In21-S6-CS3940 - Professional Portfolio

Final Portfolio Report - Group 29

Group Members

210167E Fernando T.H.L.

210176F Gamage M.S.

210339J Lishan S.D.

210382H Meddawitage K.C.D.

210588U Senarathna L.P.S.U.K.

1. Table of Contents

2. Executive Summary

2.1 Overview of the Application

GoTogether is a comprehensive travel management solution designed specifically for Sri Lanka, comprising both mobile and web applications. The platform addresses critical gaps in travel planning and transportation information that affect both international tourists and local travelers. By providing personalized route planning, real-time travel support, and cultural insights, GoTogether serves as an intelligent travel companion that transforms the way people explore Sri Lanka.

2.2 Problem Statement

Tourists visiting Sri Lanka frequently encounter significant challenges including language barriers, inefficient route planning, and lack of accessible transportation information. These difficulties result in wasted time, increased travel costs, and suboptimal travel experiences. Similarly, Sri Lankan travelers face comparable obstacles due to the absence of optimized trip planning tools tailored to local conditions. The current travel landscape is characterized by fragmented information sources, manual planning processes, and limited access to real-time transportation data, creating a clear need for an integrated, intelligent travel solution.

2.3 Target Users

GoTogether is designed to serve two primary user segments:

Foreign Travelers: International tourists visiting Sri Lanka who require comprehensive travel guidance, cultural context, and language support to navigate the country effectively and safely.

Local Travelers: Sri Lankan residents seeking efficient trip planning, optimized routes, and access to comprehensive transportation information for domestic travel within the country.

2.4 Key Features

The application delivers six core functionalities designed to address specific travel pain points:

Personalized Tour Planning: Enables users to create customized itineraries by selecting starting points and destinations, with intelligent route optimization and personalized recommendations.

Transport Details: Provides comprehensive information about available transportation options between destinations, including schedules, fares, and real-time availability.

Travel Guidance: Offers curated recommendations for cultural sites, entertainment venues, and hidden attractions, enhancing the discovery aspect of travel.

Trip Review & Travel Feed: Creates a social platform where users can share trip plans, engage through comments and reactions, and follow other travelers for inspiration and insights.

Emergency Support: Ensures traveler safety by providing quick access to essential services including hospitals and police stations during emergencies.

Food/Stay Suggestions: Delivers location-based recommendations for accommodations and restaurants tailored to specific travel destinations and user preferences.

2.5 Project Outcomes

GoTogether successfully addresses the identified travel challenges by creating a unified platform that reduces unplanned expenses through local insights and cost-effective suggestions. The application overcomes transportation information gaps and language barriers while providing personalized routes and real-time travel support. The project delivers measurable value by streamlining the travel planning process, reducing

travel-related stress, and enhancing overall travel experiences for both international and domestic users in Sri Lanka.

3. Introduction

3.1 Motivation for the Application

The motivation for developing GoTogether stems from the recognition that Sri Lanka's tourism industry and domestic travel sector lack a comprehensive, technology-driven solution that addresses the unique challenges faced by travelers. Despite Sri Lanka's rich cultural heritage and natural beauty, visitors often struggle with fragmented information sources, language barriers, and inefficient transportation systems. The absence of a locally-focused, intelligent travel platform creates opportunities for enhanced travel experiences and improved tourism outcomes.

3.2 Importance of Smart Travel Planning

Smart travel planning has become essential in the modern travel landscape, where travelers expect personalized, efficient, and real-time solutions. Effective travel planning reduces costs, minimizes time wastage, enhances safety, and improves overall satisfaction. For a destination like Sri Lanka, with its diverse attractions and complex transportation network, intelligent planning tools can significantly impact travel quality. Smart planning also contributes to sustainable tourism by optimizing resource utilization and reducing environmental impact through efficient route planning.

3.3 Project Objectives

The primary objectives of the GoTogether project are to develop a comprehensive travel management platform that eliminates language barriers and transportation information gaps for travelers in Sri Lanka. The project aims to provide personalized, real-time travel support that reduces unplanned expenses and enhances travel experiences through local

insights and cost-effective recommendations. Additionally, the platform seeks to create a collaborative travel community where users can share experiences and support fellow travelers.

3.4 Scope and Limitations

Key limitations include restricted access to government transport fare data, requiring manual data extraction and validation processes that may affect real-time accuracy.

The project also faces challenges with limited access to proprietary APIs from major third-party platforms such as Uber and Booking.com, preventing direct real-time booking integration within the application.

3.5 Stakeholders

The project involves multiple stakeholder groups including international tourists seeking comprehensive Sri Lankan travel guidance, local travelers requiring efficient domestic trip planning tools, tourism industry operators who benefit from increased visitor satisfaction, and government tourism authorities interested in promoting sustainable and organized travel within the country.

3.6 Challenges in Existing Solutions

Analysis of existing travel applications reveals significant gaps in addressing Sri Lankan travel needs, highlighting the necessity for GoTogether's specialized approach.

Tikalanka: Represents the only Sri Lanka-focused solution but operates through a manual, website-based custom itinerary form system. The platform lacks interactive capabilities and real-time functionality, requiring users to submit requests manually without live planning support. The absence of app-based route guidance makes it unsuitable for dynamic travel situations where travelers need instant access to information and route adjustments.

TripIt: Despite achieving over 10 million downloads globally, focuses exclusively on itinerary organization from email confirmations rather than active trip planning. While

effective for managing pre-booked travel arrangements, it provides no local cultural context or real-time guidance specific to Sri Lanka's unique transportation landscape. The platform cannot facilitate planning or customization of multi-stop local trips, limiting its utility for comprehensive Sri Lankan travel experiences.

Wanderlog: Offers collaborative trip planning features with maps and notes, accumulating over 1 million downloads. However, its generalized global approach fails to incorporate local language support or Sri Lanka-specific data insights. Users report inability to access basic transportation information such as train schedules and routes within Sri Lanka, demonstrating the platform's inadequacy for local travel requirements.

RoutePerfect: Provides AI-powered trip planning capabilities but restricts its focus to European destinations and popular international tourist regions. The platform offers no support for Sri Lankan destinations and fails to account for regional transportation options or local cultural recommendations that are essential for effective travel planning in Sri Lanka.

4. Feasibility and Need Assessment

4.1 Market Demand for Travel Planning Tools

Travel has evolved post-pandemic, with a growing demand for digitally enabled, self-guided, and personalized travel experiences. Tourists increasingly avoid generic packages in favor of tools that help them customize their own journeys. In Sri Lanka, both international and local travelers face route inefficiencies, poor transport data, and language barriers. There is clear market potential for a smart travel assistant that can address these needs. Google Trends, app store analytics, and the rise of platforms like Rome2Rio and TripIt show strong user interest in such tools.

4.2 SWOT Analysis

Strengths	Weaknesses
AI-powered personalized itineraries	Limited access to real-time government fare data
Integration with booking platforms	Difficulty in integrating third-party APIs like Uber
Emergency support features	Reliance on manual data collection for transport info
Travel feed for user engagement	Initial infrastructure and marketing costs

Opportunities	Threats
Rise in inbound tourism in Sri Lanka	Competition from global apps like TripIt or Wanderlog
Expansion to other regions	Data privacy & legal constraints
Collaboration with local travel agencies and tourism boards	Resistance to adoption from non-tech-savvy users

4.3 Cost-Benefit Analysis for Users

Costs for a User

Type	Costs	Benefits
Monetary	$2.99/month for PRO version subscription.	Saves money by avoiding inefficient routes and overpriced accommodations.
	No cost for Free version, but includes ads.	Helps choose cost-effective transport options with fare estimations.

Non-Monetary	Time required to learn and navigate the app.	Personalized AI-based tour planning for preferences (e.g., heritage, wildlife, beaches).
	Mobile data consumed by maps, real-time updates, and AI services.	Access to real-time transport info (bus/train schedules and fares).
	Ads shown in the Free version may reduce experience.	Collaborative planning with friends share and edit trips together.
	User data like location/preferences is collected, raising privacy concerns.	Hotel and restaurant recommendations with booking and review options.
	Reliance on the app may cause issues if offline or experiencing technical problems.	Emergency support with quick links to hospitals, embassies, and police.
		Built-in language translation for overcoming communication barriers.
		Better user experience: dark mode, responsive layout, and smooth navigation.

		Cultural insights and discovery of lesser-known attractions enhance the travel experience.

Conclusion

For a user, the "GoTogether" application presents significant benefits, primarily in convenience, time-saving, and potential cost reduction through efficient planning and informed choices. The non-monetary benefits, such as personalized experiences, language assistance, and emergency support, greatly enhance the overall travel experience.

While there are minor monetary costs for the PRO version and non-monetary costs like data usage and exposure to ads (for the free version), these are generally outweighed by the comprehensive features and the value derived from a streamlined and guided travel experience. The app appears particularly beneficial for both local and international travelers in Sri Lanka who currently face challenges with planning and information access.

4.4 Related Work & Competitive Apps

The proposal evaluated several existing apps to identify market gaps:

1. TripIt (10M+ Downloads)
    - Strength: Automatic itinerary organization from emails.
    - Limitation: No real-time or localized planning; not useful for exploring Sri Lanka’s local sites.

2. Wanderlog (1M+ Downloads)
    - Strength: Collaborative trip planning with maps and notes.
    - Limitation: No local language support, no SL-specific transit or cultural data.

3. RoutePerfect

- Strength: AI-powered trip planning for Europe.
- Limitation: Doesn't support Sri Lankan destinations or transport data.

4. Tikalanka
    - Strength: Focuses on SL tourism.
    - Limitation: Not interactive, lacks mobile experience, and needs manual itinerary submission.

None of the existing competitors offer real-time, personalized trip planning with collaborative, culturally contextual features for Sri Lanka. Your app directly targets these unmet needs.

4.5 Legal and Ethical Considerations

1. Data Privacy and Security:
   Compliance with global standards like GDPR is crucial. Users' location, preferences, and emergency contacts must be securely stored and encrypted. Authentication through OAuth2 (Keycloak) ensures secure access.

2. Content Moderation:
   The trip feed and comment system must have moderation filters to detect abusive content or misinformation. AI-based moderation or admin approval may be required.

3. Fair API Use:
   All third-party APIs (Google Maps, Booking.com, Gemini, ChatGPT-4o) must be legally integrated, respecting their usage limits and licensing policies.

4. Inclusivity and Accessibility:
   The app must support multilingual features and ensure interface accessibility (e.g., font sizes, screen reader support) for differently-abled users.

5. Emergency Features:
   Ethical responsibility requires ensuring that emergency contacts are always accurate and region-specific.

5. Requirement Analysis

5.1 Functional Requirements

1. User Registration & Authentication
2. Trip Planning Engine: AI-based personalized itineraries
3. Collaborative Planning: Users can create, share, and edit trips with friends.
4. Transport Module: Show bus/train/hired vehicle routes, schedules, and fare estimates.
5. Accommodation & Restaurant Suggestions: Integrated booking links and local reviews.
6. Trip Feed: Social media-style posting, commenting, liking, and following.
7. Emergency Services: Quick access to hospitals, police stations, embassies.
8. Language Assistance: Text and voice translation for key travel phrases.
9. Push Notifications: Updates on weather, itinerary changes, transport alerts.

5.2 Non-Functional Requirements

1. Scalability: Cloud deployment on AWS and Vercel ensures scalability
2. Performance: Fast API responses, optimized map rendering, pagination to reduce load times.
3. Reliability: Redundant infrastructure using Docker, CI/CD (GitHub Actions), and Supabase backup.
4. Security: OAuth2 login, encrypted user data, HTTPS routing via NGINX and ACM.
5. Maintainability: Modular microservice backend allows easy updates and debugging.
6. Responsiveness: Mobile-first design with responsive layout for tablet and desktop.

