# WayPoint

CSIS 4495

Section 1

Simone Lue (Team Lead) - 300276605

Russell Han Josef - 300369073

Video demo link: https://www.youtube.com/watch?v=ByDyuL8F20o

## Introduction

Travel planning is often a complex and time-consuming process, especially for travelers seeking personalized experiences. Many existing travel applications offer generic recommendations that fail to align with individual preferences, making itinerary creation an overwhelming task. WayPoint aims to bridge this gap by leveraging AI-powered recommendations, geolocation services, and real-time interactivity to enhance travel planning efficiency.

Existing research highlights the increasing role of AI in the travel sector, with applications like Google Travel and TripIt streamlining planning. However, these platforms lack in-depth personalization, leaving a gap for solutions that cater to specific traveler styles. Our research addresses this gap by developing a personalized travel planning app that dynamically tailors recommendations to user preferences.

Our initial hypotheses include:

- Users will engage with a travel style quiz to receive tailored recommendations.
- Integration of APIs such as Google Places, Eventbrite, and OpenWeatherMap will enrich the travel experience.
- AI-driven recommendations will streamline trip planning and increase user satisfaction.
- Real-time updates and offline mode will enhance usability.

Potential Benefits

- For travelers: Enhanced personalization, reduced research time, and enriched travel experiences.
- For BC’s tourism industry: Increased user engagement with local attractions and businesses.
- For research: A scalable model for personalized travel applications applicable to other regions.

## Summary of the Initially Proposed Research Project

The proposed research project focuses on developing WayPoint, a mobile application that simplifies travel planning through AI-powered recommendations and interactive features.

The app includes:

- User Profiles & Travel Style Quiz: Users set preferences through a quiz that determines their travel style.
- Personalized Recommendations: Suggestions based on user preferences and real-time location.
- Interactive Map: A visual representation of recommendations using Google Maps API.
- Event Browsing: Local event discovery via Eventbrite API.
- Gamification: Users earn badges for exploring new locations.
- Chatbot Assistance: AI-driven travel tips.

The tech stack includes:

- Frontend: React Native (CLI) for UI and user interactions.
- Backend: FastAPI (Python) for database and API integrations.
- Databases: PostgreSQL (Heroku) and Firebase for real-time updates.
- APIs: Google Places, Google Maps, OpenAI, Eventbrite, OpenWeatherMap.

## Changes to the Proposal

1. Shift from React Native Expo to React Native CLI

- Reason: Expo had limitations with native modules required for Firebase and API integrations.

- Justification: React Native CLI provided more flexibility for dependency management and native module integration.

2. Modification of Database Architecture

   - Initial Approach: Firebase-only for real-time data.
   - New Approach: Hybrid approach using PostgreSQL for structured data and Firebase for real-time updates.
   - Justification: Ensures structured data integrity while allowing real-time updates for collaborative trip planning.

3. Revised MVP Priorities Based on User Surveys

   - Initial MVP: All features developed without priority. Development based on what Russell, and I deemed fit.
   - Revised Approach: Prioritized features based on user feedback. There were certain features that users wanted more than others and those are now the priority features to work on. We sorted the remaining features into “medium priorities” and “nice-to-haves".

4. API Integration Adjustments

   - Initial Plan: Use Google Places API without caching.
   - New Plan: Store Google Places data in PostgreSQL to reduce API calls and improve performance.
   - Justification: Avoid exceeding API request limits and speed up app performance.

5. Task Allocation

   - Initial Plan: Simone takes care of frontend majority; Russell takes care of backend majority.
   - New Plan: Simone and Russell assigned one MVP component each week during the meetings which involves backend, frontend, and integration.

– Justification: Both Simone and Russell can have equal learning opportunities for frontend and backend. In addition, it makes developing MVP components smoother as there is no handoff on an incomplete feature. This also reduces the time it takes to develop the feature.

## Project Planning and Timeline

Updated Timeline (Current - End of Term)

Jan 24 – 29: Design Phase **(Completed)**

- Deliverables:
    - Wireframes showcasing the app's user interface and flow.
    - A well-documented database schema outlining the structure for user data, itineraries, and recommendations.

Jan 30 – Feb 5: Project Setup **(Completed)**

- Deliverables:
    - A ready-to-use centralized GitHub repository for version control.
    - Fully configured frontend and backend environments ready for development.

Feb 6 – Mar 25: Development of Minimal Viable Product **(In Progress)**

- Responsibilities:
    - Each team member will be assigned an MVP feature every week during the weekly meetings.
    - Each MVP feature will be taken from beginning to the end by the assigned team member (frontend + backend)
    - Simone: Lead on frontend
    - Russell: Lead on backend

- Milestone Features:
    - Travel Style Quiz (Feb 6-11)
    - Personalized Recommendations (Feb 11-25)
    - Interactive Map (Feb 11-25)
    - User Profile Management (Feb 25-Mar 11)
    - Itinerary Planning (Feb 25-Mar 11)
    - Chatbot Assistance (Mar 11-25)
    - Gamified Exploration (Mar 11-25)
- Weekly Deliverables:
    - Week 1: Users are able to take a quiz to determine their travel style (e.g., Relaxation, Culture, Adventure). Results are saved to the database to be used in other features.
    - Week 2-4: Tailored suggestions for destinations, activities, and events based on user preferences and location, using Google Places API.
    - Week 2-4: Visualize recommended places on an interactive map using Google Maps API.
    - Week 4-6: Users can create an account, manage profiles, and save preferences (e.g., travel style, favorite destinations).
    - Week 4-6: Users can create and edit trip itineraries in real-time, with live updates and syncing
        - Optional: Collaborative itineraries with friends (if time permits)
    - Week 6-8: A simple chatbot powered by OpenAI to assist users with travel-related FAQs and personalized advice.

    - Week 6-8: Users earn badges and collectibles (e.g., 'Beach Explorer') as they interact with the app, with real-time updates.

Mar 25-31: Integration and Deployment

- Responsibilities:
    - Simone and Russell will confirm all features are well integrated together.
    - Feature integrations will be divided between the team members during the weekly meeting.
    - Both team members will ensure the application works on both their own separate devices.
    - Deployment research will be done by both team members.
    - Deployment of the application will be done together in person or over a virtual meeting.
- Milestones:
    - Frontend and backend integration completed
    - Backend deployed to Heroku
- Deliverables:
    - A seamlessly integrated application deployed on both iOS.

Apr 1-5: User Testing Phase

- Responsibilities:
    - Both team members will reach out to a total of 10 users for user testing
    - Simone will create open-ended user testing questions
    - Russell will create an analysis of the user testing results with a summary of the next steps for adjustments.
- Milestones:

    - User testing for Waypoint
    - Noting of any problems/bugs/issues with user flow
- Deliverables:
    - User testing analysis
    - Documentation of app issues

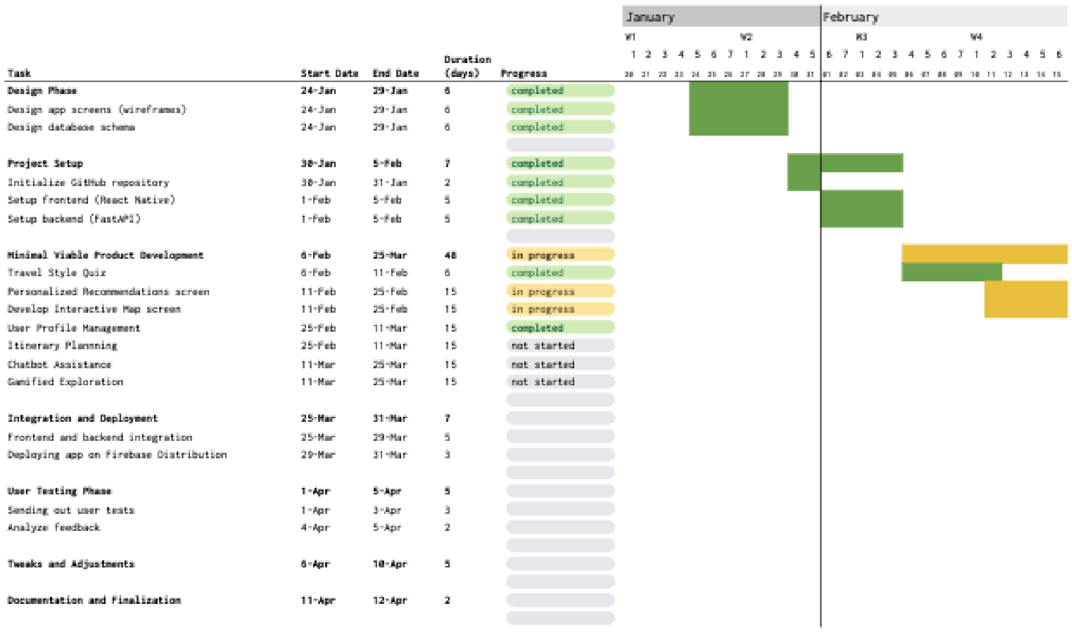
Apr 6-10: Tweaks and Adjustments

- Responsibilities:
    - Both team members will be assigned features to adjust based on user testing results.
- Milestones:
    - Fixing any issues that arose from user testing
    - Tweaking any additional features if necessary or if time permits
- Deliverables:
    - Fully functional and user tested WayPoint deployed on iOS.

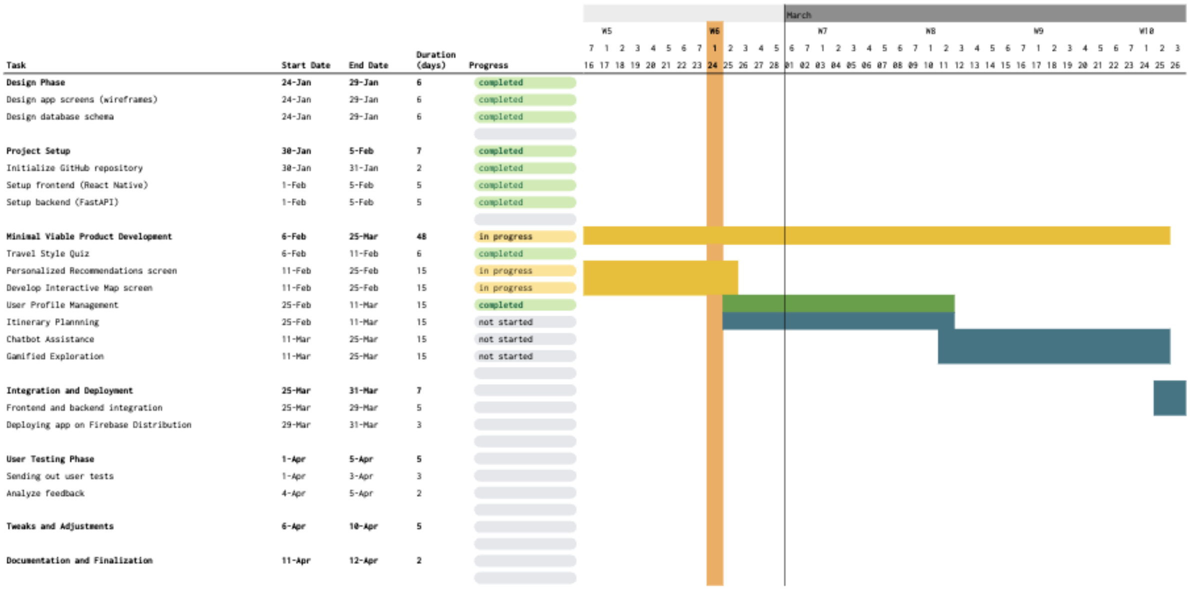
April 11-12: Documentation and Finalization

- Responsibilities:
    - Simone will handle the project submission report.
    - Russell will handle the README and user guides.
- Milestones:
    - README and user guide completed
    - Final project review and polish
- Deliverables:
    - Comprehensive user guide, project README, and developer notes.
    - A polished and fully functional version of WayPoint ready for submission.

