

National College of Ireland

BSHCSD4

Software Development

2021/2022

Emanuel Ivan

X18313591

X18313591@student.ncirl.ie

NowDrive

Technical Report

Contents

1	Executive Summary.....	3
2	Introduction	3
2.1	Background	3
2.2	Aims.....	3
2.3	Technology	4
2.4	Structure	4
3	System.....	5
3.1	Requirements.....	5
3.1.1	Functional Requirements	5
3.1.2	Data Requirements	13
3.1.3	User Requirements	14
3.1.4	Environmental Requirements	15
3.1.5	Usability Requirements	16
3.2	Design & Architecture	17
3.2.1	Activities:.....	17
3.2.2	Firebase Database:.....	17
3.3	Implementation	19
3.3.1	Registration input validation:.....	19
3.3.2	Settings change user details/delete account:.....	20
3.3.3	Forgot password:	21
3.3.4	Accidents Data JSON:	21
3.3.5	Location Services:.....	22
3.3.6	Map Initialization & Marker Creation:	23
3.3.7	Calculate Route/Routes:	24
3.3.8	Create Directions API Request:	28
3.3.9	Previous Route:	30
3.3.10	Remove current Route:	31
3.3.11	Save Current Route:	32
3.3.12	Search Saved Routes:	34
3.4	Graphical User Interface (GUI).....	36
3.4.1	NowDrive Login (Figure 2.2).....	36
3.4.2	NowDrive Registration (Figure 2.3).....	37
3.4.3	NowDrive Password Reset (Figure 2.4).....	37
3.4.4	NowDrive Home Page (Figures 2.5, 2.5.1, 2.5.2)	38

3.4.5	NowDrive Home Menu (Figure 2.6)	39
3.4.6	NowDrive Settings (Figure 2.7)	39
3.5	Testing	39
3.5.1	Unit Testing	39
3.5.2	Manual Testing/Debugging.....	42
3.6	Evaluation	49
3.6.1	Routing	49
3.6.2	Scalability Costs.....	51
4	Conclusions	52
4.1	Advantages and Strengths	52
4.2	Disadvantages and Weaknesses	53
5	Further Development or Research	53
6	Appendices.....	55
6.1	Project Proposal	55
6.1.1	Objectives.....	56
6.1.2	Background	57
6.1.3	State of the Art.....	57
6.1.4	Technical Approach.....	58
6.1.5	Technical Details	59
6.1.6	Special Resources Required	60
6.1.7	Project Plan	60
6.1.8	Testing.....	63
	References	64
6.2	Reflective Journals	65
6.2.1	October:	65
6.2.2	November:	66
6.2.3	December:.....	67
6.2.4	January 2022:	68
6.2.5	February:	69
6.2.6	March:	70
6.2.7	April.....	71
	Technical Report References	72

1 Executive Summary

This report is a summary of the NowDrive project, NowDrive is an application that focuses on calculating car routes from point A to point B with safety in mind and to avoid potentially dangerous roads that are accident prone. The target group for this application would be New or Learner drivers which would help them plot easier and safer routes which helps them avoid possible accidents that may lower insurance premiums if accident free for the first 5 years of getting the insurance.

The report addresses various headings that discusses a wide variety of details about the project. A high-level view of the current state of NowDrive would be that currently where the project stands it has a very solid backbone that can easily be worked from, it also achieves all requirements outlined in this document.

There are however no APIs or sources of data that can be utilized for the calculation of routes as required currently and, in its place, static data had been created manually by referencing the map of collisions in Ireland provided by the RSA (RSA, n.d.) to be used in the calculation of routes. This has resulted in the number of accidents in the IFSC area being as the only factor in the algorithm when calculating routes. Regardless, all features are functional and scaling upwards to support additional factors can easily be implemented once the data is made available.

2 Introduction

2.1 Background

I undertook this project as I wish to fill a gap in the GPS/Navigation application market as I feel there is no GPS/Navigation mobile application out there with the focus being finding the safest possible route from point A to point B. The reason why this is important is that it would help keep insurance costs low as the application will help you avoid risky roads for driving, this is especially good for New and Learner drivers as the first 5 years of their insurance are crucial for keeping it as claim free as possible to help lower the cost of insurance after the 5-year period.

2.2 Aims

What this project aims to achieve is the following:

Provide an android mobile application with navigation features for the user.

Implementation of an algorithm to calculate safest possible route for the user but at the same time is not too long of a route.

Provide information to the user regarding various choices of routes.

To be deployed around the IFSC area.

Have a Login/Registration feature for the user and provide an interface for optional feedback to improve quality of the mobile application.

2.3 Technology

There are various technologies that have been used or considered for this project:

- Android Studio: This is an IDE that will be used to develop the application, it is used android app development and contains various languages primarily Kotlin/Java and XML.
- Java: This is one of the programming languages I will be using which is used for the logical aspects of the application such as the algorithm, running specific tasks when certain buttons are pressed, handling of data and communication with the database.
- XML: This is another programming language I will be using and it's primarily used for the GUI elements in Android Studio and how data would be presented to the user.
- Firebase: Is a cloud database which is supported by Android Studio and is provided by google and contains all the features that you would expect from a cloud database provider with a free plan.
- Google Maps API: The google maps API will be used as a display for the map where markers are set, and the polyline would be displayed onto.
- HERE Directions API: This API is used in the creation of a polyline that would be displayed on the google maps API showcasing the route.
- data.gov.ie API: This API was supposed to be responsible for gathering and handling historical data to be used in route calculation however due to unforeseen circumstances (see headings 3.3.4, 4.2 and 5) it was replaced with static data.
- data.tii.ie API: This API was supposed to be responsible for gathering and handling concurrent data such as weather and congestion of a route to be used in it's calculation however due to unforeseen circumstances (see headings 3.3.4, 4.2 and 5) it could no longer be considered part of the calculation.

2.4 Structure

The following structure of this document will be:

- Executive Summary: A high-level summary of the report regarding the NowDrive mobile application.
- Introduction: Description of the Background, Aims, Technologies and structure of the report.
- System: The technical aspect of the report discussing Requirements, Design & Architecture, Implementation, Graphical Interfaces and Testing and Evaluation.
- Conclusions: Outlines the advantages and disadvantages of NowDrive currently and concludes all details about functionality of the application.
- Further Development or Research: An in-depth outlined consideration of future development for NowDrive.
- Appendices: Includes attached relevant documentation.
- Technical Report References: Includes references related to the technical report.

3 System

3.1 Requirements

3.1.1 Functional Requirements

The functional requirements for NowDrive are in the following order:

1. User Login: Very High Priority
2. User Registration: Very High Priority
3. Map Route: High Priority
4. User Forgetting Password: High Priority
5. User Settings: Medium Priority

3.1.1.1 Requirement 1: User login

3.1.1.1.1 Description & Priority

This requirement is addressing the login for the User and it is a very high priority as this requirement is fundamental because the User will need to log in to use the features provided by NowDrive and the processes outlined in this requirement will need to have a near perfect implementation on the NowDrive application.

3.1.1.1.2 Use Case

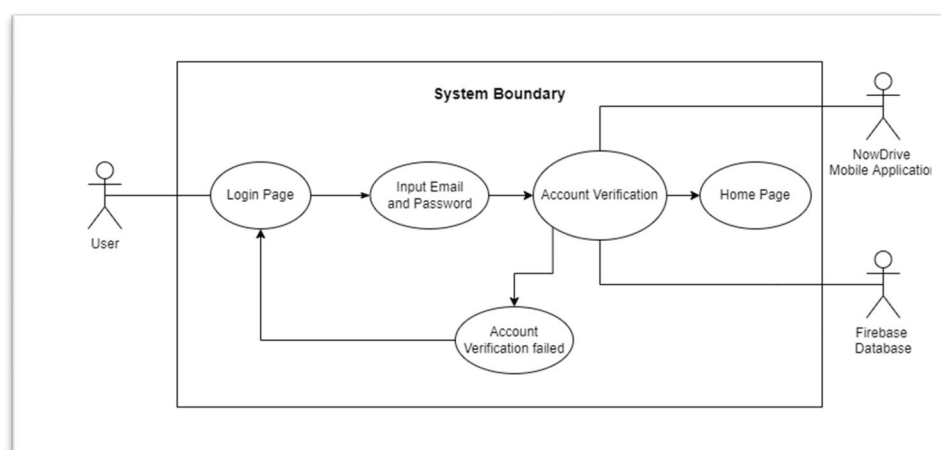
Scope

The scope of this use case is to provide a login feature to the user.

Description

This use case describes the process of the user logging into the NowDrive application.

Use Case Diagram



Flow Description

Precondition

The user has an account already created.

Activation

This use case starts when User starts the application which lands them on the Login page and they wish to log into the Application.

Main flow

1. The User lands on the Login Page.
2. The User inputs Email and Password.
3. The Mobile Application and Firebase Database Verify the inputted Email and Password (See E1).
4. The User Passes verification and lands on Home Page.

Alternate flow

No Alternative flows for this use case.

Exceptional flow

E1 : Account Verification Failure

1. Account Verification Fails for the User
2. The use case continues at position 1 of the main flow

Termination

The User successfully passes the Account Verification and lands on the Home Page.

Post condition

The user is logged in and on the home page.

3.1.1.2 Requirement 2: User Registration

3.1.1.2.1 Description & Priority

This requirement is addressing the registration of a new account by the user and it is a very high priority because if the user is not able to register a new account they will not be able to log into the NowDrive Application and use it's features. This will need to have a near perfect implementation code wise on the application to make sure it works as intended.

3.1.1.2.2 Use Case

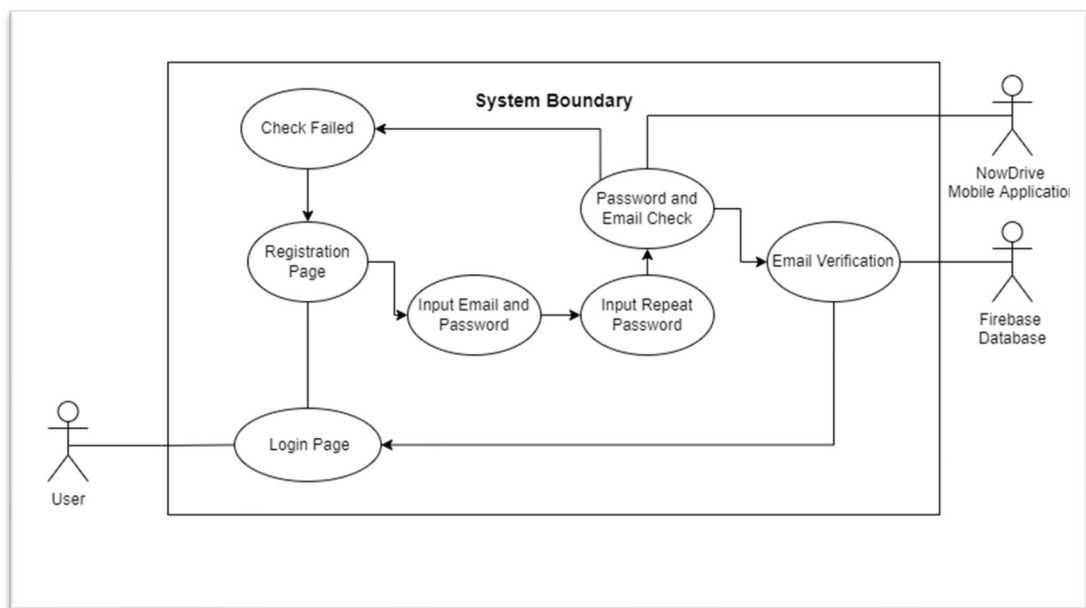
Scope

The scope of this use case is to provide a system where the user can register a new account.

Description

This use case describes the process of the user creating a new account on the NowDrive mobile application.

Use Case Diagram



Flow Description

Precondition

The user has no existing account.

Activation

This use case starts when User starts the application which lands them on the Login page and they don't have an account wishing to create a new account with NowDrive.

Main flow

1. The User lands on the Login Page.
2. The User navigates to Registration Page.
3. The User inputs their Email and Password.
4. The User inputs their Password again.
5. NowDrive performs a check on the inputted information (See E1)
6. Firebase verifies Email.
7. The User lands back on the Login Page.

Alternate flow

No Alternative Flows.

Exceptional flow

E1 : Password and Email Check Failure

1. Password and Email Check fails for User
2. The use case continues at position 2 of the main flow.

Termination

Email verification sent and User is redirected to Login Page.

Post condition

The user had registered an account and has landed back on the Login Page.

3.1.1.3 Requirement 3: Map Route

3.1.1.3.1 Description & Priority

This requirement is addressing how the User interacts with the NowDrive Application when they wish to map a route using the application. This is a high priority although it is not fundamental to the functioning of this application it is important as it addresses the main aim of the NowDrive Application, and it is the main feature being provided.

3.1.1.3.2 Use Case

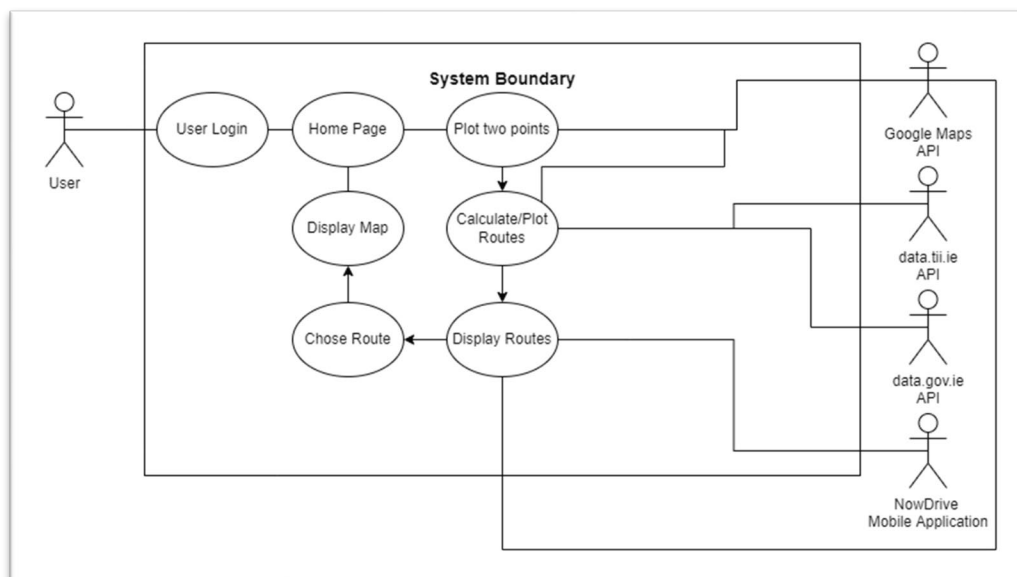
Scope

The scope of this use case is to map the route for the User upon request.

Description

This use case describes the flow of how the route would be plotted for the User.

Use Case Diagram



Flow Description

Precondition

The User has an account.

Activation

This activates when the User opens up the app and wishes to find a route.

Main flow

1. The User logs in (See Requirement 1)
2. The User lands on the Home Page (See A1)
3. The User plots two points on the Home Page
4. Routes are then calculated with the given two points on the map using the Google Maps API, data.tii.ie API, data.gov.ie API.
5. The routes are then displayed using the Google Maps API and NowDrive Mobile Application System.
6. The User chooses a Route
7. The User can see the Display Map

Alternate flow

A1: Previous Route

1. The User lands on Home Page.
2. The User can view routes previously calculated with a press of a button which will load on the map replacing the current route.

Exceptional flow

There are not exceptional flows.

Termination

The use case ends once a map is displayed with the routes.

Post condition

The user has a route that they are following.

3.1.1.4 Requirement 4: User Forgetting Password

3.1.1.4.1 Description & Priority

This requirement is addressing the Password reset requirement for the User if they forget their password, this has a priority of High because if the User is unable to reset their password and use their account they will not be able to use the NowDrive Application.

3.1.1.4.2 Use Case

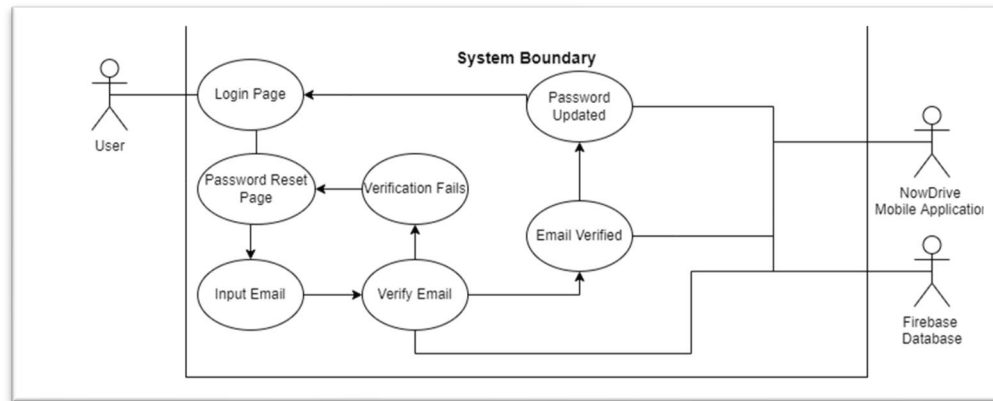
Scope

The scope of this use case is to provide a forget password feature so that the User can reset their password if forgotten.

Description

This use case describes the process of how the User will be able to reset their password in NowDrive.

Use Case Diagram



Flow Description

Precondition

The User has an existing account but they forgot their password.

Activation

This use case starts when they open the mobile application and wishes to reset their password as they had forgotten it.

Main flow

1. The User lands on the Login Page.
2. The User navigates to the Password Reset Page.
3. The User Inputs their Email.
4. The Email goes through verification process with Firebase and Mobile application checks (See E1).
5. The Email is verified and by the mobile application and Firebase and an email is sent to the User.
6. The password is updated via email by the user and is updated on the application and Firebase database.
7. User land on Login Page.

Alternate flow

There are no alternative flows

Exceptional flow

E1 : Email verification fails

1. Email verification fails for User
2. The use case continues at position 2 of the main flow.

Termination

The User successfully resets his password using the email send to them and is back on the Login Page.

Post condition

The User had successfully reset their password which is updated on the firebase database and would be able to log in (See requirement 1) with the new password and is currently back on the Login Page.

3.1.1.5 Requirement 5: User Settings

3.1.1.5.1 Description & Priority

This requirement is addressing how the User interacts with the Settings in the NowDrive mobile application, this has a priority of medium as it is not fundamental to the operations of the mobile application but more of a quality of life and giving the user more options, such as changing their Email and Password as well as removing their account if they want.

