



E-Leap Interim Report

**DT211-C
BSc in Computer Science (Infrastructure)**

Sampanna Pathak

D18124115

Eoin Rogers

School of Computer Science
Technological University, Dublin

09/12/2022

Abstract

The aim of this project is to make an application that helps users to pay for their Bus fares from their Android devices. E-Leap will be built using Android Studio as its Framework, with Java as the main programming language. The Database that the application run on will be Firestore Database. Also, Python script will be used to communicate between the client and the server, while Stripe payment gateway will be integrated into the Android Studio to make payments and GTFS (General Transit Feed Specification) API will be used to get Live route information.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

*Sampanna Pathak*_____

Sampanna Pathak

09/12/2022

Acknowledgements

I would like to thank Eoin Rogers, my project supervisor, for the continued guidance and valuable advice throughout this project

I would also like to thank Jonathan Mccarthy for his guidance on the documentation process.

Table of Contents

1. Introduction	9
1.1 Project Background.....	9
1.2 Project Description.....	10
1.3 Project Aims and Objectives	10
1.4 Project Scope	10
1.5 Thesis Roadmap	11
2. Literature Review	12
2.1 Introduction	12
2.1.1 What is NFC?.....	12
2.1.2 A Brief History of NFC	12
2.1.3 NFC in public transportation in Ireland.....	12
2.2. Alternative Existing Solutions to Your Problem	13
2.2.1 TFI Leap Top Up.....	13
2.2.2 TFI Real Time	14
2.3. Technologies you've researched.....	16
2.3.1 Operating System.....	16
2.3.1.1 Android	16
2.3.1.2 IOS.....	16
2.3.1.3 Selection (Android)	17
2.3.2 Database	18
2.3.2.1 PostgreSQL.....	18
2.3.1.4.2 Oracle.....	18
2.3.1.4.3 Selection (Firestore Database).....	18
2.3.3 Programming Language	19
2.3.3.1 Swift	19
2.3.3.2 Kotlin.....	20
2.3.3.3 Selection (JAVA)	20
2.3.4 Back-End.....	21
2.3.4.1 Node.js	21
2.3.4.2 Spring Boot.....	21
2.3.4.3 Selection (Python using RESTful API)	22
2.3.4.3.1 Python.....	22
2.3.4.3.2 RESTful API	22
2.3.4.4 Selection (Firebase).....	23
2.4 Application Program Interface.....	23

2.4.1 GTFS	24
2.4.2 Stripe Payment Gateway	24
2.5 GIT	24
2.5 Existing Projects	24
2.5.1 NFC Based Smart Urban Public Bus Transport Payment System.....	24
2.5.2 Driving Instructor Assistor.....	25
2.6. Conclusions	25
3. System Design	25
3.1 Software Methodology	25
3.1.1 Waterfall Methodology.....	25
3.1.2 Scrum Methodology.....	26
3.1.3 Selected Methodology (Agile Methodologies)	27
3.2 Design and Architecture	28
3.2.1 Overview of System	28
3.2.1.1 Route Process.....	28
3.2.1.2 Log In.....	29
3.2.1.3 Top-Up	29
3.2.1.4 Tag On	29
3.2.2 Front End.....	30
3.2.2.1 Use- Case Diagram	30
3.2.2.2 High Level Wireframe	31
3.2.3 Database	33
3.2.4 Back- End.....	34
3.2.4.1 Sequence Diagram	34
3.3 Conclusions	35
4. Testing and Evaluation	36
4.1. Introduction	36
4.1.2 System Testing Methodology	36
4.1.2.1 Black-Box Testing	36
4.1.2.1 White-Box Testing.....	36
4.2 Plan for Testing	37
4.2.1 Testing tools from Android Studios	37
4.2.1 Unit Testing	38
4.3 Plan for Evaluation	38
4.4 Conclusion.....	39
5. Prototype Development	39

5.1 Introduction	39
5.2 Prototype Development	39
5.2.1 Firebase Authentication.....	39
5.2.2 NFC Tag Reader.....	41
5.2.3 GTFS API	43
5.4 Conclusions	43
6. Issues and Future Work	43
6.1. Introduction	43
6.1.1 Implementation of NFC cards	43
6.1.2 Implementation of routing process	43
6.1.3 Time Constraints	44
6.2 Future Work	44
6.2.1 GANTT Chart	45
Bibliography	46

Figure 1: An overview of how NFC tag works (Bi, 2013)	9
Figure 2: The Transparent NFC solution (Dorgan, 2017)	13
Figure 3 Leap top-up UI.....	14
Figure 4: TFI Real Time UI menu	15
Figure 5: TFI Real Time app UI	15
Figure 6: All the tools available in Firebase (Android Studio, 2017).....	19
Figure 7: Java Programming model. (Mandula, 2018).....	21
Figure 8: Node JS single thread operation (Capan, 2013).....	21
Figure 9: Spring Boot multi thread operation (Garai, 2022).....	22
Figure 10: Rest API (Benharosh, 2018)	23
Figure 11: Firebase authentication workflow	23
Figure 12: Waterfall Methodology (Hughey, 2009).....	26
Figure 13: Scrum Methodology (PM Partners, 2021)	26
Figure 14: Agile Methodology (Dreissigacker, 2022)	27
Figure 15: Technical High-Level Architecture	28
Figure 16: Menus User Case Diagram	30
Figure 17: NFC Tap User Case diagram	30
Figure 18: GTFS API User case diagram	31
Figure 19: Log in Page	31
Figure 20: Sign Up page	32
Figure 21: Home page	32
Figure 22: Top Up page	32
Figure 23: Route Page	33
Figure 24: ERD.....	33
Figure 25: User table.....	34
Figure 26: Sequence Diagram for Routing process.....	34
Figure 27: Sequence Diagram for Top Up process.....	35
Figure 28: Sequence Diagram for Tag on Functionality	35
Figure 29: Black Box Testing	36
Figure 30: White- Box Testing.....	37
Figure 31: Firebase authentication enabled.	40
Figure 32: Authenticated user in the firebase console.....	40
Figure 33: Authenticating user.....	40
Figure 34: Registering User	40
Figure 35: Checking if user is logged in.....	41
Figure 36: Register and Log In UI	41
Figure 37: Reading a Tag Android Studio-1.....	42
Figure 38: Reading a Tag Android Studio-2.....	42
Figure 39: Example of the UI for the NFC tag reader.....	42
Figure 40: Connecting to GTFS API and dumping the JSON data in a JSON file.....	43

