

CHAPTER 1

INTRODUCTION

1.1 PROJECT TITLE:

NavAR: Augmented Reality Assisted Navigation Application using Google Maps.

1.2 OBJECTIVE:

To use the phone's camera and Augmented Reality along with Google Maps to provide real-time navigation assistance.

1.3 OVERVIEW:

Satellite navigation plays a very important role in our daily lives, be it in travel assistance, taxis or finding new information about a place.

Our project aims to enhance the usability of Google Maps by providing an additional virtual assistance using augmented reality. Users can attain a real-time direction assistance through the app, which utilises the phone's camera, digital compass and GPS to project the directions on the phone's screen as an overlay on the camera feedback.

This will help users to better navigate complex roadways and urban environments.

1.4 EXISTING SYSTEM

- 1. ARCity by blippAR (Available only on iOS) [3]** - The app helps you navigate and explore 300 cities worldwide using augmented reality and computer vision at scale.



Fig 1.1: ARCity by blippAR¹

2. AR Navigation & Nearby GPS, AR Walking Navigation by AR Technology | GP¹S Navigation | Future Apps (Available only on Google Play Store) [4] - AR Navigation 2019 – AR GPS Navigation AR Maps, AR Driving Directions is augmented reality navigation app that let you explore the world map see your current location or find any places with Nearby GPS & AR Navigation.



Fig 1.2: AR Navigation & Nearby GPS, AR Walking Navigation²

¹ Source: <https://www.blippar.com/blog/2017/11/06/welcome-ar-city-future-maps-and-navigation>

² Source: <https://play.google.com/store/apps/details?id=com.augmentednavigation.routefinder.famousplaces.roads&hl=en>

1.5 PROBLEM DEFINITION

While modern GPS systems have evolved a long way in providing users a detailed navigation system, the readability of maps remains a struggle for many users. This application aims to solve this very problem by adding another layer of readability by utilising the phone's camera and incorporating augmented reality to provide the necessary directions.

1.6 PROPOSED PLAN

With the above problem clearly defined, our aim is to create an augmented reality application to work simultaneously with Google Maps to provide camera assisted navigation using the phone's camera.

We aim to complete the above in the following ways:

1. Create a basic AR app with plane recognition with custom markers
2. Create an Android application with Google Maps SDK to extract location information
3. Integrate the modules to obtain AR assisted navigation

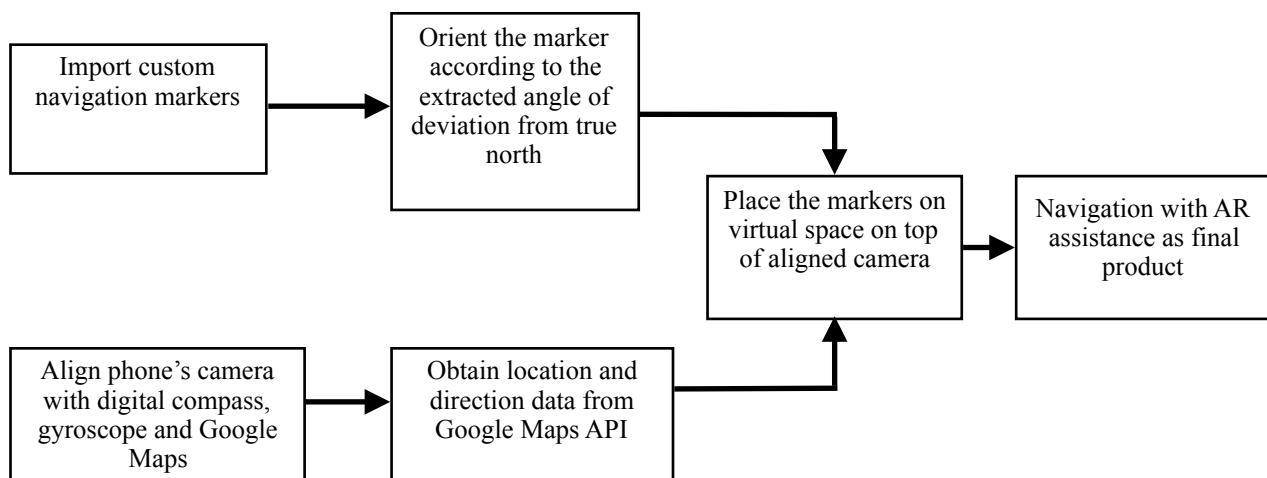


Fig 1.3: Proposed Workflow

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 SYSTEM REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS

Processor: Intel Pentium 4 (With SSE2 instruction set support) or later [1]
RAM: 3 GB or more [1]
GPU: DirectX 10 (shader model 4.0) support [1]
Disk Space: 3 GB or more [1]
Screen Resolution: 1280X800 or more [1]
Android Phone with Android 7 and above [5]

2.1.2 SOFTWARE REQUIREMENTS

OS: Windows 7 SP1 or later, MacOS 10.11 or later [1]
Android Studio 3.3
JDK 8
ARCore 1.7
Programming Languages: Java [1]

2.2 FEASIBILITY STUDY

2.2.1 ECONOMIC FEASIBILITY

2.2.1.1 COCOMO MODEL

The basic Constructive Cost Model (COCOMO) [3] equation will take the following form:

$$\text{Effort Applied (E)} = a_b(KLoC)^{b_b} \text{ [person-months]}$$

Development Time (D)= $c_b(Effort\ Applied)^{d_b}$ [months]

People Required (P)=Effort Applied/Development time [count]

Where KLoC is the estimated number of delivered lines (expressed in thousands) of code

