Fiachra O'Connor

Submitted in partial fulfilment of the requirements for the degree of

BSc in Business Computing

Technological University Dublin (City Campus)

May 2021

Supervisor: Dr. Jenny Munnelly

Declaration

This is an original work. All references and assistance are acknowledged.

Signed: Fiachra O'Connor

Date:

Acknowledgements

I would like to say a special thanks to everyone who helped me in not only me FYP and 4th, but throughout my whole journey at TUD. It has had its ups and downs, but all in all, an experience I will never forget.

Firstly, I would like to thank my family who have kept me motivated throughout 4th and encouraged me to keep pushing on when I didn't have the motivation. I don't think I would have achieved what I have achieved this year without their continued support.

Secondly, all the lectures I've had the opportunity to learn from have played a key role in the completion of my final year project. The key values of time management, hard work and teamwork were skills I would have not learned without them and I am very grateful for that.

First, I would like to thank Catherine Higgins for being the driving force behind the course throughout my first few years in Business Computing, her guidance and willingness to go the extra mile has not gone unnoticed. Secondly, a special thanks to Jenny Munnelly, for her continued supports throughout my final year project. Jenny was always there when I needed some guidance or motivation. Her ability to open mind to many different avenues and ideas for my application and her constant encouragement really meant a lot. Also, a big thank you to Farrah Higgins, my second reader, for not only her time and efforts during my checkpoints but her continuous support throughout 3rd and 4th year. Finally, I would like to thank my classmates who have provided me with some very great moments throughout my time at TUD.

Abstract

SAFIO is an android application which its main goal is to decrease the spread of Covid 19. The application uses the user's location and keeps them informed of what streets, shops and Dublin Bikes stations to avoid throughout the day.
In this report I will demonstrate the systems main features and how these features were designed, and the technologies involved in making my application. I will discuss how I developed each feature and any problems I encountered throughout the whole process.

The goals of the project were to:

- Ensure the safety of the city of Dublin by helping them choose the most safe to travel
- Lower the spread of Covid 19 by minimizing the amount of large groups gathered on streets, shops and restaurants
- Give users a better understanding off what a safe streets is what a not safe streets is during a pandemic
- Keep users informed of how at risk they would be in a shop or restaurants

Introduction of Project

The summer before coming into final year was massively hindered by Covid 19. By this time, the virus had spread throughout the world and killed people in its masses. There is no doubt Covid-19 has had a massive impact on our lives over the past year and many of us did our part to hinder such a further widespread of the virus. I knew instantly that I wanted to use the skills I have gained in college to design an app which will help lower the spread of the virus and help my city of Dublin to tackle this widespread problem. I studied the market extensively and investigated all the apps which are linked to Covid-19 and social distancing. One that really stood out to me was the Covid -19 tracker app, which informs users when they have come into close contact with an infected individual. I thought, "what if there was an app which helped its users to avoid people in general and completely eliminate the chances of coming into close contact?". Therefore, I decided to use this deadly pandemic to my advantage, and this has resulted in the creation of my android application, SAFIO.

SAFIO is an android application which has the ability of lowering the spread of the virus throughout Dublin city center and to keep its users safe, whether it's on the roads or on the streets.

Firstly, once the user has registered and logged in, SAFIO has already started to calculate their health score. This is then followed by a multitude of health-based questions which will determine how at risk a certain individual is to the effects of Covid-19. This is calculated by their age and the score they get in the health exam. Once this is done, the user can start to use the app much like Google Maps. Wherever the user would like to go, SAFIO will ensure that this location is safe, whether it's a street in the city center, a local shop or their nearest Dublin Bikes station. This is achieved using Google APIs, Smart Dublin, which is an open data source with datasets on streets and footfall foot fall Dublin and the Dublin Bikes API along with a multitude of complex algorithms and databases.

Objectives of Project

The objective of this project is to allow the people of Dublin city to be able to travel throughout the city without coming into close contact with other people. So long as they do not come into close contact, it will result in a drop in the number of cases of Covid 19.

At the moment, the main objective of Dublin is to get back to normal as soon as possible. This will not happen unless we continue to stay away from each other and keep our distances. SAFIO

will make this a lot easier, before you get to your desired location SAFIO will have already informed you of the safety of this location. It is then up to you whether you would like to listen to its recommendations. After all, we all want to go back to normal, and using SAFIO will hopefully aid that process.

Users should be able to:

- Register, Login and Logout
- Answer Health Exam Questions
- See current location on Google Maps.
- Insert desired destination in search bar.
- See location suggestions.
- Receive walking direction to location.
- Find out if a shop or restaurant is safe or not depending on Google popular times.
- Check safety based on different factors.

1. Users' health score
2. Streets average footfall algorithm
3. Users current Time & Date

- Check safety of Roads in Dublin for driving
- Use toggle to see Dublin Bikes stations on maps.
- Check availability & safety of Dublin Bikes stations
- Report a location on the map which may have more people than usual.
- Can retake the health exam if circumstances have changed.
- Edit personal information.

System should provide the following:

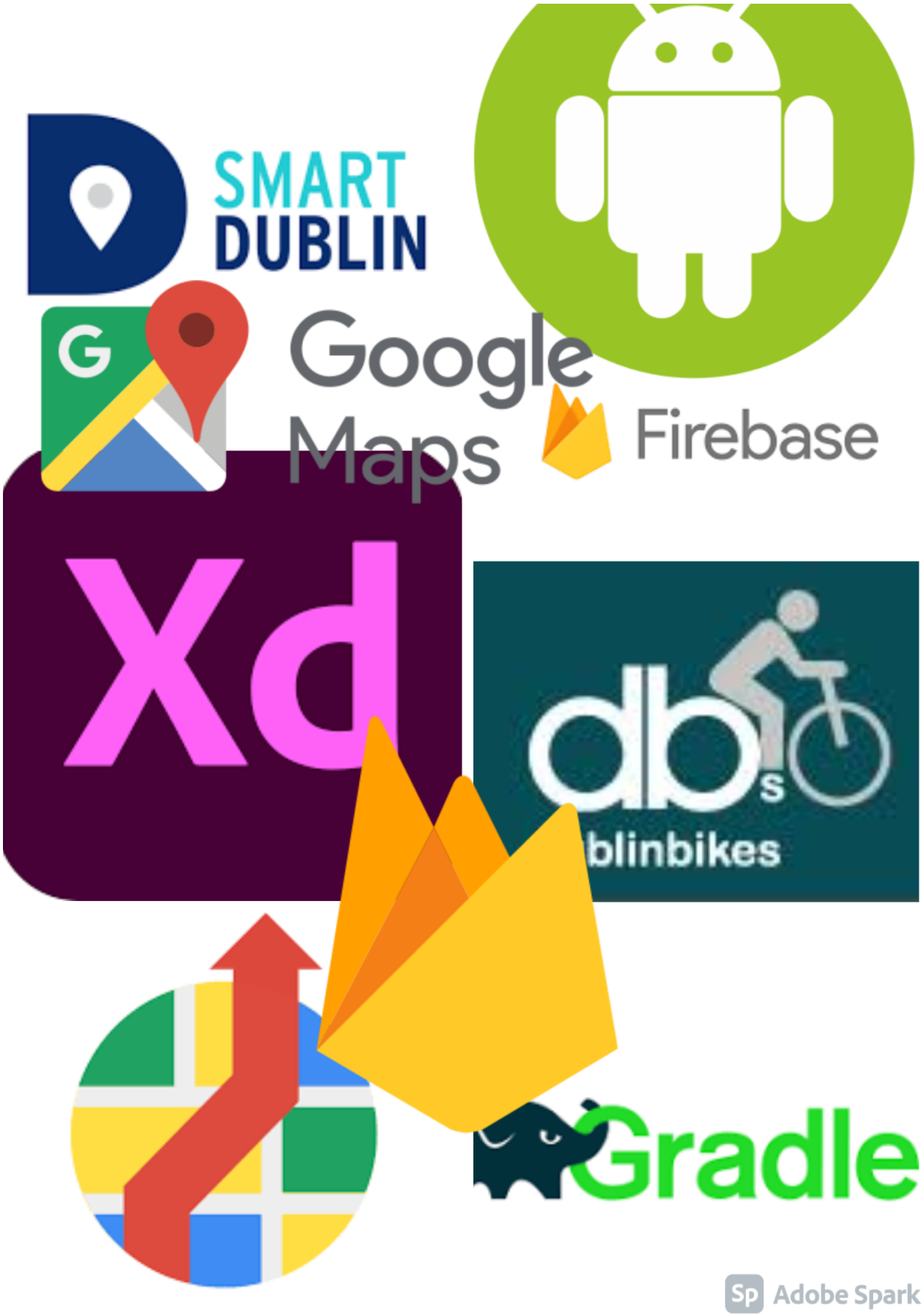
- A way to access core functionality.
- Authenticate a user.
- Generate the weather a location is safe or not for the user based on the health score.
- Give directions for users to desired locations.

Business Case

SAFIO will be very useful, especially in today's world. We are all aware of the impact the virus is having on our society. The fact that it simply offers a safe or not safe ultimatum for the users makes it seamless and easy for them to plan their trip or to visit somewhere within Dublin City Centre. Once a user has completed their online health check, they can avoid certain areas and over all contribute massively to society. Covid 19 is easily spread within large groups. By using SAFIO, a user can simply avoid these large groups.

From my research of the market, there is no app which replicates what SAFIO overs in terms of determining whether a shop, street or restaurant is safe. The closest would be the Google Popular Times function that Google offers on each shop and restaurant, but this is not tailored to the user. They are assuming that you're at zero risk. With SAFIO, by taking the health exam, we can determine how safe a street, shop and restaurant is in accordance with your exam results. For example, a street may be safe for someone who has no health problem, but not safe for someone who has answered yes in the exam to having an organ transplant. This is a truly unique service. Currently, SAFIO is restricted to Dublin City Centre as there is only one small amount of data available, and within the Dublin City Centre there is only 20 streets which are accessible. To take SAFIO to the next step, the aim would be to work extensively with Smart Dublin to start to retrieve footfall data from more and more streets so that we can manage the next virus a lot better and more efficiently.