1. Introduction

The aim of this project is to make an application that helps users to pay for their Bus fares from their Android devices. E-Leap will be built using Android Studio as its Framework, with Java as the main programming language. The Database that the application runs on will be Firestore Database. Also, backend server written in Python will be used to communicate between the client and the server, while Stripe payment gateway will be integrated into the Android Studio to make payments and GTFS (General Transit Feed Specification) API will be used to get Live route information.

Users can make 2 types of accounts, a “Student” account and “Adult” account. Depending on the type of account, a certain fixed amount of payment will be deducted from their account. The users can also top up their account, which will be implemented through Stripe payment system. It will also include features where the users are able to see the bus routes depending on the bus number or stop number. The E-Leap application aims to utilize the current Android technology in combination with Near field communication (NFC) chips to make it easier for users to pay for their fares.

1.1 Project Background

E-Leap is a project that aims to utilize this NFC technology to pay for transportation fares. National Transport Authority (NTA) is planning to launch a similar application in production in Ireland to pay for Transportation with a digital card. (Murray, 2018) While NTA is planning to deploy this solution into production, E-Leap is a demo version of a similar solution which aims to implement a simpler fare and ticketing system.

Near-field Communication (NFC) is a set of communication protocols that allows communication between 2 electronic devices over 10 cm less. (Madlmayr, et al., 2008) It is an evolution of an already existing technology called Radio Frequency Identification (RFID). Both the NFC and RFID works on the principle of inductive coupling. By passing an electric current through a coil, the reading device generates a magnetic field. When a tag (which has its own coil) is brought close, the field creates an electric current inside the tag without using any cables or physical touch. Once the reader and tag make initial contact, any stored data on the tag is wirelessly transmitted to the reader.

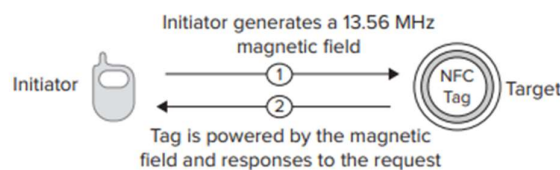


Figure 1: An overview of how NFC tag works (Bi, 2013)

Two billion NFC-enabled smartphone devices are in use today (IHS). In other words, 20%+ of the world's population have access to NFC. This technology has been an emerging technology in the market and have been utilized in several different projects.

1.2 Project Description

E-Leap will be an Android application made in Android Studio so the users will first need to make an account either a student or an adult. After they have an account, they need to top up their account using Stripe payment gateway where they will be able to pay through their bank cards. After topping up the users can pay for their fares by tapping on a pre-configured NFC tag which is for different agencies of bus namely Dublin Bus, Bus Éireann etc. Upon tapping, a fixed amount of fare will be cut off from their account balance depending on what type of account they have and what agency it is. The account page will display the user's information, including their details and balance. Additionally, the user can check the route of a bus and more details on the route details page which will work by getting live information from the General Transit Feed Specification - GTFS-Realtime API handled by the National Transport of Ireland authorities.

1.3 Project Aims and Objectives

The Overall aim of this project is to build an Android Application that makes use of NFC technology and helps the user to pay for their fares more conveniently through their handheld mobile device with the Android Operating System.

Some milestones along the way to achieve the aim

- Evaluate 2 other Applications finding Pros and Cons of those applications.
- Identify an appropriate design methodology.
- Research and identify all the appropriate technologies needed to develop the mobile application.
- Create and design Use-Case diagrams, Wireframes, and other necessary system designs of the application.
- Create necessary prototypes of each functionality.
- Implement a user-friendly front end with necessary functionality on the chosen framework.
- Implement the backend from the chosen technology.
- Implement the backend as per the required functionality.
- Continue producing prototypes to progress the project.
- Test and evaluate each working prototype.
- Document all the processes and the finished application.

1.4 Project Scope

The project's aim is to build an Android application that helps the user to create an account, top up their account, pay through their phone and search for routes. This will be achieved by focusing on Front- End which will be developed through the Android Studio framework using Java as the programming language. Firebase authenticator will be used to authenticate the user which will work like a backend server.

The user will be able to pay for their fare using the NFC chip on their phone by tapping a preconfigured NFC tag. Also, the Stripe gateway will handle the top-up part of the application. Firestore Database will be used as the Database and the GTFS API for route information which will be handled by the Backend server programmed in Python.

1.5 Thesis Roadmap

This section will provide an overview of what the following chapters are about.

1. Literature Review

The purpose of this chapter is to examine articles and conduct research on why a particular technology is being used and the options that were available.

2. System Design

This chapter provides the software methodologies and design of various UML diagrams.

3. Prototype Development

This section provides a rough sketch of both the front end and back end of the application will be implemented.

