their domain
 - Viewing activity details and monitoring student applications
 - Verifying student participation and issuing certificates upon successful completion of activities
 - Managing and updating records for each activity they are responsible for.
3. **Student:** Students are the primary participants in campus activities, and they can use the system to view available events, apply for participation, and track their performance in different activities. The system offers students the following features:
 - Viewing and browsing through available campus activities
 - Applying for participation in activities and cancelling their participation if necessary
 - Downloading participation certificates once they complete an activity
 - Viewing their activity points and semester-wise performance reports, which help track their extracurricular engagement and achievements.

1. System Flow:

- **User Registration & Authentication:** The system supports secure user registration and authentication using a login mechanism. Students, faculty, and admins all have individual access based on their roles and credentials.
- **Activity Management:** Admins can create, modify, and delete campus activities. Faculty members can add and modify activities as well, specifically those related to their subject area or domain.
- **Participation Management:** Students can browse and apply for available activities. Upon completing an activity, students can download participation certificates.
- **Reports and Analytics:** Admins and faculty can generate reports and track student participation and performance over time, providing valuable insights into student engagement.

2. Technical Architecture:

The backend of the system is powered by the **Django web framework**, ensuring efficient and secure handling of user requests, authentication, and data management. **MySQL** serves as the database to store data related to users, activities, and participation, offering a reliable and scalable solution for managing large datasets. The frontend of the system is developed using HTML, CSS, and JavaScript for a responsive and interactive user interface. The system is fully responsive and optimized for both desktop and mobile access, allowing users to manage their activities from any device.

3. Key Features:

- **Role-based Access Control:** Different functionalities are provided based on user roles (Admin, Faculty, and Student).
- **Seamless Activity Management:** Easy creation, management, and modification of campus activities for faculty and admins.
- **Real-time Participation Tracking:** Students can instantly apply for activities, track their progress, and access certificates.
- **Interactive Reports:** Generation of semester-wise activity reports and student performance analytics.

4. Benefits of the System:

- **Increased Efficiency:** Streamlining the management and tracking of campus activities reduces administrative overhead and increases operational efficiency.
- **Enhanced User Experience:** Simplified user interface for all stakeholders ensures an intuitive and smooth interaction with the system.

- **Real-time Updates:** Instant access to activity details and participation records for both students and faculty.
- **Data Transparency:** Provides students with detailed information about their participation and performance, while also offering faculty and admins visibility into student engagement.

In summary, **CAMS** serves as an essential tool for modernizing campus activity management, offering a comprehensive solution to manage, track, and report on extracurricular participation with ease and efficiency.

5.Design Considerations

a. Assumptions and Dependencies:

In the design and development of the **Campus Activity Management System (CAMS)**, several assumptions and dependencies have been made regarding the software environment, user interactions, and related technologies. These assumptions are critical to the successful operation of the system.

1. Related Software or Hardware:

- **Django Framework:** The system is built on the **Django** framework (Python-based), which provides an efficient and secure environment for backend development.
- **MySQL Database:** The system depends on **MySQL** as the primary database for data storage. It assumes that MySQL is available and correctly configured on the user's machine or server.
- **Web Browser:** The platform is a web-based application, requiring users to have access to modern web browsers (Google Chrome, Mozilla Firefox, Safari, etc.) to access the system.
- **File Storage:** The system assumes that adequate storage is available to handle the upload and download of activity-related documents such as certificates and reports. This may involve local storage for development or cloud storage solutions for larger-scale deployments.
- **Email Service:** For sending notifications to users (students and faculty), the system assumes integration with an **email service provider** for sending email confirmations, reminders, and certificates.

2. Operating Systems:

- **Development Environment:** The system is designed to run on any operating system that supports Python and Django. This includes:
 - **Windows, macOS, and Linux** for local development.
- **MySQL:** The database is compatible with all major operating systems and will be assumed to run on **Linux** or **Windows** servers for deployment.

3. Network Connectivity:

- The system requires **Internet access** for all users to interact with the web-based platform, including uploading/downloads of files (e.g., certificates) and viewing activities.

b. General Constraints:

Several constraints have been identified that will influence the design and implementation of the **CAMS** software. These constraints may stem from limitations imposed by hardware, software, or user environments.

1. Hardware or Software Environment:

- **Server Resources:** The system's performance and scalability may be limited by the available hardware resources on the server hosting the application. For larger institutions, adequate server specifications (RAM, CPU, and storage) should be ensured to handle the increased load of concurrent users and activity data.
- **Database Size:** As the number of users and activities grows, the database size can become quite large, which could impact performance if not properly optimized. The system must be designed with scalable database structures to handle large datasets.
- **Internet Connectivity:** As a web-based system, CAMS requires reliable **internet access**. Users with slow or intermittent internet connectivity may experience slower response times or difficulties in interacting with the system.
- **Cross-Browser Compatibility:** The system must be compatible with all major web browsers. Some features may appear differently on different browsers, and the user interface (UI) design needs to ensure cross-browser consistency for an optimal user experience.

2. End-User Environment:

- **End-User Device Compatibility:** The system is designed to be accessed via web browsers, but different users may use a variety of devices (e.g., desktop computers, laptops, tablets, and smartphones). The system should be responsive and support

mobile devices, ensuring that students, faculty, and admins can interact with the platform seamlessly, regardless of device type.

- **User Authentication:** The system relies on secure **user authentication** for all roles (admin, faculty, and student). While secure authentication mechanisms (such as Django's built-in user authentication) are implemented, the design assumes that users will have basic technical skills to manage passwords and login credentials. The system should also include features like password recovery and email verification.
- **User Access Control:** Access to various features of the system is determined by the user's role (Admin, Faculty, and Student). While roles and permissions are clearly defined, users with insufficient privileges will be restricted from accessing specific data or features. This assumes that users will follow the intended access structure, with no overlapping or unauthorized access attempts.

3. System Maintenance and Updates:

- **Ongoing Maintenance:** The system assumes that there will be periodic updates and maintenance to ensure that bugs are fixed, new features are added, and security vulnerabilities are addressed. This requires ongoing server support and the ability to roll out updates smoothly.
- **Backup and Data Recovery:** Regular backups of the MySQL database are essential for preventing data loss in case of system failure. The system assumes that data backup processes will be implemented at regular intervals to ensure the integrity of user and activity data.

4. Security Constraints:

- **Data Privacy and Protection:** Since personal information of users (students, faculty, and admins) will be stored. Secure data handling, encryption of sensitive data, and user consent must be considered during the design phase.
- **User Authentication & Authorization:** The system relies on robust authentication mechanisms to ensure that users are who they claim to be and that they only access the parts of the system they are authorized to. This implies that the platform will need secure methods for storing passwords, enforcing strong password policies, and preventing unauthorized access.

5. Time Constraints:

- The system is expected to be delivered within a specified time frame, and this timeline may impact the scope of features included in the initial release. The design needs to prioritize essential functionalities first, with additional features and optimizations added in later phases.

