AVIJOZI Project - Full System and Database Design

# 1. Project Overview

AVIJOZI is an event management platform built to support a dynamic, real-time environment for large-scale conferences. The system facilitates session scheduling, attendee check-ins, room occupancy tracking, feedback collection, and sponsor booth analytics.

# 2. Database Design

The following tables will form the core of the Supabase PostgreSQL database, designed to enable fast querying and real-time updates.

## 2.1 Table Overview

|  |  |
| --- | --- |
| **Table Name** | **Purpose** |
| attendees | Store user profiles, preferences |
| speakers | Session speakers and bios |
| rooms | Physical rooms (room name, capacity, etc.) |
| sessions | Event sessions (time, room, speaker) |
| checkins | Stores time-based check-ins via QR scan |
| feedback | Ratings and optional text |
| sponsors | Sponsor data for reports |
| booths | Sponsor booths info |

## 2.2 SQL Schema

**Attendees**  
CREATE TABLE attendees (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

full\_name TEXT NOT NULL,

email TEXT UNIQUE NOT NULL,

age INT CHECK (age > 0),

gender TEXT CHECK (gender IN ('Male', 'Female', 'Non-Binary', 'Prefer not to say')),

occupation TEXT,

organization TEXT,

country TEXT,

city TEXT,

interests TEXT[],

registered\_at TIMESTAMP DEFAULT now()

);  
  
**Speakers**  
CREATE TABLE speakers (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 full\_name TEXT NOT NULL,  
 bio TEXT,  
 photo\_url TEXT  
);  
  
**Rooms**  
CREATE TABLE rooms (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name TEXT NOT NULL,  
 capacity INT,  
 location TEXT  
);  
  
**Sessions**  
CREATE TABLE sessions (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 title TEXT NOT NULL,  
 description TEXT,  
 start\_time TIMESTAMP,  
 end\_time TIMESTAMP,  
 room\_id UUID REFERENCES rooms(id),  
 speaker\_id UUID REFERENCES speakers(id)  
);  
  
**Check-ins**  
CREATE TABLE checkins (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 attendee\_id UUID REFERENCES attendees(id),  
 session\_id UUID REFERENCES sessions(id),  
 checked\_in\_at TIMESTAMP DEFAULT now()  
);  
  
**Feedback**  
CREATE TABLE feedback (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 session\_id UUID REFERENCES sessions(id),  
 attendee\_id UUID REFERENCES attendees(id),  
 rating INT CHECK (rating BETWEEN 1 AND 5),  
 comment TEXT,  
 submitted\_at TIMESTAMP DEFAULT now()  
);  
  
**Sponsors**  
CREATE TABLE sponsors (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name TEXT NOT NULL,  
 logo\_url TEXT  
);  
  
**Booths**  
CREATE TABLE booths (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 sponsor\_id UUID REFERENCES sponsors(id),  
 location TEXT,  
 traffic\_count INT DEFAULT 0  
);

# 3. System Architecture

The system will be divided into three main components:

1. FastAPI Backend
2. Supabase Database (PostgreSQL + Realtime)
3. Frontend Apps (React Native mobile app and Admin dashboard)

## 3.1 Component Overview

- **FastAPI REST API**: Handles business logic, authentication, QR check-ins, and data endpoints.

- **Supabase**: PostgreSQL database with real-time listeners on check-ins and feedback.

- **React Native App**: Used by attendees for check-ins, browsing sessions, and submitting feedback.

- **Admin Dashboard**: Used for monitoring occupancy, analyzing feedback, and managing schedules.

- **Display Screens**: Static frontends showing session info and live room data.

# 4. Real-Time Sync Design

Supabase Realtime is used to stream check-in and feedback changes. Screens and dashboards listen for changes on the `checkins` and `feedback` tables using a WebSocket subscription or Supabase client library.

# 5. API Design (FastAPI)

API endpoints will be structured under versioned routes (e.g., /api/v1/...).

Auth tokens will be required for most endpoints.

## 5.1 Sample Endpoints

- POST /api/v1/auth/login  
- GET /api/v1/sessions/  
- GET /api/v1/sessions/{id}  
- POST /api/v1/checkins/ (with QR data)  
- POST /api/v1/feedback/  
- GET /api/v1/rooms/live  
- GET /api/v1/admin/dashboard/