4. Testing and Evaluation

This section provides the plan for testing and evaluation to be conducted.

5. Issues and Future Work

This section discusses the problems that may arise while developing the application and outlines the work that will need to be done moving forward.

2. Literature Review

This chapter will introduce the brief history of the NFC card, its evolution throughout the years, and how it is used in the current Transportation system of Ireland and other countries.

The chapter will also investigate similar technologies currently available in the market and the number of possible technologies that can be used to build and implement this project. Lastly, existing journals and research papers on similar technology will be evaluated.

2.1 Introduction

2.1.1 What is NFC?

NFC or Near Field Communication is in simple terms a set of communication protocols that allows the transfer of data wirelessly over 2 technologies that are in close proximity.

NFC card is an evolution of RFID (Radio Frequency Identification) which is based on the principle of inductive coupling via induced magnetic fields between transmit and receive coils (loop antennas). Thin NFC sheets made of soft magnetic materials are inserted between antennas and the metal case of wireless gadgets, such as mobile phones or tablets, to reduce the degradation of antenna gain and radiation efficiency due to the generation of eddy currents. (Lathiya & Wang, 2021)

2.1.2 A Brief History of NFC

NFC is rooted in RFID (Radio Frequency Identification) which uses radio waves to communicate with passive or un-powered tags.

- May 17, 1983: The first patent to be associated with the abbreviation "RFID" was granted to Charles Walton.
- December 8, 2003: NFC was approved as an ISO/IEC standard and later as an ECMA standard.
- 2004: Nokia, Philips, and Sony established the NFC Forum.
- 2009: NFC was first used in transportation by China Unicom and Yucheng Transportation Card in the tramways and bus of Chongqing on 19 January 2009, then implemented for the first time in a metro network, by China Unicom in Beijing on 31 December 2010. (Clark, 2011)
- May 21, 2010: Nice, France launches "Cityzi", the "Nice City of contactless mobile" project, first time in Europe to provide inhabitants with NFC bank cards and mobile phones (like Samsung Player One S5230), and services such as transportation (tramways and bus), tourism and student's services.

2.1.3 NFC in public transportation in Ireland

The TFI Leap card, which is a card that has an NFC chip in it, was introduced by Transport of Ireland in December 2011; as part of the rollout of an integrated ticketing scheme for public transport in Dublin city; in the greater Dublin area for Dublin bus, LUAS, DART and Iarnród Éireann. Since then, it has become widely used across the country. There were over 2.5 million Leap Card users according to the National Transport Authority in 2017 and this has only been increasing since then. In 2016, the NTA (National Transport of Ireland), introduced "Transparent NFC". (Dorgan, 2017) solution, which allows the user to top up their leap card through

their Android phone (iOS version was released in 2021) with the help of their bank card as a mobile application.

NTA (National Transportation Authority of Ireland) has announced “In addition to tapping on with Leap Cards, the new ticketing system will allow people to use contactless payment on buses, rail, trams, TFI Local Link, and the planned Metrolink. It will also cover intercity rail services using barcode-based mobile ticketing.” (Zapryanova, 2022)

2.2. Alternative Existing Solutions to Your Problem

Several similar applications have been developed which combine NFC technology and mobile phones to implement ticketing systems. Some of the technologies are:

2.2.1 TFI Leap Top Up

TFI Leap top up is a mobile application that was introduced in 2017, to help the user to top up their physical leap card using mobile phone that has NFC chip integrated into them. Leap tip up is based on ‘Transparent NFC solution’ which enables the user to interact with a smartcard via their mobile phones. The Transparent NFC server allows read/write and authentication of the Leap card.

All the data are handled in server and no encryption key are store in the mobile phone rather it is store in Amazon Web Services for scalability and security.

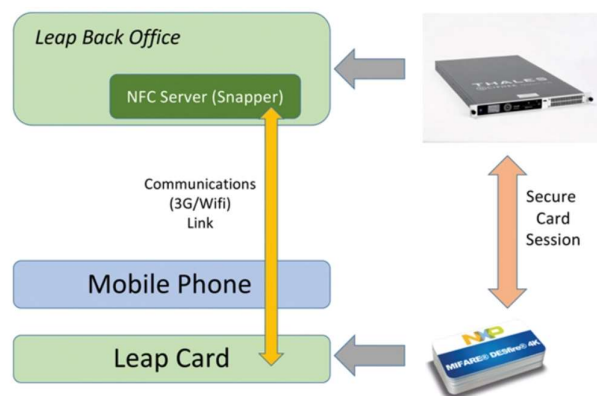


Figure 2: The Transparent NFC solution (Dorgan, 2017)

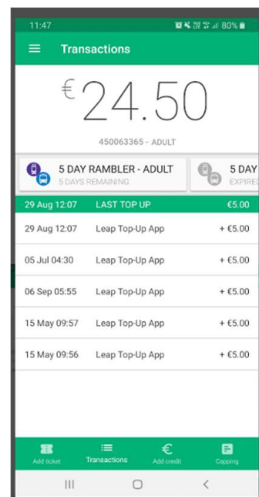


Figure 3 Leap top-up UI

Advantages

1. Secure as no data are stored in the mobile phone
2. Scalable as it is hosted in AWS

Disadvantage

1. It is only used for topping up and can only be used if you have a physical card.

The top up part of this application is one of the solutions that this project will integrate, which allows the user to top up their account through the payment gateway.

2.2.2 TFI Real Time

TFI Real time application combines all the real time information for Bus Éireann, Dublin Bus, Go-Ahead Ireland Iarnród Éireann and Luas. And allows the user to search for the route information based on the stop number, stop name or the route information or from the maps.

