Campus Event Management Platform - Design Document

Table of Contents

---

1. Data to Track

Primary Data Entities

Event Creation Data

- **Event Metadata**: Title, description, type (workshop/hackathon/fest/seminar)
- **Scheduling**: Date, time, duration, timezone
- **Logistics**: Venue, capacity, registration deadline
- **Management**: Created by (admin), creation timestamp, last modified
- **Status**: Active, cancelled, completed, draft

Student Registration Data

- **Registration Details**: Student ID, event ID, registration timestamp
- **Registration Status**: Confirmed, waitlisted, cancelled
- **Additional Info**: Dietary preferences, team formation (for hackathons)
- **Communication**: Email confirmations sent, reminders sent

Attendance Tracking Data

- **Check-in Information**: Check-in timestamp, check-in method (QR, manual)
- **Attendance Status**: Present, absent, late arrival

- **Session Tracking**: For multi-day events, track daily attendance

- **Location Verification**: GPS coordinates (if mobile check-in)

Feedback Data

- **Ratings**: Overall satisfaction (1-5), content quality, organization

- **Qualitative Feedback**: Comments, suggestions for improvement

- **Specific Metrics**: Would recommend (yes/no), likelihood to attend similar events

- **Submission Info**: Feedback timestamp, completion status

Analytics & Reporting Data

- **Event Performance**: Registration rates, attendance rates, capacity utilization

- **Student Engagement**: Events attended per student, favorite event types

- **College Metrics**: Most popular events per college, seasonal trends

- **Operational Data**: Peak registration times, check-in efficiency

---

2. Database Schema

Entity Relationship Diagram

```
COLLEGES
 ├── id (PK)
 ├── name
 ├── domain
 ├── address
 ├── contact_email
 ├── created_at
 └── settings (JSON)
```

```
USERS
 ├── id (PK)
 ├── college_id (FK → COLLEGES.id)
 ├── email (UNIQUE)
 ├── password_hash
 ├── role (admin/student)
 ├── first_name
 ├── last_name
 ├── student_id (for students)
 ├── department
 ├── year_of_study
 ├── created_at
└── last_login

EVENTS
 ├── id (PK)
 ├── college_id (FK → COLLEGES.id)
 ├── created_by (FK → USERS.id)
 ├── title
 ├── description
 ├── event_type (workshop/hackathon/fest/seminar/competition)
 ├── start_date
 ├── end_date
 ├── start_time
 ├── end_time
 ├── venue
 ├── capacity
 ├── registration_deadline
 ├── status (draft/active/cancelled/completed)
 ├── requirements (JSON)
 ├── created_at
└── updated_at

REGISTRATIONS
 ├── id (PK)
 ├── user_id (FK → USERS.id)
 ├── event_id (FK → EVENTS.id)
 ├── registration_status (confirmed/waitlisted/cancelled)
 ├── registered_at
 ├── waitlist_position
 ├── additional_info (JSON)
└── notification_sent

ATTENDANCE
 ├── id (PK)
 ├── registration_id (FK → REGISTRATIONS.id)
 ├── check_in_time
 ├── check_out_time
 ├── attendance_status (present/absent/late)
 ├── check_in_method (qr/manual/app)
 ├── session_date (for multi-day events)
 ├── location_lat
 ├── location_lng
└── verified_by (FK → USERS.id)

FEEDBACK
 ├── id (PK)
 ├── registration_id (FK → REGISTRATIONS.id)
 ├── overall_rating (1-5)
 ├── content_rating (1-5)
 ├── organization_rating (1-5)
 ├── would_recommend (boolean)
 ├── comments (TEXT)
 ├── suggestions (TEXT)
 ├── submitted_at
└── is_anonymous

EVENT_ANALYTICS
 ├── id (PK)
 ├── event_id (FK → EVENTS.id)
 ├── total_registrations
 ├── total_attendance
 ├── attendance_rate
 ├── average_rating
 ├── peak_registration_time
 ├── calculated_at
└── metrics (JSON)
```

Key Constraints & Indexes

Unique Constraints
- `(user_id, event_id)` in REGISTRATIONS (prevent duplicate registrations)
- `(registration_id, session_date)` in ATTENDANCE (one attendance per session)
- `email` in USERS (unique across platform)

Indexes for Performance
- `college_id` in EVENTS, USERS (college-specific queries)
- `event_id` in REGISTRATIONS, ATTENDANCE (event-specific lookups)
- `start_date` in EVENTS (date-range queries)

- `registered_at` in REGISTRATIONS (chronological queries)

---

3. API Design

Authentication Endpoints
```
POST   /api/auth/login
POST   /api/auth/logout
POST   /api/auth/register
POST   /api/auth/forgot-password
PUT    /api/auth/reset-password
GET    /api/auth/profile
PUT    /api/auth/profile
```