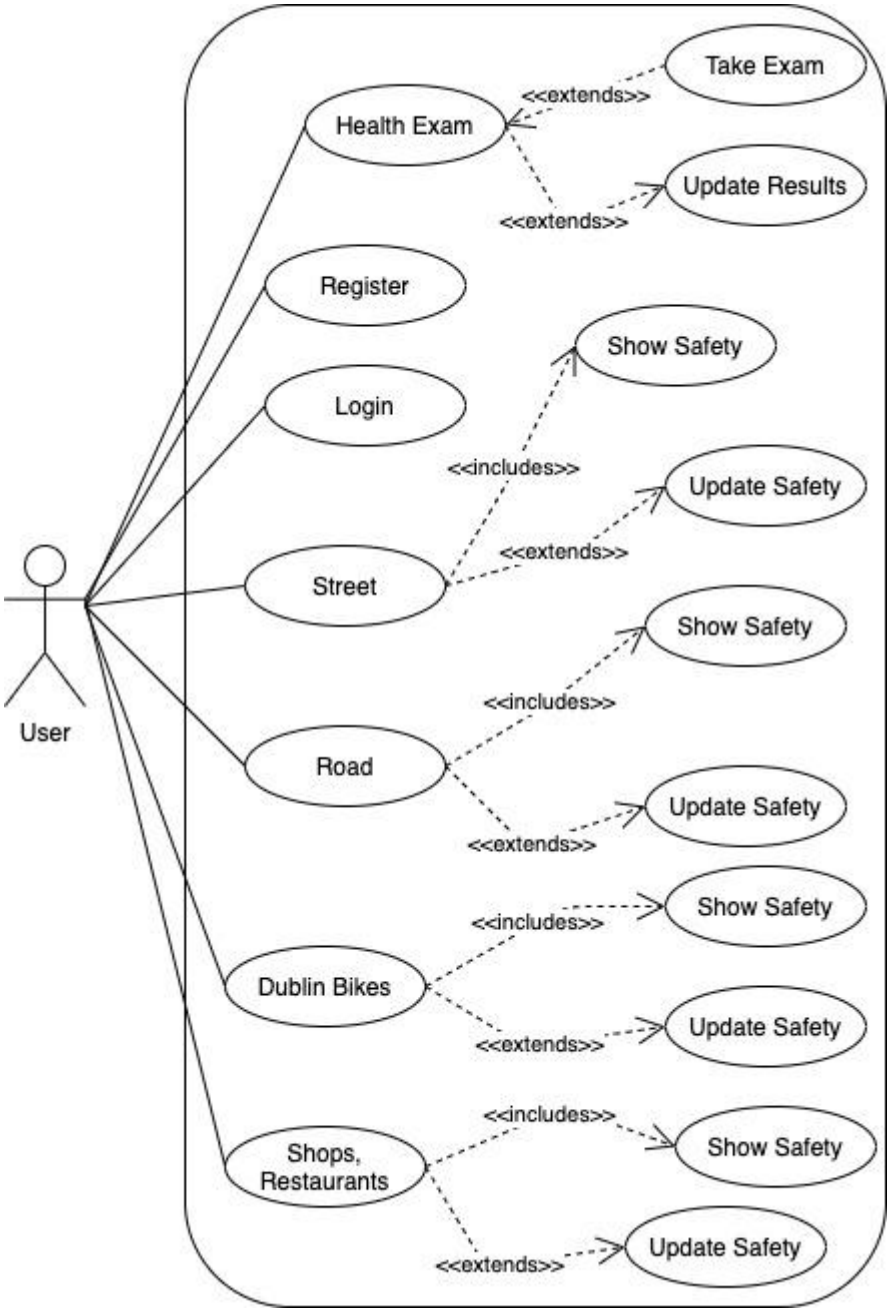
Overview of Technologies



Requirement Capture & Analysis

Business Actors

In my application there is only one business actor and that is the user.



Requirement Analysis

Function	Requirement	Justification
Register	New users can be created with pharmacy details	User needs to be logged in to use the application.
Log in/Log out	Existing users can log in log out	User needs to be logged in to use the application.
Health Exam	Users answer questions about their health	User can get a better more specified safety return depending on their health score
Retake Health Exam	User can reanswer health exam if predicament has changed	Helps user get a more accurate safe or not safety
View Street Safety	User can check if street is safe or not	Helps user avoid close contact with other people on the street
Update Street Safety	If health exam is retaken users or different date or time user can get a new safety value	Users can get a more accurate safe or not safe return value
View Road Safety	Users can check the safety of the roads	Helps users planning a driving trip to know if the roads are safe to drive
Update Road Safety	Road safety updates depending on users date and time	Helps user to avoid the busiest times on the road to avoid traffic and a potential accident
View Establishments Safety	View all establishment busy times using Google busy times	Users can avoid coming into close contact inside an establishment
Update Establishments Safety	Update safety depending on the time and date of the user	Users can get a more accurate safe or not safe
View Dublin Bikes Safety	Users can check the safety of a dublin bikes station of their choice	Users can avoid coming into contact with people at a dublin bikes station
Update Dublin Bikes Safety	Update safety depending on users date and time	Gives the user a more accurate determination of safety at a Dublin Bikes Stand

Non Functional Requirement

Ease of use

Make the User Interface layout easy and understandable. Whenever a user sees any components like buttons, it should be straightforward to understand the underlying functionality or what the component does.

Simplicity

Screens in the app should be simple with no nonessential buttons or non useful information for the task or feature at hand. Users should find it easy to see or find any clickable components or available functionality to them from any specific screen. Avoid having hidden menusor core functionality in a place where it is not common sense for the user to find it in. Any names within the app should be simple and understandable without the use of complicated words.

Theme

The user should be presented with the same theme within the app when accessing different screens. Having common colours, buttons or text style is essential to the user experience. Any lists, dialogs or links must use the similar design.

Logs

Any calls to the backend or processes that can result in different outcomes orthrow exceptions should be logged to the console. Therefore, when debugging, the logs can be analysed to see what went wrong.

Usability

Users should have their location enabled so that SAFIO can track their current location and give them the best safe or not safe for all the locations.

Availability

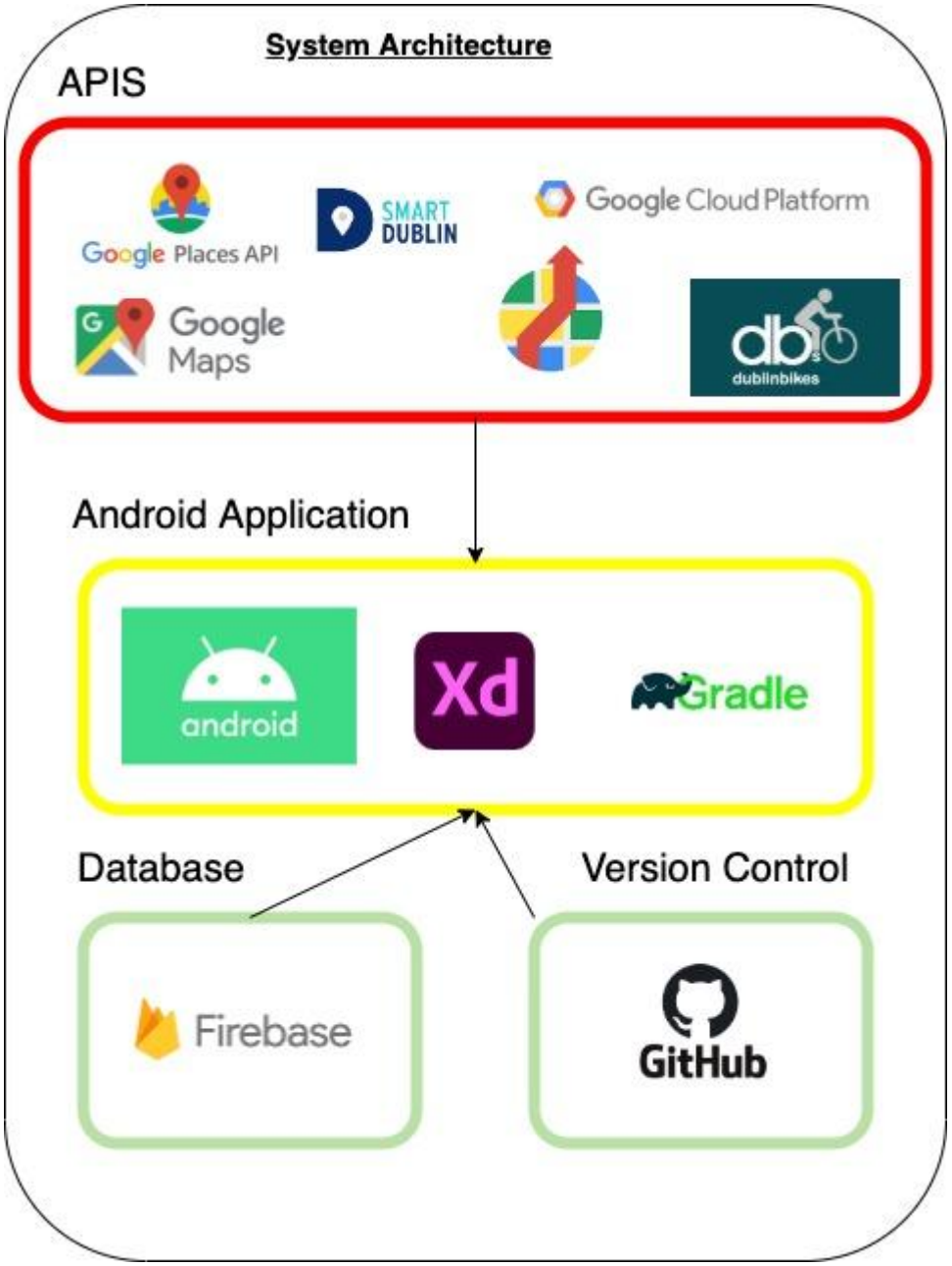
Users should be able to connect to the internet otherwise the app will not function at all.