Advantages

1. Users can find route information based on the stop number, stop name, and map
2. Live update

Disadvantages

1. It does not support payment system.

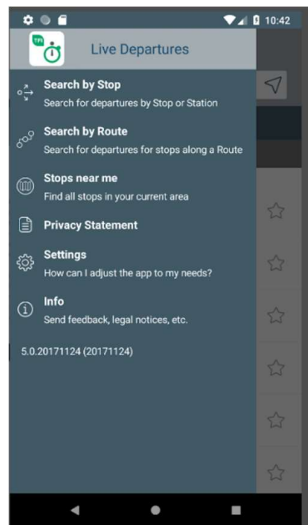


Figure 4: TFI Real Time UI menu

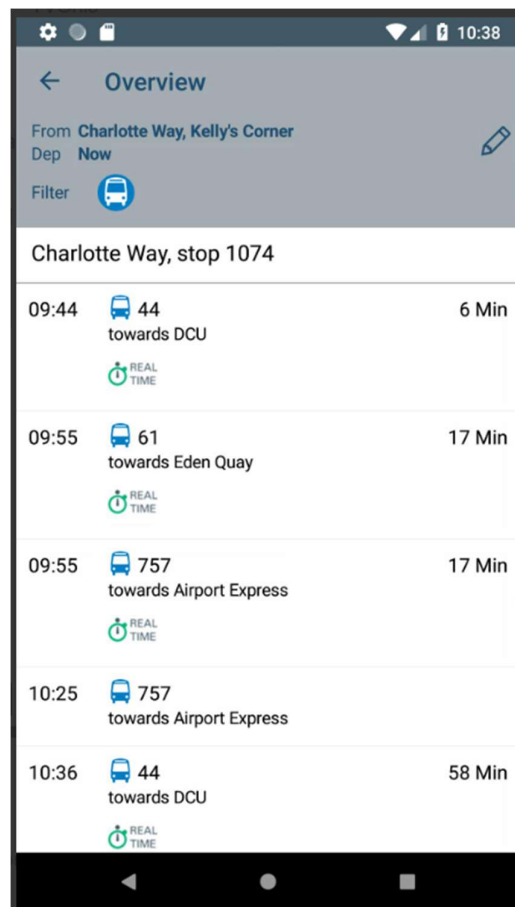


Figure 5: TFI Real Time app UI

2.3. Technologies you've researched

This section goes through the multiple different technology for Operating system, IDE, Programming Language, Database and APIs. A selection of technologies will be made based on the requirements of the project.

2.3.1 Operating System

2.3.1.1 Android

Android is a mobile operating system developed by Android inc., which was later acquired by Google in 2005. It was primarily used in touch screen devices such as mobile phone and tablets.

Android operating system is based on Linux kernel. Due to its open-source nature, it is researched, attacked, and fixed by many research developers. So, Linux has become stable and secure kernel. (Singh, 2014)

Android Software development kit (SDK) is used to develop application for Android which includes, debugger, libraries, an emulator based on QEMU (Quick Emulator), documentation, sample code, and tutorials. Java is the primary programming language for application development, but there are other Language such as Kotlin which we will discuss in the later section.

Advantages of Android OS:

1. Open-source operating System.
One of Android OS's benefits is its Open-Source Operating System. Developers can access the source code and modify the code to suit their hardware and application compatibility. Which in thus results in more contribution to improve the OS from the community.
2. Phone option
Android OS is used by different phone and manufactures which allows the user to have freedom purchasing different brands and gives the manufacturer ability to customize their UI based on their mobile phones.
3. NFC
Near-field communication chips are integrated in most of the Android mobile phones which makes it easier for them to utilized by developers, and it is well documented which makes it easier to research and develop.

2.3.1.2 IOS

Apple mobile devices run on Apple iOS operating system. First released in 2007, It is the second most popular mobile operating system. And it continues to power Apple iPhones, iPads, and other apple related products.

Apple's operating system is based on the Unix kernel, and it is a proprietary operating system owned by Apple.

The iOS SDK (iOS Software Development Kit) is used for the development of mobile apps on Apple's iOS and iPadOS operating systems. With Swift and

Objective-C being the main programming language for development. The SDK contents are separated into the following sets Cocoa Touch, Media, Core Service, Mac OS x Kernel.

Advantages of iOS:

1. Easy to Use Interface
The simplicity and compatibility of the interface are some of its best features. It is consistent across all the upgrades and there is no major change to the User interface.
2. Security
The iOS devices are secure and stable. Since it is a proprietary operating system, Apple restricts the use of websites or applications from third parties to download other apps.
3. Software lifecycle.
In iOS, it's far easier for iPhones to update. This also benefits developers so they can integrate new iOS features easily.

2.3.1.3 Selection (Android)

The project will be built as an Android application for the following reasons:

1. IDE
Android Studio is an integrated development environment (IDE) used for developing an application for the Android OS. It was announced on May 16, 2013, at the Google conference by Google's Product Manager, Katherine Chou.
It is powered by the IntelliJ IDEA, which provides fast code completion time and instant evaluation of the workflow.
Android Studio offers many testing tools and frameworks to test Android apps by using functional UI testing tools.
It also has an integrated Firebase module that can be used to connect to Firestore Database which we will discuss later in this section.
2. NFC

This project is mainly based on utilizing the NFC tag to simulate the Leap card as a virtual one. And Android has been utilizing this technology since 2010, which has led to a lot of documentation and forum where one can easily learn and build Android applications using NFC.

Whereas in iOS it is harder to utilize and manipulate NFC chips, one of the reasons being how closely it is tied to Apple pay. Also, Android Studio provides many built-in APIs and libraries for implementing NFC reading. This is important for implementing the Tag on functionality of the project.

