Product Requirements Document (PRD)

Product Name: ***The Middle***

Product Type: Web App (Mobile-first, responsive)

Version: v1.0

Source Document: The Middle: Product Overview

The Middle_ Product Overview

---

1. Overview

1.1 Product Summary

The Middle is a location-based social discovery web app that helps groups of friends find the ideal place to meet by:

- Detecting each friend's live location (with permissions),
- Calculating the geographic midpoint of the group,
- Recommending nearby venues (restaurants, bars, cafés, activities),
- Allowing the group to swipe/vote on venues Tinder-style,
- And revealing a final matched venue when all participants vote positively, or whichever venue has the most positive votes if no consensus is found.

The product combines functionalities from **Find My Friends**, **Yelp**, and **Tinder**, with emphasis on group decision-making.

1.2 Problem Statement

Meeting friends often requires negotiating location, preferences, and convenience. Groups waste time texting “Where should we go?” The Middle eliminates this friction by automatically recommending fair, central, group-friendly meetups.

1.3 Goals & Objectives

- Reduce time needed for groups to agree on a meeting place
- Provide fair, neutral, midpoint-driven venue selection
- Increase fun and engagement via swipe-based interaction
- Enable real-time multiplayer decision making

---

2. Target Users

2.1 Primary Users

- Friend groups ages 18–40
- Urban professionals and social groups
- College students organizing meetups
- Couples meeting halfway

2.2 Secondary Users

- Event planners or groups planning multi-person meetups
- Travelers coordinating meetups in new cities

---

3. Core Features & Requirements

3.1 Flexible User Location System (Live or Manual Input)

Description

Users may participate in a session by sharing their **real-time device location** or by providing a **manually set location** if they prefer not to enable permissions or are planning a future meetup.

Functional Requirements

- Present users with two options upon joining:
  A. Share Live Location (via browser/device permissions)
  B. Enter Manual Location (address search or pin-drop)
- If a user declines permissions, automatically route them to manual location selection.
- Distinguish visually between live vs. manual locations in the UI.
- Real-time locations refresh periodically; manual locations remain fixed until edited.
- Users may switch from manual → live or live → manual at any time during the session.
- All participants' locations—live or manual—must feed into midpoint calculation equally.

Dependencies

- Requires: **Geolocation API**, **Map provider**, **Address/Geocoding API**
- Used by: **Midpoint Calculator**, **Venue Recommendations**, **Session System**

3.2 Friend Location Visibility

Description

Show all participant locations (live or manually set) in an interactive map view, with clear indicators for each type of location.

Functional Requirements

- Display each participant's chosen location with a labeled pin.
- Use unique visuals: e.g., "Live" badge vs. "Manual" badge.
- Refresh the map when users update their location settings or when live location updates propagate.
- Show participants who have not yet joined or who have not provided a location.

Dependencies

- Requires: **Flexible User Location System (3.1)**
- Feeds into: **Midpoint display**, **Map View**, **Voting interface (distance info)**

3.3 Midpoint Calculator

Description

Calculate the geographic midpoint among all participant locations (live or manual) to identify a fair central meeting point for the group. The midpoint may remain dynamic or be locked by the user for planning and stability.

Functional Requirements

Midpoint Calculation (MVP)

- Accept any combination of live locations and manually set locations.
- Compute geographic centroid using participant coordinates.
- Display midpoint pin prominently on the map.
- Recalculate midpoint automatically if any participant updates their location **when midpoint is in Dynamic mode**.
- Provide fallback handling if a user's live-location temporarily fails (retain last known coordinate).

Midpoint Locking

- Provide a toggle or control allowing users to switch between:
    - **Dynamic Midpoint** (default) – midpoint updates automatically as participant locations change.
    - **Locked Midpoint** – midpoint remains fixed regardless of subsequent location changes.
- Midpoint lock state must be shared and synchronized across the session.
- Only the session host (or all participants—final decision TBD) may toggle lock/unlock.
- When the midpoint is locked:
    - Display a clear visual indicator (e.g., lock icon next to midpoint pin).
    - Suppress automatic recalculation even if multiple participants update locations.
- Unlocking the midpoint triggers an immediate recalculation using the current set of locations.

Dependencies

- Requires:
    - **Flexible User Location System (3.1)**
    - **Friend Location Visibility (3.2)**
    - **Session System (3.6)** to store lock state
    - **Map Provider** for visual indicator
- Used by:
    - **Venue Recommendations Engine (3.4)** (midpoint drives search radius & ranking)
    - **Map View (3.7)** (responds to dynamic vs. locked midpoint changes)

3.4 Venue Recommendations Engine

Description

Suggest venues (restaurants, bars, cafés, activities) near the midpoint, sourced from a third-party API.

Functional Requirements

- Query the venue API centered on the calculated midpoint.
- Pull venue metadata (name, rating, distance, photos, price tier, category).
- Rank results by a combination of **distance + popularity + rating**.
    - Distance can be measured in miles or minutes
- Preload the next several venues to support smooth swiping.
- Auto-refresh venue list when midpoint updates.
- Support future enhancements like filters and category selection.

