1. System Mission & Core Value (Spec document)

- The project is a high-availability, **Full-Stack Event Orchestration Platform** engineered to eliminate the logistical friction inherent in high-traffic venue management. By bridging the gap between physical space constraints and the attendee journey, the system transforms chaotic queues into a streamlined, data-driven experience.

- ### Core Objectives:

- **Elevated Attendee Experience:** We empower attendees by providing transparency. Through a dedicated mobile interface, users can view live wait times, their current queue position, and specific event data. This reduces "queue anxiety" and allows them to manage their time effectively within the venue.
- **Intuitive Self-Service (Self-Healing Queues):** Attendees have the autonomy to drop out of events directly through the app. The system intuitively handles these cancellations by immediately promoting the next eligible attendee to fill the vacant spot. This ensures 100% room utilization and faster movement for everyone involved.
- **Infrastructure Agility (Offline-First):** Recognizing that large-scale venues often suffer from "dead zones" or saturated networks, the platform is built with a **Local-First Architecture**. This ensures that staff can perform check-ins, capacity updates, and waitlist management with zero latency, even in totally offline environments.
- **Staff Workload Optimization:** By automating the promotion of attendees and the detection of "no-shows," we significantly reduce the cognitive load on staff. The interface is designed for high-stress environments, replacing complex spreadsheets with intuitive, color-coded dashboards.
- **Feature Innovation:** Beyond traditional check-in tools, this platform introduces unique capabilities such as **Monte Carlo simulations** for strategy testing, **vibration-alert promotions** for attendees, and **AI-driven no-show probability indicators**.

2. Technical Stack & Rationale

- ## A. Frontend (Staff): Next.js & Figma AI

- **The Choice:** Next.js (Web) with Figma AI integration.
- **Rationale:** For a high-density staff dashboard, we need the **Server-Side Rendering (SSR)** and **Virtualization** capabilities of Next.js to handle 5,000+ waitlist entries without browser lag.
- **Design Agility:** We are utilizing **Figma AI** to bridge the gap between UI/UX design and code. This allows the team to rapidly iterate on layout changes based on user feedback and generate high-fidelity components that map directly to our React code, ensuring the staff interface remains intuitive under high-pressure scenarios.

- ## B. Frontend (Attendee): React Native

- **The Choice:** React Native (Mobile Framework).
- **Rationale:** Unlike traditional web apps, React Native gives us **native access** to mobile hardware.
  - **Native Performance:** It allows for smooth 60 FPS animations (crucial for our queue position arrows) and robust background processing.
  - **Reliable Alerts:** It provides superior integration with **Firebase Cloud Messaging (FCM)** for push notifications and haptic feedback (vibrations), ensuring attendees are alerted even if the app is in their pocket.
  - **Cross-Platform Efficiency:** A single codebase allows us to deploy to both iOS and Android simultaneously, ensuring no attendee is excluded regardless of their device.

- ## C. Backend: Node.js & FastAPI (Hybrid)

- **The Choice:** Node.js for Real-Time; FastAPI for Logic/AI.
- **Rationale:** We are using a hybrid approach to play to the strengths of both environments.
  - **Node.js (Socket.io):** Best-in-class for maintaining thousands of concurrent **WebSocket** connections. This ensures that when an attendee drops an event, every staff member's screen updates in sub-200ms.
  - **FastAPI (Python):** This acts as our "Brain." Python is the industry standard for AI and Data Science. FastAPI allows us to serve our **Wait-Time Regression Models** and **LSTM forecasts** with high performance and type safety, making it easy to integrate our ML retraining pipelines.

