ANNA UNIVERSITY REGIOAN CAMPUS COIMBATORE

Wanderlust: A Personalized Travel Planning and Tracking App

PRAVEEN.S

NM_id:9EE35E88AA21DD8852A49B5A13E29CD6

BALAVENGATESH.S

NM_id: 04A6612D5DD604F9A32436734B2ED0F0

KABISH.D

NM_id: 1DE40F7B3F53A14B49944329E25574F2

ADITHYAN.C

NM_id: 46C200E9420CCE79223462BC43FECA50

Wanderlust: A Personalized Travel Planning and Tracking App

Introduction

The travel industry is increasingly leaning towards mobile platforms to provide users with a more personalized and convenient experience. This project aims to develop a simple travel app using Android's Jetpack Compose UI toolkit to showcase recommended accommodations based on user-selected locations. With Compose, the project explores declarative UI principles to build a seamless, dynamic, and visually appealing user interface that adapts to user preferences.

Objectives:

1. **Personalization:**
    - Provide customized recommendations for destinations, itineraries, accommodations, and activities based on user preferences, budget, and travel history.
2. **Simplified Planning:**
    - Offer tools to streamline the planning process, such as itinerary creation, budget tracking, and booking integration.

3. **Real-Time Assistance:**
    - Enable real-time travel tracking, notifications for itinerary changes, and updates about local events or alerts at the destination.
4. **Engagement and Social Sharing:**
    - Facilitate social interaction by allowing users to share their travel plans, experiences, and reviews with friends and fellow travelers.
5. **Data Analytics and Insights:**
    - Provide insights into travel trends, budget optimization, and journey statistics to improve future travel experiences.

Functionalities:

1. User Account and Profile Management:

- User registration/login via email, phone, or social media accounts.
- Profile customization to include travel preferences (e.g., preferred destinations, activities, budget).
- Option to save past trips and frequently visited locations.

2. Travel Planning:

- **Destination Search and Recommendations:**

    - Suggestions based on interests, weather, budget, and trends.
    - AI-powered destination matching to user preferences.
- **Itinerary Builder:**
    - Drag-and-drop interface to add attractions, activities, and meals.
    - Integration of travel times, operating hours, and estimated costs.
- **Budget Tracker:**
    - Allocate a budget per trip and track expenses in real-time.
    - Integration with currency converters.

3. Booking Integration:

- One-click booking for flights, hotels, and activities via third-party APIs (e.g., Skyscanner, Booking.com).
- Centralized access to booking details, including tickets and confirmations.

4. Real-Time Tracking and Alerts:

- GPS-based location tracking for navigation and progress monitoring.
- Notifications for flight delays, weather changes, or local emergencies.
- Map integration for viewing planned routes and nearby attractions.

5. Social and Sharing Features:

- Option to share itineraries and experiences with friends and family.
- A built-in travel journal for documenting trips with photos and notes.
- Integration with social media platforms for sharing highlights.

6. Analytics and Insights:

- Post-trip statistics: distance traveled, budget adherence, and time spent.
- Personalized suggestions for future trips based on user travel history.
- Tracking of visited locations with a “travel map” to visualize destinations covered globally.

7. Offline Mode:

- Ability to access itineraries, maps, and essential trip details without an internet connection.

8. Multilingual and Multi-Currency Support:

- Support for multiple languages and real-time currency conversion for international travelers.

Tools Used

1. **Android Studio**: The primary IDE used for developing Android applications, providing a comprehensive environment with support for Kotlin, Jetpack Compose, and various Android SDK tools.
2. **Kotlin**: The programming language used for the entire project, known for its expressive syntax and modern features that streamline Android development.
3. **Jetpack Compose**: Android's modern UI toolkit for creating native interfaces in a declarative way, enabling the creation of reusable UI components.
4. **Jetpack Compose Navigation**: A navigation component designed for Compose, allowing seamless transitions between different screens within the app.
5. **Material Design Components**: Compose's Material Design library provides pre-built UI components, such as buttons, cards, and text fields, ensuring that the app adheres to Android's design guidelines.
6. **Android Architecture Components**: Used for managing data and lifecycle, including ViewModel for handling UI-related data and LiveData or State for observing data changes.

7. **Coroutines**: Kotlin's asynchronous programming tool used for managing background tasks like data fetching, ensuring smooth and responsive UI interactions.

Key Features

1. **Compose UI for Declarative Interface**:
    - The app's UI is built using Composable functions that define sections, such as the home screen, location selector, and accommodation recommendations. ◦ UI state is managed directly in the code, simplifying UI updates as data changes.
2. **Personalized Feed**:
    - A dynamic feed presents accommodation recommendations based on the user's location preferences, making the app more engaging and relevant to users.
3. **Location-Based Recommendations**:
    - The app fetches recommendations for locations chosen by the user, providing a personalized experience tailored to travel interests.