6. Development Methods

1. Agile Development Methodology (Scrum Framework)

The Agile approach was chosen because it aligns well with the dynamic nature of the project, where requirements may evolve over time. Agile allows for iterative development and continuous feedback, enabling the development team to adapt the system based on user feedback, ensuring that the final product is more likely to meet user expectations.

Key Features of Agile Adopted:

- **Incremental Development:** Features were developed and delivered in small, manageable increments (sprints), allowing the team to focus on key functionalities at each stage.
- **Collaboration and Feedback:** Regular collaboration with stakeholders (faculty, students, and admins) was encouraged to gather feedback early and often, ensuring the system design meets their needs.
- **Continuous Improvement:** At the end of each sprint, the team conducted retrospective meetings to discuss what went well and areas of improvement.
- **User Stories and Prioritization:** Features were defined as **user stories** (functional requirements) and prioritized based on their importance to users. This ensured that the most critical features were developed first.

Why Agile/Scrum Was Chosen:

- **Flexibility:** Since campus activities and requirements might change depending on events, academic schedules, and user needs, the agile approach offers flexibility to handle changing requirements.
- **Stakeholder Involvement:** Agile allows for constant feedback from the users (students, faculty, and admins), leading to a better understanding of their needs and higher user satisfaction.
- **Rapid Prototyping:** With Agile, we could deliver working prototypes at the end of each sprint and fine-tune the system based on actual user input and real-world testing.

2. Model-View-Controller (MVC) Design Pattern

The **MVC** design pattern was used to structure the system for ease of maintenance, scalability, and separation of concerns. This pattern is commonly used in web development frameworks like **Django** and is highly effective in structuring large applications like CAMS.

- **Model:** Represents the data layer (MySQL database) and encapsulates the core business logic. It defines how data is stored, validated, and manipulated.
- **View:** Responsible for the user interface and presentation logic. Views are typically HTML templates in Django that present data to users in a structured format.
- **Controller (Views in Django):** Acts as an intermediary between the Model and the View. The controller handles user requests, interacts with the model (database), and renders the appropriate view (UI) for the user.

Why MVC Was Chosen:

- **Separation of Concerns:** The MVC pattern allows developers to focus on distinct components of the system without worrying about the entire application at once, leading to cleaner and more maintainable code.
- **Scalability and Maintenance:** As the project grows (with more users, activities, etc.), the MVC structure makes it easier to manage and extend the system with minimal changes to other parts of the codebase.
- **Django Framework Support:** Django inherently follows the MVC design pattern (known as **MTV** - Model, Template, View), which aligns perfectly with the development environment

7. System Architecture

The **Campus Activity Management System (CAMS)** was designed with a modular architecture that divides the overall system functionality into smaller, manageable components or subsystems. Each subsystem is assigned specific responsibilities to handle particular tasks within the system. This modular approach ensures that the system is scalable, maintainable, and adaptable as new features are added or existing ones are modified.

At a high level, the system architecture consists of three main layers: **Presentation Layer (Frontend)**, **Application Layer (Backend)**, and **Data Layer (Database)**. These layers collaborate to provide the desired functionality, ensuring smooth interaction between users (admins, faculty, and students) and the system.

A) High-Level System Responsibilities and Roles

The main responsibilities of the **Campus Activity Management System** are:

1. **User Authentication and Authorization:** Managing user roles (Admin, Faculty, and Student) and ensuring that only authorized individuals can access specific system features.
2. **Activity Management:** Allowing administrators and faculty to create, update, and manage activities, while ensuring students can browse and apply for activities.
3. **Participation Tracking:** Tracking student participation in activities, including status updates and verification by faculty.
4. **Certificate Generation:** Enabling faculty to generate participation certificates for students who successfully complete activities.
5. **Reporting and Analytics:** Providing admins and faculty with detailed reports of student participation, activity points, and semester-wise performance.

- **Subsystems and Components**

The system is divided into several subsystems, each responsible for a specific set of tasks:

1. User Management Subsystem

- **Responsibilities:**
 - Handle user registration, login, and authentication for Admins, Faculty, and Students.
 - Manage user roles and permissions, ensuring that different types of users have access to only relevant functionalities.
 - Implement features for user profile management (viewing and editing personal details).
- **Components:**
 - **Authentication Service:** Manages user login and session management.
 - **Role Management:** Determines the access level and permissions for each user type (admin, faculty, and student).

2. Activity Management Subsystem

- **Responsibilities:**
 - Admins and Faculty can create, modify, and delete activities.
 - Students can browse, view, and apply for participation in activities.
 - Activity status tracking, including approval, cancellation, and status updates by faculty members.
- **Components:**

- **Activity Creation & Modification:** Provides interfaces for faculty and admins to add or edit activity details.
- **Activity View & Search:** Allows students to search for activities by type, date, or category.
- **Activity Participation:** Allows students to apply for activities and cancel participation if needed.

3. Participation Management Subsystem

- **Responsibilities:**
 - Track students' participation in different activities, including registration, status updates, and verification by faculty.
 - Generate and manage participation status (pending, confirmed, cancelled).
- **Components:**
 - **Participation Tracking:** Records each student's participation and status.
 - **Faculty Verification:** Allows faculty members to verify student participation and update status accordingly.

4. Certificate Generation Subsystem

- **Responsibilities:**
 - Generate certificates for students who successfully complete activities.
 - Allow students to download their participation certificates in PDF format.
- **Components:**
 - **Certificate Generator:** Automatically generates a certificate based on activity completion.
 - **Download Management:** Provides students with access to download their certificates.

5. Reporting and Analytics Subsystem

- **Responsibilities:**
 - Provide detailed reports of student participation, semester-wise activity points, and performance metrics.
 - Admins and faculty can generate customized reports based on various filters such as activity type, student name, or semester.
- **Components:**
 - **Activity Reports:** Displays participation statistics for each activity.
 - **Semester Reports:** Generates detailed reports on student performance and engagement in activities throughout the semester.

- **Activity Points:** Tracks and displays the total points accumulated by students from various activities.

6. Notification and Communication Subsystem

- **Responsibilities:**
 - Send email notifications to users (students, faculty, and admins) regarding activity updates, participation status, and certificates.
- **Components:**
 - **Email Notification Service:** Automatically sends email notifications for activity updates, application status, and certificates.
 - **Alert System:** Provides alerts within the system for critical actions, such as activity approval or participation cancellations.

7. Data Management Subsystem

- **Responsibilities:**
 - Store all system data, including user information, activity details, student participation, and reports.
- **Components:**
 - **Database (MySQL):** Stores data in structured tables, ensuring efficient data retrieval and management.
 - **Data Backup & Recovery:** Regular backups and data recovery processes to prevent data loss.

Component Interaction and Collaboration

The components of the **Campus Activity Management System** work together through clear interfaces and API endpoints, ensuring smooth communication between the different subsystems. Here's a brief overview of how they collaborate:

1. **User Authentication** (User Management Subsystem) interacts with the **Activity Management** subsystem to ensure that only authorized users (admins, faculty, students) can create, manage, or participate in activities based on their roles.
2. **Activity Management** and **Participation Management** work together to allow students to apply for activities. Faculty can verify student participation and update the status of each application.
3. The **Certificate Generation Subsystem** is triggered by the **Participation Management** subsystem when a student's participation is confirmed. Once verified, the system generates a certificate and stores it for download.

