Project Proposal – Team #7 BiteBuddy

BiteBuddy

Project Overview

BiteBuddy is a group restaurant matching mobile app — essentially "Tinder for restaurants" for friend groups. Friends add each other, start a matching session, and swipe left/right on the same stream of nearby restaurants. When everyone in the session likes the same restaurant, the app declares a Match Found and saves the result.

The problem it solves: deciding where to eat with friends is slow and chaotic (group chats, indecision, uneven travel). BiteBuddy streamlines this into a fast, fair flow by searching for restaurants in a convenient shared area (a midpoint/meet-in-the-middle approach) and letting the group converge on a choice through simple swipes.

Project Scope

- Must-Haves (MVP):
  - Supabase Auth: sign up / login.
  - Friends list: add friends (basic) and view friend list.
  - Create / join a matching session with participants.
  - Midpoint discovery: compute a reasonable "in-between" location and fetch candidate restaurants from Yelp API.
  - Swipe queue: each participant swipes left/right on restaurants.
  - Store votes in the database and detect a match when all participants like the same restaurant.
  - Match Found screen + session marked completed (history saved).
  - Basic realtime/polling so all participants see progress and match notification.
- Nice-to-Haves / Stretch Goals:
  - Session filters (price, categories, distance radius) with UI.
  - Better "fairness" midpoint logic (outlier handling) and/or multiple candidate midpoints.
  - Restaurant detail cards (photos, map preview).
  - Invite links / share codes.

    - Session history and "rematch" from previous sessions.
    - Push notifications.
- Non-Goals for MVP:
    - Payments, reviews, deep personalization, complex recommendation systems.

Project Objectives

1) Deliver a functional end-to-end application that supports group sessions, restaurant discovery, swiping, and match detection.
2) Keep the initial implementation minimal and modular to avoid large merge conflicts
3) MVP built by a single assigned member during Week 1/2).
4) Stand up a reliable backend/API layer and database schema that supports realtime session updates.
5) Deploy a working backend (Vercel) and a testable Expo build for internal testing.

Measurable Goals / Success Metrics:

- Restaurant discovery API responds in < 2 seconds for typical queries.
- Match detection updates propagate to all participants within < 1 second (realtime or polling).
- 95%+ crash-free sessions during internal testing.
- At least 5 complete group sessions successfully executed by the team end-to-end (create → swipe → match).

Specifications

User Interface (UI) Design

- Platform: Mobile (React Native + Expo).
- Key screens:
    - Auth (Login/Signup)
    - Home / Friends (add/view friends)
    - Create/Join Session (participants + basic filters)
    - Swipe Session (restaurant cards + like/dislike)
    - Match Found (final selection + session summary)
    - Session History (basic list of completed sessions)
- Interactions: buttons for create/join/invite, swipe gestures, filter form inputs, progress indicator (e.g., "3/5 liked").

Backend & APIs

- Backend: Next.js API routes (serverless) deployed on Vercel.
- Database: Supabase Postgres.
- Core tables: users, friends, sessions, session_participants, restaurants (cached), votes, matches.
- Auth: Supabase Auth (JWT-based). Row Level Security (RLS) policies to restrict session/vote access to participants.
- External APIs: Yelp Fusion API for restaurant search. (Optional) Google Maps/Places for geocoding/midpoint validation.

Data & AI Model (Dataset Source and Model Architecture)

- AI in MVP: lightweight LLM-powered enrichment to improve the swipe experience (e.g., generate 1–2 sentence summaries, highlight pros/cons, or normalize tags from Yelp data). This is intentionally minimal and can be disabled without breaking core functionality.
- Data source: Yelp API responses (name, rating, price, categories, coordinates, photos, URL).
- Preprocessing: field normalization + caching in Postgres; optional prompt templating for consistent summaries.
- Model/API: hosted LLM (e.g., OpenAI GPT or equivalent) called from backend with strict rate limits and caching to control cost/latency.

Tech Stack

- Frontend: React Native + Expo (EAS)
- Backend: Next.js (API routes / serverless functions) on Vercel
- Database: Supabase (Postgres)
- Auth & Realtime: Supabase Auth + Realtime channels (or polling fallback)
- AI/ML: Hosted LLM API for restaurant-card enrichment (minimal MVP usage)
- Cloud & Hosting: Vercel (backend), Expo EAS (mobile builds)

Hardware Requirements

- Development laptops/desktops capable of running Node.js, Expo, and Android Studio / Xcode (if applicable).
- At least 2 physical test devices (1 iOS, 1 Android) recommended for swipe/realtime testing.

Software Requirements

- Node.js (LTS), npm/yarn/pnpm
- Expo CLI + EAS CLI
- Next.js / TypeScript toolchain
- Supabase CLI (optional but recommended)
- Git + GitHub (PRs, rulesets, Projects)
- Postman/Insomnia for API testing
- Figma (design)

Project Timeline (Gantt Chart or Milestone Table).

Required: Provide a Phase-based schedule with specific tangible deliverables for each phase.

Include specific deadlines for each task. You can add further categories if needed.

Phase	**Duration**	**Tasks**			**Status/Deliverable**
		Front end	**Back end**	**General**	
Phase 1	02/13/2026 - 02/20/2026	Deliverable: Architecture diagram + working project skeleton. Note: Week 1 MVP built by a single assigned member to reduce merge conflicts.			Architecture Diagram, Minimal MVP
Phase 2	02/20/2026 - 03/06/2026	Deliverable: Figma/Wireframes + API contract + DB schema migrated in Supabase.			UI/UX Design, Configured DB
Phase 3	03/06/2026 - 03/20/2026	Deliverable: Functional Alpha (MVP) end-to-end: create/join → swipe → match.			Functional Alpha (MVP).
Phase 4	03/20/2026 - 04/03/2026	Deliverable: Beta build + QA test report (test cases, issues, fixes).			E2E Test results, additional improvements
Phase 5	04/03/2026 - 04/17/2026	Deliverable: Final demo + deployed backend + GitHub handover package.			Final Demo & GitHub Handover.

Team Leader Rotation

Duration	Team Leader
02/13/2026 - 02/20/2026	Jonathan Lee
02/20/2026 - 03/06/2026	Rafid Nasery
03/06/2026 - 03/20/2026	Abdala Aljiewarane
03/20/2026 - 04/03/2026	Neon Zheng
04/03/2026 - 04/17/2026	Paola Isabel
04/17/2026 - End of Project	Billy Nguyen

Project Team

Role	Team Member	Responsibilities
Tech Lead / Architecture	Neon Zheng	Creating architecture diagrams, end-to-end flows
Backend Developer	Abdala Aljiewarane	Next.js API routes, Supabase schema/RLS, Yelp integration, session + vote logic.
Yelp Integration	Jonathan lee	Yelp integration, session + vote logic. Working on a specialized segment of backend
UI/UX Frontend	Paola Isabel-Reyes	User stories, wireframes/Figma, acceptance criteria, demo script.
Mobile Setup/Testing	Billy Nguyen	Concurrent testing on mobile device environments, catch UI/UX bugs and sometimes fix them
DevOps / Github PM	Rafid Nasery	Github setup, CI/CD checks, Vercel and Expo deployment, Supabase setup, Task creation and assignment

Links

- GitHub Repository: https://github.com/BiteBuddy-CS4485/BiteBuddy
- Agile Board (GitHub Projects/Jira/Trello): https://github.com/orgs/BiteBuddy-CS4485/projects/1
- Design Document (Figma/Canva/etc.): Leaving up to the assignee of design tasks