T.A.P.

Transport Accessibility Platform

Shay de Barra

x16115864

x16115864@student.ncirl.ie

Higher Diploma in Science in Computing

Mobile Application Specialisation

03/04/2018

Table of Contents

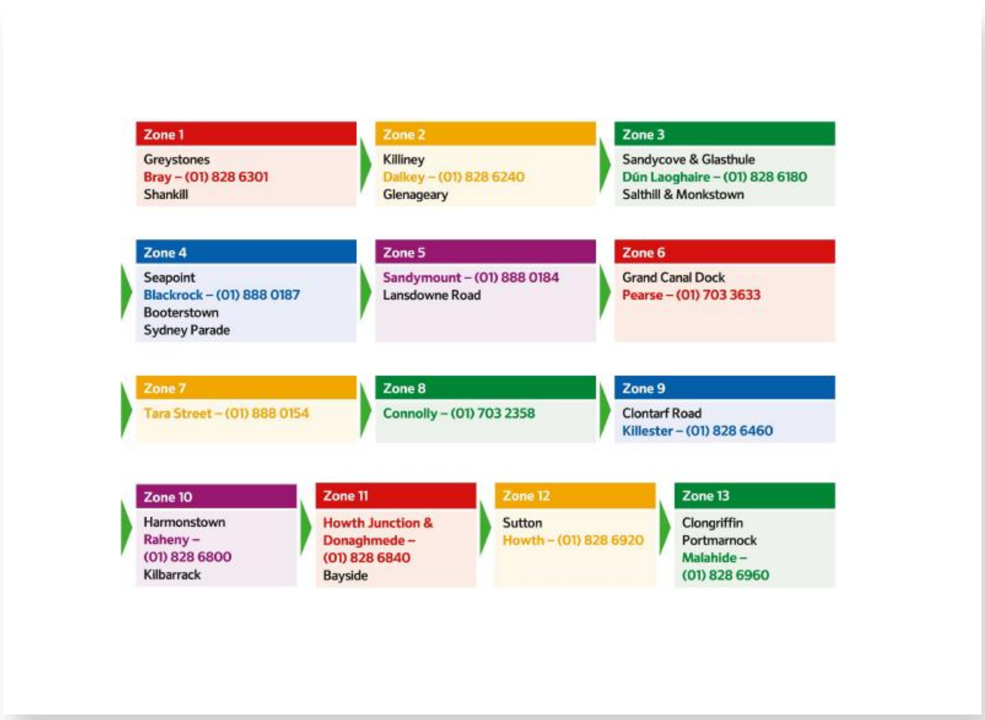
1 Introduction

The Problem

Disability advocacy groups have been protesting for many years with regard to availing of public service transportation without any special requirements. Rail commuters with disabilities, specifically wheelchair users, require assistance from Irish Rail staff to embark and disembark Dart train services. This assistance involves the provision of a portable ramp.

In the recent past, members of the public requiring wheelchair access while travelling with Irish Rail have had to give prior notice of a minimum of twenty-four hours' notice.

Due to the protestations of many advocacy groups, Irish Rail introduced the new *DART Improved Accessibility Pilot scheme*. This scheme requires users to provide an advanced notice provision of four hours prior to travelling. The scheme is further complicated by the addition of zoned areas which are staffed specifically for customers requiring assistance and a contact phone number.



Zone 1	Greystones Bray – (01) 828 6301 Shankill
Zone 2	Killiney Dalkey – (01) 828 6240 Glenageary
Zone 3	Sandycove & Glasthule Dún Laoghaire – (01) 828 6180 Salthill & Monkstown
Zone 4	Seapoint Blackrock – (01) 888 0187 Booterstown Sydney Parade
Zone 5	Sandymount – (01) 888 0184 Lansdowne Road
Zone 6	Grand Canal Dock Pearse – (01) 703 3633
Zone 7	Tara Street – (01) 888 0154
Zone 8	Connolly – (01) 703 2358
Zone 9	Clontarf Road Killester – (01) 828 6460
Zone 10	Harmonstown Raheny – (01) 828 6800 Kilbarrack
Zone 11	Howth Junction & Donaghmede – (01) 828 6840 Bayside
Zone 12	Sutton Howth – (01) 828 6920
Zone 13	Clongriffin Portmarnock Malahide – (01) 828 6960

Figure 1

“Once guided to the relevant Zone, call the number provided for the Zone (see Station Contact Hours) and inform the staff member of your travel details as recommended, four hours in advance. Then on arrival at your departure station, make yourself known to staff and they will ensure your safe passage onto the train and arrival at your destination station.” (Rail, 2018)

This scheme attempts to address the issue at hand without any consultation from the affected travelling public.

Rail users with disabilities (hereinafter referred to as the client) seek to access public transportation in the same manner as the general travelling public.

1.1 Aims

The Solution

“Picking up the phone at five in the morning so that I can get to work is not a workable solution … anyway, the phones aren’t staffed at five in the morning” (O'Kelly, 2018).

The solution to the problem is one of clear communication between customers and staff and it is the opinion of the author that the implementation of this software solution is of a cost-benefit to both parties.

This application addresses this problem through the implementation of a mobile software application. The scope of the solution is specific to the Dart rail services as a proof of concept. In order to achieve this, the problem of communication must be clearly defined.

1. The Client needs to know if a station is manned <u>prior</u> to traveling or disembarking.
2. The Client should have full knowledge of station staffing in real time.
3. If a station is manned, the client should be able to communicate with staff members upon arrival and or when ready to disembark.
4. Station staff need to be made aware of the clients impending arrival in order to prepare for their needs.
5. Station employees should be able to contact clients to make them aware of any delays or impediments.

Mobile devices equipped with location services along with the ability to send notifications and communicate with a third-party server can solve this issue in real-time.

1.2 Background

While researching for a project to submit for the completion of this course, the author happened upon a radio interview with a member of an advocacy group in debate with an Irish Rail representative.

The solution *(as a completed idea)* was clear and immediate to the author and began researching possible developmental avenues that could represent closely the finished idea.

During this time the author contacted the advocacy group representative, Sean O Kelly and the consultation process began.

There were no other applications on the market of a similar nature and to date the author has failed to find any example of such. This is possibly due to the bespoke nature of the solution required and whilst advantageous on one hand, there were no other applications from which inspiration could be drawn upon.

1.3 Technologies

Android Studio 3.1.1 was used in the development of this project. Despite assurances that the current release is a stable release, the developer experienced consistent problems especially with regard to debugging the applications.

The main debugger *(Logcat)* on a regular basis froze or simply did not give an output of the application under test. Despite spending a considerable amount of time researching a solution to the problem, to date, the problem for the developer and others (members of Stack Overflow) remains unresolved. (Stackoverflow.com, 2018)

The A.D.B. *(Android Debug Bridge)* also proved problematic particularly when running an emulator with API 26+ *Oreo.* The build system took considerably longer to build each test iteration *(several minutes and sometimes up to five minutes)* and the emulator behaved in a bloated fashion. Eventually it was decided to test on an emulator running API 25 which proved much more efficient and life-like.

Testing was also done via USB on an Android phone running API 25 *Nougat*, and due to the nature of the application under development, it was often in combination with the ADB and monitor displaying a web view of the firebase database.

For practical reasons the ADB was later replaced with a much faster emulator *Genymotion*. (Genymotion – Android Emulator for app testing, 2018)

This was done for two primary reasons...

1. Speed – The emulator reduced build times considerably
2. Realism – the *Genymotion* emulator processed code in a manner closer to a real device. For example, the method checking for an internet connection (ping test) worked flawlessly on the *Genymotion*, where it failed on the ADB supplied with Android Studio.

2 System

2.1 Requirements

2.1.1 Purpose

This document outlines the requirements to develop of a software solution that addresses the issue of rail transport accessibility for people with disabilities. It will demonstrate the issue at hand and the necessary requirements needed in order to complete a satisfactory solution.

The software solution will require two applications. An application for the intended user and an application for the *Irish Rail* employee.

The intended customers for the client application are primarily people with physical disabilities whose physical needs require the assistance of Irish rail members to board the train.

The intended customers for the staff application are Irish Rail employees who are rostered to man the required stations and assist customers.

2.1.2 Project Scope

This software solution is primarily targeted at the *Dart* rail service using the Android Studio Environment for development.

The scope of the project is to develop a system that will gather the required information on *Irish Rail Dart* real-time staffing arrangements, relay that information to the user and enable the user to send notifications to the relevant station employee.

A cloud-based database will facilitate the information exchange between both applications in real time.

The *Irish Rail* API will also facilitate the provision of live Dart rail movements and times to the system.

Project objectives

- The system will enable the user to know which stations are manned prior to travelling.
- The system will give up to date live feedback on employee movements to the user when travelling.
- Employees will be kept up to date on customer locations while travelling.
- Employees will be notified when customers are within the vicinity and in need of assistance.

Project restrictions

- The *Irish Rail* API is the only available resource provided by Irish Rail and is limited in its functionality.

- *Irish Rail* in an effort to address the problem has further complicated the issue by providing a division of Dart stations into zones. Thirteen zones represent the Dart stations where each zone will be staffed accordingly. This software solution must take the zone staffing into account when providing the solution. (Irishrail.ie, 2018)

2.1.3 User Requirements Definition

This application is designed primarily to meet the clients' needs.

A point to note: Users must have an internet connection to avail of this service. Keeping in mind that many users (the author included) are limited financially to the disability allowance scheme, this application makes allowances for this in keeping its internet use within design parameters.

Irish Rail's *DART Improved Accessibility Pilot* scheme has failed to meet any of the user's primary requirements as they were never consulted. The developer of this application is continuing to work with the users in gaining a satisfactory outcome.

The **clients'** prioritised needs are as follows.

1. Access rail services as other rail users
2. <u>Not</u> to have to give four hours' notice prior to travelling
3. Be made aware of what stations are manned in real-time
4. Have clear communication between Client and Staff to avail of assistance when needed
5. The interface should be clean and simple for ease of use. Allowances can to be made for people with manual dexterity issues. A separate interface can be developed in the future to cater for their requirements. The scope of this application is primarily for those without dexterity issues.

The **staff** requirement are as follows.

1. Staff members have the ability to sign in and sign out of the application.
2. Staff members can share a common mobile device if required.
3. Staff can change their selected place of work (station) with ease.

2.1.4 Client Functional Requirements

The Client Application

1 The user logs into the system

2 The user selects *"Select Journey"*

3 The user selects the origin and destination station

4 The user proceeds with the journey selection

Client Use Case Diagram

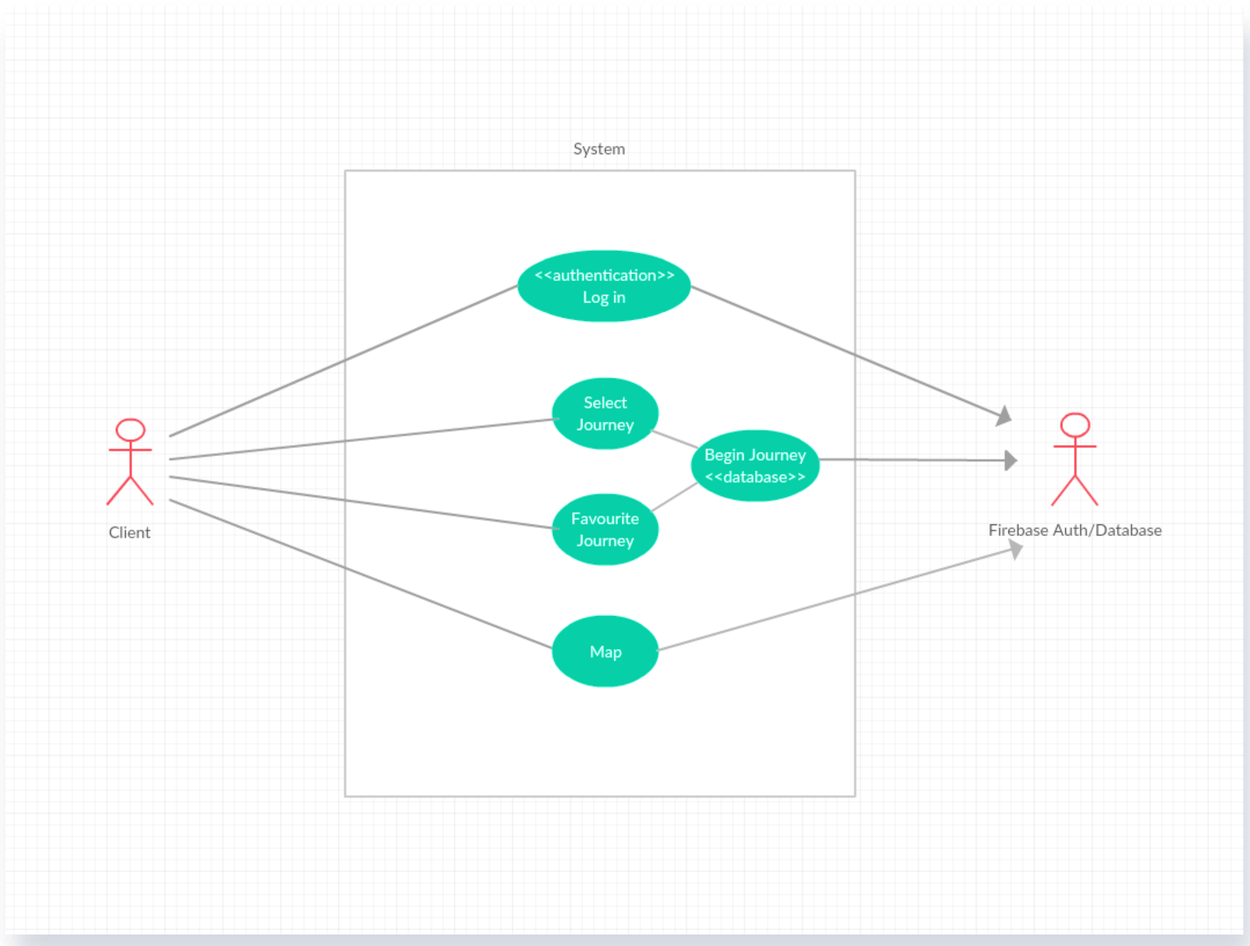


Figure 2

2.1.5 Requirement 1

Client Log in

2.1.6 Description & Priority

- The Client logs into the system.
- The client must sign in to the system to gain access to the firebase database and get an AUTH signature.

2.1.6.1 Use Case

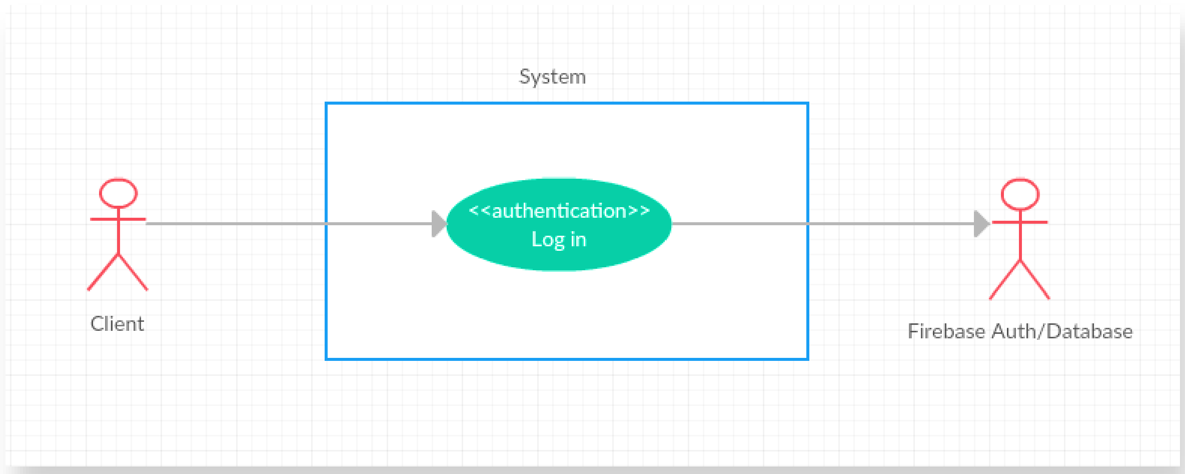


Figure 3

Scope

The scope of this use case is to log the user into the system using firebase AUTH.