7. Localization: Language support for English, Sinhala, and Tamil with future expansion.

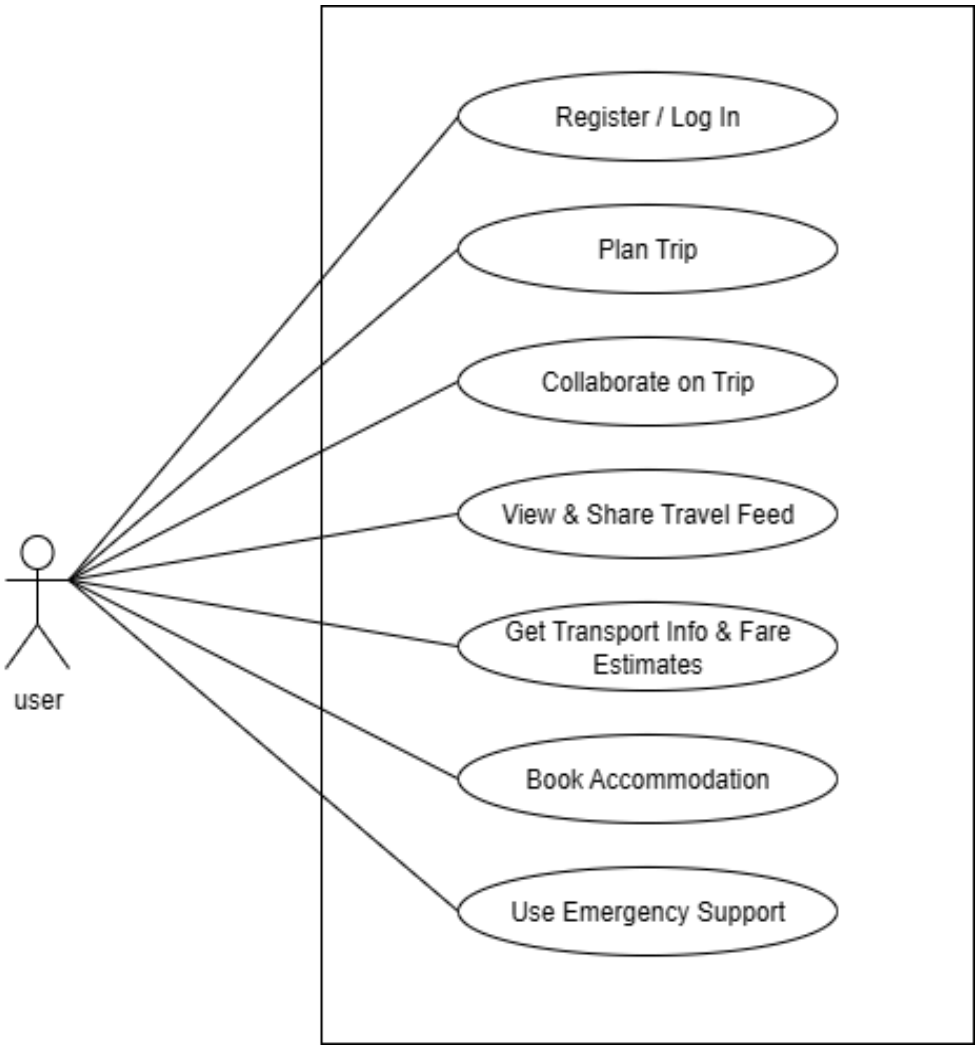
5.3 User Personas

1. Emily – The Solo Tourist
    - Age: 30, from the UK
    - Wants to explore SL safely and affordably
    - Needs emergency contacts, cultural tips, and language help
    - Benefits from AI-planned itineraries and easy booking

2. Nimal – Local Weekend Explorer
    - Age: 24, lives in Colombo
    - Plans weekend trips with friends
    - Uses collaborative trip planning to coordinate with group
    - Needs transport cost estimations and budget-friendly food

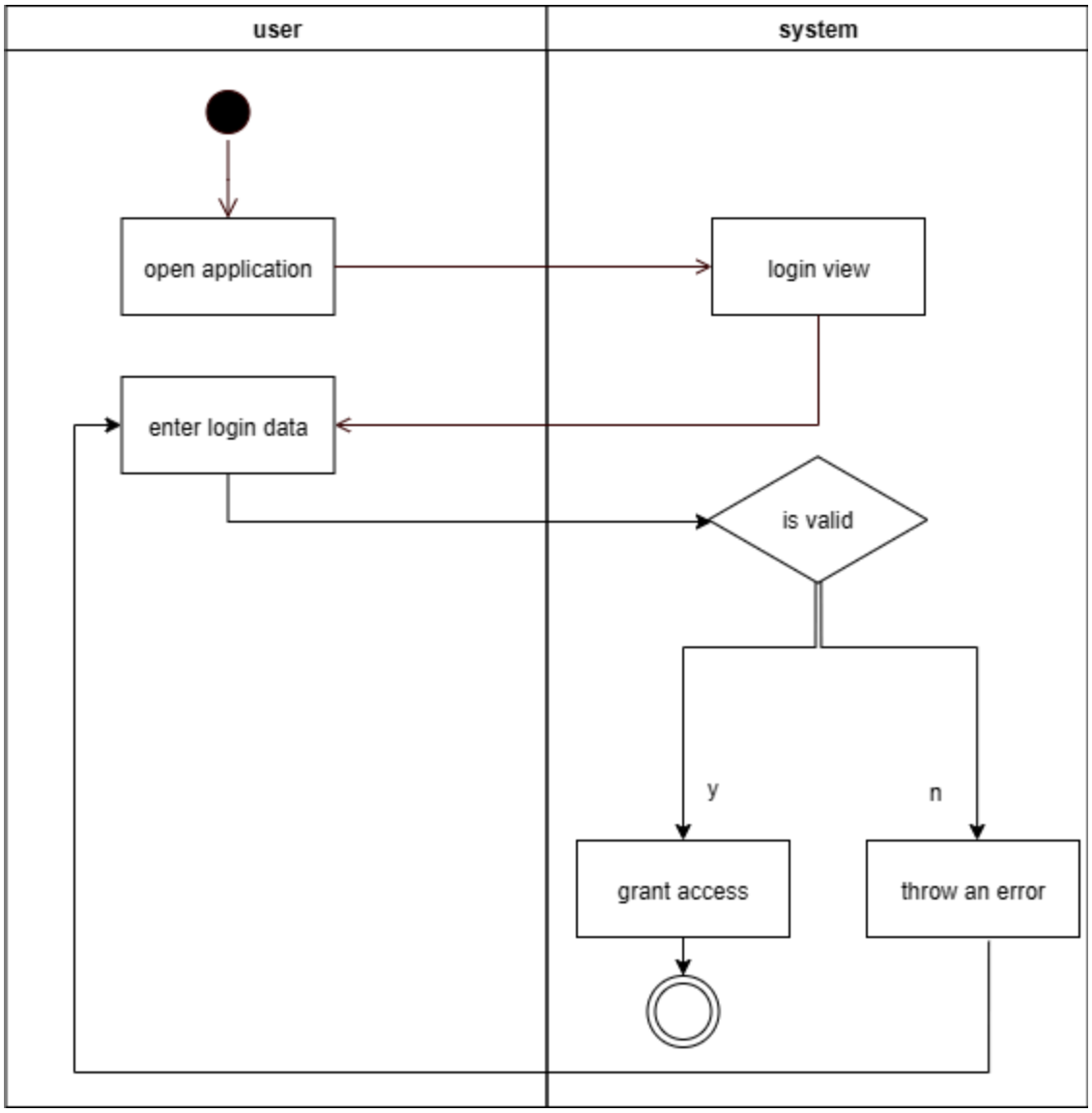
3. Sara – Travel Blogger
    - Age: 27, active on Instagram
    - Shares her trips with followers
    - Uses the app to generate, review, and post itineraries
    - Benefits from integration with real-time data and user-generated reviews