3.1.1.5.2 Use Case

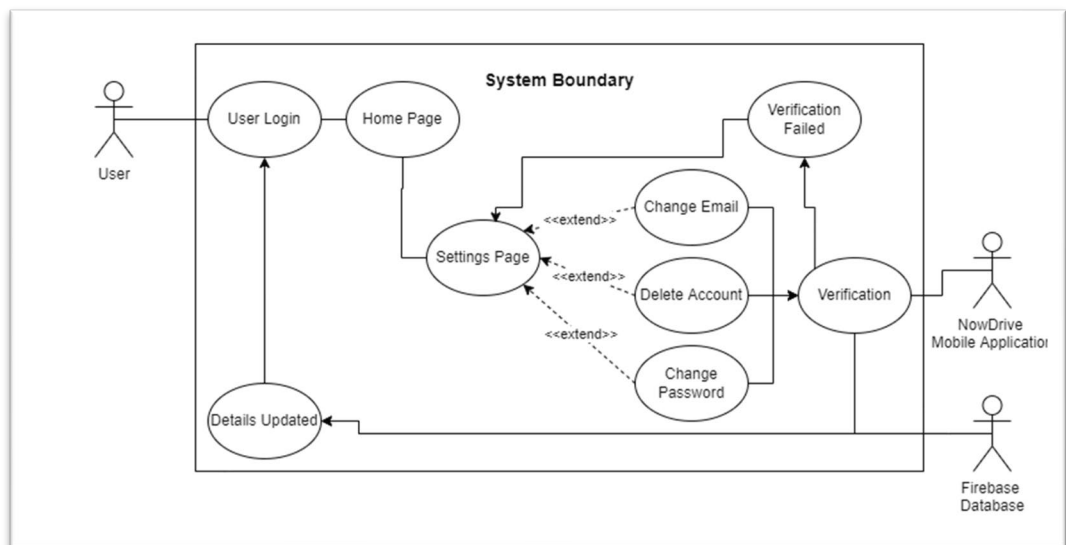
Scope

The scope of this use case is to provide Settings to the User so that they can change their email or passwords or, remove their account.

Description

This use case describes the process of how the User will be able to interact with the Settings in the NowDrive mobile application.

Use Case Diagram



Flow Description

Precondition

The User has an existing account.

Activation

This use case starts when he opens the mobile application and wishes to go to his settings.

Main flow

1. The User logs in (See Requirement 1)
2. The User lands on the Home Page
3. The User navigates to the Settings Page (See A1, A2 and A3)

Alternate flow

A1: Change Email

1. The User Changes their email
2. The NowDrive Mobile Application verifies the account (See E1)
3. The Details are updated by the Firebase Database and NowDrive Mobile Application
4. User lands on Login Page

A2: Change Password

1. The User Changes their Password
2. The NowDrive Mobile Application verifies the account (See E1)
3. The Details are updated by the Firebase Database and NowDrive Mobile Application
4. User lands on Login Page

A3: Delete Account

1. The User deletes their account which is updated on the firebase database as well.
2. The NowDrive Mobile Application verifies the account (See E1)
3. The Details are updated by the Firebase Database and NowDrive Mobile Application
4. The User lands on the Login Page

Exceptional flow

E1 : Password or Email change verification fails

1. Password or Email verification fails for User
2. The use case continues at position 3 of the main flow.

Termination

The User accessed the settings and/or removed their account, changed their password, changed their email.

Post condition

The User had accessed the Settings and/or removed their account, changed their password, changed their email for their account.

3.1.2 Data Requirements

There are various data requirements that have been identified and are in the following order:

1. User Login, Registration and Settings
2. Map Route

3.1.2.1 Requirement 1: Account Data

The following table below describes the data in an “Account” object addressing login, registration and settings functional requirements, this object contains the data addressing those three requirements and will need to be implemented onto both the application and Firebase database.

Attribute:	Data Type:
UID (Primary Key)	int
email	String
password	String
routes	Route[]

Table 1: Account Object Attributes and Type

3.1.2.2 Requirement 2: Map Data

The following table describes the data being used in the Map Route requirement and is represented in a data flow diagram using Yourdon and Coad notation. This shows the relationship between the APIs implemented in NowDrive that supports the Map Route requirement.

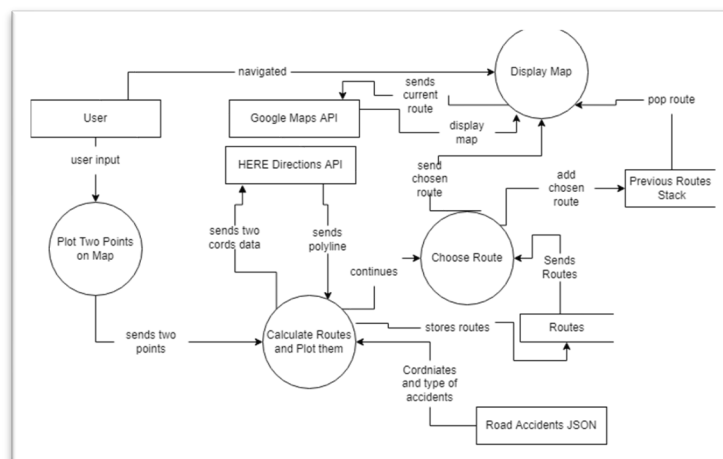


Figure 3.1: Data flow Diagram Map Route

The route data object is primarily used and stored locally on the app but the chosen route will be added to a Routes Stack if they ever want to check their previous chosen routes in the same app instance. Below is a table describing the attributes and type for the “Route” object.

Table 2: Route Object Attributes and Type

Attribute:	Data Type:
route_id (Primary Key)	int
route_name	String
originLat	double
originLng	double
destLat	double
destLng	double
polyline	String
avoidHighways	Boolean
avoidTolls	Boolean
duration	int
length	int

3.1.3 User Requirements

There are various user requirements that have been currently identified and is in the following order:

1. Login: Very High Priority
2. Register new account: Very High Priority
3. Plot Two points: High Priority
4. Receive Routes: High Priority
5. Choose Routes: High Priority
6. Search Saved Routes: High Priority
7. View Previous Created Routes: High Priority
8. View Map/Route on map: High Priority
9. Alter Account/Remove Account: Medium Priority

3.1.3.1 Requirement 1: Login

This requirement acknowledges that the user should be able to login into the mobile application as it would allow them to use the features provided by NowDrive making this requirement a Very High Priority.

3.1.3.2 Requirement 2: Register new account.

This requirement acknowledges that the user should be able to register an new account with NowDrive which then can be used to log into the NowDrive application and because you need to have an account to use NowDrive features this is a Very High Priority.

3.1.3.3 Requirement 3: Plot Two Points

This requirement acknowledges that the user should be able to plot two points on the google maps API of their choosing if it is within the IFSC area. This has a high priority as it is one of the main features of the NowDrive application.

3.1.3.4 Requirement 4: Receive Routes

This requirement acknowledges that the user should be able to receive multiple routes from the two points they had plotted within the IFSC area and has a high priority because of it being one of the main features.

3.1.3.5 Requirement 5: Choose Routes

This requirement acknowledges that the user should be able to choose a route they want from the provided number of routes from NowDrive and has a high priority as it is one of the main features of the NowDrive application.

3.1.3.6 Requirement 6: Search Saved Routes

This requirement acknowledges that the user should be able to search of previously saved routes that is stored on the database and when selected it would replace the current route with the selected one from the database.

3.1.3.7 Requirement 7: View Previously Created Routes

This requirement acknowledges that the user should be able to view previously created routes with a press of a button, which would replace the current routes with routes that have been created previously.

3.1.3.8 Requirement 8: View Map/Route on map

This requirement acknowledges that the user should be able to view the map with or without routes present on it and has a high priority as it is a main feature of the NowDrive application.

3.1.3.9 Requirement 9: Alter Account/Remove Account

This requirement acknowledges that the user should be able to change their Email and/or Password or remove their account if they wish to do so, this has a medium priority as it is a more of a quality-of-life feature and doesn't greatly impact the fundamentals of the NowDrive application and isn't a main feature.

3.1.4 Environmental Requirements

There are two environmental requirements which is in the following order:

1. Device: Very High Priority
2. Internet Connection: Very High Priority

3.1.4.1 Requirement 1: Device

This environmental requirement acknowledges that the device will need to be operational on an android smartphone with the android version 5.X or greater and must have the minimum specifications of the following:

- CPU: Quad Core 1.2GHz
- RAM(GB): 2
- Internal (GB): 16
- Supports: Geolocation

This is a Very High Priority we need to have the mobile application working on the provided specifications as the provided specifications are low and the NowDrive application will not be too demanding that requires a much stronger specification. It does not require any speakers or cameras but will require to be able to send out it's geolocation to the application when given permissions to do so by the User and upon request from the application.

3.1.4.2 Requirement 2: Internet connection

This environmental requirement acknowledges that an internet connection is required for the usage of this application and the time it takes to perform calculations for routes, login/registration as well as updating the google maps API real-time is dependant on the internet connection of the User because they are dealing with external APIs not part of the NowDrive application locally. The priority is Very High because if the user does not have an internet connection they will not be use any of the features provided by NowDrive.

3.1.5 Usability Requirements

There are various usability requirements which are in the following order:

1. Easy to learn: Very High Priority
2. Performance: Very high Priority
3. Flow: Low Priority

3.1.5.1 Requirement 1: Easy to learn

This usability requirement acknowledges that the user is able to learn how to use the mobile application quickly without having any sort of training but have a common understanding of how to use a smart phone, the priority is Very High because if the User is not able to understand how to use NowDrive quickly it would delay their journey as they spend more time trying to figure out how to use the app instead of travelling to their destination.

3.1.5.2 Requirement 2: Performance

This usability requirement acknowledges that at least 90% of users should be able complete the main task of selecting two points on the map and selecting a route that would be displayed allowing them to follow it assuming that they are travelling in the IFSC area, have an account and meet the minimum device requirements. The priority is Very High as the mobile application is not being deployed out for Many Users and is limited in area and so long as conditions are met the application should work for at least more than 90% of Users.

3.1.5.3 Requirement 3: Flow

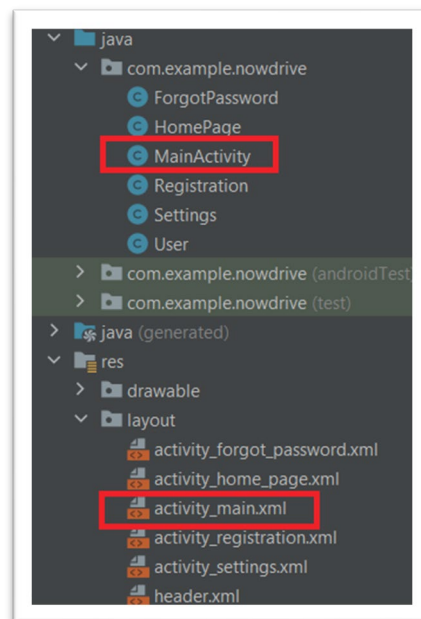
This useability requirement acknowledges that the user should also be able to close the app at anytime and if a Route had been plotted and the application had been closed suddenly it would open the last Route that was plotted for the User on the Google Maps API from a Route Stack Object. This has a priority of Low as

although it is nice to have a previous routes feature, it is not fundamental or a main feature of the application.

3.2 [Design & Architecture](#)

3.2.1 [Activities:](#)

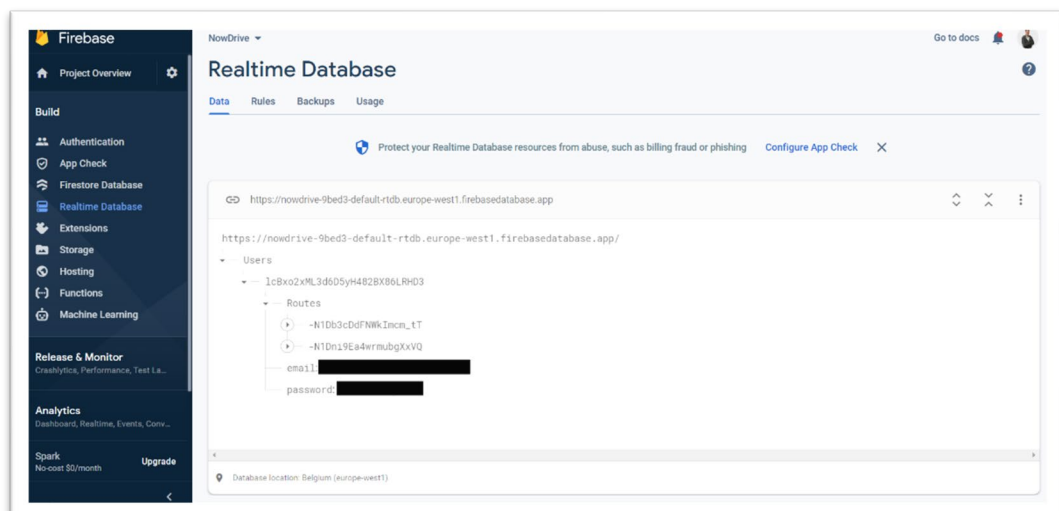
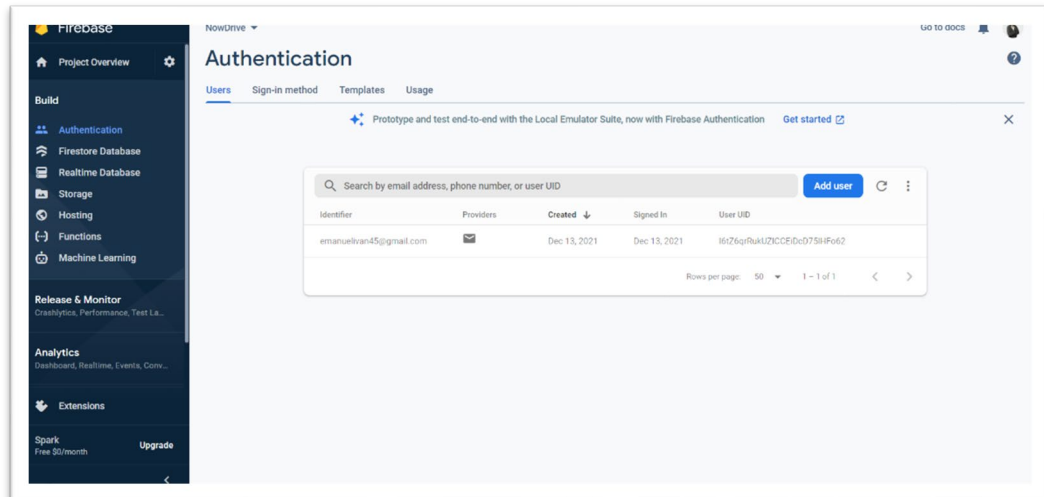
In Android Studio each feature is broken up into an “Activity” that the user can engage with which is then split into an xml file and java file. The xml file is responsible for the UI elements of the activity such as the search bar, buttons, text and user inputs which has an ID allocated to each of the UI elements presented in the xml file where it then gets utilized in the java file as it is able to identify the element using it’s ID and is able to perform modifications to the targeted UI element, gather values from the UI element inputted by the user or perform an action when the UI element is tapped by the User. Below is an example of a screenshot taken from the IDE which shows the two files that handle the Login feature for the application, which is the MainActivity java file and axctivity_main xml file.



3.2.2 [Firebase Database:](#)

The Firebase Database is what currently handles the data for the nowDrive application. Firebase is being utilised using two of it’s features which is its user Authentication and Realtime database. Implementation of firebase User Authentication is currently present in all activities of the NowDrive application which get’s the instance of the User Authentication at the top of the Java file and is declared as the variable **mAuth**. The realtime firebase implementation stores both user login details as well as their saved routes. The database works hand in hand with the User java class which contains the two variables password and email it also contains the Route java class which contains data regarding a route. When structing the data the inputs would be sent as a User or Route object to the

database for consistency. Below I included two screenshots of the Firebase User Authentication at work which is on the top and the bottom is the Database at work containing user information with two routes saved under the user, both of which are accessible on the Firebase website.



3.3 Implementation

3.3.1 Registration input validation:

```
private void registerUser() {
    String email = reg_email.getText().toString().trim();
    String password = reg_pass.getText().toString().trim();
    String passwordTwo = reg_pass_two.getText().toString().trim();

    if(email.isEmpty()){
        reg_email.setError("Email is required");
        reg_email.requestFocus();
        return;
    }

    if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        reg_email.setError("Please provide valid email");
        reg_email.requestFocus();
        return;
    }

    if(password.isEmpty()){
        reg_pass.setError("Password is required");
        reg_pass.requestFocus();
        return;
    }

    if(password.length() < 6){
        reg_pass.setError("Password must be greater than 6 characters");
        reg_pass.requestFocus();
        return;
    }
    Pattern password_pattern = Pattern.compile("(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*~]).{6,}$");
    Matcher matcher = password_pattern.matcher(password);
    if(!matcher.matches()){
        reg_pass.setError("Password must contain a Number, a Capital letter and a symbol");
        reg_pass.requestFocus();
        return;
    }

    if(!passwordTwo.equals(password)){
        reg_pass_two.setError("Passwords must match");
        reg_pass_two.requestFocus();
        return;
    }
}
```

Above is a code snippet of the input validation for Registration. The first three lines above are simply gathering the values as a String type and is then used in if statements below. The code checks if the email is empty if it is it would stop the registerUser() method and notify the user that an email is needed, it also checks if it has a valid format (i.e johnDoe33@gmail.com) with the help of the Patterns object and if found not true it would stop the method asking the user to input a valid email. Similarly to the email the password is checked to make sure it is not empty and if it is it would stop the method and notify the user. The password also needs to be validated, in this case it must be greater than 6 characters and contain a Capital Letter, Number and Special Character to ensure a decent password is made. And the last step is that it checks that the repeat password matches their password to ensure they have their password correct before creating the account.