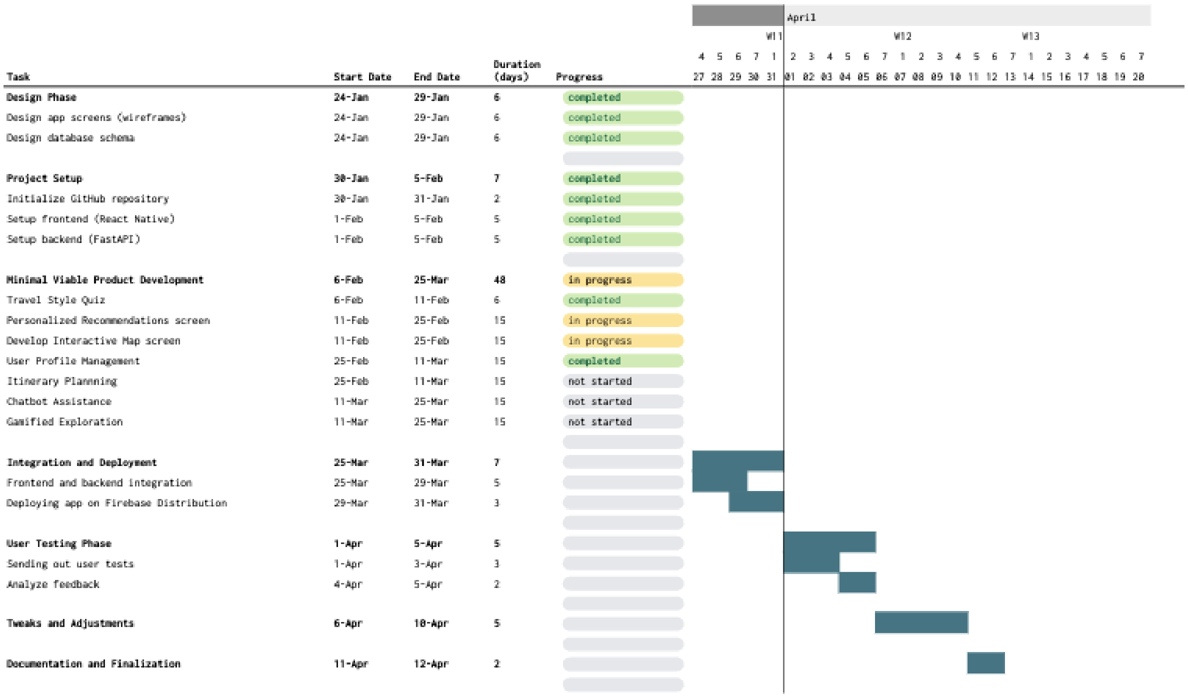
Gantt Chart:



| Task | Start Date | End Date | Duration (days) | Progress |
|---|---|---|---|---|
| **Design Phase** | **24-Jan** | **29-Jan** | **6** | completed |
| Design app screens (wireframes) | 24-Jan | 29-Jan | 6 | completed |
| Design database schema | 24-Jan | 29-Jan | 6 | completed |
| | | | | |
| **Project Setup** | **30-Jan** | **5-Feb** | **7** | completed |
| Initialize GitHub repository | 30-Jan | 31-Jan | 2 | completed |
| Setup frontend (React Native) | 1-Feb | 5-Feb | 5 | completed |
| Setup backend (FastAPI) | 1-Feb | 5-Feb | 5 | completed |
| | | | | |
| **Minimal Viable Product Development** | **6-Feb** | **25-Mar** | **48** | in progress |
| Travel Style Quiz | 6-Feb | 11-Feb | 6 | completed |
| Personalized Recommendations screen | 11-Feb | 25-Feb | 15 | in progress |
| Develop Interactive Map screen | 11-Feb | 25-Feb | 15 | in progress |
| User Profile Management | 25-Feb | 11-Mar | 15 | completed |
| Itinerary Planning | 25-Feb | 11-Mar | 15 | not started |
| Chatbot Assistance | 11-Mar | 25-Mar | 15 | not started |
| Gamified Exploration | 11-Mar | 25-Mar | 15 | not started |
| | | | | |
| **Integration and Deployment** | **25-Mar** | **31-Mar** | **7** | |
| Frontend and backend integration | 25-Mar | 29-Mar | 5 | |
| Deploying app on Firebase Distribution | 29-Mar | 31-Mar | 3 | |
| | | | | |
| **User Testing Phase** | **1-Apr** | **5-Apr** | **5** | |
| Sending out user tests | 1-Apr | 3-Apr | 3 | |
| Analyze feedback | 4-Apr | 5-Apr | 2 | |
| | | | | |
| **Tweaks and Adjustments** | **6-Apr** | **10-Apr** | **5** | |
| | | | | |
| **Documentation and Finalization** | **11-Apr** | **12-Apr** | **2** | |
| | | | | |



| Task | Start Date | End Date | Duration (days) | Progress |
|---|---|---|---|---|
| **Design Phase** | **24-Jan** | **29-Jan** | **6** | completed |
| Design app screens (wireframes) | 24-Jan | 29-Jan | 6 | completed |
| Design database schema | 24-Jan | 29-Jan | 6 | completed |
| | | | | |
| **Project Setup** | **30-Jan** | **5-Feb** | **7** | completed |
| Initialize GitHub repository | 30-Jan | 31-Jan | 2 | completed |
| Setup frontend (React Native) | 1-Feb | 5-Feb | 5 | completed |
| Setup backend (FastAPI) | 1-Feb | 5-Feb | 5 | completed |
| | | | | |
| **Minimal Viable Product Development** | **6-Feb** | **25-Mar** | **48** | in progress |
| Travel Style Quiz | 6-Feb | 11-Feb | 6 | completed |
| Personalized Recommendations screen | 11-Feb | 25-Feb | 15 | in progress |
| Develop Interactive Map screen | 11-Feb | 25-Feb | 15 | in progress |
| User Profile Management | 25-Feb | 11-Mar | 15 | completed |
| Itinerary Planning | 25-Feb | 11-Mar | 15 | not started |
| Chatbot Assistance | 11-Mar | 25-Mar | 15 | not started |
| Gamified Exploration | 11-Mar | 25-Mar | 15 | not started |
| | | | | |
| **Integration and Deployment** | **25-Mar** | **31-Mar** | **7** | |
| Frontend and backend integration | 25-Mar | 29-Mar | 5 | |
| Deploying app on Firebase Distribution | 29-Mar | 31-Mar | 3 | |
| | | | | |
| **User Testing Phase** | **1-Apr** | **5-Apr** | **5** | |
| Sending out user tests | 1-Apr | 3-Apr | 3 | |
| Analyze feedback | 4-Apr | 5-Apr | 2 | |
| | | | | |
| **Tweaks and Adjustments** | **6-Apr** | **10-Apr** | **5** | |
| | | | | |
| **Documentation and Finalization** | **11-Apr** | **12-Apr** | **2** | |
| | | | | |



| Task | Start Date | End Date | Duration (days) | Progress |
|---|---|---|---|---|
| **Design Phase** | **24-Jan** | **29-Jan** | **6** | completed |
| Design app screens (wireframes) | 24-Jan | 29-Jan | 6 | completed |
| Design database schema | 24-Jan | 29-Jan | 6 | completed |
| | | | | |
| **Project Setup** | **30-Jan** | **5-Feb** | **7** | completed |
| Initialize GitHub repository | 30-Jan | 31-Jan | 2 | completed |
| Setup frontend (React Native) | 1-Feb | 5-Feb | 5 | completed |
| Setup backend (FastAPI) | 1-Feb | 5-Feb | 5 | completed |
| | | | | |
| **Minimal Viable Product Development** | **6-Feb** | **25-Mar** | **48** | in progress |
| Travel Style Quiz | 6-Feb | 11-Feb | 6 | completed |
| Personalized Recommendations screen | 11-Feb | 25-Feb | 15 | in progress |
| Develop Interactive Map screen | 11-Feb | 25-Feb | 15 | in progress |
| User Profile Management | 25-Feb | 11-Mar | 15 | completed |
| Itinerary Planning | 25-Feb | 11-Mar | 15 | not started |
| Chatbot Assistance | 11-Mar | 25-Mar | 15 | not started |
| Gamified Exploration | 11-Mar | 25-Mar | 15 | not started |
| | | | | |
| **Integration and Deployment** | **25-Mar** | **31-Mar** | **7** | |
| Frontend and backend integration | 25-Mar | 29-Mar | 5 | |
| Deploying app on Firebase Distribution | 29-Mar | 31-Mar | 3 | |
| | | | | |
| **User Testing Phase** | **1-Apr** | **5-Apr** | **5** | |
| Sending out user tests | 1-Apr | 3-Apr | 3 | |
| Analyze feedback | 4-Apr | 5-Apr | 2 | |
| | | | | |
| **Tweaks and Adjustments** | **6-Apr** | **10-Apr** | **5** | |
| | | | | |
| **Documentation and Finalization** | **11-Apr** | **12-Apr** | **2** | |

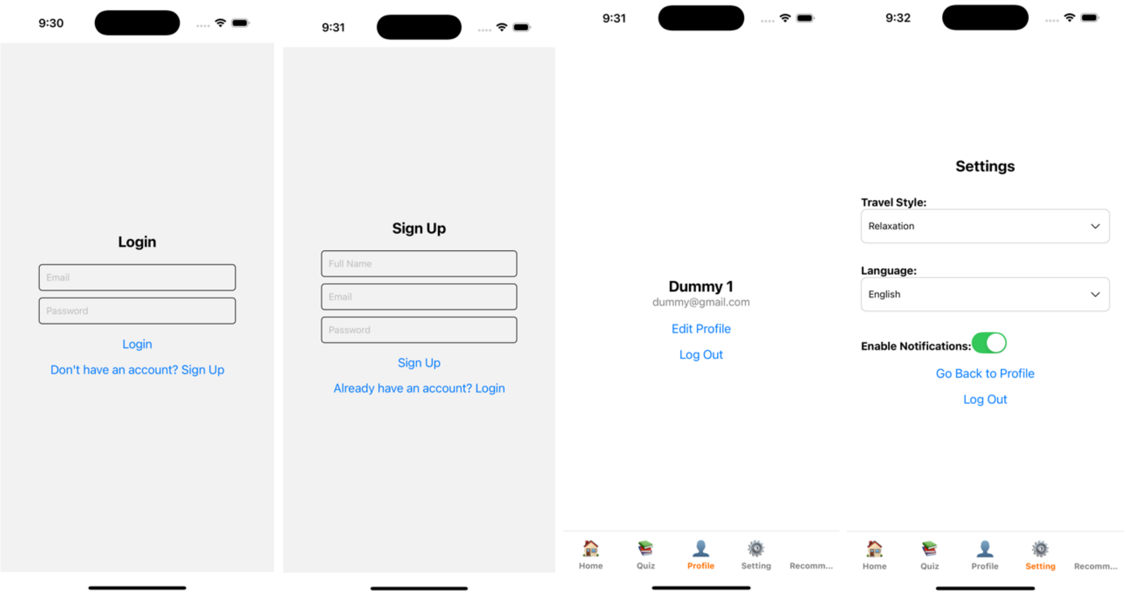
## <u>Implemented Feature: User Profile Management (Russell)</u>

- Designed and implemented the user authentication system.
- Login Process:
    - Users enter their email and password in the login form.
    - Form validation ensures the email format is correct and the password field is not empty.
    - Upon submission, credentials are sent to the FastAPI backend (`/users/auth/login`) for verification.
    - If authentication succeeds:
        - The user's details are stored in `AsyncStorage` for session persistence.
        - A login event is recorded in Firebase Realtime Database under `/logins/{user_id}`.

        - The last login timestamp is updated in Firebase.
        - The user is navigated to the Main Screen.
    - If authentication fails, an error message is displayed.

- Navigation Behaviour**:**
    - Successful login redirects users to the main app screen.
    - Users can navigate to the signup screen if they don't have an account.
- Error Handling and Edge Cases:
    - Displays appropriate error messages for incorrect credentials or missing fields.
    - Prevents duplicate login attempts while the request is processing.
    - Uses `Alert` to notify users about errors.
- Logout Functionality:
    - Users can log out via the Profile Screen or Settings Screen.
    - Clears stored session data in `AsyncStorage` and navigates users back to the login screen.
- Integration with Firebase:
    - Logs login events for tracking user activity.
    - Updates the last login timestamp in Firebase Realtime Database.
- Security Considerations:
    - Ensures passwords are not stored locally.
    - Uses `AsyncStorage` only for non-sensitive user data.
- Connected Screens:
    - Signup Screen: Allows users to create an account.
    - Profile Screen: Displays user details and provides a logout option.