The coefficients a_b b_b c_b d_b are given by:

Table 2.1: COCOMO model coefficient values

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Taking the project type to be semi-detached, we get the following calculations:

Effort Applied, $E= a_b(KLoC)^{b_b}$ [person-months]

$$E=3.0*(4K)^{1.12} PM$$

$$E=14.17 PM$$

Development Time, $D= c_b(Effort\ Applied)^{d_b}$ [months]

$$D=2.5*(14.17)^{0.35} months$$

$$D=6.32 months$$

People Required, $P=Effort\ Applied/Development\ time$ [count]

$$P=14.17/6.32 count$$

$$P=2.24 count$$

$$P \approx 3 count$$

2.2.2 TECHNICAL FEASIBILITY

The requirements for making this application are as follows:

- Android Studio
- ARCore
- Google Maps SDK for Android (free for one year)
- Google Places API (free for one year)
- Google Directions API (free for one year)

The above mentioned utilities are open source. Hence the project is technically feasible.

2.2.3 OPERATIONAL FEASIBILITY

The finished product will have an easy-to-use, simple and intuitive interface. The end user does not require any special training to use this application. Therefore, the application is operationally feasible.

2.2.4 SCHEDULE FEASIBILITY

2.2.4.1 WORK BREAKDOWN STRUCTURE

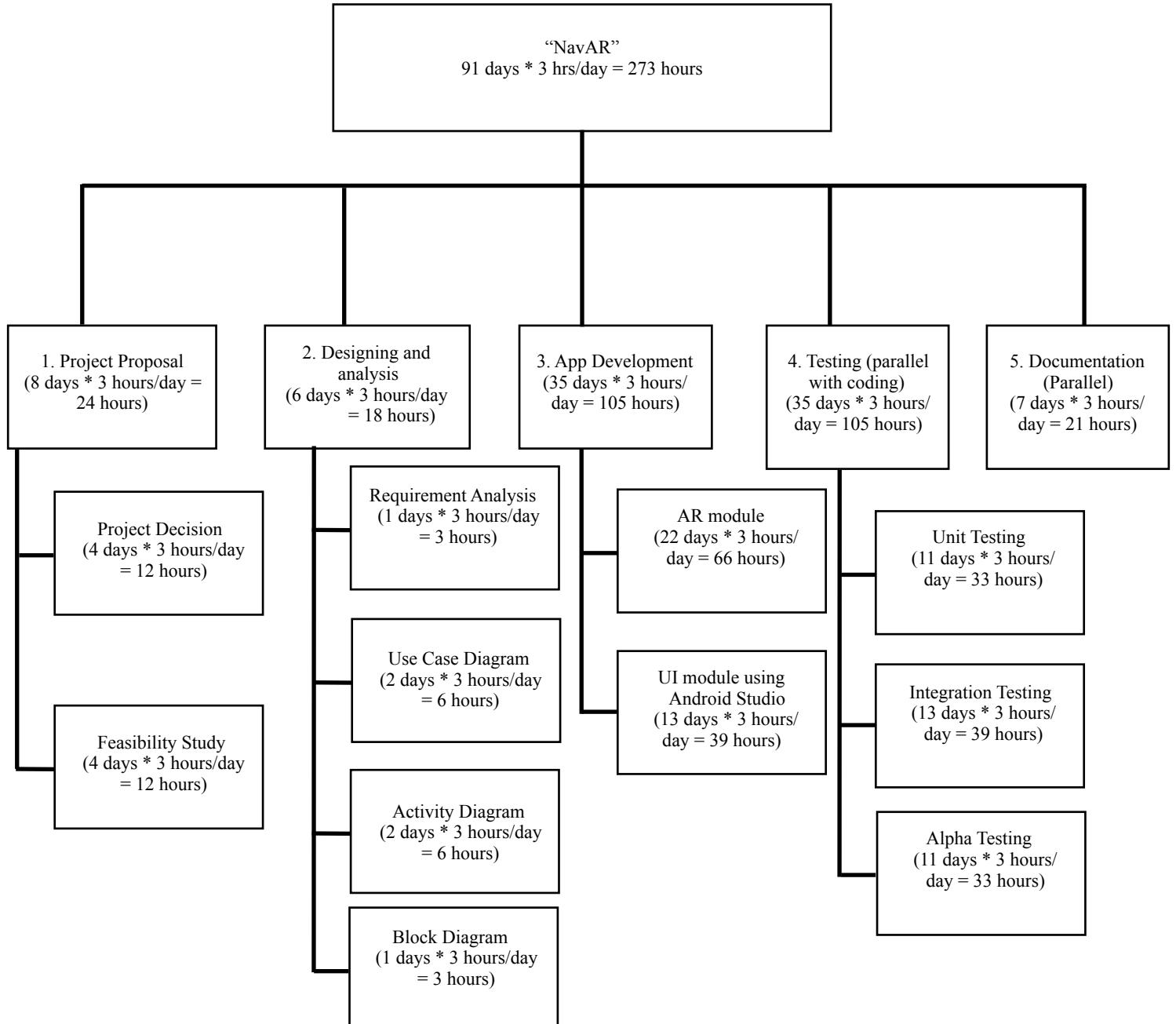


Fig 2.2: Work Breakdown Structure

2.2.4.2 GANTT CHART

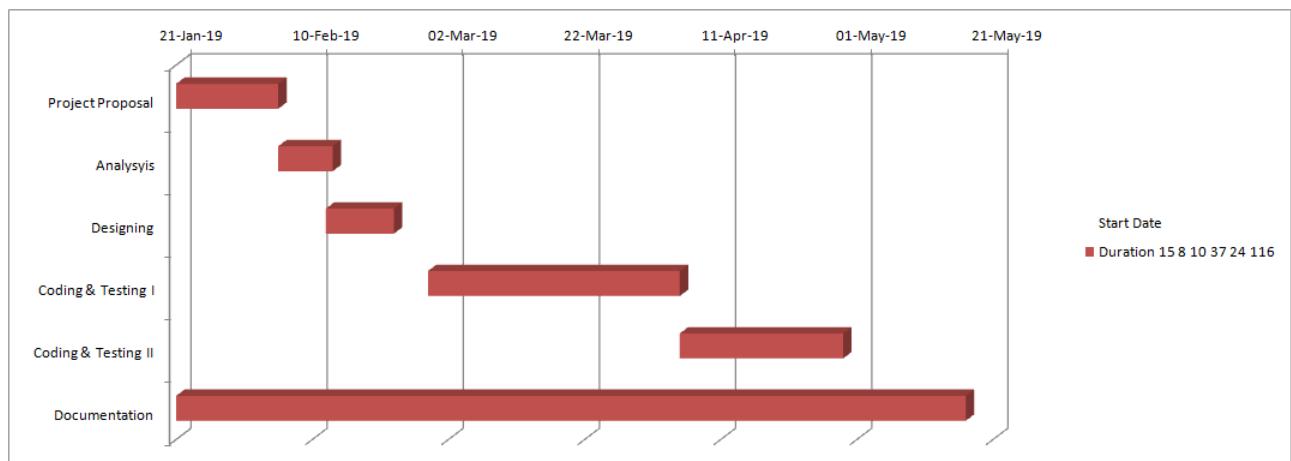


Fig 2.3: Gantt Chart

CHAPTER 3

DESIGN DIAGRAMS

3.1 USE-CASE DIAGRAM

NavAR

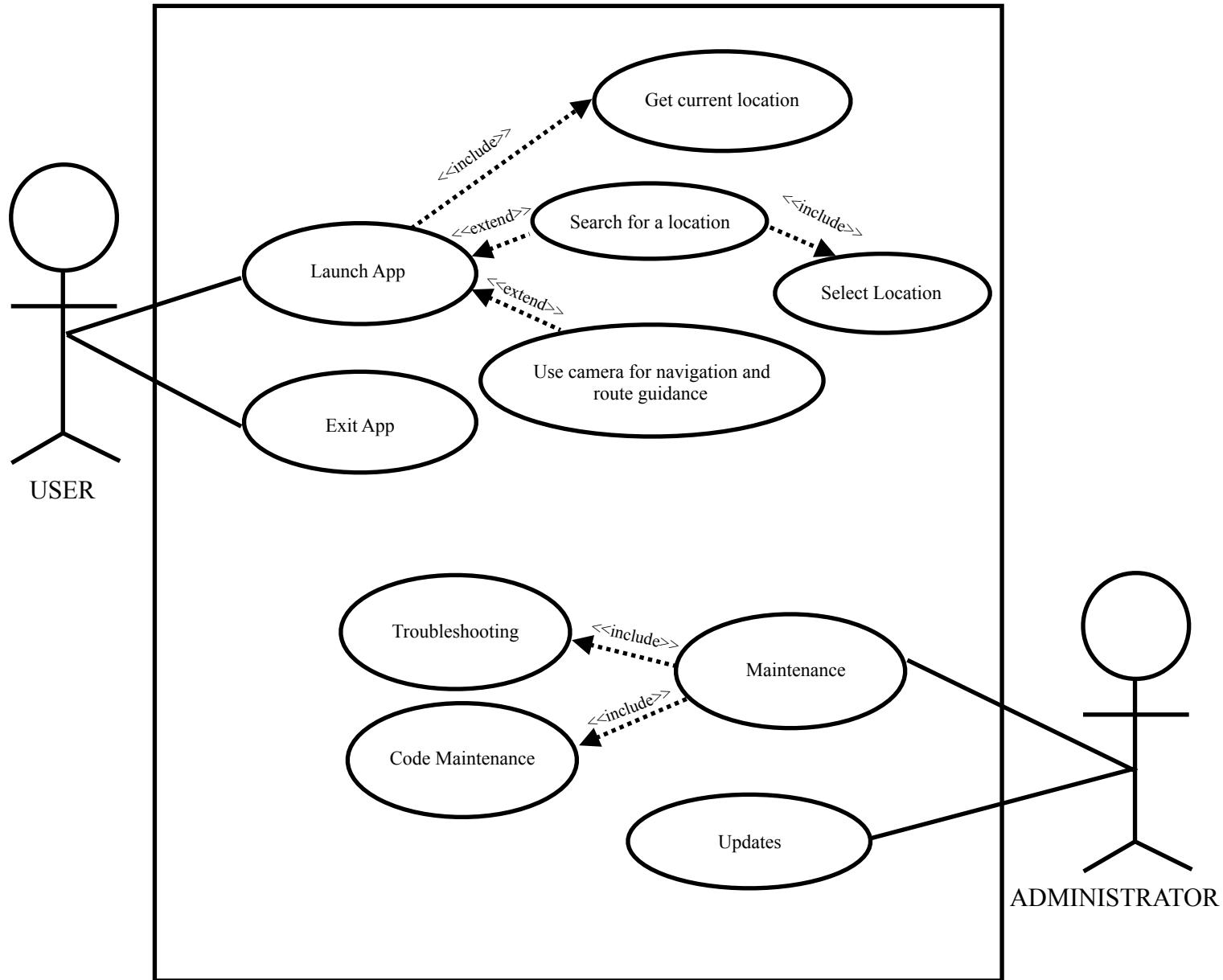


Fig 3.1: Use Case Diagram

3.2 ACTIVITY DIAGRAM

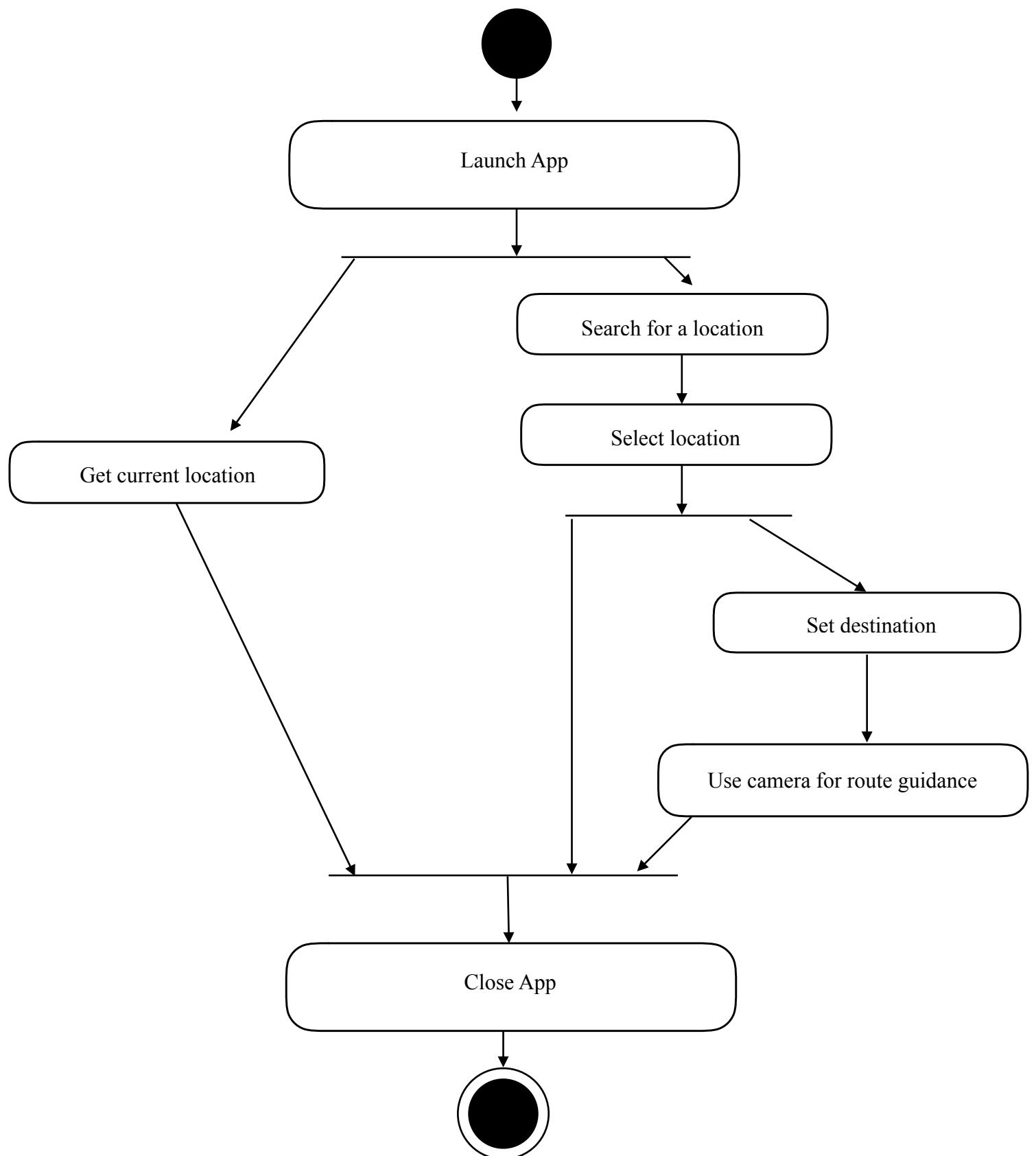


Fig 3.2: Activity Diagram

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 TESTING