Event Management Endpoints
```
GET    /api/events                  # List events (with filters)
POST   /api/events                  # Create event (admin only)
GET    /api/events/{id}             # Get event details
PUT    /api/events/{id}             # Update event (admin only)
DELETE /api/events/{id}             # Cancel event (admin only)
GET    /api/events/{id}/registrations # Get event registrations (admin)
GET    /api/events/{id}/analytics     # Get event analytics (admin)
```

Registration Endpoints
```
POST   /api/events/{id}/register      # Register for event
```

```
DELETE /api/registrations/{id}       # Cancel registration
GET    /api/users/{id}/registrations  # Get user's registrations
PUT    /api/registrations/{id}/status # Update registration status (admin)
```

Attendance Endpoints

```
POST   /api/attendance/checkin        # Check-in to event
POST   /api/attendance/checkout       # Check-out from event
GET    /api/events/{id}/attendance    # Get attendance list (admin)
PUT    /api/attendance/{id}           # Update attendance record (admin)
POST   /api/attendance/bulk-checkin   # Bulk check-in (admin)
```

Feedback Endpoints

```
POST   /api/feedback                  # Submit event feedback
GET    /api/events/{id}/feedback      # Get event feedback (admin)
GET    /api/feedback/summary/{id}     # Get feedback summary (admin)
```

Reporting Endpoints

```
GET    /api/reports/event-popularity  # Event popularity report
GET    /api/reports/student-participation # Student participation report
GET    /api/reports/attendance-trends # Attendance trends
GET    /api/reports/college-summary   # College-wide summary
GET    /api/reports/top-students      # Most active students
```

API Response Format

```json
{
  "success": true,
  "data": {
    // Response data
  },
  "message": "Success message",
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 100,
    "pages": 5
  }
}
```

Error Response Format

```json
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid input data",
    "details": {
      "field": "email",
      "reason": "Invalid email format"
    }
  }
}
```

---

4. Workflows

Student Registration Flow

```mermaid
sequenceDiagram
    participant S as Student
    participant UI as Frontend
    participant API as Backend API
    participant DB as Database
    participant Email as Email Service

    S->>UI: Browse events
    UI->>API: GET /api/events
    API->>DB: Query available events
    DB-->>API: Return events list
    API-->>UI: Events data
    UI-->>S: Display events

    S->>UI: Click "Register" for event
    UI->>API: POST /api/events/{id}/register
    API->>DB: Check capacity & duplicates

    alt Capacity available
        DB-->>API: Registration allowed
        API->>DB: Create registration record
        DB-->>API: Registration created
        API->>Email: Send confirmation email
        API-->>UI: Registration successful
        UI-->>S: Success message
    else Event full
        DB-->>API: Event at capacity
        API->>DB: Add to waitlist
        API-->>UI: Added to waitlist
        UI-->>S: Waitlist notification
    else Already registered
        DB-->>API: Duplicate registration
        API-->>UI: Error: Already registered
        UI-->>S: Error message
    end
```

Event Check-in Flow

```mermaid
sequenceDiagram
    participant S as Student
    participant App as Mobile App
    participant API as Backend API
    participant DB as Database
    participant Admin as Admin Dashboard

    S->>App: Open check-in (QR scan/manual)
    App->>API: POST /api/attendance/checkin
    API->>DB: Verify registration exists

    alt Valid registration
        DB-->>API: Registration found
        API->>DB: Check if already checked in

        alt Not checked in yet
            DB-->>API: First check-in
            API->>DB: Create attendance record
            DB-->>API: Attendance recorded
            API-->>App: Check-in successful
            App-->>S: Welcome message

            API->>Admin: Real-time attendance update
        else Already checked in
            DB-->>API: Already present
            API-->>App: Already checked in
            App-->>S: "Already checked in" message
        end
    else Invalid registration
        DB-->>API: No registration found
        API-->>App: Error: Not registered
        App-->>S: Registration required message
    end
```

Event Reporting Workflow

```mermaid
sequenceDiagram
    participant A as Admin
    participant UI as Admin Dashboard
    participant API as Backend API
    participant DB as Database
    participant Cache as Redis Cache

    A->>UI: Request event report
    UI->>API: GET /api/reports/event-popularity
    API->>Cache: Check cached report
    alt Cache hit
        Cache-->>API: Return cached data
        API-->>UI: Report data
    else Cache miss
        API->>DB: Query event statistics
        DB-->>API: Raw data
        API->>API: Process & aggregate data
        API->>Cache: Cache processed report
        API-->>UI: Report data
    end

    UI-->>A: Display interactive report

    A->>UI: Export report
    UI->>API: GET /api/reports/export
    API->>API: Generate PDF/CSV
    API-->>UI: Download link
    UI-->>A: File download
```

---

5. Assumptions & Edge Cases

Core Assumptions

Technical Assumptions
- **Database**: PostgreSQL with proper indexing for scale
- **Authentication**: JWT tokens with refresh mechanism
- **Caching**: Redis for frequently accessed data
- **File Storage**: Cloud storage for event images/documents