Security

To access the application the user must be logged in and fully authenticated. Thanks to firebase the authentication function the users data is safe and protected.

Design

System Architecture



Choice of Technologies

Java

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. Java was the first language I had learned in my time during college and felt my strengths lied with this language, hence why I chose it as my main coding language for this project.

Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. In 3rd, I had a lecture in python and took immense pleasure in solving the difficult problems that came with using python so I decided to find a way to implement it in my project to show my capability of understanding more than one coding language.

Firebase

Firebase is a database service managed by google. Firebase offers a wide range of easy to use products. One being firebase cloud storage which is exabyte scale object storage solution. I use this to store prescription images. I enjoyed using firebase and will definitely use it in future.

Google Cloud Console

Google cloud manages and gets insights into everything that powers your cloud application — including web applications, data analysis, virtual machines, datastore, databases, networking, and developer services. Cloud Console helps you deploy, scale, and diagnose production issues in a simple web-based interface. Search to quickly find resources and connect to instances via SSH in the browser. Handle DevOps workflows on the go with powerful native iOS and Android applications. Master the most complex development tasks with Cloud Shell, your admin machine in the cloud. I made great use of the google cloud platform for my Google Places API, Directions API, Places API and having the ability to host my Popular Times algorithm in the cloud functions.

Google Maps API

Google Maps is a Web-based service that provides detailed information about geographical regions and sites around the world. In addition to conventional road maps, Google Maps offers aerial and satellite views of many places. In some cities, Google Maps offers street views compromising photographs taken from vehicles. The Google Maps activity

was incredibly useful to help users identify the areas they should avoid by using the places api and directions api.

Google Directions API

The Directions API is a web service that uses an HTTP request to return JSON or XML-formatted directions between locations. You can receive directions for several modes of transportation, such as transit, driving, walking, or cycling. I made use out of the google directions api so that a user can get walking directions to their desired location.

Google Places API

The Places API is a service that returns information about places using HTTP requests. Places are defined within this API as establishments, geographic locations, or prominent points of interest. I used Google Places so that a user can search their desired destination and find out the safety of this place.

Smart Dublin API

Smart Dublin brings together technology providers, academia and citizens to transform public services and enhance quality of life. Founded by the four Dublin Local Authorities, our goal is to future-proof the Dublin region by trialing and scaling innovative solutions to a wide range of

local challenges. Smart Dublin provide their own REST API service with each dataset they have. I made use of two datasets as part of my project and learned a lot about data retrieval from apis.

Dublin Bikes API

Dublinbikes is a public bicycle rental scheme which has operated in the city of Dublin since 2009. At its launch, the scheme, which is sponsored by JCDecaux, used 450 French-made unisex bicycles with 40 stations. I made use out of Dublin Bikes own API to use their data.

Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. Having experience developing java applications helped when creating an android application. Most people's phones use android OS.

Github

GitHub is one of the world's largest community of developers. It's an intricate platform that fosters collaboration and communication between developers. GitHub has a number of useful features that enable development teams to work together on the same project and easily create new versions of software without disrupting the current versions, but it doesn't stop there. I was introduced to Github last year in a dynamic programming module and I was familiar with the basics of version control. Over the duration of the academic year I became stronger at git

commands and using github. I believe this will transfer over to help me in future when working on a team.

Bluestacks

BlueStacks is an American technology company known for the BlueStacks App Player and other cloud-based cross-platform products. The BlueStacks App Player is designed to enable Android applications to run on PCs running Microsoft Windows and Apple's macOS. I used bluestacks as my main emulator to run and test my application. I found Bluestacks to be a lot faster and more simple to use than the built in emulators of android studio.

Adobe xd

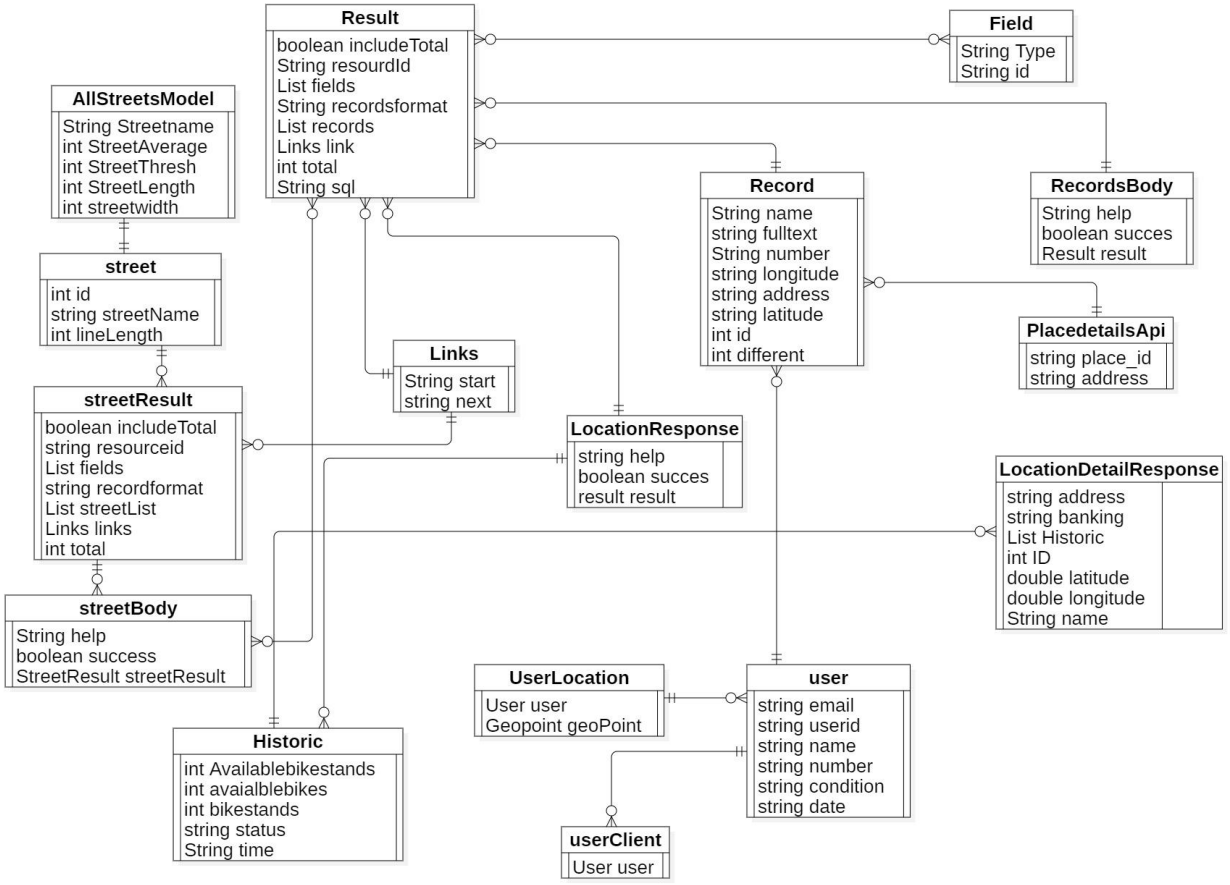
Adobe xd user experience design tool for web apps and mobile apps. It is used for making prototypes for applications. I used it to help me visualize how I want the applications to look like. I mostly used it for Android. I was able to make custom buttons, Images etc that I wanted to use

Gradle

Similar to maven, Gradle is a build automation tool that allows you easily add dependencies and is default for android studio. I believe the syntax for Gradle is easier to read and would scale nicer than maven in my opinion. I preferred Gradle as a build automation tool.

Data Model Design

Conceptual Entity Relational Diagram



Software Design

Street Safety Algorithm

The purpose of this algorithm is to return if a street is safe or not. Using google maps api, SAFIO finds out where the user is.

Firsty, a user answers a multitude of questions regarding their health. Each of these questions will determine the users health score. The health score will prove incredibly important in determining an accurate street safety for this specific user. For each “yes” answer, the health increased by a set number depending on the severity of their health complication. For example, “do you smoke?” carries a score of +2 whereas “have you had an organ transplant?” carries a score of +4. A user's health score is capped at 5.

```
yesCard.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (Constants.CLICKABLE == 1) {
            pagerInterface.addPoints( points: 2,  fragmentNumber: 1);
            pagerInterface.nextPage();
            pagerInterface.updateRiskScore();
            yesCard.setCardBackgroundColor(getContext().getResources().getColor(R.color.lightGray));
            noCard.setCardBackgroundColor(getContext().getResources().getColor(R.color.white));
        }
        else
        {
            Toast.makeText(getContext(),  text: "Please wait while the app is updating values!", Toast.L
        }
    }
});

noCard.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (Constants.CLICKABLE == 1) {
        pagerInterface.minusPoints( points: 2,  fragmentNumber: 1);
        pagerInterface.nextPage();
        pagerInterface.updateRiskScore();
        yesCard.setCardBackgroundColor(getContext().getResources().getColor(R.color.white));
        noCard.setCardBackgroundColor(getContext().getResources().getColor(R.color.lightGray));
        }
        else
        {
            Toast.makeText(getContext(),  text: "Please wait while the app is updating values!", Toast.L
        }
```

```
@Override
public void addPoints(int points, int fragmentNumber) {
    switch (fragmentNumber)
    {
        case 1:
            if (fragment1 == 0)
            {
                fragment1 = 1;
                totalRiskScore += 2;

            }
            break;
        case 2:
            if (fragment2 == 0)
            {
                fragment2 = 1;
                totalRiskScore += 2;

            }
            break;
        case 3:
            if (fragment3 == 0)
            {
                fragment3 = 1;
                totalRiskScore += 2;

            }
            break;
        case 4:
            if (fragment4 == 0)
            {
                fragment4 = 1;
                totalRiskScore += 4;

            }
            break;
        case 5:
            if (fragment5 == 0)
            {
                fragment5 = 1;
                totalRiskScore += 2;

            }
            break;
```