4.1.1 MAP

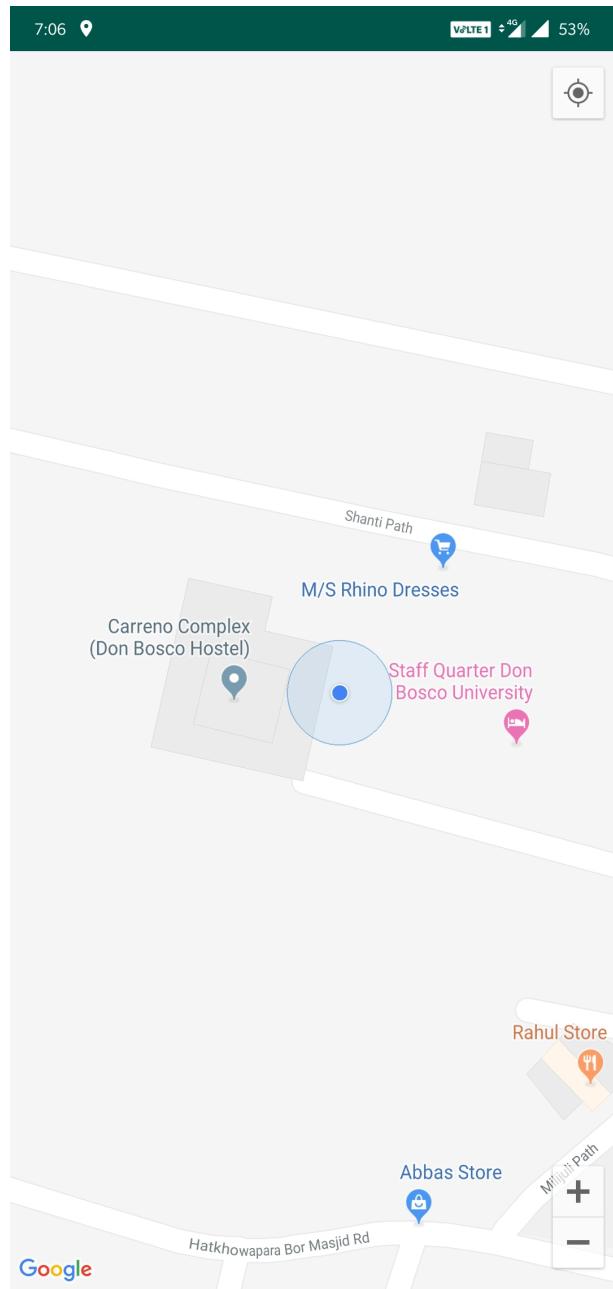


Fig 4.1: Maps interface

The map section gives the primary interface for the application and houses all the functionalities prior to navigation, that is, getting the current location, searching for a destination and reviewing the route.

4.1.2 AUGMENTED REALITY

The Augmented Reality section will provide the route guidance using the camera input and location markers as shown below.

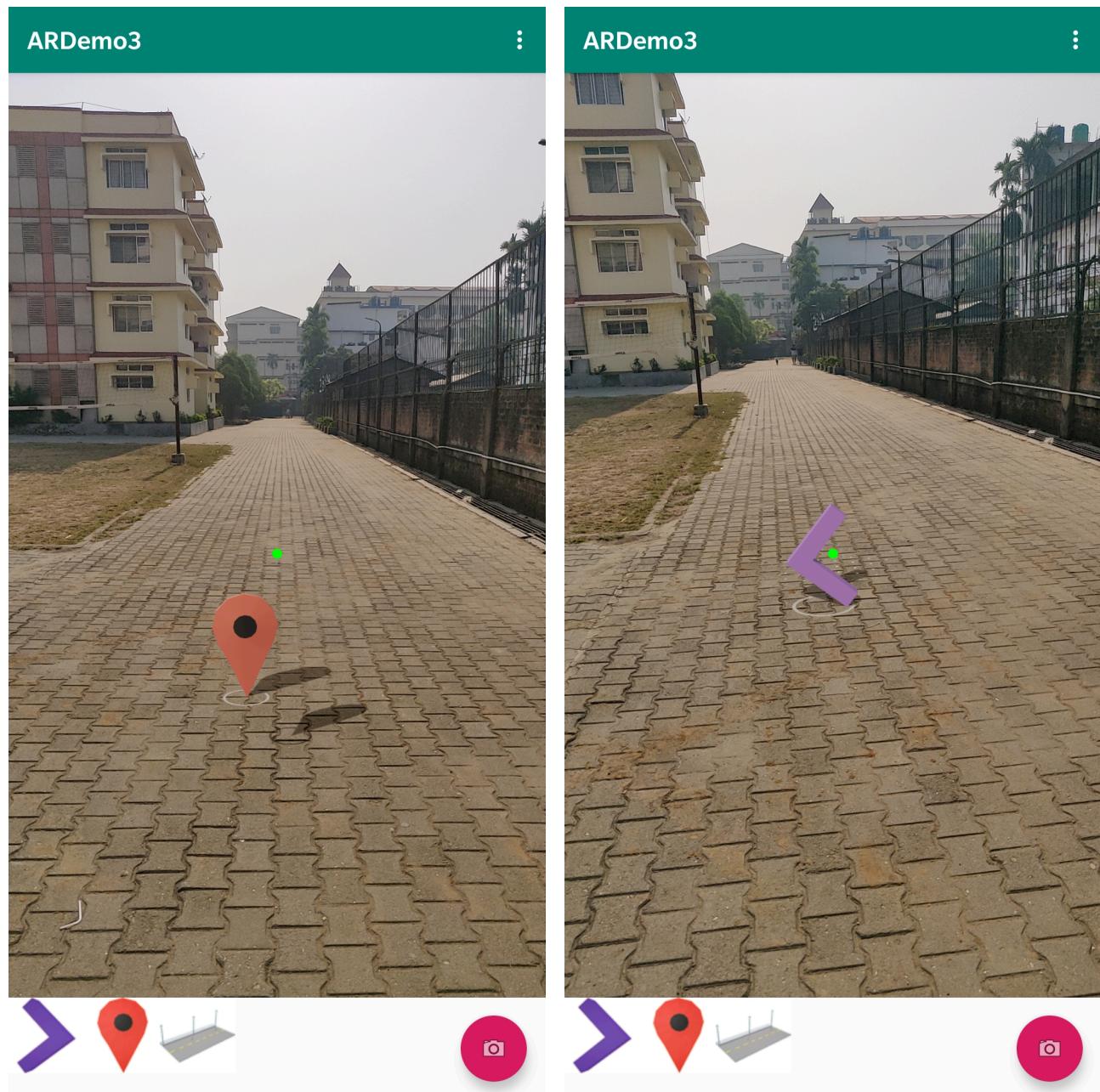
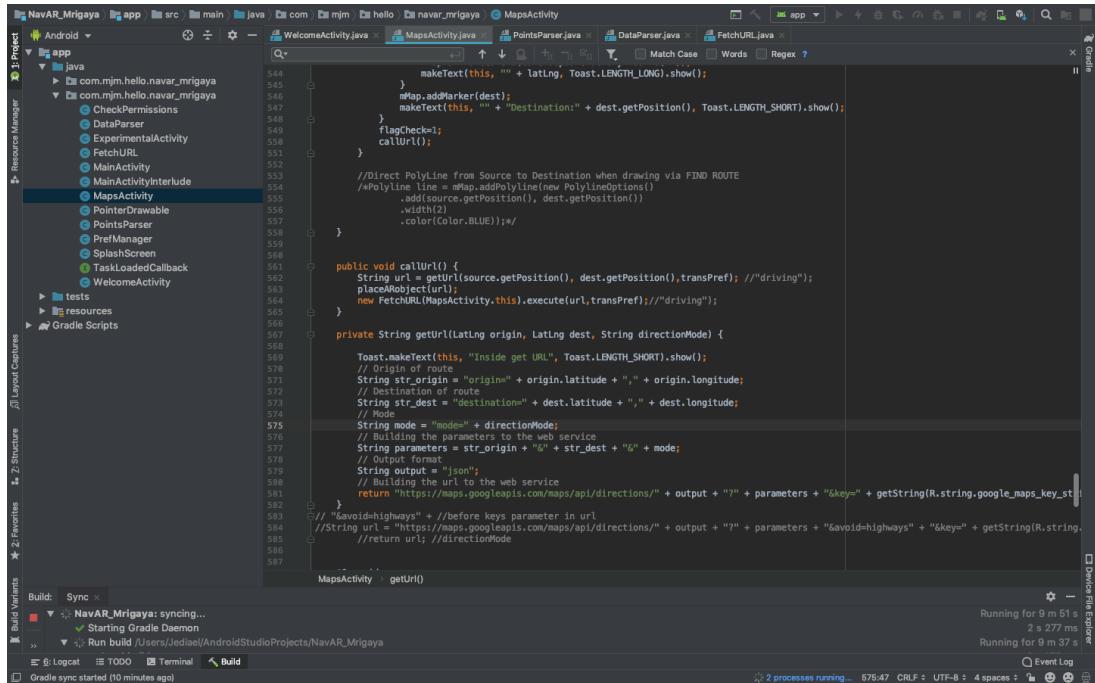