3.3.2 Settings change user details/delete account:

```
private void verifyAccount() {
    String email = ver_email.getText().toString().trim();
    String pass = ver_pass.getText().toString().trim();
    AuthCredential credential = EmailAuthProvider.getCredential(email, pass);
    user.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if(task.isSuccessful()){
                if(changeType.equals("pass")){
                    user.updatePassword(changedPass).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful()){
                                DatabaseReference ref = FirebaseDatabase.getInstance().getReference().child("Users");
                                HashMap hashMap = new HashMap();
                                hashMap.put("password", changedPass);
                                ref.child(user.getId()).updateChildren(hashMap).addOnSuccessListener(new OnSuccessListener<Void>() {
                                    @Override
                                    public void onSuccess(@NonNull Object o) {
                                        Toast.makeText( context, Settings.this, text, "Password data successfully updated!", Toast.LENGTH_LONG).show();
                                    }
                                });
                                Toast.makeText( context, Settings.this, text, "Password Successfully updated!", Toast.LENGTH_LONG).show();
                                Intent i = new Intent( context, Settings.this, MainActivity.class);
                                i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                                startActivity(i);
                                finish();
                            } else {
                                Toast.makeText( context, Settings.this, text, "An Error has occurred updating the password", Toast.LENGTH_LONG).show();
                            }
                        }
                    });
                    ver_email.setVisibility(View.GONE);
                    ver_pass.setVisibility(View.GONE);
                    ver_btn.setVisibility(View.GONE);
                    return;
                }
            }
        }
    });
}
```

Above is a code snippet which only a part of the settings feature, in this case it is displaying the update password feature. After a user verifies that it is them by typing in their current email and password it checks what they are trying the change which is stored in the changeType variable. There are three possible values for this: "pass", "email", "acc". "pass" would target the logic for changing the password, "email" is for changing email and "acc" is for deleting the account. After it passes account verification and checks the changeType variable it updates the password/email or deletes the account on the Firebase Authenticator and then performs that same action for the Firebase database. Regardless of if it was successful or not it would disappear the verification fields used to verify the user which only had appeared when they pressed the button to change account details or delete their account. After a successful action they are redirected to the login page and logged out.

3.3.3 Forgot password:

```
private void resetPassword() {
    String email = fp_email.getText().toString().trim();

    if(email.isEmpty()){
        fp_email.setError("Email is required!");
        fp_email.requestFocus();
        return;
    }

    if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        fp_email.setError("Please provide a valid email");
        fp_email.requestFocus();
        return;
    }

    fp_progressBar.setVisibility(View.VISIBLE);
    auth.sendPasswordResetEmail(email).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if(task.isSuccessful()){
                Toast.makeText(context, "Check your email to reset your password!", Toast.LENGTH_LONG).show();
                fp_progressBar.setVisibility(View.GONE);
                finish();
            } else{
                Toast.makeText(context, "Something went wrong! Please make sure your email is correct", Toast.LENGTH_LONG).show();
                fp_progressBar.setVisibility(View.GONE);
            }
        }
    });
}
```

This code snippet is for the forgot password feature, here it performs a similar check as seen in registration which is to make that there is a email inputted and that it is valid, it then uses the Firebase Authenticator to send out a verification email if an account exists with the email provided, otherwise it would fail and notify the user.

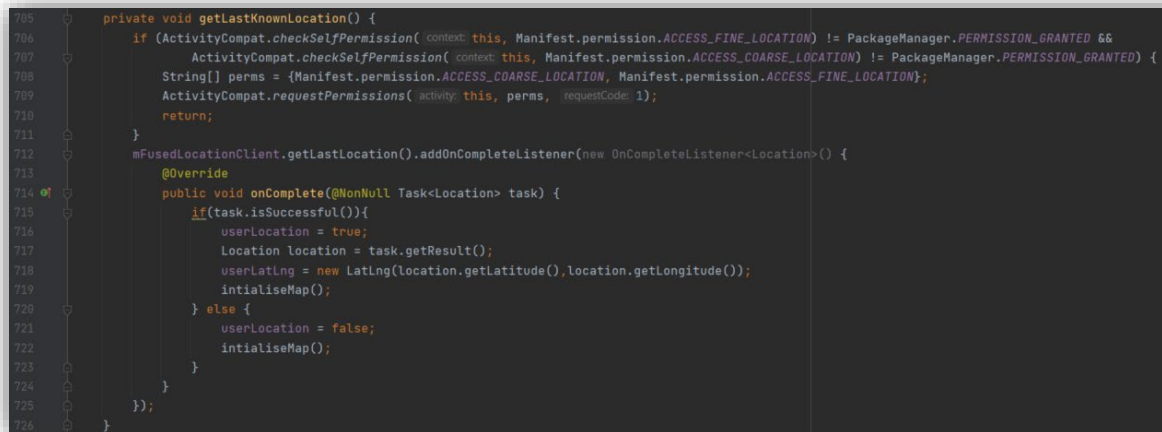
3.3.4 Accidents Data JSON:

```
1 {
2   "accidents": [
3     {
4       "cordLat": 53.348098,
5       "cordLng": -6.248252,
6       "minor": 3,
7       "serious": 0,
8       "fatal": 0
9     },
10    {
11      "cordLat": 53.347957,
12      "cordLng": -6.245945,
13      "minor": 3,
14      "serious": 0,
15      "fatal": 0
16    },
17    {
18      "cordLat": 53.348288,
19      "cordLng": -6.251594,
20      "minor": 5,
21      "serious": 2,
22      "fatal": 0
23    },
24  ],
25 }
```

Before we delve into the more complex aspects of NowDrive regarding Routes, Maps and Locations, we will have to look on where the data is being gathered from. The original plan was to have data extracted from various APIs to be inputted into the algorithm for calculation. Unfortunately, that was not possible (more details under headings 4 and 5). To circumvent this, I had to manually create a JSON file to be used locally in the algorithm to simulate the handling of a JSON response from an API. I manually had to look at the map of road collisions in Ireland on the RSA's website (RSA, n.d.) and pin point the coordinates of them as well as

the three different types of accidents being Minor, Serious and Fatal all of which can be noted in the code snippet above. This JSON file get's utilized in the creation of requests for the direction API (more info under heading 3.3.8). The JSON file is called dockland_accidents.json is under the assets folder in the project.

3.3.5 Location Services:



```
705 private void getLastKnownLocation() {
706     if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
707         ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
708         String[] perms = {Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION};
709         ActivityCompat.requestPermissions( activity: this, perms, requestCode: 1);
710         return;
711     }
712     mFusedLocationClient.getLastLocation().addOnCompleteListener(new OnCompleteListener<Location>() {
713         @Override
714         public void onComplete(@NonNull Task<Location> task) {
715             if(task.isSuccessful()){
716                 userLocation = true;
717                 Location location = task.getResult();
718                 userLatLng = new LatLng(location.getLatitude(),location.getLongitude());
719                 initialiseMap();
720             } else {
721                 userLocation = false;
722                 initialiseMap();
723             }
724         }
725     });
726 }
```

From this point on we will be focusing on Location, Maps and Routes codes of NowDrive all of which are in the HomePage.java file. Firstly, we will be discussing on how NowDrive gathers user location.

As you can see from the code sniper above there is a method called getLastKnownLocation() which gets called in the constructor of the HomePage.java file. The first if statement is responsible in checking if permission was given to the application to Utilize user's location, if not it will then return empty to the constructor.

There is logic in place to handle if the user decides not to give permissions although the app strongly suggests turning it on, regardless with or without the permissions the app will continue to work the only difference is that if there's no permission the User cannot set their origin at their current location.

After the if statement it will then call on the location client object type which had been in the initialized in the constructor of the HomePage.java and if it is successfully the global Boolean variable of user location will be set to true, gathering the result and setting the setting the global variable of userLatLng which is an object type of LatLng to the location of the user before continuing on to call initialiseMap() which is responsible for initializing the map for NowDrive.

3.3.6 Map Initialization & Marker Creation:

```
671 private void initialiseMap() {
672     if (ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION)
673         != PackageManager.PERMISSION_GRANTED
674         && ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_COARSE_LOCATION)
675         != PackageManager.PERMISSION_GRANTED) {
676         String[] perms = {Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION};
677         ActivityCompat.requestPermissions(activity: this, perms, requestCode: 1);
678     }
679
680     if(userLocation){
681         map.setMyLocationEnabled(true);
682     }
683
684     map.setOnMapLongClickListener(new GoogleMap.OnMapLongClickListener() {
685         @Override
686         public void onMapLongClick(@NonNull LatLng point) {
687             if(markers.size()<2){
688                 MarkerOptions userMarker = new MarkerOptions().position(point);
689                 if(markers.size()<1){
690                     userMarker.title("Point A");
691                 } else {
692                     userMarker.title("Point B");
693                 }
694                 Marker userMark = map.addMarker(userMarker);
695                 markers.add(userMark);
696                 Toast.makeText(context: HomePage.this, text: "Point created!", Toast.LENGTH_LONG).show();
697             }
698             else {
699                 Toast.makeText(context: HomePage.this, text: "Number of points exceeds 2 please clear the points.", Toast.LENGTH_LONG).show();
700             }
701         }
702     });
703 }
```

As seen from the previous heading the method seen in the code snippet above (initialiseMap()) is called after a user's location is gathered but it is also called when there is no location permissions either.

The first if statement like in the getLastKnownLocation() method is responsible for checking if user location permissions had been given, if not it will ask permissions once again from the user. The next if statement is responsible for setting myLocationEnabled on the global variable of map which is an object type of GoogleMap.

After that, an OnMapLongClick listener is created which checks if the global variable of markers (which is an array of Marker objects) if the size is less than 2, if it is then it would create markers (otherwise it would alter the user that it exceeds the maximum size of markers) , if the size is less than 1 it will give the marker title of "Point A" if it's the 2nd one it will be given the title of Point B. It will then create a Marker variable with the MarkOptions variable as seen in the code snippet, afterwards adding it to the markers array and displaying a notification via Toast to the user saying the post was created.

3.3.7 Calculate Route/Routes:

3.3.7.1 Code Segment 1

```
326 calRoutes.setOnClickListener(new View.OnClickListener() {
327     @Override
328     public void onClick(View view) {
329         LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
330         View popupView = inflater.inflate(R.layout.cal_route_popup, root: null, attachToRoot: false);
331         int width = LinearLayout.LayoutParams.WRAP_CONTENT;
332         int height = LinearLayout.LayoutParams.WRAP_CONTENT;
333         boolean focusable = true;
334
335         PopupWindow popupWindow = new PopupWindow(popupView, width, height, focusable);
336         popupWindow.setBackgroundDrawable(new BitmapDrawable());
337         popupWindow.setOutsideTouchable(true);
338
339         popupWindow.showAtLocation(view, Gravity.CENTER, 0, 0);
340
341         popupWindow.getContentView().setVisibility(View.GONE);
342
343         if(currentPoly!=null){
344             currentPoly.remove();
345         }
346         if(markers.size()<2){
347             Toast.makeText(context: HomePage.this, text: "Two points must be placed!", Toast.LENGTH_LONG).show();
348             popupWindow.dismiss();
349         }
350     }
351 }
```

The code segment above represents the start of the OnClickListener for when the calculate routes button is pressed. It starts off by inflating a popup that may or may not be used depending if there is going to be multiple options for routes available thus, lines 329 to 341 is all responsible for setting up the popup that may be set visible later on. The if statement on line 343 is checking if the global variable currentPoly of type PolyLine is not null, if it is not null then it would remove the PolyLine from the map. It also checks if the number of markers is less than 2 on the map on line 346, if so it then notifies the user that two points must be placed, dismissing the invisible popup and ending the method there.

3.3.7.2 Code Segment 2

```
352 } else {
353     gatheredRoutes.clear();
354     Marker originMark = null;
355     Marker destMark = null;
356     for(int i=0;i<markers.size();i++){
357         if(i==0){
358             originMark = markers.get(i);
359         } else {
360             destMark = markers.get(i);
361         }
362     }
363     queue.cancelAll( "Here Directions API Call");
364     calBar.setVisibility(View.VISIBLE);
365     ArrayList<StringRequest> strReqs = createStringRequest(originMark.getPosition().latitude,originMark.getPosition().longitude,destMark.getPosition().latitude,destMark.getPosition().longitude, modeReq: false);
366
367     for (int i=0;i<strReqs.size();i++){
368         StringRequest strReq = strReqs.get(i);
369         strReq.setTag("Here Directions API Call");
370         queue.add(strReq);
371     }
372
373     Marker finalOriginMark = originMark;
374     Marker finalDestMark = destMark;
375 }
```

This code segment is a continuation of the OnClickListener method at the use case where the number of markers is not less than 2. It first clears the global variable of gatheredRoutes variable which is an array of the Route object and initializing originMark and destMark Marker variables.

It then performs a for loop, looping through the number of markers and setting the originMark the value of the Route in position 0 of the markers array and the destMark of the Route in position 1 in the Markers array.

The global variable of queue which is type of RequestQueue(Originates from Volley Gradle (Google, n.d.)) cancels all Requests with the tag (Here Directions API call) before setting the loading bar animation to be visible. Afterwards an array of StringRequest types is created by calling the createStringRequest() method with the route Flag of false (more information under heading 3.3.8).

It then loops through the array giving it the requests tags of “Here Directions API call” and adding them to the queue. Afterwards the variables of finalOriginMark and finalDestMark are assigned the same value as originMark and destMark.

3.3.7.3 Code Segment 3

```
372 queue.addRequestFinishedListener(new RequestQueue.RequestFinishedListener<Object>() {  
373     @Override  
374     public void onRequestFinished(Request<Object> request) {  
375         ArrayList<StringRequest> finalStrReqs = new ArrayList<>();  
376         finalStrReqs = createStringRequest(finalOriginMark.getPosition().latitude, finalOriginMark.getPosition().longitude, finalDestMark.getPosition().latitude, finalDestMark.getPosition().longitude, false, true);  
377         if(finalStrReqs.isEmpty()){  
378             Route selectedRoute = gatheredRoutes.get(0);  
379             calBar.setVisibility(View.GONE);  
380             previousRoutes.add(selectedRoute);  
381  
382             PolylineOptions polyOpt = new PolylineOptions().addAll(PolyUtil.decode(selectedRoute.encodedPolyLine)).width(5).color(Color.RED).geodesic(true).jointType(JointType.ROUND);  
383             currentPoly = map.addPolyline(polyOpt);  
384             currentDur = Integer.parseInt(selectedRoute.duration);  
385             currentLength = Integer.parseInt(selectedRoute.length);  
386             int duration = (int) Math.round(Integer.parseInt(selectedRoute.duration)/60);  
387             durView.setText("Duration: " + duration + " minutes");  
388             durView.setVisibility(View.VISIBLE);  
389             lengthView.setText("Length: " + selectedRoute.length + " m");  
390             lengthView.setVisibility(View.VISIBLE);  
391             popupWindow.dismiss();
```

This code segment is a continuation from the code segment before. The queue as mentioned from the previous segment is given a RequestFinishedListener which is handles the logic when the queue is finished sending requests. A new array of requests is created and is assigned the value gathered from the called method of createStringRequest with the route flag true (more information under heading 3.3.8). If it is empty after it was assigned the value then it will create a new Route variable and assigning the value of index 0 of the requests array to it. The bar then disappears to show that the calculation is finished for the user and the new route is added to the previousRoutes stack array type.

Then from lines 382-391 is responsible for handling the display of polyline, duration text, length text and dismissing the invisible popup.

3.3.7.4 Code Segment 4

```
392 } else {
393     queueTwo.cancelAll("Here Directions API Call");
394     for (StringRequest strReq: finalStrReqs){
395         strReq.setTag("Here Directions API Call");
396         queueTwo.add(strReq);
397     }
398     queueTwo.addRequestFinishedListener(new RequestQueue.RequestFinishedListener<Object>() {
399         @Override
400         public void onRequestFinished(Request<Object> request) {
401             if (gatheredRoutes.size()>2){
402                 popupWindow.getContentView().setVisibility(View.VISIBLE);
403                 popupView.findViewById(R.id.cancel_btn).setOnClickListener(new View.OnClickListener() {
404                     @Override
405                     public void onClick(View view) {
406                         popupWindow.dismiss();
407                     }
408                 });
409                 ListView routeView = popupView.findViewById(R.id.route_list);
410                 ArrayList<String> routeNames = new ArrayList<>();
411                 for(int i=0;i<gatheredRoutes.size();i++){
412                     int dur = (int) Math.round(Integer.parseInt(gatheredRoutes.get(i).duration)/60);
413                     routeNames.add("ROUTE "+(i+1)+"\nDuration: "+dur+" minutes"+" \nLength: "+gatheredRoutes.get(i).length+" m");
414                 }
415                 ArrayAdapter<String> routeAdapter = new ArrayAdapter<>(context, HomePage.this, android.R.layout.simple_list_item_1, routeNames);
416                 routeView.setAdapter(routeAdapter);
417                 calBar.setVisibility(View.GONE);
418             }
419         }
420     });
421 }
```

This code segment is a continuation of the previous segment if the final requests array was not empty. The second queue global variable cancels all requests with the same tag as the previous queue, a for loop goes through the values in the final requests array and setting the tag as well as adding it to the second queue.

A listener is then added to the second queue when it had finished doing all the api calls. It then checks if the global gathered routes (which is an Route array) size is greater than 2, if it is then the popup is displayed adding an on click listener to the cancel button to dismiss the popup.

A ListView variable is created by getting the view from the popup, afterwards it initializes a String array of route names.

It then loops through the gathered routes, parsing the duration string result gathered from the Route object as an int which is then used when a new string is added to the route names array. An array adapter is then created for the popup list to display the string values in route names (setting the adapter to the route view on line 419) and then making the loading bar disappear afterwards.

3.3.7.5 Code Segment 5

```
422 routeView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
423     @Override  
424     public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
425         Route selectedRoute = gatheredRoutes.get(i);  
426         previousRoutes.add(selectedRoute);  
427  
428         PolylineOptions polyOpt = new PolylineOptions().addAll(PolyUtil.decode(selectedRoute.encodedPolyline)).width(5).color(Color.RED).geodesic(true).jointType(JointType.ROUND);  
429         currentPoly = map.addPolyline(polyOpt);  
430         currentLength = Integer.parseInt(selectedRoute.length);  
431         currentDur = Integer.parseInt(selectedRoute.duration);  
432         int duration = (int) Math.round(Integer.parseInt(selectedRoute.duration)/60);  
433         durView.setText(duration + "duration" + "minutes");  
434         durView.setVisibility(View.VISIBLE);  
435         lengthView.setText(currentLength + "selectedRoute.length" + "m");  
436         lengthView.setVisibility(View.VISIBLE);  
437         constraints.clear();  
438         popupWindow.dismiss();  
439     }  
440 })  
441  
442 }  
443  
444 }  
445  
446 }  
447  
448 map.moveCamera(CameraUpdateFactory.newLatLngZoom(originMark.getPosition(), zoom));  
449 }  
450 }  
451 }
```

This code segment is a continuation from the code segment before. A on item click listener is added to the route view where it gets the index from the selected view, uses that index in getting the Route from the gathered routes array which then gets added to the previous routes stack. From lines 428-438 is responsible for displaying the polyline on the map, showing the duration, length and clearing the route names array as well as dismissing the popup. Line 448 then moves the map camera on the origin location of the route selected.