As mentioned above Android will be used over iOS due to it being open source and offers more freedom and customization options than iOS. The IDE for Android, Android Studio comes with Firebase Assistant which allows connecting any app with the Firebase server and has a lot of documentation on it. Also, NFC manipulation is more convenient and easier in Android compared to iOS.

2.3.2 Database

2.3.2.1 PostgreSQL

PostgreSQL is a powerful, open-source object-relation database system.

PostgreSQL, originally called Postgres, was created by a computer science professor named Michael Stonebraker at UCB. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres, which now is owned by Computer Associates.

PostgreSQL supports modern application features like JSON, XML, etc and is complete ACID compliant. It offers support for both SQL and NoSQL.

2.3.1.4.2 Oracle

Oracle makes use of SQL or 'Sequel' language for operations on data and data manipulation.

The Oracle database was established by Larry, Bob & Ed Oates on 16th June 1977, as the Software Development Labs. In 1995, this was again changed to the Oracle Corporation which is what known as today.

The Oracle database runs on several operating systems. Oracle database is ACID-compliant & provides data backups. Oracle RAC allows the users to run one database on several different servers.

2.3.1.4.3 Selection (Firestore Database)

Firestore Database will be used for this project to store users' information and to store route information. Firebase is a web application platform developed by google and launched in 2011. It can be also used as a backend which will be used in this project for authentication, recovery and adding users so can be also called 'Back-end as a Service'.

Firestore Database stores the data in JavaScript Object Notation (JSON) format, An API is provided to the application developer which allows application data to be synchronized across clients and stored on Firebase's cloud. (Khawas & Shah, 2018).

Firestore Database is an integrated database service provided by Firebase which is accessed through the Firebase API.

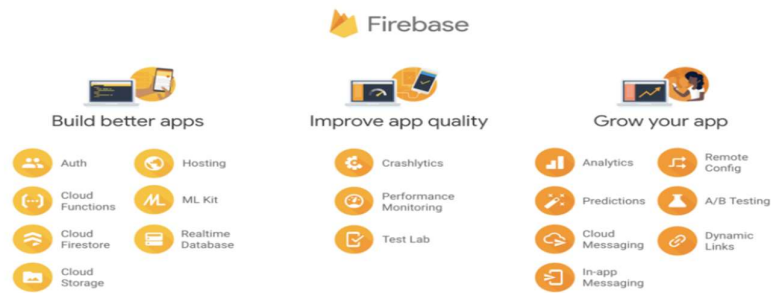


Figure 6: All the tools available in Firebase (Android Studio, 2017)

Firestore Database will be used compared to Oracle and PostgreSQL because most of the data that will be handled is in NoSQL format, and its simple and straightforward integration with the Android Studio makes it easier to use for this project.

2.3.3 Programming Language

2.3.3.1 Swift

Swift is a powerful general-purpose programming language for the Apple ecosystem which was released in 2014 as 'Objective C- without C'.

Swift is built upon Objective-C and has all the techniques, but it modernizes them to include shorter syntax and easier readability, maintainability, and safety to prevent crashes.

Advantages:

1. Rapid development process

Swift is easier to read and write. It is very concise, which means less code is needed to perform the same task, in comparison to Objective-C. It usually takes less time to build iOS apps with Swift.

2. Full stack potential

Swift, can be used to program both front-end and back-end, speeding up the development process and reducing development efforts.

Disadvantage:

1. New Language

Swift was introduced in 2014 so it has only been 7 years compared to other programming languages like Java or objective C, because of this the support community is not big and support is limited.

2. Compatibility

Before Swift's Application Binary Interface was released, there were problems with code compiled with different versions (3.0,2.0,1.0) of Swift. While that's now possible with swift 5.1, there are still issues implementing those dependencies into a project.

2.3.3.2 Kotlin

Kotlin is an open-source, statically typed programming language that was introduced in 2016 by JetBrains.

This open-source language is designed to work on platforms such as Android, JVM, JavaScript, and Native.

It is interoperable with Java – all Java frameworks and libraries are compatible with Kotlin, so the two can co-exist so it is also sometimes known as “the alternative Java”.

Advantages:

1. Interoperability with JAVA

As mentioned, it is consistent with Java and all related tools and frameworks. Also, both languages can be comfortably used at the same time.

2. Fewer bugs

The code in production is more stable and consistent because of Kotlin’s cleaner and compact codebase. Bugs are found during compilation, allowing developers to correct them before runtime.

Disadvantage:

1. Limited Resources for Learning

Kotlin’s learning resources may not be as comprehensive as those for Java.

2. Build Time

Compared to Java to develop a clean build for Android apps, Kotlin is slower.

2.3.3.3 Selection (JAVA)

Java is an open-source and object-oriented, general-purpose programming language that have few implementation dependencies as possible.

Java was originally developed by James Gosling at Sun Microsystems. It was released in May 1995.

Java contains a lot of libraries, and it is relatively easy for a developer to use these libraries. Rather than compiling and optimizing native code on multiple devices, Java allows using the independent features for Android development. It also includes the WORA (Write Once Run Anywhere) feature, which makes it platform-independent language.

For this project we will be using Android Studio as our IDE which uses JAVA as its core programming language. Also, as it runs in JVM, it does not need to be re-compile.