Description

This use case describes the firebase AUTH login process

Flow Description

Precondition

The system has an internet connection

Activation

This use case starts when the user presses *Sign in*

Main flow

1. The system displays previously signed in personnel
2. The User selects their identity
3. The system retrieves the AUTH signature from Firebase
4. The system logs the selected user in

Alternate flow

A1: The User is a new user
1. The User clicks "add account"
2. The AUTH system accepts email and password
3. The use case continues at position 2

A2: The User is already signed in
1. The use case continues at position 3

Exceptional flow

E1: The User is not registered with Google
1. The User is unable to proceed

Termination

The system exits and the home screen is presented

Post condition

The system goes into a wait state

2.1.7 Requirement 2

Select Journey

2.1.7.1 Description & Priority

The user selects the *Select Journey* button to initiate the process. This is where the core of the main application processes begins.

2.1.7.2 Use Case Select Journey

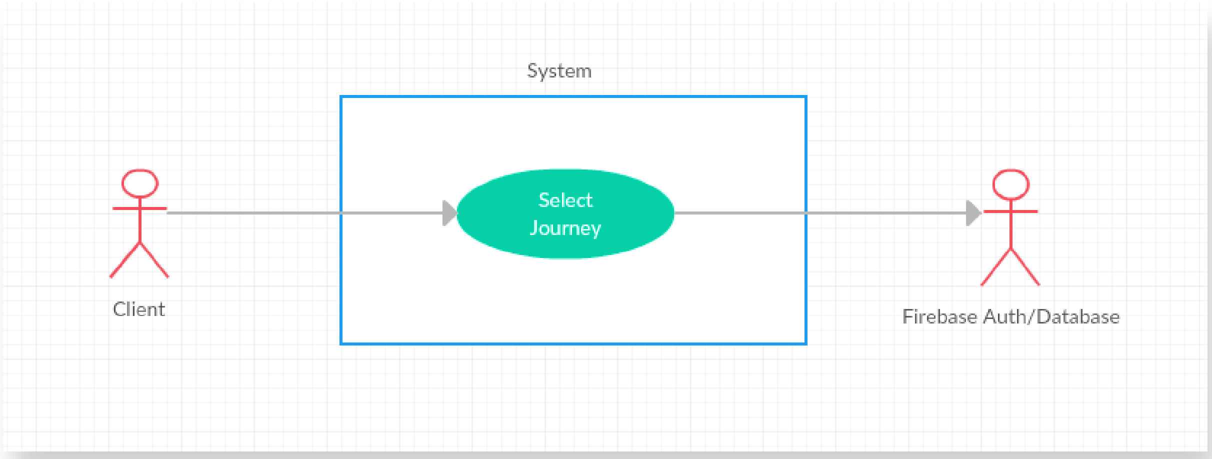


Figure 4

Scope

The scope of this use case is to begin the selection process.

Description

This use case describes the selection of *Select Journey*

Flow Description

Precondition

The user is logged in

Activation

This use case starts when the user presses the *Select Journey* button.

Main flow

1. The system identifies the signed in User and presents a list of stations.
2. The User selects the origin station.
3. The system records the origin station and displays it
4. The User selects the destination station
5. The system records the destination station and displays it
6. The User clicks the accept button to proceed

Alternate flow

A1: The User selects the back button

1. The system exits the use case

A2: The User does not press the accept button

1. The system goes into idle state and proceeds to main flow 1

Exceptional flow

E1: The User has made an invalid selection

1. The system displays a Toast message "Invalid selection"
2. The system clears the selected text and proceeds to main flow 1

Termination

The activity is closed, and the system presents the next Begin Journey activity

Post condition

The system begins the next activity

2.1.8 Requirement 3

Favorite Journey

2.1.8.1 Description & Priority

- The user is presented with a database list of favored stations to choose from.

2.1.8.2 Use Case Favorite Journey

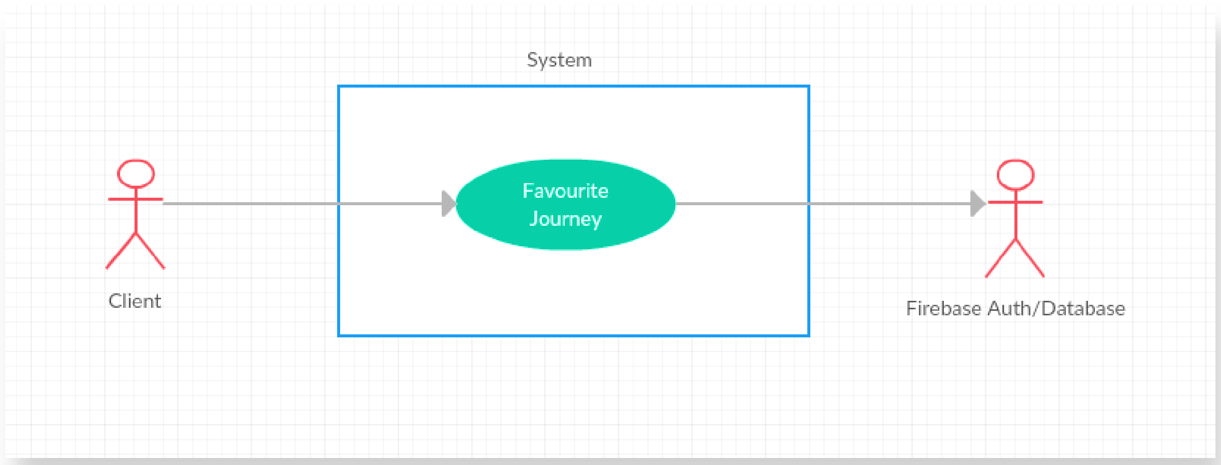


Figure 5

Scope

The scope of this use case is to complete the favorite journey selection process.

Description

This use case describes the process of selecting a previously favored journey.

Flow Description

Precondition

The user is signed in.

Activation

This use case starts when the user selects the favorite journey button.

Main flow

1. The system identifies the signed in User and presents a list of previously favored journeys.
2. The User selects a journey from the list.
3. The system records the selection and proceeds with the next activity.

Alternate flow

None

Exceptional flow

E1: The system does not have any stored journeys to display.

1. The user is presented with a blank list and the back button is set to visible.

Termination

The activity is closed, and the system presents the next activity

Post condition

The system begins the next activity

2.1.9 Requirement 4

Begin Journey

The Client logs into the system.

2.1.9.1 Description & Priority

The client has selected the origin and destination and will now proceed to initialise the selection.

2.1.9.2 Use Case Begin Journey

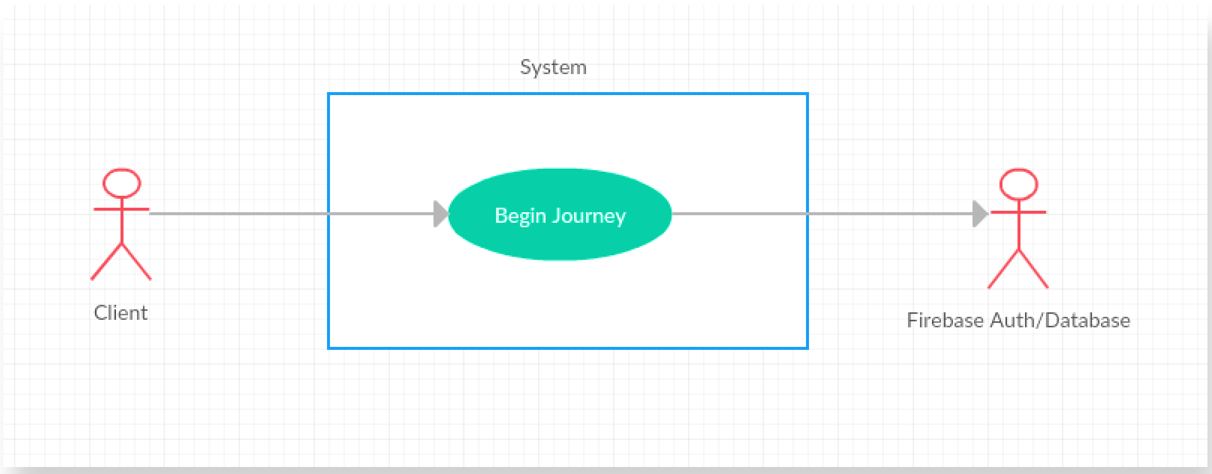


Figure 6

Scope

The scope of this use case is start the selected journey process

Description

This use case describes starting the journey process flow

Flow Description

Precondition

The system has an internet connection and the origin and destination selections have been made.

Activation

This use case starts when the *Favorite Journey* or *Select Journey* activities have been completed.

Main flow

1. The system starts a background location service
2. The system notifies firebase database of the system state
3. The system displays a dialog box to the User to travel to the origin station.
4. The system dismisses the UI

Alternate flow

None

Exceptional flow

E1: The background location service has been declined permissions

The User is unable to proceed unless they grant the permissions

1. The system displays the default permissions dialog
2. The User accepts the permissions and proceeds to main flow 3

Termination

The system exits, and the home screen is presented

2.1.10 Requirement 5

Map View

2.1.10.1 Description & Priority

The user is presented with a map view of the stations along with current live train movements.

2.1.10.2 Use Case Map View

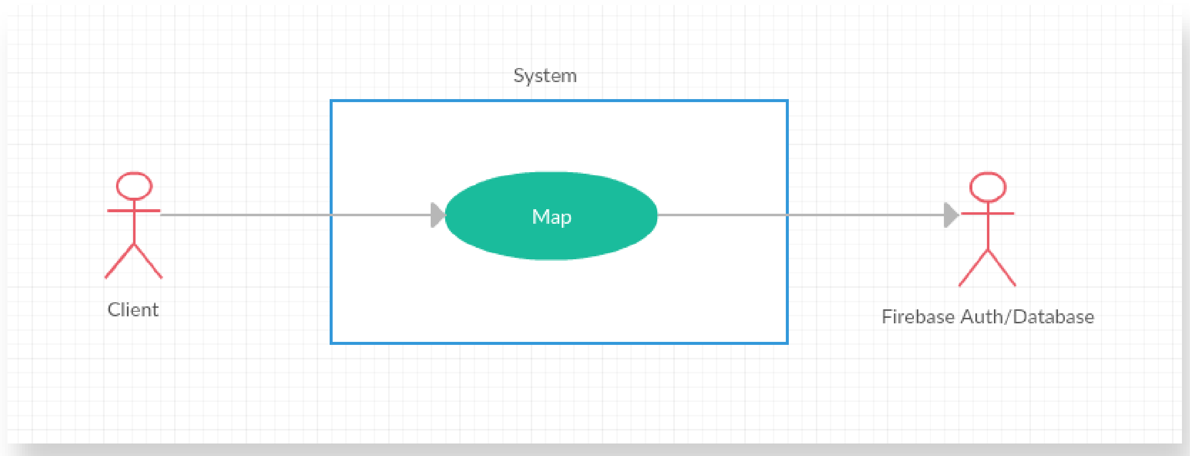


Figure 7

Scope

The scope of this use case is the user accessing the Map view

Description

This use case describes the Map view processes

Flow Description

Precondition

The system has an internet connection

The user has set the "home" local station

Activation

This use case starts when the user presses *Map* button*.*

Main flow

1. The system displays a map of the train line
2. The system fetches the train data from the API and displays it
3. The system displays the data and highlights the users local station
4. The user selects the camera view

Alternate flow

None

Exceptional flow

E1: The system cannot access the Irish Rail API

1. The Map is displayed, and the exception is caught and handled

Termination

The system exits, and the home screen is presented

Post condition

The system goes into a wait state

2.1.11 Staff Functional Requirements

The Staff **Application**

1 The user logs into the system

2 The user selects the station/place of work

3 The user selects a client requesting assistance

4 The user logs out

Staff Use Case Diagram

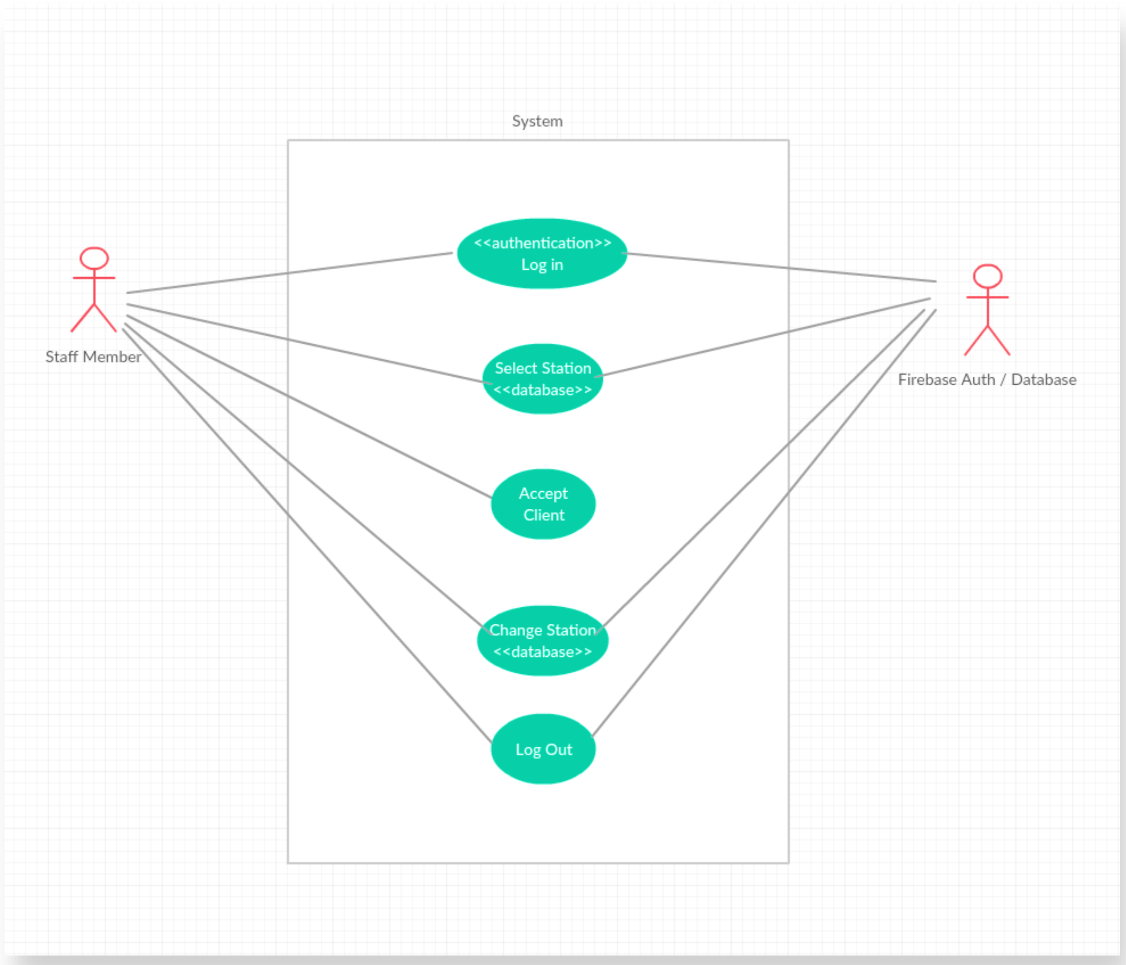

System
<<authentication>> Log in
Select Station <<database>>
Accept Client
Change Station <<database>>
Log Out
Staff Member
Firebase Auth / Database

Figure 8

2.1.12 Requirement 6

Staff Log in

The Staff member logs into the system.