4. **Material Design Integration**:
    - Compose's Material Design components (e.g., Button, Card, Text, LazyColumn) ensure a cohesive, user-friendly design following Android's design guidelines.
5. **Responsive Layouts**:
    - The app supports various screen sizes and orientations, providing a responsive experience on both phones and tablets.

Challenges

1. **Learning Curve with Jetpack Compose**:
    - Jetpack Compose is relatively new, and its declarative paradigm requires a shift from the traditional XML-based layouts. Understanding how to manage UI state effectively within Composable functions presented an initial learning challenge.
2. **State Management**:
    - Compose's State and remember functions simplify state management, but keeping track of different states across multiple screens can be challenging, especially for beginners. Proper usage of ViewModel with Compose was essential to

maintain separation of concerns and ensure data consistency across recompositions.

3. **Navigation and Passing Data**:
    - Compose Navigation differs from the traditional navigation component, requiring a different approach for managing navigation routes and passing data between screens. This required adjustments to handle arguments and retain data states between different composables.
4. **Performance Optimization**:
    - Ensuring smooth performance, especially with dynamic lists and images, required optimization. Lazy-loading lists with LazyColumn and proper use of remember were necessary to prevent unnecessary recompositions and improve app responsiveness.
5. **Handling UI Responsiveness**:
    - Developing a responsive design that adapts well to different screen sizes was a challenge, as Compose layouts are structured differently than XML-based layouts. Testing the UI on various devices and configurations helped ensure a consistent experience.

Potential Extensions

- **Map Integration**: Adding maps using Google Maps or Mapbox would enable users to view accommodation locations visually, enhancing the user experience.
- **User Profiles and Authentication**: Implementing user accounts to save preferences, track past bookings, and improve recommendations.
- **Real-Time Location Tracking**: Incorporating GPS data to suggest nearby accommodations and attractions based on the user's current location.
- **Backend Integration**: Connecting to a backend API for real-time data on accommodations, user preferences, and recommendations.

Program

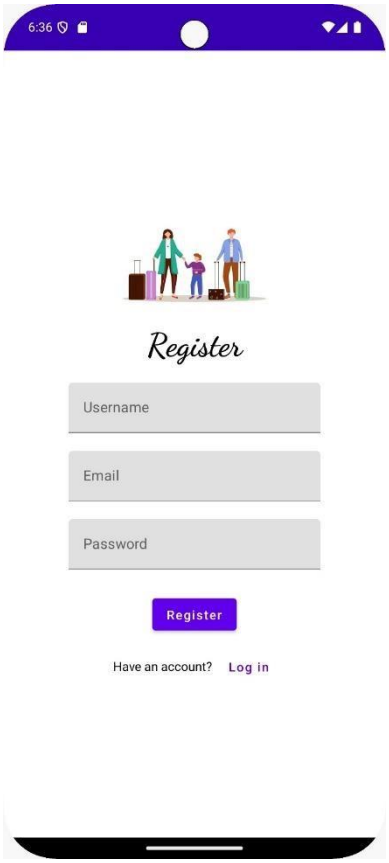
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">

   <application        android:allowBackup="true"
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
```

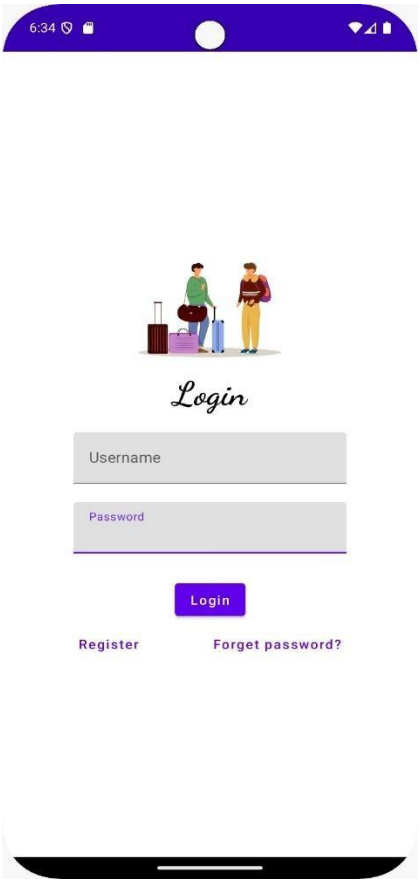
```
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/Theme.TravelApp"
tools:targetApi="31">


    <activity
android:name=".MainActivity"
android:exported="true"
android:label="@string/app_name"
android:theme="@style/Theme.TravelApp">
        <!-- Main Activity can be opened by an explicit intent -->
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>