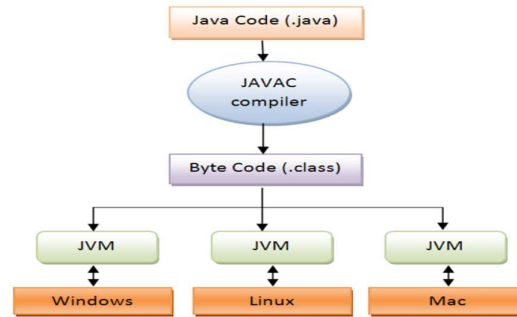


Figure 7: Java Programming model. (Mandula, 2018)

Another reason is when we search about Java problems; we are bound to get answers because it is an established and very popular language for Android development there are vast resources and tutorials, that will help to implement several functionalities of this project. The same cannot be said for Kotlin or Swift, which are still upcoming programming languages.

2.3.4 Back-End

2.3.4.1 Node.js

Node.js is made of Google's V8 JavaScript engine, the libUV platform abstraction layer, and a core library that is written in JavaScript. Also, Node.js is based on the open web stack (HTML, CSS, and JS), and operates over the standard port 80.

Node.js operates on a single thread, using nonblocking I/O calls.

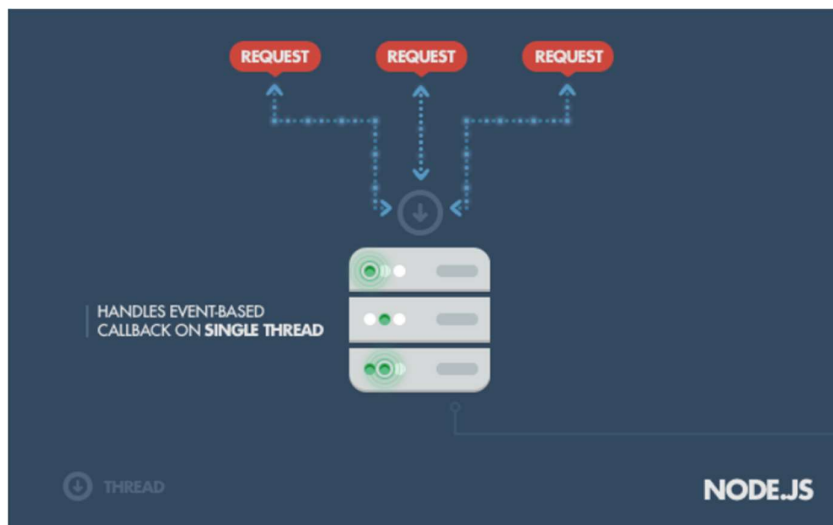


Figure 8: Node JS single thread operation (Capan, 2013)

2.3.4.2 Spring Boot

Spring framework is an open-source Java-based framework used to create a micro-Service that run on Java Virtual Machine (JVM). Spring boot is an extension of

Spring framework. Micro Service is an architecture that allows development and deployment of services independently.

It is easy to auto-configure all the components and functions for a production app with this framework. Developers can access Spring's ecosystem that consists of: Spring Data, Spring Security, Spring ORM and Spring JDBC3. Also, it is easy to integrate Spring Boot with Oracle DB, MongoDB, MYSQL, etc.

Spring boot is multi-threaded which means that several tasks can be performed concurrently.

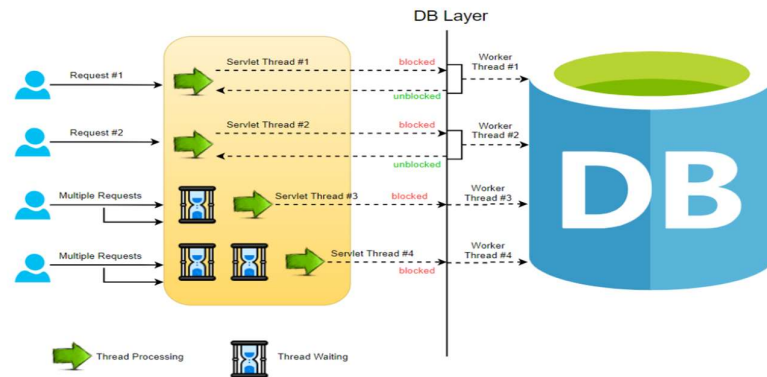


Figure 9: Spring Boot multi thread operation (Garai, 2022)

2.3.4.3 Selection (Python using RESTful API)

2.3.4.3.1 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a dynamically typed and garbage-collected programming language.

Guido van Rossum during 1985- 1990 created python. Python source code is available under the GNU General Public License (GPL).

Reason for choosing Python as a backend for this project.

1. Its vast libraries that deal with JSON objects will be helpful to interact with the GTFS API and Firebase database.
2. It is one of the easiest languages to work with and has a lot of documentation because of its popularity.
3. Because large amounts of data are needed for Route planning, it is much easier to be processed in a server rather than the Android application itself.

2.3.4.3.2 RESTful API

A RESTful (REpresentational State Transfer) API is an application program interface (API) that uses HTTP requests to GET, PUT, POST, and DELETE data.

The use of REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style because REST does not leverage as much

bandwidth, which makes it a better fit for use over the Internet. (ARYA, 2015)

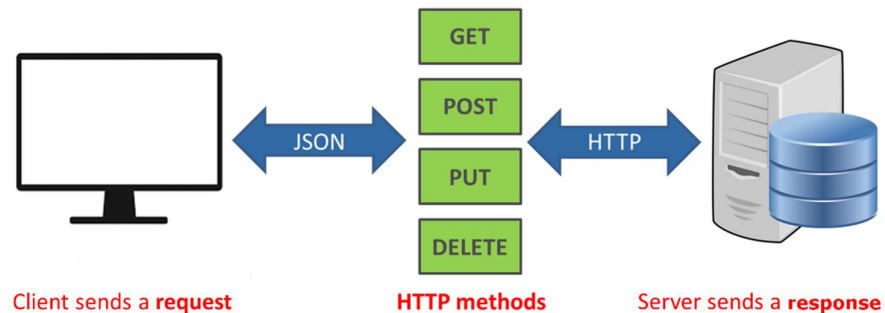


Figure 10: Rest API (Benharosh, 2018)

2.3.4.4 Selection (Firebase)

We will also be using Firebase for the backend for authentication and Create Read Update and Delete functionality of the user.