4. **Reporting and Analytics** subsystem collects data from both the **Activity Management** and **Participation Management** subsystems to generate reports and analyse student engagement across activities.
5. **Email Notification** services interact with all subsystems, sending emails to students, faculty, and admins about important updates such as activity approvals, cancellations, or certificate generation.
6. The **Data Management Subsystem** ensures that all data generated across the system is stored securely in the **MySQL** database, providing easy access to users and maintaining data integrity.

Collaboration and Flow

The following is a brief outline of the system's flow:

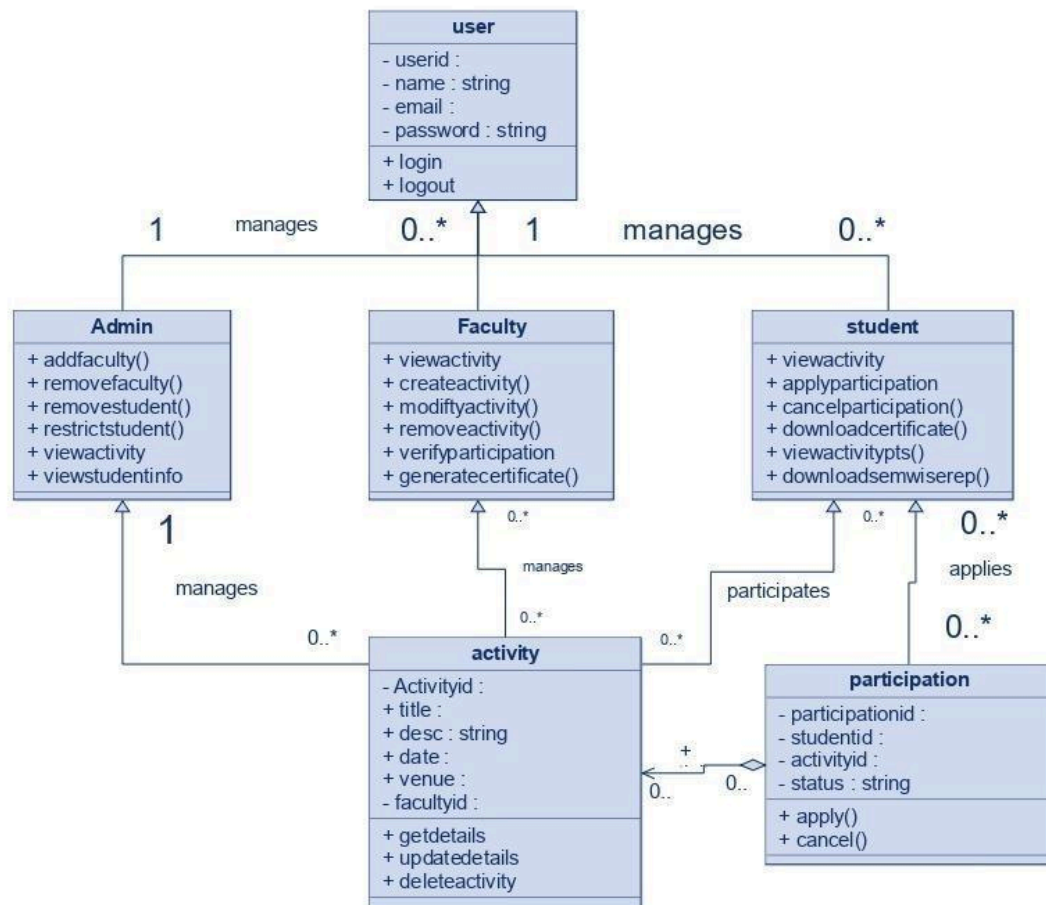
1. **Admin/Faculty Action:** Admin or faculty members log in to the system, and based on their role, they can create or modify activities in the **Activity Management** subsystem.
2. **Student Interaction:** Students log in and view available activities. They apply for participation, which is tracked in the **Participation Management** subsystem.
3. **Verification:** Faculty members verify student participation, after which a participation certificate is generated by the **Certificate Generation** subsystem.
4. **Report Generation:** Admins and faculty can generate reports using the **Reporting and Analytics** subsystem based on participation data stored in the **Data Management** subsystem.

B) Low Level Design:

System Models and Diagrams

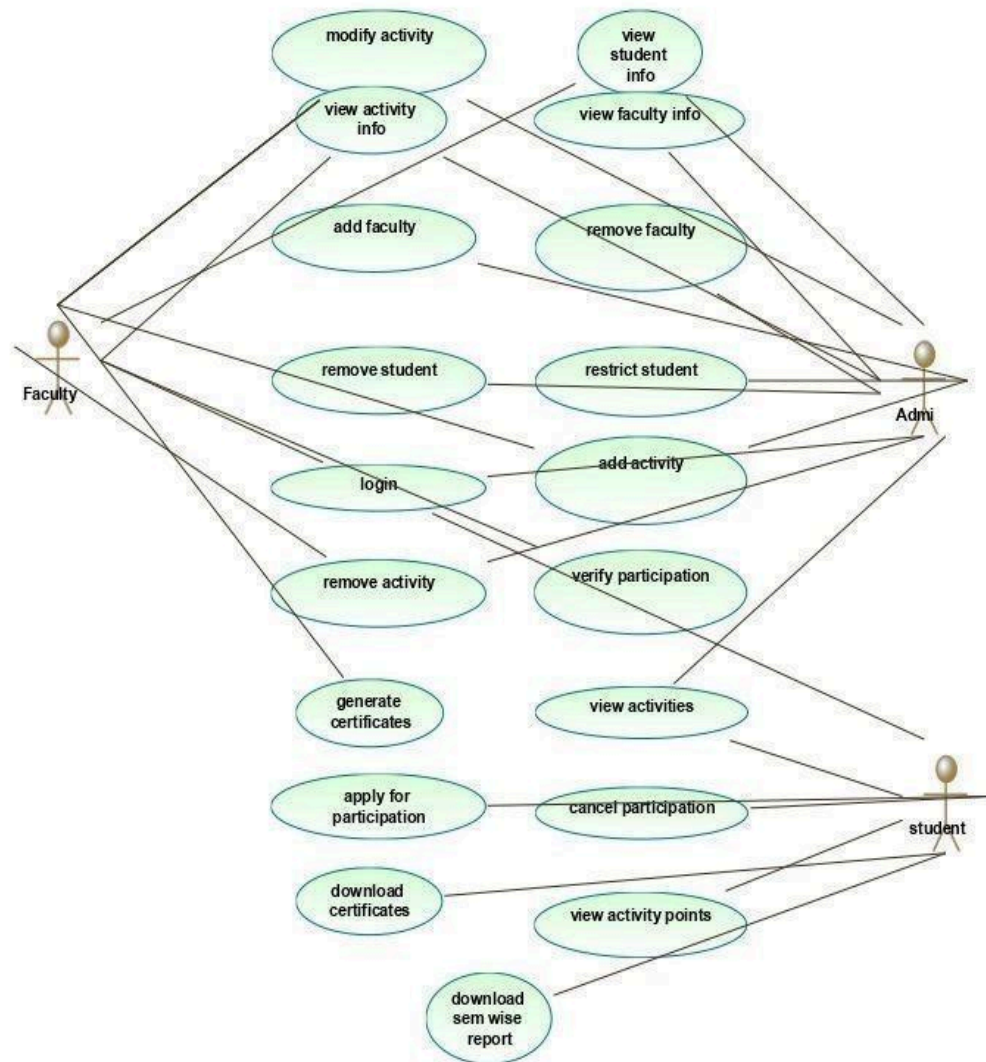
UML Diagrams: As per the earlier reference, **UML diagrams** (including use cases, class diagrams, and sequence diagrams) have been used to define the interactions and structure of the system. These diagrams clearly depict how the system is divided into subsystems and how the components interact with each other.