Fig 4.2: Augmented Reality interface

4.2 IMPLEMENTATION

4.2.1 CODE SAMPLES



The screenshot shows the Android Studio interface with the project 'NavAR_Mrigaya' open. The code editor displays the 'MapsActivity.java' file. The code implements a method to get directions from a source to a destination using the Google Maps Directions API. It constructs the URL by concatenating the base URL with parameters like origin, destination, mode (driving), and a 'gavoidhighways' parameter. The code also handles displaying the result in a toast message.

```
       .makeText(this, "" + latLong, Toast.LENGTH_LONG).show();
        mMap.addMarker(dest);
        makeText(this, "" + "Destination" + dest.getPosition(), Toast.LENGTH_SHORT).show();
    }
    flagCheck();
    callUrl();
}

//Direct PolyLine from Source to Destination when drawing via FIND_ROUTE
//Polyline line = mMap.addPolyline(new PolylineOptions()
//.add(source.getPosition(), dest.getPosition())
//.width(12)
//.color(Color.BLUE));
}

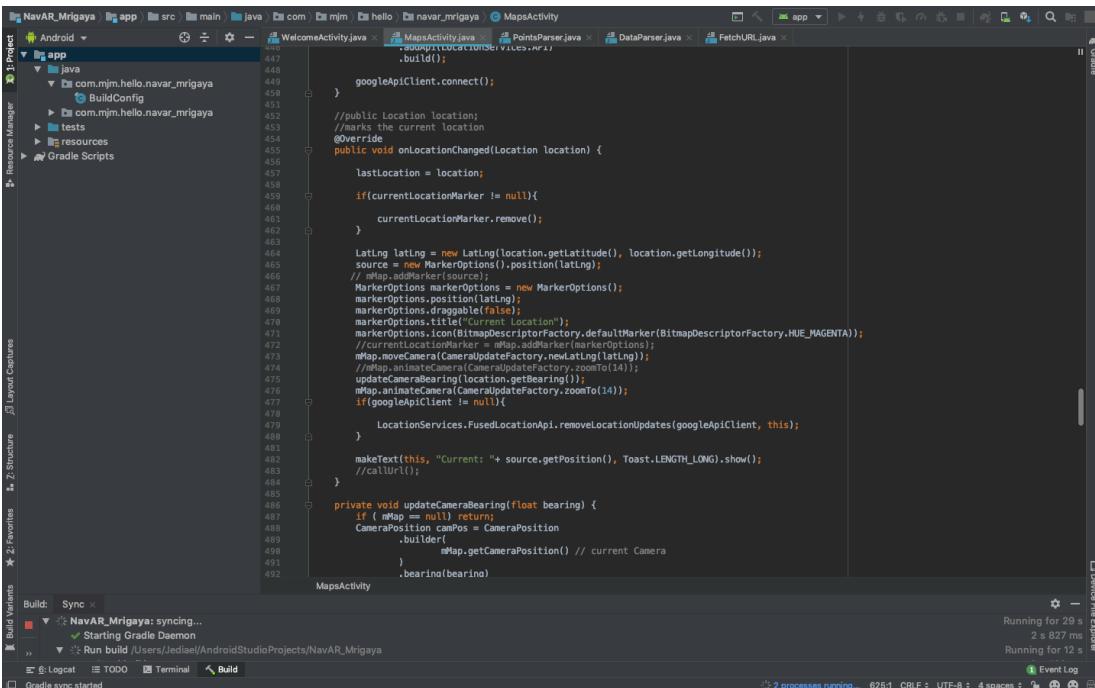
private void callUrl() {
    String url = getSource().getPosition(), dest.getPosition(), transPref); //"/driving";
    placeObject(url);
    new FetchURL(MapsActivity.this).execute(url, transPref); //"/driving");
}

private String getUrl(LatLng origin, LatLng dest, String directionMode) {
    Toast.makeText(this, "Inside get URL", Toast.LENGTH_SHORT).show();
    // Origin of route
    String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
    // Destination of route
    String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
    // Mode
    String mode = "mode=" + directionMode;
    // Building the parameters to the web service
    String parameters = str_origin + "+" + str_dest + "+" + mode;
    // Output format
    String output = "json";
    // Building the url to the web service
    return "https://maps.googleapis.com/maps/api/directions/" + output + "?t" + parameters + "&key=" + getString(R.string.google_maps_key);
}

// "gavoidhighways" + "/before key parameters in url
String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" + parameters + "Gavoid=highways" + "&key=" + getString(R.string.google_maps_key);
return url; //directionMode
}

Build: Sync x
Running for 9 m 51 s
2 s 277 ms
Starting Grade Daemon
Running for 9 m 57 s
Run build /Users/Jediee/AndroidStudioProjects/NavAR_Mrigaya
Event Log
2 processes running... 575:47 CRLF : UTF-8 : 4 spaces : 1m 277 ms
Gradle sync started (10 minutes ago)
```