- ### D. Data Layer: PostgreSQL & Redis

- **The Choice:** PostgreSQL (Primary) + Redis (Cache/Queue).
- **Rationale:** * **PostgreSQL:** Chosen for its **Strict Relational Integrity** and support for complex JOINs. In event management, data consistency is non-negotiable (e.g., an attendee cannot be checked into two rooms simultaneously). We utilize **Materialized Views** to make heavy analytics queries run instantly.
    - **Redis:** Acts as our high-speed buffer. Since the waitlist is a **FIFO (First-In, First-Out)** structure, Redis's in-memory list operations allow us to handle rapid-fire promotions and updates without hitting the main database every millisecond.

- ### E. Architecture: Offline-First (RxDB & SQLite)

- **The Choice:** IndexedDB (Web) / SQLite (Mobile) with Conflict-Resolution.
- **Rationale:** Traditional "Cloud-Only" apps fail in crowded convention centers.
    - **Local-First:** By storing the waitlist state locally on the staff's device using **RxDB**, the UI remains responsive even if the network is "flapping" (going in and out).
    - **Sync Logic:** We implement a **Delta-Sync strategy**. When a connection is restored, the system doesn't re-upload everything; it only sends the small "deltas" (changes).
    - **Conflict Resolution:** Figure out how to handle offline edits from multiple devices at the same time, or just dont allow offline edits

3. Component Deep Dive

A. The "Intelligent" Waitlist Engine

Unlike a simple list, this engine uses a **Multi-Factor Priority FIFO** logic.

- **Dynamic Promotion:** When a room's capacity drops, the engine doesn't just "ping" the next person; it triggers a **TTL (Time-to-Live) Check-in Window**.
- **Priority Tiering:** It manages overlapping queues for VIP, ADA, and General Admission, ensuring accessibility and business priorities are met automatically.
- **No-Show Logic:** Uses a Gradient Boosting model to flag "High Probability No-Shows," allowing staff to over-book or pre-promote to maintain 100% room efficiency.

B. Predictive Analytics & Simulation

This is the "Brain" of the project.

- **Wait Time Estimator:** Uses a **Regression Model** factoring in arrival rates and historical data.
- **LSTM Forecasting:** A Long Short-Term Memory (LSTM) network predicts the "next hour" of traffic, allowing staff to reallocate resources before a surge happens.
- **Monte Carlo Simulation:** A sandbox environment where organizers can test "What happens if 20% of people don't show up?" to optimize their promotion timing.

C. Real-Time Sync & Offline Resilience

Designed for "Battlefield Conditions" (e.g., a convention center with spotty Wi-Fi).

- **The Delta-Sync:** Instead of re-downloading the whole list, the system only pushes the "Delta" (the changes), saving bandwidth.
- **Local-First Architecture:** Staff can continue checking people in while offline; the system logs these in a local SQLite store and uses a **Conflict Resolution Strategy** (Last-Write-Wins) once the connection is restored.

