

National College of Ireland

Bachelor of Science (Honours) in Computing

Software Engineering

2022/2023

Alexis Roi Pechon

X19358953

X19358953@student.ncirl.ie

Mobile Application Development

Technical Report

Contents

[Executive Summary 2](#_Toc134667308)

[1.0 Introduction 2](#_Toc134667309)

[1.1. Background 2](#_Toc134667310)

[1.2. Aims 2](#_Toc134667311)

[1.3. Technology 2](#_Toc134667312)

[2.0 System 7](#_Toc134667313)

[2.1. Requirements 7](#_Toc134667314)

[2.1.1. Functional Requirements 7](#_Toc134667315)

[2.1.1.1. Use Case Diagram 7](#_Toc134667316)

[2.1.1.2. Requirement #1: Accessing the Timetables 7](#_Toc134667317)

[2.1.1.3. Description & Priority 7](#_Toc134667318)

[2.1.1.4. Use Case 7](#_Toc134667319)

[2.1.2. Data Requirements 9](#_Toc134667320)

[2.1.3. User Requirements 11](#_Toc134667321)

[2.1.4. Environmental Requirements 12](#_Toc134667322)

[2.1.5. Usability Requirements 13](#_Toc134667323)

[2.2. Design & Architecture 14](#_Toc134667324)

[2.3. Implementation 14](#_Toc134667325)

[2.4. Graphical User Interface (GUI) 16](#_Toc134667326)

[2.5. Evaluation 18](#_Toc134667327)

[3.0 Appendices 18](#_Toc134667328)

[3.1. Project Proposal 18](#_Toc134667329)

[3.2. Objectives 18](#_Toc134667330)

[3.3. Background 18](#_Toc134667331)

[3.4. State of the Art 19](#_Toc134667332)

[3.5. Technical Approach 19](#_Toc134667333)

[3.6. Technical Details 19](#_Toc134667334)

[3.7. Project Plan 20](#_Toc134667335)

[3.8. Testing 21](#_Toc134667336)

# Executive Summary

The purpose of my project was to develop a Mobile Application that, once downloaded and installed by the user, would send a notification for the next available bus of their choice, while also enabling them to adjust when the next notification is going to appear while also keeping track of their current location, so it allows the user to find the next, and shortest, route to a bus stop.

The concept behind this project was that I would have different buses such as Bus Eireann and Matthews and be able to use an API that connects the application to a server that manages their timetable for the next available bus.

# Introduction

## Background

The reason why I undertook this project stems from a personal experience with keeping track of time, especially making sure I catch the next available bus on time, resulting in my main goal of making the lives of the user a little easier when it involves timetable management in terms of catching their next bus.

## Aims

As briefly mentioned underneath “1.1 Background”, I referred to how I aim to make the lives of the general users significantly less stressful by allowing my mobile application to handle the important timetabling issues about commuting on public transportation, specifically the use of busses and trains. Furthermore, the dual functionality of the app enabling the users to find the nearest bus stop or train station with the implementation of the Google Maps API, making it more time efficient for mobile users. Likewise, the application itself will also have an Irish Translation of the information being displayed as the main target audience are Irish Residents or Mobile Users, allowing an additional user accessibility to those who understand Irish.

## Technology

One of the core technologies I will be utilising in this project would be the Java Programming Language as well as using Android Studios as my means of developing a mobile application. Android Studio will act as my IDE for my project and a place where I can program the main technologies that will be featured throughout my mobile application. Since Android Studio utilises the Java Programming language, an SDK is required to build my project. Thankfully, within Android Studio, you will be given the option to choose from a variety of different minimum SDKs, and depending on the SDK that you have chosen, this would have a substantial impact on the project as well as the features that I plan to implement.

Currently, the project is being built on a Minimum SDK called “API 30: Android 11.0 (R)”. The reason I chose this version is that it would enable me to implement the major feature of my project by making use of the Java libraries that come with this version.

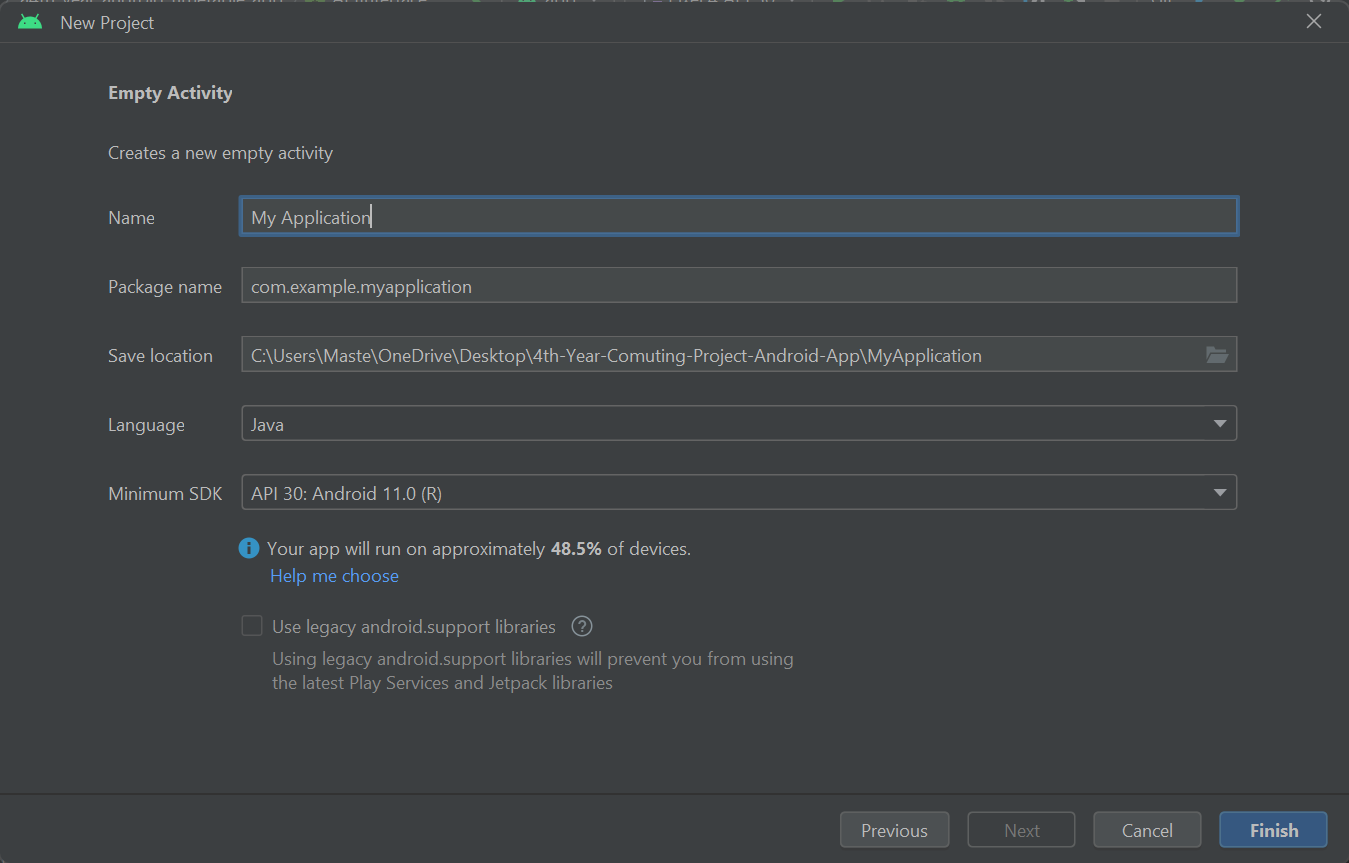


Figure 1: The screenshot displays the menu that allows the user to create their project. Inside of the Minimum SDK field I have chosen “API 30: Android 11.0 (R)”, which is what the current build of my project will be using.

