



North South University

CSE 299: Junior Design Project

Section: 09

Final report

**Project Title: Local Tourist Guide Android App
“TownTrek”**

Submitted By:

Ristian Uddin [2021188642]

Mohammod Abdullah Bin Hossain [2022351642]

Submitted To:

Muhammad Shafayat Oshman(MUO)

Lecturer, Department of Electrical & Computer Engineering

North South University

Table of Contents

SL	Content	Page
1.	Introduction	3
2.	Problem Statement	3-4
3.	Project Description	4-12
4.	Project Flowchart	13
5.	Technologies Used	14
6.	Cost Analysis	14-15
7.	Conclusion	16
8.	GitHub Repository Link	16

Introduction

In an era where tourism plays a pivotal role in global economies, the demand for seamless and dynamic travel experiences is more pronounced than ever. The ubiquity of smartphones offers a promising avenue for addressing the challenges faced by modern tourists. This project, titled "Local Tourist Guide App - TownTrek," emerges as a response to the persistent hurdles encountered by travelers seeking real-time information about nearby attractions, amenities, and services. The current state of tourism often leaves individuals grappling with outdated or unreliable sources, highlighting the pressing need for an innovative solution that empowers tourists with accurate and timely guidance.

Our project aims to create a user-friendly mobile application that not only integrates real-time data but also provides a comprehensive map interface for intuitive navigation. By offering a detailed guide to nearby tourist spots, restaurants, hotels, ATMs, and other essential services, the app seeks to redefine the way individuals explore new destinations. Beyond mere convenience, the significance of this endeavor extends to its potential contributions to the tourism industry, local businesses, and the overall well-being of travelers. The envisioned impact encompasses enhanced safety, enriched experiences, and the facilitation of meaningful connections between tourists and the local culture.

At the heart of our project lies a commitment to innovation. Leveraging cutting-edge technologies, including real-time data integration and advanced mapping interfaces, our app promises a novel approach to tourist guidance. Throughout this report, we will delve into the intricacies of our project's design, development, and the seamless flowchart that guides users through its functionalities. Additionally, we will provide insights into the technologies employed, offering a comprehensive view of our cost-effective solution. As we navigate through each section, the overarching goal is to demonstrate not only the technical prowess of our local tourist guide app but also its potential societal impact.

Problem Statement

In the realm of modern tourism, despite the widespread use of technology, tourists often encounter significant challenges in accessing timely and reliable information during their journeys. These hurdles range from outdated guidebooks to the limitations of conventional mapping services, leaving travelers in need of a more comprehensive and dynamic solution. The aim of our project is to address these persistent issues and revolutionize the way individuals explore new destinations. The following key problems underscore the necessity for an innovative solution:

Outdated Information Source

Tourists frequently rely on outdated guidebooks or online sources that may not reflect the latest information about attractions, services, or local events.

Lack of Real-Time Guidance

Existing mapping services often fail to provide real-time data, leaving tourists unaware of sudden changes in opening hours, closures, or events in the vicinity.

Difficulty in Navigation

Navigating unfamiliar surroundings can be challenging, especially when tourists need to locate specific services like restaurants, hotels, ATMs, hospitals, or emergency facilities.

Inefficient Trip Planning

Tourists face difficulties in efficiently planning their trips, considering factors such as proximity to attractions, weather conditions, and the availability of essential services.

Limited Weather Information

Traditional travel apps often lack real-time weather data, depriving travelers of crucial information for preparing themselves adequately as they transition from one city to another.

The culmination of these challenges underscores the need for a comprehensive, user-centric solution that not only mitigates the identified issues but also enhances the overall travel experience for individuals seeking to explore new destinations. Through the integration of real-time data and innovative features, TownTrek aspires to bridge these gaps and redefine the landscape of modern tourism.

Project Description

Frontend

User Authentication and Map Interface

The Local Tourist Guide App begins its journey with robust user authentication facilitated by Firebase. Users seamlessly sign up and log in to the application, unlocking a personalized travel experience. The core of the user interface is built upon Google Maps, offering an interactive and intuitive map experience. Utilizing the FusedLocationProviderClient, the app ensures real-time tracking of the user's location, while the LocationServices package aids in updating and retrieving location information. The map interface is enriched with markers, providing a visual guide to nearby attractions and services.

