Public Transport Optimization

Phase 4: Development Part 2

Here, we develop the real-time transit information platform.

Tasks:

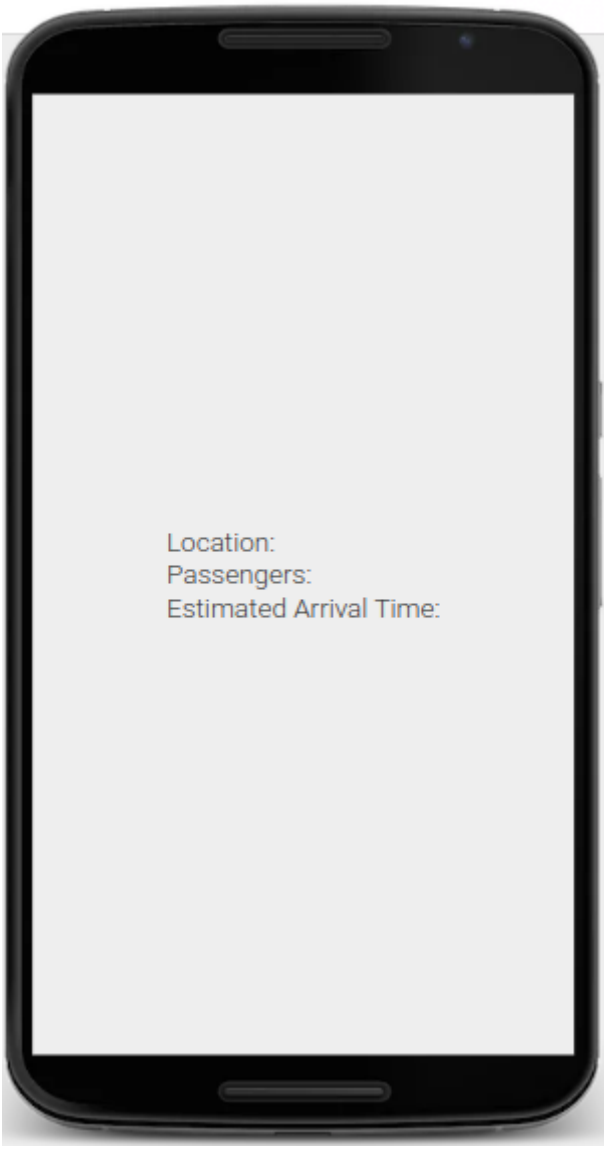
- *Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time transit information.*
- *Design the platform to receive and display real-time location, ridership, and arrival time data from IoT sensors.*

We are building an app based platform. XML layout defines the look and feel of the platform.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center">

    <TextView
        android:id="@+id/locationTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Location: "
        android:textSize="18sp" />

    <TextView
        android:id="@+id/passengerCountTextView"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Passengers: "
        android:textSize="18sp" />

    <TextView
        android:id="@+id/arrivalTimeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Estimated Arrival Time: "
        android:textSize="18sp" />

</LinearLayout>
```



Our app receives the information of real-time location, ridership data from the server through http requests. The arrival time is estimated using the current location of the user and the location of the public transport received from the server (received from the gps location) considering various factors like real-time traffic, speed, intermediate stops.

MainActivity.java

```
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.content.Context;
import android.content.pm.PackageManager;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.os.Bundle;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.concurrent.TimeUnit;
import java.util.List;
import java.util.Arrays;

import com.google.android.gms.maps.model.LatLng;
import com.google.android.libraries.places.api.Places;
import
com.google.android.libraries.places.api.net.FindCurrentPlaceRequest;
import
com.google.android.libraries.places.api.net.FindCurrentPlaceResponse;
import com.google.android.libraries.places.api.net.PlacesClient;
import com.google.android.libraries.places.api.model.Place;
```