2.1.12.1 Description & Priority

The client must sign in to the system to gain access to the firebase database and get an AUTH signature.

2.1.12.2 Use Case Staff Log in

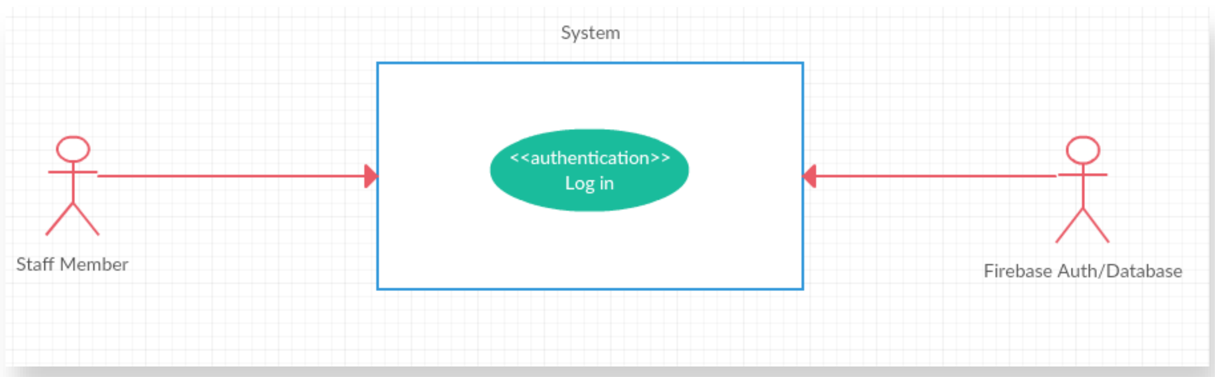


Figure 9

Scope

The scope of this use case is to log the Staff member into the system using firebase AUTH.

Description

This use case describes the firebase AUTH login process

Flow Description

Precondition

The system has an internet connection

Activation

This use case starts when the user presses *Sign in*

Main flow

1. The system displays previously signed in personnel
2. The User selects their identity
3. The system retrieves the Firebase AUTH signature and logs the selected user in

Alternate flow

A1: The Staff member is a new user

4. The User clicks “add account”
5. The AUTH system accepts email and password
6. The use case continues at position 2

A2: The User is already signed in

1. The use case continues at position 3

Exceptional flow

E1: The User is not registered with Google

5. The User is unable to proceed

Termination

The system exits and the home screen is presented

Post condition

The system goes into a wait state

2.1.13 Requirement 7

Select Station

The Staff member selects from a list of stations.

2.1.13.1 Description & Priority

The staff member must select a station in order to update the firebase database of real-time staffing details. This is the core client requirement of the application.

2.1.13.2 Use Case

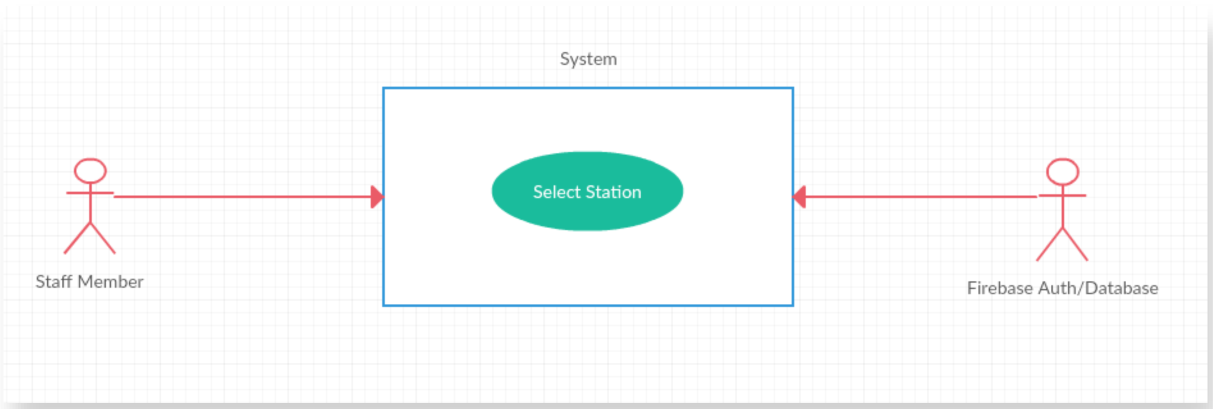


Figure 10

Scope

The scope of this use case is to update firebase database.

Description

This use case describes the process of updating firebase when the staff member makes a selection.

Flow Description

Precondition

The staff member has logged in and the system has retrieved the AUTH unique user key.

Activation

This use case starts after the sign in activity.

Main flow

1. The system displays a list of stations
2. The staff member selects their station
3. The system updates the Firebase data base

Alternate flow

A1: The Staff member is not a new user

a.The staff user clicks “add account”
b.The AUTH system accepts email and password
c.The use case continues at position 2

A2: The User is already signed in

1. The system retrieves the previously stored station
2. The use case continues at position 3

Exceptional flow

None

Termination

The activity is closed, and the system presents the next activity

Post condition

The system goes into a wait state

2.1.14 Requirement 8

Accept Client

The Staff member acknowledges receipt of client notification

2.1.14.1 Description & Priority

The staff member receives a list of potential clients currently travelling on Irish rail. Clicking on a notification will send an acknowledgement to receiving client.

2.1.14.2 Use Case Accept Client

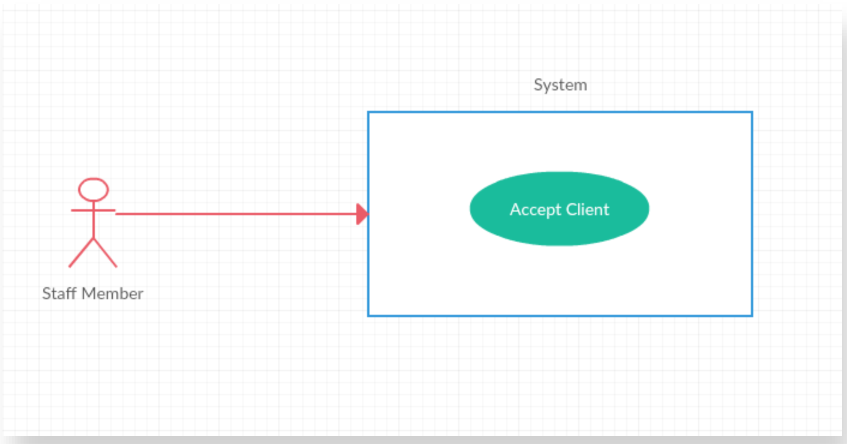


Figure 11

Scope

The scope of this use case is to update firebase database.

Description

This use case describes the process of updating firebase when the staff member makes a selection.

Flow Description

Precondition

The staff member has logged in and the system has retrieved the AUTH unique user key.

Activation

This use case starts when the user clicks on a notification

Main flow

1. The system displays a list of clients
2. The staff member selects a client appropriate for their station
3. The system notifies the client of receipt

Alternate flow

None

Exceptional flow

None

Termination

The activity is closed, and the home screen is presented

Post condition

The system goes into a wait state

2.1.15 Requirement 9

Change Station

The Staff member changes the selected station

2.1.15.1 Description & Priority

The staff member can edit the station they are registered with and Firebase is updated simultaneously

2.1.15.2 Use Case

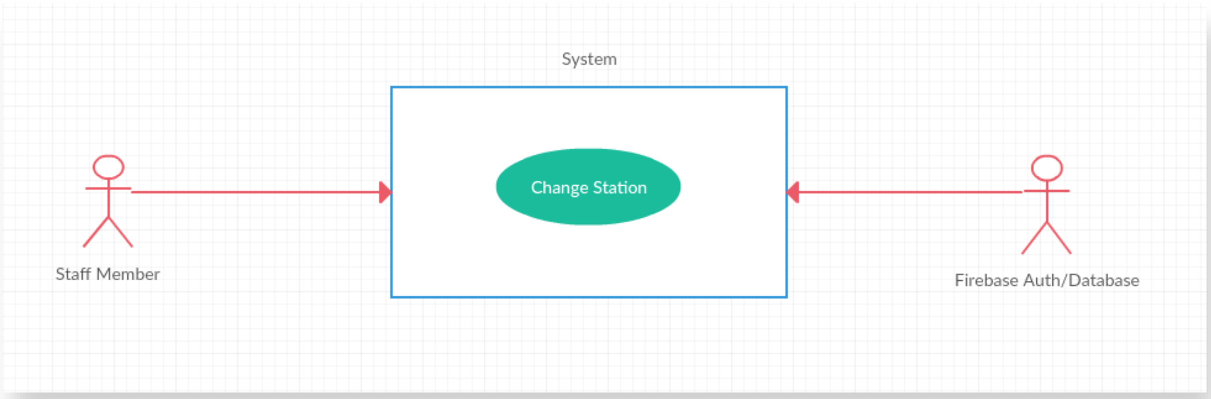


Figure 12

Scope

The scope of this use case is to change the allocated station and update firebase database accordingly.

Description

This use case describes the process of changing the allocated station

Flow Description

Precondition

The staff member has logged in and the system has retrieved the AUTH unique user key.

Activation

This use case starts when the staff user clicks on the menu list button

Main flow

1. The system removes the staff user details from firebase displays a list of stations
2. The staff member selects a new station
3. The system updates Firebase of the new selection

Alternate flow

None

Exceptional flow

None

Termination

The activity is closed, and the home screen is presented

Post condition

The system goes into a wait state

2.1.16 Requirement 10

Staff Log out

The Staff member log out

2.1.16.1 Description & Priority

The staff member logs out of the system and firebase is updated of the state

2.1.16.2 Use Case Staff Logout

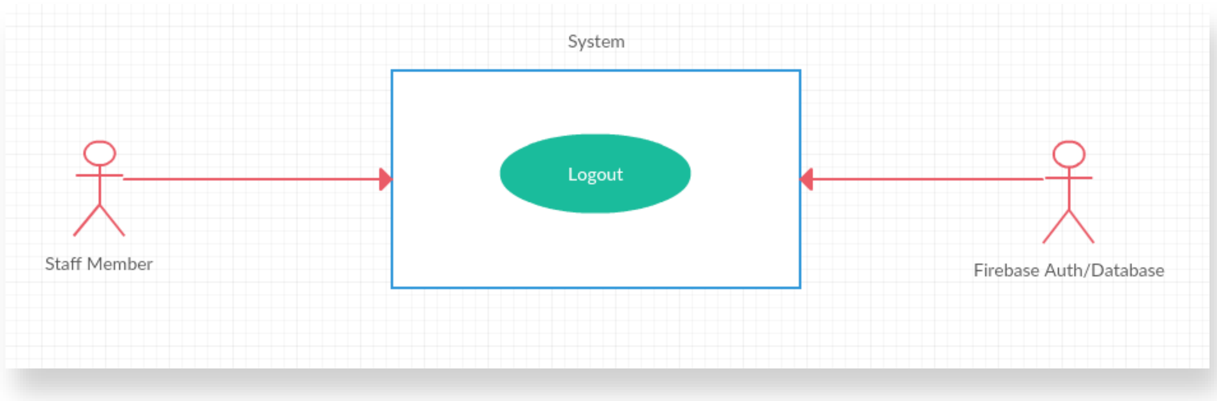


Figure 13

Scope

The scope of this use case is to log the staff member out of the system and update firebase database accordingly.

Description

This use case describes the process of logging out of the system

Flow Description

Precondition

The staff member has logged in and the system has retrieved the AUTH unique user key.

Activation

This use case starts when the staff user clicks on the menu log out button

Main flow

1. The system logs the staff user out from the database
2. The system requests Firebase AUTH to log the user out
3. The system receives confirmation and exits

Alternate flow

None

Exceptional flow

None

Termination

The activity is closed, and the home screen is presented

Post condition

The system goes into a wait state

2.1.17 Non-Functional Requirements

2.1.17.1 Performance/Response time requirement

The system relies heavily on firebase and the Irish Rail API, all functional operations are performed asynchronously, and the system is designed to operate in such an environment using call-backs for when the async task is completed.

2.1.17.2 Availability requirement

An independent server is employed to communicate with the Irish Rail API due to the poor nature of maintenance and reliability.

This gives the developer independent control as to the nature of the API response to the system.

Furthermore, overall system access is restricted to users making requests using the firebase AUTH token.

2.1.17.3 Recover requirement

The system Async Task class for getting train data is robust and designed to manage irregular or empty response values. Errors are caught and handled accordingly where a JSON object array *(multiple trains)* is returned as a JSON object only *(one train)* or when an empty *(no trains {})* array is returned.

2.1.17.4 Security requirement

System access is restricted to users making requests using the firebase AUTH token. Users must be logged into a google account in order to make read/write operations on the firebase database.

2.1.17.5 Maintainability requirement

System functionality caters for Android API 19 → API 26 *(90.1% of devices)* and code changes may be facilitated by updates. The system APK files are signed files in order to manage this functionality.

2.1.17.6 Portability requirement

The system is designed to operate on Android based devices, however due to the extendibility of the firebase database tree - this functionality could be further extended to devices using web-based hybrid mobile technologies such as *React.* *(Reactjs.org, 2018)*

2.1.17.7 Extendibility requirement

This software is written in an iterative modularised form, lending itself to further expansion and modernisation. There is no impediment for further development of any kind.

2.1.17.8 Resource utilization requirement

The system must have an internet connection to operate. All functionality with regard to database operations, user identity management and train data utilization are reliant upon this.

Android Studio additional libraries

- *Picasso* - for image loading
- *Firebase UI*, *android support design*, *recyclerView* and *cardView* – for UI displays
- Google Play Services – a requirement for location support
- Maps math-utility - library for location calculations

2.2 Design and Architecture

2.2.1 General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

- **Create a software solution in Android to enable real-time communication between Irish Rail commuters with disabilities and Irish Rail Staff**
  The solution comprises of two applications.

- **The Client application**
  This application takes in the user's origin and destination data and sends it to the Firebase database.

- **The Staff application**
  This application displays the user's data taken from the Firebase database.

- **Firebase database**
  This database stores the user's information as well as the staffing locations to facilitate communications between both applications.

- **Android applications and Irish Rail API**
  Both applications avail of the Irish Rail API to display train details and location.

- **Firebase AUTH Integration**
  This allows users to securely send and receive data to Firebase securely as well as providing logged in user details.

2.2.2 Assumptions / Constraints / Standards

- System should have an internet connection
- The apps must be able to access the Irish Rail API as well as Firebase
- Users must have a Google account
- Staff members will be able to share a common mobile device with their login credentials stored uniquely.

