

Smart Parking Finder

1. Problem Statement & Project Scope

Finding affordable and available parking in urban areas such as San Jose is a major challenge for students and commuters. Existing solutions are fragmented, often limited to specific garages or lots, and do not provide a unified experience.

Smart Parking Finder aims to provide a cross-platform mobile application that:

- Shows available parking spots near the user's destination.
- Allows filtering based on price, distance, and availability.
- Provides offline caching of recent searches.
- Integrates an **AI-powered assistant** to interpret natural language queries like:
 - "Find me the cheapest parking near SJSU."
 - "Where can I park overnight near the library?"
 - "Find the safest parking near me"
 - "Give me the top 5 parking spots around "<ADDRESS>".

The app will combine real-time parking APIs, geolocation, and user-friendly design into one unified system.

2. Proposed Features

1. Core Features

- Real-time parking availability map (via external API / simulated backend data).
- Search & filter (by price, distance, availability, type).
- Reserve spot (simulated backend, dummy workflow).
- Offline caching of recent searches.

2. Advanced Features (course requirement)

- **AI Assistant Integration:** LLM backend converts natural language queries → structured parking searches.
- **Push Notifications:** Alert users of expiring parking sessions or nearby availability.

3. User Experience

- Simple, intuitive design based on Google Material Design & Apple HIG.
 - Personas: students, commuters, and event-goers.
 - Storyboards & user journeys will capture end-to-end parking workflows.
-

3. Technology Stack

- **Frontend (Mobile App):** React Native (cross-platform for iOS and Android).
 - **State Management:** Redux or Context API.
 - **Backend:** Node.js + Express (for parking API aggregation & AI query handling).
 - **Database:** Firebase Firestore (storing user history, favorites).
 - **APIs:** Geoapify Places API (parking category), HERE Parking API (if needed).
 - **AI Integration:** OpenAI API (or other LLM service) for natural language query parsing.
 - **Version Control:** GitHub (all commits, PRs, collaboration).
-

4. Timeline & Deliverables

- **Sept 23, 2025 – Project Proposal (this document)**
Problem statement, features, scope, tech stack, timeline.
- **Oct 1 – Oct 25, 2025 – Design & Setup**
 - UX research (personas, storyboards, user journeys).

- Wireframes, mockups.
 - System architecture diagrams.
 - Backend scaffolding & GitHub repo setup.
 - **Nov 4, 2025 – Progress Report**
 - Parking map & search functional.
 - Basic API integration & caching complete.
 - Summary of achievements, challenges, adjustments.
 - **Nov 5 – Nov 25, 2025 – Development Phase 2**
 - AI assistant integration.
 - Push notifications.
 - Testing (unit + integration).
 - Code reviews & refinements.
 - **Dec 2, 2025 – Final Submission & Presentation**
 - Fully functional Smart Parking Finder app.
 - GitHub repo with README & build instructions.
 - Final Report (requirements, design, system architecture, limitations, ethical considerations).
 - 20-min live demo & presentation.
-

5. Expected Outcomes

- **Technical Deliverable:** A cross-platform mobile app with working parking search, real-time updates, offline caching, and AI query interpretation.

- **Documentation:**
 - Project proposal (this).
 - Progress report.
 - Final report
 - **Presentation:** Live demo + technical Q&A.
-

6. Risks & Mitigation

- **API Coverage / Costs:** Some parking APIs have limits. → Mitigation: Use free tiers + fallback mock dataset.
- **AI Query Latency:** LLM calls may be slow. → Mitigation: Cache frequent queries and allow default filters.
- **Time Constraints:** Balancing advanced features with deadlines. → Mitigation: Prioritize core features, treat AI assistant as optional enhancement.