3.3.8 Create Directions API Request:

3.3.8.1 Code Segment 1

```
728 private ArrayList<StringRequest> createStringRequest(double originLat, double originLng, double destLat, double destLng, Boolean routeFlag) {
729     ArrayList<StringRequest> strArr = new ArrayList<>();
730     String prefix = "https://router.hereapi.com/v8/routes?transportMode=car&";
731     String origin = "origin="+originLat+","+originLng+"&";
732     String dest = "destination="+destLat+","+destLng+"&";
733     String avoid = "";
734     String avoidFeatures = "";
735     String expect = "return=polyline,summary&";
736     String key = "apikey="+getString(R.string.here_api_key);
737
738
739     System.out.println("Highway: "+highwaySwitch.isChecked()+" Toll: "+tollSwitch.isChecked());
740     if(highwaySwitch.isChecked()){
741         avoidFeatures = "avoid[features]=controlledAccessHighway";
742     }
743     if(tollSwitch.isChecked()){
744         if(avoidFeatures.equals("")){
745             avoidFeatures = "avoid[features]=tollroad";
746         }
747         else {
748             avoidFeatures += ",tollroad";
749         }
750     }
751     if(avoidFeatures.equals("")){
752         avoidFeatures += "&";
753     }
754
755     if(routeFlag) {
756         ArrayList<Accident> accidents = new ArrayList<>();
757         try {
758             JSONObject jsonObject = new JSONObject(getJSONAsset());
759             JSONArray jsonArray = jsonObject.getJSONArray( "name": "accidents");
760             for(int i=0;i<javascriptArray.length();i++){
761                 JSONObject accident = jsonArray.getJSONObject(i);
762                 double cordLat = accident.getDouble( "name": "cordLat");
763                 double cordLng = accident.getDouble( "name": "cordLng");
764                 int minor = accident.getInt( "name": "minor")*2;
765                 int serious = accident.getInt( "name": "serious")*6;
766                 int fatal = accident.getInt( "name": "fatal")*10;
767                 int weight = minor+serious+fatal;
768                 Accident newAccident = new Accident(cordLat,cordLng,weight);
769                 accidents.add(newAccident);
770             }
771         } catch (JSONException e){
772             e.printStackTrace();
773         }
774
775         Boolean routeCheck = false;
776
777         for(Accident accident: accidents){
778             LatLng pos = new LatLng(accident.cordLat,accident.cordLng);
779             if(PolyUtil.isLocationOnEdge(pos, PolyUtil.decode(gatheredRoutes.get(0).encodedPolyLine), geodesic: true, tolerance: 15)){
780                 routeCheck = true;
781             }
782         }
783     }
```

This is a code segment showcasing the start of the createStringRequest method, it takes in the parameters originLat (origin latitude), originLng (origin longitude), destLat (destination latitude), destLng (destination longitude) and a route flag. From lines 729-736 it is declaring the different parts of the string as well as initializing a new array of requests to be returned.

It then checks if the highway switch is checked and if it is then change the avoidFeatures variable to support avoiding highways when built in the final api url string. If the toll switch is checked it also checks if the avoidFeatures string is empty or not, if it is then it will only support avoiding the toll roads if it's not empty it is just added to the current value of avoidFeatures. It then checks if it's empty and if it's not it adds an "&" symbol at the end to continue to the next parameter in the url.

It then checks if the routeFlag is true, if it is then a new array is created with type Accident and then the from lines 767-773 it gets the JSON file (more info under heading 3.3.4) containing the accident data in the docklands area. It then creates a new Boolean value set to false which will be the routeCheck variable.

The accidents are then looped through with a new LatLng variable created and then checks if the position is close to the edge (line 779), if it is then routeCheck turns into True.

3.3.8.2 Code Segment 2

```

763
764
765     if(routeCheck) {
766         avoid = "avoid[areas]=";
767         for (Accident accident: accidents){
768             if(accident.weight<20){
769                 if(avoid.equals("avoid[areas]=")){
770                     avoid += "bbox:"+(accident.cordLng-0.000188)+","+(accident.cordLat+0.000171)+","+(accident.cordLng+0.000204)+","+(accident.cordLat-0.000194);
771                 } else {
772                     avoid += "|bbox:"+(accident.cordLng-0.000188)+","+(accident.cordLat+0.000171)+","+(accident.cordLng+0.000204)+","+(accident.cordLat-0.000194);
773                 }
774             }
775         }
776         avoid += "&";
777         String mediumUrl = prefix+origin+dest+avoid+avoidFeatures+expect+key;
778         strArr.add(sendRequest(mediumUrl, originLat, originLng, destLat, destLng));
779
780         avoid = "avoid[areas]=";
781         for (Accident accident: accidents){
782             if(avoid.equals("avoid[areas]=")){
783                 avoid += "bbox:"+(accident.cordLng-0.000188)+","+(accident.cordLat+0.000171)+","+(accident.cordLng+0.000204)+","+(accident.cordLat-0.000194);
784             } else {
785                 avoid += "|bbox:"+(accident.cordLng-0.000188)+","+(accident.cordLat+0.000171)+","+(accident.cordLng+0.000204)+","+(accident.cordLat-0.000194);
786             }
787         }
788         avoid += "&";
789         String highUrl = prefix+origin+dest+avoid+avoidFeatures+expect+key;
790         strArr.add(sendRequest(highUrl, originLat, originLng, destLat, destLng));
791     }
792 } else {
793     String url = prefix+origin+dest+avoid+avoidFeatures+expect+key;
794     System.out.println("url: "+url);
795
796     StringRequest req = sendRequest(url, originLat, originLng, destLat, destLng);
797     strArr.add(req);
798 }
799 return strArr;
800 }
801

```

This code segment is a continuation from the previous one, where it then continues to check if routeCheck is true. If it is then it will change the avoid string to support avoid areas in the plotting of the route. It then loops through the accidents and if the accident weight is less than 20, if true it then checks if the avoid variable is the same as default which only adds the first instance of a bbox to the string if true otherwise it would add on an additional bbox, the difference being the one added on has a "|" at the beginning to tell the API to add this one as well to avoid.

The reasons why the values had to be changed when putting in the string is because we are pinpointing the corners for where to box to be drawn. In this case the formula for creating a box would be:

- Top left corner: longitude-0.000188, latitude+0.000171
- Bottom right corner: longitude+0.000204, latitude-0.000194

The values were gathered from pin pointing two small markers on google maps to symbolize the corners and then comparing the differences with a marker in centre of the would-be box.

The method then continues to add an “&” symbol at the end so the url can accept the new parameter in the url. A url is then constructed on line 796 and then added to the requests array. Lines 799-809 is a repeat of lines the only difference being is there it doesn’t take weight into consideration and accepts all accidents to be avoided regardless of weighting.

The else statement on line 811 occurs when the route flag is false and constructs only 1 api with the default parameters with no changes, which then gets added to the array of requests. At the end the array of requests get’s returned.

3.3.9 Previous Route:

```
546 prevRoute.setOnClickListener(new View.OnClickListener() {
547     @Override
548     public void onClick(View view) {
549         if(previousRoutes.isEmpty()){
550             Toast.makeText(context, HomePage.this, "There are no previous routes!", Toast.LENGTH_LONG).show();
551         } else {
552             Route prev = previousRoutes.pop();
553             if(previousRoutes.isEmpty()){
554                 Toast.makeText(context, HomePage.this, "There are no previous routes!", Toast.LENGTH_LONG).show();
555             } else {
556                 locationSwitch.setChecked(false);
557                 locFlag = false;
558                 searchFlag = true;
559                 removePoints();
560                 searchFlag = false;
561
562                 double originLat = Double.parseDouble(prev.originLat);
563                 double originLng = Double.parseDouble(prev.originLng);
564                 LatLng origin = new LatLng(originLat, originLng);
565
566                 double destLat = Double.parseDouble(prev.destLat);
567                 double destLng = Double.parseDouble(prev.destLng);
568                 LatLng dest = new LatLng(destLat, destLng);
569
570                 MarkerOptions originOpt = new MarkerOptions().position(origin).title("Position A");
571                 MarkerOptions destOpt = new MarkerOptions().position(dest).title("Position B");
572
573                 Marker originMark = map.addMarker(originOpt);
574                 markers.add(originMark);
575                 Marker destMark = map.addMarker(destOpt);
576                 markers.add(destMark);
577
578                 List<LatLng> cords= new ArrayList<>();
579                 PolylineOptions polyOpt = new PolylineOptions();
580                 cords = PolyUtil.decode(prev.encodedPolyLine);
581                 polyOpt.addAll(cords).width(5).color(Color.RED).geodesic(true).jointType(JointType.ROUND);
582                 currentPoly = map.addPolyline(polyOpt);
583                 currentDur = Integer.parseInt(prev.duration);
584                 currentLength = Integer.parseInt(prev.length);
585                 int duration = (int) Math.round(Integer.parseInt(prev.duration)/60);
586                 durView.setText("Duration: "+duration+" minutes");
587                 durView.setVisibility(View.VISIBLE);
588                 lengthView.setText("Length: "+prev.length+" m");
589                 lengthView.setVisibility(View.VISIBLE);
590                 map.moveCamera(CameraUpdateFactory.newLatLngZoom(originMark.getPosition(), zoom: 13));
591             }
592         }
593     }
594 }
```

This code block showcases the logic of how previous routes are displayed. It starts of by checking if the previous routes stack is empty, if it is then it would notify the user that there are no more previous routes, it would then continue on and pop the value out of the stack into an Route variable and will check if the stack is empty again afterwards, if it is then it will notify the user that there’s no more previous routes.

If it's false then it will switch the location switch to off and turn the global Boolean var of locFlag to false and the searchFlag to true before calling removePoints()(More information under heading 3.3.10) and after the method is called it then reverts the searchFlag back to false. From lines 562-298 it just creates the markers and polylines on the map which is gathered from the popped route variable before moving the map camera to the origin of the route.

3.3.10 Remove current Route:

```

634 private void removePoints() {
635     if(markers.size()==0&&!locFlag&&searchFlag){
636         Toast.makeText(context, HomePage.this, text: "No points on map!", Toast.LENGTH_LONG).show();
637     } else {
638         durView.setVisibility(View.GONE);
639         lengthView.setVisibility(View.GONE);
640         System.out.println("Size: "+markers.size());
641         ArrayList<Marker> removeMarksList = new ArrayList<>();
642         for (int i=0;i<markers.size();i++){
643             System.out.println("Index: "+i);
644             Marker currentMark = markers.get(i);
645             if(i==0&&isUserLocationChecked){
646                 continue;
647             }
648             currentMark.remove();
649             removeMarksList.add(currentMark);
650         }
651         markers.removeAll(removeMarksList);
652         Boolean checker = currentPoly==null;
653         if(currentPoly!=null){
654             if (locationSwitch.isChecked()&&markers.size()!=0){
655                 currentPoly.remove();
656                 map.clear();
657                 MarkerOptions currMarker = new MarkerOptions().position(markers.get(0).getPosition()).title("Position A");
658                 markers.clear();
659                 Marker newMarker = map.addMarker(currMarker);
660                 markers.add(newMarker);
661             }
662             else {
663                 currentPoly.remove();
664                 map.clear();
665                 markers.clear();
666             }
667         }
668     }
669 }

```

This code block shows the method removePoints() which is called when the remove route button is pressed or when there is some sort of replacement of the current route. It first checks if there are any markers (assuming all flags are false) if it's true it will notify that there is no points on the map. If it's false it would disappear the length and duration text, loop through the array of markers and if user location is checked (the global Boolean value) it will continue at index 0 instead of removing it, it then get's added to an array of makers that needs to be removed which that array then used in line 651 to remove all the markers that had been requested to be removed.

An if statement then checks if the current polyline is not null, if it's false it would end the method there but if it's true then it will then check if the location switch is checked and the markers isn't empty, if it's true the current polyline is removed, the map is cleared and so is the markers array, and the origin marker is placed back in its spot on the map and is re-added to the markers array. If it's false it would remove the current polyline as well as clearing the map and markers.

3.3.11 Save Current Route:

3.3.11.1 Code Segment 1

```
453 saveRoute.setOnClickListener(new View.OnClickListener() {
454     @Override
455     public void onClick(View view) {
456         if(currentPoly==null){
457             Toast.makeText( context HomePage.this, text: "A route must be plotted in order to be saved!", Toast.LENGTH_LONG).show();
458         } else {
459             LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
460             View popupView = inflater.inflate(R.layout.save_route_popup, root null);
461             int width = LinearLayout.LayoutParams.WRAP_CONTENT;
462             int height = LinearLayout.LayoutParams.WRAP_CONTENT;
463             boolean focusable = true;
464
465             PopupWindow popupWindow = new PopupWindow(popupView, width, height, focusable);
466             popupWindow.showAtLocation(view, Gravity.CENTER, 0, 0);
467
468             popupView.findViewById(R.id.cancel_btn).setOnClickListener(new View.OnClickListener() {
469                 @Override
470                 public void onClick(View view) { popupWindow.dismiss(); }
471             });
472
473             popupView.findViewById(R.id.submit_btn).setOnClickListener(new View.OnClickListener() {
474                 @Override
475                 public void onClick(View view) {
476                     TextView routeName = popupView.findViewById(R.id.route_name_input);
477                     if(routeName.getText().toString().isEmpty()){
478                         Toast.makeText( context HomePage.this, text: "A name for the route must be assigned!", Toast.LENGTH_LONG).show();
479                     } else {
480                         ProgressBar progBar = popupView.findViewById(R.id.prog_bar);
481                         String route_name = routeName.getText().toString();
482                         String originLat = ""+markers.get(0).getPosition().latitude;
483                         String originLng = ""+markers.get(0).getPosition().longitude;
484                         String destLat = ""+markers.get(1).getPosition().latitude;
485                         String destLng = ""+markers.get(1).getPosition().longitude;
486                         String encodedPolyline = PolyUtil.encode(currentPoly.getPoints());
487                         String dur = ""+currentDur;
488                         String length = ""+currentLength;
489                         String avoidHighways = ""+highwaySwitch.isChecked();
490                         String avoidTolls = ""+tollSwitch.isChecked();
491
492                         progBar.setVisibility(View.VISIBLE);
493
494                         DatabaseReference ref = FirebaseDatabase.getInstance().getReference("Users")
495                             .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
496                             .child("Routes");
497
498                         Task<DataSnapshot> task= ref.get();
499
500
```

The code segment above is the beginning of the on click listener for the save route button. It first checks if the current polyline is null, if it is then the user is notified that a route must be plotted. If it isn't a popup get's inflated (initialized) for use where the cancel button has an on click listener to dismiss the window when pressed. The popup's button for saving the route has a on click listener as well where it first checks if the popup's input is empty, if it is then it notifies that a name must be assigned to the saved route as it will be used as a primary key in the database. If it's false it displays a loading bar and it gathers all the necessary values from the markers, current polyline, current duration, current length, and highway/toll switch status. The database is then referenced going into the Users() table, accessing the user's object and the list of routes in that object.

3.3.11.2 Code Segment 2

```
501 task.addOnSuccessListener(new OnSuccessListener<DataSnapshot>() {
502     @Override
503     public void onSuccess(DataSnapshot dataSnapshot) {
504         Boolean nameFlag = false;
505         DataSnapshot ss = dataSnapshot;
506         for (DataSnapshot child : ss.getChildren()) {
507             if (child.child("routeName").getValue().equals(route_name)) {
508                 System.out.println("Value: " + child.getValue());
509                 Toast.makeText(getApplicationContext(), "Error: Route name already exists, please make a new one.", Toast.LENGTH_LONG).show();
510                 nameFlag = true;
511             }
512         }
513
514         if (!nameFlag) {
515             Route newRoute = new Route(route_name, originLat, originLong, destLat, destLong, encodedPolyline, avoidHighways, avoidTolls, dur, length);
516             FirebaseDatabase.getInstance().getReference("Users")
517                 .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
518                 .child("Routes").child(ref.push().getKey()).setValue(newRoute).addOnCompleteListener(new OnCompleteListener<Void>() {
519                 @Override
520                 public void onComplete(@NonNull Task<Void> task) {
521                     if (task.isSuccessful()) {
522                         Toast.makeText(getApplicationContext(), "Route Saved!", Toast.LENGTH_LONG).show();
523                     } else {
524                         Toast.makeText(getApplicationContext(), "Error: Route cannot be saved!", Toast.LENGTH_LONG).show();
525                     }
526                 }
527             });
528         }
529     }
530 }).addOnFailureListener(new OnFailureListener() {
531     @Override
532     public void onFailure(@NonNull Exception e) {
533         Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
534     }
535 });
536
537 progressBar.setVisibility(View.GONE);
538 popupWindow.dismiss();
539
540 }
541
542 }
543
544 };
```

This code segment is a continuation from the code segment before where an on-success listener is added to the reference task. If it is not successful it displays an error message on the app. If it is successful, it will then add a Boolean variable “nameFlag” to false and will gather the data snapshot adding it to a variable. It then loops through the children in the data snapshot and if it finds that the name already exists on the database it will change the nameFlag to true and will display a message saying the name already exists.

An if statement then checks if the nameFlag is not true, if that if statement passes then it creates a new route object with the parameters created earlier and it then adds that to the database. At the end of the method the loading bar disappears, and the popup is dismissed.

3.3.12 Search Saved Routes:

3.3.12.1 Code segment 1

```
182 routeSearch.setOnClickListener(new View.OnClickListener() {
183     @Override
184     public void onClick(View view) {
185         DBRoutes.clear();
186         DBRoutesName.clear();
187         routeSearch.setIconified(false);
188         DatabaseReference ref = FirebaseDatabase.getInstance().getReference().child("Users").child(FirebaseAuth.getInstance().getCurrentUser().getUid()).child("Routes");
189         ref.addValueEventListener(new ValueEventListener() {
190             @Override
191             public void onDataChange(@NonNull DataSnapshot snapshot) {
192                 for(DataSnapshot postSnap: snapshot.getChildren()){
193                     Route collectedRoute = postSnap.getValue(Route.class);
194                     DBRoutes.add(collectedRoute);
195                     DBRoutesName.add(collectedRoute.routeName);
196                 }
197             }
198             @Override
199             public void onCancelled(@NonNull DatabaseError error) {
200                 Toast.makeText(context, HomePage.this, error.getMessage(), Toast.LENGTH_LONG).show();
201             }
202         });
203         adapter = new ArrayAdapter<String>(context, HomePage.this, android.R.layout.simple_list_item_1, DBRoutesName);
204         routesView.setAdapter(adapter);
205     }
206 }
```

This code segment is the beginning of the route search bar on click listener. It starts off by clearing all from the database routes global array and the database route names global array, it sets the iconified of the search bar to false which will allow the enable text regardless of where the user presses on the bar instead of having to be forced pressing an icon.