2.2.3 Software Architecture

TAP Client Class Diagram

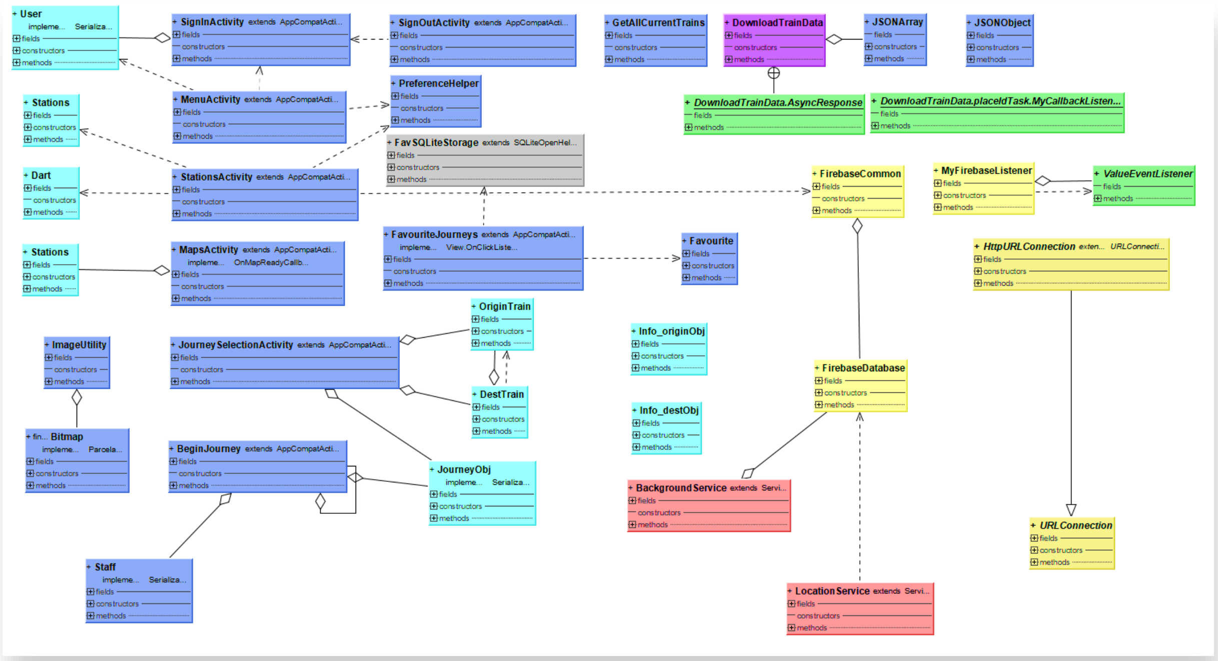

User
SignInActivity extends AppCompatActi...
SignOutActivity extends AppCompatActi...
GetAllCurrentTrains
DownloadTrainData
JSONArray
JSONObject
Stations
MenuActivity extends AppCompatActi...
PreferenceHelper
DownloadTrainData.AsyncResponse
DownloadTrainData.placeIdTask.MyCallbackListen...
FavSQLiteStorage extends SQLiteOpenHel...
Dart
StationsActivity extends AppCompatActi...
FirebaseCommon
MyFirebaseListener
ValueEventListener
Stations
MapsActivity extends AppCompatActi...
FavouriteJourneys extends AppCompatActi...
Favourite
HttpURLConnection exten... URLConnecti...
OriginTrain
ImageUtility
JourneySelectionActivity extends AppCompatActi...
Info_originObj
DestTrain
FirebaseDatabase
Info_destObj
Bitmap
BeginJourney extends AppCompatActi...
JourneyObj
BackgroundService extends Servi...
URLConnection
Staff
Location Service extends Servi...

Figure 14

Class Structure

- *helper_classes*
- *models*
- *services*

TAP Staff Class Diagram

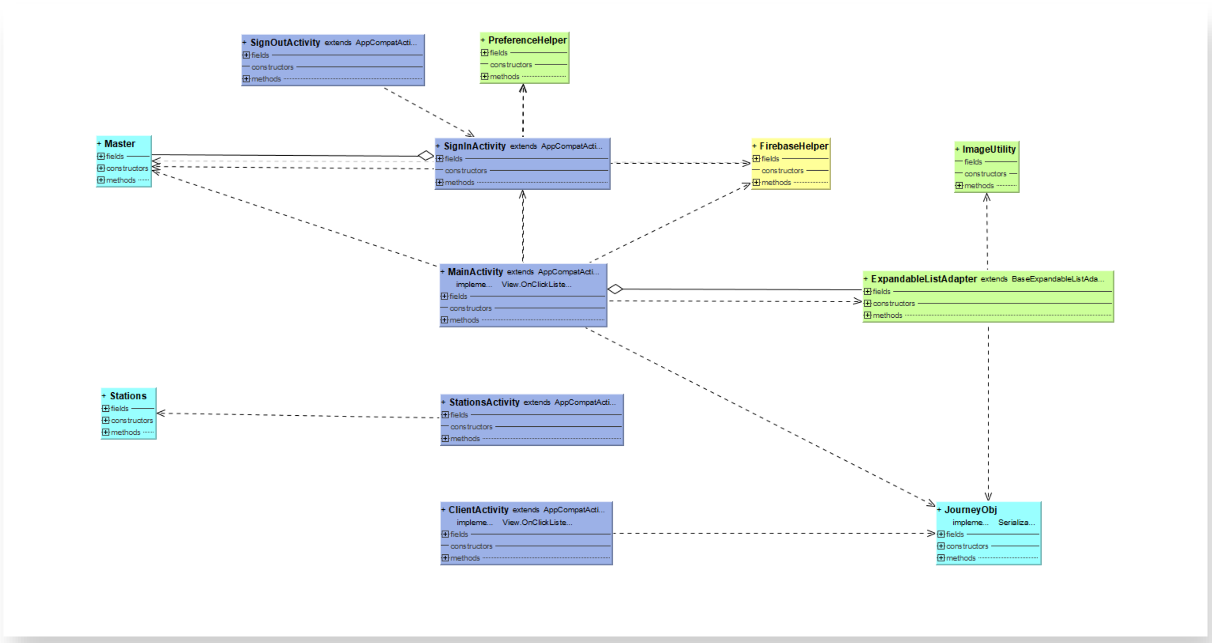
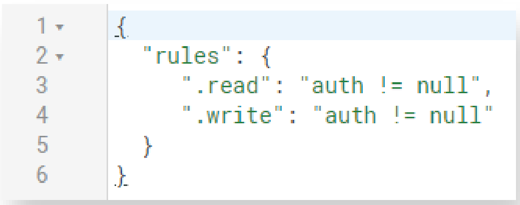

SignOutActivity extends AppCompatActi...
PreferenceHelper
Master
SignInActivity extends AppCompatActi...
FirebaseHelper
ImageUtility
MainActivity extends AppCompatActi... impleme... View.OnClickListe...
ExpandableListAdapter extends BaseExpandableListAda...
Stations
StationsActivity extends AppCompatActi...
ClientActivity extends AppCompatActi... impleme... View.OnClickListe...
JourneyObj impleme... Serializa...
fields
constructors
methods

Figure 15

2.2.4 Security Architecture

AUTH token required for application functionality.



```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Both applications employ a sign in / sign out facility in order to maintain security concerns regarding users' personal data and location. Upon signing in a firebase

AUTH token is acquired and used in all read/write operations of the firebase database to ensure the authenticity of the users and the given accessible rights.

2.2.5 Communication Architecture

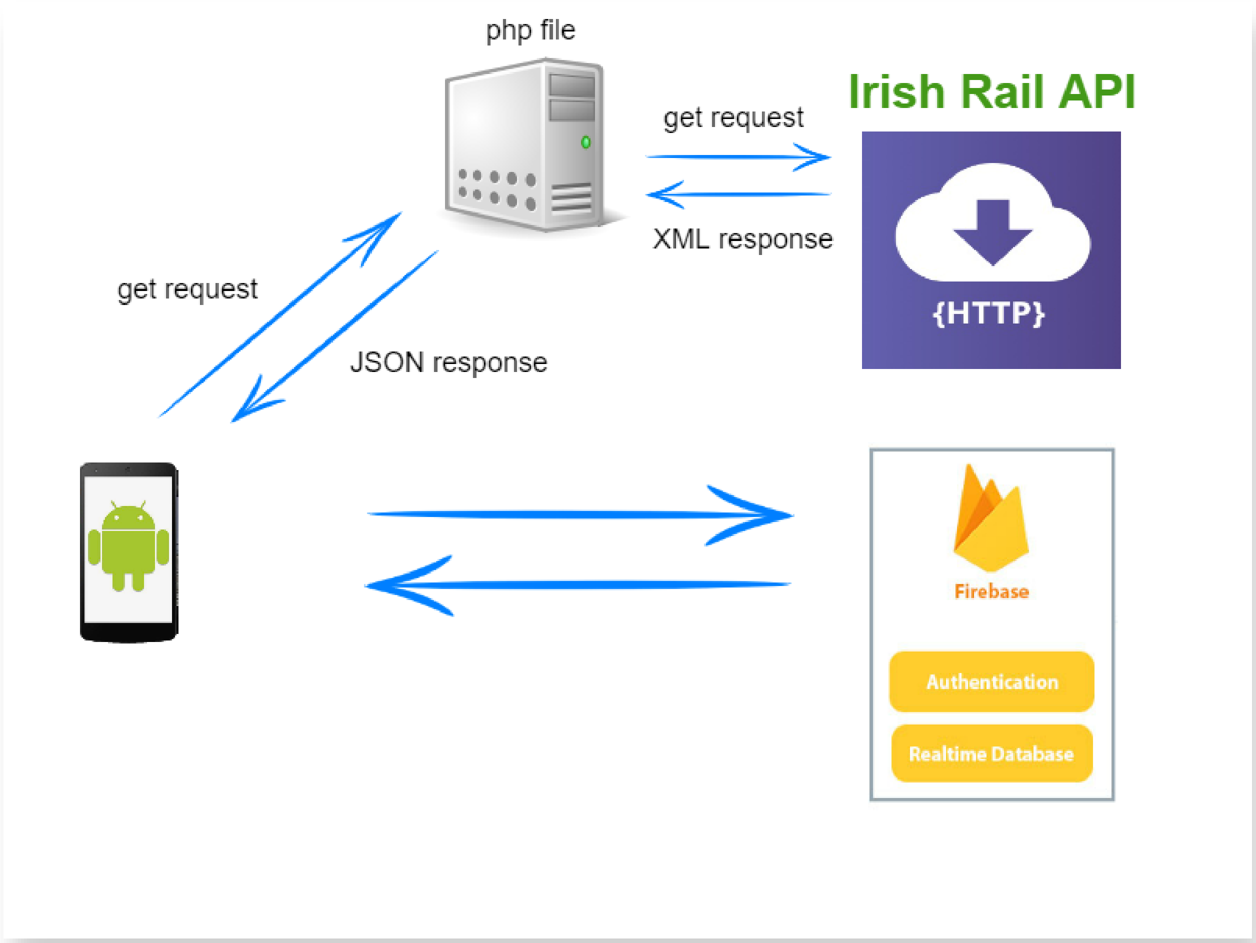


Figure 16

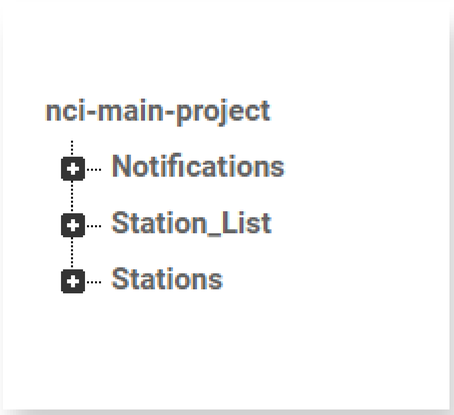
2.2.6 Database Design

Firebase database structures should not be deeply nested. This favours read/write functionality and is a key component in the database layout design.

It is for this reason that a unique key is produced using the push method, where this key can be used to reference additional data structures in a separate folder rather than have them nested further down the tree.

An example is the origin and destination structures each have their own unique key. References to the relevant viewed folder are obtained using the same unique key generated by firebase.

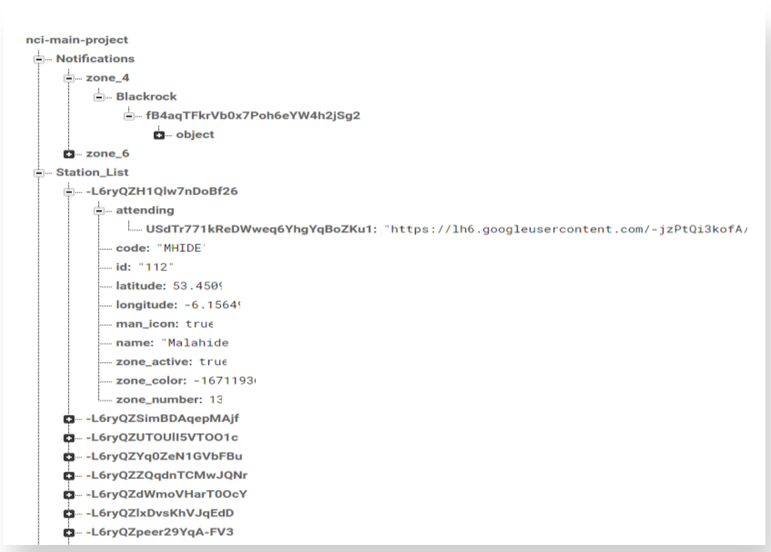
Firebase data structure closed



```
nci-main-project
  Notifications
  Station_List
  Stations
```

Figure 17

Firebase data structure open



```
nci-main-project
  Notifications
    zone_4
      Blackrock
        fB4aqTFkrVb0x7Poh6eYW4h2jSg2
          object
    zone_6
  Station_List
    -L6ryQZH1Qlw7nDoBf26
      attending
        USdTr771kReDWweq6YhgYqBoZKu1: "https://lh6.googleusercontent.com/-jzPtQi3kofA/
      code: "MHIDE'
      id: "112"
      latitude: 53.450
      longitude: -6.1564
      man_icon: true
      name: "Malahide
      zone_active: true
      zone_color: -1671193
      zone_number: 13
    -L6ryQZSimBDAqepMAjf
    -L6ryQZUTOUlI5VTOO1c
    -L6ryQZYq0ZeN1GVbFBu
    -L6ryQZZQqdnTCMwJQNr
    -L6ryQZdWmoVHarT0OcY
    -L6ryQZlxDvsKhVJqEdD
    -L6ryQZpeer29YqA-FV3
```

Figure 18

2.2.7 Data Conversions

See communications diagram ***Figure 16***. The applications make a request via a server with a PHP file. The request is forwarded to the Irish Rail API and the XML response is converted and relayed back in JSON format.

2.3 *Implementation and Testing*

“else if”

This software solution consisting of two applications, heavily reliant on interactivity, external data CRUD operations and AsyncTask API access, required careful consideration regarding the overall project development. Along with the DRY methodology, the author continues to implement the *“else if”* methodology in all production work.

When writing a conditional statement *if, then ... else*, should the author feel the need for an additional condition *else if*, it is the opinion of the author that the condition itself needs re-assessment. Many problems can be addressed first and foremost by prevention. It is with this attitudinal ethos that the author developed and tested this project.

Features introduced, were firstly modularised into smaller components and tested iteratively, firstly as an individual section of code then incrementally as a cohesive component. When all preliminary tests were satisfied, the respective modules were introduced to the application for further interactivity testing.

This approach was adopted throughout the applications development lifecycle.

2.3.1 Sign In functionality

Both applications required an internet connection to function. An internet connection isNetworkAvailable() method was implemented and was the first additional component to be introduced to the sign-in class, invoked prior to sign-in activation. The Sign in class requires connectivity to access firebase Auth.