Within Android Studio, it is mandatory to choose the appropriate version that best suit the project as the different version as well as the different SDKs will have different libraries, methods, or even API Calls that would be removed or be widely considered deprecated and unusable. Choosing the minimum SDK “API 30: Android 11.0 (R)”, enables me to add the technologies required.

Additionally, the other technology that I will be utilising in my project would be the use of APIs. As previously mentioned in my Executive Summary, I will be utilising the API technologies to obtain the information of the next available bus or train by extracting the information from the API calls and allowing my Java program to interpret and then display that information accordingly onto my mobile application.

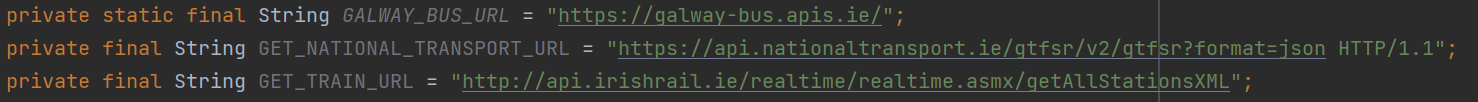


Figure 2: The following screenshot that was shown above shows an excerpt of code that was taken from one of my Java Classes that holds the link to the website that contains the API that contains the information that I will display on my mobile app. The links to the website are then kept inside of a variable which will be called throughout my application.

Following on from the topic of APIs, another technology would become essential for the development of the project as well as for the overall functionality of the mobile application. Java as a standalone programming language is unable to normally read the API calls within my project. There is a special type of technology that allows me to bridge the gap between Java and my HTTP API calls. This technology is called Retrofit.

In layman’s terms, Retrofit is simply a REST Client Library for Java as well as Android that allows a program to create an HTTP Request while simultaneously being capable of processing the HTTP response from a REST API. Likewise, the Retrofit Client isn’t just limited to reading and interpreting HTTP Requests, Retrofit could be also used to receive data structures in JSON, SimpleXML as well as Jackson.

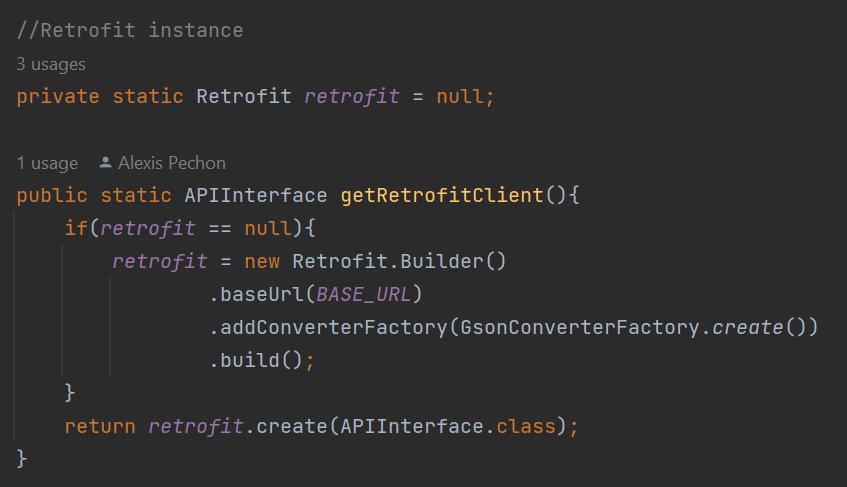


Figure 3: The screenshot shown above is an excerpt of code that was taken from my RetrofitClient Java Class in my application. What this code is doing is that it is simply creating a Retrofit Instance where I will be able to take in the information of the links that were store in variable, just as shown in Figure 2 of the report.

However, it is also worth noting that the Retrofit Client cannot work on its own, especially within the context of my Java application. Retrofit is a dependency that must be added to my Java project. The Retrofit Dependency must be added inside of the build.gradle file of the project before the entire application could access the RetrofitClient libraries.

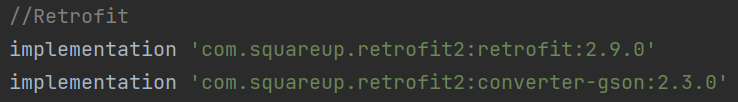


Figure 4: The screenshot shows the necessary dependencies that must be implemented inside of the build.gradle file of my Java project in order for the project itself to recognise the Retrofit libraries.

Furthermore, there was another technology that I will be using through the development of my project, and that is by using Git and GitHub. Both are essential throughout the project development as they help with keeping track of progress as well as changes that were being made to my application. As I am following the Agile Methodology, I will be frequently adding and removing different changes to my application depending on whether the feature is relevant or possible to implement.

In this case, these changes need to be documented and stored in a repository where the different changes could be monitored and documented for future reference. This is where GitHub and Git become useful assets in my application development. Git is a control system that monitors and tracks changes being made to any computer file. Likewise, Git could also be connected to GitHub where I could push changes remotely through the Git Console.

GitHub is effectively a code hosting platform in which code from various projects is stored and could be accessed through the Internet. The platform itself enables versioning control as well as collaboration between different people. This is needed for my project as there will be continuous changes being made to the different iterations of my project, hence is the main purpose of using GitHub as it allows me to create a repository to store my code where I can then monitor the different versions of the project as well as track different bugs within the software.

For context in my project, GitHub is especially necessary when it comes to creating two different branches where I can control the version and the different commits I have made to my repository. Currently, my repository has two branches, the main and devBranch. The main branch will contain the code that I initially started my project with while the devBranch contains the changes that I have made to the code while also removing features that were no longer needed or weren’t essential to the project.