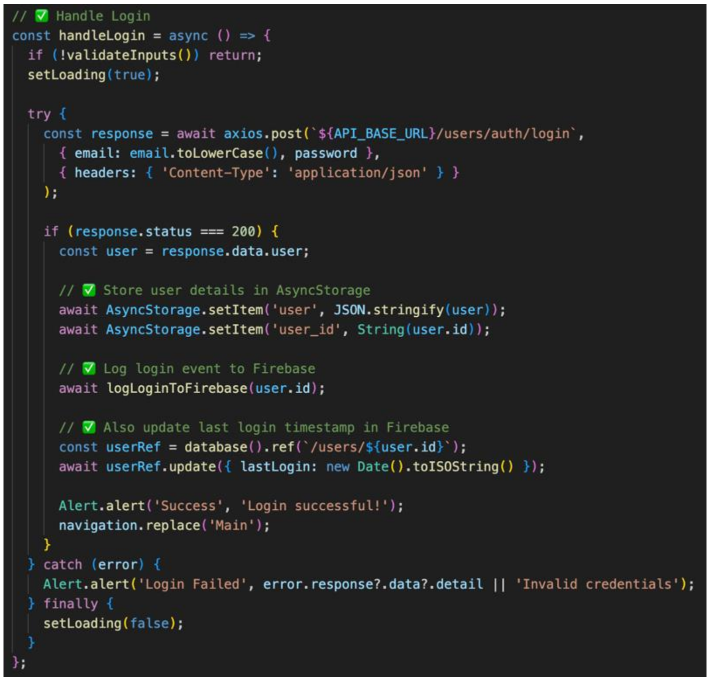
- Settings Screen: Allows users to manage preferences, such as travel style and language.



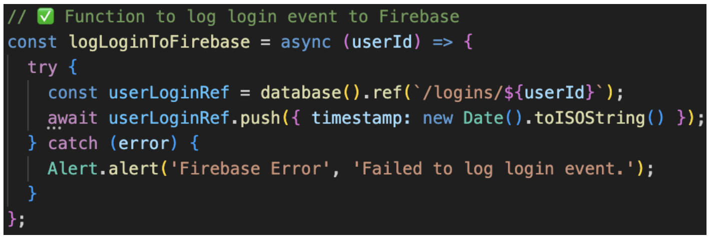
Travel Style: Relaxation

Language: English

Enable Notifications: ●



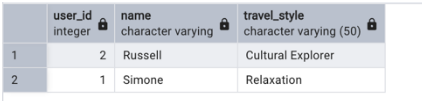
```
// ✅ Handle Login
const handleLogin = async () => {
  if (!validateInputs()) return;
  setLoading(true);

  try {
    const response = await axios.post(`${API_BASE_URL}/users/auth/login`,
      { email: email.toLowerCase(), password },
      { headers: { 'Content-Type': 'application/json' } }
    );

    if (response.status === 200) {
      const user = response.data.user;

      // ✅ Store user details in AsyncStorage
      await AsyncStorage.setItem('user', JSON.stringify(user));
      await AsyncStorage.setItem('user_id', String(user.id));

      // ✅ Log login event to Firebase
      await logLoginToFirebase(user.id);

      // ✅ Also update last login timestamp in Firebase
      const userRef = database().ref(`/users/${user.id}`);
      await userRef.update({ lastLogin: new Date().toISOString() });

      Alert.alert('Success', 'Login successful!');
      navigation.replace('Main');
    }
  } catch (error) {
    Alert.alert('Login Failed', error.response?.data?.detail || 'Invalid credentials');
  } finally {
    setLoading(false);
  }
};
```



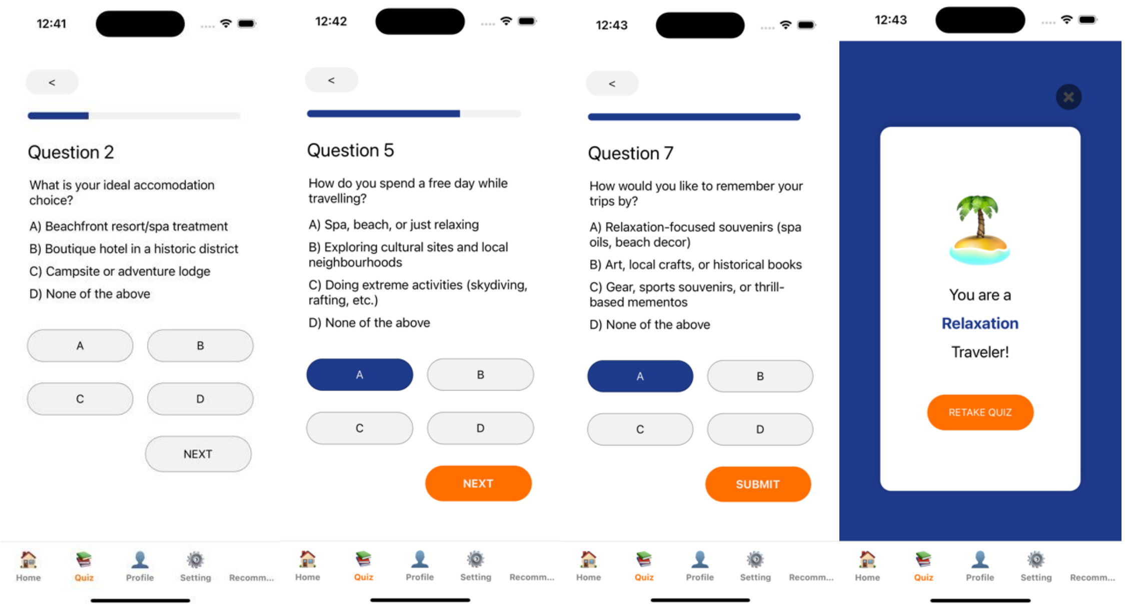
```
// ✅ Function to log login event to Firebase
const logLoginToFirebase = async (userId) => {
  try {
    const userLoginRef = database().ref(`/logins/${userId}`);
    await userLoginRef.push({ timestamp: new Date().toISOString() });
  } catch (error) {
    Alert.alert('Firebase Error', 'Failed to log login event.');
  }
};
```

**<u>Implemented Feature: Travel Style Quiz (Simone)</u>**

- Designed and implemented the travel style quiz screens.
- Navigation through quiz questions
    - Back button:
        - Enabled after the first question has been submitted
        - Saves state of previous question so it is highlighted when navigating back
        - Going back erases the state of questions ahead so upon navigating forwards, questions provide a clean slate for score recalculation
    - Next button:
        - Enabled after an option has been selected
        - Stores the selected answer for the current question and adds score
    - Submit button:
        - Enabled on the last question/replaces the NEXT button
        - Determines the user's travel style based on quiz scoring
        - Retrieves user_id and sends the determined travel style to the backend
- Progress bar at the top for visual representation of quiz completion
- Result screen:
    - Text display changes dynamically based on test results to accomodate all travel styles
    - Retake option which replaces/updates the old travel_style with the new one
        - Resets all previous quiz states
- Screenshots:



| | user_id<br>integer | name<br>character varying | travel_style<br>character varying (50) |
|---|---|---|---|
| 1 | 2 | Russell | Cultural Explorer |
| 2 | 1 | Simone | Relaxation |



12:41

Question 2

What is your ideal accomodation choice?

A) Beachfront resort/spa treatment

B) Boutique hotel in a historic district

C) Campsite or adventure lodge

D) None of the above

12:42

Question 5

How do you spend a free day while travelling?

A) Spa, beach, or just relaxing

B) Exploring cultural sites and local neighbourhoods

C) Doing extreme activities (skydiving, rafting, etc.)

D) None of the above

12:43

Question 7

How would you like to remember your trips by?

A) Relaxation-focused souvenirs (spa oils, beach decor)

B) Art, local crafts, or historical books

C) Gear, sports souvenirs, or thrill-based mementos

D) None of the above

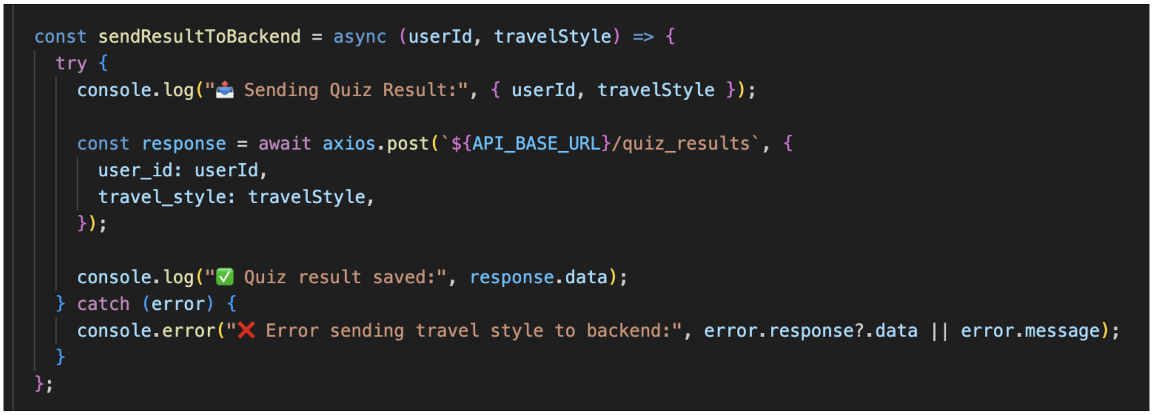
12:43

You are a

Relaxation

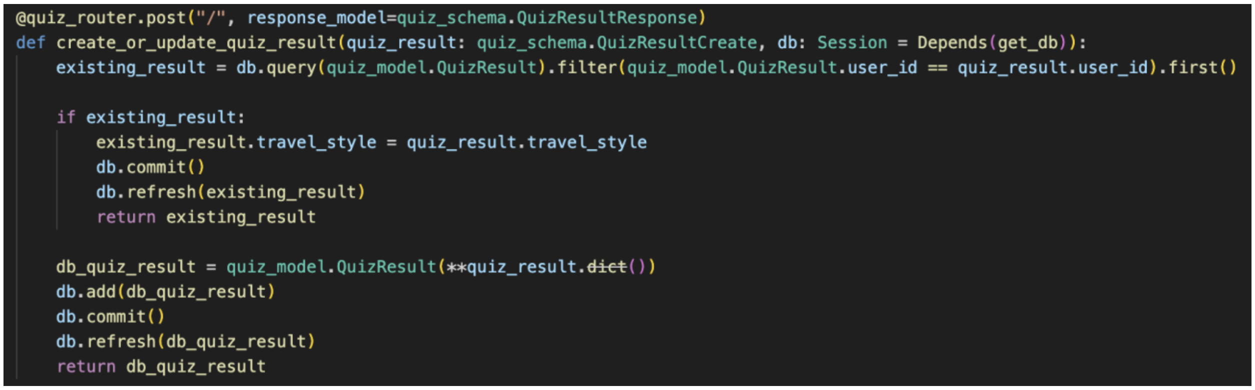
Traveler!