```
// Check connectivity before launching
online = isNetworkAvailable(this);

Log.e(TAG, "isNetworkAvailable: " + online);
if (!online) {
    showInfoDialog(this);
    signInButton.setEnabled(false);
    return;
}
signInButton.setEnabled(true);
mAuth = FirebaseAuth.getInstance();
```

Name	System Status	Expected	Actual	Result
Sign in	Network available	Show Dialog: false Continue: true	Same	Pass
Sign in	No Network	Show Dialog: true Continue: false	Same	Pass
Sign in	Network available, no credit	Show Dialog: true Continue: false	Show Dialog: false Continue: true	Fail

Preliminary tests failed due to the code checking for connectivity with the Android ConnectivityManager and not additionally whether networking data was available i.e. *Mobile credit*. This was resolved by the addition of ...

```
        try {
            Process p1 = java.lang.Runtime.getRuntime().exec("ping -c 1
www.google.com");
            int returnVal = p1.waitFor();
            return (returnVal == 0);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return false;
    }
    return false;
}
```

... a ping test for networking functionality.

Name	System Status	Expected	Actual	Result
Sign in	Network available	Show Dialog: false	Same	Pass

		Continue: true		
Sign in	**No Network**	**Show Dialog: true** **Continue: false**	**Same**	**Pass**
Sign in	**Network available, no credit**	**Show Dialog: true** **Continue: false**	**Same**	**Pass**

2.3.2 DownloadData AsyncTask

The Client application relied heavily upon the *Irish Rail API* for real-time train data. Much of the authors development work on the project would be achieved during the twilight hours (2AM - 5AM) where no API data was available. Furthermore, the possibility of rail strikes, weather restrictions on travel and other unknown outcomes had to be considered.

The API which is accessed using AsyncTask via the authors own private server using a PHP file was further modified with a JSON file to return a previously captured data response for testing purposes.

```
private static JSONObject getTrainJSON(String station) {
    try {

        URL url = new URL(String.format(IE_DATA_URL_TESTING, station));
        Log.e(TAG, "getTrainJSON: " + url);
        HttpURLConnection connection =
                (HttpURLConnection) url.openConnection();

        BufferedReader reader = new BufferedReader(
                new InputStreamReader(connection.getInputStream()));
```

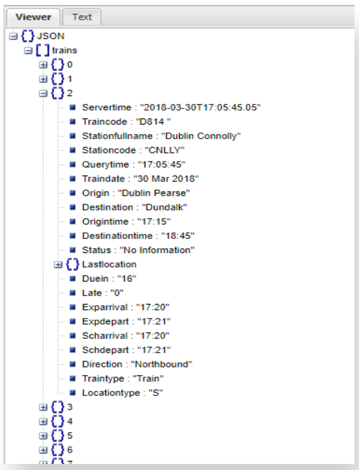
An interface call-back method was used in retrieving the data due to the asynchronous nature of the task. Firstly, to get the data for the origin and latterly on the call-back complete method to retrieve the destination data.

```
// callbackOnPermissionsResult interface when finished all downloads
asyncTask.setCallbackListener(new DownloadTrainData.placeIdTask.MyCallbackListener()
{
    @Override
    public void myCallback(int obj) {
        Log.e("DestinationDownLoad", " arraylength : " + obj);
        getMatchingTrains();
        progressBar.setVisibility(View.INVISIBLE);
        go_btn.setVisibility(View.VISIBLE);
    }


});

asyncTask.execute(dest_station);
```

Each response was checked during the procedure to ensure it contained data else the system displayed a “No train information available” text view.

This was put in place precautionarily due to the poor response from the *Irish Rail API* when trains were unavailable and is thrown in a try-catch block statement.

The API response was monitored on a continuing basis at random times throughout the day/night. The efficiency of the authors code to ascertain common trains between the origin and destination was further evaluated manually using an online JSON viewer. (Turi, 2018)



```
{} JSON
  [] trains
    {} 0
    {} 1
    {} 2
      Servertime : "2018-03-30T17:05:45.05"
      Traincode : "D814 "
      Stationfullname : "Dublin Connolly"
      Stationcode : "CNLLY"
      Querytime : "17:05:45"
      Traindate : "30 Mar 2018"
      Origin : "Dublin Pearse"
      Destination : "Dundalk"
      Origintime : "17:15"
      Destinationtime : "18:45"
      Status : "No Information"
    {} Lastlocation
      Duein : "16"
      Late : "0"
      Exparrival : "17:20"
      Expdepart : "17:21"
      Scharrival : "17:20"
      Schdepart : "17:21"
      Direction : "Northbound"
      Traintype : "Train"
      Locationtype : "S"
    {} 3
    {} 4
    {} 5
    {} 6
```

Figure 19

2.3.3 Stations Activity Data Management

Displaying the railway stations manning status in a recycler view proved to be the biggest problem regarding testing.

The initial code worked flawlessly until the author began incremental testing on all possible states.

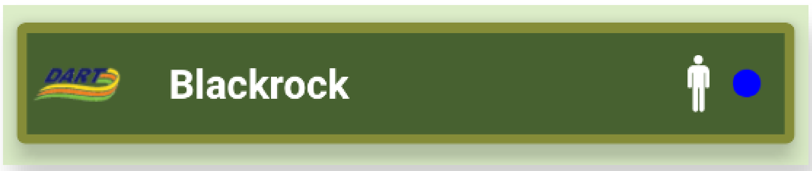
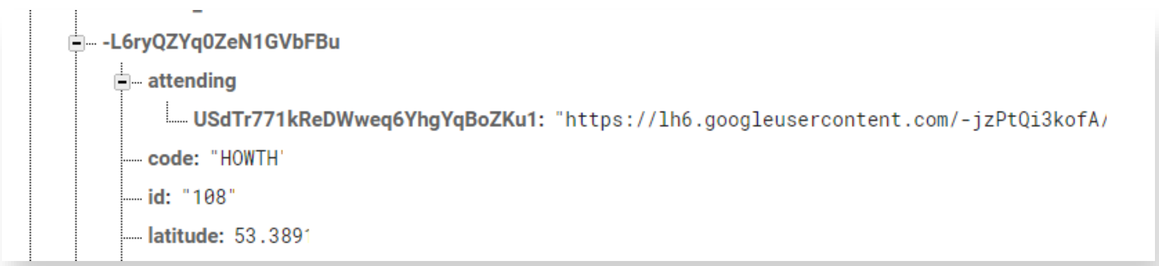


Figure 20

The illustration **figure 20** indicates what was required. A Boolean value alone to represent the station status proved insufficient.

- Is the station manned? *Boolean manned: true*
- If manned show active state zone colour. *Boolean zone_active: true*. All the associated stations in that zone will show a coloured dot to indicate the zone is manned
- Where manned, set the *attending* attribute with the relevant URL staff member photo identifier.



```
-L6ryQZYq0ZeN1GVbFBu
   attending
      USdTr771kReDWweq6YhgYqBoZKu1: "https://lh6.googleusercontent.com/-jzPtQi3kofA/
   code: "HOWTH"
   id: "108"
   latitude: 53.389
```

Figure 21

Due to the asynchronous nature of firebase, altering the staffing details failed repeatedly in testing. The cause of which was identified as the misunderstanding of the firebase *Snapshot* when capturing a result from an event listener.

The snapshot is exactly that – a snapshot in time. Any subsequent alterations will not be reflected until the next snapshot.

Moving staffing details from one station (in a zoned area) to another had to be rewritten.

Logical Flow Diagram Solution

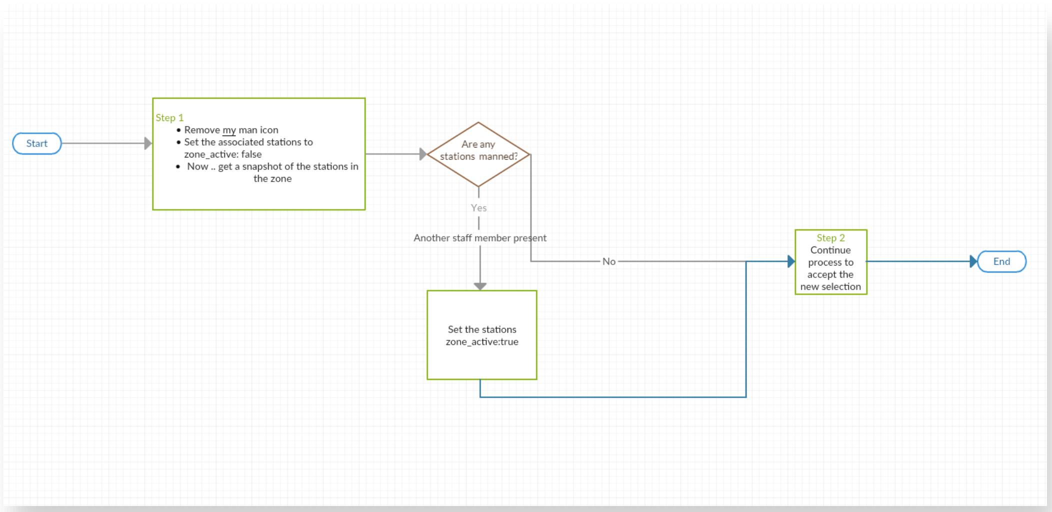
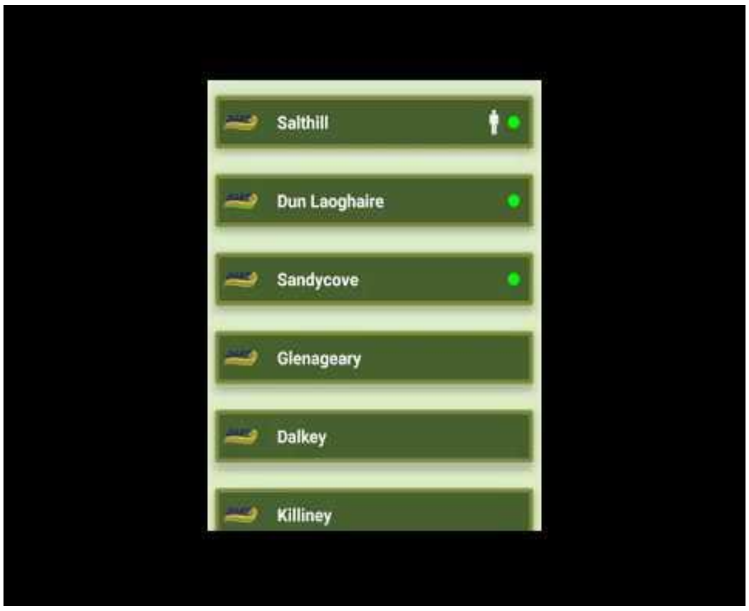

Start
Step 1
• Remove my man icon
• Set the associated stations to zone_active: false
• Now .. get a snapshot of the stations in the zone
Are any stations manned?
Yes
Another staff member present
Set the stations zone_active:true
No
Step 2
Continue process to accept the new selection
End

Figure 22

The above logical flow diagram **figure 22** implementation proved to be very robust. Testing was carried out (using emulators and real-world devices) by adding several attending members to different stations in the same zone and moving them from station to station, zone to zone etc. Finally, all tests for this module proved successful.

Name	System Status	Expected	Actual	Result
Sign in to station	**Empty station, empty zone**	**man icon: true zone_active: true**	**Same**	**Pass**
Sign out from station	**Empty station, empty zone**	**man icon: false zone_active: false**	**Same**	**Pass**
Sign in to station	**Occupied station**	**man icon: true zone_active: true**	**Same**	**Pass**
Sign out from station	**Occupied zone**	**man icon: true zone_active: true**	**Same**	**Pass**

Logical Flow Solution Video

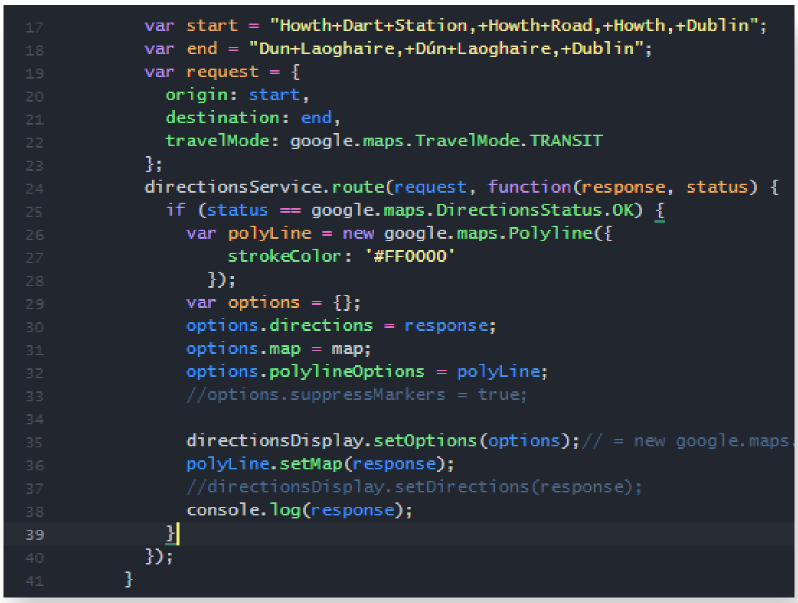


2.3.4 Google Map Polyline “*Hack*”

For authenticity purposes, the map displaying the dart line polylines, composed of the latitude and longitude points was gently extracted from the original Google maps document using the following method.

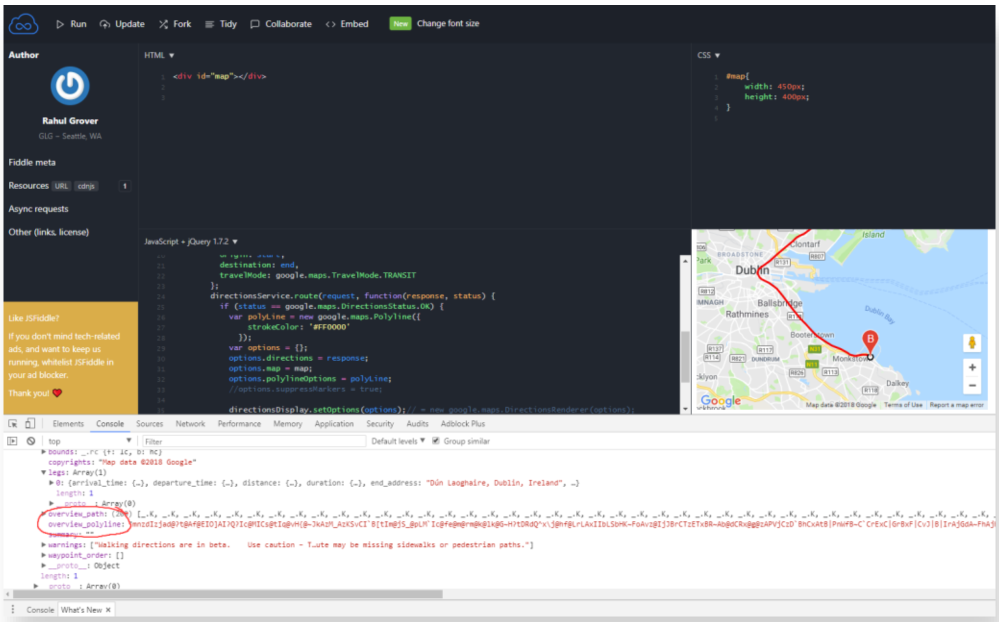
- Open Google Maps and search for directions between Greystones dart station and Howth Junction dart station *(for example)*
- Copy the search terms from the URL i.e. the result for Greystones station becomes ...
  Greystones+Station,+Iarnród+Éireann,+Church+Rd,+Killincarrig,+Greystones,+Co.+Wicklow
- Run a simple JavaScript fiddle (https://jsfiddle.net/) using the Google maps directions API with the origin and destination titles taken from Google maps and ensuring the travel mode selected is google.maps.TravelMode.Transit.
- Console log the API’s response (within which lies the encoded polyline)

The JSfiddle example script