A database reference is then created to access the user's routes and an value event listener is added to that reference. It then goes through the data and adds the route gathered from the database to the database routes arrays and it gets its name to be added to the database route names array. If it's cancelled for any reason an error message would appear. An adapter is then created using the array of route names and is then assigned to the listView global variable of routesView.

3.3.12.2 Code Segment 2

```
206 routesView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
207     @Override
208     public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
209         String route = (String) routesView.getItemAtPosition(i);
210         Route currentRoute = null;
211         for (int j=0;j<DBRoutes.size();j++){
212             if(DBRoutes.get(j).routeName.equals(route)){
213                 currentRoute = DBRoutes.get(j);
214             }
215         }
216         if (currentRoute == null){
217             Toast.makeText(context, HomePage.this, "Error in gathering saved route.", Toast.LENGTH_LONG).show();
218         } else {
219             if(searchFlag){
220                 removePoints();
221                 searchFlag = false;
222             }
223         }
224     }
225 }
```

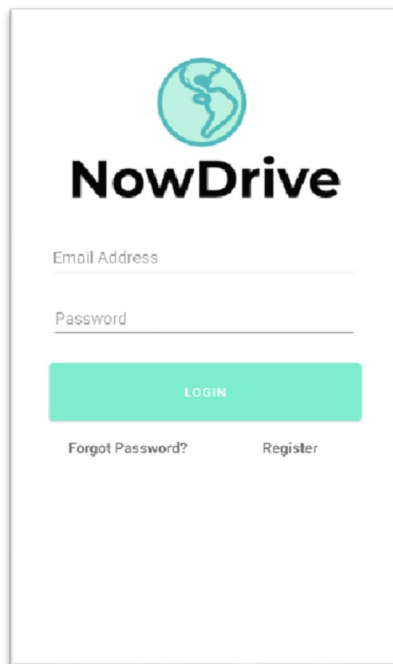
This code segment is a continuation of the segment from before. An on item click listener is then assigned to the global variable and when it is clicked a route is gathered from the index of the selected and a new route variable is initialized. Database routes is then looped through and when the names match it get assigned to the currentRoute variable. If the current route is null then a message display that an error has occurred getting the route, otherwise it will continue on to check if the searchFlag is true which calls the removePoints() method (check heading 3.3.10) and change the global search flag variable to false.

3.3.12.3 Code Segment 3

```
222     } else {
223         searchFlag = true;
224         removePoints();
225         searchFlag = false;
226         PolylineOptions polyOpt = new PolylineOptions();
227         List<LatLng> cords= new ArrayList<>();
228
229         Boolean avoidHighways = Boolean.parseBoolean(currentRoute.avoidHighways);
230         Boolean avoidTolls = Boolean.parseBoolean(currentRoute.avoidTolls);
231
232         locationSwitch.setChecked(false);
233         highwaySwitch.setChecked(avoidHighways);
234         tollSwitch.setChecked(avoidTolls);
235
236         double originLat = Double.parseDouble(currentRoute.originLat);
237         double originLng = Double.parseDouble(currentRoute.originLng);
238         LatLng origin = new LatLng(originLat,originLng);
239
240         double destLat = Double.parseDouble(currentRoute.destLat);
241         double destLng = Double.parseDouble(currentRoute.destLng);
242         LatLng dest = new LatLng(destLat,destLng);
243
244         MarkerOptions originOpt = new MarkerOptions().position(origin).title("Position A");
245         MarkerOptions destOpt = new MarkerOptions().position(dest).title("Position B");
246
247         Marker originMark = map.addMarker(originOpt);
248         markers.add(originMark);
249         Marker destMark = map.addMarker(destOpt);
250         markers.add(destMark);
251
252         cords = PolyUtil.decode(currentRoute.encodedPolyLine);
253         polyOpt.addAll(cords).width(5).color(Color.RED).geodesic(true).jointType(JointType.ROUND);
254         currentPoly = map.addPolyline(polyOpt);
255         currentDur = Integer.parseInt(currentRoute.duration);
256         currentLength = Integer.parseInt(currentRoute.length);
257         int duration = (int) Math.round(Integer.parseInt(currentRoute.duration)/60);
258         durView.setText("Duration: "+duration+" minutes");
259         durView.setVisibility(View.VISIBLE);
260         lengthView.setText("Length: "+currentRoute.length+" m");
261         lengthView.setVisibility(View.VISIBLE);
262         routeSearch.clearFocus();
263         map.moveCamera(CameraUpdateFactory.newLatLngZoom(originMark.getPosition(), 13));
264     }
265 }
266 }
267 }
268 }
269 }
270 };
```

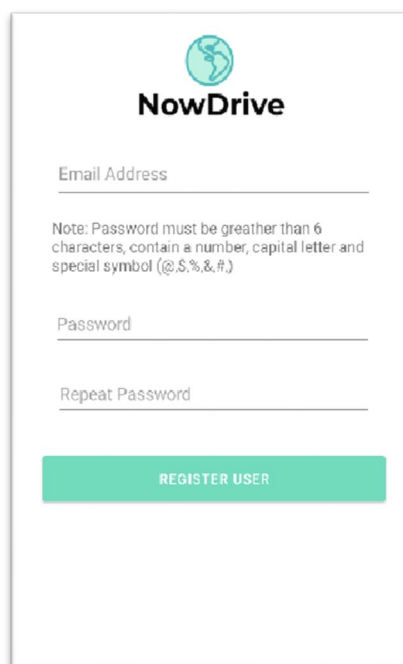
This code segment is a continuation of the code segment before. If the search flag is not true it is then changed to true and the removePoints() method is called. After the method is called, it will change it back to false. From lines 226-263, it gathers all the values needed from the currentRoute variable that is then used to add markers, polyline, duration, and length views for the user. At the end, the focus on the bar is cleared and the map camera is then moved to the point of origin of the route.

3.4 Graphical User Interface (GUI)



The login screen features the NowDrive logo at the top, which consists of a teal globe icon and the text "NowDrive". Below the logo are two input fields: "Email Address" and "Password". A teal "LOGIN" button is positioned below the password field. At the bottom, there are two links: "Forgot Password?" and "Register".

Figure 2.3.2 Now Drive Login



The registration screen features the NowDrive logo at the top. Below the logo is an "Email Address" input field. A note specifies: "Note: Password must be greather than 6 characters, contain a number, capital letter and special symbol (@, \$, %, &, #)". Below this are "Password" and "Repeat Password" input fields. A teal "REGISTER USER" button is at the bottom.

Figure 2.3 NowDrive Registration

3.4.1 NowDrive Login (Figure 2.2)

This is the current GUI implementation for the login feature of NowDrive. It has an email and password field that the user can input their details into which will get checked by the application using java and then checked with the firebase API before allowing to user to move on to the home page (figure 2.5). If the User taps on *Register* it would redirect them to the registration page (Figure 2.3) and if they tap on *Forgot Password?* It would redirect them to the forgot password page (Figure 2.4).

3.4.2 NowDrive Registration (Figure 2.3)

This is the current GUI implementation for the registration feature of NowDrive. It has an Email field and two password fields which all need to be filled to make an account with the application. The data is then verified using Java locally before sending the data to the Firebase API, upon success it would redirect the user to login page (Figure 2.2) and would send an email to the user's provided email address for verification.

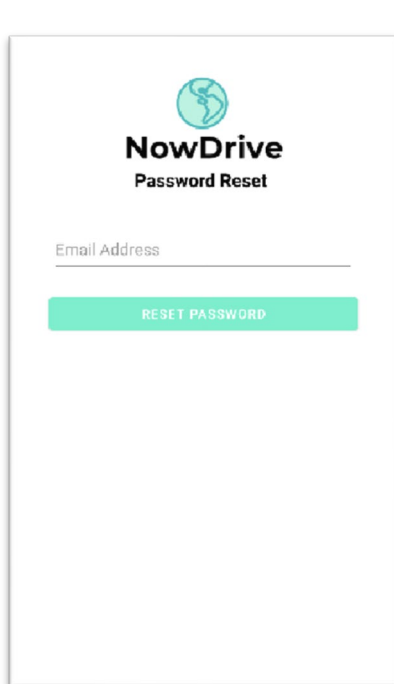


Figure 2.4 NowDrive Password Reset

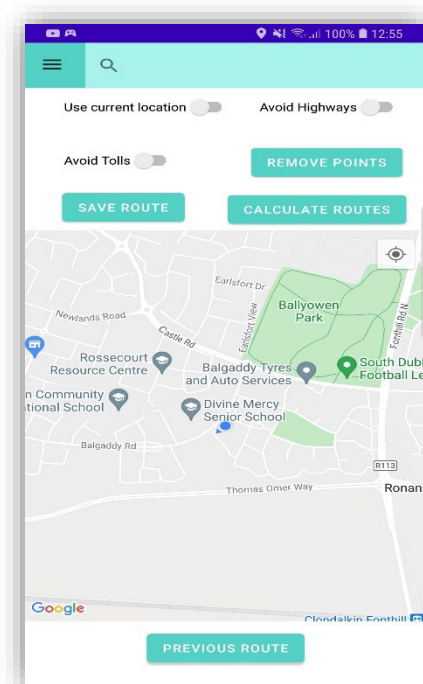


Figure 2.5 NowDrive Home Page No Route

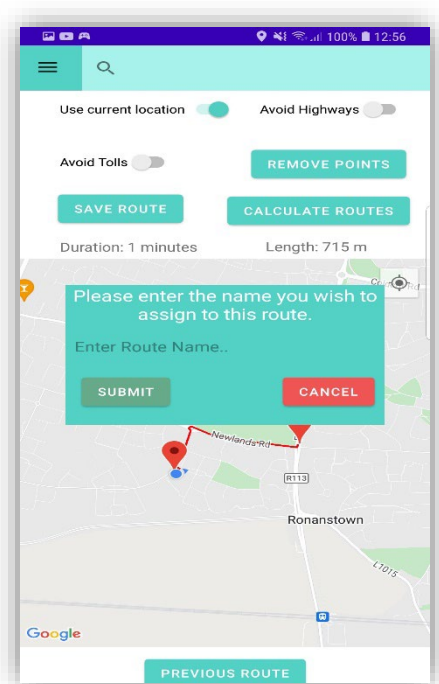


Figure 2.5.1 Home Page save route

3.4.3 NowDrive Password Reset (Figure 2.4)

This is the current GUI implementation for the Password Reset feature of NowDrive, it has an email field where a password reset email would be sent to the User and upon success it would redirect the user to the login page (Figure 2.2).

3.4.4 NowDrive Home Page (Figures 2.5, 2.5.1, 2.5.2)

This is the current implementation for the homepage feature of NowDrive. As you can see in figure 2.5, there is many factors at play. The top left symbol with the three bars when pressed will open up the NowDrive Home Menu (Figure 2.6). The search bar represents the saved routes that can be search by the user, if they user were to start typing in the bar it would display a filtered list of search results of saved route names made by the user. The use current location switch is for using the current location of the user as the origin in the calculation of the route where location access has been provided to the app. Avoid Highways and Avoid Tolls switches ensures when switched on will avoid any routing through highways or roads that contain tolls. The previous route button displays the route previously calculated for the user. The map display shows the map of the current location they are viewing with the geo symbol type right re-directing to the user's location when enabled.

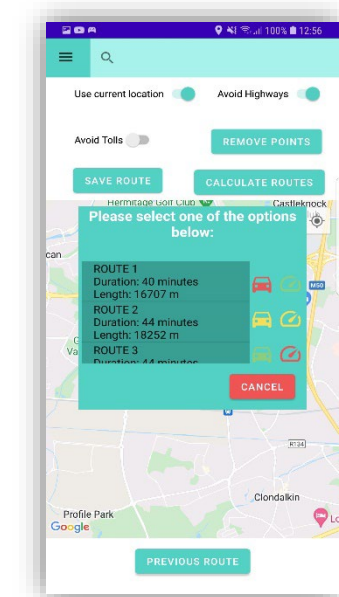


Figure 2.5.2 NowDrive Multiple Routes

To plot a route the user will either enable current location as origin or hold on the map where they want to place their origin. Then they hold on a location for it's destination. The user then must press the calculate routes button where then a route would be created.

Figure 2.5.1 is showcasing what UI appears when a route is calculated and the save button is pressed. We can see a duration and length appears for the route that had been calculated. A popup can be seen asking for a route name where you can cancel your save, or you can type in a route name and submit it to the database to be saved.

Figure 2.5.2 shows us the UI what occurs when multiple routes have been detected in the calculation, this occurs when a route happens to end, go through, or start in docklands IFSC. It provides 3 options that are colour coded with the following:

- Red Car: Symbolizes dangerous route
- Yellow Car: Symbolizes moderate danger route
- Green Car: Symbolizes safe route
- Red Speed Dial: Symbolizes slow route
- Yellow Speed Dial: Symbolizes moderate speed route
- Green Speed Dial: Symbolizes fastest route

Within the options contains a route number (1,2 or 3), the duration of a route and a length of the route. This gives an indication of what the route is like and would give options for the user to chose from, informing their decision. The box that these options are contained in are scrollable which supports the amount of information being given about these routes. The cancel button means you can cancel the routing of this route if you do not wish to calculate a particular route anymore.

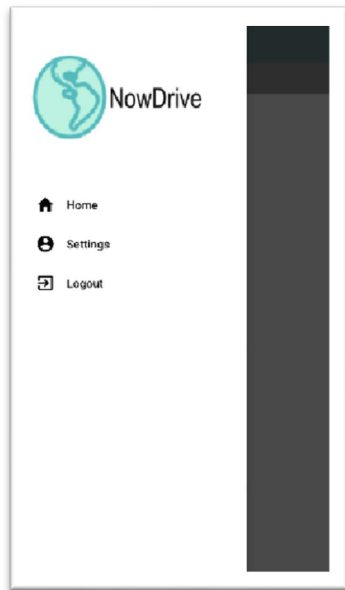


Figure 2.6 NowDrive Home Menu

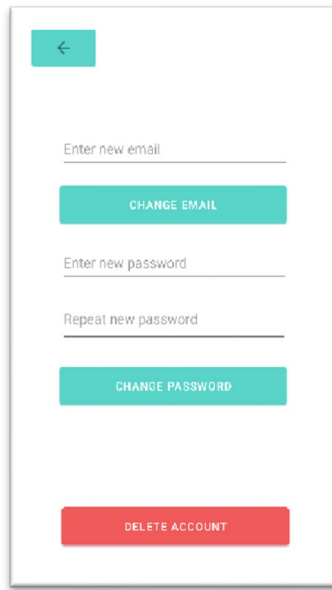


Figure 2.7 NowDrive Settings

3.4.5 [NowDrive Home Menu \(Figure 2.6\)](#)

This is the current GUI Implementation for the NowDrive Home Menu, which contains three buttons, the home button returns you back to the home page (Figure 2.5), the Settings button would redirect the user to the Settings page (figure 2.7) and the Logout button would log the User out of the mobile application and redirect them to the login page (Figure 2.2).

3.4.6 [NowDrive Settings \(Figure 2.7\)](#)

This is the current implementation for the NowDrive Settings page which contains 3 fields visible which handles the new data being inputted which gets verified by the System locally but also has an additional 2 fields that appears for verification (Currently not visible) to verify the user changes to their account which then communicates to the Firebase API. On success it would redirect the user to the Login Page (Figure 2.2).

3.5 [Testing](#)

3.5.1 [Unit Testing](#)

Two unit tests were created specifically for NowDrive. The purpose of these unit tests is to test the input validation logic for Login and Registration. The reason why I opted for two simple unit tests is that the main priority/focus was on manual testing and debugging of the application in particular the route creation, map display and multiple route options (more details under heading 3.5.2) regardless two unit tests were created for the input validation of Login and Registration to verify that the logic was handling the input correctly as intended.

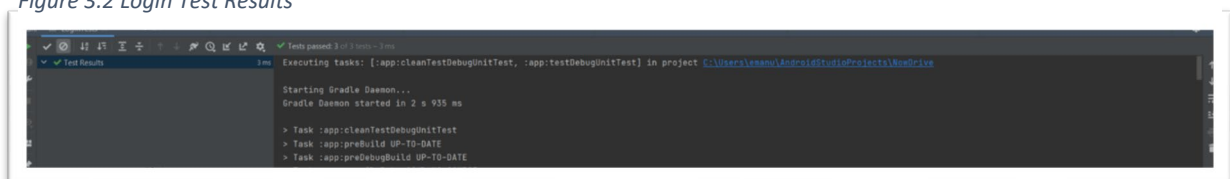
3.5.1.1 Login Tests

```
1 package com.example.nowdrive;
2
3 import ...
4
5 public class LoginTests {
6     String inputEmail, inputPass;
7
8     @Test
9     public void userLogin_isCorrect() {
10         inputEmail = "testEmail@gmail.com";
11         inputPass = "1234A@";
12
13         if(inputEmail.isEmpty()){
14             fail();
15         }
16
17         else if(inputPass.isEmpty()){
18             fail();
19         }
20
21         else if(inputPass.length()<6){
22             fail();
23         }
24
25         else {
26             assertTrue( condition: true);
27         }
28     }
29 }
```

Figure 3.1 Login Test code

As you can see from figure 3.1 this is the code responsible for performing the unit testing for the login input validation. What it simply does is while having a similar structure to the logic handling login validation it performs the task of checking pre-defined values and making sure the expected result occurs. For example in the figure we can see the test method of `userLogin_isCorrect()` where it is checking that when the correct values are inputted that it does not get caught by the if or else if statements and if it does it would throw a fail (where method `fail()` fails the test and `assertTrue(true)` passes it), this is ensuring that the logic in place (currently) is working as intended. There are also two other test methods which is `userLogin_passFail()` which tests the logic handling for incorrect password input and `userLogin_emailFail()` which tests the logic for handling incorrect email input. All these tests can be found in the file `LoginTests.java` under `app/src/test/java/com/example/nowdrive`.

Figure 3.2 Login Test Results



When you run the test you get an output as seen in figure 3.2 above, it tells us that all three tests have passed successfully but if it were to fail it would throw an assertion fail as the test manually fails if a specific condition is set. While the logic is very simplistic currently it does allow room for expanding complexity with regards to mocking (more details under heading 5 Further Development or Research).