Fig 4.3: Getting directions from Directions API



The screenshot shows the Android Studio interface with the project 'NavAR_Mrigaya' open. The code editor displays the 'MapsActivity.java' file. The code implements a method to get the current location and update the map's camera bearing. It uses the LocationServices.FusedLocationApi to remove updates and then updates the camera bearing based on the current location.

```
        googleApiClient.connect();
    }

    //public Location location;
    //marks the current location
    @Override
    public void onLocationChanged(Location location) {
        lastLocation = location;
        if(currentLocationMarker != null){
            currentLocationMarker.remove();
        }
        LatLong latLong = new LatLong(location.getLatitude(), location.getLongitude());
        source = new MarkerOptions().position(latLong);
        // mMap.addMarker(source);
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLong);
        markerOptions.draggable(false);
        markerOptions.title("Current Location");
        markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
        //Animate camera
        mMap.animateCamera(CameraUpdateFactory.newLatLng(latLong));
        //mMap.animateCamera(CameraUpdateFactory.zoomTo(14));
        updateCameraBearing(location.getBearing());
        mMap.animateCamera(CameraUpdateFactory.zoomTo(14));
        if(googleApiClient != null){
            LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient, this);
        }
        makeText(this, "Current: " + source.getPosition(), Toast.LENGTH_LONG).show();
        //callUrl();
    }

    private void updateCameraBearing(float bearing) {
        if ( mMap == null) return;
        CameraPosition camPos = CameraPosition
        .builder()
        .target(mMap.getCameraPosition()) // current Camera
        .bearing(bearing)
        .build();
        mMap.animateCamera(CameraUpdateFactory.newCameraPosition(camPos));
    }
}

Build: Sync x
Running for 29 s
2 s 827 ms
Starting Grade Daemon
Running for 12 s
Run build /Users/Jediee/AndroidStudioProjects/NavAR_Mrigaya
Event Log
2 processes running... 626:1 CRLF : UTF-8 : 4 spaces : 1m 827 ms
Gradle sync started
```

Fig 4.4: Getting current location

```

public void initAutoComplete() {
    String apiKey = "AIzaSyC-3sLHSRQNDG0zhWMyMs5WoCKT3tax5Q";
    // Initialize Places.
    Places.initialize(getApplicationContext(), apiKey);
    AutocompleteSupportFragment autocompleteFragment = (AutocompleteSupportFragment) getSupportFragmentManager().findFragmentById(R.id.autocomplete_fragment);
    // Specify the types of place data to return.
    autocompleteFragment.setPlaceFields(Arrays.asList(Place.Field.ID, Place.Field.NAME));
    //autocompleteFragment.setTypeFilter(TypeFilter.REGIONS);
    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
           .makeText(MapsActivity.this, "" + place.getName(), Toast.LENGTH_LONG).show();
        }
    });
    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            makeText(MapsActivity.this, "" + place.getName(), Toast.LENGTH_LONG).show();
        }
    });
    @Override
    public void onPlaceSelected(Place place) {
        makeText(MapsActivity.this, "" + place.getName(), Toast.LENGTH_LONG).show();
    }
    String searchLocation = place.getName();
    List<Address> addressList = null;
    MarkerOptions markerOptions = new MarkerOptions();
    if (!searchLocation.equals("")) {
        Geocoder geocoder = new Geocoder(MapsActivity.this);
        try {
            addressList = geocoder.getFromLocationName(searchLocation, 1);
        } catch (IOException e) {
            e.printStackTrace();
        }
        for (int i = 0; i < addressList.size(); i++) {
            Address myAddress = addressList.get(i);
            LatLng latLng = new LatLng(myAddress.getLatitude(), myAddress.getLongitude());
            dest = new MarkerOptions().position(latLng);
            markerOptions.position(latLng);
            markerOptions.title("Result");
            markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED));
            mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
            mMap.animateCamera(CameraUpdateFactory.zoomTo(14));
            Toast.makeText(MapsActivity.this, "" + latLng, Toast.LENGTH_LONG).show();
        }
    }
}