Class Diagram:



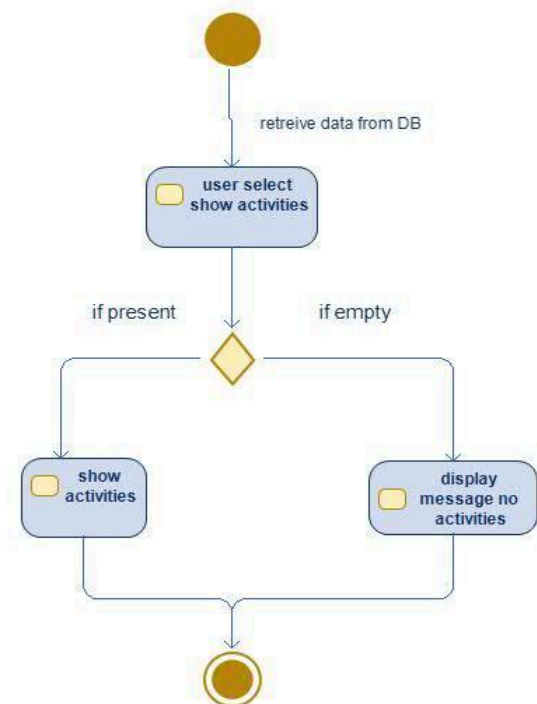
Use Case Diagram:

USE CASE DIAGRAM-CAMS

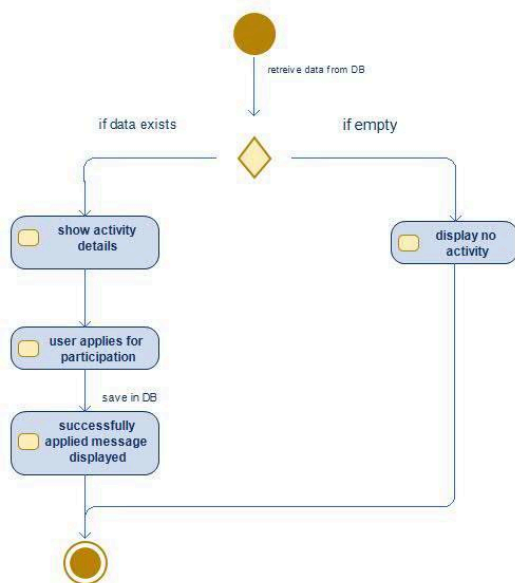


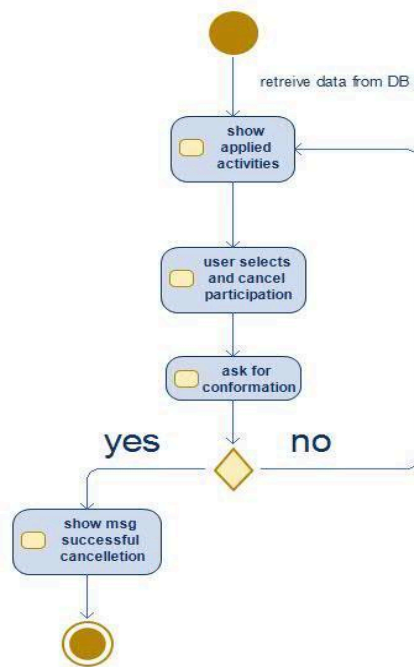
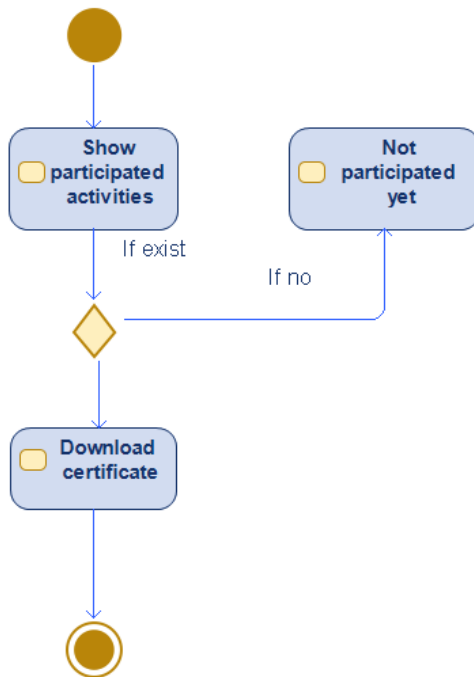
Student Activity Diagrams:

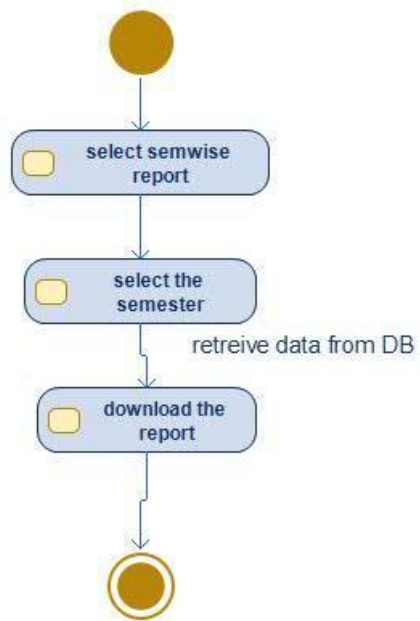
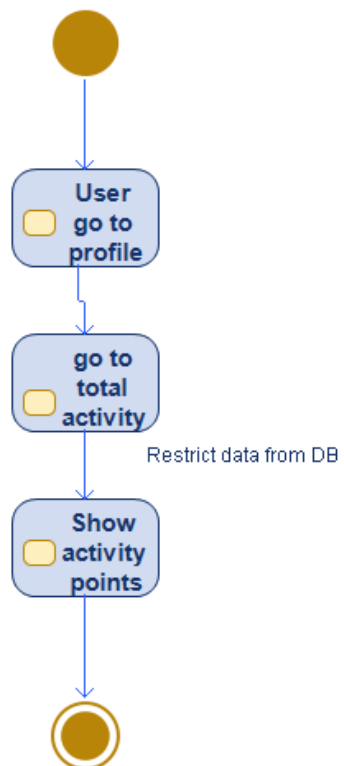
View Activity info