5.4 Use Case Diagram

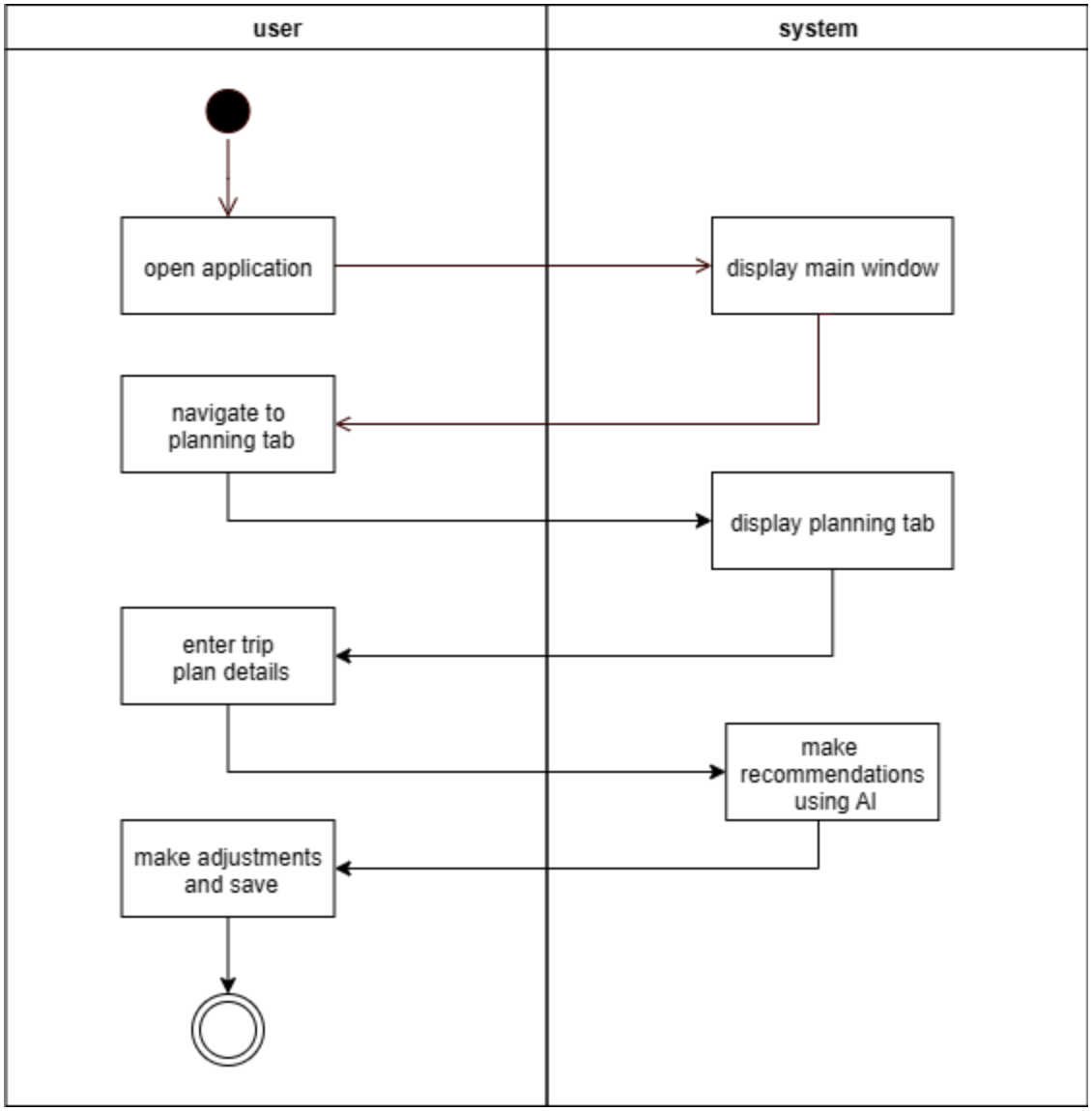


5.5 Activity Diagrams

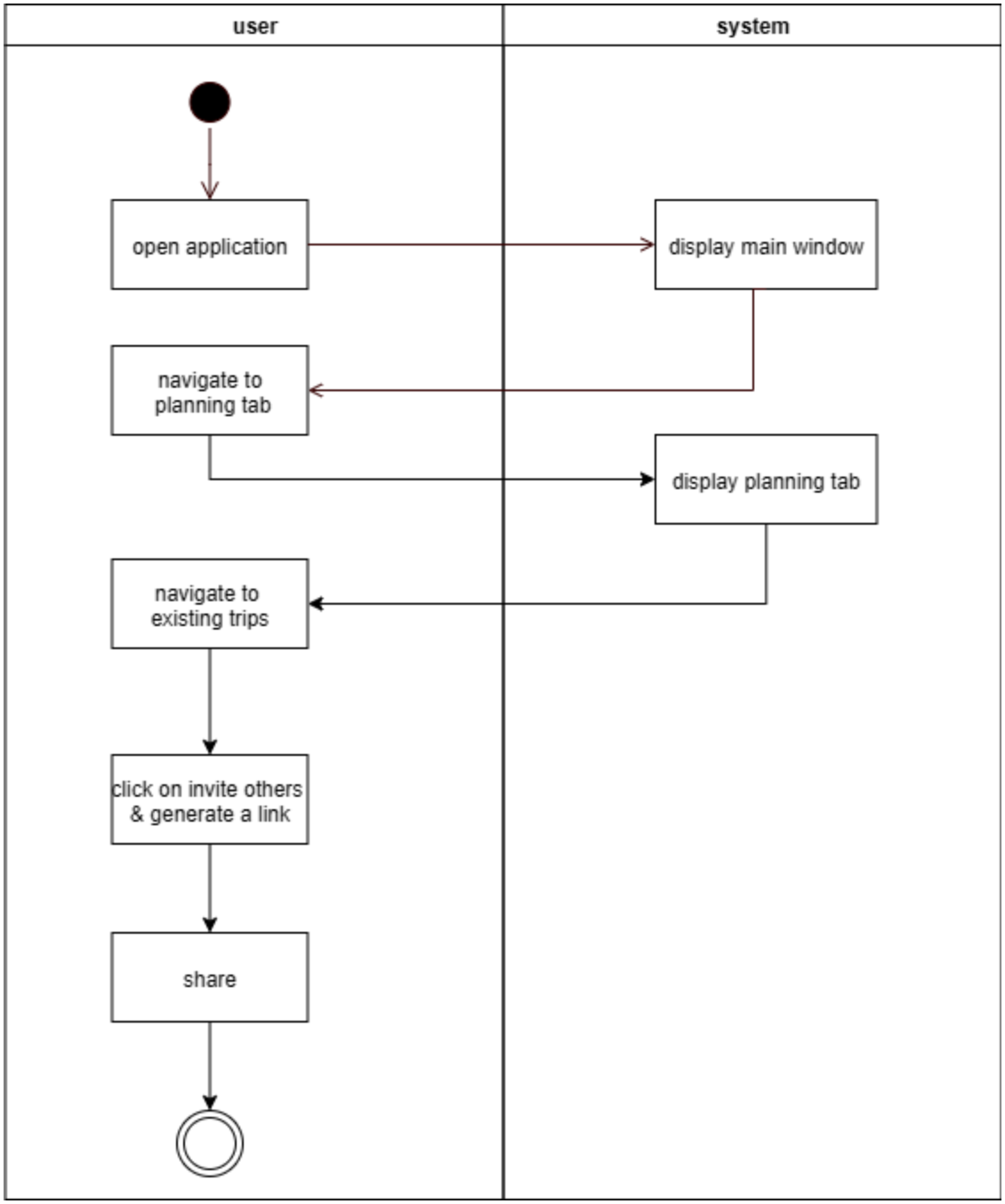
1. Login



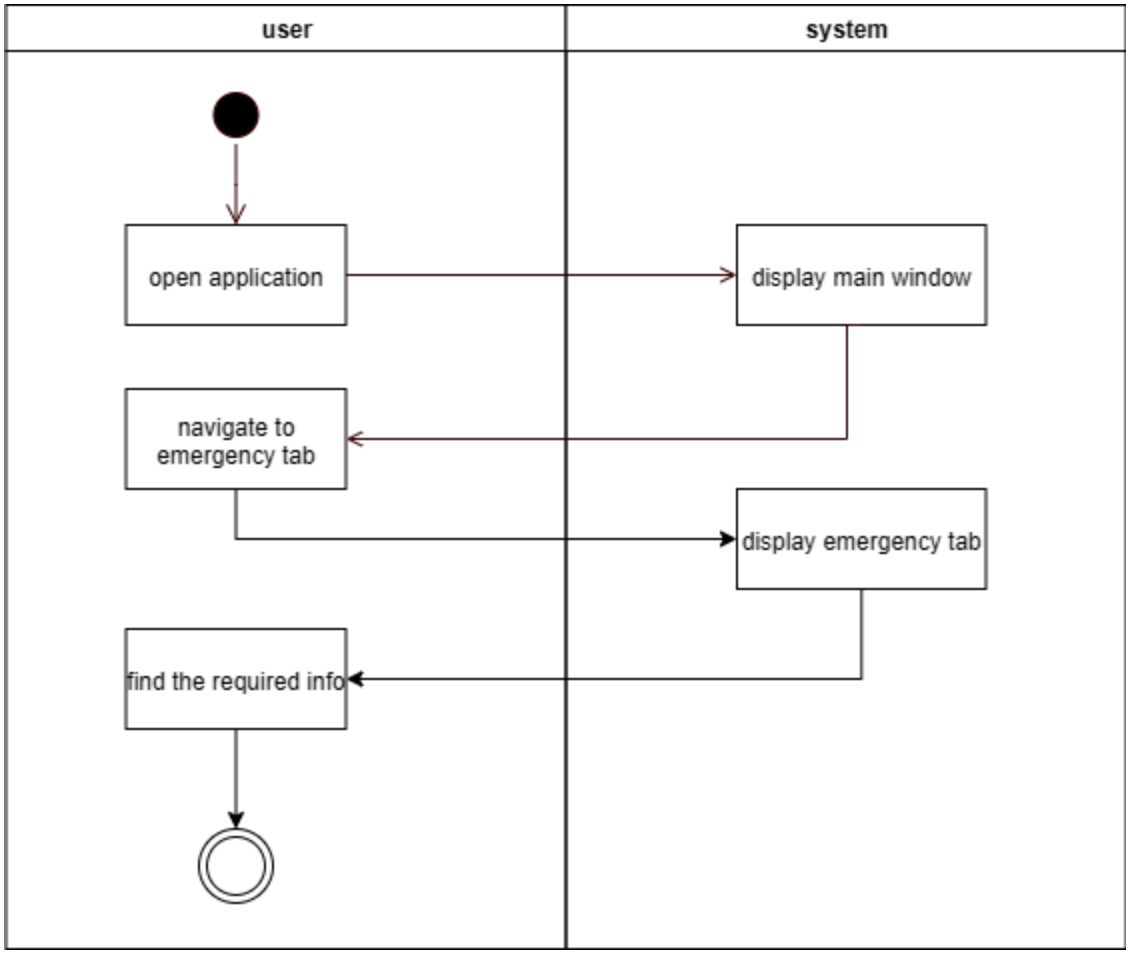
2. Plan a Trip


user
system
open application
display main window
navigate to planning tab
display planning tab
enter trip plan details
make recommendations using AI
make adjustments and save

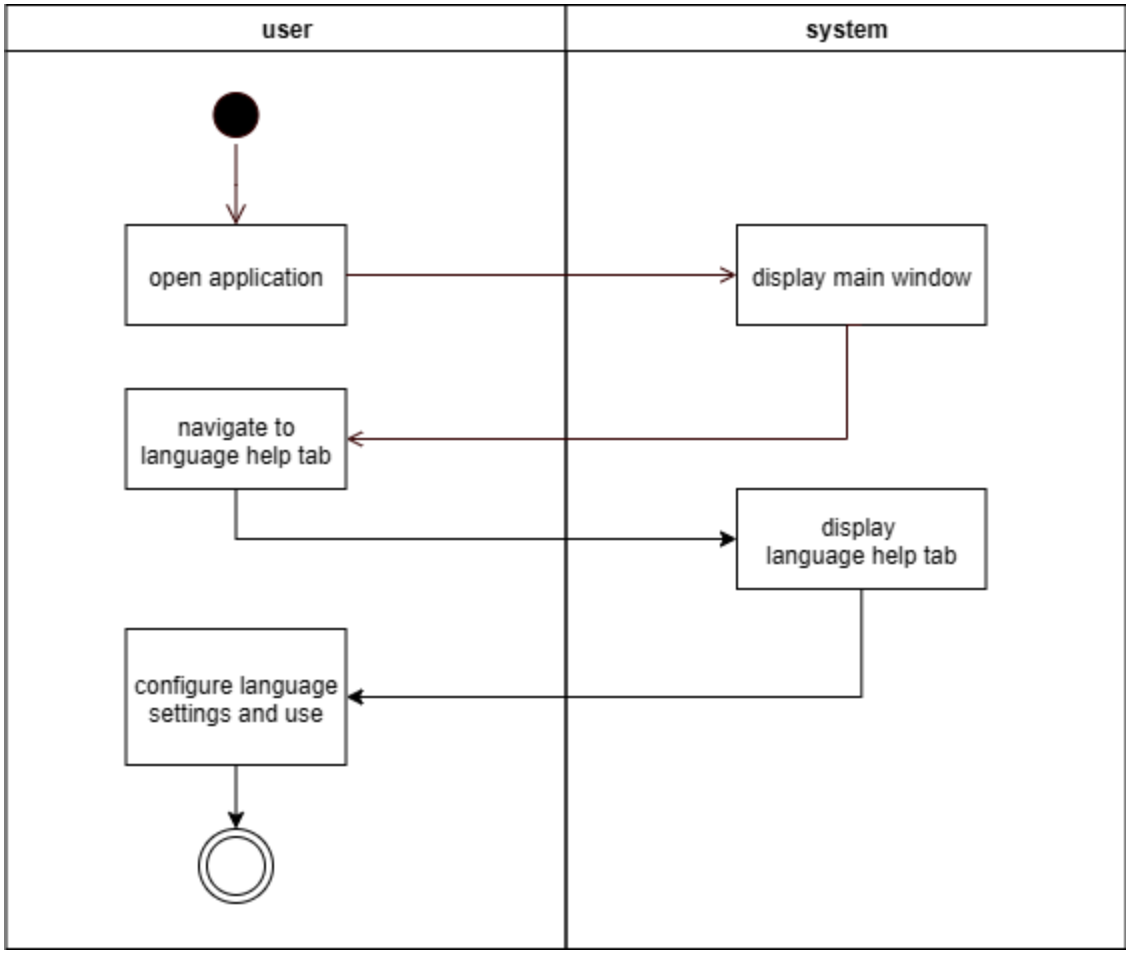
3. Collaborate with friends



4. Emergency help



5. Language assist

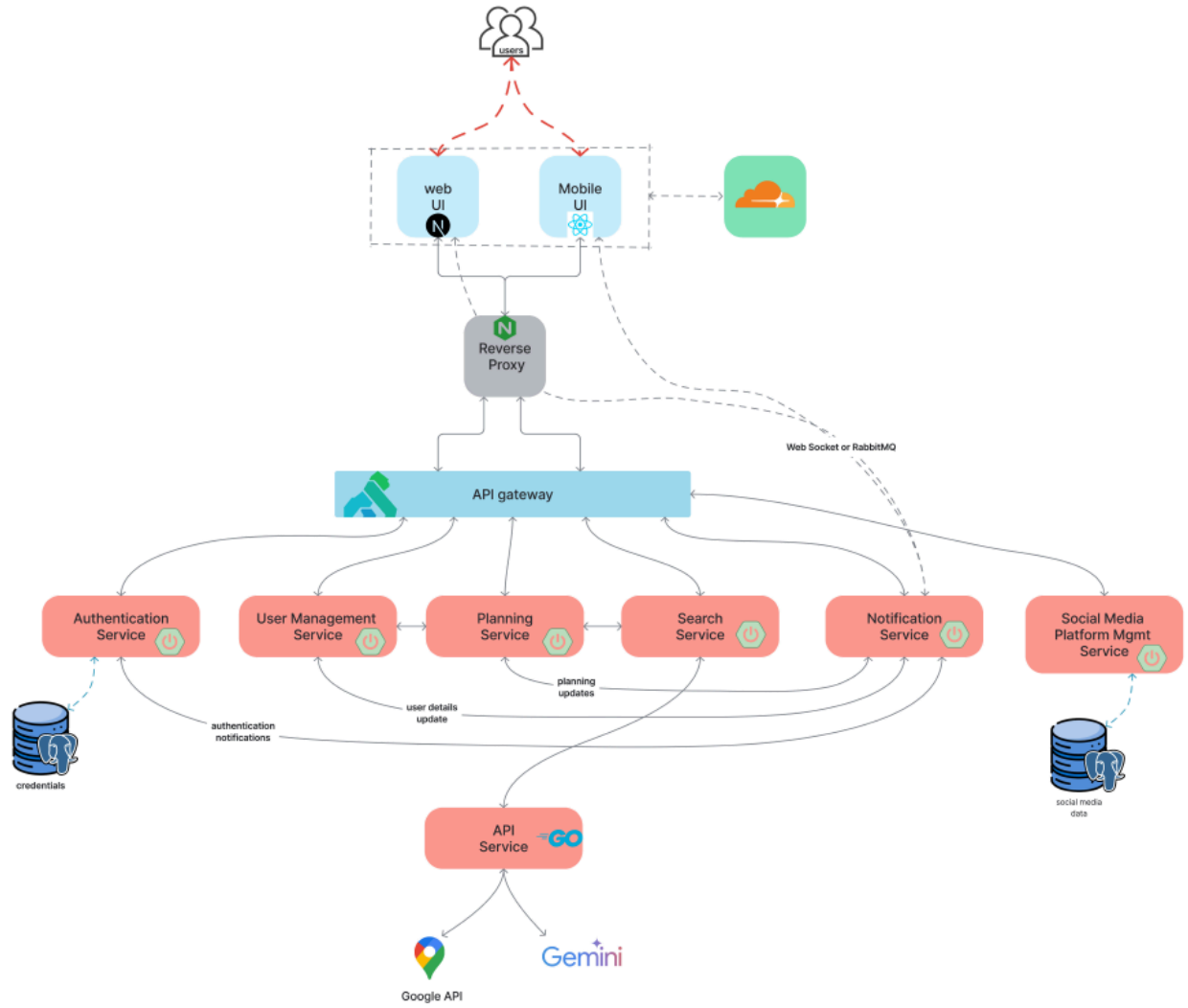


5.6 Data Requirements

1. User Data: ID, preferences, planned trips, history, authentication tokens
2. Trip Data: Start/end locations, itinerary, participants, posts
3. Transport Data: Routes, schedules, fare charts for bus/train/taxi
4. Accommodation & Restaurant Data: Ratings, prices, availability, links
5. Emergency Contacts: Geo-tagged list of hospitals, police stations, embassies
6. Language Data: Common phrases, AI translation models
7. Geospatial Data: Coordinates, maps, distances (Google Maps/Supabase)

6. System Design and Architecture

6.1 High-Level Architecture



The system adopts a distributed microservices architecture, ensuring scalability, modularity, and cloud-native capabilities. It cleanly separates concerns across frontend, backend, AI services, and third-party integrations, using a blend of managed platforms and infrastructure-as-a-service components.

