TripPlanr — Product Requirements Document (PRD)

Version: MVP / Gemini CLI Implementation

Owner: Marlon

Platform: iOS Mobile App + Gemini CLI backend + iMessage Extension

---

1.

Product Overview

TripPlanr is an intelligent meetup planning engine that passively monitors a groupchat (after being toggled on), analyzes conversation context, captures shared user locations, and outputs the **top 3 optimal meetup spots** based on travel fairness, user preferences, activity type, budget, and vibe.

The app focuses on **4+ person groupchats**, using AI to infer activities and constraints and producing **iMessage polls** (or fallback in-app polls) plus a **map view** showing user positions and candidate venues.

The main differentiator:

➡️ **Passive, contextual, real-time trip planning triggered by natural conversation + shared location updates.**

2.

Core MVP Workflow

2.1 Trip Mode Activation

A user presses **“Enable Trip Mode”** inside a specific iMessage groupchat via the app’s extension.

When Trip Mode is ON:

- App receives **periodic snapshots** of:
    - New messages in the groupchat
    - User-shared approximate locations

Snapshots occur every **2–5 minutes** or when a significant message is posted.

2.2 Data Inputs

A. iMessage Parsing (via App Extension)

Parse new messages and extract:

- Activity desires (e.g., “burgers,” “bar,” “sushi,” “something fun”)
- Preferences (cheap, fast, fancy, quiet, walkable, scenic, etc.)
- Soft constraints (tired, hungry, don’t want to walk far, etc.)
- Time windows (“after 7,” “I’m free now”)
- Conflicts (sushi vs pizza)

Gemini CLI endpoint:

- extract_trip_context(messages[])

B. Location Inputs (Shared by Users)

Each group member is prompted to **share approximate location** with the app.

We collect:

- Latitude/longitude
- Approximate radius
- Movement speed (optional)
- Last updated timestamp

Find-My-like accuracy is not required.

Gemini CLI endpoint:

- digest_locations(locations[])

2.3 AI Processing

A. Activity Inference Engine

Gemini identifies:

- Dominant activity category
- Secondary categories
- Multi-category matching (pizza + sushi → “general restaurants” or “plaza with multiple food options”)
- Contextual cues (“I’m starving,” “craving Asian,” etc.)

B. Venue Candidate Generator

Based on inferred activity:

- Query Yelp/Google Places

- Pull 15–20 candidates
- Filter by:
    - Activity type
    - Price
    - Rating
    - Hours
    - Distance from centroid

C. Travel Optimization Engine

Calculate:

- **Fairness score** (minimize difference in travel time)
- **Total travel burden**
- **Approximate driving time**
- **Uber estimated price** (heuristic pricing)
- **Convenience score**

Ranking is done using a weighted heuristic:

- Travel fairness (~40%)
- Rating (~25%)
- Relevance to activity (~20%)
- Price reasonability (~10%)
- Misc contextual cues (~5%)

D. Final Output Generation

Gemini selects top 3:

- **Option A: Fastest overall trip for the group**
- **Option B: Most fair travel time**
- **Option C: Vibe/Preference optimized venue**

Gemini produces:

- Title
- One-line vibe description
- Ratings + price
- Estimated travel times for each user
- Optional Uber cost range

3.

User-Facing Features

3.1 iMessage Poll Output

TripPlanr automatically posts:

- Three options
- Each includes:
    - Name

    - Summary
    - Estimated travel time
    - Price
    - Rating
    - Mini-emoji vibe tag (🍔 🎉 ☕)

If native iMessage poll API is restricted:

➡️ App posts a smart link that opens an in-app poll.

Poll results sync back to the groupchat.

3.2 Interactive Map

Inside the app:

- Users see:
    - Their own location
    - Friends' approximate locations
    - The 3 candidate meetup spots
- Tapping any spot shows:
    - Driving time
    - Uber estimate
    - Directions (open Apple Maps)

3.3 Real-Time Trip Reassessment

If:

- Someone moves significantly
- A new preference emerges
- A venue closes

  The model recomputes options and posts updated recommendations.

(“Trip Recompute” posts only when materially different.)

4.

Technical Architecture

4.1 Components

1. **iOS App**
    - Trip Mode toggle
    - Location sharing
    - Map rendering
    - Settings & permissions
2. **iMessage Extension**
    - Polls
    - Context message posting
    - Chat parsing
3. **Backend API**

- Built using Node.js/Python/Go
- Endpoints for:
  - Message ingestion
  - Location ingestion
  - Recommendation request
  - Poll result tracking

4. **Gemini CLI (Model Orchestration)**
   - Local prompts → calls Gemini APIs
   - Action workflows triggered by snapshots
   - Offers structured JSON outputs

5.

Gemini CLI Specification

5.1 Model Prompts / Tasks

A. parse_messages

gemini prompt --file messages.json \
"Extract all activities, preferences, constraints, times, conflicts,
and summarize the trip context in JSON."

Expected JSON:

```
{
  "activities": ["burgers", "bar"],
  "generalized_activity": "restaurants",
  "preferences": ["cheap", "casual"],
```

```
  "time_window": "after 7pm",
  "constraints": ["hungry", "dont want to walk far"],
  "conflicts": ["sushi", "pizza"]
}
```

B. process_locations

gemini prompt --file locations.json \
"Compute centroid, fairness metrics, distances, and travel clusters."

C. rank_venues

Gemini takes candidate venues + scoring rules:

gemini prompt --file venues.json \
--params weights.json \
"Rank these venues by fairness, rating, price, relevance. Output top 3."

D. generate_poll_message

gemini prompt \
"Create an iMessage-friendly poll text for these top 3 venues."

6.

Success Metrics

MVP Goals

- 80% accurate activity inference
- Poll posted within **<10 seconds** of decision
- 90%+ user location permission acceptance (friendly UX flow)
- Users successfully reach consensus within 1–2 polls

Qualitative

- Users feel the app “understands the vibe”
- Groupchat needs **no manual input** besides Trip Mode toggle

---

7.

Constraints & Limitations

Technical

- Native iMessage polls may need fallback
- Uber price is approximate
- Fine-grained real-time location may be limited

Ethical / Privacy

- All processing must respect:
    - Location sharing opt-in
    - Visibility into what data is used
- Trip Mode ON must be explicit