Sending travel style to backend:



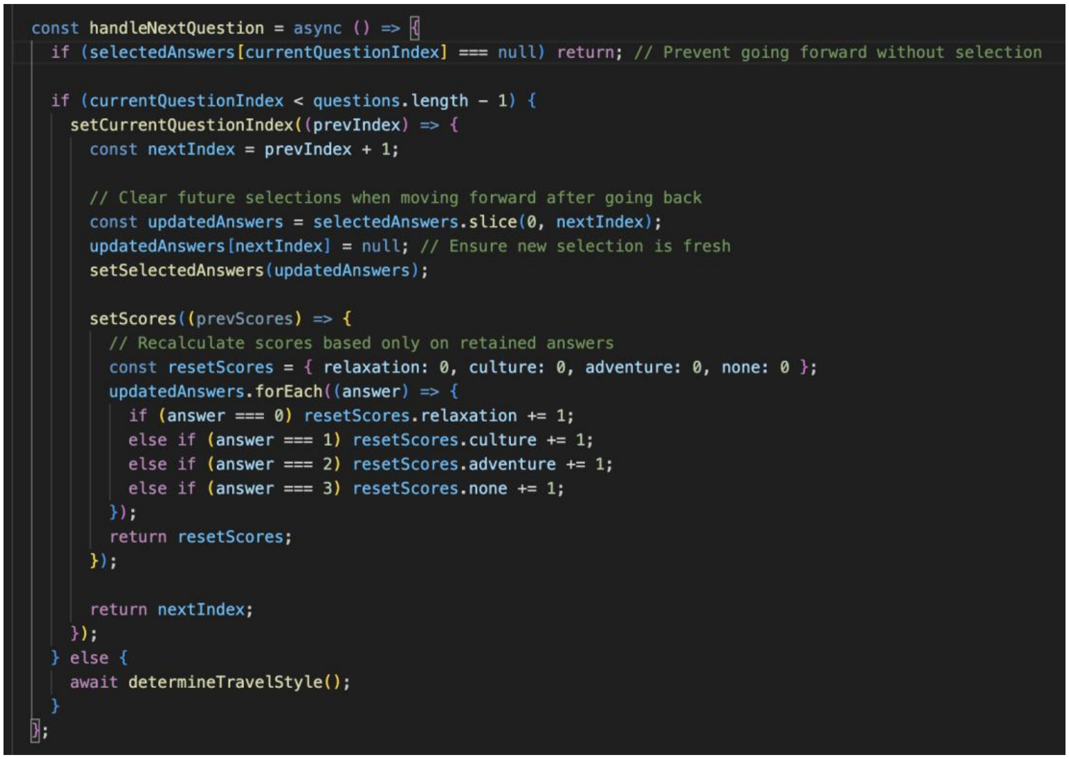
```
const sendResultToBackend = async (userId, travelStyle) => {
  try {
    console.log("📤 Sending Quiz Result:", { userId, travelStyle });

    const response = await axios.post(`${API_BASE_URL}/quiz_results`, {
      user_id: userId,
      travel_style: travelStyle,
    });

    console.log("✅ Quiz result saved:", response.data);
  } catch (error) {
    console.error("❌ Error sending travel style to backend:", error.response?.data || error.message);
  }
};
```

Quiz router to create new record for a user's travel style or update it



```
@quiz_router.post("/", response_model=quiz_schema.QuizResultResponse)
def create_or_update_quiz_result(quiz_result: quiz_schema.QuizResultCreate, db: Session = Depends(get_db)):
    existing_result = db.query(quiz_model.QuizResult).filter(quiz_model.QuizResult.user_id == quiz_result.user_id).first()

    if existing_result:
        existing_result.travel_style = quiz_result.travel_style
        db.commit()
        db.refresh(existing_result)
        return existing_result

    db_quiz_result = quiz_model.QuizResult(**quiz_result.~~dict~~())
    db.add(db_quiz_result)
    db.commit()
    db.refresh(db_quiz_result)
    return db_quiz_result
```

Scoring Logic:



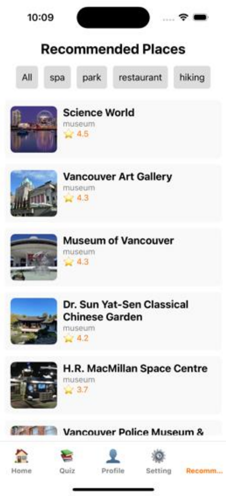
```
const handleNextQuestion = async () => {
  if (selectedAnswers[currentQuestionIndex] === null) return; // Prevent going forward without selection

  if (currentQuestionIndex < questions.length - 1) {
    setCurrentQuestionIndex((prevIndex) => {
      const nextIndex = prevIndex + 1;

      // Clear future selections when moving forward after going back
      const updatedAnswers = selectedAnswers.slice(0, nextIndex);
      updatedAnswers[nextIndex] = null; // Ensure new selection is fresh
      setSelectedAnswers(updatedAnswers);

      setScores((prevScores) => {
        // Recalculate scores based only on retained answers
        const resetScores = { relaxation: 0, culture: 0, adventure: 0, none: 0 };
        updatedAnswers.forEach((answer) => {
          if (answer === 0) resetScores.relaxation += 1;
          else if (answer === 1) resetScores.culture += 1;
          else if (answer === 2) resetScores.adventure += 1;
          else if (answer === 3) resetScores.none += 1;
        });
        return resetScores;
      });

      return nextIndex;
    });
  } else {
    await determineTravelStyle();
  }
};
```

## Implemented Feature: Personal Recommendations (Simone)

- Designed and implemented a personalized recommendations feature using the Google Places API and FastAPI backend.
- Fetching and Displaying Recommendations:
    - Users receive travel recommendations based on:

        - Their travel style (e.g., Relaxation, Adventure, Cultural, Foodie).
        - Their current location (default: Vancouver, BC).
    - Recommendations are retrieved from the FastAPI backend (/places/recommendations), which:
        - Fetches user travel_style stored in PostgreSQL.
        - Returns cached recommendations if available.
        - Calls Google Places API if new recommendations are needed.
    - Recommendations are displayed in a scrollable list with:
        - Place image (Google Places photos, fallback to a placeholder).
        - Place name, category, and rating.
- Optimization and Performance Enhancements:
    - Uses AsyncStorage to retrieve the user's ID for backend queries.
    - API requests are optimized:
        - Caching mechanism prevents redundant calls to Google Places.
        - Dynamic updates only fetch new places when necessary.
    - Displays a loading indicator while fetching recommendations for better UX.
- UI and Interactive Elements:
    - Styled using React Native components and RecommendationsScreenStyles.js:
        - Clean card layout for each recommendation.
        - Image previews from Google Places.
- Integration with Other Features:
    - Works alongside the Travel Style Quiz to provide tailored recommendations.
    - Seamless Google Places API integration for accurate location-based results.

- ○ Designed to complement the interactive map, allowing users to explore locations visually.



Recommended Places

Science World
museum
⭐ 4.5

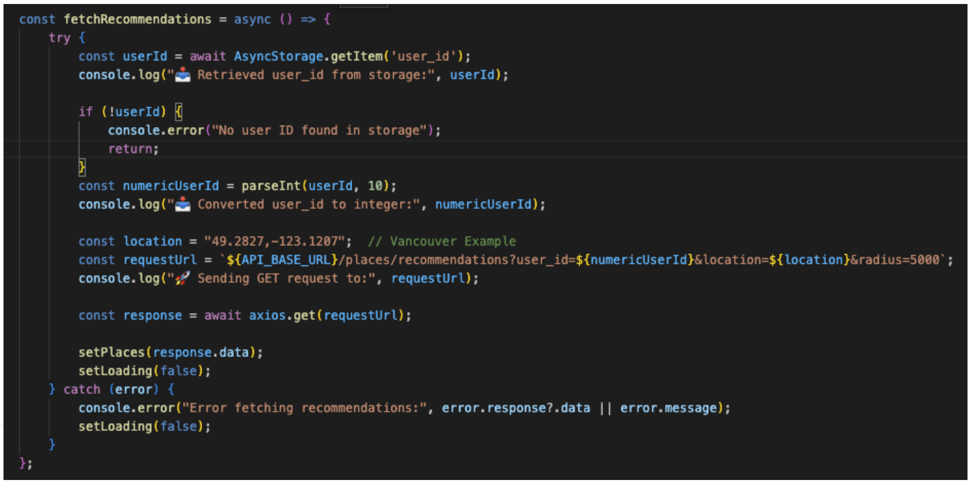
Vancouver Art Gallery
museum
⭐ 4.3

Museum of Vancouver
museum
⭐ 4.3

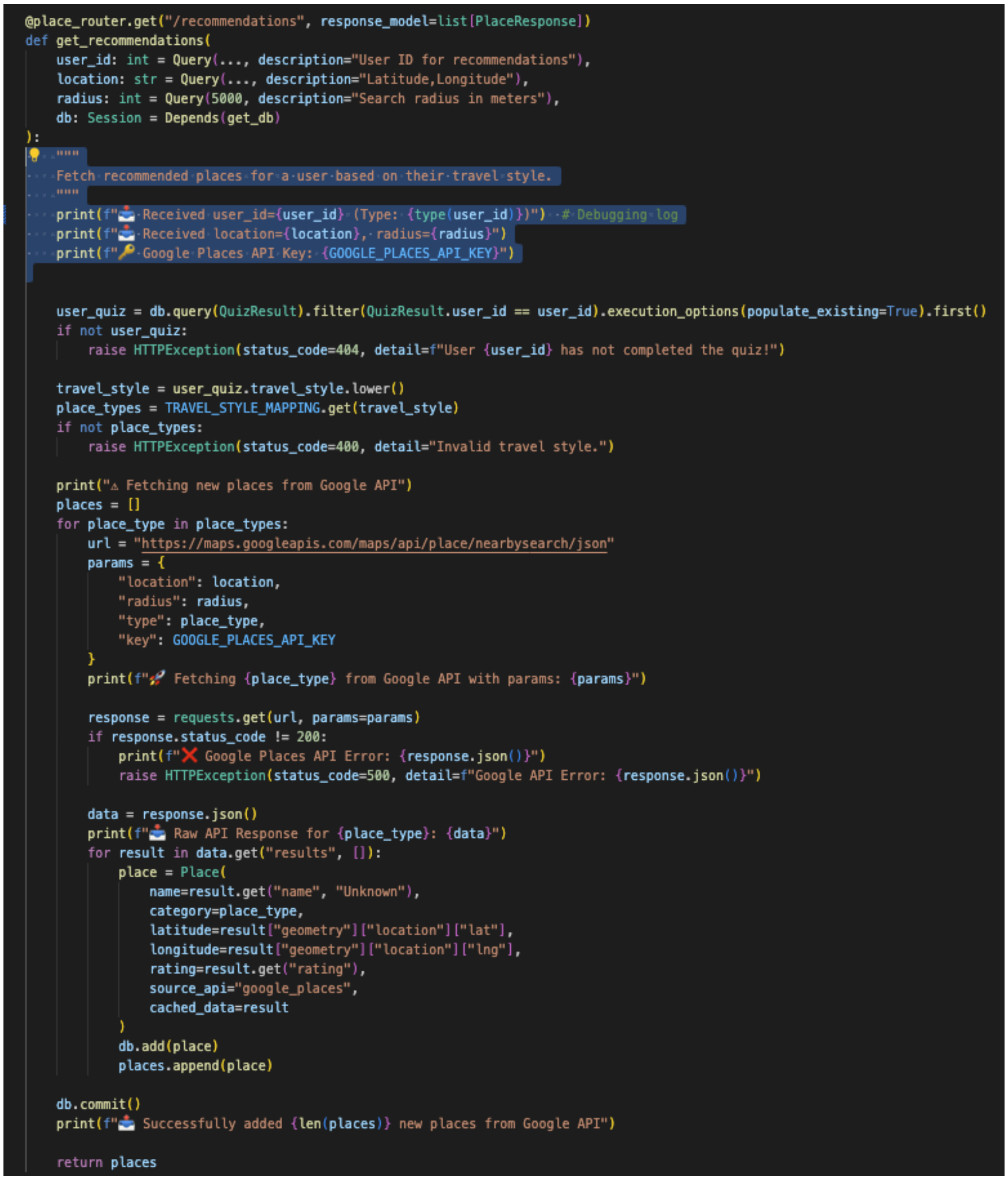
Dr. Sun Yat-Sen Classical Chinese Garden
museum
⭐ 4.2