```

Fig 4.5: Getting autocomplete suggestions from Places API

4.2.2 APP INTERFACE

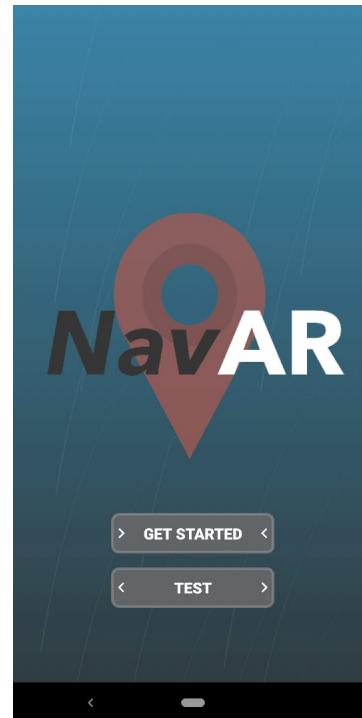


Fig 4.6: Start Screen

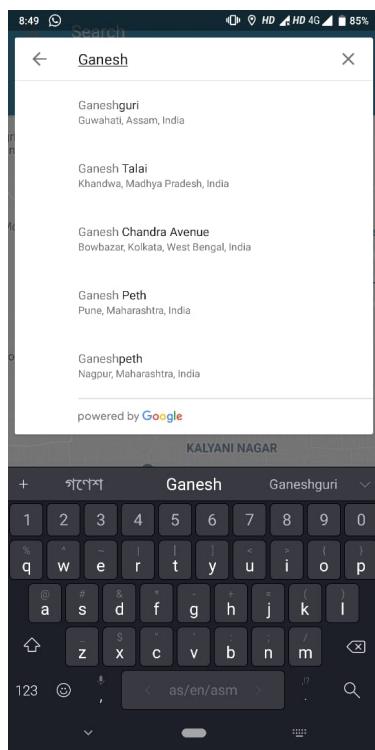


Fig 4.7: Autocomplete in Search

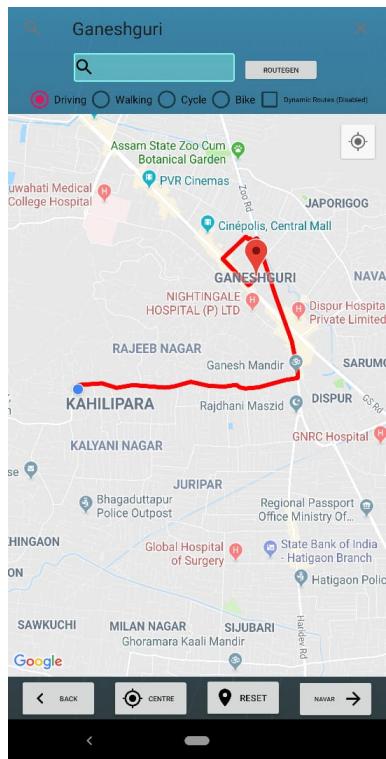


Fig 4.8: Driving Directions

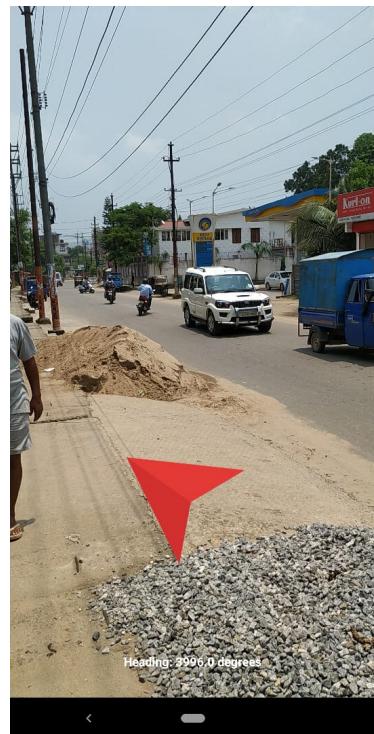


Fig 4.9: Augmented Reality Navigation

Sl. No.	Input	Screenshot No.	Expected Output	Actual Output
1	Ganesh	Fig 4.7	Ganeshguri	Ganeshguri
2	Ganeshguri	Fig 4.8	Marker at Ganeshguri	Marker at Ganeshguri
3	Destination: Dharapur	Fig 4.9	AR navigation to Dharapur	AR navigation to Dharapur

Table 4.1: Test Table

CONCLUSION

With this application, we hope to incorporate Augmented Reality in making our everyday lives easier and more convenient. Our aim for this application is to service navigation in complex roadways, and therefore, help the users to make correct decisions while driving or navigating.

REFERENCES

[1] **Android Studio System Requirements:**

<https://developer.android.com/studio> (02/02/2019)

[2] **COCOMO Model:**

<https://www.geeksforgeeks.org/software-engineering-cocomo-model> (02/02/2019)

[3] **ARCity by blippAR:**

<https://www.blippar.com/blog/2017/11/06/welcome-ar-city-future-maps-and-navigation>
(05/02/2019)

[4] **AR Navigation & Nearby GPS, AR Walking Navigation by AR Technology | GPS Navigation | Future Apps:**

<https://play.google.com/store/apps/details?id=com.arnavigation.poi.browser&hl=en> (09/02/2019)

[5] **ARCore:**

<https://developers.google.com/ar/discover/supported-devices> (20/03/2019)