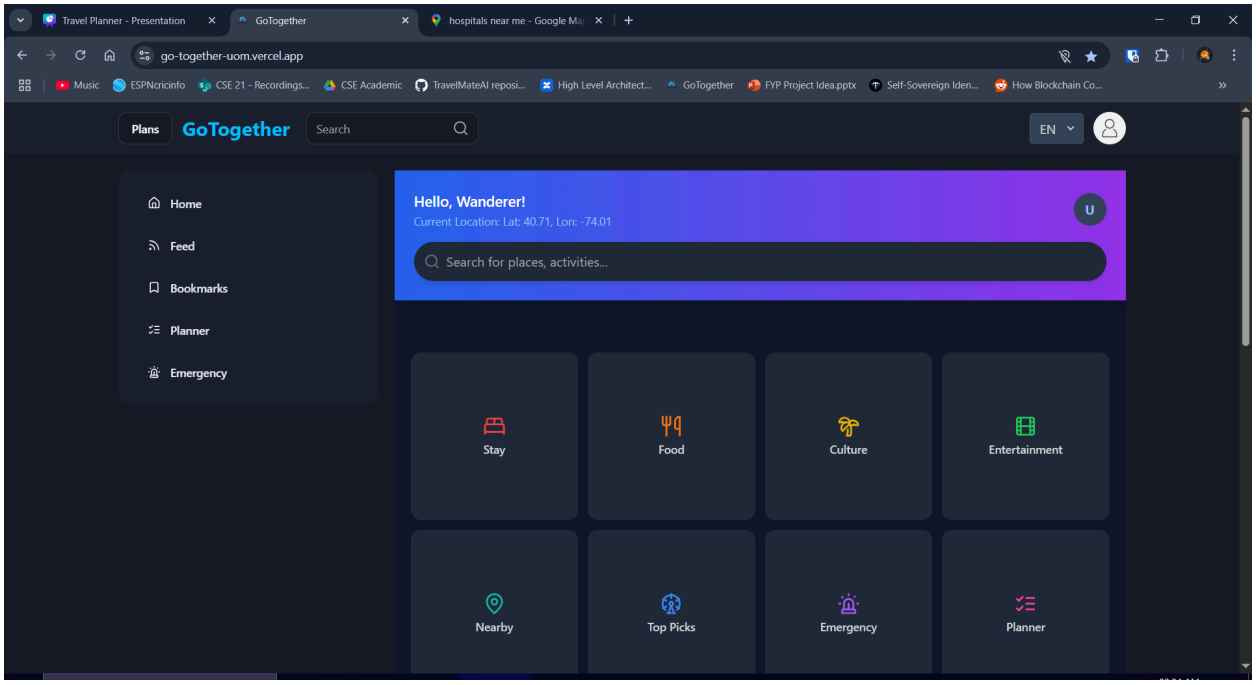
Key Components

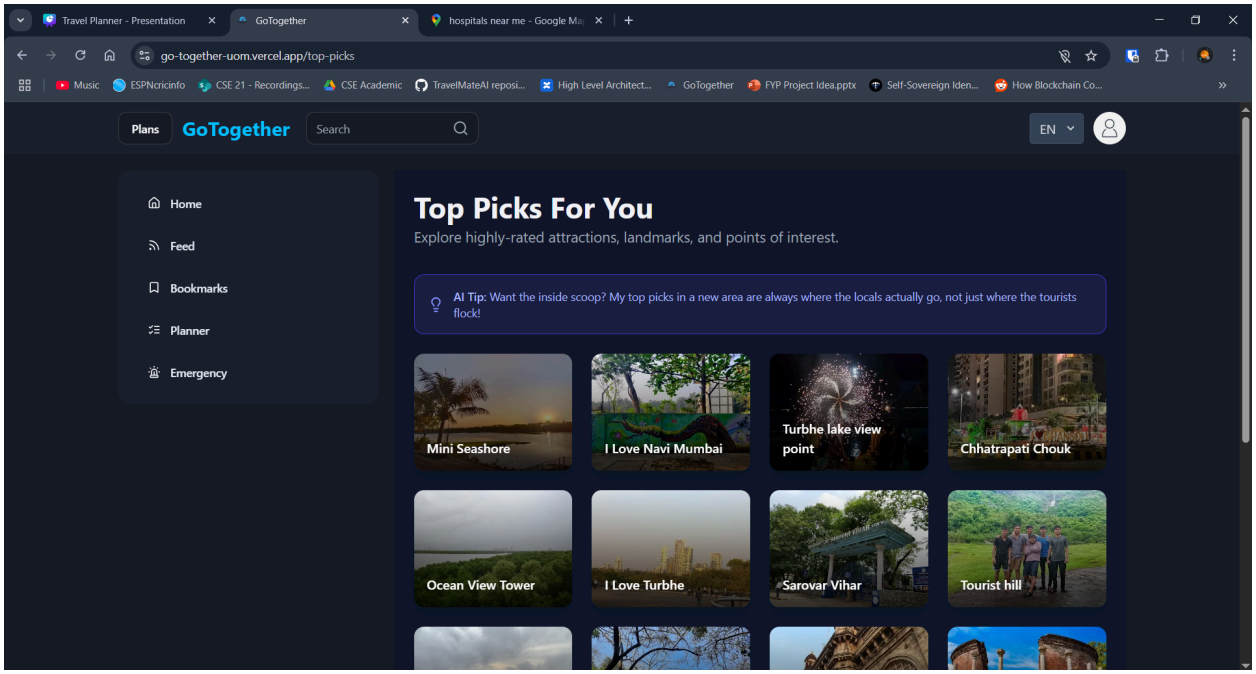
- **Frontend**
    - Web App: Built with Next.js, deployed on Vercel for global scalability and fast content delivery.
    - Mobile App: Developed using React Native, sharing logic with the web stack where possible.
- **Backend**
    - Composed of Dockerized microservices using Spring Boot and Go.
- **API Gateway**
    - Kong Gateway handles all inbound traffic.
    - Provides routing, rate limiting, service discovery, and delegated authentication.
- **Authentication**
    - Keycloak manages user login, registration, sessions, and OAuth2/OpenID Connect flows.
    - Integrated with Kong for secure token-based access control.
- **Database**
    - Supabase PostgreSQL stores persistent data including users, trips, social metadata, and preferences.
- **Image CDN**
    - Cloudflare Images stores and serves user-uploaded media such as profile pictures and post images, with optimizations for delivery speed and bandwidth.
- **DevOps & CI/CD**
    - GitHub Actions automates the build, test, and containerization pipeline.
- **Communication**
    - Internal services communicate via REST and gRPC, depending on performance and data structure needs.

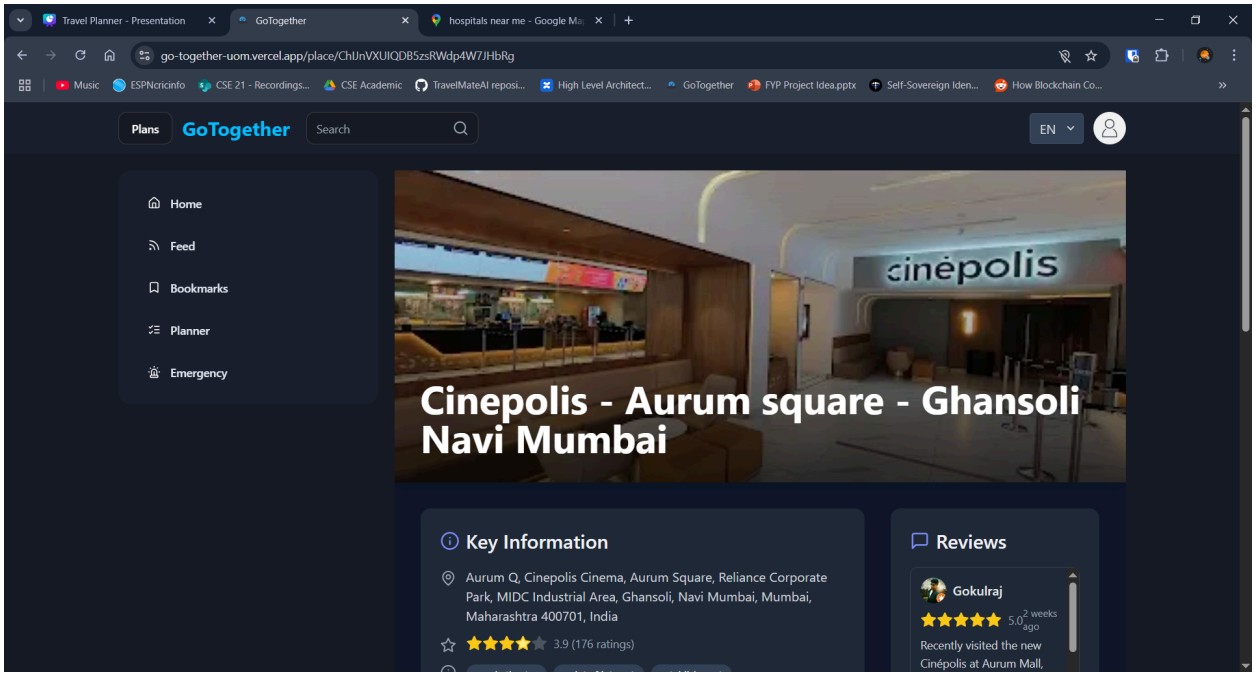
- **Reverse Proxy**
    - NGINX serves as an optional HTTPS reverse proxy, securing access to backend services and enabling SSL termination when needed outside the EKS environment.

