Campus Event Manager - System Design Document

1. Introduction

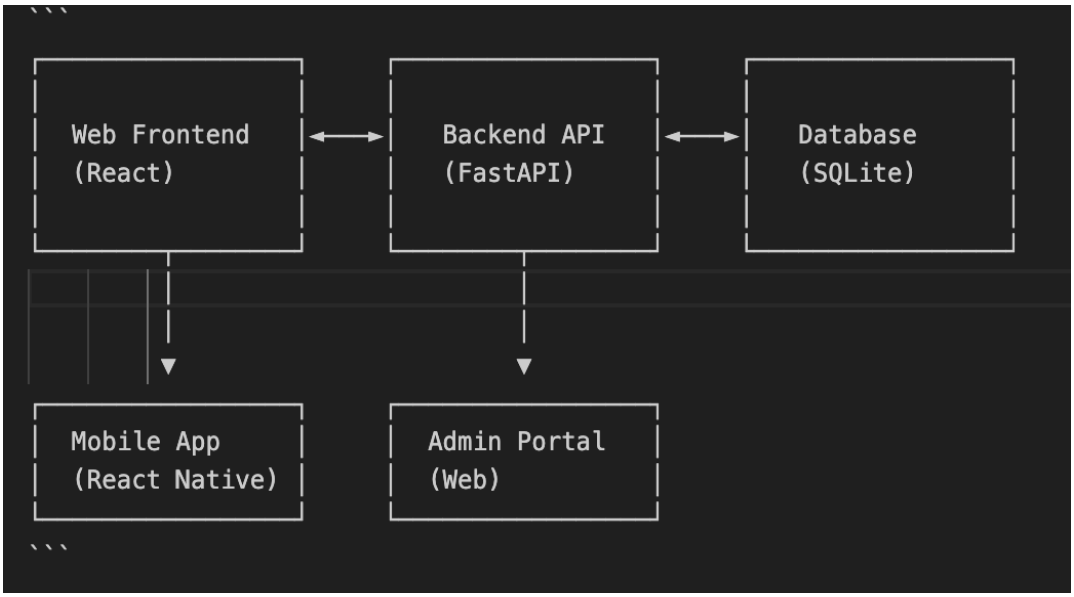
1.1 Purpose

This document outlines the system design for the Campus Event Manager, a comprehensive platform for managing college events, student registrations, attendance tracking, and analytics.

1.2 System Overview

The Campus Event Manager is a web-based application that enables event creation and management, student registration, QR code-based attendance tracking, real-time analytics, and feedback collection.

2. System Architecture



2.1 High-Level Architecture

The system follows a modular architecture with React-based frontend, FastAPI backend, and SQLite/PostgreSQL database.

2.2 Technology Stack

- Frontend: React.js with TypeScript - Mobile: React Native - Backend: FastAPI (Python) - Database: SQLite (Dev), PostgreSQL (Prod) - Authentication: JWT - Real-time: WebSockets - Caching: Redis - Containerization: Docker - CI/CD: GitHub Actions

3. Core Components

3.1 Authentication Service

Includes JWT-based authentication, role-based access control, session management, and password reset.

3.2 Event Management

Supports event CRUD, scheduling, recurring events, and capacity management.

3.3 Registration System

Handles student registration, waitlist, email notifications, and validation.

3.4 Attendance Tracking

Provides QR code attendance, real-time monitoring, reports, and offline mode.

3.5 Feedback System

Collects feedback, ratings, performs sentiment analysis, and generates reports.

4. API Design

Authentication

`/auth/register`, `/auth/login`, `/auth/refresh`, `/auth/password-reset`

Events

`/events`, `/events/{id}`, `/events/{id}/update`, `/events/{id}/delete`

Registrations

`/events/{id}/register`, `/events/{id}/registrations`, `/events/{id}/register/cancel`

Attendance

`/attendance/scan`, `/events/{id}/attendance`, `/attendance/manual`

5. Database Schema

Users Table

Stores user details with role-based access.

Events Table

Defines events, schedule, and venue information.

Registrations Table

Manages student registrations and waitlist.

Attendance Table

Tracks attendance using QR codes and manual entry.

6. Security Considerations

Authentication & Authorization

Uses JWT, bcrypt hashing, role-based access, and rate limiting.

Data Protection

Encrypted storage, HTTPS, input sanitization, SQL injection protection.

API Security

CORS, CSRF protection, request validation, versioning.

7. Performance Considerations

Caching Strategy

Redis caching with invalidation and compression.

Database Optimization

Indexes, optimized queries, connection pooling.

Frontend Performance

Code splitting, lazy loading, asset optimization.

8. Error Handling

Error Responses

Standardized format, correct HTTP codes, user-friendly messages.

Monitoring & Logging

Centralized logging, error tracking, alerts.

9. Deployment

Development

Docker Compose, SQLite, hot reload.

Production

Docker containers, PostgreSQL, load balancing, auto-scaling.

10. Future Enhancements

Short-term

Push notifications, calendar integration, bulk admin ops.

Long-term

AI-based recommendations, analytics dashboard, offline mobile support, integration with college systems.

11. Conclusion

This design provides a scalable, secure, and user-friendly event management system, ensuring smooth experiences for students and administrators.