Once the health score is determined, the user is then brought to the main maps activity where they can check their current location. Checking their current takes will now give them the current info of this street and will tell them if it is safe or not. The safe or not safe will be determined on the risk of the user and the amount of people that SAFIO believes that are on the street. During this process, SAFIO is taking in the users location, and then going into the Smart Dublin API to find this location in the data and then retrieving the size of the street and footfall.

```
if (location_Address.contains("O'Connell Street Lower")) {
    if (record.getOConnellStOutsideClerys() == null) {
        footFallList.add(0);
    } else {
        footFallList.add(record.getOConnellStOutsideClerys());
    }
    streetName = "O'Connell Street Lower";
    width = 10;
} else if (location_Address.contains("O'Connell Street Upper")) {
    if (record.getOConnellStParnellStAIB() == null) {
        footFallList.add(0);
    } else {
        footFallList.add(record.getOConnellStParnellStAIB());
    }
    streetName = "O'Connell Street Upper";
    width = 10;
} else if (location_Address.contains("Mary St")) {
    if (record.getMaryStreet() == null) {
        footFallList.add(0);
    } else {
        footFallList.add(record.getMaryStreet());
    }
    streetName = "Mary Street";
    width = 5;
```

Once this data is retrieved, the number of people on this street is determined. This number is calculated by getting the average footfall of the user's current location and current time and date in 2020. For example, if Joe is on Grafton Street at 9am on a Friday in January in 2021, the number of people currently on this street will be all the footfall figures at 9 am every Friday in January 2020. We will call this the average footfall and this estimated figure. This figure is then divided by the street area and then multiplied by the users risk score. We then get an overall risk score for this user. If this final risk score is above equals to or more than 1, the street is not safe, else the street is safe. I came to this conclusion from multiple test cases of different users.h was an accurate representation of unsafety. Firstly, I decided a user's risk will go from 1-5. Then, I divided the street area by the average footfall which gave me a street density risk. Then I multiplied the street density risk by the users risk score from the health exam which gives me the users total risk.

Person risk	Street density risk	Total risk score
1	0.03	0.03
1	0.05	0.05
2	0.03	0.06
3	0.03	0.09
2	0.05	0.1
1	0.12	0.12
4	0.03	0.12
5	0.03	0.15
3	0.05	0.15
1	0.19	0.19
4	0.05	0.2
1	0.22	0.22
2	0.12	0.24
5	0.05	0.25
3	0.12	0.36
4	0.22	0.88
2	0.45	0.9
5	0.19	0.95
5	0.22	1.1
2	1.57	3.14
3	1.57	4.71
4	1.57	6.28
5	1.57	7.85

		Average footfall	area
0.05303030303	Nassau Street 9am	175	3300
0.1948484848	Nassau Street 2pm	643	3300
0.03484848485	Nassau Street 8pm	115	3300

Person risk	Street density risk	Total risk score
1	0.03	0.03
1	0.05	0.05
2	0.03	0.06
3	0.03	0.09
2	0.05	0.1
4	0.03	0.12
5	0.03	0.15
3	0.05	0.15
1	0.19	0.19
4	0.05	0.2
5	0.05	0.25
2	0.19	0.38
3	0.19	0.57
4	0.19	0.76
5	0.19	0.95

```
int thresh = ((length * width) / 2);
int area = length * width;
double streetRisk = ((average * 1.0) / area);
String uid = user.getUid();
DocumentReference docRef = firebaseFirestore.collection( collectionPath: "user").document(uid);
Map<String, Object> edited = new HashMap<>();
edited.put("street_area", String.valueOf(area));
edited.put("street_density", String.valueOf(streetRisk));
docRef.update(edited);
double finalRisk = streetRisk * Constants.USER_RISK_SCORE;
if (finalRisk >= 1) {
    safeStreet.setText("NOT SAFE" +
            "");
} else {
    safeStreet.setText("SAFE");
}
```

Shops, Restaurants and Supermarkets Safety Algorithm

The purpose of this algorithm is for the user to determine if a shop, restaurant or supermarket is safe or not safe.

Firstly, the user must be on the maps activity and search their desired location on the autosuggestions search bar. A multitude of places will come up as I am using a places API. Once they search their desired location, SAFIO will determine whether there is a popular time api

associated with this location. Like so

This information is not available and Google has not made this api live for developers to use. I had to create this API myself from the information given to me. Thus, I was forced to design an algorithm which copied the functions of the popular time api. By using the technique of web scraping, I was able to retrieve the data for these popular times of the location that the user has searched.

```
RequestQueue requestQueue = Volley.newRequestQueue( context: NewMainActivity.this);
StringRequest stringRequest = new StringRequest(Request.Method.POST,  url: "https://us-central1-fir-registerlogin-14dba.cloudfunctions.net/getPo
    @Override
    public void onResponse(String response) {
        Log.d( tag: "Data: ", response);
        String[] res = response.split( regex: "//");

        if (res[1].equals("No Popular Time")) {

        } else {
            new AlertDialog.Builder( context: NewMainActivity.this, R.style.CustomAlertDialog)
                    .setMessage("Live Status : " + res[1] + "\n\nSafety : " + res[0])
                    .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            dialogInterface.cancel();
                        }
                    })
                    .show();
        }

          Toast.makeText(getApplicationContext(), response, Toast.LENGTH_LONG).show();
```

I have then hosted a getPopularTimes algorithm in the google clouds functions which this code above calls. Upon studying the data available from the Popular Times API by web scraping it, I have been able to find out how the popular times determine how popular a certain place is. It is done numerically.

- 0-20 = Not Busy
- 21-50 = Not too busy
- 51-80 = A little Busy
- 81-100 = Busy

For my algorithm, if the getPopularTimes return 51 or above, this establishment is determined as not safe, otherwise it is safe.

```
try:
    wait_time_array = r['popular_times'][int(day)][1]

    if(r['current_popularity']!='None'):
        if(r['current_popularity']>50):
            for wait in wait_time_array:
                hour_wait = wait[0]
                if (hour_wait == int(hour)):
                    # print("Not Safe//" + wait[2])
                    if(r['current_popularity']>=81):
                        return "Not Safe//Busy"
                    else:
                        return "Not Safe//A little Busy"
        else:
            for wait in wait_time_array:
                hour_wait = wait[0]
                if (hour_wait == int(hour)):
                    # print("Safe//" + wait[2])
                    if (r['current_popularity'] >= 21):
                        return "Safe//Not too busy"
                    else:
                        return "Safe//Not Busy"


    for wait in wait_time_array:
        hour_wait = wait[0]
        if(hour_wait==int(hour)):
            # print(wait)
            if(wait[2]==""):
                return "Not safe//"+wait[2]
            if(wait[2]=="Usually not too busy"):
                return "Safe//"+wait[2]
            if(wait[2]=="Usually not busy"):
                return "Safe//"+wait[2]
            return "Not safe//"+wait[2]
except:
    return "Not safe//No Popular Time"
```

Dublin Bikes Safety Algorithm

The purpose of the Dublin Bikes algorithm is to determine the safety of an Dublin station for when a user would like to avail of their service.

Using the Dublin Bikes API which is given, I retrieved the location for each station and implemented them on my maps activity as markers

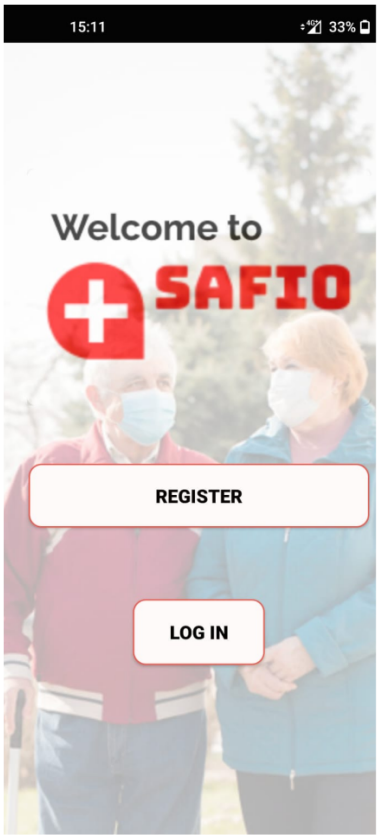
```
private void loadDataToMap() {
    Log.e(TAG,  msg: "loadDataToMap: ");
    final ProgressDialog progressDialog = new ProgressDialog( context: NewMainActivity.this);
    progressDialog.setCancelable(false);
    progressDialog.setMessage("Getting locations...");
    progressDialog.show();
    Call<LocationResponse> apiForRecord = RestClient.getInstance().getApiServices().requestForBikeLocations(Constants.BIKE_PLACES_UR
    apiForRecord.enqueue(new Callback<LocationResponse>() {
        @Override
        public void onResponse(Call<LocationResponse> call, retrofit2.Response<LocationResponse> response) {
            progressDialog.dismiss();
            if (response.body().getResult().getRecords() == null)
                return;
            bikesPlaces.clear();
            bikesPlaces.addAll(response.body().getResult().getRecords());
            LatLng latLng = null;
            for (Record a : bikesPlaces) {
                double lat;
                double lon;
                lat = Double.parseDouble(a.getLatitude());
                lon = Double.parseDouble(a.getLongitude());

                latLng = new LatLng(lat, lon);
                getMarkerOptions(latLng, a);
            }
```

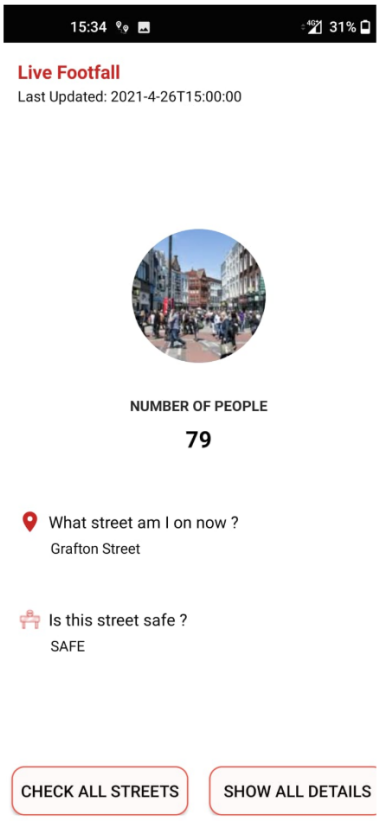
All a user must is press their desired location and it will show if it is safe or not. My safety Dublin bikes safety solves if there is a drop in the available bikes from one update to another, then I am imagining there are x amount of people at this Dublin Bikes location. For example Joe is at the Dublin Station at 2:01pm, at 1:55pm there are 10 bikes available, but the latest update which is 2:00pm says there are 8 bikes available. Because of this, I am imagining that there are 2 people currently at the Dublin Bikes station right now for Joe.