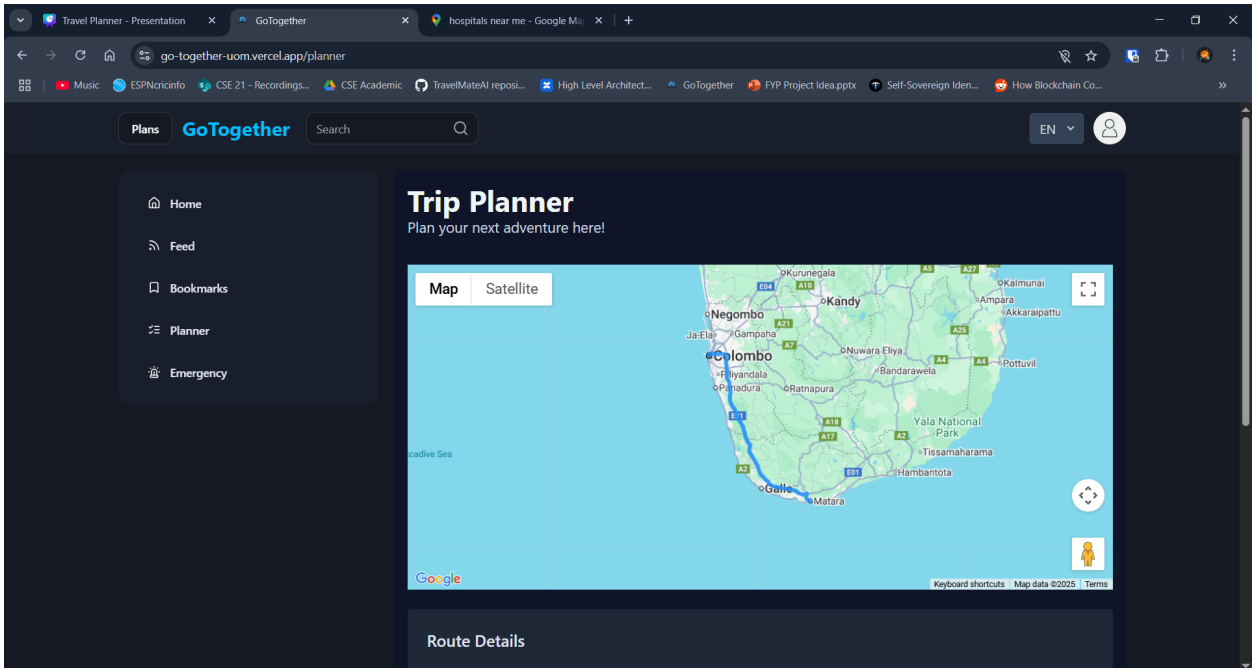
6.2 Web Frontend Design

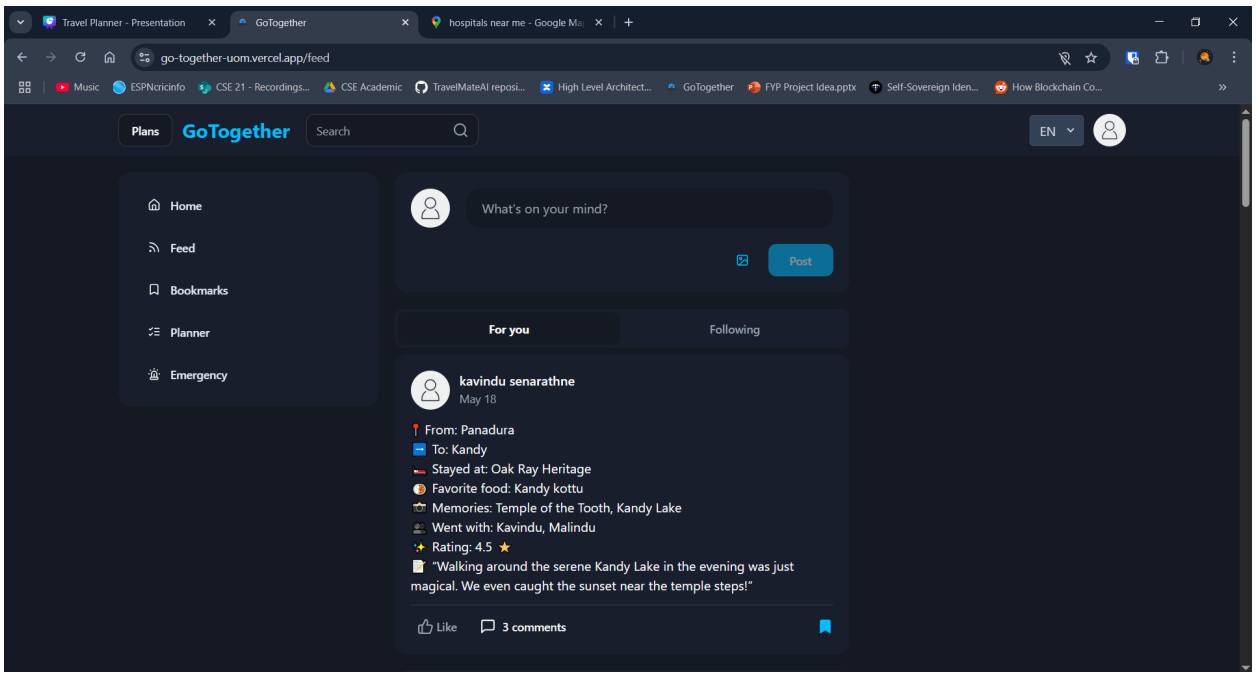
The web frontend is developed using Next.js, offering a modern, fast, and SEO-friendly user experience. It leverages a hybrid rendering strategy (SSR + CSR) and built-in features for performance and accessibility.

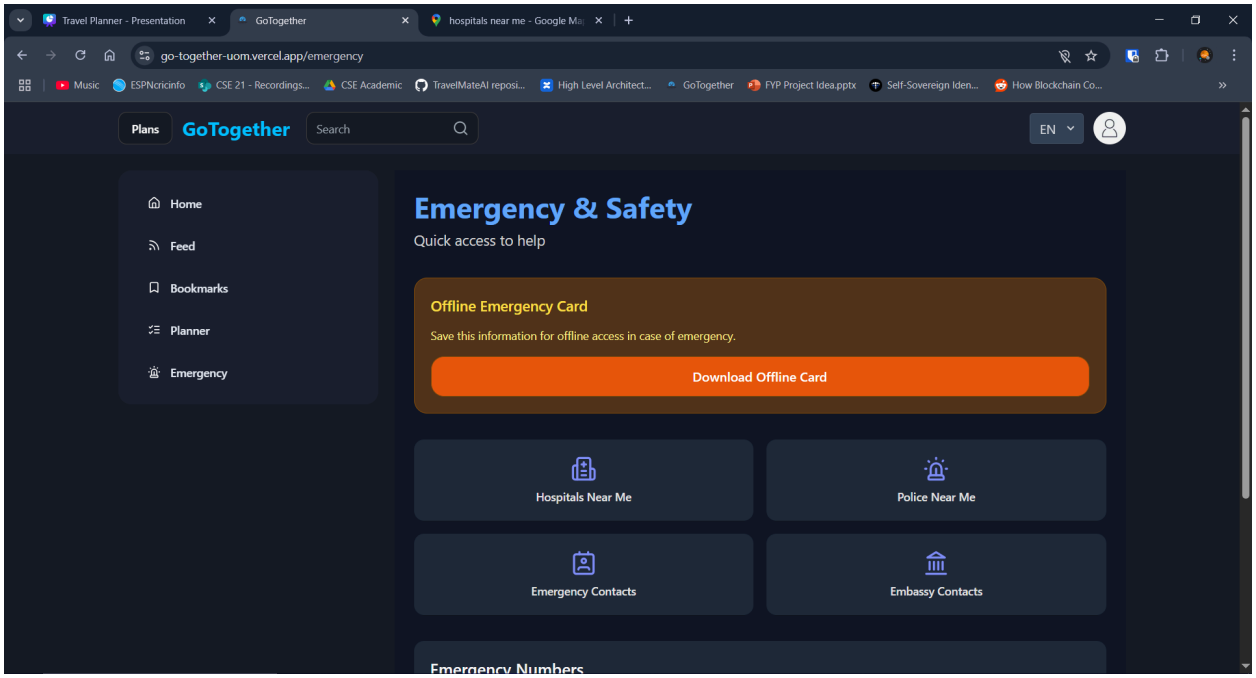



Travel Planner - Presentation
GoTogether
hospitals near me - Google Ma
go-together-uom.vercel.app/top-picks
Plans
GoTogether
Search
EN
Home
Feed
Bookmarks
Planner
Emergency
Top Picks For You
Explore highly-rated attractions, landmarks, and points of interest.
AI Tip: Want the inside scoop? My top picks in a new area are always where the locals actually go, not just where the tourists flock!
Mini Seashore
I Love Navi Mumbai
Turbhe lake view point
Chhatrapati Chouk
Ocean View Tower
I Love Turbhe
Sarovar Vihar
Tourist hill


Travel Planner - Presentation
GoTogether
hospitals near me - Google Ma
go-together-uom.vercel.app/place/ChIJnVXUIQDB5zsRWdp4W7JHbRg
Plans
GoTogether
Search
EN
Home
Feed
Bookmarks
Planner
Emergency
cinépolis
Cinepolis - Aurum square - Ghansoli Navi Mumbai
Key Information
Aurum Q, Cinepolis Cinema, Aurum Square, Reliance Corporate Park, MIDC Industrial Area, Ghansoli, Navi Mumbai, Mumbai, Maharashtra 400701, India
3.9 (176 ratings)
Reviews
Gokulraj
5.0
2 weeks ago
Recently visited the new Cinépolis at Aurum Mall,


Travel Planner - Presentation
GoTogether
hospitals near me - Google Ma
go-together-uom.vercel.app/planner
Plans
GoTogether
Search
EN
Home
Feed
Bookmarks
Planner
Emergency
Trip Planner
Plan your next adventure here!
Map
Satellite
Kurunegala
Kandy
Kalmunai
Ampara
Akkaraipattu
Negombo
Gampaha
Ja-Ela
Colombo
Nuwara Eliya
Pottuvil
Bandarawela
Peliyandala
Panadura
Ratnapura
Yala National Park
Tissamaharama
Hambantota
Galle
Matara
Google
Keyboard shortcuts
Map data ©2025
Terms
Route Details


Travel Planner - Presentation
GoTogether
hospitals near me - Google Ma
go-together-uom.vercel.app/feed
Plans
GoTogether
Search
EN
Home
Feed
Bookmarks
Planner
Emergency
What's on your mind?
Post
For you
Following
kavindu senarathne
May 18
From: Panadura
To: Kandy
Stayed at: Oak Ray Heritage
Favorite food: Kandy kottu
Memories: Temple of the Tooth, Kandy Lake
Went with: Kavindu, Malindu
Rating: 4.5 ★
"Walking around the serene Kandy Lake in the evening was just magical. We even caught the sunset near the temple steps!"
Like
3 comments



Key Features

- **Responsive Design**

  Ensures seamless interaction across mobile, tablet, and desktop devices. Layouts are tailored for travel planning on the go, with adaptive navigation and content scaling.

- **Dark Mode Support**

  Built-in toggle allows users to switch between light and dark themes effortlessly. Helps reduce eye strain, especially in low-light travel environments.

- **SSR + CSR Performance**

  Uses Server-Side Rendering (SSR) for fast initial loads and Client-Side Rendering (CSR) for dynamic interactions.Ensures quick page loads and fluid transitions.

- **Skeleton Screens**

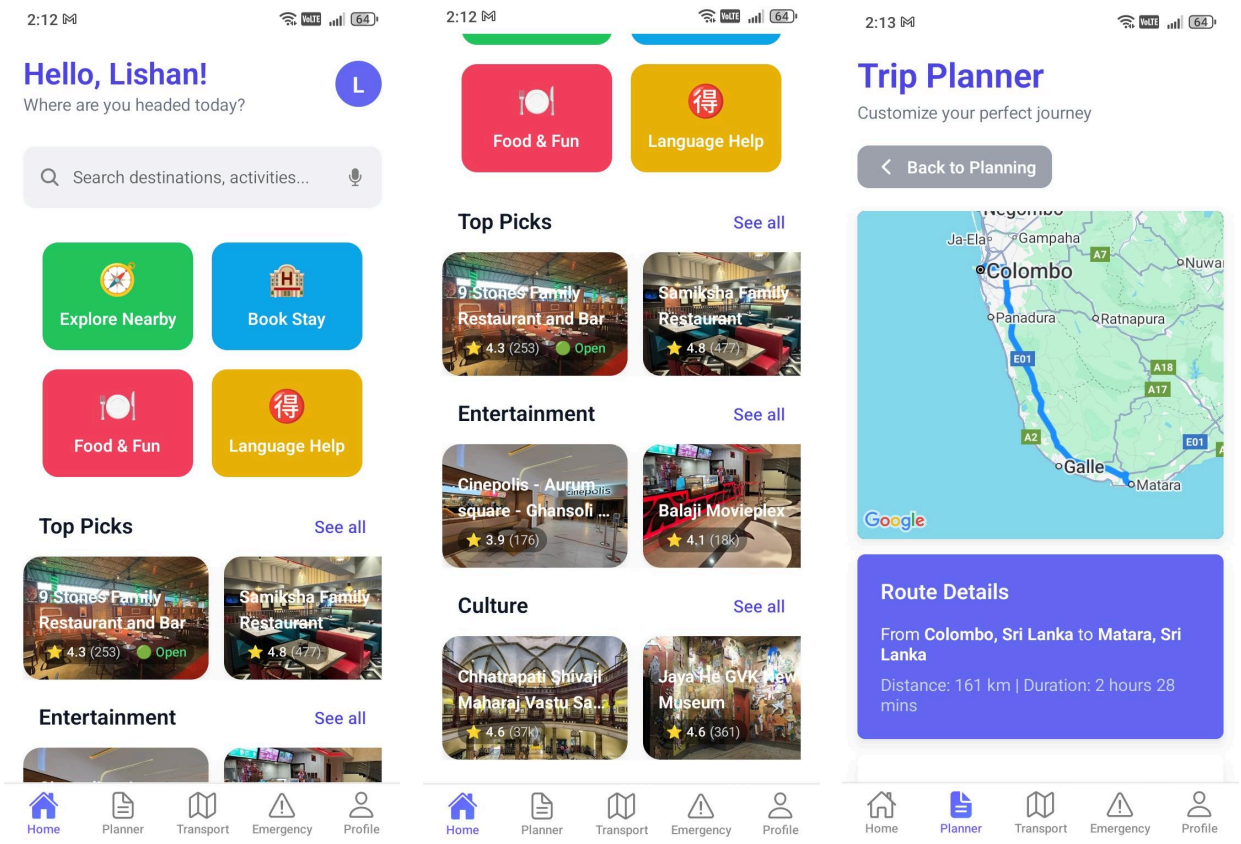
  Displays placeholders during content fetch (e.g., trips, posts, places), keeping the UI responsive and users engaged while data loads.