Dependencies

- Requires: **Midpoint Calculator**, **External Places API** (Yelp/Google), **Internet connectivity**
- Used by: **Swipe Voting Interface**, **Map View**, **Session History**

3.5 Swipe Voting Interface

Description

A Tinder-style swipe interface for voting on venue suggestions.

Functional Requirements

- Swipe right = approve; swipe left = dismiss.
- Only one venue is displayed at a time.
- When a user swipes, their vote must sync to the session in real-time.

- Trigger a “Match” when *all* participants approve the same venue.
- Upon match:
    - Display confirmation modal
    - Reveal venue details
    - Freeze voting and end the session
- Deck must refresh automatically when top card is swiped.

Dependencies

- Requires: **Venue Recommendations**, **Real-time Session System**, **Frontend gesture handling**
- Used by: **Session History**, **Match Result screen**

3.6 Group Session System

Description

A real-time shared session where friends join, set locations, vote, and see live updates.

Functional Requirements

- Create a unique session ID and join link.
- Allow users to join without an account (MVP).
- Track which friends have joined.
- Store each user’s:
    - Live/manual location
    - Voting state
    - Join/leave status

- Sync all session changes in real-time (e.g., with WebSockets or Firebase).
- Provide a simple “Session Overview” showing active participants and vote progress.
- End the session once a venue match occurs.

Dependencies

- Requires: **Real-time database/service**, **Flexible User Location System**, **Swipe Interface**
- Used by: **History**, **Voting**, **Map View**

3.7 Map View (Mobile + Desktop)

Description

Interactive map showing friend locations, manually set locations, recommended venues, and the midpoint.

Functional Requirements

- Desktop layout: split map + venue card panel.
- Mobile layout: map header + card stack below OR tabbed nav.
- Display:
    - Midpoint pin
    - Friend pins (live/manual indicators). If friends have a profile created, display their avatar on their pin (similar to Find My Friends UI).
    - Candidate venue pins. The pin marker should be dimmed or muted for venues that the user has already rejected this session)
- Selecting a pin opens a simple details preview.
- Auto-update when session data changes.

Dependencies

- Requires: **Map provider**, **Locations**, **Venue Recommendations**
- Used by: **Voting Interface**, **Planning UX**

3.8 Profile & Account Settings (Non-critical MVP)

Description

Lightweight user profiles for managing account info and preferences.

Functional Requirements

- Edit username, email, and password.
- View and manage blocked venues.
- View connected accounts (future: Google, Apple, Yelp!, Beli, FourSquare).
- Delete account option.
- Toggle to show/hide live location. If this setting was ON during a session, then toggled off, trigger a prompt: “Use last location or manually set location?” with an input box showing the last known address, which can be edited to update their location to set a manual location.

Dependencies

- Requires: **Authentication**, **Basic user database**
- Used by: **History**, **Preferences**, **Privacy settings**

3.9 Session History & Reviews

Description

Stores previous sessions and allows users to review past matched venues.

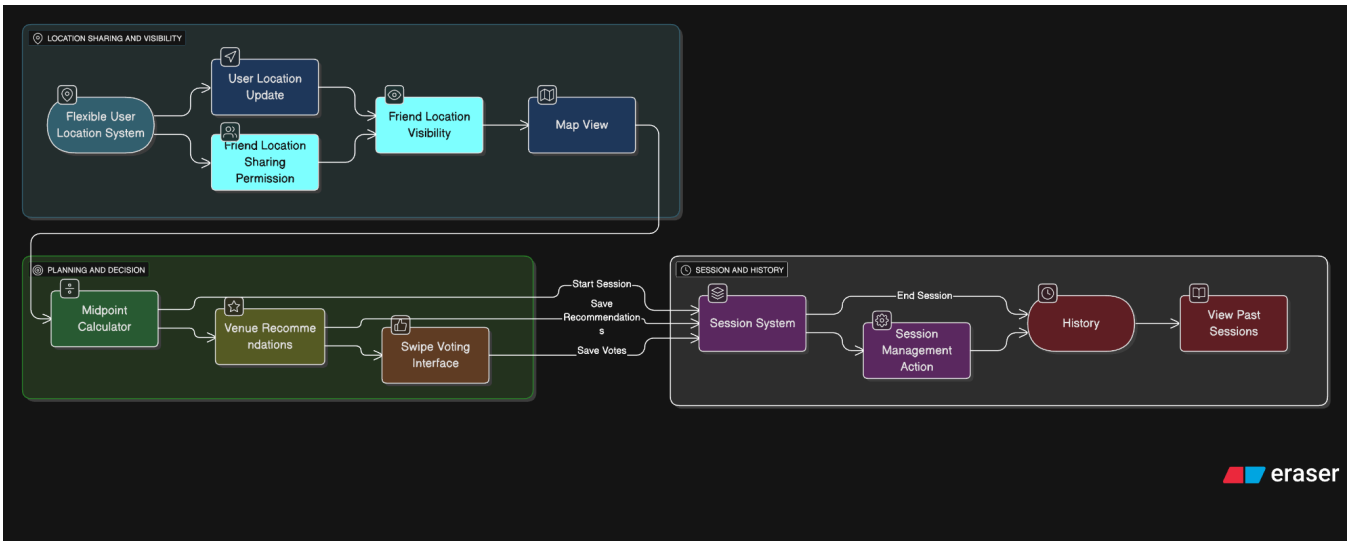
Functional Requirements

- List of previous sessions with date and matched venue.
- Tapping opens a modal with:
    - Venue details
    - 5-star rating input
        - If a user rates a venue as 0 stars, trigger a prompt asking if they want to add this venue to their Blocked Venue list.
    - Short review text (auto-save)
    - Names of group members, excluding self (ex: Abby, Caroline, Sara).
- Option to close via X or tapping outside modal.