```
        var start = "Howth+Dart+Station,+Howth+Road,+Howth,+Dublin";
        var end = "Dun+Laoghaire,+Dún+Laoghaire,+Dublin";
        var request = {
          origin: start,
          destination: end,
          travelMode: google.maps.TravelMode.TRANSIT
        };
        directionsService.route(request, function(response, status) {
          if (status == google.maps.DirectionsStatus.OK) {
            var polyLine = new google.maps.Polyline({
                strokeColor: '#FF0000'
              });
            var options = {};
            options.directions = response;
            options.map = map;
            options.polylineOptions = polyLine;
            //options.suppressMarkers = true;

            directionsDisplay.setOptions(options);// = new google.maps.
            polyLine.setMap(response);
            //directionsDisplay.setDirections(response);
            console.log(response);
          }
        });
      }
```

Figure 23



```html
<div id="map"></div>
```

```css
#map{
    width: 450px;
    height: 400px;
}
```

```javascript
      destination: end,
      travelMode: google.maps.TravelMode.TRANSIT
    };
    directionsService.route(request, function(response, status) {
      if (status == google.maps.DirectionsStatus.OK) {
        var polyLine = new google.maps.Polyline({
            strokeColor: '#FF0000'
          });
        var options = {};
        options.directions = response;
        options.map = map;
        options.polylineOptions = polyLine;
        //options.suppressMarkers = true;

        directionsDisplay.setOptions(options);// = new google.maps.DirectionsRenderer(options);
```

```
  ▶ bounds: _.rc {f: Ic, b: hc}
    copyrights: "Map data ©2018 Google"
  ▼ legs: Array(1)
    ▶ 0: {arrival_time: {…}, departure_time: {…}, distance: {…}, duration: {…}, end_address: "Dún Laoghaire, Dublin, Ireland", …}
      length: 1
    ▶ __proto__: Array(0)
  ▶ overview_path: (208) [_.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K, _.K,
    overview_polyline: "mnzdIzjad@?t@Af@EIO]AI?Q?Ic@MICs@tIq@vH{@~JkAzM_AzKSvCI`B[tIm@jS_@pLM`Ic@fe@m@rm@k@lk@G~H?tDRdQ^x\j@hf@LrLAxIIbLSbHK~FoAvz@IjJBrCTzETxBR~Ab@dCRx@g@zAPVjCzD`BhCxAtB|PnWfB~C`CrExC|GrBxF|CvJ|B|IrAjGdA~FhAj
    summary: ""
  ▶ warnings: ["Walking directions are in beta.    Use caution – T…ute may be missing sidewalks or pedestrian paths."]
  ▶ waypoint_order: []
  ▶ __proto__: Object
  length: 1
▶ __proto__: Array(0)
```

Figure 24

The encoded polyline *(part of ...)*

```
overview_polyline:"mnzdIzjad@?t@Af@EIO]AI?Q?Ic@MICs@tIq@vH{@~JkAzM_AzKSvCI`B[tIm@jS_
                   @pLM`Ic@fe@m@rm@k@lk@G~H?tD .... "
```

The polyline can be further edited using the *Interactive Polyline Encoder Utility* (Google Developers, 2018) provided by Google maps API.

A point of note for Android Studio developers ... pasting the polyline encoded string into a resource variable in Strings.xml is not advisable as will corrupt the encoded string.

Better to use the String variable in the relevant class and using ALT+Enter to enable the studio IDE to create the reference itself in Strings.xml.

Three encoded strings were used in this project to create the train lines.

Greystones → Howth Junction

Howth Junction → Howth

Howth Junction → Malahide

2.3.5 Checking for valid journeys

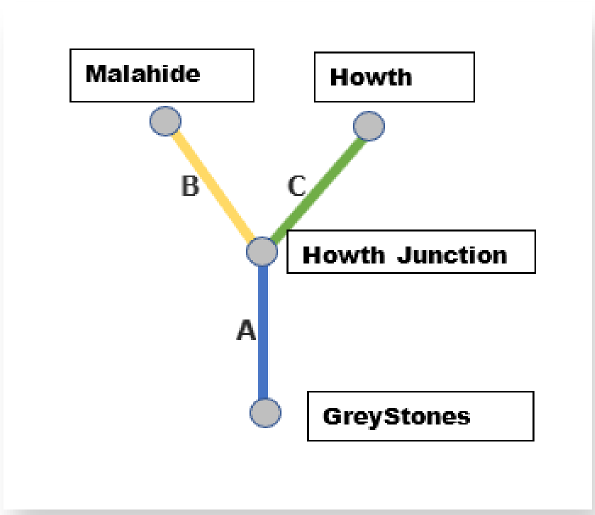


Figure 25

Valid journeys in the above diagram are from line A→B, A→C and vice a versa. Journeys from B→C and C→B are invalid, requiring a changeover in Howth Junction.

To check for this user selection error and in keeping with the *"else if"* philosophy, the author began working on the assumption that all journeys are valid and to check for invalid journeys as a matter of exception.

The resulting logical checking procedure is as follows.

- If the origin station **OR** the destination station is in line B
- **AND**
- If the origin station **OR** the destination station is in line C
- Return false
- *else* return true

```
// allow for either fork (ONLY 1) to belong to the main line
public static boolean getValidJourney(String source, String destination) {
    // Convert String Array to List

    if ((howth_line_list.contains(source) || howth_line_list.contains(destination)) &&
(malahide_line_list.contains(source) || malahide_line_list.contains(destination))) {
        return false;
    }
    return true;
}
```

2.4 Location Testing

A key feature of the Genemotion emulator is that the developer can manually allow access to virtual location data.

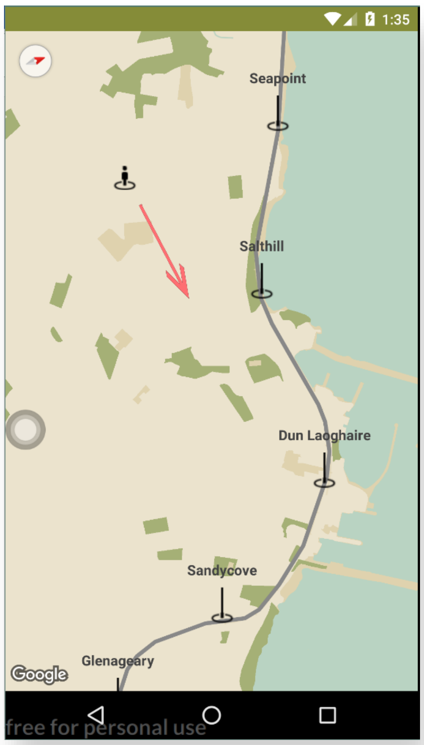
To take advantage of this feature, an additional application was developed to assist in user location testing.

When the client application is running on the emulator, a location request is made by the staff application. The client emulator request is ignored by the tester allowing the test application to “hunt” the firebase tree looking for the location request, upon finding it, it inserts a location defined by the test users draggable marker manipulation.

This gave the tester the ability to mimic user motion in real-time and getting actual data feedback from the applications involved.

The Location Test Application

Figure 26



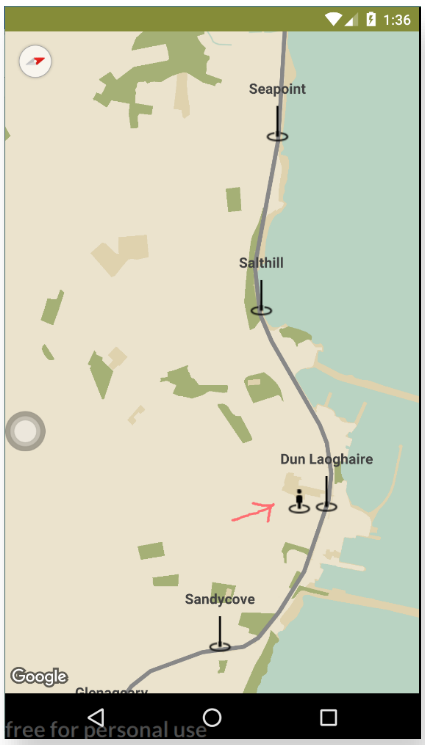
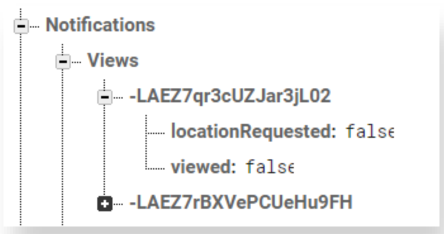


Figure 27

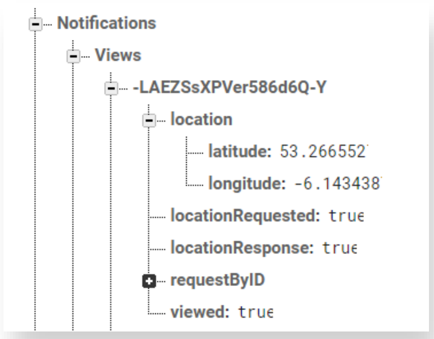
In the test application, long pressing on the icon enables the tester to move the user's icon to the desired location thus updating the firebase database in real time.

The Resulting Location Object in Firebase

Figure 28



```
Notifications
  Views
    -LAEZ7qr3cUZJar3jL02
      locationRequested: false
      viewed: false
    -LAEZ7rBXVePCUeHu9FH
```