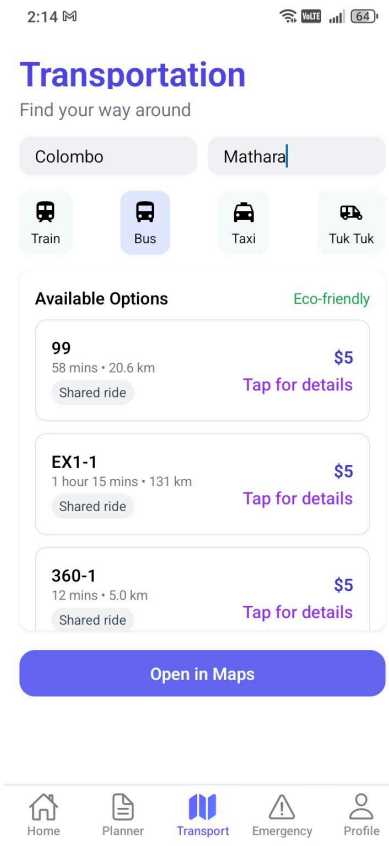
- **Image Optimization**

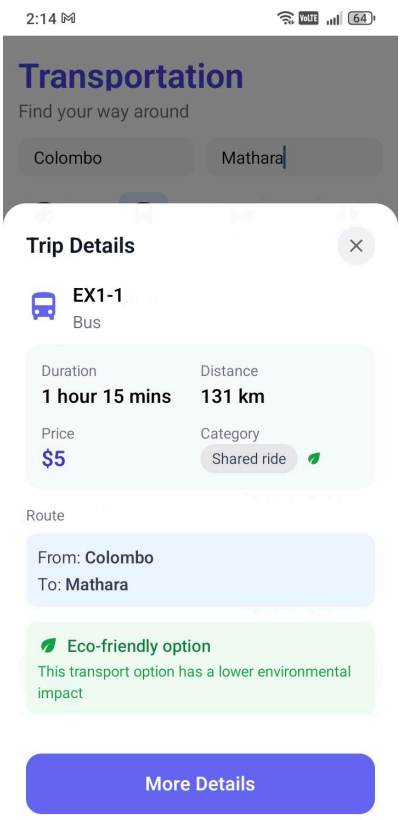
  Utilizes Next.js Image component for automatic image resizing, lazy loading, and CDN delivery—critical for rich media experiences in travel apps.

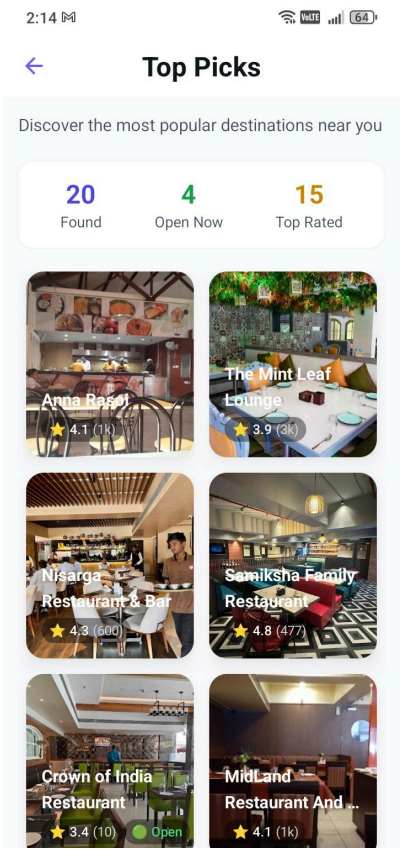
6.3 Mobile Frontend Design

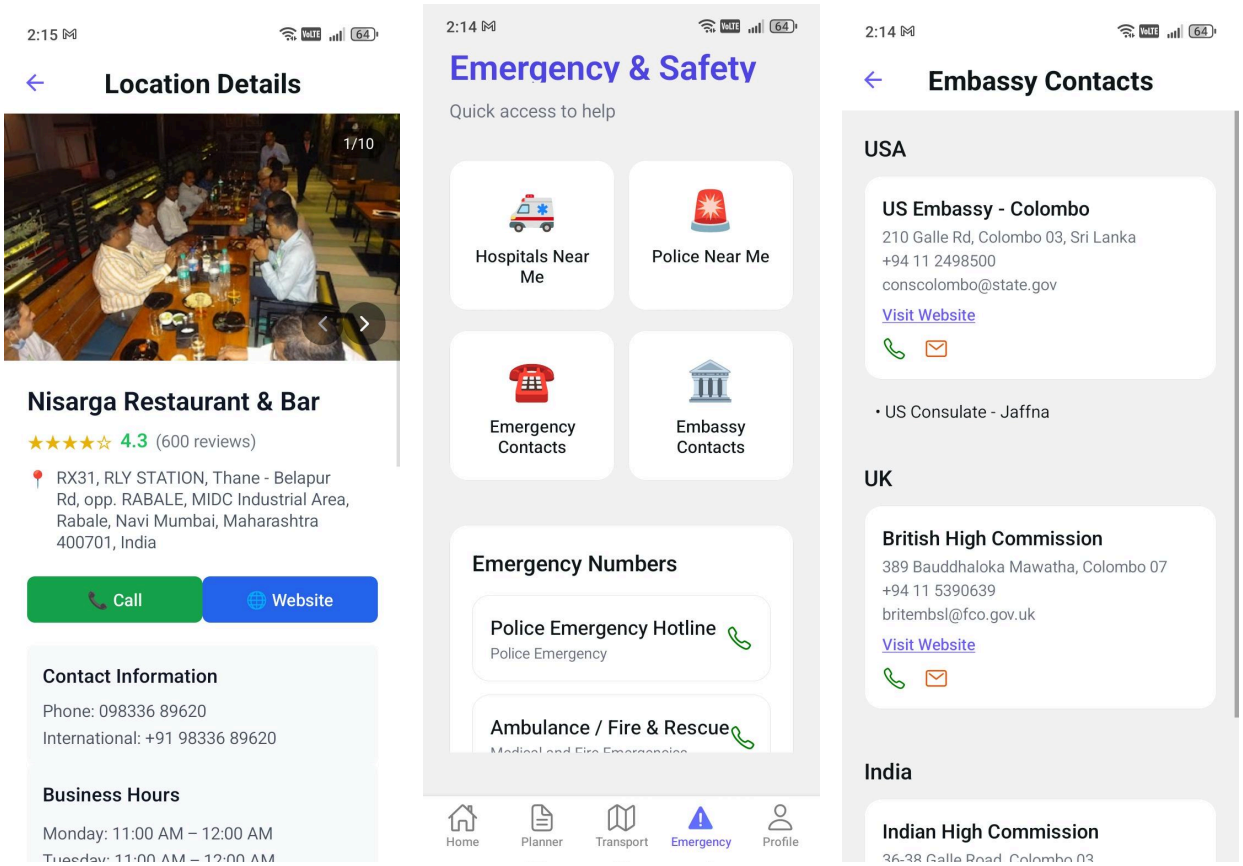
The mobile experience is built using React Native, enabling cross-platform development and consistent logic sharing with the web app.




2:14
Transportation
Find your way around
Colombo
Mathara
Train
Bus
Taxi
Tuk Tuk
Available Options
Eco-friendly
99
58 mins • 20.6 km
Shared ride
$5
Tap for details
EX1-1
1 hour 15 mins • 131 km
Shared ride
$5
Tap for details
360-1
12 mins • 5.0 km
Shared ride
$5
Tap for details
Open in Maps
Home
Planner
Transport
Emergency
Profile


2:14
Transportation
Find your way around
Colombo
Mathara
Trip Details
EX1-1
Bus
Duration
1 hour 15 mins
Distance
131 km
Price
$5
Category
Shared ride
Route
From: Colombo
To: Mathara
Eco-friendly option
This transport option has a lower environmental impact
More Details


2:14
Top Picks
Discover the most popular destinations near you
20
Found
4
Open Now
15
Top Rated
Anna Rasa
4.1 (1k)
The Mint Leaf Lounge
3.9 (3k)
Nisarga Restaurant & Bar
4.3 (600)
Samiksha Family Restaurant
4.8 (477)
Crown of India Restaurant
3.4 (10)
Open
MidLand Restaurant And ...
4.1 (1k)



Highlights

- **Platform-Native Experience**

  Delivers smooth interactions and UI responsiveness aligned with both iOS and Android guidelines**.**

- **Shared Logic with Web**

  Core logic, models, and API services are shared with the Next.js frontend to ensure maintainability and consistent user experience.

- **Responsive Theming**

  Adaptive UI scaling, and intuitive mobile gestures for seamless planning and collaboration.

6.4 Backend Design

Microservices deployed via containers in Docker Compose :

`api-service` (Go):

- Handles place search, mapping, geocoding
- Talks to Google Maps API and trip planner
- Gemini calls

`planning-service` (Spring Boot):

- Manages event/trip creation, editing, participation
- Validates user preferences and generates travel plans using business logic and AI suggestions

`social-media-service` (Spring Boot):

- Handles post creation, comments, likes, and user feed rendering vvia paginations
- Stores data in PostgreSQL (via Supabase)

`auth-service` (Keycloak):

- Handles authentication, registration, and session/token management
- Secured via HTTPS, integrated with Kong Gateway and JWT-based access

`user-service(springboot)`:

- Manages user profiles, including name, bio, and profile picture
- Implements follow/unfollow system with tracking of social connections

Storage:

- **Supabase PostgreSQL**

  Used for structured storage (users, trips, social metadata, relationships)

- **Cloudflare Images**

  Optimized image CDN for storing and delivering profile pictures, post media, etc.

6.5 Security and Privacy Design

Authentication and Authorization

- **OAuth2.0 + OpenID Connect (OIDC)**:
  Implemented via Keycloak, enabling secure login, token-based session management, and third-party login extensibility.
- **JWT Access Tokens**:
  APIs are protected using signed JSON Web Tokens for stateless authentication and secure delegation of identity.

Secure Communication Channels

- **HTTPS Everywhere**:
  All public endpoints are secured using TLS (via AWS ACM) to ensure data confidentiality and integrity in transit.
- **WSS (WebSocket Secure)**:
  Real-time communication (e.g., notifications) uses encrypted WebSocket channels, protecting message streams against interception.
- **Reverse Proxy Security**:
  All traffic is routed through an NGINX-based proxy, enforcing HTTPS redirection and secure gateway routing.

Data Privacy and Protection

- **Email Verification**:
  Users must confirm their email address during registration, reducing the risk of spam accounts or abuse.

7. Technology Stack

7.1 Frontend Technologies

Layer	Technology	Purpose
Web Frontend	Next.js (React)	SSR/CSR support, routing, image optimization, and seamless UI rendering
Mobile App	React Native	Cross-platform mobile app for Android/iOS
Styling	Tailwind CSS	Utility-first styling with responsive design support
UI Features	Dark Mode, Skeleton Views	Enhanced UX with fast loading and adaptive themes

7.2 Backend Technologies

Service	Technology	Description
API Services	Go	Handles business logic, REST/gRPC APIs, and external integrations
Authentication	Keycloak	OAuth2-based login, session management, role-based access control
User Service	Spring Boot	User profile, follow/unfollow system
Planning Service	Spring Boot	Trip/event creation, preference handling, itinerary generation
Social Media Service	Spring Boot	Posts, comments, likes, user feed rendering
Notification	Spring Boot	WebSocket + email alerts for

Service		real-time updates

7.3 Database

Storage Type	Technology	Description
Relational DB	Supabase PostgreSQL	Main structured storage for users, events, and social data
Media Storage	Cloudflare Images	Optimized image CDN for profile pictures and post media

7.4 Cloud Infrastructure

Component	Provider/Tool	Purpose
Frontend Hosting	Vercel(Serverless)	Serverless deployment of the Next.js web frontend
Backend Hosting	Oracle Cloud VM + Docker Compose	Hosts all microservices (Go, Spring Boot, Keycloak, etc.) using containerized setup
Database Hosting	subabase(Serverless)	Fully managed PostgreSQL database with API and RLS features
Image Delivery	Cloudflare Images CDN	Global CDN for optimized and secure media delivery

7.5 DevOps and CI/CD Tools

Tool / Service	Role
GitHub Actions	CI pipeline for code build, testing, and Docker image publishing

Docker Compose	Local service orchestration and deployment on Oracle Cloud VM
Docker	Containerization of all backend microservices (Go, Spring Boot, Keycloak)
Certbot + NGINX	Local reverse proxy with HTTPS support for testing or fallback

8. Personalized Recommendation Engine

8.1 Context-Aware Personalization

The Travel Planner application leverages a Large Language Model (LLM), specifically the Gemini API, to enable intelligent, context-aware travel recommendations. The core idea is to tailor each user's experience based on their current search intent and historical travel behavior. For instance, a user who has shown an interest in religious sites and is currently planning a trip to Kandy may receive a search query like:

> *"religious places near Kandy"*

This eliminates the need for users to explicitly enter detailed preferences and helps deliver more meaningful and personalized suggestions.