Apply for participation

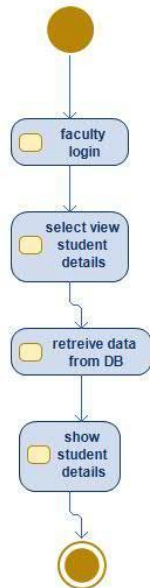


Cancel participation:**Download certificate:**

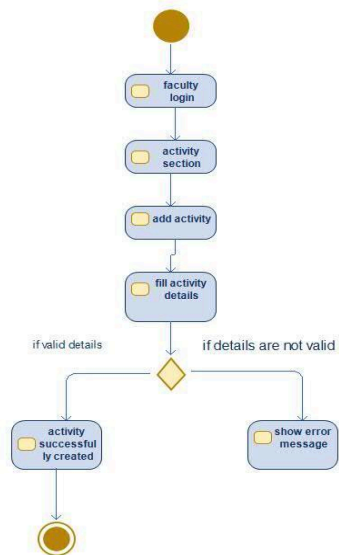
Download Sem-wise report:**View activity points:**

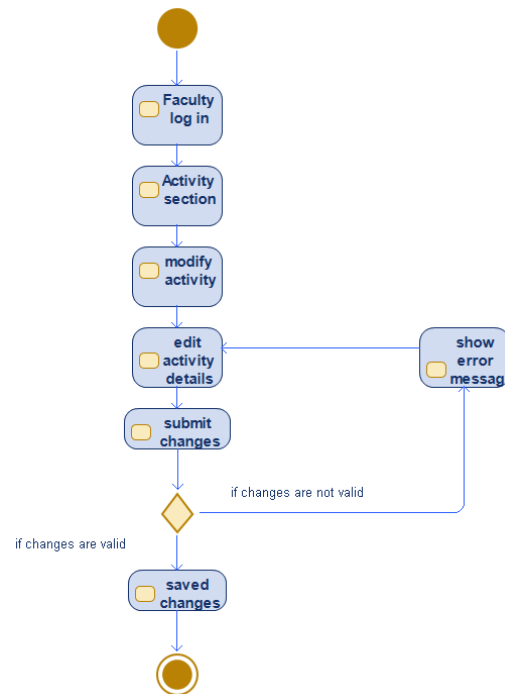
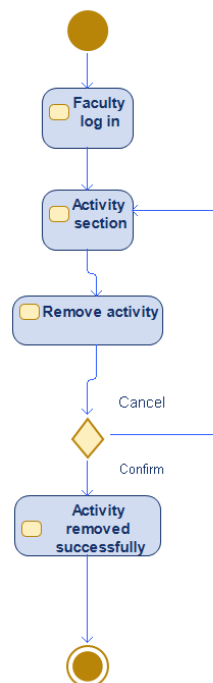
Faculty Activity Diagrams:

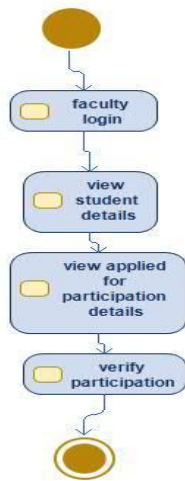
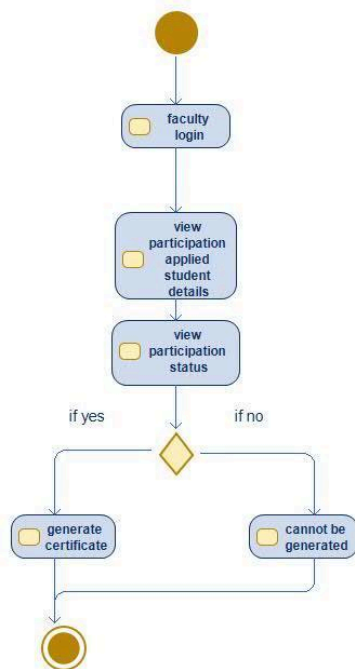
View student details:

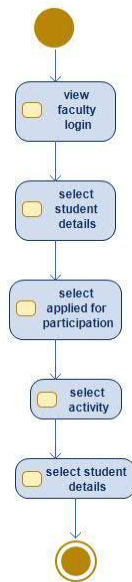
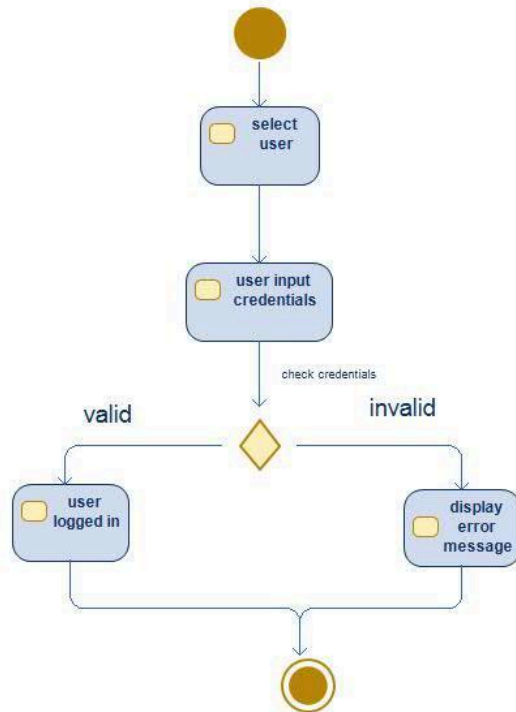