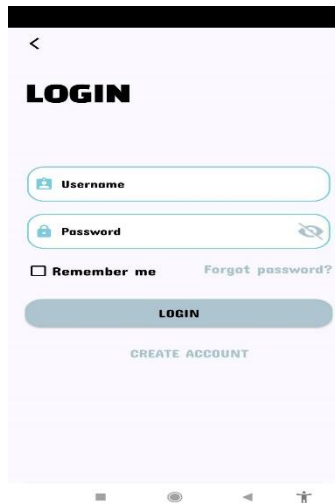


Fig. for Login Interface

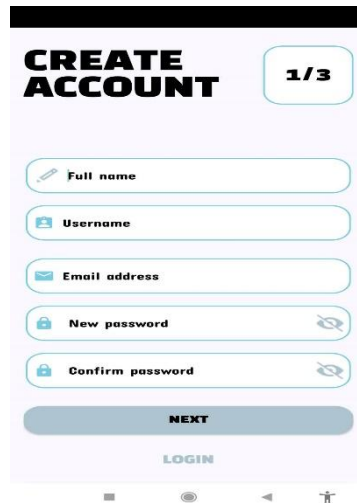


Fig. for SignUp Interface

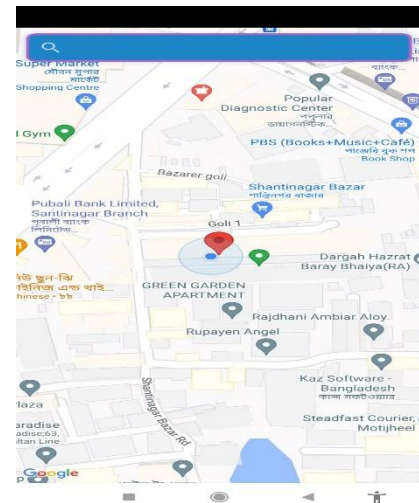


Fig. for Map Interface

Dynamic Map Features with Google Maps API

The dynamic map features of the app leverage various components from the Google Maps API. The Direction API, implemented through custom helper classes, enables users to obtain real-time directions for driving or walking between selected locations. The Places API empowers users to explore their surroundings, with custom helper classes facilitating searches for nearby restaurants, hotels, hospitals, ATMs, and more. The integration of these APIs ensures that users receive accurate and up-to-date information tailored to their location and preferences.

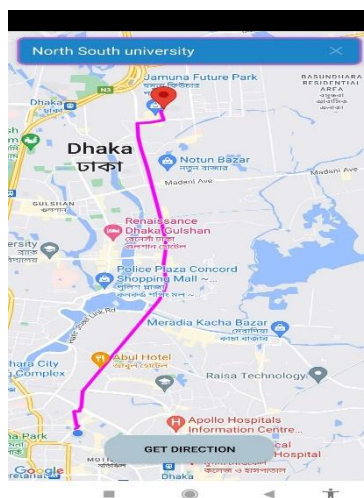


Figure for direction API

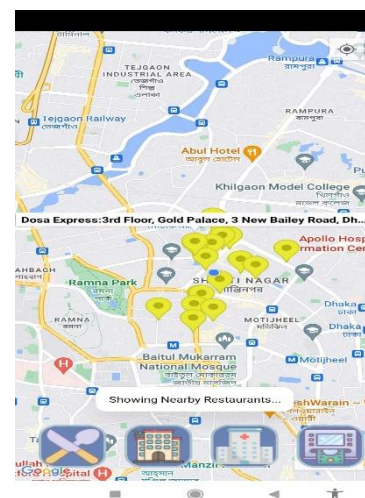


Figure for Place's API

Tourist Spot Exploration and Custom Dataset

The app goes beyond conventional mapping services by incorporating a unique feature for exploring nearby tourist spots. As Google Maps lacks specific Tourist Spot APIs, a custom dataset was created, encompassing popular destinations in Bangladesh. The TouristSpot helper class manages the details of each spot, including name, description, category, district, latitude, longitude, and an image URL. Users can seamlessly access information about these spots, view images, and even get walking directions from their current location.