```
Notifications
  Views
    -LAEZSsXPVer586d6Q-Y
      location
        latitude: 53.266552
        longitude: -6.143438
      locationRequested: true
      locationResponse: true
      requestByID
      viewed: true
```

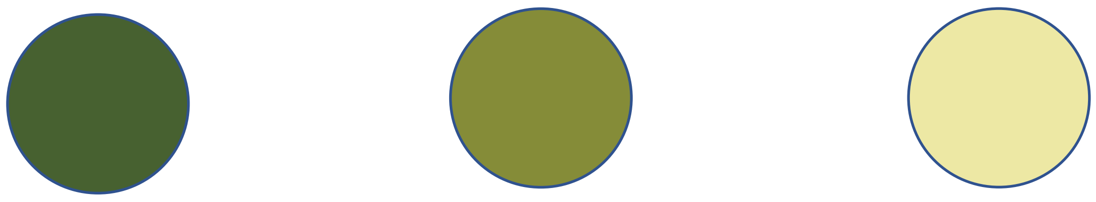
Figure 29

Location Testing in Android Studio and Firebase Recording

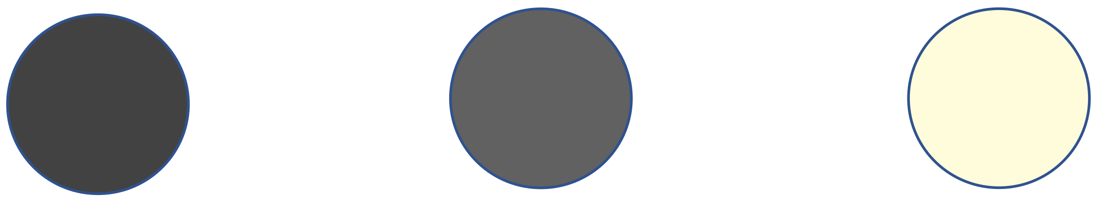
2.5 *Graphical User Interface (GUI) Layout*

Google Material design provided inspiration with regard to the applications colour scheme. Soft pastel colours were utilised throughout.

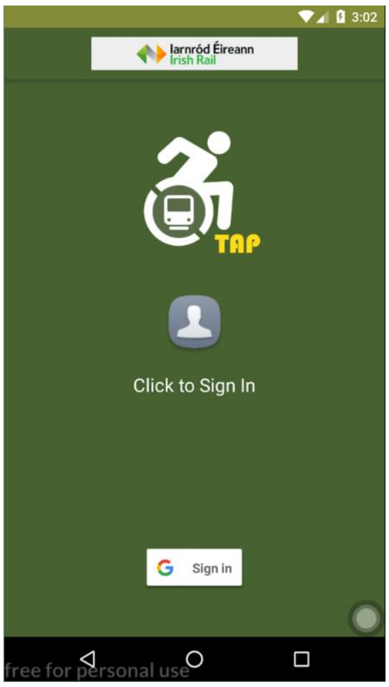
Irish Rails *Dart* colours were utilised in activities connected with journey selection...

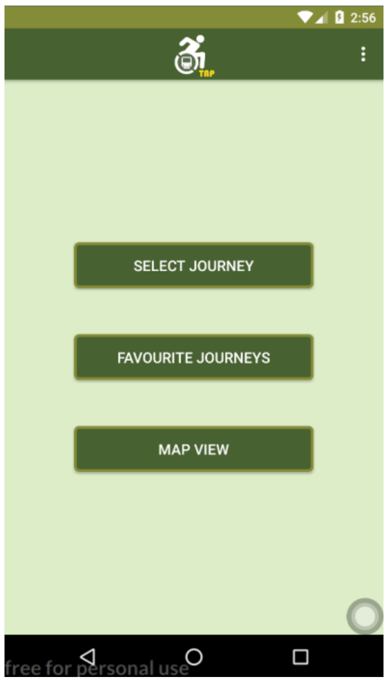


and a darker contrasting design theme applied to live data presentation.

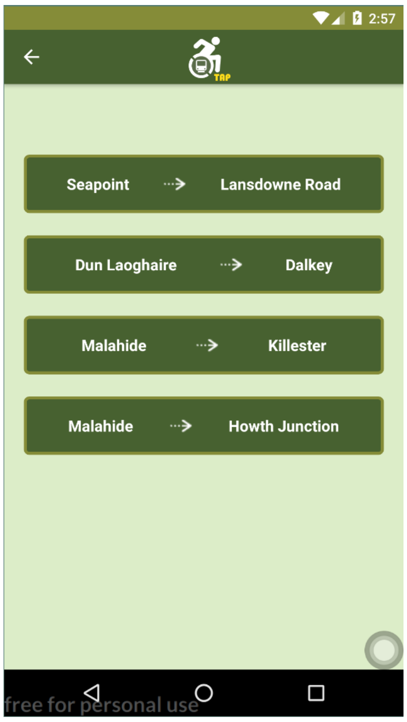


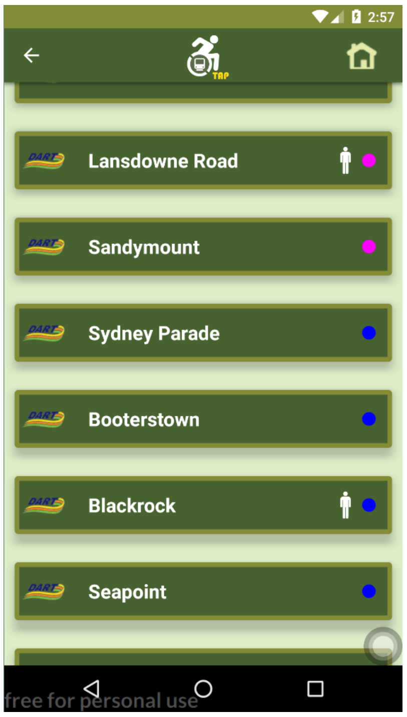
Sign in and Menu Activities



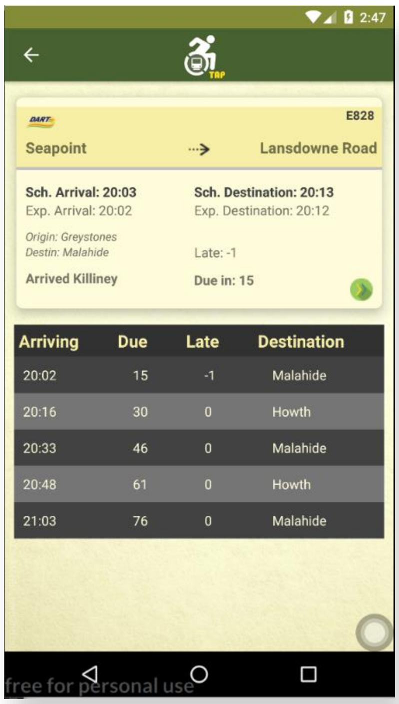


Favourite and Station Activities





Journey Selection and Begin Journey (selection complete) Activities



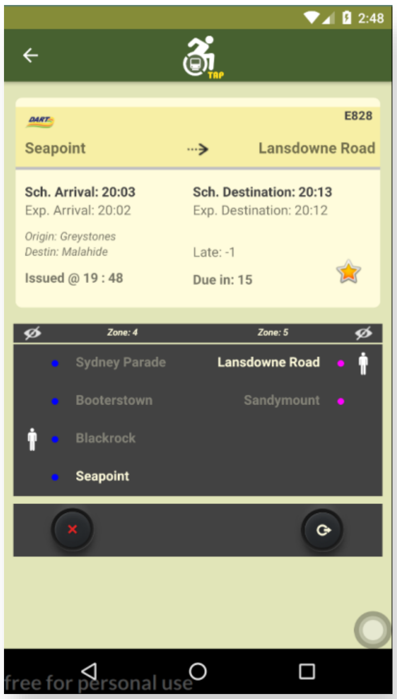
E828

Seapoint → Lansdowne Road

Sch. Arrival: 20:03
Exp. Arrival: 20:02
Origin: Greystones
Destin: Malahide
Arrived Killiney

Sch. Destination: 20:13
Exp. Destination: 20:12
Late: -1
Due in: 15

Arriving	Due	Late	Destination
20:02	15	-1	Malahide
20:16	30	0	Howth
20:33	46	0	Malahide
20:48	61	0	Howth
21:03	76	0	Malahide



E828

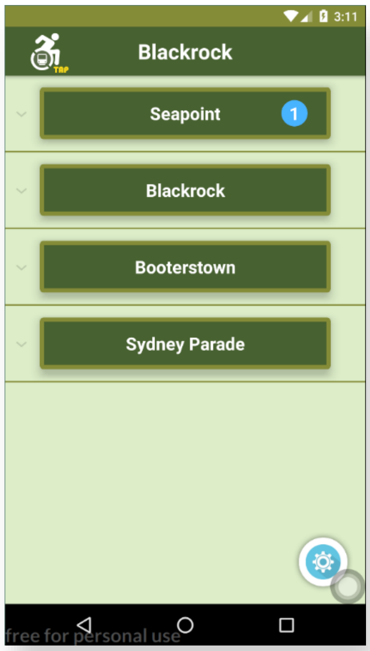
Seapoint → Lansdowne Road

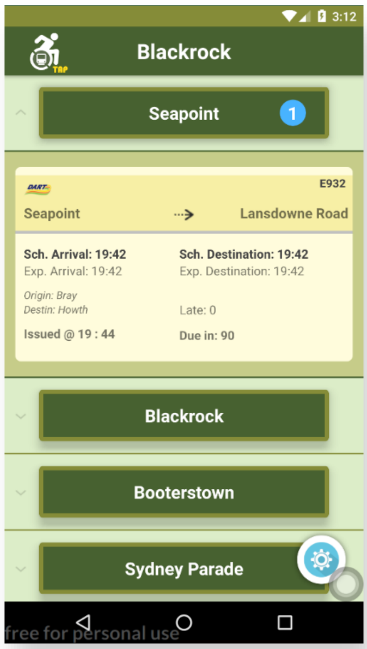
Sch. Arrival: 20:03
Exp. Arrival: 20:02
Origin: Greystones
Destin: Malahide
Issued @ 19 : 48

Sch. Destination: 20:13
Exp. Destination: 20:12
Late: -1
Due in: 15

Zone: 4	Zone: 5
Sydney Parade	Lansdowne Road
Booterstown	Sandymount
Blackrock	
Seapoint	

Staff Expandable View before and after





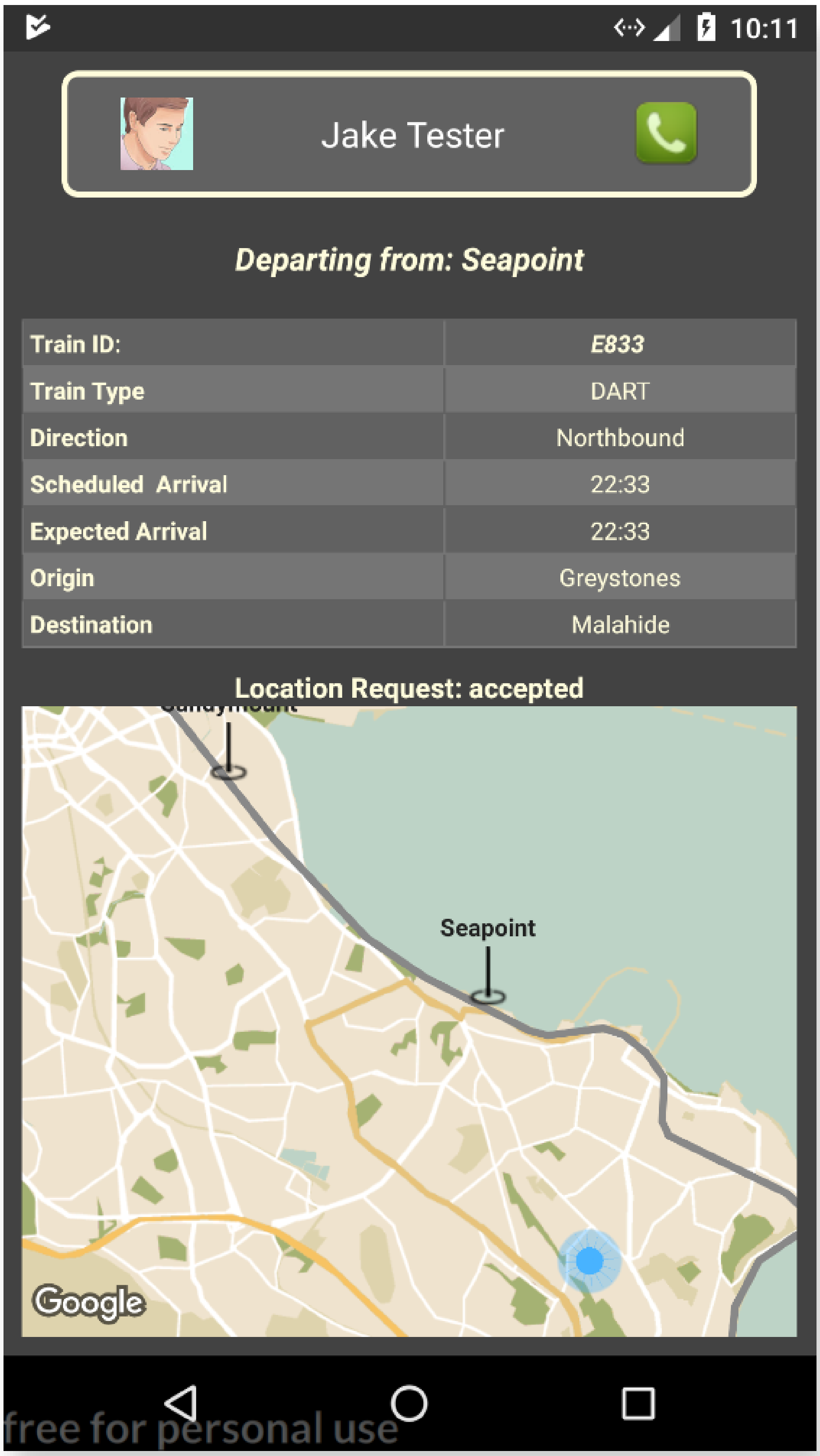
E932

Seapoint → Lansdowne Road

Sch. Arrival: 19:42
Exp. Arrival: 19:42
Origin: Bray
Destin: Howth
Issued @ 19 : 44

Sch. Destination: 19:42
Exp. Destination: 19:42
Late: 0
Due in: 90

Staff Activity for viewing individual clients



10:11

Jake Tester

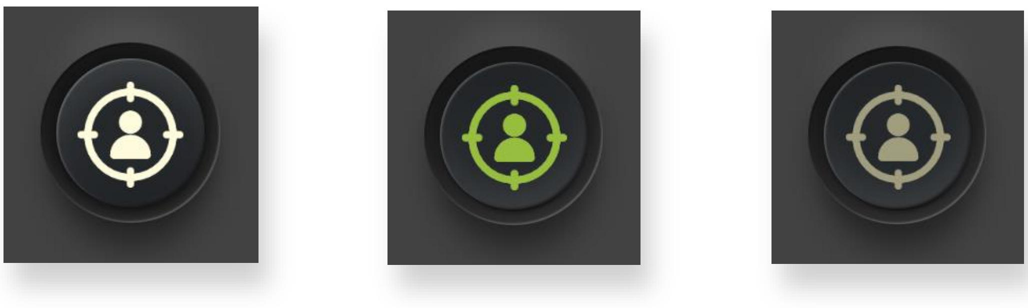
Departing from: Seapoint

Train ID:	*E833*
Train Type	DART
Direction	Northbound
Scheduled Arrival	22:33
Expected Arrival	22:33
Origin	Greystones
Destination	Malahide

Location Request: accepted

The darker theme applied to this activity relays the importance of user’s data privacy.

Buttons design default, pressed and selected states



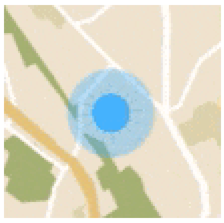
location_btn_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/target_over" android:state_selected="true"/>
    <item android:drawable="@drawable/target_pressed"
        android:state_pressed="true"/>
    <item android:drawable="@drawable/target"/>

</selector>
```

Buttons were created using *pixlr* online visual editing tool. The button state is managed by the xml selector file. (Pixlr.com, 2018)

The java class can then access the button and set the state after the button has been pressed. *my_button.setSelected(true)*

Custom animated Map marker



```
    @Override
    public void onAnimationUpdate(ValueAnimator valueAnimator) {
        float animatedFraction = valueAnimator.getAnimatedFraction();
        circle.setRadius(animatedFraction * radius * 2);
        circle.setStrokeWidth(animatedFraction*stroke_width*4);
    }
});

valueAnimator.start();
```

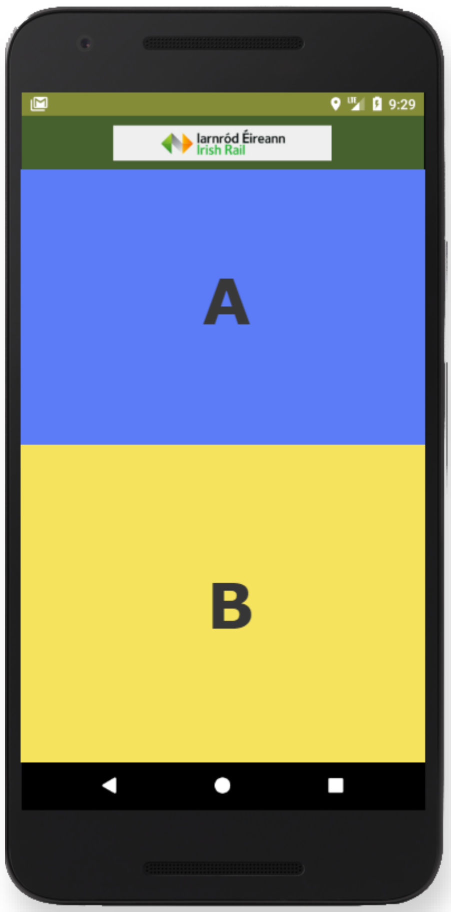
Using a value animator to update the radius and stroke width of a circle in the drawUserMarker method in DartMaster/ClientActivity.java

3 Further development or research

Many people with disabilities suffer from spasticity, particularly relevant with regard to this project, when it's the cause of manual dexterity issues.

People with dexterity issues have limited ability with UI manipulation and as a result, a completely bespoke application would be conducive to addressing this issue.

Some of the clients consulted during the development of this application solution raised this issue and having researched the inclusion of an adapted UI for this purpose, it became clear that the functional requirements and overall project development would need to be developed specifically with this in mind and <u>not</u> as an additional feature of the existing application.



In the above image station selection could be achieved by dividing the display into two buttons.

The application would begin with the users "home" station, *button A* would be labelled Northbound and *button B* Southbound.

Subsequent activities would further divide the possible destination selection into two button categories thus rendering the selection available to someone using the application with a fist or finger.

1. Dalkey → Northbound? or Southbound? *(Northbound was selected)*
2. Connolly onwards or until Tara Street? *(up to Tara Street selected)*
3. Seapoint onwards or until Salthill? *and so on ...*

All subsequent UI features would be designed in this fashion and as such all classes would need to be rewritten to suit.

Text to Speech could be incorporated as an optional feature in this current application for UI manipulation and selection. The author was mindful of this, however some of the clients consulted, some of whom had dexterity issues also had speech difficulties rendering a text to speech option unusable.

3.1 Definitions, Acronyms, and Abbreviations

T.A.P. Transport Accessibility Platform

Iarnród Éireann: Irish Rail

API: Application Programming Interface

Dart: Dublin Area Rapid Transit – electrified rail network.

The Client: The user (with a disability) of the TAP Application availing of Irish Rail services.

Staff: Members or employees of Irish Rail who are users of the Staff Application

Origin: The start railway station also referred to as Source

4 References

Google Developers. (2018). Interactive Polyline Encoder Utility | Google Maps APIs | Google Developers. [online] Available at: https://developers.google.com/maps/documentation/utilities/polylineutility [Accessed 12 Apr. 2018].

Turi, G. (2018). Online JSON Viewer. [online] Jsonviewer.stack.hu. Available at: http://jsonviewer.stack.hu/ [Accessed 24 Mar. 2018].

Rail, I. (2018). DART Improved Accessibility Pilot. [online] Irish Rail. Available at: http://www.irishrail.ie/access-dart [Accessed 5 Feb. 2018].

Firebase. (2018). Authenticate Using Google Sign-In on Android | Firebase. [online] Available at: https://firebase.google.com/docs/auth/android/google-signin [Accessed 8 Feb. 2018].

Api.irishrail.ie. (2018). [online] Available at: http://api.irishrail.ie/realtime/ [Accessed 28 Jan. 2018].

oguzbilgener. (2018). oguzbilgener/CircularFloatingActionMenu. [online] Available at: https://github.com/oguzbilgener/CircularFloatingActionMenu [Accessed 1 Feb. 2018].

Square.github.io. (2018). Picasso. [online] Available at: http://square.github.io/picasso/ [Accessed 3 Feb. 2018].

Blake Knox, K. (2018). Ross snubs rail access launch as wheelchair users stranded on Dart - Independent.ie. [online] Independent.ie. Available at: https://www.independent.ie/irish-news/ross-snubs-rail-access-launch-as-wheelchair-users-stranded-on-dart-36545469.html [Accessed 29 Jan. 2018].

(O'Kelly, 2018). A day in my wheels. [online] Facebook.com. Available at: https://www.facebook.com/adayinmywheels2016/ [Accessed 9 Feb. 2018].