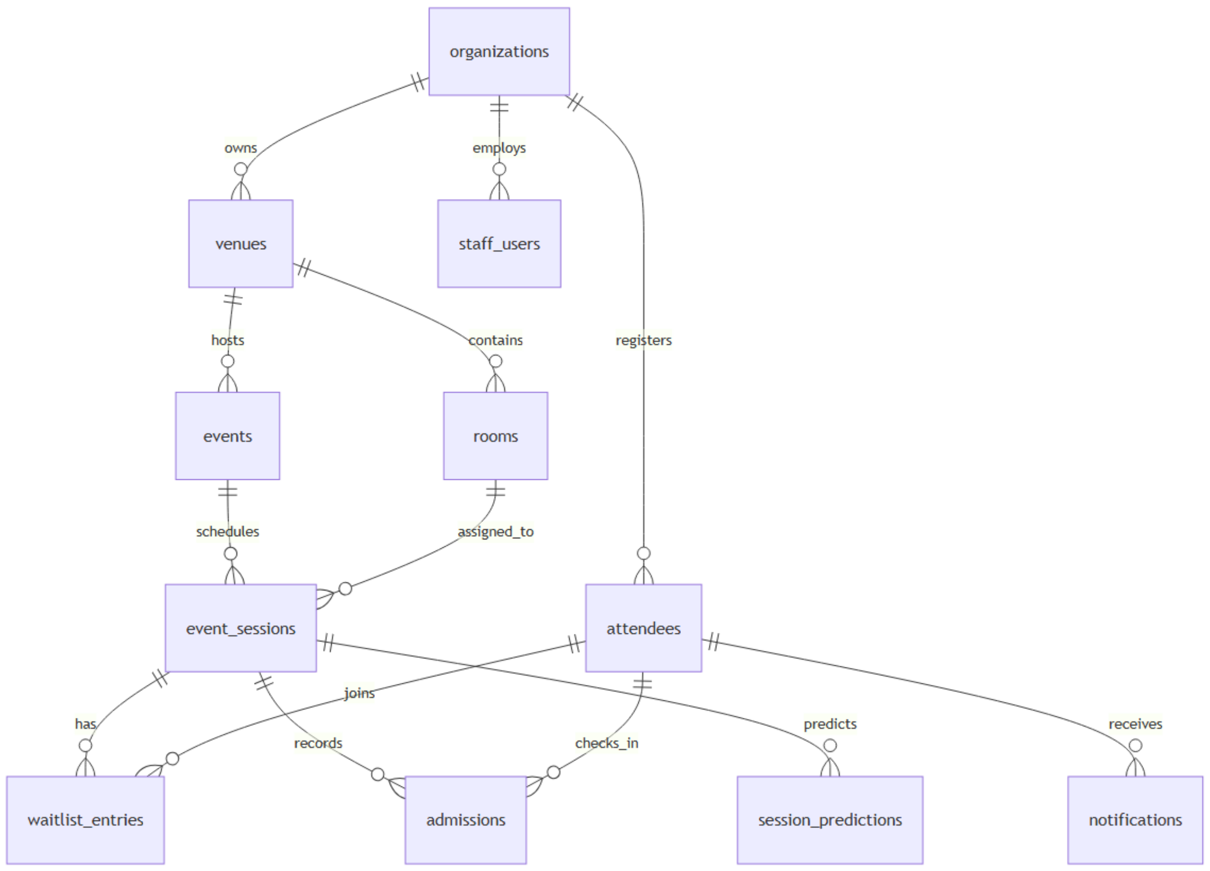
4. Key Feature Implementation

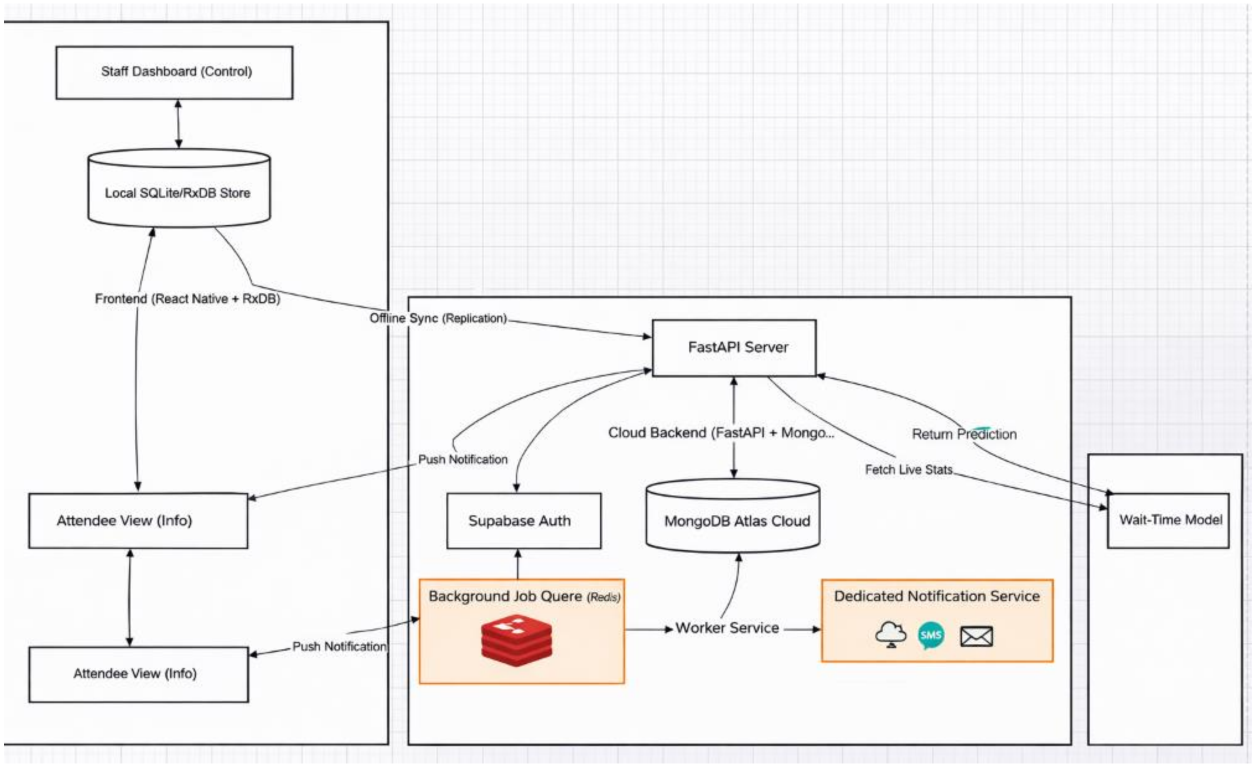
- **Real-Time Capacity Progress Bar**
  - **Tech:** WebSockets + D3.js
  - **Benefit:** Instant visual safety thresholds (Green/Yellow/Red) for immediate crowd control.
- **Virtualized Waitlist**
  - **Tech:** React-Window
  - **Benefit:** Efficiently renders 5,000+ entries by only loading visible rows, preventing browser lag.
- **Multi-Channel Notification Fallback**
  - **Tech:** Firebase + Twilio
  - **Benefit:** Automatically sends SMS if Push notifications fail due to poor venue Wi-Fi.
- **Immutable Audit Logging**
  - **Tech:** Middleware + PostgreSQL

    - **Benefit:** Creates a permanent security trail for all manual staff overrides and data changes.
- **Heatmap Analytics**
    - **Tech:** Materialized Views
    - **Benefit:** Pre-calculates traffic "hot zones" for instant dashboard rendering without slowing down the database.
- **Self-Healing Queue Promotion**
    - **Tech:** Redis + Node.js
    - **Benefit:** Automatically promotes the next attendee the moment someone drops, maximizing room utility.

5. Non-Functional Specifications

- **Scalability:** Partitioned PostgreSQL tables to handle massive attendance spikes.
- **Latency:** WebSocket updates must propagate to all clients in under 200ms.
- **Reliability:** 99.9% uptime for the notification service to prevent venue bottlenecks.
- **Security:** Role-Based Access Control (RBAC) ensures a volunteer can't accidentally delete an entire venue's data.


organizations
owns
employs
registers
venues
staff_users
hosts
contains
events
rooms
schedules
assigned_to
event_sessions
attendees
joins
has
records
checks_in
predicts
receives
waitlist_entries
admissions
session_predictions
notifications


Staff Dashboard (Control)
Local SQLite/RxDB Store
Frontend (React Native + RxDB)
Offline Sync (Replication)
FastAPI Server
Cloud Backend (FastAPI + Mongo...
Return Prediction
Fetch Live Stats
Push Notification
Supabase Auth
MongoDB Atlas Cloud
Wait-Time Model
Attendee View (Info)
Background Job Quere (Redis)
Worker Service
Dedicated Notification Service
SMS
Push Notification
Attendee View (Info)