- **Email Service**: Third-party service (SendGrid/SES) for notifications

Business Assumptions

- **College Independence**: Each college operates independently

- **Registration Deadline**: Events have configurable registration deadlines

- **Capacity Management**: Hard limits enforced at database level

- **Feedback Timing**: Feedback collected within 7 days after event

- **Data Retention**: Event data retained for 2 years for analytics

Edge Cases & Handling

Registration Edge Cases

Duplicate Registration Attempts

- **Scenario**: Student tries to register multiple times

- **Prevention**: Unique constraint on (user_id, event_id)

- **Handling**: Return appropriate error message

- **UI Response**: Disable register button after first click

Simultaneous Registration at Capacity

- **Scenario**: Multiple students register when 1 spot remaining

- **Prevention**: Database-level capacity checking with transactions

- **Handling**: First successful transaction gets spot, others waitlisted

- **UI Response**: Real-time capacity updates via WebSocket

Registration After Deadline

- **Scenario**: Student attempts registration after deadline

- **Prevention**: API-level date validation

- **Handling**: Return deadline exceeded error

- **UI Response**: Hide register button after deadline

Event Management Edge Cases

Event Cancellation with Registrations

- **Scenario**: Admin cancels event with existing registrations
- **Handling**:
  - Update event status to 'cancelled'
  - Send cancellation emails to all registered students
  - Offer automatic registration to similar events
- **Data**: Maintain registration history for reporting

Venue Change After Registration

- **Scenario**: Event venue changes after students register
- **Handling**:
  - Update event details
  - Send notification emails with new venue
  - Log change in audit trail
- **UI Response**: Prominent notification on event page

Capacity Increase/Decrease

- **Scenario**: Admin changes event capacity
- **Handling**:
  - If increased: Auto-confirm waitlisted students (FIFO)
  - If decreased: Move excess confirmed registrations to waitlist
  - Send appropriate notifications
- **Data**: Log all capacity changes with timestamps

Attendance Edge Cases

Late Check-in

- **Scenario**: Student arrives after event start time
- **Handling**: Allow check-in with "late" status

- **Reporting**: Track late arrivals separately
- **Business Rule**: Define late threshold (e.g., 30 minutes)

Check-in Without Registration
- **Scenario**: Non-registered student tries to check-in
- **Handling**:
  - Deny check-in
  - Offer on-spot registration if capacity available
  - Log attempt for security monitoring

Bulk Check-in Errors
- **Scenario**: Admin bulk check-in fails partially
- **Handling**:
  - Process successful records
  - Return detailed error report
  - Allow retry for failed records
- **UI**: Progress indicator with error details

Data Integrity Edge Cases

Missing Feedback
- **Scenario**: Low feedback response rates
- **Handling**:
  - Send reminder emails (2-3 times max)
  - Track response rates by event type
  - Incentivize feedback with rewards
- **Reporting**: Indicate confidence levels based on response rates

Orphaned Records
- **Scenario**: Referenced records deleted (cascade failures)
- **Prevention**: Foreign key constraints with appropriate cascade rules

- **Handling**: Regular data integrity checks
- **Recovery**: Maintain audit logs for reconstruction

Time Zone Issues
- **Scenario**: Multi-campus events across time zones
- **Handling**:
  - Store all times in UTC
  - Display in user's local time zone
  - Clear time zone indication in UI
- **API**: Include timezone information in responses

System Performance Edge Cases

Registration Rush
- **Scenario**: Popular event causes traffic spike
- **Handling**:
  - Implement rate limiting per user
  - Use queue system for registration processing
  - Auto-scaling infrastructure
- **UI**: Queue position indicator, estimated wait time

Large Event Check-ins
- **Scenario**: 500+ students checking in simultaneously
- **Handling**:
  - Implement QR code batch scanning
  - Offline check-in capability with sync
  - Multiple check-in stations
- **Infrastructure**: Load balancing and database connection pooling

Security Edge Cases

Fraudulent Registrations

- **Scenario**: Automated bot registrations
- **Prevention**: CAPTCHA, email verification, rate limiting
- **Detection**: Pattern analysis (multiple registrations from same IP)
- **Response**: Account suspension, manual review process

Admin Account Compromise

- **Scenario**: Admin credentials stolen
- **Handling**:
  - Multi-factor authentication required
  - Audit log of all admin actions
  - Session timeout and IP restrictions
- **Recovery**: Emergency admin override process

Business Logic Edge Cases

Event Conflicts

- **Scenario**: Student registers for overlapping events
- **Handling**:
  - Warn during registration
  - Allow registration but flag conflicts
  - Provide conflict resolution suggestions
- **Analytics**: Track conflict patterns for better scheduling

Waitlist Management

- **Scenario**: Multiple cancellations create complex waitlist scenarios
- **Handling**:
  - FIFO promotion with time-based expiry
  - Automatic notification with acceptance deadline
  - Manual admin override capability
- **Transparency**: Clear waitlist position communication