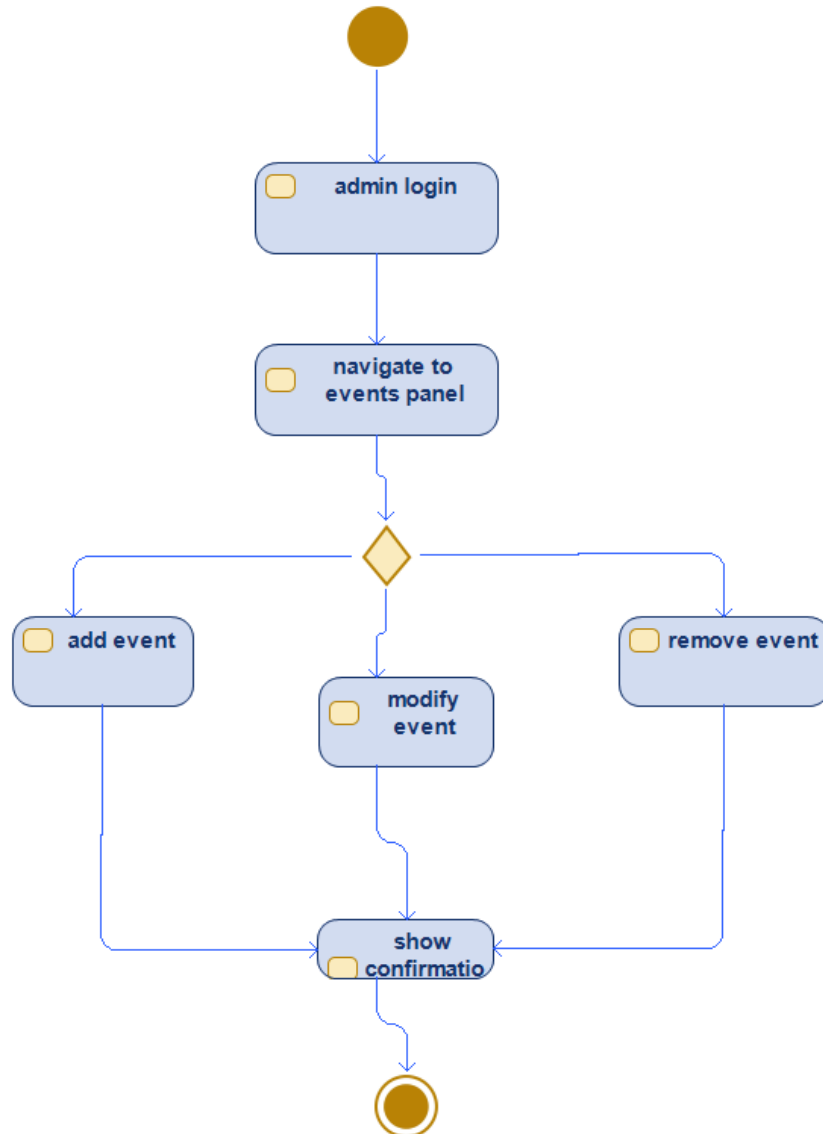
Add Activity:

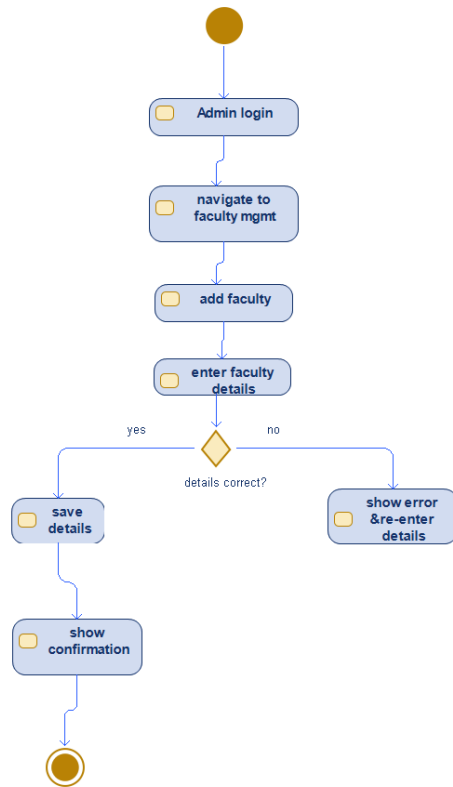
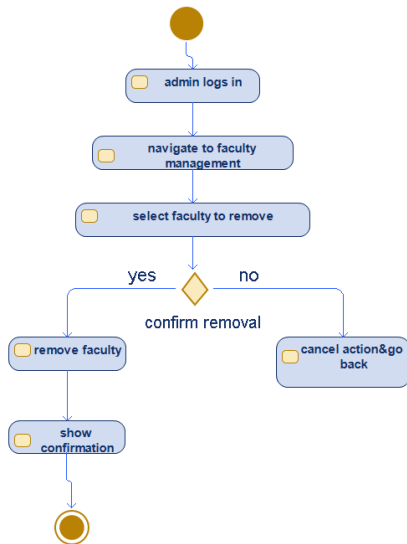


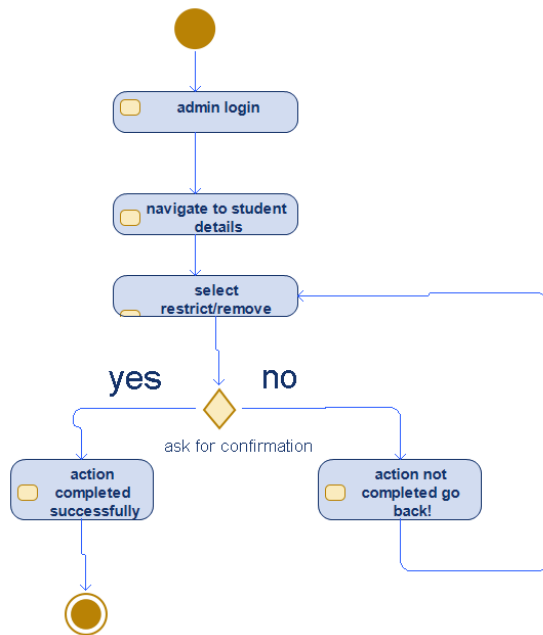
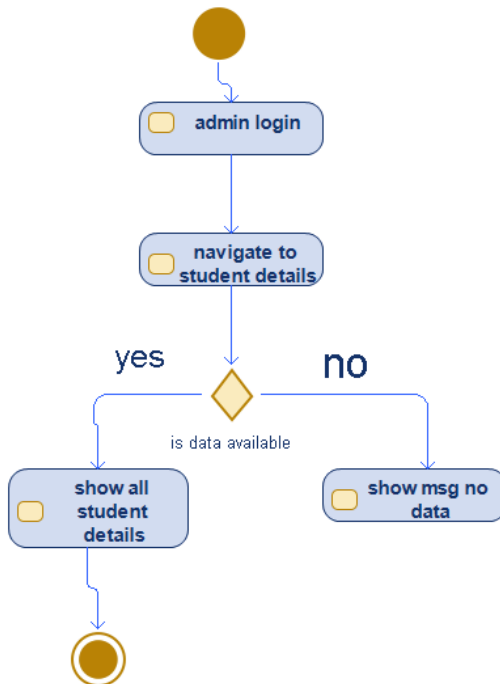
Modify Activity:**Remove Activity:**

Verify participation:**Generate certificates:**

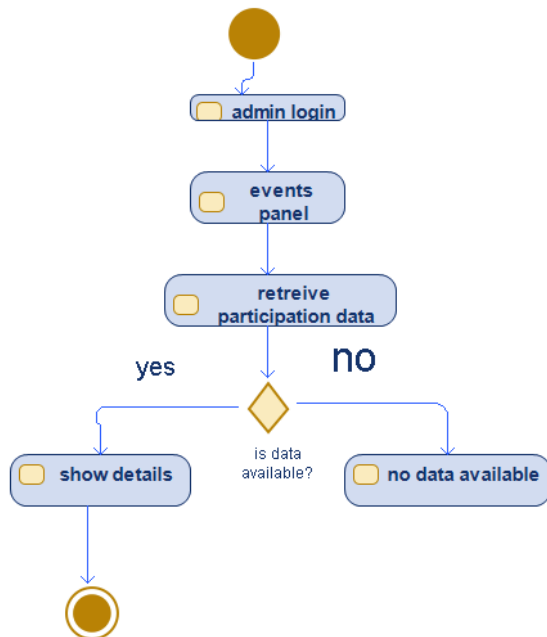
View info of participation applied students:**Common login page for all users:**