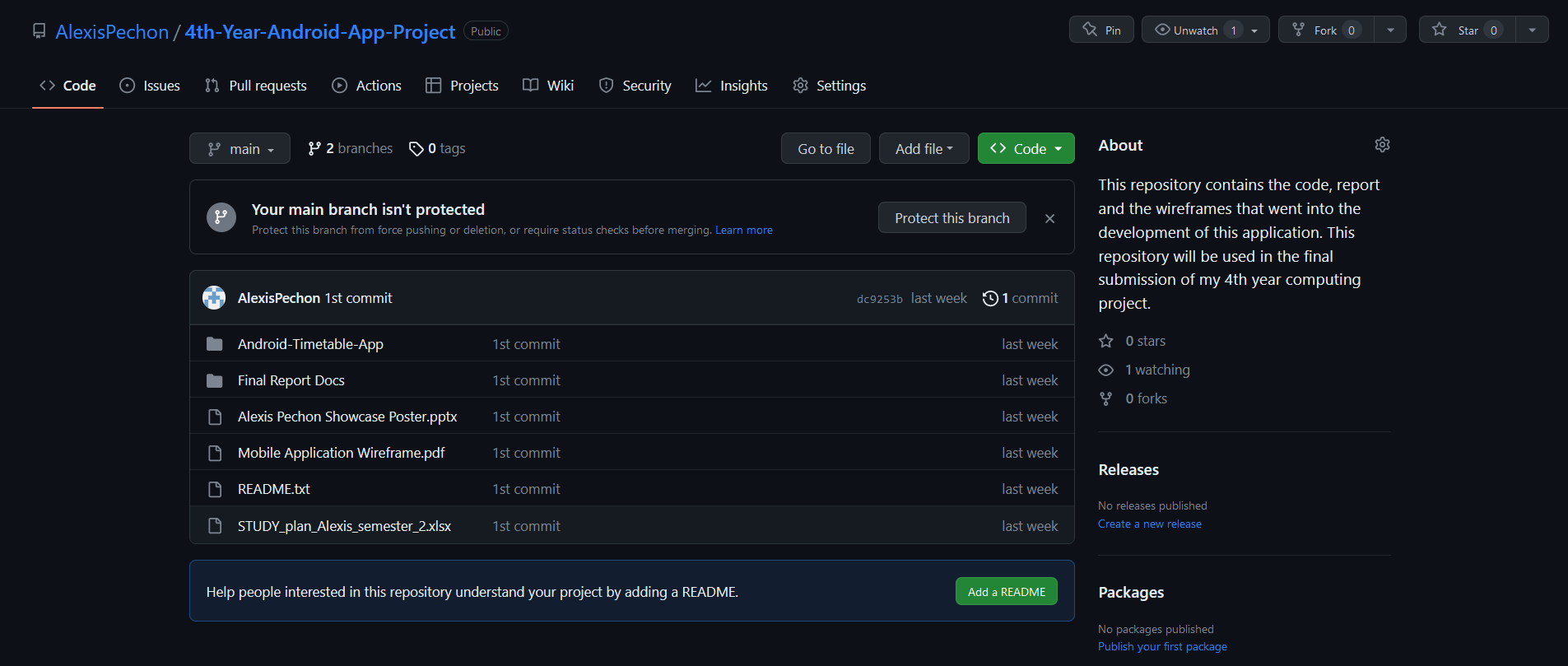


Figure 5: The screenshot above shows the current project repository that I have saved onto my GitHub. The screenshot is displaying the main branch of my project where the initial code of my project is saved in.

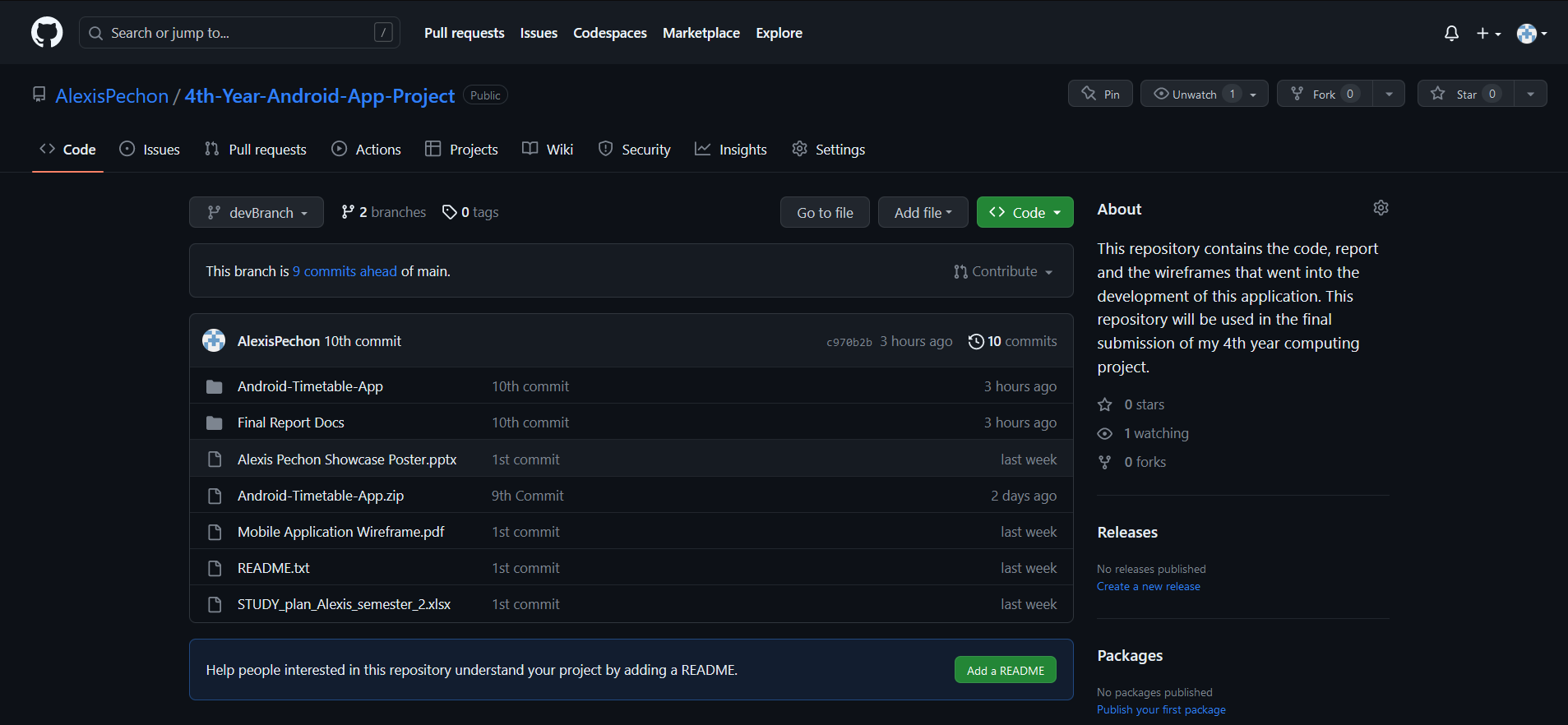


Figure 5.1: The screenshot above shows the devBranch in my Android Project GitHub repository, currently this branch controls and monitors the different changes that I have made to my code after I have committed them on Git. This branch documents these changes to my code and file.

In retrospect, the following technologies that were listed are all mandatory for the development of the project. However, using them offered a great challenge, especially since I lack adequate experience to correctly utilise them in my project. The first example would be choosing the minimum SDK that was required when first creating my mobile application. Using an outdated or older version of the minimum SDK would result in different libraries and methods being removed or changed, resulting in unintended problems relating to the version the project is being built on. This was an issue for me as I ultimately realised that the version my application was running on was outdated for the libraries and imports that I was using. This caused a lot of issues but was ultimately resolved by changing the minimum SDK version to Android 11.0 (R) and the problem was correctly handled.

Another challenge that I faced during my project was handling the technologies was trying to extract the information from the API calls. The API calls were created using JSON and they took the JSON Format. This is where the RetrofitClient Libraries become useful as they help interpret the JSON format in a way that the Android Studio could understand. However, after going through the development of the application I realised that the JSON format was too complicated for the program to understand which began to cause issues in which the information from the API wasn’t being correctly displayed on the screen. In this case, I had to create my customised API in which the JSON was simple enough for the program to understand, and hence be able to print the information onto the screen.