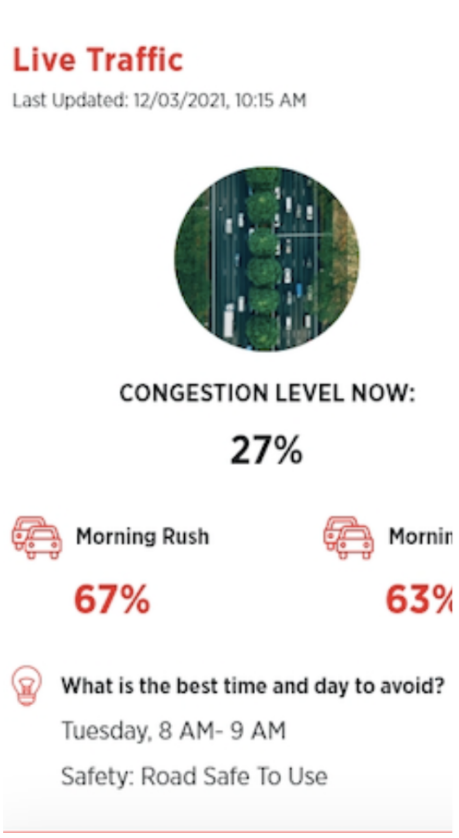
```
@Override
public void onResponse(Call<List<LocationDetailResponse>> call, retrofit2.Response<List<LocationDetailResponse>> response) {
    progressDialog.dismiss();
    Log.e(TAG,  msg: "onResponse: " + response.body());
    if (response.body().size() == 0 || response.body().get(0).getHistoric() == null)
        return;

    if (response.body().get(0).getHistoric().size() >= 2) {
        List<Historic> historics = response.body().get(0).getHistoric();
        int difference = historics.get(historics.size() - 1).getAvailableBikes() - historics.get(historics.size() - 2).getAvailableBikes(
        Log.e(TAG,  msg: "onResponse: last:" + historics.get(historics.size() - 1).getAvailableBikes());
        Log.e(TAG,  msg: "onResponse: Slast:" + historics.get(historics.size() - 2).getAvailableBikes());
        Log.e(TAG,  msg: "onResponse: adding marker");
        if (difference >= 2) {
            //not safe
            MarkerOptions place2 = new MarkerOptions().position(latLng).title(a.getName()).snippet("Station : " + a.getName() + "\n" +
                    "Available Bikes : " + historics.get(historics.size() - 1).getAvailableBikes() + "\n" +
                    "Safety : Not safe");
            mMap.addMarker(place2);

        } else {
            //safe
            MarkerOptions place2 = new MarkerOptions().position(latLng).title(a.getName()).snippet("Station : " + a.getName() + "\n" +
                    "Available Bikes : " + historics.get(historics.size() - 1).getAvailableBikes() + "\n" +
                    "Safety : safe");
            mMap.addMarker(place2);
```

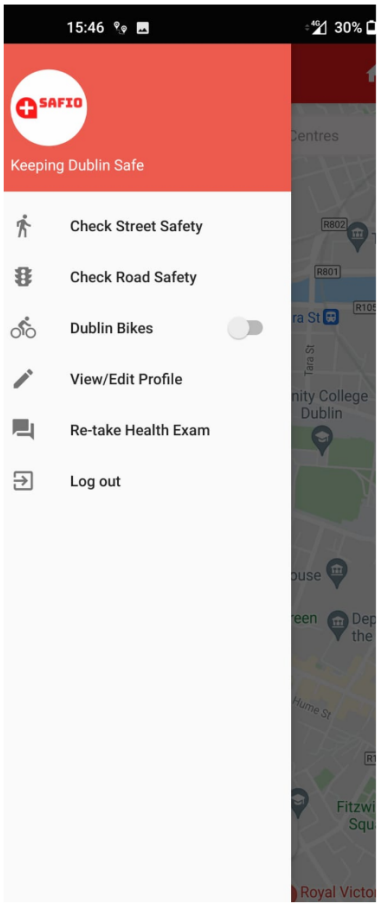
My algorithm then determines that if there are more than two people at the station, the station will be considered not safe, otherwise it will be safe.