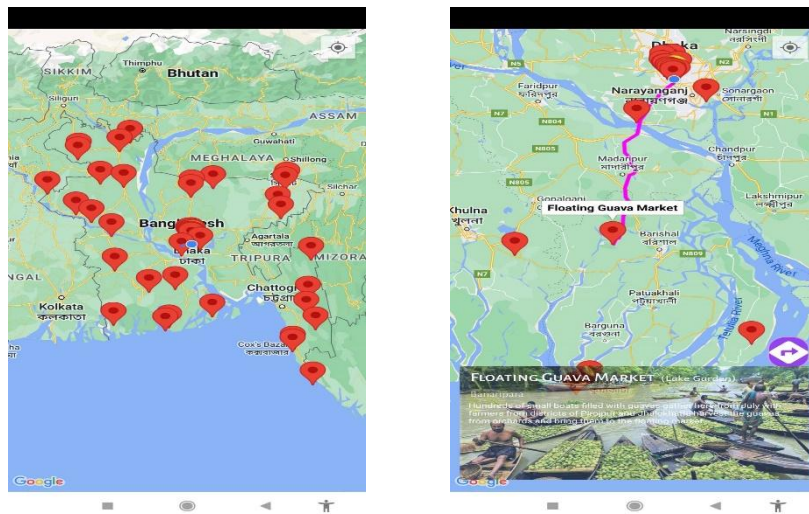


Figure for Custom Tourist Spots Dataset with Direction function

Weather Forecast Integration

To enhance the travel experience, the app extends its functionality to include real-time weather forecasts. OpenWeatherMap API integration, facilitated by Retrofit, allows users to retrieve weather data for any city. The weather information, including temperature and weather type, is displayed in a user-friendly manner, ensuring that travelers are well-prepared for changes in weather conditions as they move between cities.

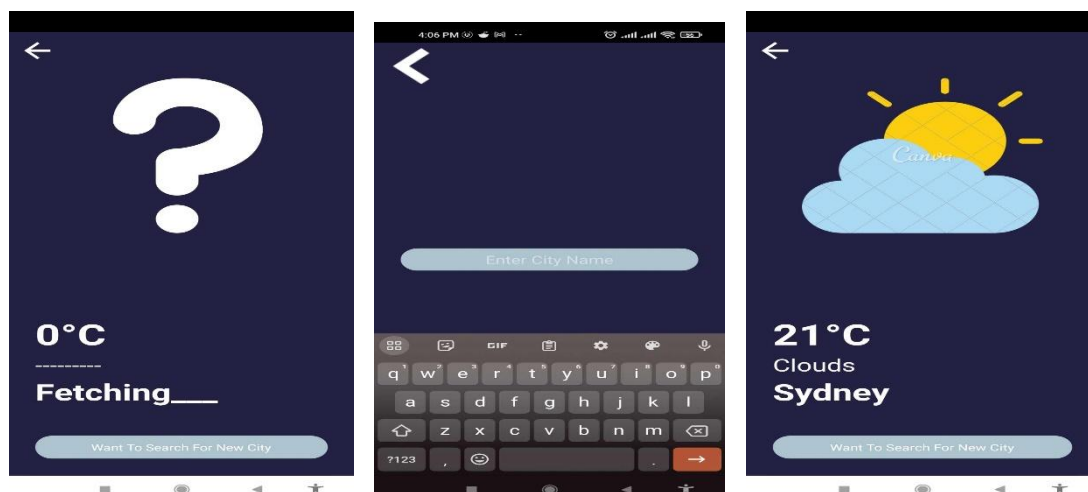


Figure for Weather API function and it shows weather condition of Sydney as we type “Sydney”:

Backend

Dependencies and Technologies

The Backend of the Local Tourist Guide App is powered by a range of essential dependencies and technologies. The app is developed using Android Studio, making extensive use of the AndroidX libraries for app compatibility and UI design. Key dependencies include Google Maps APIs for mapping services, Firebase for user authentication, and OpenWeatherMap API for weather forecasts. Retrofit is employed for efficient API communication, while Picasso handles image loading for a seamless visual experience.