# System

## Requirements

## Functional Requirements

## Use Case Diagram

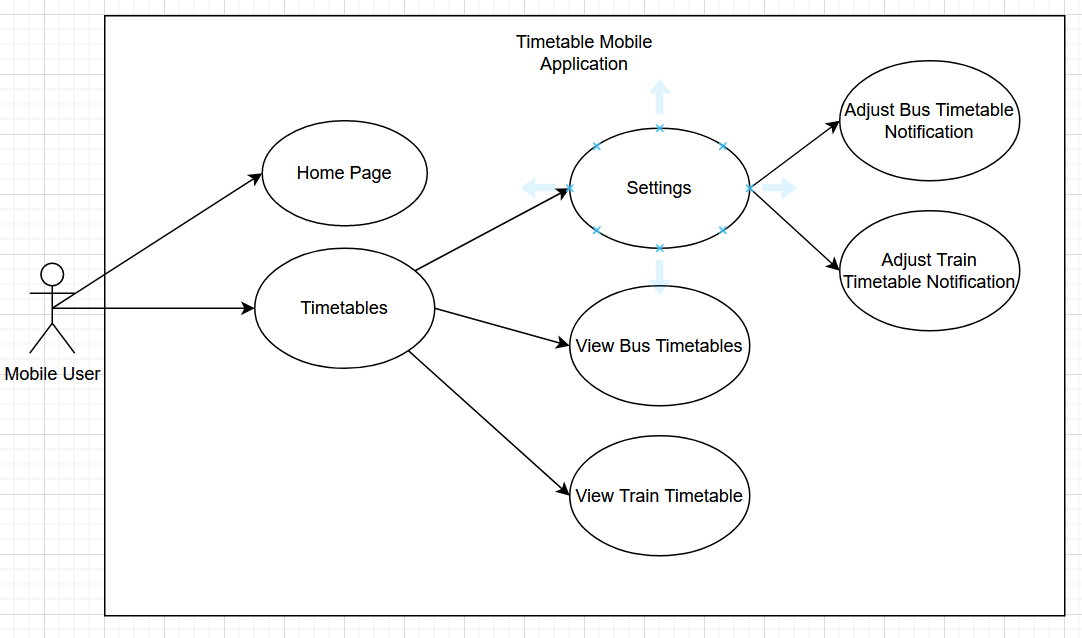


Figure 6: The screenshot above shows the Use Case Diagram for my project and how the user is expected to interact with the mobile application.

## Requirement #1: Accessing the Timetables

## Description & Priority

When the user first opens the application, they will be given two options where they can click between two tabs, one of which opens the main page of the application, and the other tab where they can have access to the different timetables that are available.

## Use Case

**Scope**

The scope of this use case is to illustrate the main functionality of my project when the actor opens the application for the first time. When they load into the application, they will be greeted with the homepage of the application which would show them the latest news or updates regarding the app, or the changes made to the timetables. Likewise, there will be a timetable tab where the user can view the timetables for the different trains or busses as well as have the option to change the time the application will send the reminder to the users.

**Description**

The use case diagram describes the interaction that the user will have when they are using my mobile application.

**Use Case Diagram**

**Flow Description**

**Precondition**

The application will begin to initialise when the app is downloaded on the phone.

**Activation**

The use case starts when the actor interacts or opens the timetabling application, when they do that, they will be promptly given two choices of viewing the homepage or being able to view the different timetables that the app currently supports. They are displayed in the form of interactable tabs that the actor can click.

When they clicked on the timetables section, they will be immediately given a table or list of the different times that a bus or train will arrive, giving them an idea of how they would adjust their push notifications. Likewise, the option to change when the notification can show up can also be adjusted in the timetables tab as well, making it have a dual purpose.

**Main flow**

1. The system initialises the application.
2. The actor opens the mobile app.
3. The system shows the home page.
4. The actor chooses between either looking at the homepage or viewing the timetables.

**Alternate flow**

A1: Timetables

1. The system waits for the user to select the “Timetables” tab.
2. The actor selects the timetables option.
3. The user chooses to view the timetables or change the push notification.

**Exceptional flow**

E1: Settings

1. The system awaits the user’s input.
2. The actor then selects the settings option in the timetables tab.
3. The user can customise the time in which the app will notify them or even select the bus or train they wish to be notified for.

**Termination**

The application will simply terminate when the actor exits the app. However, the push notification feature within the app will always remain active even during an idle state.

**Post condition**

When the user opens the mobile app, the application itself will wait for the actor’s choice of whether they want to view the home page. Timetables, or want to change the notification settings. If the actor decides to select the Homepage tab, they will be sent to the main page, similarly, if they chose to open the timetables, the app will wait for their response.

## Data Requirements

Since my project utilises API Calls, the main source of data in that scenario would be getting the information for the timetables on websites like Bus Eireann and Irish Rails. This would then be processed in the timetabling section of my application where it would result in the user being able to customise and adjust their preferred time for when they wish to be notified of the next available train or bus.

Likewise, I also aim to utilise the Google Maps API, which in turn allows the user to find the shortest possible route to their destination, adding a similar GPS-like functionality to my project. Data in this situation would have to make use of the location data so the application can track what is the most time-efficient route the user can take.

As previously mentioned under the Technology section of the report, I mentioned how I need to utilise APIs in my application for it to functionally work. However, upon creating the project there was a problem regarding the API calls not extracting the information that was required, in which case, I had to create a custom API using JSON to get the information that I need for the application to work. In this scenario, I would need the dependencies for Retrofit for my application to be able to read the API calls as well as the dependencies for Espresso which is also required for testing the application.

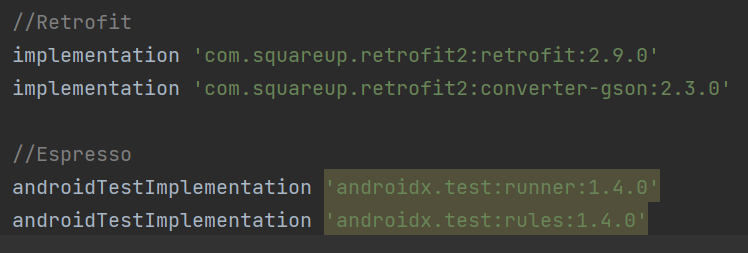


Figure 7: The screenshot above shows the dependencies that need to be installed into my application. This was taken from the build.gradle file and after syncing the gradle.file with the rest of my application, I was then able to use the RetrofitClient Libraries which was required in getting and putting the API information, and data, on the mobile app.

