Mobile Apps 2

Assignment Seven

Submission Date 02/12/2024

Percent 10%

Include all workings and references

Section A.

1. Give the main **advantages** and **disadvantages** of using APIs in your code (*3 marks*)

Advantages:

1. **Ease of Integration**: APIs enable seamless integration of third-party services, reducing the need to build everything from scratch (e.g., using Google Maps API for geospatial features).
2. **Access to Advanced Features**: APIs provide access to powerful and complex features (e.g., authentication through Firebase) that would otherwise require extensive development.
3. **Consistency and Reliability**: APIs often come from well-tested and maintained libraries, ensuring stable and consistent behaviour in your application.

Disadvantages:

1. **Dependency on Third Parties**: Reliance on external APIs makes your application dependent on the provider's availability and updates, which could cause issues if the API changes or becomes deprecated.
2. **Performance Overhead**: External API calls might introduce latency, affecting the performance of your application, especially with network-dependent APIs.
3. **Limited Customization**: You are restricted by the functionality and constraints of the API, which might not cover all your specific requirements.

---

2. If there is no actual working code within the methods, what is the reason for the use of **Interfaces**. (*2 marks*)

Reason:
Interfaces are used to define a **contract** or **blueprint** for classes. They specify what methods must be implemented, without dictating how they should be implemented. This ensures:

1. **Flexibility and Modularity**: Different classes can implement the same interface differently, enabling polymorphism and reusability of code.
2. **Decoupling**: Interfaces allow developers to program to an abstraction rather than a concrete implementation, making code more maintainable and testable.

For example, an interface like MapInteraction might define methods such as onMarkerClick() or onMapLoad(), allowing flexibility in how different map features are implemented.

Section B.

1. Using either your own code from Assignment 6, or the template code (will be made available soon), implement a means of the user inputting latitude and longitude settings and then placing that as a marker on the Google Map (4 *marks*)

2. Change either the color or icon of the check mark (1 Mark)

GoogleMaps Project Report

Introduction

The **GoogleMaps Project** is an Android application designed to provide an interactive map feature where users can input geographical coordinates or an address, navigate to a map screen, and customize the appearance of a marker. Built using **Jetpack Compose**, the app demonstrates the integration of modern Android technologies, including the Google Maps API, navigation components, and state management.

This report details the development process, technologies used, challenges faced, and key learnings from the project.

Project Overview

Features

1. **Interactive Input Screen**
    - Users input a latitude and longitude or an address.
    - A button becomes enabled when both fields are valid, allowing navigation to the map screen.
2. **Google Maps Integration**
    - A map displays a single customizable marker at the specified location.
    - Users can change the marker's colour using a button on the map screen.
3. **Navigation**
    - Seamless navigation between the input screen and the map screen.
    - Users can return to the input screen by clicking a "Back" button on the map screen.

Technologies Used

- **Jetpack Compose**: Declarative UI for building responsive and dynamic layouts.
- **Google Maps API**: Geospatial visualization of user-specified coordinates.
- **Navigation Component**: Simplifies screen transitions and state management.
- **Kotlin Coroutines**: Handles asynchronous tasks like address resolution.

- **Material Design 3**: Provides a clean and user-friendly interface.

Technical Details

Key Components

1. **Input Screen**
    - Accepts user inputs for latitude and longitude or address.
    - Validates the inputs and enables the "Go to Map" button only when valid data is entered.
2. **Map Screen**
    - Displays a map centred on the user-specified coordinates or address.
    - A customizable marker is placed at the centre.
    - Buttons allow changing the marker's colour and navigating back to the input screen.

Challenges and Solutions

Challenge 1: Input Validation

- **Problem**: Ensuring accurate validation for both numerical inputs (latitude/longitude) and text-based inputs (address).
- **Solution**: Implemented regex-based checks for coordinates and asynchronous validation for address resolution.

Challenge 2: Dynamic Marker Customization

- **Problem**: Allowing users to change marker colours interactively.
- **Solution**: Used Google Maps API's BitmapDescriptorFactory to dynamically set marker colours based on user actions.

Challenge 3: Navigation Management

- **Problem**: Ensuring that back navigation resets app state appropriately.
- **Solution**: Used Jetpack Navigation Component to manage stateful navigation cleanly.

Key Learnings

1. **Integration of Google Maps API**: Understanding how to use Maps Compose APIs to render dynamic content.
2. **Jetpack Compose Best Practices**: Leveraging state management for reactive UI updates.
3. **Navigation Component**: Managing navigation flows in a declarative and scalable manner.
4. **User Input Validation**: Ensuring robust input handling to enhance the user experience.

Conclusion

The **Google Maps Mobile App** is a demonstration of modern Android development practices, showcasing the integration of external APIs, dynamic UI components, and effective state management. The project serves as an example of building interactive and user-centric mobile applications using Jetpack Compose.

This app not only reinforces foundational Android skills but also provides a platform for exploring advanced features like geocoding and real-time map updates in future iterations.

References

1. Google Maps API Documentation
2. Jetpack Compose Documentation
3. Material Design Guidelines
4. Kotlin Coroutines
5. Navigation Component
6. Marker Customization in Google Maps
7. Input Validation in Kotlin
8. APIs Advantages and Disadvantages
9. Interfaces

- **GitHub Project Repository (GoogleMaps)**