# **Campus Events Portal — Design Document**

## **1. Overview**

The **Campus Events Portal** is a lightweight web application designed to manage and display campus events.

* **Admins** can create and manage events.
* **Students** can view all available events, register for a chosen event, and receive a confirmation of successful registration.

The application uses a **Flask backend** with **SQLite** for data storage and a **HTML/CSS/JS frontend** for both admin and student portals.

## **2. Objectives**

1. Allow admins to create and manage events.
2. Display events dynamically in a student-friendly format.
3. Enable students to register for events easily.
4. Provide real-time feedback on successful registrations.
5. Maintain simple, responsive, and accessible UI design.

## **3. Architecture**

### **3.1 Backend**

* **Framework:** Flask
* **Database:** SQLite
* **Data Models:**
  + colleges: Stores college information.
  + students: Stores student information.
  + events: Stores event details.
  + registrations: Tracks which student is registered for which event.
  + attendance: Optional, tracks student attendance.
  + feedback: Optional, stores student feedback on events.
* **REST APIs:**
  + /api/events — Create and list events.
  + /api/events/available — Get events available for registration.
  + /api/events/<event\_id>/register — Register a student for an event.
  + /api/reports/\* — Generate reports on events and student participation.

### **3.2 Frontend**

* **HTML/CSS/JS** based portals:  
  + **Admin Portal:** Form-based event creation and table of all events.
  + **Student Portal:** Event cards with dynamic registration form.
* **Dynamic Features:**
  + Events fetched from backend and displayed immediately.
  + Event registration form pops up when a student clicks on an event.
  + Displays success/failure messages under the registration form.
* **Styling:**
  + Responsive design for mobile and desktop.
  + Card-style display for events.
  + Hover effects for better interactivity.

## 

## **4. Database Schema**

### **4.1 Colleges**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| college\_id | TEXT | Primary key |
| name | TEXT | College name |
| timezone | TEXT | Timezone of the college |

### **4.2 Students**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| student\_uuid | TEXT | Primary key |
| college\_id | TEXT | Foreign key to colleges |
| student\_local\_id | TEXT | Local student ID (unique per college) |
| name | TEXT | Student full name |
| email | TEXT | Email address |
| department | TEXT | Department |
| year | INTEGER | Year of study |

### **4.3 Events**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| event\_id | TEXT | Primary key |
| college\_id | TEXT | Foreign key to colleges |
| title | TEXT | Event title |
| description | TEXT | Event description |
| event\_type | TEXT | Workshop/Hackathon/Seminar/Fest |
| start\_ts | TEXT | ISO formatted start time |
| end\_ts | TEXT | ISO formatted end time |
| location | TEXT | Event location |
| capacity | INTEGER | Maximum participants |
| status | TEXT | Published/Draft |

### **4.4 Registrations**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| registration\_id | TEXT | Primary key |
| event\_id | TEXT | Foreign key to events |
| student\_uuid | TEXT | Foreign key to students |
| registered\_at | TEXT | ISO timestamp of registration |
| status | TEXT | Default 'registered' |

## **5. Workflow**

### **Admin Portal**

1. Admin opens the portal.
2. Fills in event details in a form.
3. Clicks **Create Event**.
4. Event is stored in the database and appears in admin event list.

### **Student Portal**

1. Student opens the portal.
2. All available events are displayed in card format.
3. Student clicks on an event card.
4. Registration form appears.
5. Student enters name and email.
6. Clicks **Register**.
7. Success message is displayed under the form.

## 

## 

## **6. Security and Validation**

* Mandatory fields checked before submission.
* Capacity limits enforced during registration.
* Duplicate registrations prevented using unique constraints in the database.
* Input validated for correct formats (email, ISO date, positive numbers).

## **7. Technology Stack**

* **Backend:** Python, Flask
* **Database:** SQLite
* **Frontend:** HTML, CSS, JavaScript
* **Libraries:** Standard Flask, no additional frontend frameworks

## **8. Future Enhancements**

* Add **student login system** to track registrations per student.
* Send **email notifications** for successful registration.
* Implement **attendance check-in** using QR codes.
* Add **event feedback system** for students.
* Enhance frontend using **React or Vue.js** for better interactivity.