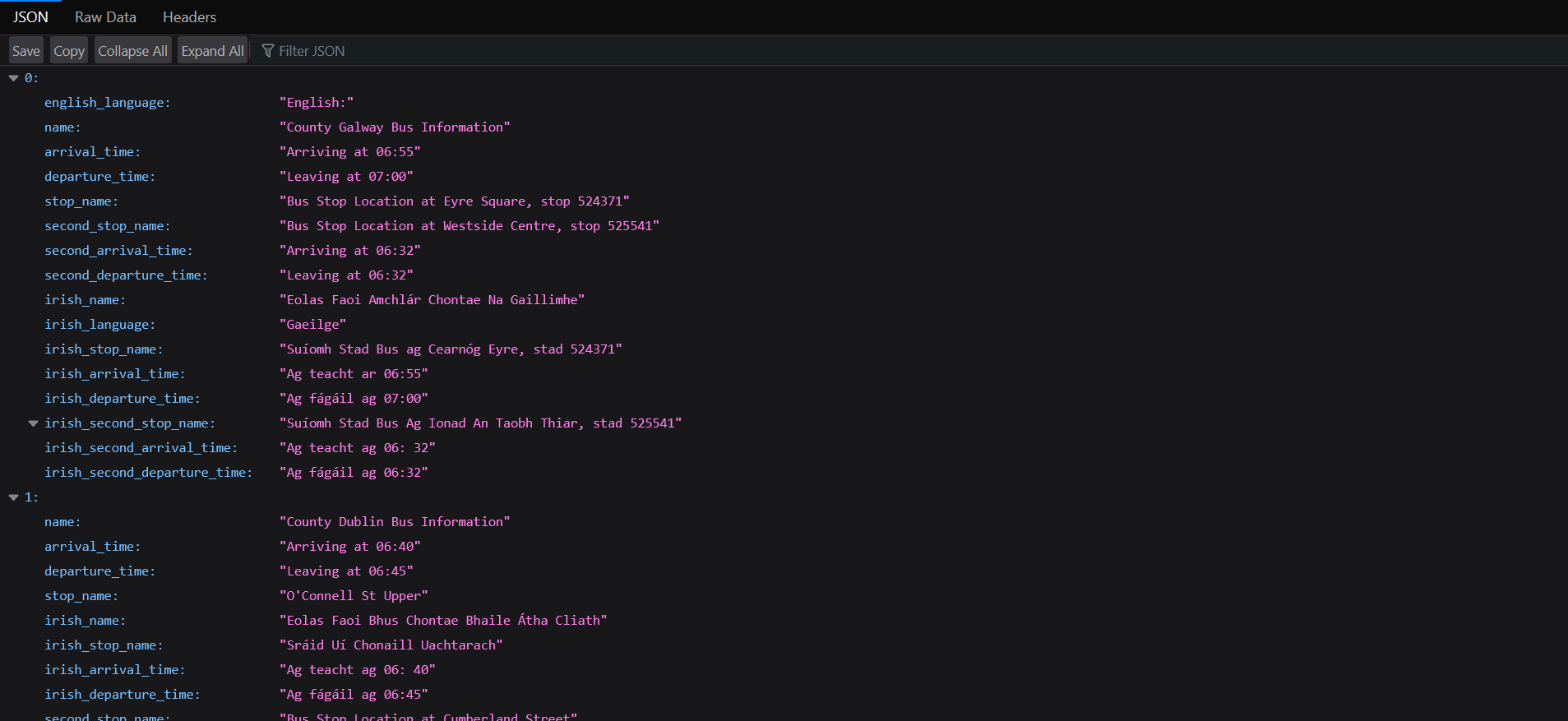


Figure 8: The screenshot above is taken from a custom JSON that was created to put the necessary and relevant information about each bus timetable onto the screen. This is an essential data requirement as this contains all the information the application will need to post the details to the user via the mobile app.

## User Requirements

On a general level, the application itself should be expected to open and direct the user to the home page of the application where it will give them a short message explaining to them how to navigate and use the application. The tabs in the application will serve as a user-friendly way of switching between the main page and the timetables page where the user will adjust the push notification settings according to their preferences.

Furthermore, other elements need to be considered when thinking about the User Requirements. The major component of that would be the user experience. The purpose of my project was to develop a mobile application where the user can view the timetables efficiently and smoothly without any problems. The idea behind this was to create an app that allows the user to scroll through the different timetables naturally without implementing any features that may seem jarring to the mobile user.

This means that the information displayed on the screen must appear as the user is scrolling through the application. This means there should be no delay between the user scrolling through my application and the information being displayed on the screen. This will make the user experience more natural as mobile users would be normally used to seeing information appear on screen as they scroll through their phones.

How this could be implemented into my project would be using recycler views, meaning that the information will be displayed all at once on the screen as the user scrolls through to find the information that is required.



Figure 9: The screenshot above displays an excerpt of code from the MainActivity class of my Java Project, this excerpt of code is creating the recycler view model when the project is executed, allowing for the mobile app to have the scrolling functionality.

## Environmental Requirements

The application itself should be able to read API calls from sites that contain the bus or train timetabling schedules so that my project can utilise it and allow my application to send a reminder or a push notification letting the user know about the next available bus or train.

Another added functionality is that the application will utilise the Google Maps API to let them know of their current location and what would be the closest route that they can take to promptly reach their destination.

Referring to what I mentioned in the 1.3 Technology section of the report, I explained how that there was an additional dependency that I would need to consider if I were to include the RetrofitClient Library in my computing project. As part of the Environmental Requirements for the project, those dependencies must be implemented into my build.gradle project file and then it has to be synced up to the rest of my project. Once that has been completed, then I have full access to the Retrofit Client Helper Library which enables my Java application to read and interpret the HTML API Calls.

Another Environmental Requirement that must be considered is that my application should have access to the Internet. This is arguably an essential step in my project as the entire mobile application centres around the HTML API Calls, which require the Internet to access the information. If the internet access in my application isn’t enabled, then my app wouldn’t be able to display the information on the screen since the program won’t be able to read the API Calls.

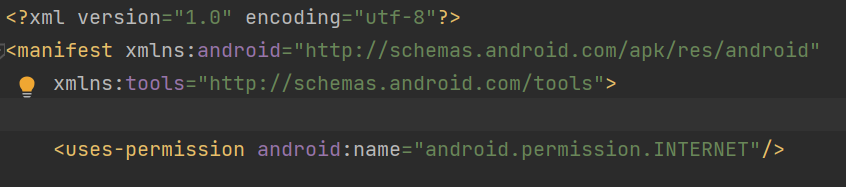


Figure 10: The screenshot shown above here is an excerpt of code that was taken from the AndroidManifest.xml file in my project where there is a code that enables the mobile application to connect to the internet, which is necessary for the HTML API Calls.

## Usability Requirements

Due to the application being developed using “Android Studio”, one of the technical limitations of my project is that it could only be accessed by users who are using an Android Device, meaning that users using an iOS device wouldn’t have the ability to download the application normally or would work in the same way it would on a standard Android device.

However, the GUI and resolution will be curated to fit any smartphone device while also being able for Android Users to access it and customise the push notification to match their desired preferences.

As the development of my project continued through the passing months, some features were added and removed to accommodate for a smooth and seamless user experience where the mobile user can use the application naturally without any jarring experiences when using the app. One such feature that was removed was the notification which was later replaced with a recycler view where the user can scroll through various timetables that were supported by the app. This was the experience a lot more natural since people could just scroll through the app as opposed to clicking tabs and then scrolling for the timetables, which was a former feature in my first iteration of the project.

This means that people can simply click on the app and then search for the timetables by scrolling through the different existing timetables on the app.

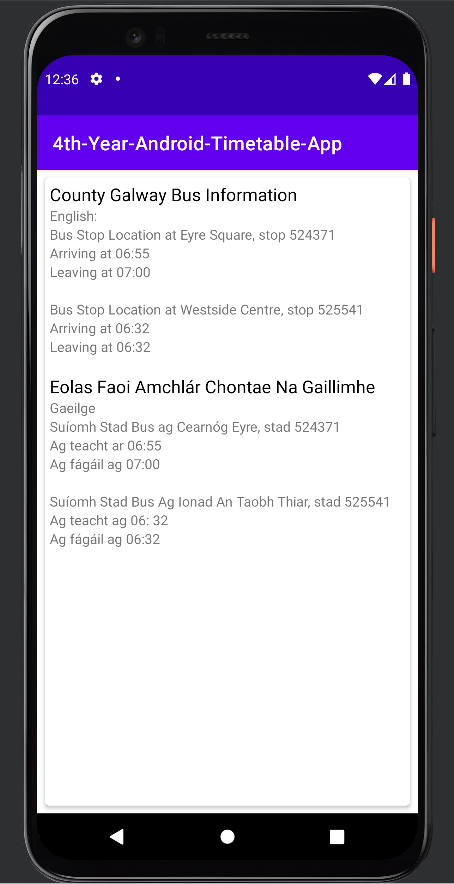


Figure 11: The following screenshot shows the first timetable the user will see when they log in. As they scroll down, they will be shown different times for different buses.

## Design & Architecture

The core design and overall architecture for my project will contain the two following layers which I deem the be the most important components in the overall project. The first layer I would consider is the presentation layer, this is the front-end section of my mobile application and handles the interfaces that the user will interact with. This is crucial in the project as I aim to make the GUI accessible to all by making it comfortable for the users to look at.