```
import com.google.android.libraries.places.api.model.PlaceLikelihood;

public class MainActivity extends AppCompatActivity implements
ActivityCompat.OnRequestPermissionsResultCallback {

    private TextView locationTextView;
    private TextView passengerCountTextView;
    private TextView arrivalTimeTextView;
    private PlacesClient placesClient;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        View rootView =
LayoutInflater.from(this).inflate(R.layout.activity_main, null);
        setContentView(rootView);

        locationTextView = rootView.findViewById(R.id.locationTextView);
        passengerCountTextView =
rootView.findViewById(R.id.passengerCountTextView);
        arrivalTimeTextView =
rootView.findViewById(R.id.arrivalTimeTextView);

        // Check and request permissions
        if (checkLocationPermission()) {
            // Permissions are already granted. Proceed with your
location-related tasks.
            initializePlaces();
            new RetrieveDataFromServerTask().execute();
        }
    }

    private boolean checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // Permission is not granted. You can request it.
```

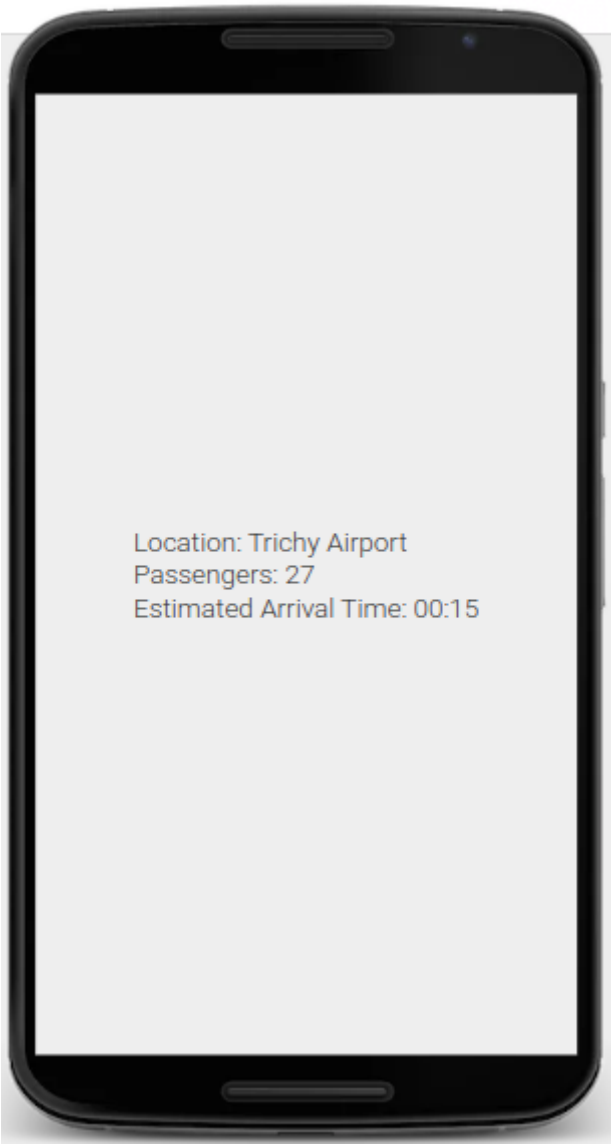
```
            String[] permissions =
{Manifest.permission.ACCESS_FINE_LOCATION};
            int requestCode = 1;

            ActivityCompat.requestPermissions(this, permissions,
requestCode);
            return false; // Permissions not granted yet.
        } else {
            return true; // Permissions are already granted.
        }
    }

    private void initializePlaces() {
        // Initialize the Places API client
        Places.initialize(getApplicationContext(), "API_KEY");
        placesClient = Places.createClient(this);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if (requestCode == 1) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                // Permission was granted. You can now proceed with
location-related tasks.
                initializePlaces();
                new RetrieveDataFromServerTask().execute();
            } else {
                // Permission was denied. Handle this situation, e.g., by
showing a message to the user.
                locationTextView.setText("Location permission denied.
Please grant the permission in app settings.");
            }
        }
```