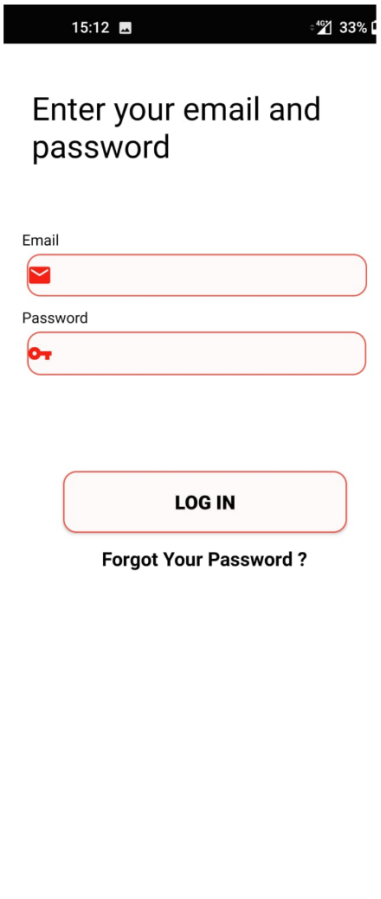
User Design Interface

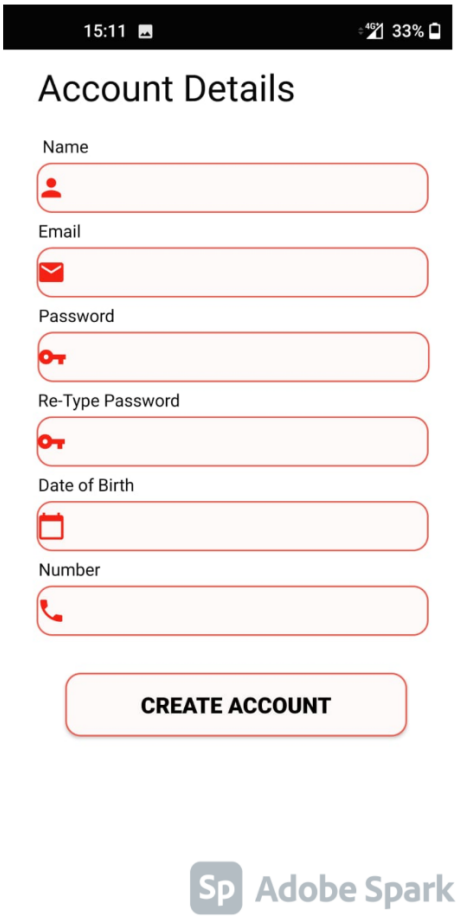


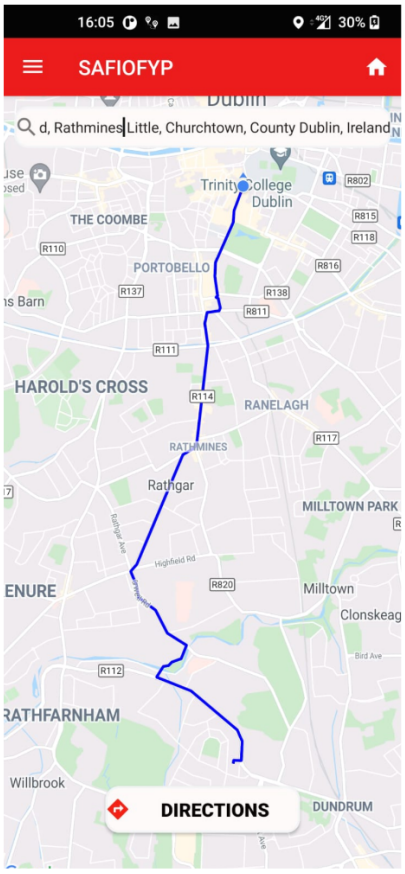


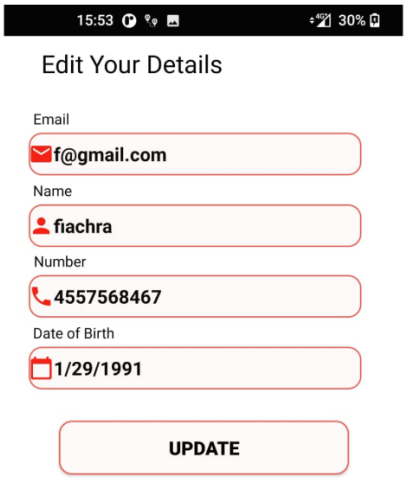




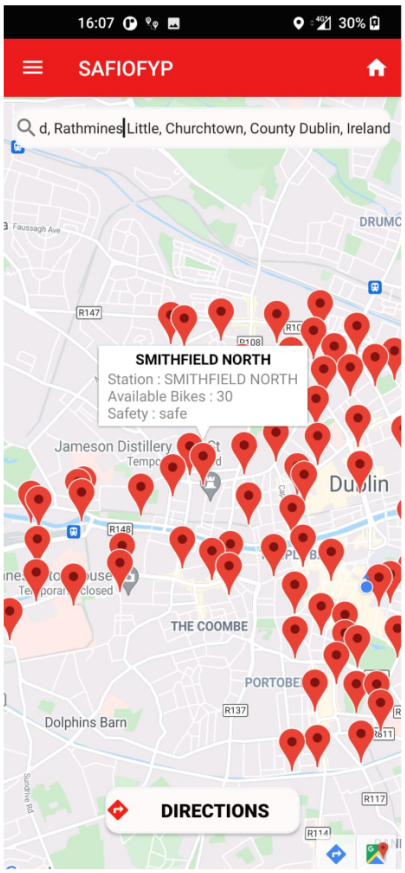


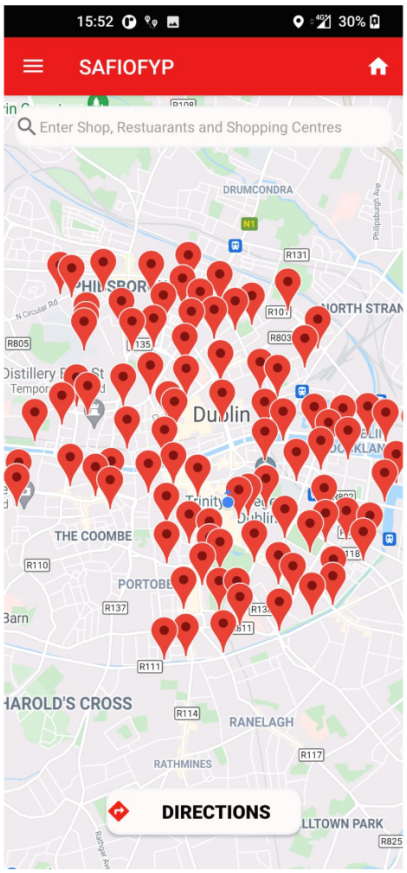


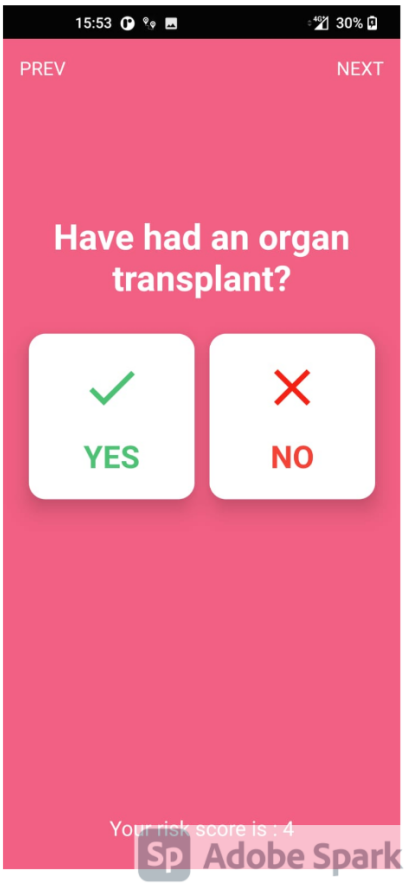

16:05
30%
SAFIOFYP
d, Rathmines Little, Churchtown, County Dublin, Ireland
Trinity College Dublin
THE COOMBE
PORTOBELLO
HAROLD'S CROSS
RANELAGH
RATHMINES
Rathgar
MILLTOWN PARK
Milltown
Clonskeagh
RATHFARNHAM
Willbrook
DUNDRUM
DIRECTIONS


15:53
30%
Edit Your Details
Email
f@gmail.com
Name
fiachra
Number
4557568467
Date of Birth
1/29/1991
UPDATE


17:16
23%
ALL STREETS DATA
O'Connell Street Lower SAFE
O'Connell Street Upper SAFE
Mary Street SAFE
Nassau Street SAFE
Capel Street SAFE
Aston Quay SAFE
D'Olier Street SAFE
Dawson Street SAFE
Dame Street SAFE
Talbot Street SAFE
Parnell Street SAFE
Suffolk Street SAFE
College Green SAFE
Henry Street SAFE
Westmoreland Street SAFE
Liffey Street SAFE
Bachelors Walk SAFE


16:07
30%
SAFIOFYP
d, Rathmines Little, Churchtown, County Dublin, Ireland
SMITHFIELD NORTH
Station : SMITHFIELD NORTH
Available Bikes : 30
Safety : safe
Jameson Distillery
Dublin
THE COOMBE
Dolphins Barn
DIRECTIONS


15:52
30%
SAFIOFYP
Enter Shop, Restuarants and Shopping Centres
DRUMCONDRA
Dublin
THE COOMBE
HAROLD'S CROSS
RANELAGH
RATHMINES
DIRECTIONS

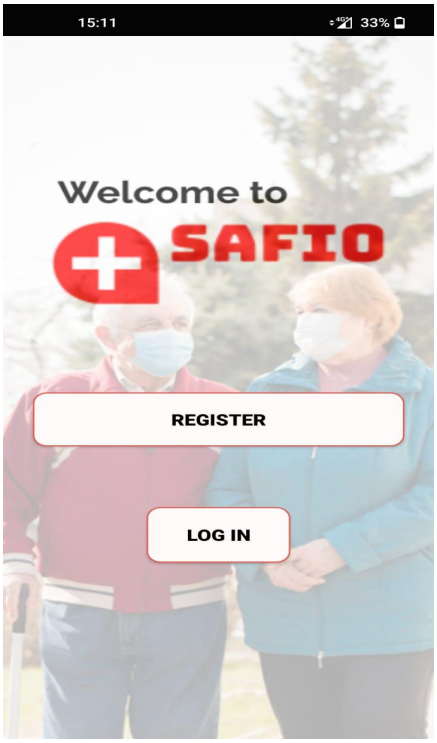

15:53
30%
PREV
NEXT
Have had an organ transplant?
YES
NO
Your risk score is : 4
Adobe Spark

Implementation

Overview of Implementation

Home Activity

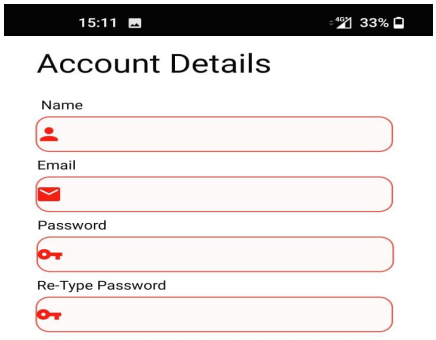
The user is first introduced to the home screen which has sliding background images which are hosted on firebase. An intent is used to bring the user from the homeactivity for them to either login or register. All the user must do is simply choose to register or login. If the user is already logged in they will skip this page when they open the app. Also, the images in this activity are a slideshow and this was done by using sliders and hosting the images on firebase.




```
btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            startActivity(new Intent( packageContext: HomeActivity.this, NewMainActivity.class));
        } else {
            startActivity(new Intent( packageContext: HomeActivity.this, LoginActivity.class));
        }
    }
});

btnRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intRegister = new Intent( packageContext: HomeActivity.this, RegisterActivity.class);
        startActivity(intRegister);
    }
});
}
```

Register Activity

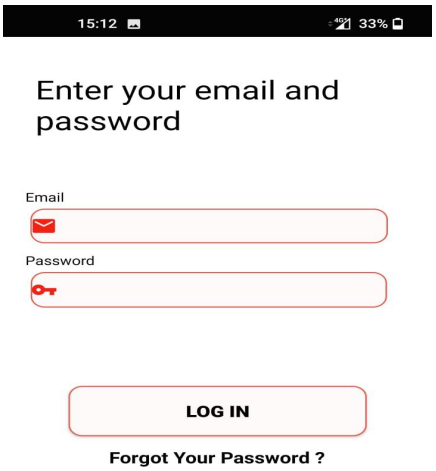
The user is then brought to the register page where they can register as a new user. These details are stored in the firebase database. Using the authentication function which is given, each user is added so that when they wish to login, their details are saved and stored.




```
Toast.makeText( context: RegisterActivity.this, text: "User Created.
userID = fAuth.getCurrentUser().getUid();
DocumentReference documentReference = fStore.collection( collectionP
User user = new User();
user.setUsername(fullName);
user.setEmail(email);
user.setNumber(phone);
user.setDate(dob);
user.setUser_id(FirebaseAuth.getInstance().getUid());
```

Login Activity

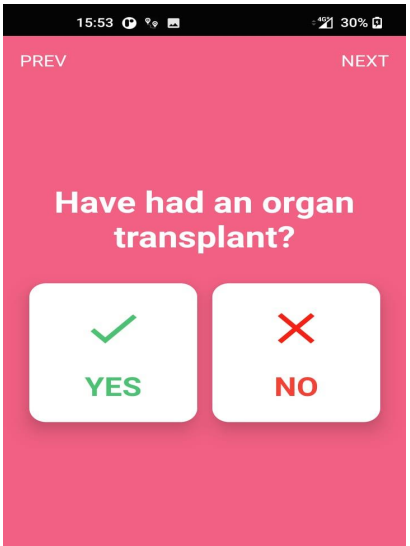
Once a user has registered they can now login. They are able to login because of the authentication functions in firebase. Once the user logs in, they are then brought to the health exam page to answer a multitude of questions on their health.




```
fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new O
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            setRiskScore();

        }else {
            Toast.makeText( context: LoginActivity.this, text: "Error ! " + ta
        }

    }
});
```

