Project Proposal: Team 12

1. Project Overview

Project Title

Real-Time Event Waitlist Management System

Summary

Large-scale events currently rely on manual or paper-based waitlists that are inefficient and prone to failure under high-volume, multi-venue conditions with limited connectivity. This project is a real-time event waitlist management platform designed to streamline entry management and room capacity across these complex environments.

By automating waitlist promotion and attendee notifications through a centralized staff dashboard, the system reduces staff workload and improves the attendee experience. Furthermore, it leverages AI to predict wait times and attendance, allowing organizers to optimize attendee flow and room capacity effectively.

2. Project Scope

Core Functionalities (Must-Haves)

- Real-time room capacity management across multiple venues.
- Automated waitlist promotion logic to fill vacancies instantly.
- Attendee notification system for real-time updates.
- Centralized staff dashboards for monitoring flow and entry.
- Offline-first capability to handle low-connectivity environments.
- AI-driven predictive modeling for highly accurate wait time estimations.
- Modularize the attendance tracking in order to predict wait times and no-shows for places like theme parks, museums, etc.

Stretch Goals (Nice-to-Haves)

- Advanced flow optimization algorithms based on historical attendance patterns.
- Allow for the app to work with limited WiFi connection

3. Project Objectives

- **Digital Transformation**: Replace manual and paper-based tracking with a real-time digital interface to reduce human error.
- **Capacity Optimization**: Use AI to predict attendance and wait times, ensuring venues operate at peak efficiency.
- **Performance Reliability**: Maintain high system responsiveness with success metrics such as <2s load times and at least 90% accuracy in AI predictions.

4. Specifications

User Interface (UI) Design

- **Platform**: The system will be developed as a mobile application to serve both staff on-site and attendees on their own devices.
- **Key Screens**:
  - **Staff Dashboard**: Displays real-time occupancy, waitlist counts, and manual override controls.
  - **Attendee View**: Shows current waitlist position and estimated wait time predictions.

Backend & APIs

- **Structure**: A real-time API structure to handle high-frequency status updates across venues.
- **Data & AI Model**: The ML model will use historical and real-time event data to estimate wait times and identify potential bottlenecks in attendee flow.

5. Tech Stack

- **Frontend:** React Native for local caching and offline-first UI rendering.
- **AI Tools:**
    - **Gemini Coding**: Frontend scaffolding, UI logic, cross-platform component reasoning
    - **RooCode (or GitHub Copilot)**: Code comprehension, debugging, and refactoring
    - **GitHub Copilot**: Continuous AI pair-programming for React Native development
- **Backend:** Next.js or FastAPI (Python) to manage high-concurrency requests and AI logic.
- **AI Tools:**
    - **Claude Code**: Long-context code review, refactoring, and validation of complex offline and concurrency logic
    - **GitHub Copilot (or RooCode)**: Backend code assistance and AI pair programming
- **Database:** MongoDB for cloud synchronization and SQLite/RxDB for local on-device data residency.
- **Authorization:** Supabase
- **AI/ML:** Scikit-learn or TensorFlow for wait-time prediction models.
- **AI Tools for Support & Planning:**
    - **Emergent**: Frontend architecture, UX flow design, and system-level planning (limited free tier access for MVP)
    - **Codewiki**: AI-powered documentation and architectural knowledge base
- **Hardware:** Mobile devices for staff and attendees; potential edge gateways for local venue processing.

6. Project Timeline

Phase	Duration	Tasks	Status/Deliverable
Phase 1	Feb 10 – Feb 16 (1 week)	Project kickoff, scope finalization, and offline-sync strategy research.	Architecture Diagram

Phase 2	Feb 17 – Feb 23 (1 week)	Backend environment setup and database schema design for waitlists.	Environment Ready
Phase 3	Feb 24 – Mar 2 (1 week)	UI/UX Design (Figma) for user and admin interfaces.	High-fidelity Wireframes
Phase 4	Mar 3 – Mar 9 (1 week)	Data collection/sourcing for AI and initial API endpoint setup.	Core API Setup
Phase 5	Mar 10 – Mar 16 (1 week)	Development of core waitlist logic (joining, queuing, and status updates).	Logic Prototypes
Phase 6	Mar 17 – Mar 23 (1 week)	Integration of waitlist logic into the app frontend.	Functional Alpha (MVP)
Phase 7	Mar 24 – Mar 30 (1 week)	Initial AI integration for predictive wait times or user sorting.	Integrated AI Modules
Phase 8	Mar 31 – Apr 6 (1 week)	**Testing Start:** Stress testing for high-volume event scenarios.	QA Report (Initial)
Phase 9	Apr 7 – Apr 13 (1 week)	Beta version deployment for class peer-testing and bug logging.	Beta Version
Phase 10	Apr 14 – Apr 20 (1 week)	Final bug fixes, UI polishing, and deployment to hosting platform.	Final Demo Version
Phase 11	Apr 21 – May 1 (1.5 weeks)	Presentation preparation, final documentation, and GitHub handover.	GitHub Handover

7. Team Leader Rotation

Duration	**Team Leader**
February 4 – February 24	Disha Shetty
February 25 - March 17	Angela Lopez
March 18 - April 7	Neel Vavilapalli
April 8 - May 1	Kyle Albuquerque

8. Project Team

Role	Team Member	Responsibilities
Frontend Developer	**Angela Lopez**	UI development and user interaction design.
Backend Developer	**Neel Vavilapalli**	API development and database management.
Backend Developer	**Kyle Albuquerque**	Backend architecture and Testing
ML Engineer	**Disha Shetty**	AI/ML model development for wait-time predictions + Frontend

9. Links

- GitHub Repository: https://github.com/zap-bit/CS4485-Final-Project
- Agile Board (Jira/Trello): Jira - Project Manaement Board
- Design Document: Figma demo