3.5.1.2 Registration Tests

A screenshot of a code editor showing the Java code for the RegistrationTests class. The code is as follows:

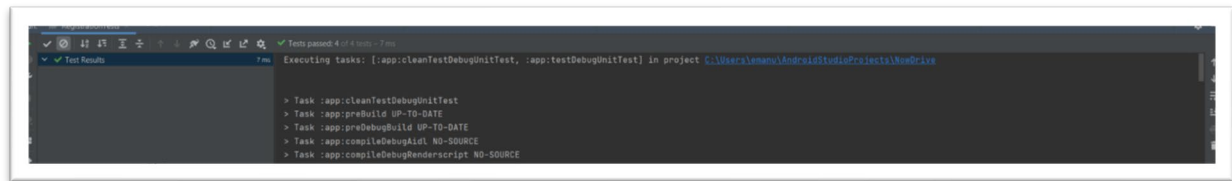
```
1 package com.example.nowdrive;
2
3 import ...
4
5
6
7
8
9 public class RegistrationTests {
10     String inputEmail, inputPass, inputPassTwo;
11
12     @Test
13     public void userReg_isCorrect() {
14         inputPass="1234A@";
15         inputPassTwo="1234A@";
16         inputEmail="testEmail@gmail.com";
17
18         if(inputEmail.isEmpty()){
19             fail();
20         }
21
22         else if(inputPass.isEmpty()){
23             fail();
24         }
25
26         else if(inputPass.length()<4){
27             fail();
28         }
29
30         else if(!inputPass.equals(inputPassTwo)){
31             fail();
32         }
33
34         else {
35             assertTrue( condition: true);
36         }
37     }
38 }
```

Figure 3.3 Registration Test Code

Above in figure 3.3 is the registration test code which is responsible for testing input validation for registration and follows the same logic in place (currently) for such logic deployed on the application. Similar to the login test we can see that there is a method of `userReg_isCorrect()` where it is checking how the input handled when correct and would throw a fail if the inputs were caught in any of the if/else if statements (where the method `fail()` fails the test and `assertTrue(true)` passes the test). There are also three other tests in place for registration, `userReg_passFail()` ensures that the password is picked up by the relevant if/else if statement when the input for password is incorrect, `userReg_emailFail()` ensures that the email is handled correctly and `userReg_passNoMatch()` which checks to make sure that when the inputted password and input repeated password don't match that it get's flagged by the correct if/else if statement. These tests can be found in the file `RegistrationTests.java` under `app/src/test/java/com/example/nowdrive`.

As you can see from figure 3.3 below, it is the results when you run the registration test and all 4 tests pass, if one of them were to fail it would throw an assertion fail into the appropriate test as the fails are manually being added if a specific condition is set. Although this test might seem simplistic there is a solid plan for expansion which is outlined more under the heading 5 Further Development or Research.

Figure 3.3 Registration Test Results



3.5.2 Manual Testing/Debugging

I have conducted many debugging tasks in NowDrive as well as perform specific test cases manually to spot such bugs, some were spotted automatically but most of the bugs were detected manually through performing test cases. Below outlines the test cases that I had performed on NowDrive as well as the bugs found and their solution underneath the outlined test cases.

3.5.2.1 Test Cases

Login Test Case

ID	01
Description	A test to make sure that the login feature is functional.
Assumptions and Pre-Conditions	An account was created, and email was verified and user is on Samsung 8 device.
Test Data	Email: emanuelivan45@gmail.com Password: Testing1232@B
Steps	<ol style="list-style-type: none"> 1. Open up the NowDrive App 2. Input Email and Password 3. Tap Login Button
Expected Result	User is redirected to home page and login was successful.

Forgot Password Test Case

ID	02
Description	A test to make sure that the forgot password feature works correctly.
Assumptions and Pre-Conditions	An account was created and email is verified and user is on Samsung 8 device.
Test Data	Email: emanuelivan45@gmail.com
Steps	<ol style="list-style-type: none"> 1. Open up NowDrive App 2. Tap "Forgot Password?" text

	3. Input email and tap Reset Password Button
Expected Result	An email is sent out to the user that would allow them to reset their password.

Registration Test Case

ID	03
Description	A test to make sure that registration works as intended.
Assumptions and Pre-Conditions	A completely new user with no account and is on Samsung 8 device.
Test Data	Email: emanuelivan45@gmail.com Password: Testing1232@B Repeat Password: Testing1232@B
Steps	<ol style="list-style-type: none"> 1. Open NowDrive app 2. Tap "Register" text 3. Input Email, Password and repeat password. 4. Go to email and verify account
Expected Result	A new account is created and the user is able to log into the application now.

Change Email Test Case

ID	04
Description	A tests to make sure the user is able to change their email for their account.
Assumptions and Pre-Conditions	User has a valid new email they wish to change to, has a current account with verified email, is logged in and on home page and is on Samsung 8 device.
Test Data	Email: emanuelivan45@gmail.com
Steps	<ol style="list-style-type: none"> 1. Tap the menu button top left 2. Tap Settings 3. Input Email 4. Tap Change Email
Expected Result	An email is sent to the new email to verify it and upon verification email is changed successfully and is able to log into their account with it.

Change Password Test Case

ID	05
Description	A test to see if the user can change their password for their account successfully.
Assumptions and Pre-Conditions	The user has an account with a verified email, is logged in, has a new desired password and on Home Page and is on Samsung 8 device.
Test Data	Password: Testing1232@B Repeat Password: Testing1232@B
Steps	<ol style="list-style-type: none">1. Tap the top left menu button2. Tap Settings3. Input Password and Repeat Password4. Tap Change Password Button
Expected Result	The password is changed successfully, and they can log in using the new password from now on.

Delete Account Test Case

ID	06
Description	A test to see if the user can successfully delete their account.
Assumptions and Pre-Conditions	The user has an account with a verified email, is logged in, is on home page and is on a Samsung 8 device.
Test Data	N/A
Steps	<ol style="list-style-type: none">1. Tap top left menu button2. Tap Settings3. Tap delete account button
Expected Result	Account is deleted and user is redirected to the login page.

Plot Simple Route Test Case

ID	07
Description	A test to see if the user can successfully plot a simple route on the map.
Assumptions and Pre-Conditions	A user is logged in with their account that has a verified email, is on home page and is on a Samsung 8 device.
Test Data	N/A
Steps	<ol style="list-style-type: none">1. Tap and hold on your desired start on map to create point A

	<ol style="list-style-type: none"> 2. Tap and hold your desired destination nearby to create point B 3. Tap Calculate Routes button
Expected Result	A red line should be plotted between the two points showing the route on the map.

Plot Route Using Location Test Case

ID	08
Description	A test to see if location services work in context of creating a route.
Assumptions and Pre-Conditions	The user must have an account with a verified email and is logged in, on the home page and is on Samsung 8 device.
Test Data	N/A
Steps	<ol style="list-style-type: none"> 1. Tap Use Current Location Switch 2. Tap and hold to create your desired destination "Point B" 3. Tap Calculate Routes
Expected Result	A red line should be plotted between the user's current location and the destination showing the route on the map.

Plot Route avoiding Tolls and Highway Test Case

ID	09
Description	A test to see if Tolls and Highways are avoided in the route when checked.
Assumptions and Pre-Conditions	The user has an account with a verified email, is logged in, is on home page and is on a Samsung 8 device.
Test Data	N/A
Steps	<ol style="list-style-type: none"> 1. Tap Avoid Highways Switch 2. Tap Avoid Tolls Switch 3. Tap and hold a position on the map to create your start point (Point A) on the map. 4. Tap and hold a position on the map to create your destination (Point B) on the map. 5. Tap Calculate Routes Button
Expected Result	A red line should be plotted between Point A and B avoiding Highways and Tolls showing the route on the map.

Multiple options for route Test Case

ID	10
Description	A test to make sure when a factor is considered (Number of accidents) in the calculation of routes in Docklands.
Assumptions and Pre-Conditions	A user has a verified account, in on Samsung 8 Device, is on Home Page and destination is in Docklands IFSC area.
Test Data	N/A
Steps	<ol style="list-style-type: none">1. Tap and hold a position on the map to create your start point (Point A) on the map.2. Tap and hold a position on the map to create your destination (Point B) on the map (Must be in Docklands IFSC Area).3. Tap Calculate Routes Button4. Select one of the three options that appear on the popup for your desired level of speed and safety.
Expected Result	A red line is plotted between the two points taking into consideration in avoiding certain areas depending on user selection showcasing a route on the map.

Save and search for route Test Case

ID	11
Description	To test that the user can save a route and search for it and select it again for re-use.
Assumptions and Pre-Conditions	The user has a verified account, is on Home Page, a Route is calculated and is on a Samsung 8 Device.
Test Data	Route Name: Test Route
Steps	<ol style="list-style-type: none">1. Tap Save Route Button2. Input Route Name3. Tap Submit Button4. Type Route Name in Search Bar5. Tap the route when it appears
Expected Result	The route appears on the map exactly as it was from the point it was saved.

Previous Route Test Case

ID	12
Description	A test to make sure that the previous route that was plotted is displayed.
Assumptions and Pre-Conditions	More than 1 route was plotted previously, user has a verified account, is on home page and is on Samsung 8 device.
Test Data	N/A
Steps	1. Tap Previous Route Button
Expected Result	The route previously should be displayed on the map.

3.5.2.2 Bugs

Bug ID	01
Bug Description	Google maps API key appearing on GitHub can be security vulnerability.
Solution	This was fixed by editing the Gi ignore file pushing it and regenerating the key to a different one.

Bug ID	02
Bug Description	Google maps location not running
Solution	Need to add permissions to manifest and add code to check for permissions and ask if it is not enabled.

Bug ID	03
Bug Description	Google maps routing not following roads
Solution	Removed old code that was adding the points manually from the API into a polyline, replaced it with extracting the overview polyline from the code and then used PolyUtils to decode the string into a List of LatLng objects to be used in the creation of the polyline.

Bug ID	04
Bug Description	Number of points/markers exceeding 2
Solution	Logic issue in the code changed markers.size() <= 2 to markers.size() < 2

Bug ID	05
Bug Description	Remove Points only removing one at a time
Solution	RemovePoints() Method changed because when removing inside of the loop it causes the size of the array to be reduced, breaking the loop.

Bug ID	06
Bug Description	Location switch crashes app when switched.
Solution	The Boolean value userLocation needed to be instantiated before being used, thus I made the default false in the onCreate method on the homePage.

Bug ID	07
Bug Description	Search bar only working when icon is clicked
Solution	Created an onClick method for the searchView and made it setIconified to false when clicked.

Bug ID	08
Bug Description	Marker not being created when location is off
Solution	Map intialized inside of OnMapReady method instead of the onCreate method.

Bug ID	09
Bug Description	Map not avoiding tolls/highways when toggled
Solution	The API wasn't adding the "avoid=" correctly as it was missing from the code, I then corrected it to include it.

Bug ID	10
Bug Description	Not all data is being saved for routes
Solution	Firebase Realtime database only accepts string values for it's database, I had to convert the object to string data types only.

Bug ID	11
Bug Description	Not all of the saved routes appearing on ListView
Solution	Routes View (A list View) was inside a scrollView to fix this I had to move the Routes View outside of the scroll view into an independent constraint.

Bug ID	12
Bug Description	Indexing incorrect for list selection of route
Solution	Resolved by making the route name as a primary key and adding in logic to make sure no duplicates exist.

Bug ID	13
Bug Description	Previous Route button cuasing crashing
Solution	Had to update the polyline to the new decoder that was getting it's polyline from the new API (HERE API)

Bug ID	14
Bug Description	Map not rendering correctly
Solution	In order for the map to render correctly you will need to google maps API key regardless if you utilize any API calls because map view is reading that key regardless.

Bug ID	15
Bug Description	Popup not dismissing after the 2nd time appearing when item is clicked
Solution	New instances of the popup were being called in the api loop, I had to move it into the gatheredRoutes.size>2 if statement and make sure only one instance is created at the beginning when the calculate button is first pressed.

3.6 Evaluation

3.6.1 Routing

The routing of routes was evaluated for Now Drive, to perform this analysis two routes had been plotted on Now Drive which is a simple and small route and a longer route that takes into consideration number of accidents in the IFSC area and which option the user chose in relation to which type of route they wish to take, for the purposes of evaluation we the safest route possible determined by Now Drive had been chosen. We will be comparing to Google Maps as it is the most widely used maps service that provides routing features.

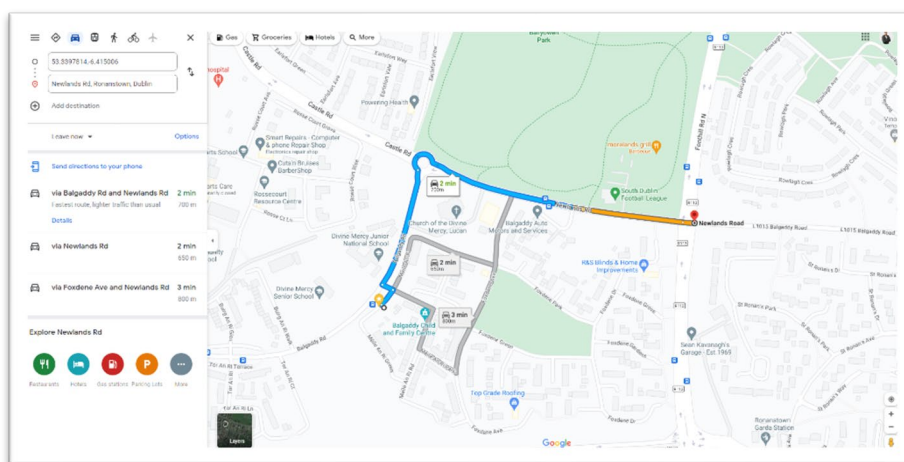


Figure 4.1 Google Maps Simple Route

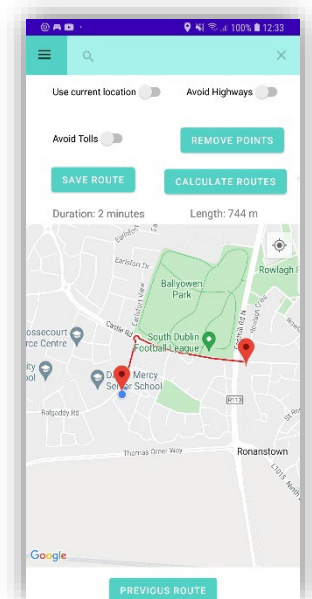


Figure 4.2 NowDrive Simple Route

Regarding the simple route when performing the evaluation for NowDrive we only take accuracy into consideration as google maps and NowDrive should be the same in this regard as no additional factors are being considered in the calculation of the routing in NowDrive as it is located outside the IFSC area. Preferably we would want the results between NowDrive and Google Maps to be as similar as possible.

As you can see from figures 4.1 and 4.2 the results are similar, they both follow the exact same route, provide the same duration of the journey (in this case 2 minutes) and have a similar length with a difference of 44 meters between the both of them. The reason why this difference exists is when inputting the destination coordinates into google maps that matched the same one being used in NowDrive it resulted in Google Maps Routing it closer to the intersection and focusing on the road instead on the exact pin point marker. The very minor difference can also visually be seen on the two figures and would match the 44-meter difference. This is the result we wanted from this evaluation between Google Maps and NowDrive.

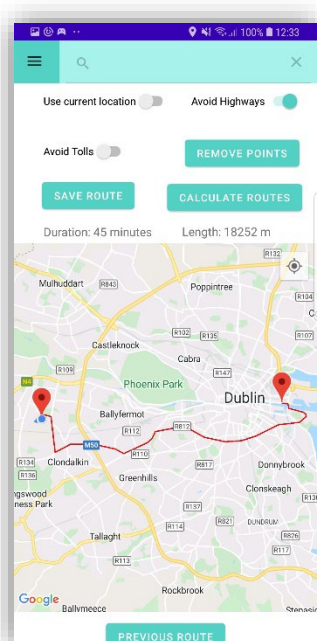


Figure 4.3 NowDrive IFSC Route

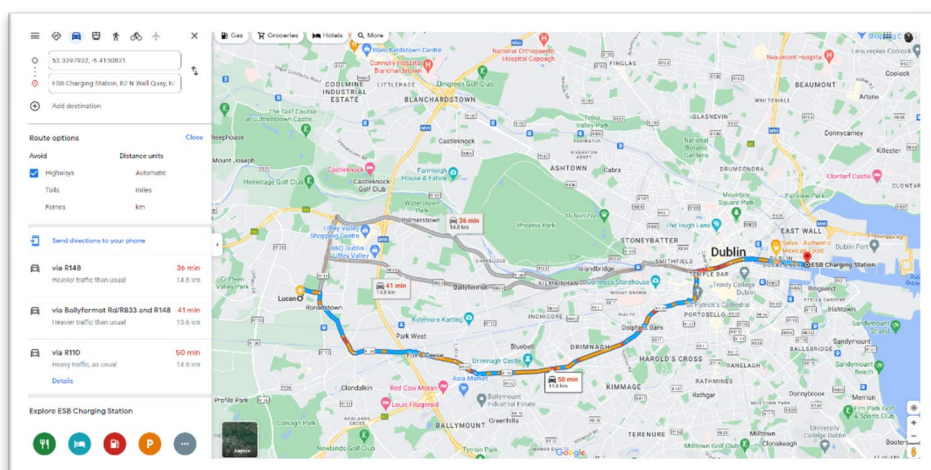


Figure 4.3.4 Google Maps IFSC Route

With regards to the IFSC route what should be expected is that the NowDrive Route would be longer and would have a longer duration because it takes the number of accidents that have occurred in the IFSC area into consideration whereas google maps does not take that factor into consideration. Something that should be noted is that the IFSC route had been plotted in avoiding the highway in both NowDrive and Google maps to view the route leading up to the IFSC area. As you can see from figures 4.3 and 4.4 there are many differences between the two.

The first key difference to notice is the routes are different from each other which was to be expected, google maps has the route go directly into the centre of Dublin and continue onto the IFSC area whereas NowDrive avoids the centre as much as possible since we picked the safest route option and has detected that this route has the least amount of

accidents according to historical data meaning it prioritized this route as the safest compared to others. The second difference is because of the different routing it has resulted in different lengths and durations of the routes where Google Maps has the shorter length and duration and NowDrive has the longer length and duration. This was expected as well because of the difference in routing although they both have the same origin, destination and avoid the highway it has resulted in vastly different routes, durations, and lengths. This is the result we wanted from this evaluation as we wanted NowDrive to differentiate itself from other routing services such as Google maps.

3.6.2 Scalability Costs

Another factor that had been considered in the evaluation of NowDrive is it's scalability costs with regards to it's use of database and APIs. Starting off with the database, the current database implementation for NowDrive is the Firebase Realtime Database under the Spark plan. Under the spark plan we can have 1GB stored in the Realtime database, up to 100 simultaneous connections, have a maximum download of 10GB a month and the database only being limited to 1 project (Google, n.d.). With regards to authentication, it is limited to 10,000 verifications a month under the spark plan (Google, n.d.).