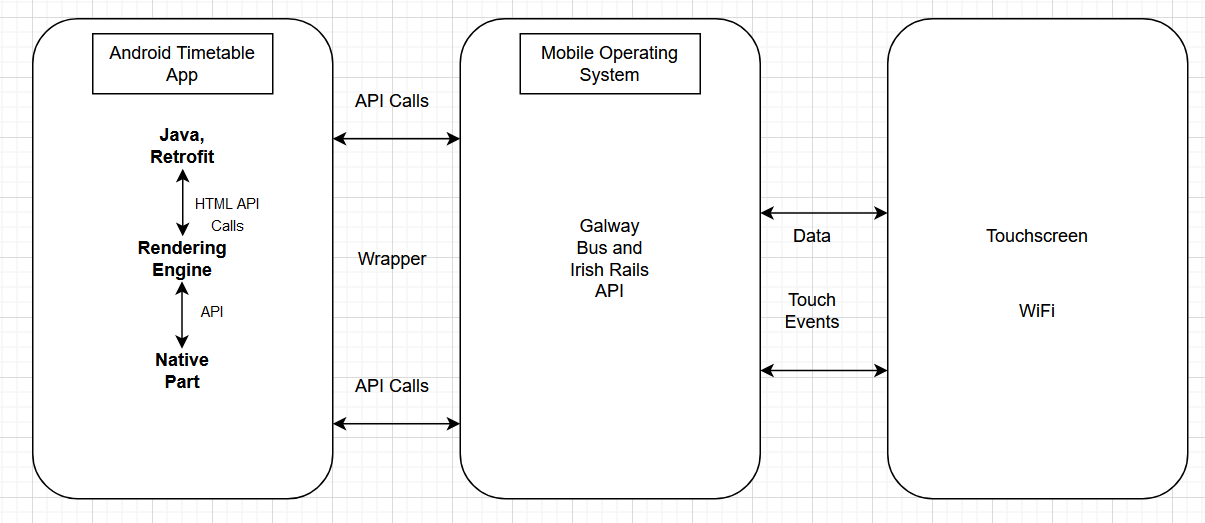


Figure 12: The screenshot above shows the Design and Architecture of my application and how I expected to run and operate.

## Implementation

The first feature I had to implement before adding anything else to the project, was the basic Graphical User Interface, or GUI, that the mobile user will be interacting with. During the development process, I had to make sure the GUI was user-friendly and accessible to all. After acknowledging those major components of my project, I decided to utilise Android Studio’s GUI templates which allowed me to customise and change the different aspects of the interface to match the standards of the mobile users.

After creating the GUI, I implemented a Java class that allowed me to edit and program different aspects of the application, allowing me to add text or change different components to how the overall app looks. Likewise, another thing to consider when implementing features into the application is that the features need to make sense or have some kind of purpose. This is following the principle of Keeping it Simple Silly or the KISS Principle for short. What this means is that I should try to keep the application straightforward by only adding features that contribute to the overall quality of the application. Throughout the report, there will be features that have been removed as they were too complicated to implement and that they provided no value to the application.

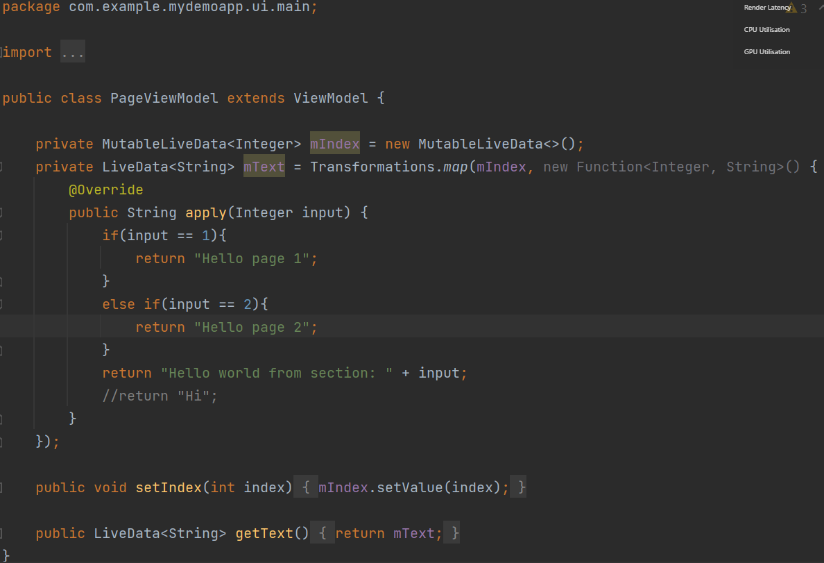


Figure 13: The screenshot above shows the Java class that handles the text and information to be displayed to the user.

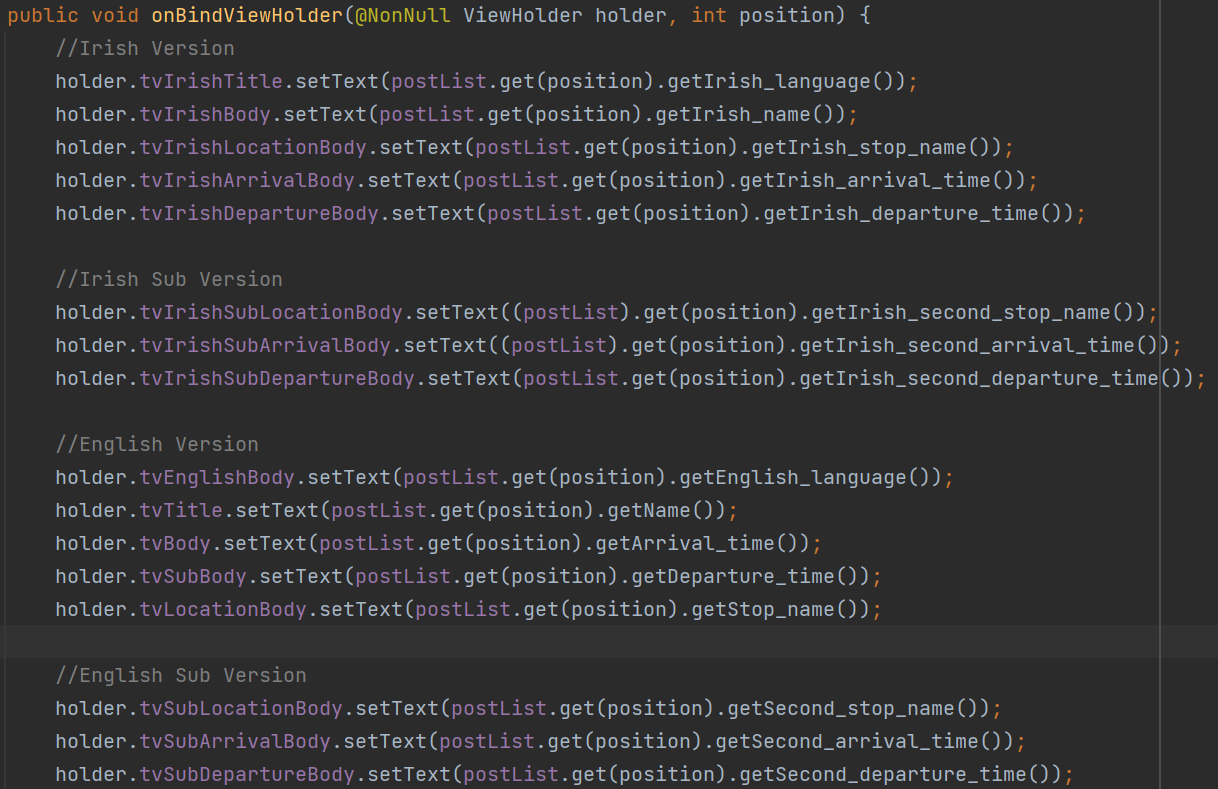


Figure 14: The following screenshot above shows an excerpt of code from the PostsAdapter class in my Java Project. In the latest version of the project, this is how information is going to be displayed onto the mobile screen as it is taking the information from the API. This screenshot is taken from the latest version of the code.