H.R. MacMillan Space Centre
museum
⭐ 3.7

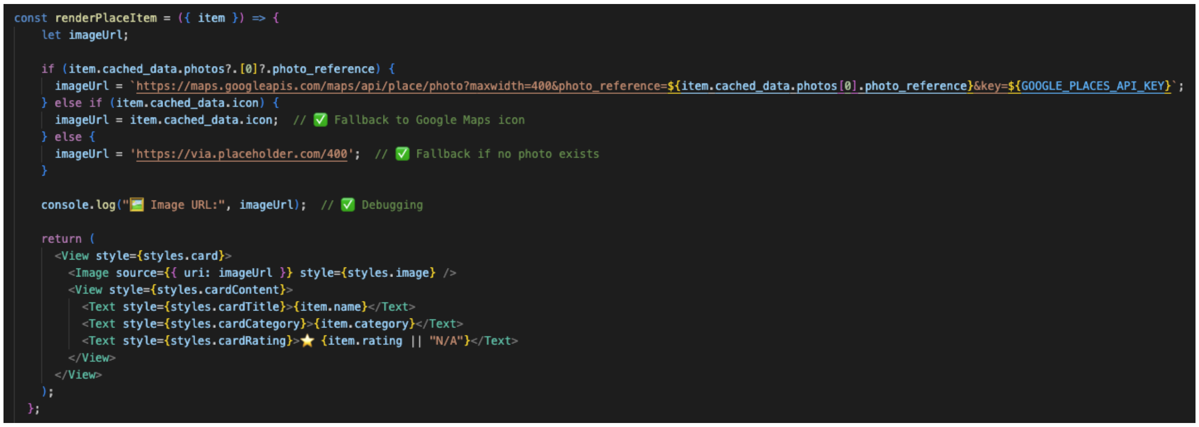
Vancouver Police Museum &



```
const fetchRecommendations = async () => {
    try {
        const userId = await AsyncStorage.getItem('user_id');
        console.log("📥 Retrieved user_id from storage:", userId);

        if (!userId) {
            console.error("No user ID found in storage");
            return;
        }
        const numericUserId = parseInt(userId, 10);
        console.log("📥 Converted user_id to integer:", numericUserId);

        const location = "49.2827,-123.1207";  // Vancouver Example
        const requestUrl = `${API_BASE_URL}/places/recommendations?user_id=${numericUserId}&location=${location}&radius=5000`;
        console.log("🚀 Sending GET request to:", requestUrl);

        const response = await axios.get(requestUrl);

        setPlaces(response.data);
        setLoading(false);
    } catch (error) {
        console.error("Error fetching recommendations:", error.response?.data || error.message);
        setLoading(false);
    }
};
```



```python
@place_router.get("/recommendations", response_model=list[PlaceResponse])
def get_recommendations(
    user_id: int = Query(..., description="User ID for recommendations"),
    location: str = Query(..., description="Latitude,Longitude"),
    radius: int = Query(5000, description="Search radius in meters"),
    db: Session = Depends(get_db)
):
    """
    Fetch recommended places for a user based on their travel style.
    """
    print(f"📥 Received user_id={user_id} (Type: {type(user_id)})")  # Debugging log
    print(f"📥 Received location={location}, radius={radius}")
    print(f"🔑 Google Places API Key: {GOOGLE_PLACES_API_KEY}")


    user_quiz = db.query(QuizResult).filter(QuizResult.user_id == user_id).execution_options(populate_existing=True).first()
    if not user_quiz:
        raise HTTPException(status_code=404, detail=f"User {user_id} has not completed the quiz!")

    travel_style = user_quiz.travel_style.lower()
    place_types = TRAVEL_STYLE_MAPPING.get(travel_style)
    if not place_types:
        raise HTTPException(status_code=400, detail="Invalid travel style.")

    print("⚠ Fetching new places from Google API")
    places = []
    for place_type in place_types:
        url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
        params = {
            "location": location,
            "radius": radius,
            "type": place_type,
            "key": GOOGLE_PLACES_API_KEY
        }
        print(f"🚀 Fetching {place_type} from Google API with params: {params}")

        response = requests.get(url, params=params)
        if response.status_code != 200:
            print(f"❌ Google Places API Error: {response.json()}")
            raise HTTPException(status_code=500, detail=f"Google API Error: {response.json()}")

        data = response.json()
        print(f"📥 Raw API Response for {place_type}: {data}")
        for result in data.get("results", []):
            place = Place(
                name=result.get("name", "Unknown"),
                category=place_type,
                latitude=result["geometry"]["location"]["lat"],
                longitude=result["geometry"]["location"]["lng"],
                rating=result.get("rating"),
                source_api="google_places",
                cached_data=result
            )
            db.add(place)
            places.append(place)

    db.commit()
    print(f"📥 Successfully added {len(places)} new places from Google API")

    return places
```

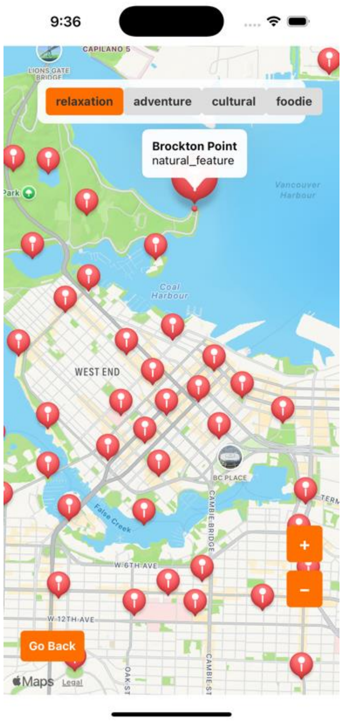


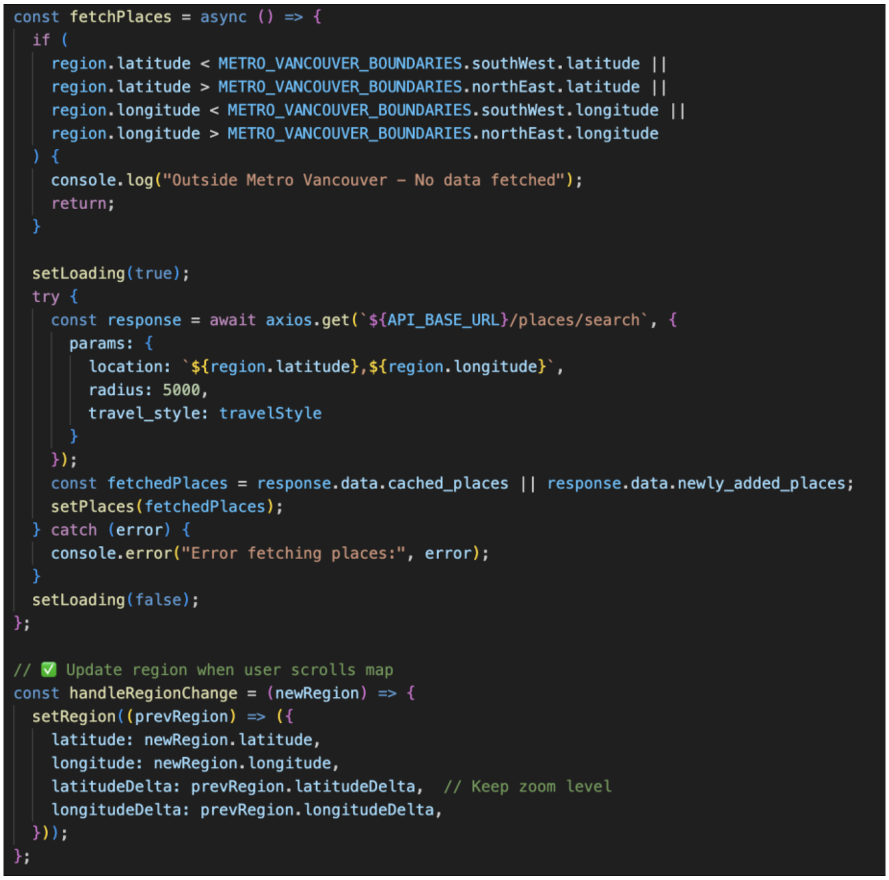
```javascript
const renderPlaceItem = ({ item }) => {
    let imageUrl;

    if (item.cached_data.photos?.[0]?.photo_reference) {
      imageUrl = `https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photo_reference=${item.cached_data.photos[0].photo_reference}&key=${GOOGLE_PLACES_API_KEY}`;
    } else if (item.cached_data.icon) {
      imageUrl = item.cached_data.icon;  // ✅ Fallback to Google Maps icon
    } else {
      imageUrl = 'https://via.placeholder.com/400';  // ✅ Fallback if no photo exists
    }

    console.log("🖼 Image URL:", imageUrl);  // ✅ Debugging

    return (
      <View style={styles.card}>
        <Image source={{ uri: imageUrl }} style={styles.image} />
        <View style={styles.cardContent}>
          <Text style={styles.cardTitle}>{item.name}</Text>
          <Text style={styles.cardCategory}>{item.category}</Text>
          <Text style={styles.cardRating}>⭐ {item.rating || "N/A"}</Text>
        </View>
      </View>
    );
  };
```

## Implemented Feature: Interactive Map (Russell)

- Designed and implemented an interactive map using Google Maps API.
- Map Display and Navigation:
  - The map centers around **Vancouver, BC**, with default coordinates.
  - Users can scroll and zoom to explore different locations.
  - **Region updates dynamically** as users move the map.
  - Zoom controls allow **zooming in and out** with smooth transitions.
  - A **back button** allows users to return to the previous screen.
- Fetching and Displaying Places:
  - The app fetches **places** based on:
  - **User's travel style** (e.g., Relaxation, Adventure, Cultural, Foodie).
  - **Current map region** (latitude, longitude).
- The data is retrieved from the FastAPI backend (`/places/search`), which:
  - Returns cached places if available.
  - Fetches new places if none are cached.
  - Places are displayed as **markers** on the map.
- Filters for Personalized Recommendations:
  - Users can **filter locations** based on their selected travel style.
  - Available filters:
    - Relaxation
    - Adventure
    - Cultural
    - Foodie
  - Selected filters dynamically update the displayed places.

- Real-Time Updates & Optimization:
  - Only fetches places when necessary:
    - If the user moves outside **Metro Vancouver**, fetching is **disabled**.
    - Prevents unnecessary API requests by **checking boundaries**.
  - Uses **loading indicators** while fetching places to improve UX.
- Marker & Callout Features:
  - Each place is marked with a **pin** on the map.
  - Tapping a marker opens a **callout** displaying:
    - **Place name**
    - **Category**
- Integration with Google Maps API:
  - Handles real-time **region updates**.
  - Supports **customized markers** and **callouts**.
  - Uses **Google Maps API key** stored securely in `.env` file.





```
const fetchPlaces = async () => {
  if (
    region.latitude < METRO_VANCOUVER_BOUNDARIES.southWest.latitude ||
    region.latitude > METRO_VANCOUVER_BOUNDARIES.northEast.latitude ||
    region.longitude < METRO_VANCOUVER_BOUNDARIES.southWest.longitude ||
    region.longitude > METRO_VANCOUVER_BOUNDARIES.northEast.longitude
  ) {
    console.log("Outside Metro Vancouver - No data fetched");
    return;
  }

  setLoading(true);
  try {
    const response = await axios.get(`${API_BASE_URL}/places/search`, {
      params: {
        location: `${region.latitude},${region.longitude}`,
        radius: 5000,
        travel_style: travelStyle
      }
    });
    const fetchedPlaces = response.data.cached_places || response.data.newly_added_places;
    setPlaces(fetchedPlaces);
  } catch (error) {
    console.error("Error fetching places:", error);
  }
  setLoading(false);
};

// ✅ Update region when user scrolls map
const handleRegionChange = (newRegion) => {
  setRegion((prevRegion) => ({
    latitude: newRegion.latitude,
    longitude: newRegion.longitude,
    latitudeDelta: prevRegion.latitudeDelta,  // Keep zoom level
    longitudeDelta: prevRegion.longitudeDelta,
  }));
};
```