    <!-- Login Activity -->
    <activity
android:name=".LoginActivity"
android:exported="true"
```

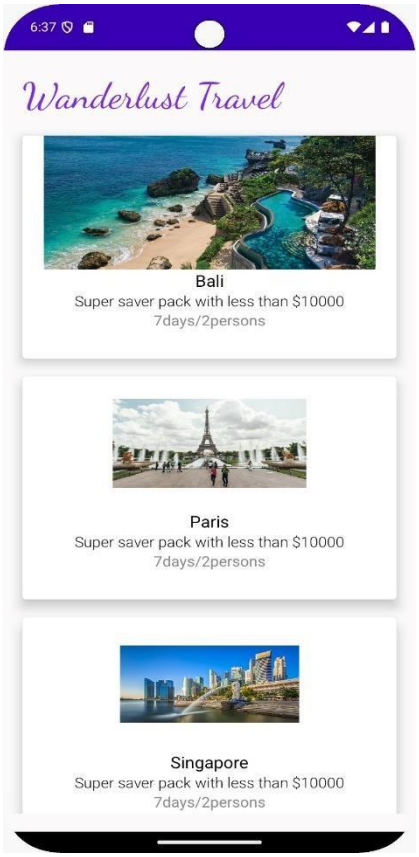
```
android:label="@string/app_name"
android:theme="@style/Theme.TravelApp">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity
android:name=".RegisterActivity"
android:exported="false"
android:label="RegisterActivity"
android:theme="@style/Theme.TravelApp" />

    <activity
android:name=".SingaporeActivity"
android:exported="false"
android:label="@string/title_activity_singapore"
android:theme="@style/Theme.TravelApp" />

    <activity          android:name=".ParisActivity"
android:exported="false"
```

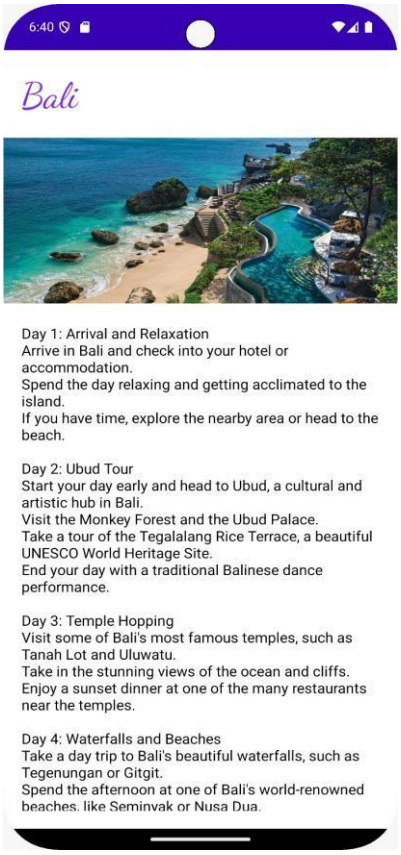
```
android:label="@string/title_activity_paris"
android:theme="@style/Theme.TravelApp" />

    <activity
android:name=".BaliActivity"
android:exported="false"
android:label="@string/title_activity_bali"
android:theme="@style/Theme.TravelApp"
/>
</application>
</manifest>
```

Output


6:36
Register
Username
Email
Password
Register
Have an account? Log in


6:34
Login
Username
Password
Login
Register
Forget password?


6:37
Wanderlust Travel
Bali
Super saver pack with less than $10000
7days/2persons
Paris
Super saver pack with less than $10000
7days/2persons
Singapore
Super saver pack with less than $10000
7days/2persons


6:40
Bali
Day 1: Arrival and Relaxation
Arrive in Bali and check into your hotel or accommodation.
Spend the day relaxing and getting acclimated to the island.
If you have time, explore the nearby area or head to the beach.
Day 2: Ubud Tour
Start your day early and head to Ubud, a cultural and artistic hub in Bali.
Visit the Monkey Forest and the Ubud Palace.
Take a tour of the Tegalalang Rice Terrace, a beautiful UNESCO World Heritage Site.
End your day with a traditional Balinese dance performance.
Day 3: Temple Hopping
Visit some of Bali's most famous temples, such as Tanah Lot and Uluwatu.
Take in the stunning views of the ocean and cliffs.
Enjoy a sunset dinner at one of the many restaurants near the temples.
Day 4: Waterfalls and Beaches
Take a day trip to Bali's beautiful waterfalls, such as Tegenungan or Gitgit.
Spend the afternoon at one of Bali's world-renowned beaches, like Seminyak or Nusa Dua.

Conclusion

This project highlights the advantages of using Jetpack Compose and Kotlin for developing Android apps with a modern, responsive UI. By leveraging Compose's declarative approach, the app provides a dynamic user experience that personalizes travel recommendations based on user preferences. Despite some learning challenges with Compose, the toolkit ultimately streamlined UI development, resulting in a scalable and maintainable codebase. This travel app serves as a valuable prototype for further expansion into a fully-featured travel assistant with real-time recommendations and user personalization.