Health Exam

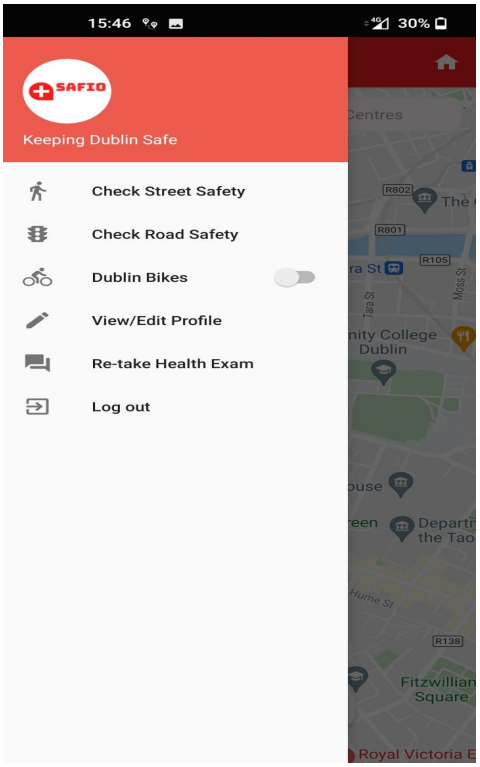
In the health exam page, users must answer a multitude of questions so that the app can calculate their health score and get a more accurate safe or not safe. Their score is shown at the bottom. Each question answered yes has a +score to go with it. A users health score is then saved within the firebase system for when they want to retake the health exam. Their health score is capped at 5.




```
public void addPoints(int points, int fragmentNumber) {
    switch (fragmentNumber)
    {
        case 1:
            if (fragment1 == 0)
            {
                fragment1 = 1;
                totalRiskScore += 2;

            }
            break;
        case 2:
            if (fragment2 == 0)
            {
                fragment2 = 1;
                totalRiskScore += 2;

            }
            break;
        case 3:
```

Navigation Menu

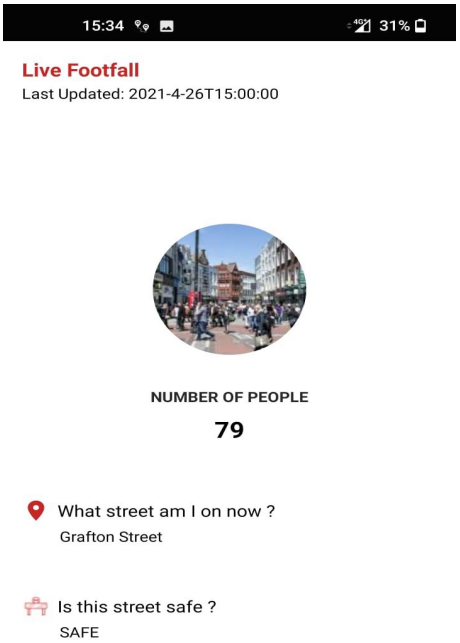
From the navigation menu users will be able to check road safety, check road safety, toggle to see Dublin Bikes stations on map to check their safety, View & Edit profile info, re take their health exam and log out.




```
navigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.edit_profile) {
            Intent intent = new Intent( packageContext: NewMainActivity.this, EditProfile.class);
            startActivity(intent);
        } else {
            if (id == R.id.check_location) {
                Intent intent = new Intent( packageContext: NewMainActivity.this, MyLocationActivity.class
                startActivity(intent);
            } else if (id == R.id.checkTrafic) {
                Intent intent = new Intent( packageContext: NewMainActivity.this, TraficIndexActivity.clas
                startActivity(intent);
            } else if (id == R.id.questionnaire) {
                Intent intent = new Intent( packageContext: NewMainActivity.this, QuestionsActivity.class
                startActivity(intent);
            } else if (id == R.id.logout) {
                Intent intent = new Intent( packageContext: NewMainActivity.this, HomeActivity.class);
                FirebaseAuth.getInstance().signOut();
                startActivity(intent);
                Toast.makeText( context: NewMainActivity.this,  text: "You Have Successfully Signed Out", T
```

Street Safety

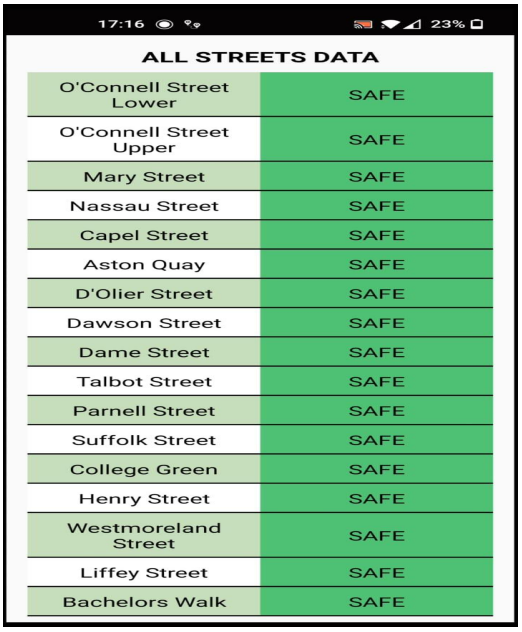
A user can check the safety on the street they are currently on. This function will tell them whether this street is safe or not along with some other key information.




```
int thresh = ((length * width) / 2);
int area = length * width;
double streetRisk = ((average * 1.0) / area);
String uid = user.getUid();
DocumentReference docRef = firebaseFirestore.collection( collectionP
Map<String, Object> edited = new HashMap<>();
edited.put("street_area", String.valueOf(area));
edited.put("street_density", String.valueOf(streetRisk));
docRef.update(edited);
double finalRisk = streetRisk * Constants.USER_RISK_SCORE;
if (finalRisk >= 1) {
    safeStreet.setText("NOT SAFE" +
            "");
} else {
    safeStreet.setText("SAFE");
```

All Streets Safety

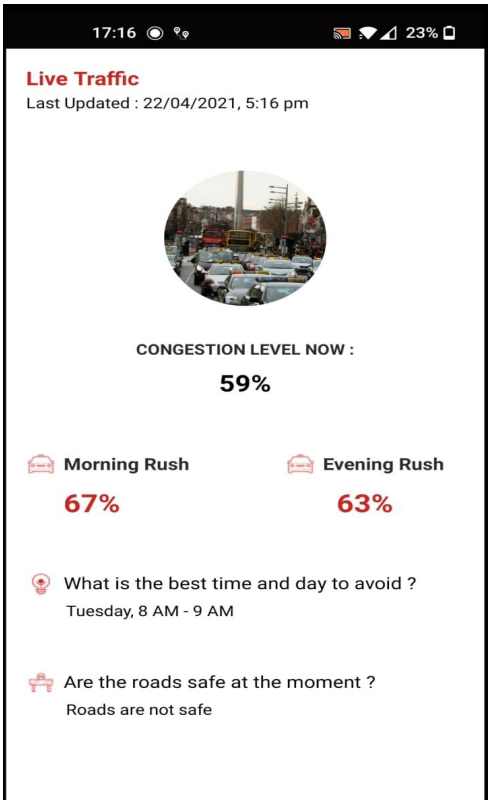
Users can also check all the streets that are available to see how safe they are in a recycler view. Green represents that it is safe, red represents that it is not safe. This activity uses a similar algorithm as Check Street safety.




```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_view"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

Road Safety

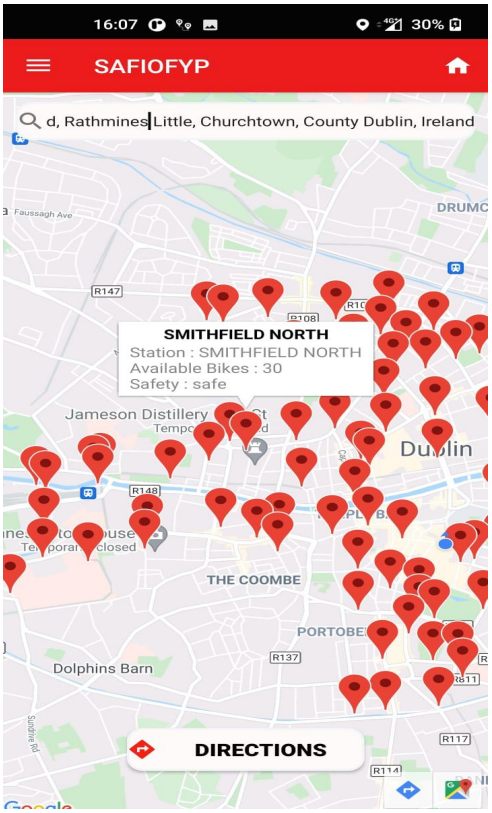
Users can check the safety of the Road in Dublin. This will help them avoid the chances of an accident. I used the technique of web scraping to retrieve the data from TomTom Dublin Traffic. If Dublin congestion level is above 30 percent, the roads are considered not safe.




```
mainActivity.progressBar1.setVisibility(View.GONE);
mainActivity.scrollView_root.setVisibility(View.VISIBLE);
// Stuff that updates the UI
mainActivity.last_update_txtView.setText(last_update.replace
mainActivity.congestion_level.setText(cong_live);
mainActivity.morning_rush.setText(morning_rush);
mainActivity.evening_rush.setText(evening_rush);
mainActivity.time_to_avoid.setText(time_avoid);


int cong_val = Integer.parseInt(cong_live.replace( target: "%"
if(cong_val >= 30){
    mainActivity.safe_or_not.setText("Roads are not safe");
}
else{
    mainActivity.safe_or_not.setText("Roads are safe");
}
```

Dublin Bikes Safety

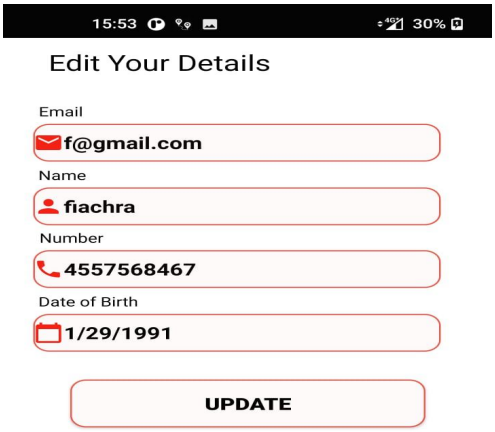
Users can also check the safety of Dublin Bikes stations. Using the Dublin bikes api I have taken in all the locations latitude and longitudes and marked them on my maps activity. Red depicts that it is not safe and green determines that it is safe. The user can also press the market to get more info on the station.