Dependencies

- Requires: **Session System**, **Review storage**, **Venue metadata**
- Used by: **Profile**, **Retention features**

3.10 Dependency Diagram



📌 Demo Accounts for Testing (Non-Verified)

To support internal testing, user-flow validation, and QA of session creation, location logic, and voting features, the system must include a set of **pre-created demo user profiles**. These accounts **do not require email verification** and should bypass any verification or onboarding constraints to streamline testing.

Demo User Accounts

Username	Password	Email
Abby	asdf	tester01@a2y.xyz
Maren	asdf	tester02@a2y.xyz
Sara	asdf	tester03@a2y.xyz
Lauren	asdf	tester04@a2y.xyz

Requirements

- These accounts must be seeded in the development and staging environments.
- Email verification should be disabled for these specific test accounts.
- They should behave as standard users for all testing purposes:
    - Join sessions
    - Set live or manual locations
    - Participate in voting
    - Lock/unlock midpoint
    - Review history
- Optional: Add a metadata tag (e.g., `isDemoUser: true`) to clearly identify them in logs.
- These accounts **must not** be included in production analytics unless filtered out.

4. User Interface Requirements

4.1 Core Screens

Home

- Create Session (FAB button)
- Join Session (input code/link)
- List of recent sessions

Map View

Desktop:

- Split-screen view → Map + Venue Card Panel
  Mobile:
- Map above / card stack below OR navigation tabs

Card Stack View

- Featured image
- Venue name
- Tagline
- Yelp rating
- Description
- Distance from user and from midpoint
- Swipe UI with smooth transitions

Profile

- Username, email, password
- Connected accounts
- Blocked venues list

History

- List of past sessions
- Tap → opens modal
    - View winning venue
    - Add a review (5 stars + text)
    - Auto-save

    - Close via X or tapping outside modal

---

5. Visual & Design System Requirements

Style Requirements

- Modern, clean, minimal
- Inspired by Airbnb design system
- Isomorphic UI elements
- Rounded cards, soft shadows, white space
- Floating Action Button for session creation

Card Design

- Large image top
- Name, rating, distance, quick stats
- Swipeable Tinder-style gestures

Map Design

- Pins for:
    - Midpoint
    - Friend locations
    - Venue recommendations
- Smooth animations on selection

---

6. Technical Requirements

6.1 Platforms

- Responsive web app (desktop + mobile browser)
- Potential future native apps (iOS, Android)

6.2 API Integrations

- Geolocation API (browser-based)
- Mapping API (Mapbox or **Google Maps**)
- Places API (**Yelp Fusion** or Google Places)
- Real-time sync (Firebase, **Supabase**, or WebSockets)

6.3 Backend Requirements

- Session creation + management
- User authentication (email/password or social later)
- Real-time room state tracking
- Location storage (ephemeral per session)
- Venue voting algorithm
- Review storage for History section

6.4 Data Model (MVP)

Tables / Collections:

Users

- id
- username
- email
- password (hashed)
- preferences

Sessions

- id
- created_by
- timestamp
- midpoint_lat
- midpoint_long

Participants

- session_id
- user_id
- lat/long
- joined_at

Venues

- session_id

- venue_id
- metadata (name, image, rating, etc.)

Votes

- session_id
- user_id
- venue_id
- vote (yes/no)

History

- session_id
- venue_id
- review stars
- review text

---

7. Non-Functional Requirements

Performance

- Map loads in <1.5s
- Venue cards preload in background
- Real-time updates <300ms latency

Privacy & Security

- Location sharing must be opt-in

- Location data is ephemeral
- Encrypted in transit (HTTPS)
- Option to immediately stop sharing

Scalability

- Able to support sessions of 2–20 participants
- Real-time sync scales to thousands of concurrent rooms

---

8. Success Metrics (KPIs)

Engagement

- % of sessions that reach a match
- Average number of swipes per session
- Average session duration

Activation

- % of users who create a second session
- Number of sessions initiated per user per month

Retention

- Weekly active users
- Number of repeat meetups per user/group

---

9. Future Enhancements (Post-MVP)

- Saved friend groups
- Multi-midpoint routing (e.g., people arriving from different transit lines)
- In-app chat
- In-app reservations through Yelp/OpenTable
- Personalized venue filters
- Gamification (badges for being the “decider,” etc.)
- Native mobile apps