## Graphical User Interface (GUI)

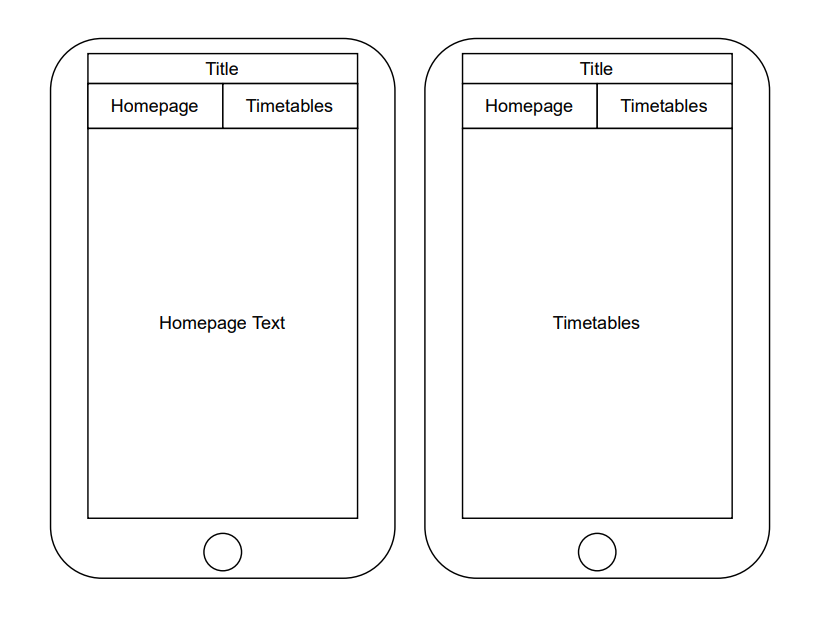


Figure 15: The following screenshot above shows the oldest iteration of the GUI for my mobile application. It shows how there were going to be tabs that the user would be able to select and information would be displayed accordingly.

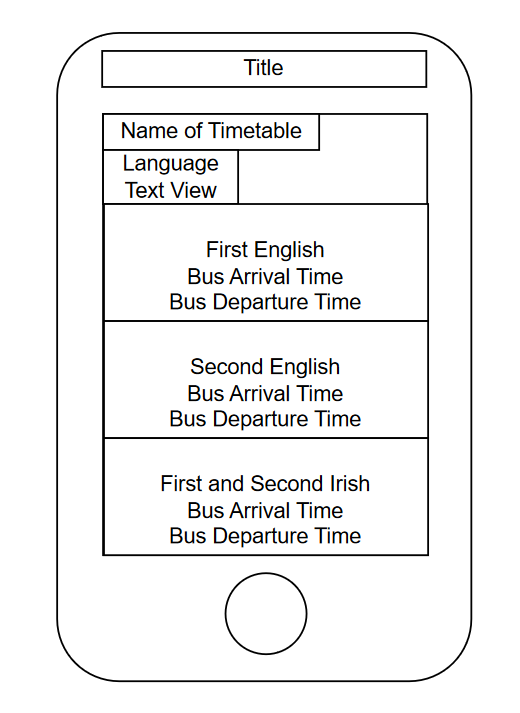


Figure 16: The following screenshot shows the wireframe for the current version of the GUI for the mobile application. Compared to the older iteration of the GUI, this version features a scroll feature so the information will be displayed as the user scrolls through the app.

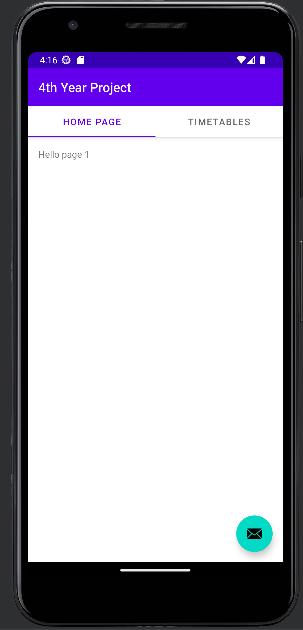




Figure 17: The two screenshots shown previously were all taken from an older build of the project where the user could select between two tabs to view the homepage and the timetable section of the app.

## Evaluation

# Appendices

## Project Proposal

## Objectives

The purpose of my project was to develop a Mobile Application that, once downloaded and installed by the user, would send a notification for the next available bus of their choice. The concept behind this project was that I would have different buses such as Bus Eireann and Matthews and be able to use an API that connects the application to a server that manages their timetable for the next available bus.

The main goal of my project, and one that I aim to achieve, is to build an application that sets out to make the lives of the user a little easier when it involves timetable management in terms of catching their next bus. This application could have a general audience but is more targeted to adults or young adults who must rely on public transportation to commute to work or school.

# Background

The idea for this project came from a personal experience where I accidentally missed my bus going into college one day due to a major misunderstanding between how I read the bus schedule, to how I would get ready before then. This cost me to miss a large portion of my classes and as a result, had to spend a lot of time trying to catch up to make sure I didn’t miss out on anything important.

However, this was only my personal experience of my poor timetable management. Another source of inspiration, and one that made me choose to undertake this project, was how I could potentially make the lives of other people who went through the same problems as I have and make it a little easier and more convenient so all could mutually benefit from the idea.

As previously mentioned, the objectives of this project were to develop a Mobile Application that a user could download and be notified of the next available bus. The first step I could take before I could start my project was to do some research on Mobile Application Development and try to understand the basics of creating one. This could be done by watching YouTube videos or looking up guides on Google.

# State of the Art

Though some applications exist on the internet and the app store that can help with timetable management, how I aim to set my project apart from the rest of the pre-existing apps on the market was that I would implement a feature where the user could choose whatever bus they’re taking, and the location that they’re in, and the application itself would be able to recognise that and be able to push a notification accordingly.

In layman’s terms, when the user inputs their location and the desired bus that they would wish to take, then the application would recognise that and send a notification for the next available bus that they choose based on the bus’s current timetable. This means that the user would only have to put in the least amount of effort in telling the application what they want, while the application itself does all the hard of letting the user know when the next bus is available.

Normally other apps on the market would take the form of a calendar, but for my application, I aim to make it a push notification application where they could read the notification on their screen and be able to understand when the next available bus is.

# Technical Approach

The first step before I would undertake this project development is to do some research on the basics of Mobile Application Development. Since I am doing this project with little to no prior knowledge of mobile application development, I will need to spend a couple of weeks trying to understand the syntax and the concept of Mobile Application Development. This could be using Google or YouTube videos, but the first step before building my project would be to understand what is required and have all the basic tools installed.

Identifying the requirements will come later in the development of my project, but it could be easily identified during my research where I would be spending time on Google on what must be installed or used before creating the project. However, the basics of developing an app, such as the IDE, could be the main prerequisites in terms of the main requirements of the project. Furthermore, additional tools and dependencies would all differ depending on the amount of research I put in. In this case, those are the different requirements for the project I must integrate to assure a functional final product.

# Technical Details

During my research on what should be installed and implemented before taking on my computing project, I discovered that one of the best languages I could use before Mobile Application Development would be Java. Following that, I would have to install an IDE that can support Java while also downloading the most recent JDKs for the programming language to run normally without any issues.