Admin Activity Diagrams:**Create/ Modify/ Remove event:**

Add faculty:**Remove Faculty:**

Restricts Student:**View student details:**

View participation details:



8. Detailed System Design

System Architecture

The system follows an MVC (Model-View-Controller) architecture using Django framework. The key components include:

High-Level Architecture

Frontend: Django templates (HTML, CSS, JavaScript, and bootstrap)

Backend: Django framework (Python)

Database: MySQL

Authentication & Authorization: Django Authentication System

Hosting: Deployed on a cloud server or on-premise setup

System Components

1. User Management System

- Role-based authentication (Admin, Faculty, Student)
- Registration & Login system (Email-based verification)
- Profile management

2. Activity Management

- CRUD operations for activities (Admin, Faculty)
- Activity categories (Sports, Arts, Cultural, etc.)
- Participation management

3. Participation Tracking

- Students can apply for activities
- Faculty approval mechanism
- Activity completion & certificate generation

4. Reporting & Dashboard

- Student activity tracking (semester-wise reports)
- Admin & Faculty dashboard for insights
- Exporting reports (PDF/Excel)

Database design

User table:

Field	Type	Description
User_id	int	Unique User ID
Name	Varchar(255)	Full Name
Email	Varchar(255)	User email(unique)
Password	Varchar(255)	Hashed password

Role	Char(255)	User role
Created at	timestamp	Account creation time

Activity table:

Field	Type	Description
Activity id	int	Unique activity id
Title	Varchar(255)	Activity name
Description	text	Activity details
Category	Varchar(100)	Sports, arts, culture, others...
Created by	int	Faculty/Admin
Start date	datetime	start time
End date	datetime	end time

Participation table:

Field	Type	Description
Participation_id	int	Unique participation id
Student_id	int	Unique student id
Activity id	int	Unique activity id
status	Boolean	Applied/approved/completed
Certificate link	varchar	Path to certificate
points	int	Activity points awarded

System Workflow:

User Authentication:

1. Users register with email verification.
2. Role-based access control restricts unauthorized actions.
3. Logged-in users access dashboards based on their roles.

Activity Management:

1. Admin/Faculty create activities.
2. Students browse and apply for activities.
3. Faculty approve/reject applications.
4. Upon completion, faculty verify participation and issue certificates.

Participation & Reporting:

1. Students track participation history and points.
2. Faculty monitor student engagement.
3. Admins generate semester-wise reports.
4. Students can download participation certificates.

Security Measures:

- Role-based access control to prevent unauthorized modifications.
- Data encryption for sensitive information.
- Input validation & SQL injection prevention.

Deployment Strategy

- Version Control: Git/GitHub
- Database Hosting: MySQL
- CI/CD: GitHub Actions for automated deployment

9.Interfaces [UI]

User Interface Screens of the parent

Login pages:

CAMPUS ACTIVITY

Student
Sign In

Welcome back

User name

Password

[Forgotten password ?](#)

Sign In

HELLO

Click here for
Faculty and Admin Sign In

Sign In

CAMPUS ACTIVITY

Faculty / Admin
Sign In

Welcome back

User name

Password

[Forgotten password ?](#)

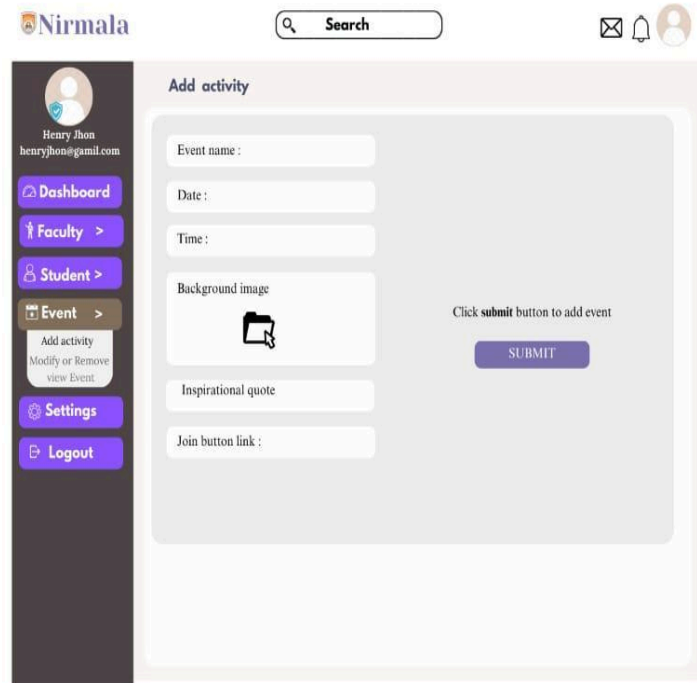
Sign In

HELLO

Click here for
Student Sign In

Sign In

Admin pages:



The screenshot displays the Nirmala Admin Dashboard. At the top left is the Nirmala logo. To its right is a search bar with a magnifying glass icon and the word "Search". Further right are icons for an envelope, a bell, and a user profile. On the left side, there is a dark sidebar with a user profile section for Henry Jhon (henryjhon@gmail.com) and a list of navigation buttons: Dashboard, Faculty >, Student >, Event >, Add activity, Modify or Remove view Event, Settings, and Logout. The main content area is titled "Add activity" and contains several input fields: "Event name :", "Date :", "Time :", "Background image" (with a file upload icon), "Inspirational quote", and "Join button link :". A "SUBMIT" button is located to the right of the "Background image" field. A text instruction "Click submit button to add event" is positioned above the "SUBMIT" button.

Nirmala

Search

Henry Jhon
henryjhon@gmail.com

Dashboard

Faculty >

Student >

Event >

Add activity
Modify or Remove
view Event

Settings

Logout

Add activity

Event name :

Date :

Time :


Background image




Click submit button to add event


INSPIRATIONAL QUOTE

Join button link :

SUBMIT







Henry Jhon
henryjhon@gmail.com

Dashboard

Faculty >


Student >

Event >

Settings

Logout

Setting



Name : Henry Jhon

Age : 46

Phone : 3850925478

E-mail : henryjhon@gmail.com

Address : ABC[house], Iceland

Sex : Male

Department : CSE


Experience : 7 years, Abd college




User name : Henryjhon1232


Password : fDg68#5\$h

Click **submit** button to save changes

SUBMIT







Henry Jhon
henryjhon@gmail.com

Dashboard

Faculty

Add or Remove


Student

Event

Settings

Logout

Add or Remove Faculty



Name :

Age :

Phone :

E-mail :

Address :

Sex :

Department :

Experience :

User name :


Password :