“Firebase Auth supports social login providers like Facebook, Google GitHub, and Twitter. It is a service that can authenticate users using only client-side code and it is a paid service. It also includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.” (Khawas & Shah, 2018)

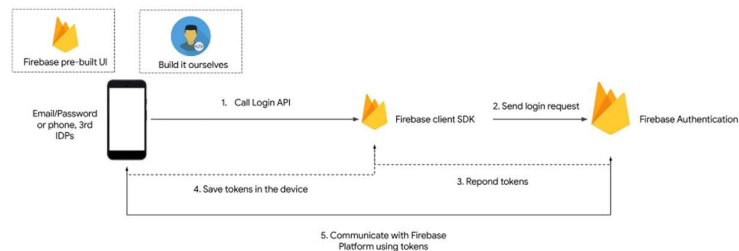


Figure 11: Firebase authentication workflow

2.4 Application Program Interface

Application Programming Interface (API) enables two applications to communicate with one another without the need for a user to get involved API, enables data interchange and communication between two different types of software.

GTFS API will be used which provides live feed of the bus and Stripe payment API will be used for topping up the account of the user.

2.4.1 GTFS

GTFS Realtime is a feed specification that allows NTA (National Transport Authority) to provide real-time updates to application developers. It is an extension to GTFS (General Transit Feed Specification), an open data format for public transportation schedules and associated geographic information. GTFS Realtime was designed around ease of implementation, good GTFS interoperability, and a focus on passenger information.

The GTFS-R API contains real-time updates for services provided by Dublin Bus, Bus Éireann, and Go-Ahead Ireland.

The URL of the GTFS-R is

<https://api.nationaltransport.ie/gtfsr/v1?format=json>

2.4.2 Stripe Payment Gateway

A payment processor — often called a payment gateway — authorizes various payment methods during the checkout process, enabling an easy and secure transfer of funds from a customer's account to the business's account. (Wallis, 2021)

Stripe takes all the complex work out of processing payment systems by providing a payment infrastructure through a single API. Not only that it has various documentation for integrating with Android projects and gives a test account for demo payment. And it has a simple UI for the payment details as well which can be directed through the API.

This project will utilize this technology for the top-up feature.

2.5 GIT

A version control system (VCS) allows you to track the iterative changes you make to code. (Blischak, et al., 2016)

Repository (aka a repo) is a “main folder”, everything associated with a specific project should be kept in a repo for that project. Repos can have folders within them, or just be separate files. There will be a local and an online copy (on GitHub) of all the files in the repository.

This project will make use of git and git hub to store the code online and make use of fork which is a copy of a repository that makes it easier for the code to be managed. Forks help to make changes to a project without affecting the original repository.

2.5 Existing Projects

There has been project done related to the domain of this project, this section will provide a review of those projects as well as some key takeaways.

2.5.1 NFC Based Smart Urban Public Bus Transport Payment System

In this paper, the author explains the current urban transportation situation in India and how it is growing rapidly. The main aim of the project is to build a simpler e-ticketing system utilizing NFC for the transportation system in India.

The author goes through and reviews some of the other case studies that have been done utilizing NFC and project build. One of the projects is Rejsekort, which is being utilized in Denmark and works on making use of an RFID chip in a smart card. The author goes through

some of the literature papers that have been written that discuss the projects and solution that makes use of RFID technology and QR codes. One of the interface is based on NFC or QR codes for smart sensors for security is reviewed by the author. (Ulz, et al., 2017).

The author then goes on to propose a system design and a mobile application that can be used to get the users' total fare and the bus journey using an NFC Tag on that can be displayed in the mobile application with the future scope of integrating an e-payment system as well and concludes that NFC secure and connectivity feature makes it an effective choice for utilization in payment systems. (Dhule, 2018)

2.5.2 Driving Instructor Assistor

Baolach Morrison projects aim to provide an Android app to assist driving instructors. It keeps all records and business information of the user and the instructor. Also, it provides map data for places that the instructor is more familiar with.

Complexities: The outcome of the project is to provide a real-world based application, so it needs to go through testing and implementation much more thoroughly. The application must be simple to use and fast for the instructor.

Technical Architectures: Android Studio, Java, Django framework.

Evaluation: This project provides a practical application for driving instructors by including time-saving elements, such as capturing financial and business information, and by aiding in class planning by marking specific places on maps for evaluating driving abilities.

2.6. Conclusions

In this chapter two application was evaluated that integrated some functionality needed for this project. Research into potential technologies that may be employed in the development of the app was conducted after evaluating the 2 applications. Finally, other similar projects were reviewed. Examining these projects also aided in determining how to plan for the amount and nature of the work needed.

Knowing what technologies could be used and having defined a list of requirements for the proposed system leads into the design chapter for this project.

3. System Design

Following chapter provides detailed research into the software methodologies implemented within this application. It will also cover the early designing stages of the project like Software Methodologies, UML, Database design, High-level architecture, and description, along with high-fidelity prototypes of the user interfaces.

3.1 Software Methodology

This section we will look through the different type of Software Methodologies and their advantages and disadvantages also, we will discuss why certain methodologies has been chosen for this project.