| Date | Number of Hours | Description of Work Done (Russell) |
|---|---|---|
| Jan 17, 2025 | 1 | Meeting. Distribution of work. Choosing team lead. Decide on the app. -> GitHub Repo -> Misc -> Applied Research_ Logo and Name Research.pdf |
| Jan 18, 2025 | 1 | Planning on the project scope and role distribution. |
| Jan 20, 2025 | 1.5 | Research on screens to have. Finding screens inspiration. App name and branding. |
| Jan 22, 2025 | 2 | Research on Heroku Dynos and Postgres workflow |
| Jan 22, 2025 | 1 | Researched about conducting surveys and user testing. |
| Jan 22, 2025 | 1 | Created survey questionnaires and shared using Microsoft Forms.<br>-> Github Repo -> Misc -> Form_Exploration App Survey_ Help Us Build Your Dream Travel Planner.pdf |
| Jan 25, 2025 | 2 | Commit Git for Proposal.pdf. Researched for APIs: OpenAI gpt model to use, Google Places, Google Maps, Eventbrite, OpenWeatherMap. Get all the API Keys needed. |
| Jan 27, 2025 | 0.5 | Regroup to discuss about the progress and knowledge sharing: Wireframe, APIs. |
| Jan 28, 2025 | 0.5 | Get Free credits from Heroku account using GitHub for Student Developer Pack |
| Jan 28, 2025 | 2 | Researched on Database Schema. Learned more about PostgreSQL vs Microsoft SQL vs MySQL. Then Firebase NoSQL. Learned about Hybrid Architecture Approach with both PostgreSQL and NoSQL. Exported 2 diagrams.<br>-> GitHub Repo -> Misc -> WayPoint-SQL-Schema.png<br>-> GitHub Repo -> Misc -> WayPoint-NoSQL-Schema.png |
| Jan 28, 2025 | 1 | Created Video to share surveys on Instagram. Link: https://www.instagram.com/reel/DFXOyODRLOl/?igsh=ZnF4OWhqbTk0NnVx |
| Jan 29, 2025 | 1 | Get more insights from Prof. Priya in how do we approach surveys collected and user testing if user doesn't reside in a same country. |
| Feb 03, 2025 | 1.5 | Created backend shell. Tested the shell and it's running. Pushed to GitHub repo. |
| Feb 03, 2025 | 1 | Regroup to discuss about Database Schema of PostgreSQL and NoSQL. Update on the frontend shell. Update on the backend shell. Troubleshoot how to run ios on the machine on the first time pulling. Ensure frontend shell can work on both machines.<br>Tasks assignment for the upcoming week. |
| Feb 04, 2025 | 1 | Frontend shell wasn't working on Russell's machine.<br><br>Troubleshooting.<br>Cocoapods installed but pod wasn't installed successfully.<br>Finding: XCode wasn't install properly.<br>When running xcode-select -p on terminal.<br>It showed other thing than "/Applications/Xcode.app/Contents/Developer"<br><br>Details: The problem was that the path to the Xcode command-line tools was not correctly set, causing the xcrun command to be unable to locate the iOS SDK (iphoneos). This resulted in the error message: SDK "iphoneos" cannot be located.<br><br>Code to run:sudo xcode-select -s /Applications/Xcode.app/Contents/Developer<br><br>Solved! |
| Feb 04, 2025 | 2.5 | I encountered an issue while deploying my FastAPI app to Heroku.<br>The deployment failed with a ModuleNotFoundError for the backend module.<br>After investigating, I realized that the folder containing the app was named Implentation with a capital "I", but the Procfile was referencing it as implementation with a lowercase "i". Instead of renaming the folder, I updated the Procfile to correctly reference the folder name with the capital "I" as it appeared in the project. After this update, the deployment was successful, and the app was properly hosted on Heroku. |
| Feb 04, 2025 | 1 | Problem:<br>Setting up PostgreSQL was challenging, especially connecting pgAdmin to both my local database and Heroku’s remote database. I struggled with authentication issues, missing roles, and ensuring my tables were correctly created in both environments.<br>Solution:<br>I configured my local PostgreSQL by setting the correct roles and connected pgAdmin to Heroku using the provided DATABASE_URL. I ensured the database schema was consistent across both environments and created tables using SQLAlchemy.<br>Explanation:<br>This helped me understand how PostgreSQL differs locally and on Heroku, how to manage database credentials, and how to properly set up pgAdmin for database administration. |

| | | |
|---|---|---|
| Feb 04, 2025 | 1 | Problem:<br>Deploying to Heroku failed due to an incorrectly placed Procfile, missing dependencies in requirements.txt, and misconfigured environment variables like DATABASE_URL and SECRET_KEY.<br>Solution: I moved Procfile to the root directory, updated requirements.txt, and set DATABASE_URL correctly in Heroku’s environment variables. Restarting the Heroku dyno applied these fixes.Explanation:This taught me the importance of directory structure and configuration files in deployment and how to debug deployment failures using heroku logs --tail. |
| Feb 04, 2025 | 2 | Problem:<br>After fixing deployment, my app still crashed on Heroku (H10 App Crashed) due to SQLAlchemy not recognizing Heroku’s DATABASE_URL format and FastAPI failing to bind to the correct port.<br>Solution:<br>I modified db.py to convert postgres:// to postgresql://, ensuring SQLAlchemy could connect. I also updated Procfile to bind FastAPI to Heroku’s $PORT.<br>Explanation:<br>This reinforced the differences between local and production environments, the need for dynamic configurations, and how Heroku manages deployments and environment variables. |
| Feb 09, 2025 | 1 | Created travel style quiz. Total 7 questions. Scoring system using point-based system. |
| Feb 10, 2025 | 2 | Regroup / Knowledge Sharing session.<br>Russell:<br>- Explained how the backend works for PSQL part.<br>- Explained how to use pqAdmin tool as database management tool.<br>- Explained  the workflow with backend: when writing new code for backend, test it on local machine before pushing to Github.<br>- Showed how deployment works with Github<br>- Heroku setup.<br>General:<br>- Discussed the next workflow in tackling MVPs.<br>- Discussed to re-order MVP priority based on survey.<br>Task Assignments |
| Feb 11, 2025 | 3.5 | 1. Foreign Key Dependency IssuesProblem: Models had incorrect import order, causing foreign key errors.Solution: Adjusted import order in __init__.py to ensure dependencies load correctly.<br>2. Circular Import IssueProblem: Importing Base from db.py led to circular dependencies. Solution: Moved Base to base.py and updated model imports.<br>3. Missing email-validator ErrorProblem: FastAPI required email-validator, despite being in requirements.txt.Solution: Reinstalled dependencies manually on Heroku.<br><br>4. uvicorn: command not found on HerokuProblem: uvicorn was missing in the runtime environment.Solution: Updated Procfile to use python -m uvicorn and verified installation.<br><br>5. Heroku App Not Restarting ProperlyProblem: Deployment changes weren’t reflecting.Solution: Restarted the app and purged Heroku build cache.<br><br>6. App Not Binding to $PORTProblem: FastAPI wasn’t binding correctly to the environment port.Solution: Ensured uvicorn runs with --port=${PORT} in Procfile.<br><br>7. Database Connection Test Failed on HerokuProblem: Remote database connection wasn’t verifying.Solution: Created /test-db endpoint and confirmed it works. |

| | | |
|---|---|---|
| Feb 11, 2025 | 0.5 | Running db.py to Create Tables on Heroku: Problems & Solutions<br><br>1. Running db.py on Heroku caused ModuleNotFoundError: No module named 'app'<br>Fix: Used PYTHONPATH=. python app/db/db.py to ensure the correct module path.<br>2. Tables were not appearing in Heroku Postgres after running db.py<br>Fix: Explicitly set Base.metadata.schema = "public" in db.py to ensure tables are placed in the correct schema.<br><br>3. Needed a way to manually trigger db.py on Heroku<br>Fix: Opened a Heroku shell with heroku run bash -a waypoint-travel, then executed:PYTHONPATH=. python app/db/db.py<br>4. Wanted to verify if tables were created in Heroku Postgres<br>Fix: Used Heroku Postgres CLI to check tables:heroku pg:psql -a waypoint-travelSELECT tablename FROM pg_tables WHERE schemaname = 'public';<br>Final Outcome:<br>Successfully ran db.py on Heroku, ensuring tables were created in the correct schema. |
| Feb 12, 2025 | 1.5 | Summary of Fixes & Progress<br>1. CRUD Implementation for Users<br>Created POST /users → Create User (with password hashing<br>Created GET /users/{user_id} → Retrieve User by ID.<br>Created PUT /users/{user_id} → Update User (name, email, password).<br>Created DELETE /users/{user_id} → Delete User.<br>2. Fixed Errors<br>InvalidRequestError → Added ForeignKey("users.id") in quiz_model.py.<br>TypeError: 'password' is an invalid keyword argument for User → Ensured password_hash is used in user_model.py.<br>NameError: name 'user_schema' is not defined → Fixed incorrect import in user_routes.py.zsh: no matches found: passlib[bcrypt] → Installed using pip install "passlib[bcrypt]".<br><br>3. Fixed Duplicate URL Path IssueIssue: "/users/users/{user_id}" in FastAPI /docs.Fix: Removed redundant /users prefix from routes in user_routes.py.<br><br>4. Tested LocallyVerified all CRUD operations using FastAPI /docs.<br>Confirmed correct URL paths after fixing duplication.Next StepsTest CRUD operations for Itineraries, Places, Badges, and Quiz Results.<br>Once confirmed, deploy to Heroku and re-test on live API. |
| Feb 12, 2025 | 1 | Log Summary for Places CRUD Implementation<br>1. Implemented CRUD for Places<br>Created POST /places → Add a new place.<br>Created GET /places/{id} → Retrieve a place by ID.<br>Created PUT /places/{id} → Update place details.<br>Created DELETE /places/{id} → Remove a place.<br>2. Fixed IssuesFixed timezone inconsistency → Ensured last_updated is stored in UTC.<br>Resolved datetime.utcnow() deprecation warning → Used datetime.now(timezone.utc).replace(tzinfo=N one).<br>Verified timestamps consistency → Matched last_updated with created_at format.<br><br>3. Successfully TestedPOST /places → Verified place creation with manual data.<br>GET /places/{id} → Retrieved created places correctly.<br>PUT /places/{id} → Updated place details without timezone mismatch.<br>DELETE /places/{id} → Successfully removed places from the database.<br>Next Steps<br>Implement User Favorites (user_favorite_routes.py).<br>Ensure Users ↔ Places relationship works correctly.<br>Prepare for Google Places API integration. |

| | | |
|---|---|---|
| Feb 12, 2025 | 0.5 | Log Summary for User Favorites Implementation<br>1. Implemented CRUD for User Favorites<br>Created POST /user_favorites → Add a place to favorites.<br>Created GET /user_favorites/{user_id} → Retrieve a user's favorite places.<br>Created DELETE /user_favorites/{favorite_id} → Remove a favorite place.<br><br>2. Fixed IssuesValidated user and place existence before adding a favorite.<br>Prevented duplicate favorites by checking existing records.Ensured added_at timestamp is stored in UTC for consistency.<br>3. Successfully Tested<br>POST /user_favorites → Added places to favorites successfully.<br>GET /user_favorites/{user_id} → Retrieved correct favorites for users.<br>DELETE /user_favorites/{favorite_id} → Removed favorites as expected.<br><br>4. Updated main.pyIncluded user_favorite_routes in FastAPI router.<br><br>Next Steps<br>Implement Badges (badge_routes.py).<br>Ensure User ↔ Badges relationship works correctly. |
| Feb 12, 2025 | 1.5 | Set Up Google Places API Integration<br>Chose Google Places API (Old Version) for simpler API key authentication.<br>Tested API manually using Postman & cURL.<br>Implemented FastAPI Route for Places Search<br>Created /places/search endpoint to fetch nearby places.<br>Integrated Google Places API (maps.googleapis.com).<br>Cached results in PostgreSQL to reduce API calls.<br>Restricted API to British Columbia (BC), Canada<br>Implemented latitude/longitude boundary check to block requests outside BC.<br>Verified restriction by testing New York (Successfully blocked).<br>Error Handling & OptimizationsImproved handling for invalid locations and API failures.<br>Implemented database caching to avoid redundant API requests.<br>Added X-Goog-FieldMask to optimize API responses.<br>Tested & Debugged API Responses<br>Verified working results for Vancouver, BC.<br>Ensured API key security using environment variables (.env, Heroku Config Vars). |
| Feb 13, 2025 | 1.5 | Tasks Completed:<br>- Configured React Navigation with StackNavigator & BottomTabNavigator.<br>- Created Login & Signup screens with placeholder values.<br>- Implemented Profile screen with user details, Edit Profile (future), and Log Out.<br>- Built Settings screen with Travel Style, Notifications, Language, and Account Management.<br>- Used SafeAreaView & ScrollView to fix UI layout issues.<br>- Replaced deprecated Picker with @react-native-picker/picker.<br>- Ensured dynamic spacing to prevent overlap with iPhone Dynamic Island.<br><br>Next Steps:<br>- Implement Edit Profile feature.<br>- Add form validation for Login & Signup.<br>- Prepare backend integration for authentication and profile updates.<br>- Enhance UI with better styling. |