Stackoverflow.com. (2018). Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Available at: https://stackoverflow.com/ [Accessed 17 Apr. 2018].

Genymotion – Android Emulator for app testing. (2018). *Genymotion – Fast & Easy Android Emulator*. [online] Available at: https://www.genymotion.com/ [Accessed 17 Apr. 2018].

Pixlr.com. (2018). Online Photo Editor | Pixlr Editor. [online] Available at: https://pixlr.com/editor/ [Accessed 17 Apr. 2018].

Reactjs.org. (2018). React - A JavaScript library for building user interfaces. [online] Available at: https://reactjs.org/ [Accessed 18 Apr. 2018].

5 Appendix

Attach all your partial submissions as appendices.

5.1 Project Proposal

Project Proposal

T.A.P.

Transport Accessibility Portal

Shay de Barra

x16115864

x16115864@student.ncirl.ie

Higher Diploma in Science in Computing

Mobile Application Specialisation

2/02/2018

"An app to enable people with disabilities avail of Dart rail services"

The Problem

Rail commuters with disabilities, specifically wheelchair users, require assistance from Iarnród Éireann staff to embark and disembark Dart train services. This assistance involves the provision of a portable ramp and requires an advanced notice provision of four hours prior to travelling as part of the new *DART Improved Accessibility Pilot scheme*.

"Once guided to the relevant Zone, call the number provided for the Zone (see Station Contact Hours) and inform the staff member of your travel details as recommended, four hours in advance. Then on arrival at your departure station, make yourself known to staff and they will ensure your safe passage onto the train and arrival at your destination station." (Rail, 2018)

This scheme attempts to address the issue at hand which is one of communication and does not in any way address the rail user's needs.

Rail users with disabilities (hereinafter referred to as the client) seek to access public transportation in the same manner as the general travelling public.

Aims

The author seeks to address this problem through the implementation of a mobile software application. The scope of the application will be specific to the Dart rail services as a proof of concept. In order to achieve this, the problem of communication must be clearly defined.

6. The Client needs to know if a station is manned <u>prior</u> to traveling or disembarking.
7. The Client should have full knowledge of station staffing in real time.
8. If a station is manned, the client should be able to communicate with staff members upon arrival and or when ready to disembark.
9. Station staff need to be made aware of the clients impending arrival in order to prepare for their needs.

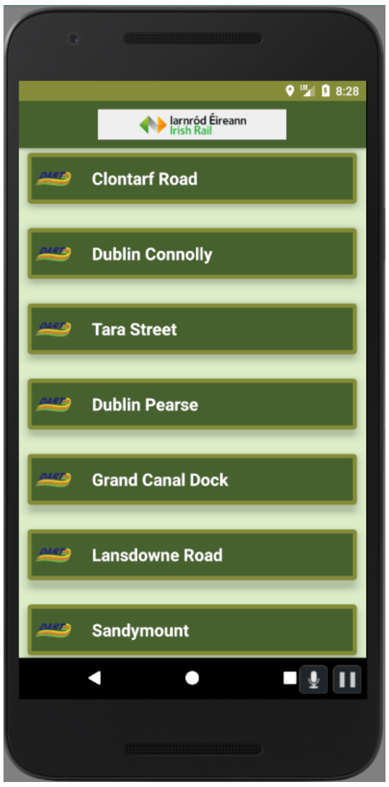
Mobile devices equipped with location services along with the ability to send notifications and communicate with a third-party server can solve this issue in real-time.

Technical Implementation

The ***TAP*** application solution will require the use of two software applications to address these issues.

The *Dart Staff* ***TAP*** application

Dart employees will login to the application using FirebaseAuth login. The next screen presented will be a list of train stations to choose from. (Firebase 2018)



Stations list view

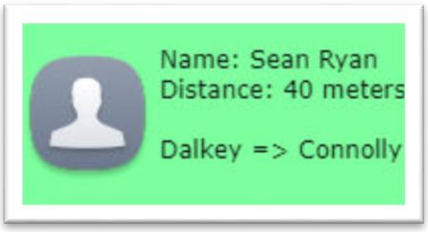
The selected station along with the controller's details will be stored in the Firebase database. The staff member will now be presented with a screen where notifications from impending clients will be posted. This is the screen the staff member will be directed to when a notification is clicked upon.

When the staff member signs out the Firebase database is updated accordingly.

The *Dart Client* ***TAP*** application

The Client can open their application to select their journey from a list of Dart stations where they will be able to see which stations (if not all) that are manned for departure and arrival.

When the Client has selected their journey i.e. *Dalkey* => *Connolly* a notification will be sent to the relevant *Dart Staff* member and will display a status change when they are within the near location of the departure station.



Name: Sean Ryan
Distance: 40 meters

Dalkey => Connolly

CardView Notification

Upon boarding the train, the arrivals station will be notified of the client's location and estimated time of arrival.

Background location services utilising the FusedLocationProviderClient will monitor the client's location whilst in transit. The completed journey will be stored for future use as a "Frequent Journeys" menu item.

Technical Details

This software solution will be coded in Java using Android Studio as the official native Android IDE (Integrated Developer Environment).

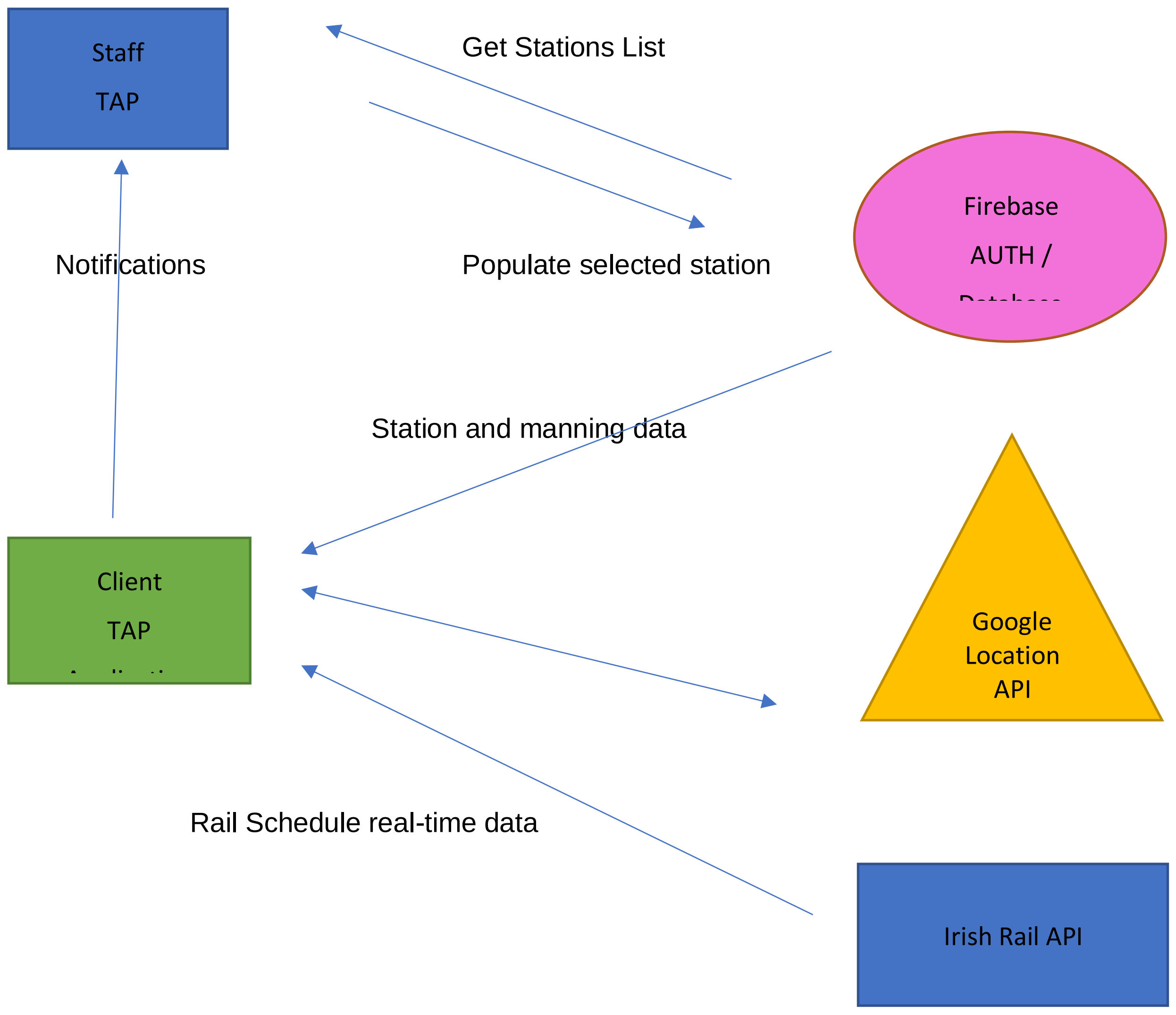
The backend server solution will be provided by the Firebase real-time database.

Furthermore, the Irish Rail API will be used to gather real-time Dart travel information. (Api.irishrail.ie, 2018)

Libraries

- *RecyclerView* along with *CardView* to display the stations list synced with Firebase
- *Picasso* to do the heavy lifting of displaying Google logged in staff URL pictures. (Square.github.io, 2018)
- *CircularFloatingActionMenu* as an ease of use floating action button for navigation. (Oguzbilgener, 2018)

Data Flow



Testing

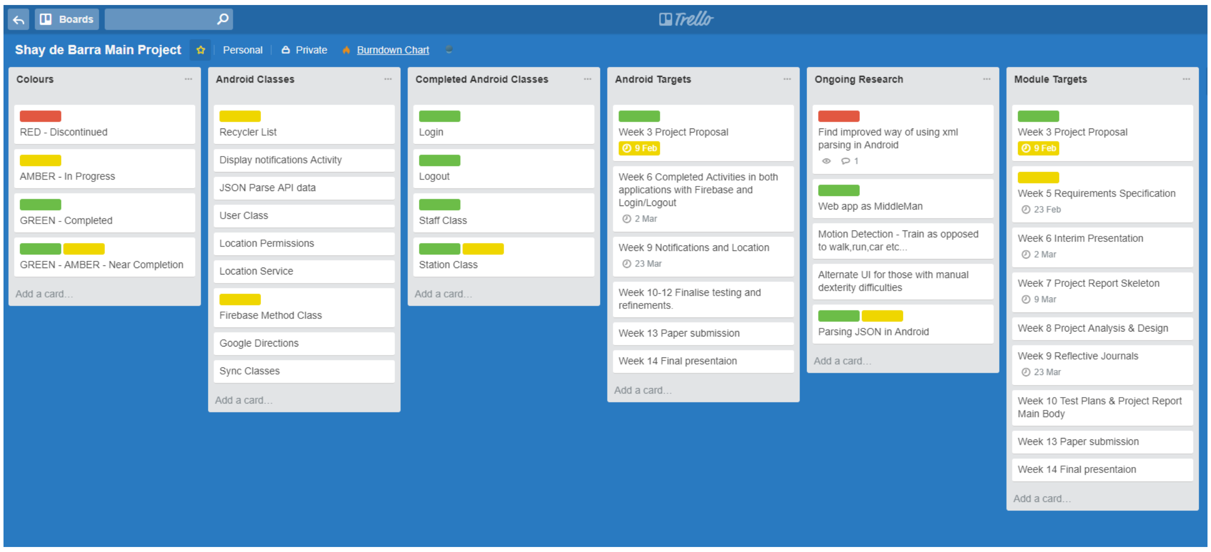
Continuous testing is to be carried out using the onboard ADB Android emulator with an additional real-world device connected via USB for debugging purposes.

Localised location testing will be accommodated by using a small test application to populate the Firebase database temporarily with local testing coordinates. This provides for a local street to emulate the source and destination stations for initial testing and evaluation purposes.

Station to station real world evaluation will be carried out when all localised test results have passed. External consultation will be availed of, on an ongoing basis.

Project Plan

Trello link



Boards

Trello

Shay de Barra Main Project · Personal · Private · Burndown Chart

Colours
- 🟥 RED - Discontinued
- 🟨 AMBER - In Progress
- 🟩 GREEN - Completed
- 🟩🟨 GREEN - AMBER - Near Completion
- Add a card...

Android Classes
- 🟨 Recycler List
- Display notifications Activity
- JSON Parse API data
- User Class
- Location Permissions
- Location Service
- 🟨 Firebase Method Class
- Google Directions
- Sync Classes
- Add a card...

Completed Android Classes
- 🟩 Login
- 🟩 Logout
- 🟩 Staff Class
- 🟩🟨 Station Class
- Add a card...

Android Targets
- 🟩 Week 3 Project Proposal · 9 Feb
- Week 6 Completed Activities in both applications with Firebase and Login/Logout · 2 Mar
- Week 9 Notifications and Location · 23 Mar
- Week 10-12 Finalise testing and refinements.
- Week 13 Paper submission
- Week 14 Final presentaion
- Add a card...

Ongoing Research
- 🟥 Find improved way of using xml parsing in Android · 1
- 🟩 Web app as MiddleMan
- Motion Detection - Train as opposed to walk,run,car etc...
- Alternate UI for those with manual dexterity difficulties
- 🟩🟨 Parsing JSON in Android
- Add a card...

Module Targets
- 🟩 Week 3 Project Proposal · 9 Feb
- 🟨 Week 5 Requirements Specification · 23 Feb
- Week 6 Interim Presentation · 2 Mar
- Week 7 Project Report Skeleton · 9 Mar
- Week 8 Project Analysis & Design
- Week 9 Reflective Journals · 23 Mar
- Week 10 Test Plans & Project Report Main Body
- Week 13 Paper submission
- Week 14 Final presentaion
- Add a card...

External Consultation

"It wasn't the most promising of starts. Two wheelchair users found themselves stranded on trains in Connelly Station yesterday.

They were on their way to the launch of the Dart's new Accessibility Pilot Programme, intended to promote Irish Rail's improved travel services for those living with disability."

(Blake Knox, 2018)

Sean O'Kelly heading up the *"A day in my wheels"* (O'Kelly, 2018) campaign and one of the unfortunate invited guests to have been caught up in the *DART Improved Accessibility Pilot* fiasco, is being consulted throughout all stages of this software project.

References

Bibliography: Rail, I. (2018). DART Improved Accessibility Pilot. [online] Irish Rail. Available at: http://www.irishrail.ie/access-dart [Accessed 5 Feb. 2018].

Bibliography: Firebase. (2018). Authenticate Using Google Sign-In on Android | Firebase. [online] Available at: https://firebase.google.com/docs/auth/android/google-signin [Accessed 8 Feb. 2018].

Bibliography: Api.irishrail.ie. (2018). [online] Available at: http://api.irishrail.ie/realtime/ [Accessed 28 Jan. 2018].

oguzbilgener. (2018). oguzbilgener/CircularFloatingActionMenu. [online] Available at: https://github.com/oguzbilgener/CircularFloatingActionMenu [Accessed 1 Feb. 2018].

Square.github.io. (2018). Picasso. [online] Available at: http://square.github.io/picasso/ [Accessed 3 Feb. 2018].

Bibliography: Blake Knox, K. (2018). Ross snubs rail access launch as wheelchair users stranded on Dart - Independent.ie. [online] Independent.ie. Available at: https://www.independent.ie/irish-news/ross-snubs-rail-access-launch-as-wheelchair-users-stranded-on-dart-36545469.html [Accessed 29 Jan. 2018].

Bibliography: (O'Kelly, 2018). A day in my wheels. [online] Facebook.com. Available at: https://www.facebook.com/adayinmywheels2016/ [Accessed 9 Feb. 2018].