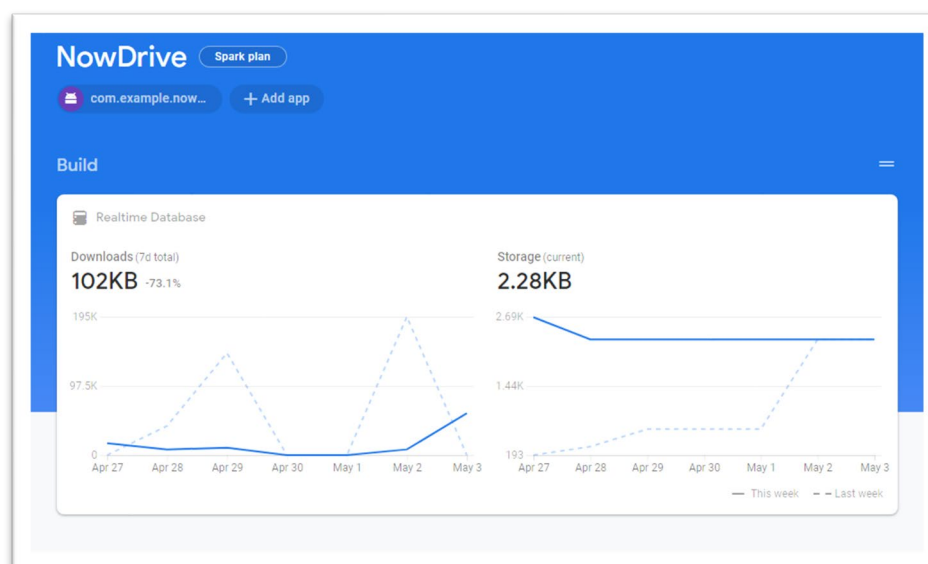


Figure 4.3.5 Firebase Overview

As you can see from the figure above (figure 4.5) firebase provides us with a dashboard to keep track of usage of our real-time database. Currently in the past week there has been 102KB downloads and the current storage for the database is 2.28KB both of which are far below the threshold of needing to upgrade to the “Blaze” plan. If NowDrive were to expand however we do need to consider what pricing would look like for NowDrive. According to the pricing of the blaze plan, it can support 200 thousand simultaneous connections, multiple projects for the database and would cost, \$5 per GB in data stored and \$1 in GB downloaded (Google, n.d.). In terms of authentication, it would cost around \$0.06 per verification (\$0.01 for US, Canada and India). If we were to estimate what a cost would look like under a blaze plan with the assumption we have 1 million users, 2.28GB stored and 102GB downloads we are looking at a once-off payment of \$60,000 for authentication and around \$112 per month for the real-time database.

For APIs we conducted an evaluation on the usage of Google Maps API. With the current free plan with google maps the benefits that are currently included are; up to 1000 static map requests and up to 1000 android google maps SDK requests (Google , n.d.). As you can see from figure 4.6 below we are under the threshold (ignoring the Directions API calls as that had been removed from the project and replaced with the HERE directions API) that would require us to pay for the service however like with firebase we would still need to consider potential costs if NowDrive were to ever expand. According to it's plan, it would cost \$2 a month per 1,000 requests for static map and \$7 a month per 1,000 android google maps SDK request (Google , n.d.). If were to require 56,000 requests for example from both the android google maps SDK and static map, it would cost us \$585 a month to use which is quite expensive to use.

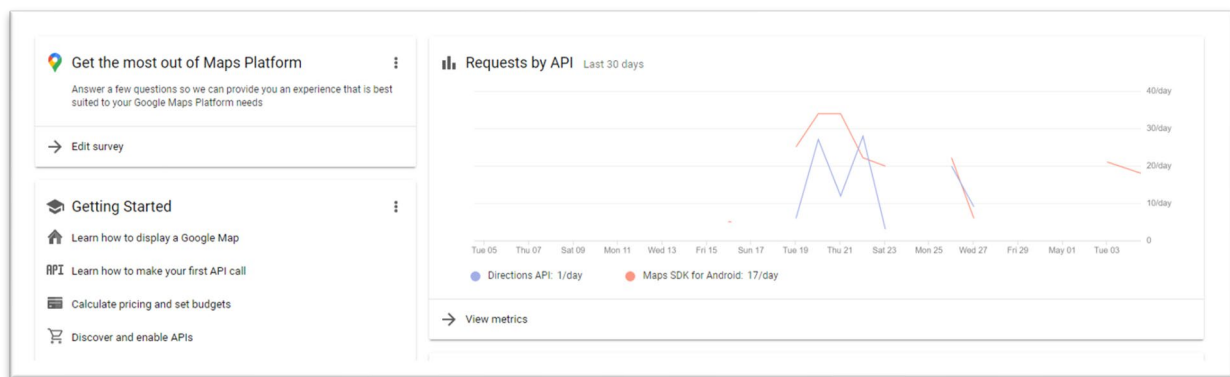


Figure 4.3.6 Google Maps Dashboard

To conclude, I had performed an evaluation on the scalability costs of NowDrive Google's Firebase and Maps services. The gathered results and estimates in costs indicate that Google's firebase is quite cheap to utilize whereas its map services are not cheap, and it would be best to seek out cheaper alternative's to Google's map service.

4 Conclusions

4.1 Advantages and Strengths

NowDrive in its current state is an excellent proof of concept with working functionality that achieves all functional requirements as well as an additional feature of being able to view previous routes created for user accessibility purposes. In its current state the app is now able to create, store and manage both user accounts and their routes, view previously created routes on the same app instance, calculate a complex route taking in account the number of accidents that have occurred in the IFSC area employing the use of weights to calculate risk/reward factors such as faster speed and more dangerous, slower speed and more safe, average speed and average safety.

This allows to give variety to the user in route selection as well as makes NowDrive stand out directly from its respective competitors. With the core functionality solidified in the project this gives room for many possible expansions and sets the standard of how to implement multiple other factors (both historical and current) other than just the number of accidents. The app is also very user friendly in terms of have simple controls that involve tapping and holding tap making this accessible to most people who can drive.

4.2 Disadvantages and Weaknesses

Of course, not every project is perfect meaning NowDrive shares its own set of unique problems. One of the major setbacks for NowDrive currently is the lack of data required it current needs to perform more advanced calculations (more information under heading 5) granted, it is primarily due to outside factors because of the lack of APIs and existing data which resulted in Data having to be manually created in the JSON file (as outlined in heading 3.3.4) to be utilized in the algorithm that calculates such routes. This limitation resulted in the calculation of routes only being pre-set to car accidents and doesn't consider current data or updated historical data because of this simplistic implementation. The data being used is also outdated, dating between the years 2005-2016 with no options of more recent data currently available. With regards to current data, due to lack of timing and resources as well as increased development time required because of lack of historical data, current factors are not currently considered in the algorithm although because of the Nature of the HERE directions API it actively avoids road closures and an implementation for avoid traffic congestion could be possible.

Another setback for NowDrive is security in the application, the application currently employs poor security practices as the primary focus was development and to get an implementation up as soon as possible. This can also be addressed in the future development (more information under heading 5). The last setback for NowDrive currently is it's testing techniques. Throughout the project manual testing has been the main method of testing for NowDrive with 2 very simple unit test cases being implemented to test account information input validation logic. This implementation of testing is poor as their better methods of testing such an application that can be considered in future development (check heading 5).

5 Further Development or Research

With regards to further development and research for NowDrive it would be to address all setbacks that current affect the app as they all can be resolved with the given time and resources required. The first major aspect that would need to be addressed however, would be the lack of data existing to be utilized by the project.

The solution to this problem is complex but very much doable with many solutions, a possible solution to this problem is to simply wait it out and wait for the governmental bodies (in this case RSA, TII and Department of Transportation) to update their APIs to be functional, some data had originally been restricted because of the road traffic collision data is under review for data sharing policies and procedures as of 26th of April 2022 (RSA, 2022). With regards to current data, as mentioned under heading 4.2, an implementation of considering current data can be implemented if we pass in an additional parameter to gather additional data to be considered in the algorithm.

Another solution to this problem would be to conduct a data mining project to collect and format data that can be used to support NowDrive. The focus on which type of data should be mined is data such as car accidents, car related crimes, accident black spots and poorly

reviewed roads/routes. Preferably the data would need to be formatted in the form of an API call send back a JSON object to be supported effectively by the algorithm.

Poor security is another setback that had been defined and needs to be addressed in future development. In its current state, passwords are not encrypted when placed in the database, although when performing authentication of the account on Firebase using its authentication, passwords and user id's are automatically encrypted but when I manually store them on the database it is not encrypted (except for the user IDs which is used as a primary key). To address this is to implement encryption for passwords before being stored onto the database, most likely firebase would have a method on android studio would provide such encryption for the data. Another security concern would be the use of `System.out.println()` throughout the project, the solution to this is to simply remove any instances of it being called in the project to avoid being exploited by attackers. API keys are another area of concern where one API key has all the privileges instead of having restrictions, to solve this would be to restrict the API keys for only what it is being utilized for instead of granting all privileges to just 1 key.

The final consideration of development and research would be improving testing on the project. A consideration for increasing complexity for the tests of NowDrive is to introduce mocking methods and classes for my unit tests to test the core functionality directly from the class/method instead of manually having to type out code like method/class. This can be done by including tools such as Mockito (Faber, n.d.) or Easy Mock (Easy Mock Contributors, 2022). The increased complexity and the mocking of classes/methods will result in better unit testing covering more lines of code and making the tests more accurate and efficient.

6 Appendices

6.1 Project Proposal



National College of Ireland

Project Proposal

NowDrive

01/11/2021

BSHC

Software Development

2021/2022

Emanuel Ivan

X18313591

X18313591@student.ncirl.ie

[Contents](#)

1.0 Objectives	56
2.0 Background.....	57
3.0 State of the Art	57
4.0 Technical Approach	58
5.0 Technical Details	59
6.0 Special Resources Required	60
7.0 Project Plan.....	60
8.0 Testing	63

6.1.1 [Objectives](#)

What I hope for this project to achieve is for the creation of an mobile application accessible for android users that will assist them in planning the safest route possible while driving, this is especially useful for new or learner drivers. More specifically, what I would like to be achieved during the duration of this project is the following:

- Usage of complicated data structures and algorithms to achieve the safest calculated routes possible for the user.
- A user interface that is interactive and easy to understand without a steep learning curve.
- To be as accessible as possible offering a day and night mode and a mode for the colour blind as well.
- An established database that will handle all possible data from the user as well as being aware of GDPR compliance and implementing measures to comply with GDPR.
- Effective testing from user feedback in the app as well as in person demo to gauge an understanding of what users prefer or do not prefer.
- Well written and up to date documentation of the mobile application to keep a backlog of all processes used and the history of development of the mobile application as well.
- Have the application working locally around the Dublin ISFC area and potentially expand if the resources allow for it.

6.1.2 Background

The project is called **NowDrive (ND)** and the reason why I had chosen to undertake this project over other idea I had is that this project clear and easy to understand objectives compared to other ideas. Not only does it have these clear and understandable objectives, but it is also very personal to me as I am currently a learner driver about to sit my first driving test and as a new driver once I get my license I would like to keep my insurance claim free as possible for the first 5 years to help keep my insurance quote as low as possible. Having this app would help me avoid dangerous routes that would cause me to get into an accident and having me needing to file a claim.

There are various ways to make sure that I will set out to achieve the objectives outlined in section 1.0. First, having a project management/Kanban app would go a long way in helping me. Apps such as Trello (Atlassian, 2021), Microsoft Project (Microsoft, 2021) and Monday.com (monday.com, 2021) are good examples, for this project though I will most likely use Trello as it is a free software and a powerful one as well. Secondly, I will make sure that the tasks are digestible and easy to understand making sure that it is possible to achieve them within the timeframe I have and I will make sure to pull back certain features if I feel that the project creep becomes bad. Thirdly, I will do the necessary research and learning to equip me with the skills necessary to achieve the technical goals of the project especially when handling very complicated and new processes that I had never implemented before such as an map mobile application and complex algorithms.

6.1.3 State of the Art

The most obvious app that would be similar to this would be Google Maps (Google, 2021). Google maps is an application that provides route planning to the user from point a to point b and allows users to add waypoints that they must go to throughout their journey if they so desired. Another app similar to mine would also be Waze (Waze Mobile, 2021), Waze informs about road closures, construction, police and other current traffic related data and gives you the option to choose an alternative route. Waze would be the most similar idea to my application by comparison compared to other apps and Google Maps.

What makes my idea stand out from Waze, Google Maps and any other navigation GPS app would be that while other apps use current data, I will be using previous data and history to help calculate routes. For example, if a route is prone to plenty of accidents and there is a noticeable and clear pattern there, the app would try redirect the user to a safer road that isn't prone to many accidents. Another example would be a road that had been reported many times to have potholes, but nothing has been done about it would also help the app calculate a safer route. Of course, it will still take into consideration other current pieces of data such as congestion and construction, but the main idea of the app is to pick up patterns and redirect the user to avoid dangerous roads based on the road's history.

6.1.4 Technical Approach

I will be approaching development with the Agile methodology. The reason why I will approach it this way as it will cut back on the documentation required for the project that would delay other key important factors of the project and as I am on a short time frame using Agile will be most efficient in achieving results in the given time frame. Using Agile will also keep me in the loop with my stakeholders as well as keeping tasks simple and easiest to understand and allowing me to easily adjust the project to the needs of the stakeholder.

To help identify requirements in my project I will be using BABOK Chapter 7 (International Institute of Business Analysis, 2015) which is "Requirements Analysis and Design Definition". I will be implementing four of the techniques outlined in chapter 7. These techniques are Data Modelling, Data Flow Diagrams, User Stories and Use Cases and Scenarios. The reason why I had chosen these is because I feel they are more relevant to the project when compared to other techniques and these techniques will be able to convey the requirements as clearly as possible helping me keep track of it and not to lose sight or focus of the main goals of the project. It is also the most effective and understanding the processes behind ND.

The reason why I feel that it is more relevant to the project and helps me understand the requirements better is the follow:

Data Modelling: used to model requirements to show how data will be used to meet stakeholder information needs (International Institute of Business Analysis, 2015), will be very useful in understanding how the data will be used in ND.

Data Flow Diagrams: used to visualize data flow requirements (International Institute of Business Analysis, 2015), will be good for further documentation for the data as this will be a very technical project having a technique that models the flow is key to understanding the requirements.

User Stories: used to specify requirements as a brief statement about what people do or need to do when using the solution (International Institute of Business Analysis, 2015), will be good in understanding specific needs and actions of the stakeholder that would be relevant in use ND.

Use Cases and Scenarios: used to model the desired behaviour of a solution, by showing user interactions with the solution, to achieve a specific goal or accomplish a particular task (International Institute of Business Analysis, 2015), this will be good by further understanding of the Stakeholder and how by clarifying their interactions with ND and what their goals are and what they wish to accomplish.

As mentioned in section 2.0 I will be using project management tools to break down tasks into small bite sized tasks that can be done over a period of a day or week leading up to the overall goals for longer time periods that would take weeks or months to complete. This will help keep in track everything and gauge a good understanding of what will need to be done and when.

6.1.5 Technical Details

The implementation of this project will be done using the IDE Android Studio (Google, 2021) and with the IDE I will be programming in Kotlin and XML majority of the time. I feel that the implementation of a firebase database and the reason why I decided to go with Android Studios and Firebase is that currently I will be focusing towards android users as android is the fastest way to get an app up and running quickly as well as majority of phone users are android and the reason why firebase would be because firebase has the superior support for android devices and provides a free plan which suits my needs and demands for this project.

Android Studio also incorporates the use of the google maps API and is supported greatly with well written documentation regarding its use in android studios. The idea is to have the selected algorithm plot out the route on google maps for the end-user.

With regards to which algorithms, I am considering the main one would be the cheapest insertion algorithm. The reason why I am considering this algorithm would be that I can add and remove weights to certain routes depending on the road for example if a road has the shorter route but has a bad history of accidents or traffic is currently bad in the area it will begin to add weight to the route and may choose another route that although might be slightly longer it is the safer route which is the main priority of this application. Ideally, we would prefer routes that are Safe and Short but in certain circumstances that may not be the case and the slightly longer alternative route might be the safest approach to it. The idea of adding and removing weights between routes is very appealing to me and an implementation of such algorithm might be what I need for this project.

I am also considering the nearest insertion algorithm. This is different from cheapest insertion algorithm because in the Nearest Insertion it will find the node that is not already on the route and will choose the closest the node that is close to any other node in the route and will stop when no more insertions remains whereas in the cheapest Insertion it will find a node that is not already in the route between two connected nodes in the sub-route and will result in the shortest route possible. I feel that both nearest insertion and cheapest insertion are very good algorithms to implement and are similar in some aspects and I will need to do extensive testing and research into the stronger option of the two before proceeding to choosing one over the other.

Another consideration for algorithms would be the greedy algorithm. This would be the opposite of the cheapest insertion algorithm as it will instead prioritize paths that have the highest weight to them regardless of how long the overall iteration took whereas cheapest insertion will take that into consideration. Because of that reason I do not like the idea of implementing the greedy algorithm as it would end up calculating ridiculously long routes for the user.

The data needed for the calculation of the weighting of per individual node will be done with the use of various APIs with some of them being located on the data.gov.ie website regarding road history (Road collisions, pothole report, etc) and also the use of APIs located on the transport Infrastructure Ireland website which is data.tii.ie for live data

about the roads (Traffic, weather, travel times, diversions, etc). With the use of the data being extracted from these APIs in the weight of plots between nodes in either of the cheapest insertion algorithm or nearest insertion algorithm it would be very beneficial to finding the most effective and safe routes for the user.

6.1.6 Special Resources Required

Because of the nature of this project no special resources will be required such as hardware needs as testing can be emulated for various types of android mobile devices.

6.1.7 Project Plan

As mentioned in section 2 I will be using a project management tool, the tool I have decided to go with is Trello and I have created many tasks with one of two labels being either “Documentation” or “Programming Task”. A Documentation card is tied into the task of documenting the project from what we can see in figure 1.0 below we can see various cards are assigned this label such as “Create Data Model” or “Create User Stories”. Programming Tasks on the other hand will be dealing with the physical task for programming certain defined features in the card and as we can see from the figure 1.0 there is various examples of it such as “Initial GitHub Setup” and “Implement Google Maps API”.