```
progressDialog.dismiss();
if (response.body().getResult().getRecords() == null)
    return;
bikesPlaces.clear();
bikesPlaces.addAll(response.body().getResult().getRecords());
LatLng latLng = null;
for (Record a : bikesPlaces) {
    double lat;
    double lon;
    lat = Double.parseDouble(a.getLatitude());
    lon = Double.parseDouble(a.getLongitude());

    latLng = new LatLng(lat, lon);
    getMarkerOptions(latLng, a);
}
```

View/Edit Profile

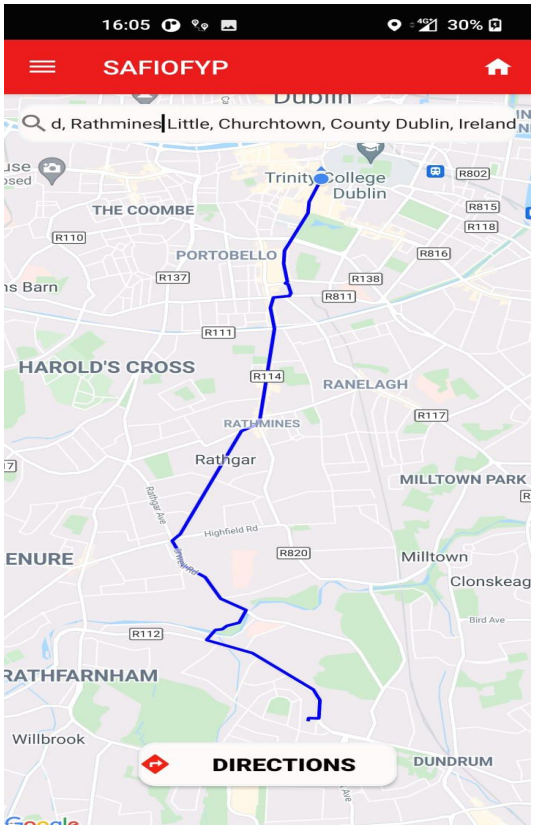
Users have the opportunity to View/Edit their personal information.




```
final DocumentReference documentReference = fStore.collection( collectionPath: "user").doc
documentReference.addSnapshotListener( activity: this, new EventListener<DocumentSnapshot>
    @Override
    public void onEvent(@Nullable DocumentSnapshot documentSnapshot, @Nullable Firebase

        profileFullName.setText(documentSnapshot.getString( field: "name"));
        profileEmail.setText(documentSnapshot.getString( field: "email"));
        profilePhone.setText(documentSnapshot.getString( field: "number"));
        mDisplayDate.setText(documentSnapshot.getString( field: "date"));
    }
});
```

Google Directions API

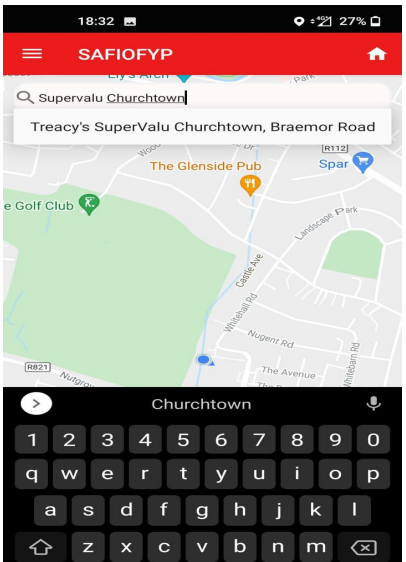
Using the directions API allows a user to get directions to their desired destination. Much like Google Maps, users must search a location and then press directions. The directions API is available on the google console. All that must be done is to enable the google cloud functions. I chose the preferred mode of transport as walking to depict a normal user's mode of transport.




```
private String getUrl(LatLng origin, LatLng dest, String directionMode) {
    // Origin of route
    String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
    // Destination of route
    String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
    // Mode
    String mode = "mode=" + directionMode;
    // Building the parameters to the web service
    String parameters = str_origin + "&" + str_dest + "&" + mode;
    // Output format
    String output = "json";
    // Building the url to the web service
    return "https://maps.googleapis.com/maps/api/directions/" + output + "?" + parameters + "&key=" +
}
```

Google Places API

The google places API is used when a user wants to go to their desired location. Once they start to search for their location, the places api will recommend the nearest and best matched location depending on the users location and key words. Once a user chooses a location, the safety of this establishment, which must be a restaurant, bar or shop.




```
String location = autoCompleteTextView.getText().toString();
List<Address> addressList = null;

PlaceAutoSuggestAdapter placeAutoSuggestAdapter = (PlaceAutoSuggestAdapter) paren
  placeAutoSuggestAdapter.placeDetailsData.get(position).getPlace_id();
Log.d( tag: "Place_id", placeAutoSuggestAdapter.getPlaceDetailsData().get(position
Geocoder geocoder = new Geocoder( context: NewMainActivity.this);
try {
    addressList = geocoder.getFromLocationName(location, maxResults: 1);

} catch (IOException e) {
    e.printStackTrace();
}
```

Issues and Resolutions with Implementation

Overall, I was fortunate enough to not run into many huge issues throughout the creation of my project. I was always on top of my works and always had the functions working for each of the checkpoints. The only issues I ran into were the normal issues that are faced when coding any application. The issues mostly happened when dealing with my two main algorithms. As these were the most difficult part of my project and the part in which I spent the most time working I often ran into small issues and thanks to increased proficiency in problem solving, I was eventually able to avoid the issue.

Smart Dublin API

The first issue I ran into was when I started working with the Smart Dublin API. After completing my register login this was the first function I wanted to get working so that it would show that my application is possible. It was my first time working with an api and retrieving data from it. I spent weeks trying to retrieve this data but to no avail as I was extremely inexperienced with this, but through continuous efforts and multiple youtube tutorials I was able to start retrieving the data that I needed to work my Street Safety Algorithm. The solution I came up with was to use a rest client to retrieve the data using the given api. I was incredibly proud of this algorithm and it was for me my greatest accomplishment in college

Popular Times Safety

The major issue I ran into when creating the popular times safety function was initially, I thought the API was live and that I was able to use it like the google maps api and places api that I am. Unfortunately this was not the case and I was forced to come up with a solution to use this data. My thinking, if it's there I can use but to match the users desired location with the popular time of that location was very difficult. Originally I tried to do it through android studio but to no avail. Upon continuous readings online I discovered the use of the google console to host your own API functions and this is what I did. It was recommended that I use python for this and python is not a language that I am comfortable with at all. Thus, I spent 2-3 weeks going over the problems and exercises that I did with Dynamic Programming in 3rd to get my mind thinking in python so I could solve this problem.

Test Plan

Name	Description	Expected Result	Result
Register User	Enter User Details Press Register	User Added to Database User set logged in Login Activity Opens	PASS
Login User	Enter user details Press login	User logged in If health exam already take: Main Activity Opens If health exam not already taken: Health Exam Opens	PASS
Take Health Exam	User presses yes or no to health exam questions	Users health score calculated Brought to main activity	PASS
Update User Details	Edit Details Press Edit	Users details Updated in Firebase	PASS
Check Street Safety	Open Nav Menu Press Check Safety	Users find out if their location is safe or not	PASS
Check All Streets	Press Check All streets	Streets safety for all streets in database appears	PASS
Check Road Safety	Open Nav Menu Check Road Safety button	Shows information on Dublin traffic and determines if Roads are safe to drive on	PASS
Toggle On Dublin Bikes Stations	Open Nav Menu Turn on Dublin Bikes locations	All Dublin Bikes Locations appear on the maps activity	PASS

Toggle Off Dublin Bikes Stations	Open Nav Menu Turn off Dublin bikes locations	All Dublin Bikes Locations are removed from the maps	PASS
Search Locations	Press On Search Bar Enter in desired location Press Desired Location	Autosuggestions comes up and user chooses a location	PASS
Check Establishment Location	User Presses location on search auto suggests	Business and how safe the establishment is comes up	PASS
Get Directions	Users chooses location from auto suggestions Presses Directions	Walking direction to desired location from users location	PASS
Home	Press Home button from main activity	User is brought back to home activity	PASS
Log Out	Open Nav Menu Press Log out	User is logged out of the app	PASS

Conclusions

Summary of Project

Overall, I was happy with how the project went. Especially during a pandemic and considering all the external factors, I felt I gave this project my absolute all. I am happy with the progress I made in the project from day one. I had formulated my idea officially in October of 2020 and to be here 7 months later and seeing the progress I have made is something I am very proud of. Initially, I had envisioned just having a safety function working for the streets of Dublin and my project being solely based on that and any other functions would have been a bonus. To have 4 functions that informs the user whether they are safe or not is all down to my continuous work rate and simply solving a problem each week, not matter how small.
The project has also given me the confidence in my abilities to be a software programmer and that's something I didn't think I would ever say. I believe my coding ability and skills have progressed leaps and bounds, I have gone from being very hesitant and having a negative attitude towards coding to really enjoying it. I further have enhanced my problem-solving skills and business analytical skills and these are skills I hope to bring into my Commercial Management role at BT.

Final Conclusions

Finally, I think this is an app that I can definitely develop further to serve a purpose other than a pandemic. Obviously, the purpose of the app is to ensure people stay safe and socially distant but society will change now. We will be more weary of diseases and sickness from now on. The flu has killed more people over the last year than covid-19 so I feel there is a gap in the market for an app like SAFIO. In conclusion, I have complete confidence in my idea that I will be applying for the European Innovation Academy with SAFIO.