➤ Enabling Firebase for user verification

```
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

 mAuth = FirebaseAuth.getInstance();

login_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        loginUser();
    }
});

private void loginUser() {
    String userEmail = username.getText().toString().trim();
    String userPassword = password.getText().toString().trim();

    if (userEmail.isEmpty() || userPassword.isEmpty()) {
        Toast.makeText(Login.this, "Please enter both email and password",
            Toast.LENGTH_SHORT).show();
        return;
    }

    mAuth.signInWithEmailAndPassword(userEmail, userPassword)
        .addOnCompleteListener(this, new OnCompleteListener() {
            @Override
            public void onComplete(@NonNull Task task) {
                if (task.isSuccessful()) {
                    FirebaseUser user = mAuth.getCurrentUser();
                    Toast.makeText(Login.this, "Authentication successful.",
                        Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(getApplicationContext(),
                        MainActivity.class));
                } else {
                    Toast.makeText(Login.this, "Authentication failed.",
                        Toast.LENGTH_SHORT).show();
                }
            }
        });
}
```

➤ Dependencies for Map API services:

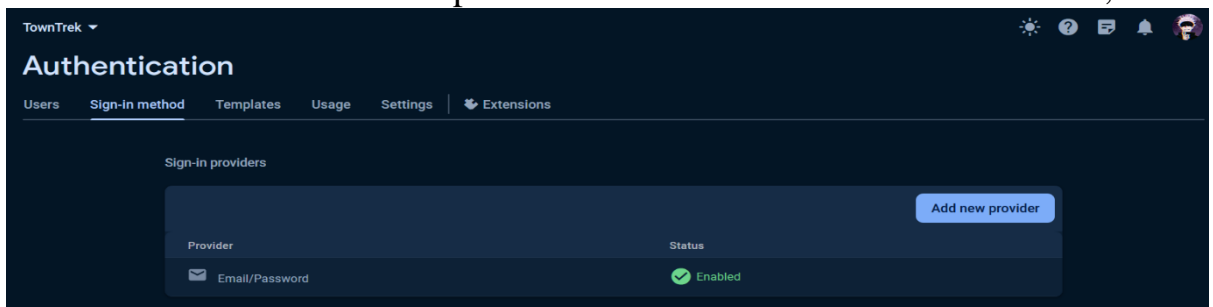
```
implementation 'com.google.android.gms:play-services-maps:18.2.0'
implementation 'com.google.maps.android:android-maps-utils:3.4.0'
implementation 'com.google.android.gms:play-services-location:21.0.1'
implementation 'com.google.maps:google-maps-services:0.17.0'
implementation 'com.google.android.libraries.places:places:3.2.0'
```


➤ Dependencies for App compatibility and UI Design:

```
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.material:material:1.9.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
implementation 'com.google.ar.sceneform:filament-android:1.17.1'
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.picasso:picasso:2.5.2'
```

Data Storage and Retrieval: Firebase Database serves as the backbone for user authentication and data storage. User information, login credentials, and preferences are securely stored and retrieved through Firebase, ensuring a seamless and secure user experience. Additionally, the custom dataset of tourist spots is managed through a local csv file in assets folder, allowing for efficient retrieval and presentation of spot details in the app.

➤ Activating Firebase for Data Storage for User Details: Now, as we activate the Email/password authentication method,



we then modify SignUp class:

```
import com.google.firebase.auth.FirebaseAuth;

public class SignUp extends AppCompatActivity {

    FirebaseAuth firebaseAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        firebaseAuth = FirebaseAuth.getInstance();
    }
    public void callNextSigupScreen(View view) {
        // Get user input (name, email, password)

        // Create a new user with email and password
        firebaseAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    Intent intent = new Intent(getApplicationContext(),
SignUp2ndClass.class);
                    // ... (existing code)
                    startActivity(intent, options.toBundle());
                } else {
                    Toast.makeText(SignUp.this, "Registration failed.",
Toast.LENGTH_SHORT).show();
                }
            });
    }
}
```