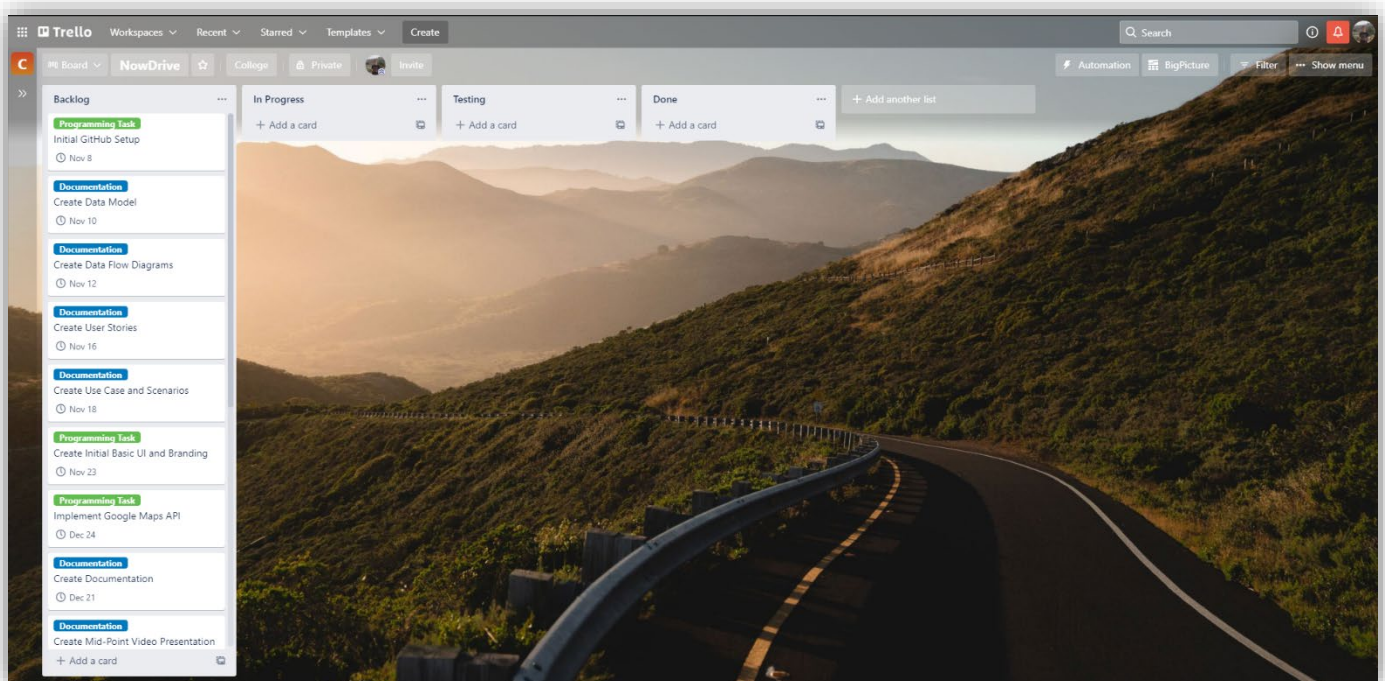


Figure 1. 0. Trello

- Documentation
- Programming Task

There are various stages that I have added on Trello which is Backlog, In Progress, Testing and Done, all are viewable in the above Figure 1.0. Backlog is the accumulation of tasks that had not been started yet but are planned for in advance and this is the first stage of any task at hand for the project. The second stage would be the “In Progress”, it is self-explanatory where any tasks assigned to this stage are in progress and have started. The stage after that would be Testing, in this stage the implemented tasks will be doubled checked and tested on using various methodologies before moving on to the final stage which is “Done”, in the Done stage the task has been successfully completed and went through testing as well.

But Trello it self is not enough to support my needs of tackling this project and I needed an implementation of an Gantt chart of sorts to help keep my tasks aligned and provide a consistent high level and clear view of all of the tasks, their start dates and their due dates. To solve this issue I have install a Plugin (Known as Power-Ups in Trello) called BigPicture. What BigPicture does is generates a Gantt chart for the Trello page as well as edit it as a traditional Gantt chart similar to how it would be done on Microsoft Project or various other project management applications. Below is a screenshot in figure 1.1 taken of the plugin in action and the current beginning of the Gantt chart.

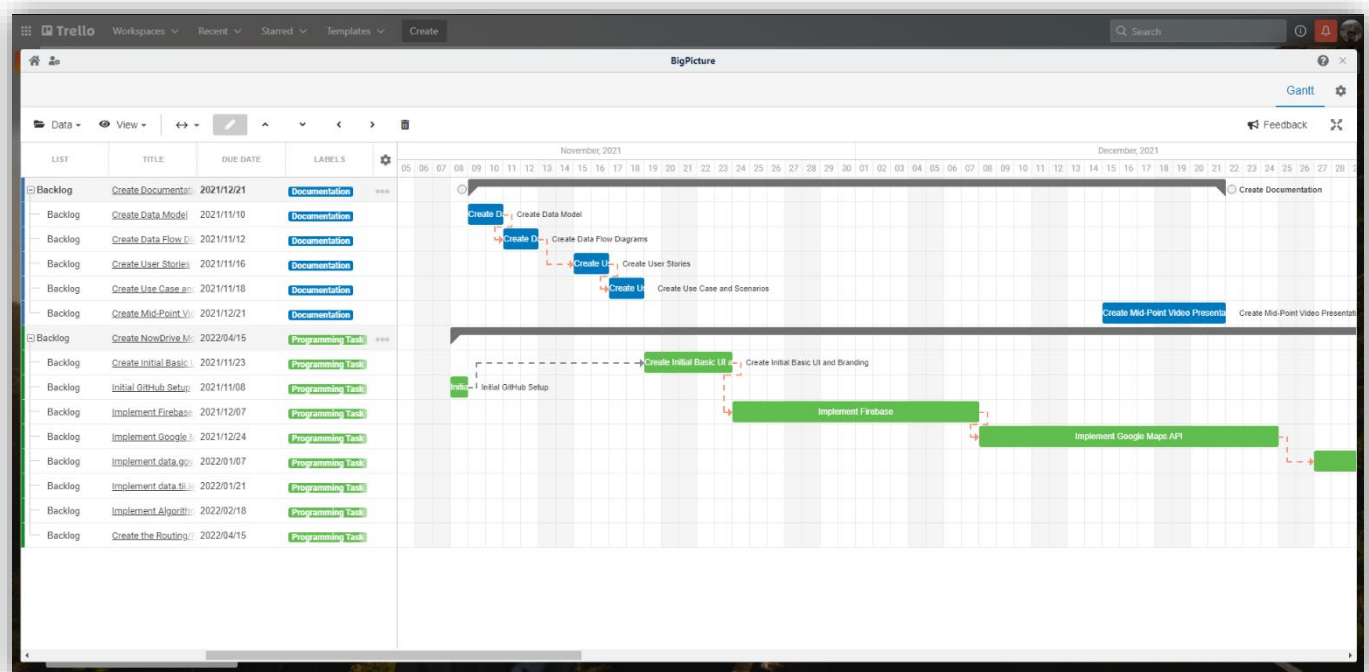


Figure 1. 1. Gantt Chart

■ Documentation
■ Programming Task

Above in figure 1.1 is the Gantt Chart and like in the Trello it is colour coded into Documentation and Programming Task. From the highest-level view now there two main tasks being “Create Documentation” and “Create NowDrive Mobile Application” and is broken up into smaller sub tasks in those two main tasks. This Gantt chart gives me a clear view of what needs to be done and when it needs to be done by as well as the time I have allocated to that task. There are plenty of sub tasks in the project for both Create Documentation and Create NowDrive Mobile Application main tasks lets delve into the sub tasks deeper and provide a more elaborate explanation for all the current subtasks.

Documentation Sub-tasks:

Create Data Model: It starts at the 9th of November and is due by the end of 10th of November. The task is to create a data model to reflect off the anticipated end result of the mobile application and how data is used to meet stakeholder requirements.

Create Data Flow Diagrams: It starts on the 11th of November and is due by the end of 12th of November. The task is to create various data flow diagrams to show how data and information will be used and distributed throughout the system.

Create User Stories: It starts on the 15th of November and is due by the end of 16th of November and the task is to create user stories that will describe what a stakeholder’s needs and wants from this project would be.

Create use Case and Scenarios: It starts on the 17th of November and is due by the end of the 18th of November and it is to create use case and scenarios for the project which is to model the desirable result from user interaction.

Create Mid-Point Video Presentation: It starts on the 15th of December and is due by the end of 21st of December, the task is to record a video presentation for the Mid-Point of the project.

Programming Sub-tasks:

Initial GitHub Setup: It starts on the 8th of November and is due by the end of the 8th of November, the task is to create a git repository for the project and set up the initial project file.

Create initial basic UI and Branding: It starts on the 19th of November and is due by the end of 23rd of November. It is to create the branding and basic UI for the mobile application.

Implement Firebase: It starts on the 24th of November and is due by the end of 7th of December, the task is to have a basic implementation of firebase working on the application.

Implement Google Maps API: It starts on the 8th of December and it is due by the end of 24th of December, the task is to implement the Google Maps API onto the project and have the basic functionality working.

Implement data.gov.ie API Data: It starts on the 27th of December and is due by the end of 7th of January 2022. The task is to get the data.gov.ie API to send data on to the project with data being structured and utilised on the project.

Implement data.tii.ie API Data: It starts on the 10th of January 2022 and is due by the end of 18th of January 2022 and the task is to be able to receive data from the data.tii.ie API and get it to be utilised in the mobile application.

Implement Algorithm: It starts on the 24th of January 2022 and is due by the end of 18th of February. The task is to implement the chosen algorithm onto the mobile application and have it be able to calculate the intended routes.

Creating the Route/Pathing generation from Algorithm to appear in Google Maps API: It starts on the 21st of February 2022 and is due by the end of 15th of April, the task is to make the working algorithm to plot the routes on the Google Maps API.

6.1.8 Testing

Android studio already comes with testing implementation into the IDE that will be used throughout the development of this project. There are two types of testing I can implement in android studios which is Local unit tests and Instrumented tests. Local unit tests would be conducted on my Java Virtual Machine and do not require Android framework dependencies although the complexity for the testing will be simplified and only used for simple logic checking. The implementation of Local unit tests however is quick and simple to set up meaning I can have tests checking the logic after a change has been implemented to make sure any major changes did not break the code.

Instrumented tests run on the emulator or hardware device and will need the use of Instrumentation APIs but offers far greater complexity for example, when you implement a test like this it would give access to information to the code such as the context of the app that is being tested and because of this it is possible to automate user behaviour in the app helping to test certain test cases and flag any issues that would arise. This would be very useful the more technical and bigger the project becomes as it allows to keep track of all the changes and how it affects user interaction with the mobile application.

Although this covers technical testing there is still the issue where I would need to test the application and receive some form of user feedback to gauge an understanding of how the app is viewed by the end-user. To do this I will require to submit an ethics form where I will have a pop up on the mobile application asking the user if they wish to leave a review of the applications and give it a simple star rating system for various aspects of the mobile application and when I feel the app is ready to be tested against live end-users

I will make sure to control access to the app and having a set number of individuals being able to use the application. With this feedback I will be able to apply necessary changes to meet the demands and improve the overall quality of the mobile application.

References

Atlassian, 2021. *Trello.com*. [Online]

Available at: <https://trello.com/>

[Accessed 28 October 2021].

Google, 2021. *Android Studio*. [Online]

Available at: <https://developer.android.com/studio/>

[Accessed 29 October 2021].

Google, 2021. *Google Maps*. [Online]

Available at: <https://www.google.ie/maps>

[Accessed 28 October 2021].

International Institute of Business Analysis, 2015. *BABOK*. 3rd ed. Toronto: International Institute of Business Analysis.

Microsoft, 2021. *Microsoft Project*. [Online]

Available at: <https://www.microsoft.com/en-ie/microsoft-365/project/project-management-software>

[Accessed 28 October 2021].

monday.com, 2021. *Monday.com*. [Online]

Available at: <https://monday.com/>

[Accessed 28 October 2021].

Waze Mobile, 2021. *Waze*. [Online]

Available at: <https://www.waze.com/>

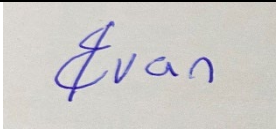
[Accessed 28 October 2021].

6.2 [Reflective Journals](#)

6.2.1 [October:](#)

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

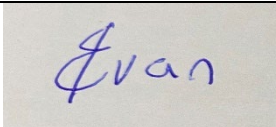
Month: October

What? I have written up the proposal and have created a Gantt Chart and Trello for the software project.	
So What? Because I have created the Gantt Chart and Trello it will become easier to stay on top of my project and see from a high level perspective what will need to be done. I have also submitted my project proposal hoping it will make it easier to convey the idea of my project to my supervisor.	
Now What? The Trello and Gantt chart will need to be kept up to date as the project goes on due to project scope changing either it decreases and increases, the challenge here is to further refine my Trello and Gantt chart with the necessary tasks at hand. I will then need to begin the project and will follow closely with the Gantt Chart and Trello starting on reading week.	
Student Signature	

6.2.2 November:

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

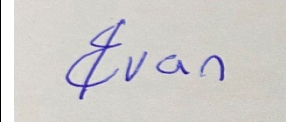
Month: November

What? I drafted up use case and data flow diagrams to be used in my midpoint technical report.	
So What? Because I have created the use case diagrams it would be easier to address the use cases and data for my midpoint technical report as I will be writing in accordance to the diagrams I have created.	
Now What? I will need to write the use cases to support the use case diagram as well as writing up the data requirements which supports the data flow diagram, I will also need to work on addressing my other requirements as well as creating a prototype for the show case which will follow some of the use cases created and represents some of the data outlined in the data requirement and then writing up documentation for the prototype on my technical report.	
Student Signature	

6.2.3 December:

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

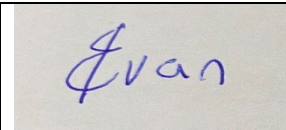
Month: December

What? I created my prototype, written up the technical report and recorded and edited my presentation.	
So What? I now have a working prototype as proof for concept as well as written up documentation for planning and details about the project in my technical report that will help effectively communicate the purpose and objectives of the NowDrive project and what has been done so far and what needs to be done in accordance with the requirements.	
Now What? I will need to continue developing my prototype next year and will need to update my documentation to reflect of it's current stage as well as be prepared to answer any questions that people may have about my project with regards to the presentation, prototype or technical report.	
Student Signature	

6.2.4 January 2022:

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

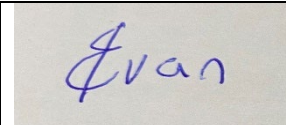
Month: January

What? I typed up the necessary summary, intro, and details for my project showcase profile, I also revisited and reorganised my Trello page for the project.	
So What? Because I have typed up the necessary info for my project showcase it means it can be evaluated by my supervisor and ready for feedback and any changes necessary. Because I revisited my Trello page it was able to change in accordance my objective for semester 2.	
Now What? I will need to take a picture for my showcase profile, and I will need to start the further development of my project.	
Student Signature	

6.2.5 February:

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

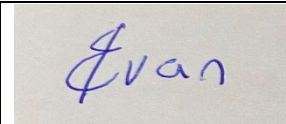
Month: February

What? I took a picture for my project showcase profile and I also edit the details as well in accordance to recommendations given to me. I created imagery to be used for the banner and project showcase.	
So What? Because I have taken a profile picture for my project showcase I will be able to come across as a professional and people would look further into my project. Because I have my images created and edited it will mean I will be able to show exactly what my project is just by using imagery.	
Now What? I will need continue work on my project from a technical perspective and strive to improve my project showcase where I can.	
Student Signature	

6.2.6 March:

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

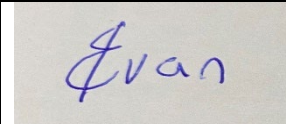
Month: March

What? I began familiarising myself with my project again and looking up tutorials and new industry standards to implement for my project.	
So What? Because I have familiarised with my project, it will allow me to work more effectively once I begin to delve deeper into the production of the project.	
Now What? I will continue to work on my project and begin rolling out more complicated features for the application.	
Student Signature	

6.2.7 April

Student Name	Emanuel Ivan
Student Number	X18313591
Course	BSHCSD

Month: April

What? I have completed the technical aspects of my project with regards to the map implementation, route calculation, firebase database, saving/removing routes and plotting routes on the map. I have also kept notes of any encountered bugs and how they were resolved. I have also completed the poster for my project.	
So What? Because I have completed the technical aspects of my project it will allow me to focus on testing and documentation of the project and because I have kept notes of my debugging it gives me much more notes to write about in the documentation about testing. Because I have the poster completed it will make my project much more visible in the showcase.	
Now What? I will now need to continue in my documentation as well as implement some test cases in the project and I will then need to record a presentation for my project which will need me to create supporting slides.	
Student Signature	

Technical Report References

Atlassian, 2021. *Trello.com*. [Online]

Available at: <https://trello.com/>

[Accessed 28 October 2021].

Easy Mock Contributors, 2022. *Easy Mock*. [Online]

Available at: <https://easymock.org/>

[Accessed 5 May 2022].

Faber, S., n.d. *Mockito*. [Online]

Available at: <https://site.mockito.org/>

[Accessed 5 May 2022].

Google, n.d. *Google Maps Platform Pricing*. [Online]

Available at: <https://mapsplatform.google.com/pricing/>

[Accessed 4 May 2022].

Google, 2021. *Android Studio*. [Online]

Available at: <https://developer.android.com/studio/>

[Accessed 29 October 2021].

Google, 2021. *Google Maps*. [Online]

Available at: <https://www.google.ie/maps>

[Accessed 28 October 2021].

Google, n.d. *Firebase Pricing*. [Online]

Available at: <https://firebase.google.com/pricing>

[Accessed 4 May 2022].

Google, n.d. *Volley*. [Online]

Available at: <https://google.github.io/volley/>

[Accessed 6 May 2022].

International Institute of Business Analysis, 2015. *BABOK*. 3rd ed. Toronto: International Institute of Business Analysis.

Microsoft, 2021. *Microsoft Project*. [Online]

Available at: <https://www.microsoft.com/en-ie/microsoft-365/project/project-management-software>

[Accessed 28 October 2021].

monday.com, 2021. *Monday.com*. [Online]

Available at: <https://monday.com/>

[Accessed 28 October 2021].

RSA, 2022. *Road traffic collision data*. [Online]

Available at: <https://www.rsa.ie/road-safety/statistics/road-traffic-collision-data>

[Accessed 26 April 2022].

RSA, n.d. *Road Collision Map*. [Online]

Available at: <https://www.rsa.ie/road-safety/statistics/collisions>
[Accessed 6 May 2022].

Waze Mobile, 2021. *Waze*. [Online]

Available at: <https://www.waze.com/>
[Accessed 28 October 2021].