8.2 User Data Collection & History

To enable personalization, the system first retrieves user-specific travel history using the following call:

```
UserTripHistoryDTO history =
userServiceClient.getUserTripHistory(request.getUserId());
```

This DTO (**`UserTripHistoryDTO`**) contains a list of previously visited places. These locations are assumed to reflect user preferences and are passed as context into the LLM prompt. If no history is found, the system gracefully degrades to a default, non-personalized search experience.

8.3 LLM-Powered Query Generation

Once user history is retrieved, the system dynamically constructs a semantic prompt to feed into the Gemini LLM. This prompt is designed to output a single, contextually relevant query string, suitable for direct use with the Google Maps Place Search API.

- **Prompt Used in Implementation:**

```
“The user has previously visited: [PLACE_1, PLACE_2, ...]. They're currently searching in [LOCATION]. Generate only one customized query string that can be used directly in the Google Maps Place Search API. Output only the query, with no explanation, markdown, or additional formatting. Do not URL encode it.”
```

Example with actual values:

```
“The user has previously visited: Temple of the Tooth, Adam's Peak. They're currently searching in Kandy. Generate only one customized query string that can be used directly in the Google Maps Place Search API. Output only the query, with no explanation, markdown, or additional formatting. Do not URL encode it.”
```

This prompt is sent to Gemini, and the system extracts the LLM response and injects it into the downstream request:

```
String customizedQuery = extractText(response);
request.setQuery(customizedQuery);
```

This enables the query to be executed via the Maps API to fetch relevant places matching inferred user intent.

8.4 Optimization Logic

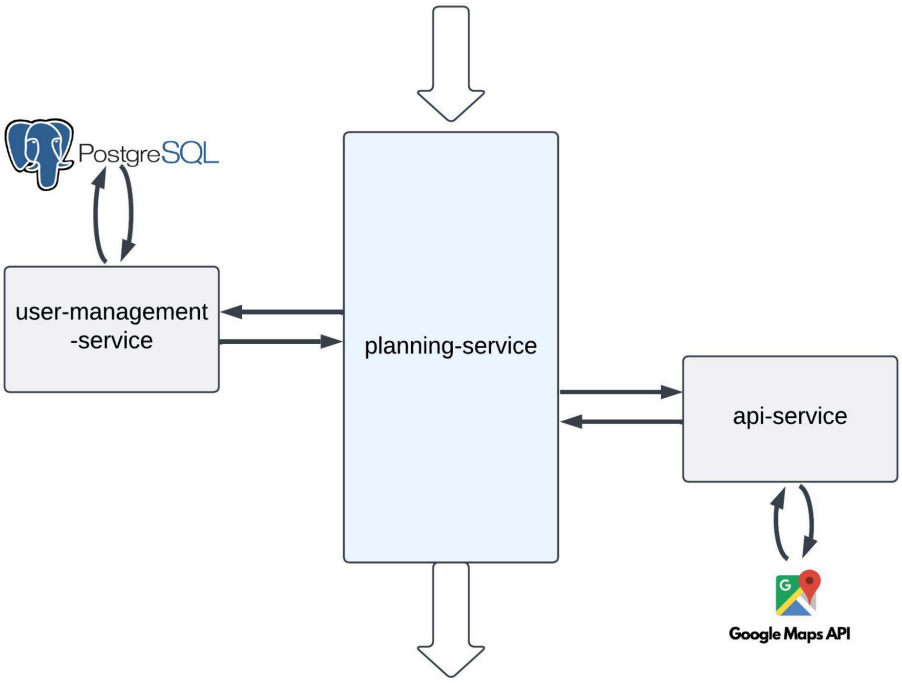
The generated query is:

- **Location-aware**: Includes the current location explicitly.
- **Interest-aware**: Leverages historical interest patterns.
- **API-ready**: Formatted to be directly consumable by the Google Maps API without post-processing.

This significantly reduces cognitive load on the user, allowing them to discover new places without manual filtering, keyword guessing, or redundant input.

Additionally, this approach:

- Enhances the precision of search results.
- Shortens the time-to-decision for users.
- Boosts the utility of third-party APIs via semantic enrichment.



8.5 Evaluation and Challenges

Evaluation Metrics

We evaluated the effectiveness of our LLM-generated query personalization through:

- **Click-through rates (CTR)** on suggested places.
- **Time-to-first-action** metrics.
- **User engagement** (favoriting, bookmarking, route initiation).

Limitations

- **Cold Start**: No personalization is possible for new users with no travel history.
- **LLM Quality**: The effectiveness heavily relies on the relevance and clarity of LLM responses.

- **Latency**: Introducing the LLM in the search pipeline adds response delay (~300–700ms depending on the network/API).

Planned Improvements

- Introduce **few-shot prompting** or context chaining to improve LLM precision.
- Implement **user feedback collection** on search relevance to reinforce learning.
- Add **local event and seasonality data** (e.g., festivals in Kandy) for time-sensitive personalization.
- Explore **fine-tuning** or **in-context learning** using user data to improve model performance.

---

9. Conclusion and Future Work

9.1 Summary of Achievements

The GoTogether application successfully delivers a robust, user-centered travel planning experience tailored specifically for Sri Lanka. Through the integration of mobile and web platforms, the system enables:

- **AI-powered personalized tour planning** using user history and location data.
- **Seamless access to transportation options**, including bus, train, and taxi routes with estimated fares.
- **Collaborative trip creation and sharing**, enabling group planning with real-time updates.

- **Cultural discovery tools**, including restaurant and accommodation suggestions, language translation, and safety alerts.
- A **modular, microservice-based architecture** using technologies like Spring Boot, Go, gRPC, and Kafka.
- **Scalable deployment and DevOps pipelines**, ensuring high availability and maintainability via Docker, GitHub Actions, Vercel, and Oracle Cloud.

This project not only bridges the gap between fragmented travel information systems in Sri Lanka but also empowers both local and foreign travelers with intelligent, real-time decision-making tools.

9.2 User Feedback and Observations

Initial usability testing and informal feedback collection highlighted several key benefits and user sentiments:

- **Ease of Use**: Users appreciated the clean UI, dark mode support, and responsive layouts across devices.
- **Effective Personalization**: LLM-powered suggestions resonated well with users, especially when they recognized contextual relevance in place recommendations.
- **Collaborative Features**: Group trip planning and feed-based social interactions were seen as valuable additions that differentiated the app from static itinerary tools.
- **Offline Gaps**: Some users noted challenges with network dependency during travel in rural areas.
- **Request for More Real-Time Info**: Users requested real-time traffic data, bus delays, and event recommendations.

9.3 Potential Improvements

Despite the success of the MVP, several areas were identified for refinement:

- **Offline Functionality**: Introduce limited offline mode with cached maps, saved itineraries, and emergency contact access.
- **Transport Data Automation**: Automate fare and schedule updates using government APIs or scraping bots to minimize manual maintenance.
- **Multilingual Chatbot Integration**: Enhance language assistance with a real-time AI chatbot that supports Sinhala, Tamil, and English.
- **User Reward System**: Implement gamified incentives (e.g., badges for reviews or trip shares) to encourage community engagement and contributions.
- **Accessibility Enhancements**: Improve experience for differently-abled users by integrating screen reader support and gesture-based navigation.

9.4 Roadmap for Future Enhancements

The roadmap for GoTogether includes several high-impact extensions:

Short-Term (Next 3–6 Months)

- Add **AI chatbot assistant** for in-app conversational support.
- Enable **user-generated POIs (Points of Interest)** and reviews with moderation.
- Improve **real-time public transport tracking** using third-party GPS integrations.
- Expand **support for personalized recommendations** via user feedback loops.

Mid-Term (6–12 Months)

- Build **in-app booking capabilities** for transportation and accommodation.
- Develop **admin dashboard** for tourism board analytics and user behavior insights.
- Integrate **currency conversion** and **weather forecasting** features.
- Localize the app fully in **Sinhala and Tamil**, including AI translations.

Long-Term (12+ Months)

- Expand to cover other South Asian travel hubs using the same personalization engine.
- Launch **AI trip companion mode**, combining LLMs and real-time location tracking for voice-guided, adaptive tour guidance.
- Partner with **local travel agencies and guides** to create curated tours, especially for eco and heritage tourism.

In summary, *GoTogether* represents a next-generation, AI-driven travel companion that fills a crucial gap in Sri Lanka's tourism and domestic travel ecosystem. It blends personalization, collaboration, and smart data usage into a seamless experience—paving the way for smarter, safer, and more memorable journeys across the island.

Annex A: Individual Contribution

Fernando T.H.L. (210167E)

As the lead developer of the web frontend, I was responsible for building the user interface, the authentication module, and the user service. This included designing features like user registration, login, and profile management to ensure users could interact with the system securely and smoothly.

On the frontend, I implemented route protection to ensure that only authenticated users could access certain pages. I used cookie-based session handling to store tokens securely, and I carefully chose between client-side rendering (CSR) and server-side rendering (SSR) based on the specific needs of each page

One of the hardest parts of the project was managing the inter-service communication between the microservices. At the beginning, I overlooked port conflicts and startup dependencies, which led to services failing unexpectedly. Writing the `docker-compose.yml` file was more complicated than I thought, especially when trying to ensure the user service only started after Keycloak was fully configured. Initially, I tried using health checks, but it didn't work because Keycloak needed to finish running a manual configuration script before the user service could connect. As a workaround, I wrote a custom wait script using a ping mechanism that only started the user service after the Keycloak configuration was completed. This experience taught me how important service readiness and orchestration is in microservice deployments.

Configuring HTTPS in Keycloak was also a challenge. It took several tries to get it working correctly, especially when setting the trusted issuer URL required by Spring Security in the user service. On top of that, every time Keycloak was created using Docker Compose, it started fresh without my custom configurations like realms, clients, and roles. To solve this, I created a custom script that takes a snapshot of a configured

Keycloak instance and uploads it automatically during startup. This made it possible for other developers to simply run `docker-compose` and get a fully configured Keycloak instance, which saved time and improved consistency across environments.

Deploying the frontend also came with its share of issues. Since we used TypeScript with strict linting rules, I had to resolve several type and dependency issues before the build would pass. In production, we ran into more problems originally, our backend used HTTP, but platforms like Vercel blocked non-HTTPS APIs. This forced us to properly configure HTTPS using NGINX and Certbot. Since HTTPS requires a domain name, I looked for free solutions and found DuckDNS, which gave us a subdomain we could use for secure deployment without extra costs.

To optimize our Docker builds, I used `.dockerignore` to reduce image size and implemented multi-stage builds. However, building the Spring Boot JAR inside Docker was taking too long, so I switched to building the JAR locally and copying it into the Docker image manually, which saved time and made the build more reliable. To further streamline the development and deployment process, I created a `deploy.sh` script with various flags. This script allowed other developers to easily build, run, stop, or deploy the application using Docker Compose without manually handling tasks like JAR building. It also supported service-specific deployment, enabling developers to spin up only the required services instead of the full stack—making local testing and incremental development much more efficient. Additionally, this script was designed to be CI/CD-friendly, making it easy to integrate into GitHub Actions pipelines for automated builds and deployments. It also simplified deployment on our Oracle VM instance by wrapping complex steps into a single, consistent workflow, reducing manual effort and minimizing room for error.

Initially, I chose AWS Free Tier to host and build the services, but I quickly ran into limitations due to low resource availability on EC2 instances. The machines couldn’t even build the services properly without running into memory or CPU constraints, making it unsuitable for our production needs. After researching alternatives, I

discovered that Oracle Cloud’s Free Tier offered more generous compute and memory resources.

I switched to using an Oracle VM, which significantly improved build and deployment performance. I also updated our GitHub Actions pipeline to replace the AWS-based setup with one designed for Oracle Cloud, resulting in a much smoother CI/CD experience. During this process, I learned how to handle VM provisioning, SSH access, and remote deployments more confidently. It was my first hands-on experience managing cloud infrastructure independently, and it helped me become more familiar with real-world DevOps practices and production environment management.