➤ Implementing Custom Dataset for Tourist Spots:

```
1.Cox's Bazar, Known for the world's longest natural sea beach., Sea Beach, Cox's Bazar-  
Chittagong, 21.4272, 92.0058, https://as1.ftcdn.net/v2/jpg/06/27/68/22/1000_F_627682287_C  
mV4UYhx7C7gxRkyJl8k8SC0kWeIo1PIU.jpg  
2.Ramna Park, A serene park offering greenery and open  
spaces., Park, Ramna, 23.7342, 90.3885, https://c8.alamy.com/comp/P2XXNT/a-couple-sits-on-  
the-bench-in-the-ramna-park-in-dhaka-bangladesh-P2XXNT.jpg  
3.....
```

Through the help of a helper class, TouristSpot.java:

```
public class TouristSpot {  
    private String placeName;  
    private String description;  
    private String category;  
    private String district;  
    private double latitude;  
    private double longitude;  
    private String imageUrl;  
  
    public TouristSpot(String placeName, String description, String category, String  
district, double latitude, double longitude, String imageUrl) {  
        this.placeName = placeName;  
        this.description = description;  
        this.category = category;  
        this.district = district;  
        this.latitude = latitude;  
        this.longitude = longitude;  
        this.imageUrl = imageUrl;  
    }  
}
```

Seamless Integration of APIs

The integration of various Google Maps APIs, including Directions and Places, requires meticulous implementation to ensure a smooth user experience. Custom helper classes are created to manage API requests and responses, facilitating dynamic map features and real-time data retrieval. The app's ability to fetch nearby places and tourist spot details is a testament to the seamless integration of these APIs.

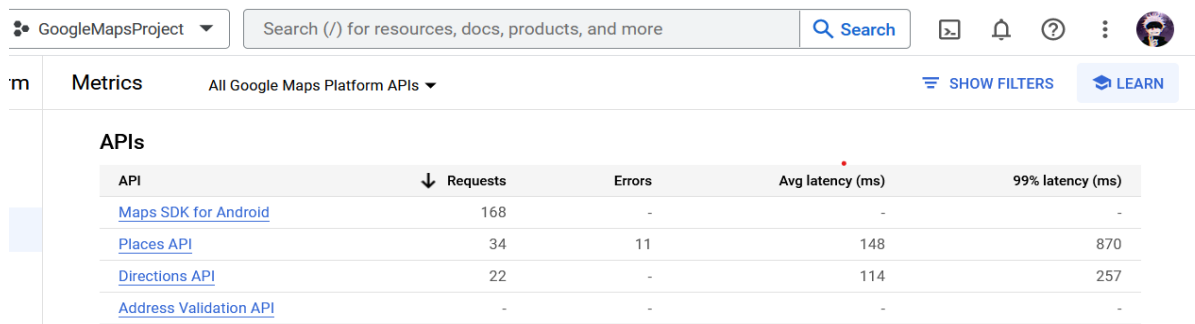
➤ Map Interface API: All the importation we needed to implement map,

```
import com.google.android.gms.location.FusedLocationProviderClient;  
import com.google.android.gms.location.LocationServices;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.MarkerOptions;  
import com.google.android.gms.maps.model.Polyline;  
import com.google.android.gms.maps.model.PolylineOptions;
```

This is the Map API key we collected from Google Map Platform:

```
Places.initialize(getApplicationContext(), getString("AIzaSyBEzLp_8t3hsrgILVMTLGpQijdmhi  
MzEAc"));
```

And now, we activate the API functions we need:



Metrics				
All Google Maps Platform APIs				
APIs				
API	↓ Requests	Errors	Avg latency (ms)	99% latency (ms)
Maps SDK for Android	168	-	-	-
Places API	34	11	148	870
Directions API	22	-	114	257
Address Validation API	-	-	-	-

➤ **Directions API Activation:** Importation from Helper classes:

```
import com.example.towntrek.MapDirectionHelpers.FetchURL;
import com.example.towntrek.MapDirectionHelpers.TaskLoadedCallback;
```

Each API has a URL function:

```
private String getUrl(LatLng origin, LatLng dest, String directionMode) {
    String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
    String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
    String mode = "mode=" + directionMode;
    String parameters = str_origin + "&" + str_dest + "&" + mode;
    String output = "json";
    String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" +
    parameters + "&key=" + getString(R.string.google_maps_key);
    return url;
}
```