| | | |
|---|---|---|
| Feb 13, 2025 | 0.75 | Tasks Completed:<br>- Backend Integration for Authentication:<br>- Reviewed backend schemas, models, and routes.<br>- Confirmed API endpoint for user registration (POST /users/).<br>- Identified and fixed login API endpoint (POST /users/auth/login).<br>- Updated SignupScreen.js:<br>- Connected to backend (POST /users/) for user registration.<br>- Handled form submission, API request, and error handling.<br>- Added navigation to Login screen upon successful signup.<br>- Updated LoginScreen.js:<br>- Integrated POST /users/auth/login using query parameters.<br>- Ensured login request matches the correct FastAPI route.<br>- Redirects users to Main app upon successful login.<br>- Displays alerts for errors and invalid credentials.<br>- Backend API Testing & Debugging:<br>- Successfully tested user registration and login via FastAPI.<br>- Ensured POST /users/auth/login worked with query parameters.<br>- Verified API response handling in React Native app.<br><br>Next Steps:<br>- Implement persistent authentication (store session/token). eg. JWT<br>- Add form validation for signup & login fields.<br>- Enhance UI styling & error messages for better user experience.<br>- Implement Edit Profile feature in ProfileScreen.js. |
| Feb 13, 2025 | 2.25 | Attempted to rename React Native app from "frontend" to "WayPoint" → Encountered issues, reverted to "frontend"<br>Updated package.json and app.json to reflect the correct app name → Reverted due to build errors<br>Checked and updated Xcode Signing & Capabilities → Used free Apple ID for provisioning<br>Attempted to set correct Bundle Identifier for Firebase setup → Reverted due to build failures<br>Installed Firebase dependencies (@react-native-firebase/app) → Successfully installedFixed CocoaPods issues with modular headers → Modified Podfile and ran pod install --repo-update<br>Configured Firebase in AppDelegate.swift → Updated to FirebaseApp.configure()<br>Encountered xcodebuild error code 65 while running iOS build → Attempted multiple fixes<br>Deleted and reinstalled CocoaPods, node_modules, and Xcode DerivedData → No success<br>Manually deleted ios/build/ and cleaned Xcode project → Issue persisted<br>Ran xcodebuild clean and pod install --repo-update → Did not resolve the issue<br>Tried running the app via Metro Bundler (npx react-native run-ios) → Still failed<br>Decided to fully reset the project by deleting and reinstalling all dependencies → Still encountering build issues |

| | | |
|---|---|---|
| Feb 14, 2025 | 4 | Summary Log:<br>Firebase Realtime Database Setup & Next Steps<br><br>Problems:<br>Multiple React-Core dependencies causing conflicts<br>React-RCTAppDelegate not linking correctly<br>FirebaseAuth/FirebaseAuth-Swift.h file not found (even though not needed)<br>Xcode build error: "unable to initiate PIF transfer session"<br>ReactCommon module redefinition error<br>CocoaPods installation issues<br><br>Solutions Attempted:<br>Refactored Podfile to use use_modular_headers! and fixed React-Core conflicts<br>Updated AppDelegate.swift with FirebaseApp.configure()<br>Removed and reinstalled dependencies (node_modules, Pods, Podfile.lock)<br>Cleared Xcode cache (DerivedData, xcodebuild clean)<br>Ensured only needed Firebase modules were installed<br><br>What's Next:<br>Start fresh to ensure a clean build<br>Get React Native running first before adding Firebase<br>Verify Podfile with default settings, then add Firebase<br>Test a basic build (npx react-native run-ios) before integrating Firebase features<br>Implement Firebase Realtime Database CRUD to confirm it works<br><br>Next attempt: Clean setup from the beginning |
| Feb 14, 2025 | 1.5 | Work Log: Firebase Integration in React Native (iOS)<br><br>Firebase Not Initializing (No Firebase App '[DEFAULT]' has been created)<br>Firebase was not auto-detecting GoogleService-Info.plist.<br>Manually initialized Firebase in firebase.js.<br>Firebase connected successfully using manual config.<br><br>Missing or Invalid FirebaseOptions Property 'apiKey' Error<br>Firebase could not find apiKey from GoogleService-Info.plist.<br>Verified plist format and corrected key names.<br>Ensured plist was inside Implementation/frontend/ios/.<br>Linked plist in Xcode under Build Phases → Copy Bundle Resources.<br>Still using manual config; plist auto-detection needs verification.<br><br>Firebase Data Not Appearing in Realtime Database<br>Firebase connection worked, but no data appeared.<br>Updated Firebase database rules to allow reads/writes.<br>Created a test function in LoginScreen.js to write data.<br>Confirmed successful data write to Firebase Console.<br><br>Next Steps<br>Remove manual Firebase config and verify plist auto-detection.<br>Fetch and display a list of data from Firebase.<br>Secure Firebase Database rules based on authentication. |

| | | |
|---|---|---|
| Feb 14, 2025 | 0.5 | Work Log: Firebase Auto-Detection Fix in React Native (iOS)<br>Problems & Solutions<br>Firebase Not Initializing Automatically<br>Firebase was not detecting GoogleService-Info.plist.<br>Manually initialized Firebase in AppDelegate.swift.<br>Confirmed Firebase auto-detection now works.<br>Missing Firebase Setup in AppDelegate.swift<br>React Native Firebase requires Firebase to be initialized in AppDelegate.swift.<br>Added FirebaseApp.configure() inside didFinishLaunchingWithOptions.<br>Restarted the app and confirmed successful Firebase initialization.<br>Plist File Not Being Read by Xcode<br>GoogleService-Info.plist was not linked in Build Phases → Copy Bundle Resources.<br>Manually added the plist file in Xcode.<br>Verified correct plist location in Implementation/frontend/ios/.<br>Next Steps<br>Fetch and display data from Firebase in the app.<br>Secure Firebase database rules based on authentication. |
| Feb 17, 2025 | 1.25 | Meeting Notes Summary (February 17, 2025)<br>1. Upcoming 1-Week Tasks<br>Focus on the next two MVPs:<br>Personalized Recommendations – Google Places API integration for recommendations<br>Interactive Map – Google Maps API integration for visualization<br>2. Past Week Progress Updates<br>Team members shared knowledge and updates on completed tasks.<br>3. Heroku Backend Server Documentation<br>Discussion on CRUD operations for backend endpoints.<br>4. Connecting Simulator to Heroku Server<br>Setting up the React Native simulator to interact with the backend hosted on Heroku.<br>5. Planning for Video Workflow<br>Outlining the video workflow for the mid-term report.<br>Deciding on tools and steps for video creation.<br>6. Firebase Realtime Database & App Distribution<br>Revisiting Firebase Realtime Database setup.<br>Setting up Firebase App Distribution for testing.<br>7. Google Places API on /search Path<br>Integrating Google Places API for search functionality.<br>Ensuring that the API can return filtered results based on user preferences. |

| | | |
|---|---|---|
| Feb 22, 2025 | 2 | Fixes & Improvements in SettingsScreen.js and Backend<br>Initial Issues & Fixes<br>Login Issues (422 Unprocessable Content)<br>Issue: FastAPI rejected login requests due to incorrect request body formatting.<br>Fix: Ensured email and password were correctly passed in the axios.post request in LoginScreen.js.<br>User Data Not Persisting After Login<br>Issue: Logged-in user details were not being saved for profile and settings.<br>Fix: Stored user data in AsyncStorage after a successful login.<br>Navigating to Home Screen After Login (REPLACE Error)<br>Issue: navigation.replace('HomeScreen') failed due to missing screen.<br>Fix: Updated App.js to correctly route users to Main after login.<br>Backend Issues & Fixes<br>Travel Style Not Saving (422 Unprocessable Content)<br>Issue: FastAPI expected user_id in the request body for PUT requests.<br>Fix:<br>Created QuizResultUpdate schema to accept only travel_style.<br>Modified PUT /quiz_results/user/{user_id} to update travel_style correctly.<br>Fetching Travel Style for User Settings<br>Issue: Travel style was not being retrieved from PostgreSQL.<br>Fix:<br>Created GET /quiz_results/user/{user_id} to fetch travel_style.<br>Updated SettingsScreen.js to call this API and store the result in AsyncStorage.<br>Frontend UI Issues & Fixes<br>VirtualizedLists Error (Nested inside ScrollView)<br>Issue: DropDownPicker (using FlatList) conflicted with ScrollView.<br>Fix:<br>Replaced ScrollView with KeyboardAvoidingView.<br>Set removeClippedSubviews={false} and adjusted zIndex.<br>Dropdown Overlapping UI<br>Issue: Travel Style dropdown was overlaying the Language dropdown.<br>Fix:<br>Wrapped dropdowns in View with zIndex.<br>Used modalProps={{ animationType: 'fade' }} to prevent overlap.<br>Language Change Incorrectly Updating Travel Style<br>Issue: Changing language was overwriting travel_style in AsyncStorage.<br>Fix:<br>Updated updateLanguage function to modify only language, preserving travel_style. |
| Feb 22, 2025 | 1 | Drafted video recording flow. |
| Feb 22, 2025 | 0.5 | Fixed QuizScreen and LoginScreen to ensure first time user is able to take the quiz and save it correctly. |
| Feb 23, 2025 | 3 | Google Maps Integration on iOS<br>Problem: Needed to integrate Google Maps SDK for iOS<br>Solution: Installed and configured Google Maps SDK (v7.0.0).<br>Problem: API key was stored in Info.plist, causing security issues<br>Solution: Moved API key to .env and dynamically loaded it in AppDelegate.swift.<br>Problem: "Tried to register two views with the same name AIRMap" error<br>Solution: Ensured only one instance of react-native-maps to fix duplicate registration.<br>Problem: API key needed to be dynamically passed to Swift<br>Solution: Updated Podfile to load .env variables and inject the API key into the build.<br>Problem: Map was not displaying if the API key was missing<br>Solution: Added error handling to InteractiveMapScreen.js to show a message when the API key is missing.<br>Problem: Needed to verify Google Maps displayed properly<br>Solution: Successfully displayed Google Maps centered on Vancouver, BC in the iOS simulator. |
| Feb 23, 2025 | 2 | Added markers with Google Places API. |
| Feb 24, 2025 | 1 | Check on Firebase app Distribution.<br>Upon checking, need to enrol in Apple Developer Program.<br>Completed: Register for enrolment under educational institution. Request to waive the fee.<br>What's next: Waiting for reply. |
| Feb 24, 2025 | 1.75 | Video Recording for Mid Term Report Checkpoint |

| Date | Number of Hours | Description of Work Done **(Simone)** |
|---|---|---|
| Jan 17, 2025 | 1 | Meeting. Distribution of work. Choosing team lead. Decide on the app. -> GitHub Repo -> Misc -> Applied Research_ Logo and Name Research.pdf |
| Jan 20, 2025 | 1 | Project Proposal Draft writing - Started the introduction section. |
| Jan 21, 2025 | 2 | Project Proposal writing - Finalizing draft writing -> Proposed Research Project, Project Planning and Timeline, Project Contract<br>- Proposal in Github Repo -> ReportsAndDocuments -> SimoneL_Proposal.pdf |
| Jan 22, 2025 | 2 | Figma initialization, developing wireframes, general idea of screens |
| Jan 22, 2025 | 1 | Wireframe for homepage |
| Jan 24, 2025 | 1 | Wireframing for quiz screens |
| Jan 25, 2025 | 0.5 | Finalized proposal, ready for submission |
| Jan 27, 2025 | 2 | Wireframing screens: interactive maps, chatbot, my trips -> Github Repo -> Misc -> Figma Wireframes.png |
| Jan 27, 2025 | 0.5 | Regroup to discuss about the progress and knowledge sharing: Wireframe, APIs. |
| Jan 31, 2025 | 3 | Research on Reactive Native Expo vs CLI, environment setup and ways to start a project + gluestack v2 ui. Decided on React Native CLI as it was more suitable for the project scope |
| Feb 01, 2025 | 2 | Started and added frontend project shell to github. Encountered issues with the folder structuring when adding to github: Empty folder was being pushed to github instead of with the code. Had to restructure multiple times before successful |
| Feb 03, 2025 | 1 | Regroup to discuss about Database Schema of PostgreSQL and NoSQL. Update on the frontend shell. Update on the backend shell. Troubleshoot how to run ios on the machine on the first time pulling. Ensure frontend shell can work on both machines.<br>Tasks assignment for the upcoming week. |
| Feb 06, 2025 | 1.5 | Started on the Quiz Screen component: Adding pressable buttons and sorting out the general layout of the text and buttons |
| Feb 07, 2025 | 2.5 | Completed QuizScreen and the styling file for it.<br>Added:<br>- Back/Next button for navigation between questions.<br>- Back button not visible on the first question<br>- Progress bar to visually represent questions done/left<br>- Color change when button is selected<br>- Have a couple of example questions set as placeholders |
| Feb 08, 2025 | 2 | Started on the layout of the HomeScreen as well as its corresponding styling file.<br>Added:<br>- Search bar (text input)<br>- Horizontal scroll for ‘My Trips’ (with placeholder trips set + default card for when there are no current trips added)<br><br>Problems encountered:<br>- Had trouble with using Carousel (react-native-snap-carousel)<br>- The dependencies had conflicts and version compatibility issues, specific issues with propTypes<br>- Nothing was working<br>Alternate solution:<br>- Used FlatList with horizontal scrolling instead |
| Feb 08, 2025 | 2 | Started on the bottom navigation menu -><br>Problem: Tried using external library for icons but icons would not show up properly (Question mark in the middle of a box shows up)<br>- Issues with linking fonts with Xcode<br>Alternate plan: For now I am using emojis as a placeholder icon, will revisit adding external library of icons at a later time<br>Bottom Navigation now has seamless navigation between Home and Quiz screens -> Quiz screen to be replaced at a later time, currently acting as a placeholder for other screens |
| Feb 10, 2025 | 2 | Regroup / Knowledge Sharing session.<br><br>Simone:<br>- Explained bottom navigation menu.<br>- Explained Quiz screen and navigation. |