Overall, these challenges were frustrating at times, but solving them taught me a lot about system integration, secure deployment, containerization, frontend-backend coordination, and infrastructure management. Most importantly, I learned how to think ahead about dependencies, automation, and environment consistency skills that are critical in real-world software projects.Overall, these challenges were frustrating at times, but solving them taught me a lot about system integration, secure deployment, containerization, frontend-backend coordination, and infrastructure management. Most importantly, I learned how to think ahead about dependencies, automation, and environment consistency skills that are critical in real-world software projects. In addition to my technical contributions, I actively participated in team presentations, report writing, Software Requirements Specification (SRS) creation, and architectural planning sessions. These collaborative efforts helped align our team's vision and ensured we maintained both technical and documentation quality throughout the project.

Gamage M.S. (210176F)

Initially, I saw this primarily as a logistical challenge: how to streamline route planning, deliver accurate public transport data, and present it within an intuitive interface. Early stakeholder feedback and market analysis made it clear that the problem extended beyond simple route optimization. Travelers not only lacked reliable transport

schedules, but also struggled with decision-making due to poor guidance and Sinhala-only resources.

The problem required more than just a mapping interface, it demanded the creation of a system that intelligently integrates transportation data, overcomes language limitations, and provides practical assistance to tourists on the ground.

My initial goal was to build an integrated travel assistant that would:

- Consolidate bus, train, and tuk-tuk schedules and fares.
- Offer real-time trip planning and fare estimates.
- Include connections to services like Uber and PickMe.
- Support users through multilingual access.

I studied similar applications in other emerging tourism markets, focusing on how they sourced and managed transit data in low-infrastructure environments. Through this research, I discovered a major constraint: Sri Lanka lacked any centralized, machine-readable transport database. Much of the information existed only in scanned Sinhala PDFs, making integration technically difficult and inconsistent.

One of the most significant challenges during the project was the difficulty in accessing usable public transport fare data. Although fare calculations were a key planned feature, we were unable to implement them because the available data was primarily published in Sinhala and only accessible through static PDF documents. These documents were not in a structured or machine-readable format, making it impractical to extract and integrate the information into the application within the project scope.

As a result, the initial version of the app focused on general route guidance, while fare estimation was deferred as a future enhancement, dependent on access to structured, digital datasets or official transport APIs.

Another anticipated feature, direct in-app integration with Uber and PickMe, proved infeasible due to API limitations. Uber’s platform required formal partnerships, and PickMe did not offer public APIs. As a compromise, I allowed users to access these services externally with instructions and pre-filled data, preserving convenience while avoiding technical roadblocks.

I was responsible for designing the early prototypes and UI/UX structure of the mobile application. My objectives were clarity, minimalism, and responsiveness particularly important in low-connectivity environments.

The designs were inspired by leading travel apps like Rome2Rio and Google Maps but were simplified for a more localized and accessible experience.

To support rapid iteration and reduce manual deployment effort, I implemented a CI/CD pipeline using GitHub Actions with deployment to Oracle Cloud Infrastructure (OCI). This automated:

- Code build and test execution upon commits.
- Deployment to development and production environments.
- Consistency in versioning and codebase health.

This infrastructure drastically improved development velocity and reduced risk during updates or integrations.

My core technical responsibility was building the backend service in Go (Golang) using the Gin framework. The decision was driven by:

- Concurrency and performance: Go's goroutines handled multiple API calls efficiently.
- Static typing and fast compilation: These improved reliability and simplified deployment.
- Gin framework: Provided a lightweight, high-performance HTTP router with built-in support for RESTful APIs, JSON responses, middleware, and modular development.

A defining feature of the backend was its integration with several Google APIs, which filled critical data gaps. I wrote middleware around these APIs to handle caching, rate limiting, and error recovery. These integrations enabled the app to function with real-time routing and intelligent suggestions, despite the absence of a reliable local transport API.

Key learnings

- Underestimating data acquisition complexity was a key mistake. The effort required to extract and standardize transport data, especially in Sinhala was significantly higher than expected.

- Fallback strategies are essential when working in environments with unreliable or non-existent digital infrastructure.

This project has been a transformative experience, challenging my assumptions and expanding my problem-solving abilities. It taught me how to work with unstructured data, build reliable APIs around external services, and adapt system design to local constraints. I learned the importance of flexibility—not just in code, but in product vision and development strategy.

Rather than simply replicating existing travel solutions from other countries, we built a system uniquely designed for Sri Lanka. It acknowledges infrastructural limitations while still providing smart, practical, and usable features for travelers.

Looking forward, I see opportunities to expand data coverage, partner with transport authorities, and further personalize the experience through continuous user feedback and iterative design. This project has laid a strong foundation for both future development and my own growth as a developer and designer.

Lishan S.D. (210339J)

As the lead mobile application developer, I took on a pivotal role throughout the development of the GoTogether travel management application, with primary responsibility for frontend development and deployment.

My core responsibility involved developing and deploying the mobile application frontend, which required translating our vision for an intuitive travel companion into a functional, user-friendly interface.

The development of major functionalities in the mobile application presented both challenges and learning opportunities. I successfully implemented the home screen, which serves as the central hub for users to access all application features. This

required careful consideration of information architecture to ensure that both novice and experienced travelers could navigate efficiently.

Creating the Transport Details screen involved integrating multiple data sources, such as Google APIs and presenting transportation information in an accessible format. This functionality directly addresses one of our core problem statements regarding transportation information gaps in Sri Lanka.

Developing the login and signup functionality provided valuable insights into user onboarding processes. A user context provider was used to access user data throughout the app and allow users to edit their profiles. I focused on creating a streamlined authentication experience that minimizes barriers to entry while ensuring security standards.

One of my most significant contributions was developing the Top Picks, Culture, and Entertainment screens, which display nearby locations for each category. This functionality represents the heart of our application's value proposition, providing personalized, location-aware recommendations. The implementation required integrating the Google Maps geolocation API in the backend API service and creating dynamic content displays that adapt to user location and preferences.

Finally, I successfully deployed the mobile application to the Expo Store. This process involved thorough testing, optimization, and ensuring compliance with platform guidelines. The deployment experience taught me valuable lessons about production-ready application development and the importance of comprehensive testing across various scenarios.

My assistance in implementing the user service module in the backend provided opportunities to work across the full application stack. This collaboration enhanced my understanding of how frontend and backend systems integrate, particularly in the context of user management, authentication, and data synchronization. Working on the

user service module deepened my appreciation for the complexity of building scalable, secure travel applications.

Throughout development, I encountered various technical challenges as well. Managing state across multiple screens while maintaining performance, handling location services across different devices, and ensuring consistent user experience across various screen sizes and operating system versions all presented learning opportunities.

This project has significantly enhanced my technical skills in mobile application development, particularly in areas of location-based services, user interface design, and cross-platform deployment. More importantly, it has deepened my understanding of how technology can be leveraged to solve real-world problems and improve user experiences.

Meddawitage K.C.D. (210382H)

Throughout our travel planning project, I contributed significantly from the very outset, deeply involved in both the initial planning and detailed implementation phases. My active participation in the planning stage significantly shaped the overall vision and functionality of the application. Initially, the project aimed to deliver a smart travel assistant primarily focused on optimizing transportation and itinerary management for travelers across Sri Lanka. However, through iterative stakeholder feedback and comprehensive user research, we expanded our scope to include a collaborative social media platform tailored explicitly for travelers.

My specific contributions during the planning stage included:

- Defining the detailed system architecture for the social media component, clearly outlining how users would follow each other, share AI-generated trip plans, and interact through posts, comments, and likes.
- Designing and establishing a robust and scalable database schema using PostgreSQL, set up through Supabase. My responsibilities covered creating and

configuring essential tables such as users, user_profiles, followers, user_follows, user_following, user_followers, posts, reactions, comments, bookmarks, and media.

- Creating and maintaining comprehensive REST APIs to ensure efficient and seamless communication between backend services.

My primary technical contributions encompassed:

- Backend development using Java Spring Boot, enabling clean, modular, and maintainable code structures with robust error handling and efficient API management.
- Frontend implementation using Next.js, providing a responsive, intuitive, and engaging user interface.
- Integration of AI-generated itinerary recommendations with user-generated content, enabling users to share their plans, learn from others' experiences, and collaboratively enhance their travel decisions.

Notable technical features and implementations included:

- Complete social media functionalities: user following and follower interactions, post creation, commenting, bookmarking, and reacting mechanisms.
- AI-driven itinerary sharing, allowing users to gain insights from shared travel experiences, recognize potential issues, avoid common pitfalls, and enhance trip quality.
- Implementation of robust input validation, graceful error handling, and fallback mechanisms to ensure system resilience, stability, and performance.

Reflectively, this project taught me valuable lessons in relational database design, system integration, frontend-backend alignment, and user-centric functionality. I overcame multiple technical challenges, particularly in database schema alignment and API management, through continuous learning, precise planning, and effective

collaboration with team members. The experience significantly improved my expertise in full-stack development, backend architecture, and cloud-based database management.

Overall, my involvement in the travel planning project provided substantial professional growth, reinforcing my technical capabilities, system integration understanding, and collaborative teamwork skills. These insights and experiences will greatly inform my approach and effectiveness in future software engineering endeavors.

Senarathna L.P.S.U.K. (210588U)

From the outset, I was deeply involved in the planning phase of the GoTogether project, both at the design level and the logistical decision-making stage. Initially, the goal was to build a smart travel assistant that streamlined transportation and itinerary planning for travelers across Sri Lanka. However, as user research evolved, it became clear that the scope had to go beyond route optimization. Language accessibility, real-time data limitations, and inconsistent local infrastructure emerged as core challenges. I played a key role in identifying and articulating these issues during the early planning sessions, helping shape the product vision into one that would be both technically feasible and genuinely useful to end users.

During the planning stage, I was responsible for:

- **Defining the system architecture for the planning engine**, including how user preferences, trip history, and destination queries would be managed.
- Designing how the platform would **generate AI-driven itineraries** while adapting to the logistical reality of unstructured or missing transportation data.
- Contributing to **decision-making about external API dependencies**, caching mechanisms, and fallback strategies to ensure reliability in low-connectivity scenarios.

These discussions laid the groundwork for many of the app's key design principles: lightweight, modular, offline-tolerant, and tailored to Sri Lanka's unique context.

My primary technical contribution was the design and implementation of the planning-service, built using Spring Boot. This service orchestrates itinerary creation, participant handling, AI query generation, and trip data persistence.

Notable features I implemented include:

- **LLM-powered query personalization** using user trip history and current search context to generate meaningful, context-aware queries via the Gemini API.
- **Trip participation and collaboration logic**, allowing users to jointly build and edit itineraries in real time.
- **Integration with the API service** (written in Go) for dynamic route information and Google Maps queries.
- Robust input validation and exception handling to ensure stable and predictable behavior across distributed services.

I worked closely with the frontend team to ensure that the planning service provided intuitive, well-structured responses that could be rendered efficiently on both mobile and web platforms.

In addition to backend development, I actively contributed to project documentation, including:

- **API contracts and endpoint specifications** for the planning service.
- **Use case scenarios and flow diagrams** for collaborative planning and personalized itinerary generation.
- Contributions to the **Confluence home page** and internal guides that helped onboard team members and align work across services.

Clear documentation was especially important given our microservices architecture and concurrent development across multiple teams. I ensured that my service interfaces were well-defined and easy to integrate.

One of the most instructive challenges was designing around data scarcity and inconsistency. While we initially intended to provide fare estimations and real-time transport schedules, the absence of machine-readable government data (mostly available as scanned Sinhala PDFs) forced us to defer fare calculation features and adopt a progressive enhancement strategy instead. This taught me the value of flexibility in product planning and the importance of decoupling dependent components for future upgrades.

Working with external APIs like Gemini and Google Maps also presented architectural considerations around rate limiting, caching, and fallback strategies, all of which I handled through custom middleware and logic within the planning-service.

This project stretched me across several domains, product design, full-stack architecture, backend development, and team coordination. I learned to:

- Make strategic design decisions under uncertainty.
- Adapt feature goals to real-world infrastructure constraints.
- Build scalable, well-abstracted microservices.
- Communicate technical vision effectively through documentation and planning.

Rather than replicating a solution from elsewhere, we built a system grounded in the realities of Sri Lankan travel, one that respects local limitations while delivering meaningful, intelligent features. I’m proud of the foundation we’ve laid and look forward to expanding the system with improved personalization, transport data partnerships, and deeper user engagement features.