And now, we call the function:

```
String url = getUrl(place1.getPosition(), place2.getPosition(), "driving");
new FetchURL(MapActivity.this).execute(url, "driving");
```

➤ **Places API Activation:** Importation from Helper Classes:

```
import com.example.towntrek.NearbyRestaurantHelpers.GetNearbyPlaces;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
```

URL function:

```
private String getUrl(double latitude, double longitude, String nearbyPlace)
{
    StringBuilder googleURL = new
    StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
    googleURL.append("location=" + latitude + "," + longitude);
    googleURL.append("&radius=" + ProximityRadius);
    googleURL.append("&type=" + nearbyPlace);
    googleURL.append("&sensor=true");
    googleURL.append("&key=" + getString(R.string.google_maps_key));
    Log.d("GoogleMapsActivity", "url = " + googleURL.toString());
    return googleURL.toString();
}
```

And now, suppose we call the function for nearby restaurants:

```
private void showNearbyRestaurants() {
    String restaurant = "restaurant";
    Object transferData[] = new Object[2];
    GetNearbyPlaces getNearbyPlaces = new GetNearbyPlaces();
    mMap.clear();
    String url = getUrl(latitude, longitude, restaurant);
    transferData[0] = mMap;
    transferData[1] = url;
    getNearbyPlaces.execute(transferData);
}
```

➤ Weather Forecast API Activation: Importation:

```
import com.loopj.android.http.AsyncHttpClient;
import com.loopj.android.http.JsonHttpResponseHandler;
import com.loopj.android.http.RequestParams;
import org.json.JSONObject;
import cz.msebera.android.httpclient.Header;
```

URL ID and Link through OpenWeatherMap:

```
final String APP_ID = "dab3af44de7d24ae7ff86549334e45bd";
final String WEATHER_URL = "https://api.openweathermap.org/data/2.5/weather";
```

And also through the help of helper class weatherData.java:

```
public static weatherData fromJson(JSONObject jsonObject)
{
    try
    {
        weatherData weatherD=new weatherData();
        weatherD.mcity=jsonObject.getString("name");

        weatherD.mCondition=jsonObject.getJSONArray("weather").getJSONObject(0).getInt("id");

        weatherD.mWeatherType=jsonObject.getJSONArray("weather").getJSONObject(0).getString("main");

        weatherD.micon=updateWeatherIcon(weatherD.mCondition);
        double tempResult=jsonObject.getJSONObject("main").getDouble("temp")-273.15;
        int roundedValue=(int) Math rint(tempResult);
        weatherD.mTemperature=Integer.toString(roundedValue);
        return weatherD;
    }
    catch (JSONException e) {
        e.printStackTrace();
        return null;
    }
}
```

Robust Networking and API Communication

The Backend of app prioritizes robust networking, exemplified by use of libraries such as Volley for API communication. Asynchronous HTTP requests are handled to ensure that the app can fetch real-time data without compromising on performance. The integration of Retrofit further streamlines API communication, making data retrieval processing a seamless part of the user experience.

- **Implementation of Volley Library:** It abstracts the process of handling network requests. It provides a framework for making asynchronous requests, handling request queuing, caching, and response parsing:

```
implementation 'com.android.volley:volley:1.2.1'
```