```
    }

    private class RetrieveDataFromServerTask extends AsyncTask<Void, Void,
JSONObject> {
        @Override
        protected JSONObject doInBackground(Void... voids) {
            try {
                URL url = new
URL("http://127.0.0.1:5000/update_vehicle_data");
                HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                connection.setRequestMethod("GET");

                InputStream inputStream = connection.getInputStream();
                BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
                StringBuilder result = new StringBuilder();
                String line;

                while ((line = reader.readLine()) != null) {
                    result.append(line);
                }

                return new JSONObject(result.toString());
            } catch (IOException | JSONException e) {
                e.printStackTrace();
                return null;
            }
        }

        @Override
        protected void onPostExecute(JSONObject result) {
            if (result != null) {
                try {
                    double latitude = result.getDouble("lat");
                    double longitude = result.getDouble("lng");
                    int passengers = result.getInt("passengers");
```

```
                    // Use the Places API to get place name from latitude
and longitude
                    LatLng latLng = new LatLng(latitude, longitude);
                    getPlaceNameFromCoordinates(latLng);

                    // Display the passenger count in the app
                    passengerCountTextView.setText("Passengers: " +
passengers);

                    // Calculate and display the estimated arrival time
                    String estimatedArrivalTime =
calculateEstimatedArrivalTime(latitude, longitude);
                    arrivalTimeTextView.setText("Estimated Arrival Time: "
+ estimatedArrivalTime);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                locationTextView.setText("Error retrieving data from the
server.");
                passengerCountTextView.setText("Passenger count not
available.");
            }
        }
    }

    private void getPlaceNameFromCoordinates(LatLng latLng) {
        List<Place.Field> placeFields = Arrays.asList(Place.Field.NAME);

        FindCurrentPlaceRequest request =
FindCurrentPlaceRequest.newInstance(placeFields);


placesClient.findCurrentPlace(request).addOnSuccessListener((response) ->
{
            if (response != null) {
                for (PlaceLikelihood placeLikelihood :
response.getPlaceLikelihoods()) {
```

```
                    String placeName =
placeLikelihood.getPlace().getName();
                    updateLocationTextView(placeName);
                    break; // Get the first place as the most likely
                }
            } else {
                updateLocationTextView("Place name not found");
            }
        }).addOnFailureListener((exception) -> {
            updateLocationTextView("Error getting place name");
        });
    }

    private void updateLocationTextView(String placeName) {
        locationTextView.setText("Location: " + placeName);
    }

    private String calculateEstimatedArrivalTime(double latitude, double
longitude) {
        LatLng destination = new LatLng(latitude, longitude);
    String origin = "Starting location";

    // Use the Google Directions API to get estimated travel time
    DirectionsApiRequest directions = new
DirectionsApiRequest(Places.createClient(getApplicationContext()));
    directions.origin(origin);
    directions.destination(destination);
    directions.mode(TravelMode.DRIVING); // You can change the mode as
needed

    try {
        DirectionsResult directionsResult = directions.await();

        if (directionsResult != null) {
            DirectionsRoute route = directionsResult.routes[0];
            DirectionsLeg leg = route.legs[0];

            // Get the estimated duration in minutes
```

```
            long durationInMinutes = leg.duration.inSeconds / 60;

            // Calculate arrival time by adding the duration to the
current time
            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.MINUTE, (int) durationInMinutes);

            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm",
Locale.getDefault());
            String estimatedArrivalTime = sdf.format(calendar.getTime());

            arrivalTimeTextView.setText("Estimated Arrival Time: " +
estimatedArrivalTime);
        } else {
            arrivalTimeTextView.setText("Error calculating arrival
time.");
        }
    } catch (ApiException | InterruptedException | IOException e) {
        e.printStackTrace();
        arrivalTimeTextView.setText("Error calculating arrival time.");
    }
    }
    }
}
```


Location: Trichy Airport
Passengers: 27
Estimated Arrival Time: 00:15