| | | |
|---|---|---|
| | | General:<br>- Discussed the next workflow in tackling MVPs.<br>- Discussed to re-order MVP priority based on survey.<br><br>Task Assignments |
| Feb 11, 2025 | 2.5 | Started on the backend for the quiz MVP -> quiz model, schema, api and routing<br><br>Problem: Had trouble with the database setup/running it on my local machine and connecting to postgresql<br><br>Solution: After consulting with Russell on the setup, server is able to run on my local device. Next step is to make sure that front end is connecting properly to the back end and able to pass information -> update to local database and make sure it is functioning as intended |
| Feb 13, 2025 | 3.5 | - Built the look for the quiz results on Figma<br>- Adding the code and styling for the results display to QuizScreen.js and QuizScreenStyles.js<br>- Will dynamically display the type of traveller depending on the quiz scoring logic from the quiz<br>- Added an “x” button that will take user back to the home page after seeing the results<br><br>Problems Encountered:<br>- Tried to do a share button functionality where user will be able to save the results as an image to their camera roll. This didn’t work out as there were Xcode dependencies that conflicted with my system. Had to remove this functionality. Can revisit once the MVP is complete<br>- Dependencies that were added to do this share implementation were not completely erased and repeatedly threw errors as it was still somewhere in the files.<br><br>Solutions and Alternatives:<br>- Had to remove node modules and reinstall dependencies and pod files multiple times to get rid of the errors.<br><br>Next Steps:<br>- Send quiz results to the backend and save it to the database<br>- Have the option to retake quiz and erase results previous from the backend<br>- (Optional once MVP is complete) Retry the share functionality |
| Feb 15, 2025 | 2 | Trying to send data to the backend and save the travel style pertaining to the user<br>- Using AsyncStorage to save and retrieve the user_id upon successful logins<br>Problem:<br>- There were AsyncStorage runtime errors due to it not being properly linked<br>- This was due to Cocoapods gem error -> broken or outdated Ruby gem<br>Solution:<br>- Reinstalled and updated ruby gem<br>- Reinstalled cocoa pods<br>- Reinstalled iOS pods<br>- No runtime errors for AsyncStorage now<br><br>Trying to retrieve userId from the user’s login session<br>- Created a method in QuizScreen.js to retrieve userId<br>- Successfully able to retrieve userid<br><br>Trying to send quiz results to the backend<br>- Modified api for posting quiz results<br>- Created method in QuizScreen.js to send results to the backend<br>Problem:<br>- 404 Error: api endpoint does not exist or incorrect<br><br>Next step: Fix the connection to API endpoint |
| Feb 16, 2025 | 1.5 | Fixing the connection to API endpoint<br>- Need to send the quiz results to the back end<br>Problem:<br>- 404 Error: api endpoint does not exist or incorrect<br>Solution: |

| | | |
|---|---|---|
| | | - main.py had “/quiz_results” as the router prefix<br>- quiz_routes.py had “/quiz_results” as the route<br>- To send the results to the quiz_results table, the quiz results endpoint should be `${API_BASE_URL}/quiz_results/quiz_results` instead of ‘${API_BASE_URL}/quiz_results`<br>- Changed the endpoint to “/“ --> proper full endpoint is now `${API_BASE_URL}/quiz_results/`<br>- The results are now stored in the quiz_results table with the correctly retrieved userId and travelStyle<br><br>When the user redoes the quiz, the results save as a new row in the table rather than updating the existing record<br>- Need to update the record instead of making a new row<br>- Modified POST “/“ in quiz_routes.py to check if a result already exists for the user, update it if it exists<br>- Now successfully modifies the record instead of creating a new one<br><br>Next Steps:<br>- Add frontend option for user to retake quiz.<br>- Fix scoring logic for when users navigate backwards during the quiz.<br>- Users should not be able to move on to the next question without selecting an answer |
| Feb 17, 2025 | 1 | Added frontend option for user to retake quiz:<br>- Added retake quiz button code and styles<br>- Added a method to trigger onPress to handle resetting all scores quiz UI<br>- New quiz results are being sent to backend, updating the user’s travel style<br><br>Next/Submit buttons are ‘disabled’ without selecting an answer<br>- Added disabled button styling<br>- Modified handleNextQuestion() to prevent moving forward when selectedAnswer is null<br><br>Next Steps:<br>- Fix scoring logic for when users navigate backwards during the quiz<br>- When nagivating backwards through questions, retain previously selected answer |
| Feb 17, 2025 | 0.5 | Analyzed and summarized survey results to determine high/medium/low priority features for the application. |
| Feb 17, 2025 | 1.25 | Navigating backwards through questions, UI retains the previously selected answer<br>- Used an array to track selected answers instead of a single selectedAnswer state<br>- Updated handleAnswerSelection to store answers in the new array (which the index corresponds to the question index)<br>- Set the selected answer when navigating to the previous question<br>- Modify handlePreviousQuestion to retrieve and display the stored selection.<br>Problem:<br>- Navigating forwards AFTER navigating backwards retains the selected answer previously selected<br>Solution:<br>- When moving forward after going backwards, clear future selections to ensure past choices aren’t remembered.<br><br>Fix scoring logic for when users navigate backwards during the quiz. The current scoring logic does not automatically adjust the previous selection's score when changing an answer. Instead, it only adds points when selecting an option but does not remove points when an answer is changed<br>- Modified handleAnswerSelection to first try to subtract points from the answers that are being changed<br>Problem:<br>- This method of scoring retained the scores from the questions that remained unchanged and added on an additional point for the new selection (whether from the same category or not)<br>Solution:<br>- Modify handleAnswerSelection to build scores from the remaining selections instead of subtracting points one by one<br>- Modify handlePreviousQuestion and handleNextQuestion to reset scores of future selections when moving forward after going back. |
| Feb 17, 2025 | 1.25 | Meeting Notes Summary (February 17, 2025)<br>1. Upcoming 1-Week Tasks |

| | | |
|---|---|---|
| | | Focus on the next two MVPs:<br>Personalized Recommendations (Google Places API integration for recommendations)<br>Interactive Map (Google Maps API integration for visualization)<br>2. Past Week Progress Updates<br>Team members shared knowledge and updates on completed tasks.<br>3. Heroku Backend Server Documentation<br>Discussion on CRUD operations for backend endpoints<br>4. Connecting Simulator to Heroku Server<br>Setting up the React Native simulator to interact with the backend hosted on Heroku.<br>5. Planning for Video Workflow<br>Outlining the video workflow for the mid-term report.<br>Deciding on tools and steps for video creation.<br>6. Firebase Realtime Database & App Distribution<br>Revisiting Firebase Realtime Database setup.<br>Setting up Firebase App Distribution for testing.<br>7. Google Places API on /search Path<br>Integrating Google Places API for search functionality.<br>Ensuring that the API can return filtered results based on user preferences. |
| Feb 20, 2025 | 2 | Implementing personalized recommendations feature<br>- Built a FastAPI route (/places/recommendations) to fetch places based on user preferences.<br>- Integrated Google Places API to get places based on user travel style (quiz results).<br>- Stored cached places in PostgreSQL to reduce API calls.<br>Problem 1: 422 Error (Unprocessable Entity)<br>- FastAPI expected user_id as an integer, but it was sent as a string.<br>Solution:<br>- Ensured user_id is an integer<br><br>Problem 2: 404 Error<br>- FastAPI couldn't find user_id in quiz_results, even though it existed in PostgreSQL<br>- The backend was using the local database instead of Heroku.<br>- FastAPI was querying the wrong database (local waypoint_db instead of Heroku).<br>Solution:<br>- Connected FastAPI to Heroku Postgres by updating .env database_url<br><br>Problem 3: API Route Conflict<br>- FastAPI treated /recommendations as {place_id}, causing a 400 Bad Request.<br>- The dynamic route @place_router.get("/{place_id}") was above /recommendations, so FastAPI assumed "recommendations" was a place_id.<br>Solution:<br>- Reordered routes in place_routes.py<br>!!Confirmed API is working!!<br>- FastAPI backend is using Heroku Postgres instead of local PostgreSQL.<br>- Recommendations API (/places/recommendations) now works as expected. |
| Feb 20, 2025 | 2 | Built Recommended Places screen and styling<br><br>Connecting frontend UI to make API requests to display the recommendations:<br>- Developed methods to fetch recommendations<br>- Developed filter options<br>- Switched back to local to work on the app<br>Problem: app UI was not displaying results<br>- The frontend was now receiving an empty array upon making requests<br>Solution:<br>- Debugging logs indicated that google places was denying the requests due to invalid api key<br>- Realized Heroku was configured with google places api but not locally<br>- Added API key in .env and was able to retrieve the recommendations which reflected in the UI<br><br>Next Steps:<br>- Images for the recommended places are not found- need to fix<br>- Refine filters<br>- Make sure recommendations work for all travel styles |
| Feb 21, 2025 | 2 | Fixing the photo display on the recommendations screen<br>- Logs show that the image can not be found<br>- At first, thought it was the API key not being used in the frontend so it was restricting access |

| | | |
|---|---|---|
| | | to images, but upon correcting that, images were still not loading<br>- Adjusted the method for image rendering a couple times and one finally worked (not too sure why)<br>- Images now loading with the recommendations |
| Feb 21, 2025 | 1 | Making sure recommendations work for all travel styles:<br>- Matching the different travel styles correctly to TRAVEL_STYLE_MAPPING<br>- Had to rename the some of the keys to match the recognized travel styles<br><br>Adjusting the displayed emoji for each travel style<br>- Tried to do multiple emojis for combined travel styles but it offset the styling<br>  - Alternative: Selected one emoji best fit for different combined styles instead<br>- Fixed the missing emoji for “No travel style”<br>- Fixed the awkward wording for some of the travel styles such as "You are a You didn't align with any specific travel style Traveler"<br>  - Modify the return inside QuizScreen.js so that the formatting dynamically adjusts<br><br>Next Steps:<br>- Recommendations displaying for mixed travel styles<br>- Recommendations fetch new results after travel style changes<br>- Make sure the filters work |
| Feb 23, 2025 | 4 | Midterm Report: Title page (complete), Introduction (complete), summary of initial proposed project (complete), changes to the proposal (complete), UPDATED project planning and timeline (IN PROGRESS: individual responsibilities; COMPLETED: Gantt chart, new proposed timelines, milestones, deiverables), Implemented Feature (IN PROGRESS: Login, Recommendations, Interactive Maps; COMPLETED: Quiz), Work Logs (IN PROGRESS), Closing and References (IN PROGRESS)<br><br>Proposal changes:<br>- Gantt Chart<br>- Responbilities<br>- Timelines<br>- MVPs Priority |
| Feb 24, 2025 | 1.75 | Video Recording for Mid Term Report Checkpoint |
| Feb 24, 2025 | 2.5 | Finishing and finalizing midterm report. Preparing documents for submission. |

## Closing and References

- Google Maps SDK for iOS: https://developers.google.com/maps/documentation/ios-sdk/overview
- Google Places API: https://developers.google.com/maps/documentation/places/web-service/overview
- Firebase Realtime Database: https://firebase.google.com/docs/database
- Firebase Installation: https://firebase.google.com/docs/ios/setup
- React Native Maps for iOS: https://github.com/react-native-maps/react-native-maps
- React Native CLI: https://reactnative.dev/docs/environment-setup
- FastAPI: https://fastapi.tiangolo.com/
- Xcode: https://developer.apple.com/xcode/
- Apple Developer Program: https://developer.apple.com/programs/
- SQLAlchemy: https://docs.sqlalchemy.org/
- Pydantic: https://docs.pydantic.dev/latest/
- email-validator: https://pypi.org/project/email-validator/