- **Asynchronous HTTP Requests:** Typically, when we make requests to an API or a server, it performs network operations asynchronously through JSONObject, preventing it from blocking the main thread and ensuring a smooth user experience. For example:

```
@Override
protected List<List<HashMap<String, String>>> doInBackground(String... jsonData) {

    JSONObject jObject;
    List<List<HashMap<String, String>>> routes = null;

    try {
        jObject = new JSONObject(jsonData[0]);
        Log.d("mylog", jsonData[0].toString());
        DataParser parser = new DataParser();
        Log.d("mylog", parser.toString());

        // Starts parsing data
        routes = parser.parse(jObject);
        Log.d("mylog", "Executing routes");
        Log.d("mylog", routes.toString());

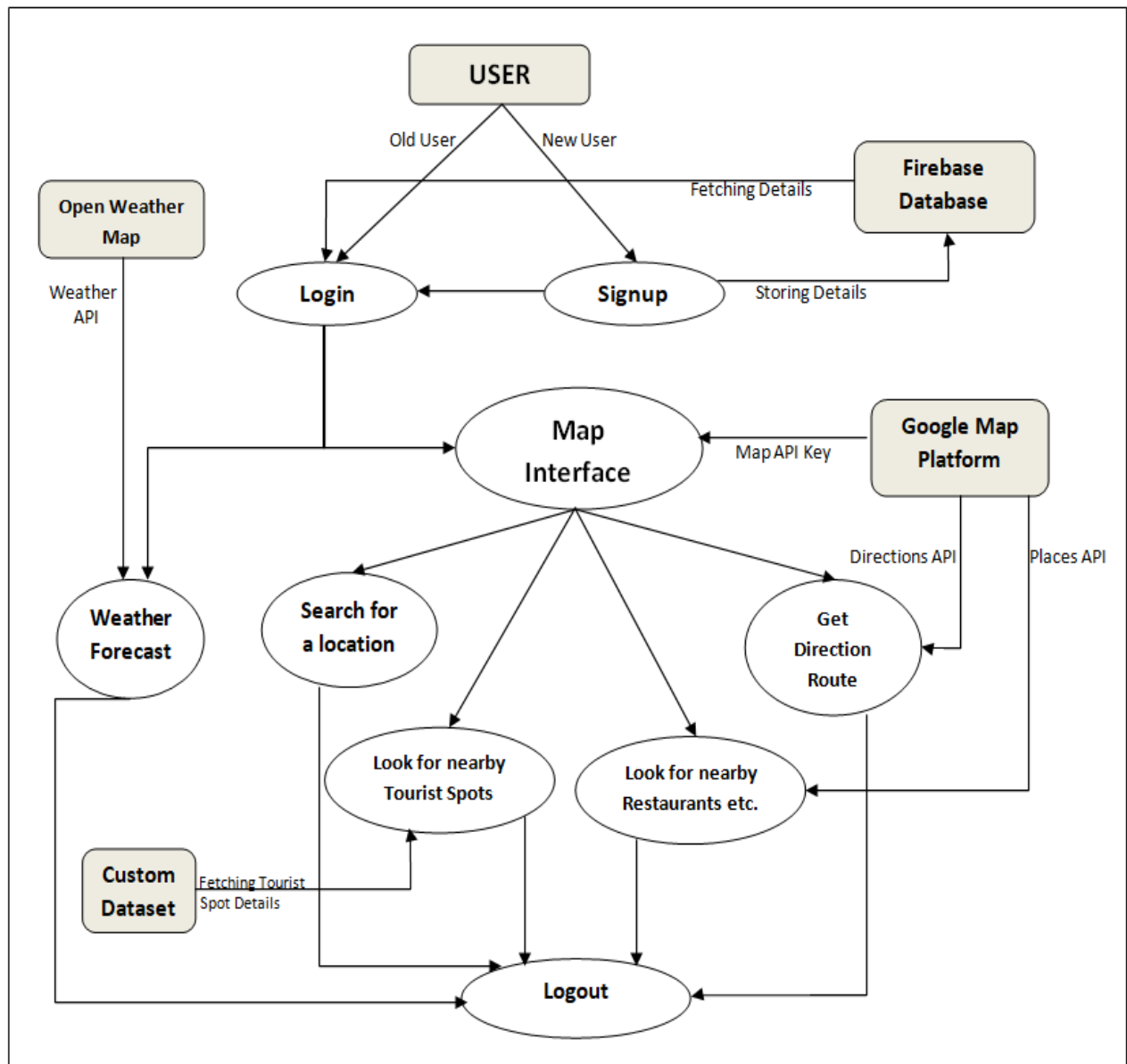
    } catch (Exception e) {
        Log.d("mylog", e.toString());
        e.printStackTrace();
    }

    return routes;
}
```

In essence, the Local Tourist Guide App represents a harmonious integration of FrontEnd and BackEnd technologies, delivering a comprehensive, user-friendly, and user-centric solution for modern travelers.