Click **submit** button to Add faculty




Submit


Click **Delete** button to Remove faculty

Delete



Search





Henry Jhon
henryjhon@gmail.com

Dashboard

Faculty >

Student >

Event >
Add activity
Modify or Remove
view Event

Settings


Logout

Modify / Remove Activity

Event name : SPORTS

Date : 01/05/2025

Time : 9.00 - 5.00

Background image



Inspirational quote: Never Ever Give Up

Click here to **SAVE** changes




SAVE


Click here to **DELETE** Event

DELETE



Search





Henry Jhon
henryjhon@gmail.com

Dashboard

Faculty >

Student >
View & Restrict
Remove

Event >

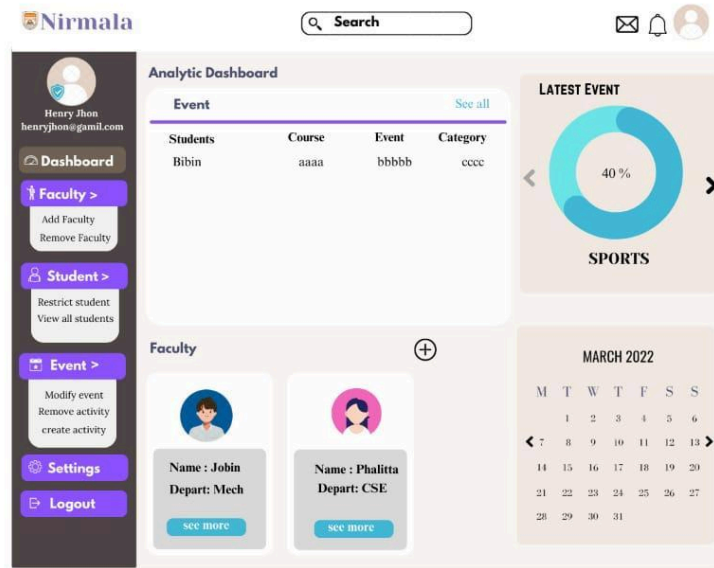
Settings

Logout

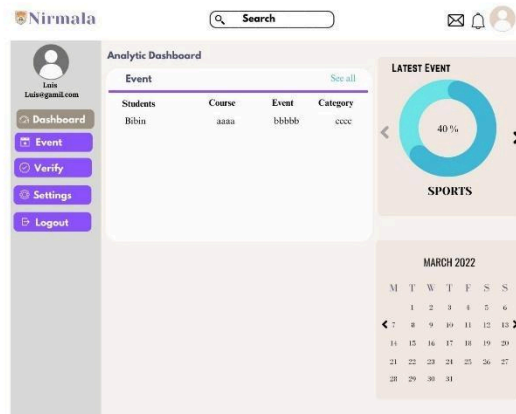
View students details

Search

Name	Department	Course	Id	Restrict	
Jain	computer	AI	888444	Restrict	See more



Faculty pages:



Luis
Luis@gamil.com

Dashboard

Event >

Modify or Remove
Add activity
view Activity

Verify

Settings

Logout

Add Activity

Event name :

Date :

Time :

Background image

Inspirational quote :

Click submit button to add event

SUBMIT

Luis
Luis@gamil.com

Dashboard

Event >

Modify event
Remove activity
Add activity
view Activity

Verify

Settings

Logout

Modify / Remove Activity

Event name : SPORTS

Date : 01/05/2025

Time : 9.00 - 5.00

Background image

Inspirational quote: Never Ever Give Up

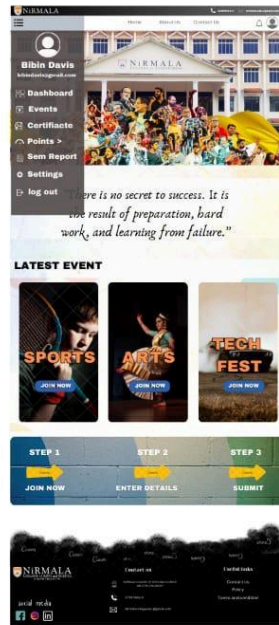
Click here to SAVE changes

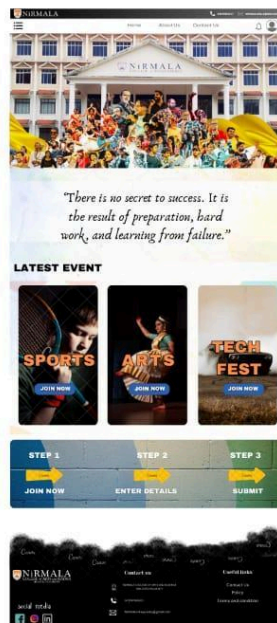
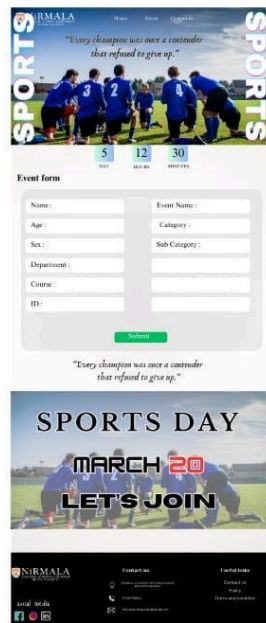
SAVE

Click here to DELETE Event

DELETE

Student pages:





10. User Roles

1. Admin

- Manages the entire system.
- Adds/removes faculty members.
- Restricts or removes students if necessary.
- Creates, modifies, or deletes activities.
- Accesses and manages all student information.
- Monitors overall participation and generates reports.

2. Faculty

- Creates, modifies, or deletes activities.
- Monitors student participation in activities.
- Approves or rejects student participation requests.
- Issues participation certificates after verification.
- Tracks student performance based on activity points.

3. Student

- Views and applies for campus activities.
- Cancels participation if needed.
- Downloads participation certificates.
- Tracks personal activity points and semester-wise reports.

11. Glossary

- **Admin** – The system's highest-level user who manages faculty, students, and activities.
- **Faculty** – A user role responsible for creating, managing, and verifying campus activities.
- **Student** – A user role that participates in activities and tracks progress.
- **Activity** – An event or program (e.g., sports, arts, workshops) that students can join.
- **Participation** – The process of a student enrolling in an activity.

- **Certificate** – A document issued to students upon successful participation in an activity.
- **Activity Points** – A scoring system to track student engagement in campus activities.
- **Semester Report** – A detailed record of a student's participation and earned points over a semester.
- **CRUD Operations** – Create, Read, Update, and Delete functions for managing activities and users.
- **Dashboard** – A visual interface displaying system data for Admin, Faculty, and Students.
- **Authentication** – The process of verifying user identity via login credentials.
- **Authorization** – The process of granting access to system features based on user roles.
- **MySQL** – The relational database used to store user, activity, and participation data.
- **Django** – The Python-based web framework used for backend development.
- **Frontend** – The user interface of the system, built with HTML, CSS, and JavaScript.
- **Backend** – The server-side logic and database interactions managed by Django.
- **Hosting** – The deployment of the system on a cloud or local server.
- **CI/CD (Continuous Integration/Continuous Deployment)** – A pipeline for automating testing and deployment of the system.