In terms of algorithms, I would have to look up an API that would be able to pull information about the timetables for each bus that I have selected and use that API to push a notification for when the next bus is going to be available.

# Project Plan

**October**

During the month of October, when the project was first announced to us, my time was spent looking into various project ideas and testing to see whether they would work or not. This took the form of various discussions with fellow peers and researching previous projects from other NCI Students using the NCI website. When the time came when I decided upon what project I wanted to take on, I began researching basic Mobile Application Development tutorials on YouTube.

**November**

During this month, I continued to research basic ideas for what I could base my Mobile Application on, but that was when I decided to turn it into a mobile application that assists users in becoming a bit more time efficient when it came to getting to the bus stop or train station on time.

Around this month was when I was assigned to my supervisor, to which I began attending meetings where we discussed what was required of my project and how I could expand and even build upon the current existing idea to make it more innovative and creative.

**December**

By this month I should have a working prototype working while having the bare basic functionalities of my project implemented. For example, the GUI and the app’s interactivity should have been completed around this time in time for the mid-point presentation. After I have completed the first mobile application prototype, I began experimenting and implementing features like adding basic texts to different pages as well as checking if the UI is user-friendly.

# Testing

There are multiple ways I can perform different tests on my mobile application app. The first and simple test would be to test the GUI’s interactivity. This could be easily done through the same IDE on Android Studio. With the use of Android Studio’s built-in emulator, I can run the app seamlessly as if it was on a physical mobile phone. Once I have done that, I can interact with the app after opening it. By clicking through the different tabs and other elements on the page to see if they’re responsive or react to basic user input.

Another form of testing that I can perform in a project would be simply checking if the APIs are working and fetching the information as intended. One such way could be trying to print the timetables onto the screen of my application or by checking if the APIs themselves are online and working.

A third way of testing the app to see if it works is by having an extra or physical phone and downloading the app there. This method can be used to check the compatibility of certain phones, devices, and even the operating systems of each phone. How this would apply to me during my testing phase is that I could get different participants to help. I would run the app in Android Studio to see if it works there, then download it onto my physical phone, if both tests pass, then I can do it again on a third participant. If all three phones can successfully run the app, then the app is functioning as intended.

In the context of testing the functionality of my app, there is a feature in Android Studio where it allows me to create a folder and a Java class where I can create different tests to ensure that my application is run as it was intended to.

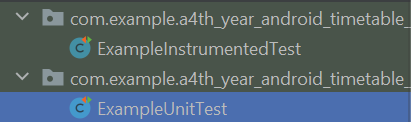


Figure x: The screenshot above displays the folders and different unit tests that I have prepared as part of the Testing Elements of my project.



Figure x: The following screenshot shows that I have set Android Studio to run the Java Test Class that I have created in the project testing folder.

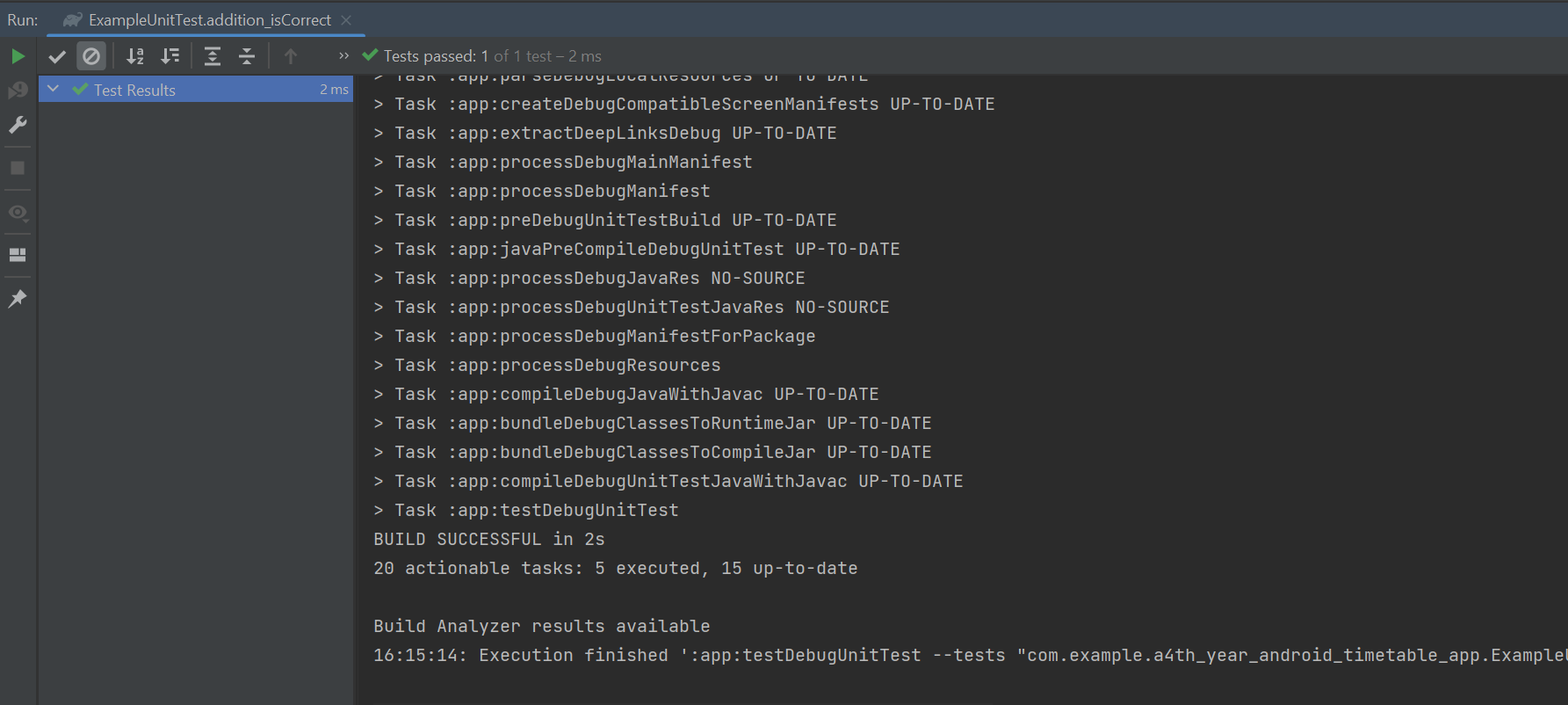


Figure x: The screenshot shown above shows the result of the test class being executed on Android Studio. The console is printing out the results of the test, in which case the test was successful.

However, within the context of testing my application using Android Studio, it is worth acknowledging that the project itself is heavily reliant on the androidx imports to maintain its functionality and be able to read the API calls so it could be displayed on my mobile application. This makes testing an added challenge on Android Studio as the androidx imports will limit the ability for me to perform tests on my application through conventional methods. However, some workarounds enabled me to perform certain tests even while using the androidx imports. This could be achieved through the Espresso framework.

As previously mentioned, Espresso is simply a framework that is available on Android Studio, and its core functionality is to enable software developers to create an automated test case for the User Interface of the application. This is essential for my project as it is required for me to test the functionalities of my project to ensure everything is running as intended before its submission.

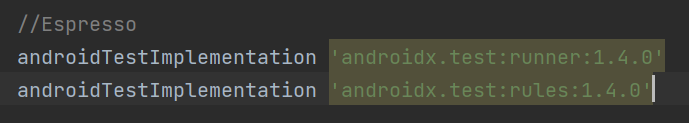


Figure x: The screenshot above shows an excerpt from my build.gradle file on Android Studio. These dependencies must be implemented into the gradle file, otherwise testing using Espresso wouldn’t be possible.