Project Flowchart

Flowchart Diagram



Technologies used

- Android Studio
- Programming Language: JAVA
- UI Design Language: XML
- Firebase Database (for user authentication)
- Google Maps Platform (for API services like Places API, Directions API)
- OpenWeatherMap (for real-time Weather forecasting API)

Cost Analysis

Approximate cost for one-time registration fee to set up a developer account on the **Google Play Console** = **\$25**

We also might need to hire some developers for ongoing maintenance and updates to keep the app compatible with the latest Android versions and to address user feedback or issues. Suppose we take in two developers to maintain this with \$50,000 salary per year, which is roughly around \$4,200 per month. So, **salary of two developers** per month would be = **$\$4,200 * 2$**
= \$8,400

As we are using third-party services or APIs, such as Firebase, Google Maps, or OpenWeatherMap, there might be associated costs. Some services offer free tiers with limitations, and charges may apply as usage increases. But that depends on the number of users we gain each day. Suppose we had a total of 50,000 users in its first month. As a result, we have to consider the premium access of each services due to the increase in usage.

Let's say, a total of 500,000 documents are getting read, written and deleted everyday in Firebase Database. According to their pricing, to read, it is \$0.036 charged per 100,000 documents, \$0.108 for writing and \$0.012 for deleting.

So, only in the case of reading 500,000 documents in one month, calculation
= **$\{(500,000 - 50,000) * (\$0.036/100,000)\} * 30$**
= **\$4.86** (50,000 is subtracted because first 50,000 are free each day and 30 is multiplied because of getting calculation for 1 month = 30 days)

Writing 500,000 documents in one month = **$\{(500,000 - 20,000) * (\$0.108/100,000)\} * 30$**
= **\$15.55**

Deleting 500,000 documents in one month = $\{(500,000 - 20,000) * (\$0.012/100,000)\} * 30 = \1.728

So, in total, charges for one month from **Firestore** = $\$ 4.86 + \$ 15.55 + \$1.728 = \$22.14$

In same way, if we have to calculate the cost for OpenWeatherMap API, we have to consider the 'Developer' or 'Professional' subscription instead of "Pay as we go" option, since they are much cheaper and has more access. 'Professional' subscription allows 30,000 calls/per minute and 100,000,000 calls/per month which is almost the same as 'Developer' but it has access to download data in bulk quantity. 'Professional' costs around \$500 per month and 'Developer' around \$190 per month, so we can roughly take an amount between them since if we decide to upgrade our **OpenWeatherMap** subscription based on the situation = **\$190 to \$500**
= **\$350**

Now, when it comes to Google Map Platform, loading Map SDK is free no matter how many times it is loaded. But, there are charges for Directions and Places API upto a certain point. Let's assume that around 50,000 API services are getting requested every day, then pricing would be as follows:

For 50,000 Directions API per day, the charge for one month will be = **50,000 * (\$ 0.02 / 1000) * 30 = \$300**

For 50,000 Places API per day, the charge for one month will be = **50,000 * (\$ 0.5/1000) * 30 = \$750**

So, total amount **Google Map Platform** will charge in 1 month = $\$ 300 + \$ 750 = \1050

Therefore, around = $\$25 + \$8,400 + \$22.14 + \$350 + \$1050 = \$ 9,847$ amount of expenses will be the minimum amount to carefully plan and budget for each stage of app development and maintenance, since it might vary depending, if we plan to improve the features and design of our app in a later period.

Conclusion

In conclusion, TownTrek marks a significant step towards enhancing user experiences and promoting tourism within the community. By seamlessly integrating Firebase for user authentication, Google Maps Platform for mapping services, and OpenWeatherMap for real-time weather forecasting, the app provides a comprehensive solution for both locals and visitors. Moving forward, there is a great opportunity to further enrich the app by incorporating features that contribute to societal welfare, such as accessibility information, eco-friendly travel options, and highlighting lesser-known attractions to distribute tourist traffic more evenly. This strategic improvement not only benefits users but also supports sustainable tourism development, contributing to the overall economic growth of local communities. As technology continues to play a pivotal role in shaping the travel landscape, this app stands as a testament to the potential for innovation to foster positive societal and economic impacts within the realm of tourism.

GitHub Repository Link

Repository Link: <https://github.com/RistianRidoy/TownTrek--Local-Tourist-Guide-Android-App>

****** THE END ******