3.1.1 Waterfall Methodology

Water Methodology is a linear development model that unfolds in strict sequence, each phase of the Waterfall methodology's life cycle begins when the previous ends.

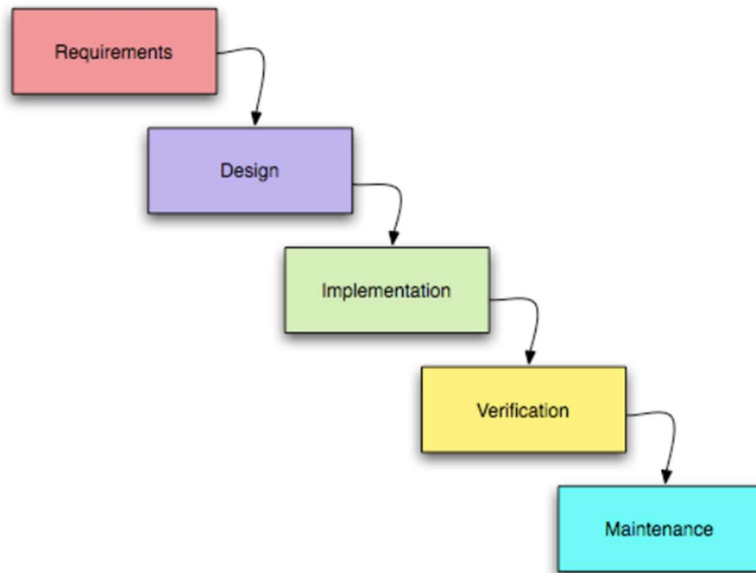


Figure 12: Waterfall Methodology (Hughey, 2009)

The methodology's is based on the traditional Waterfall model, which does not allow a software development team to return to previous phases of the life cycle once they are completed – the project, like a waterfall, flows in one direction only.

The Waterfall model is credited to Winston W. Royce in 1970. Royce originally presented it as an example of a flawed, non-working model. This is currently the way that the model is described now, criticizing a widely used software development practice. (Ragunath, et al., 2010)

3.1.2 Scrum Methodology

Scrum Methodology is a sub-group of agile methodology. The scrum methodology follows the values and principles of agile, but includes further definitions and specifications, especially regarding certain software development practices.

It is an iterative approach, which means that all the elements are repeatable, and it loops, which in Agile is called interaction, and in Scrum it is represented by Sprint, which is shown at figure 12.

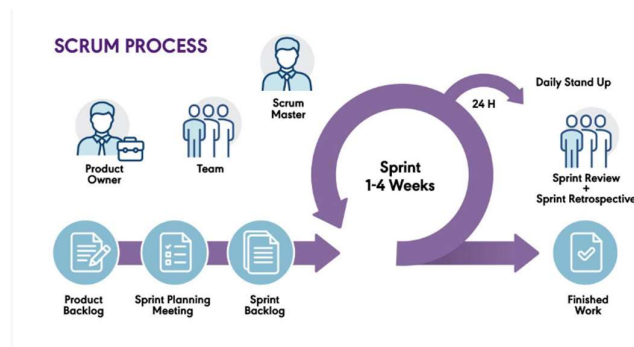


Figure 13: Scrum Methodology (PM Partners, 2021)

3.1.3 Selected Methodology (Agile Methodologies)

Agile methodology are iterative models of Software development life cycle. It as its name suggest is a flexible to changing requirements. There are many different methods of Agile, but they all share a common goal as described in the Manifesto for Agile Software Development or simply the Agile Manifesto, which are:

1. Welcome change
2. Deliver working software often
3. Businesspeople and developers work together
4. Build around motivated individuals
5. Face to face conversation
6. Working software is the primary measure of progress
7. Process promotes sustainable development
8. Attention to technical excellence
9. Self-organizing teams
10. Reflection at regular intervals

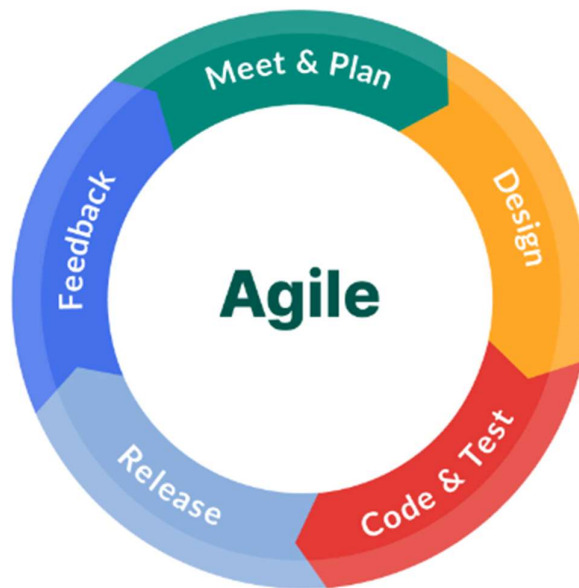


Figure 14: Agile Methodology (Dreissigacker, 2022)

Figure 13 shows the Agile development Methodology.

This project will use Agile Methodology for the Software Development Life cycle. Some of the reasons are:

1. In the agile software development method, the software is developed over several iterations. Changes can be implemented much more easily.
2. Since this project has a lot of moving parts and functionality is very different from one another, analysis, design, implementation, and testing are needed for every iteration.
3. This methodology requires active stakeholder involvement throughout the development and research process.

4. As functionalities are added incrementally. It helps to discover errors quickly and they are fixed immediately.

3.2 Design and Architecture

This section will go through the High-Level architecture and Design of Front end, Database and Back end respectively for this project.

3.2.1 Overview of System

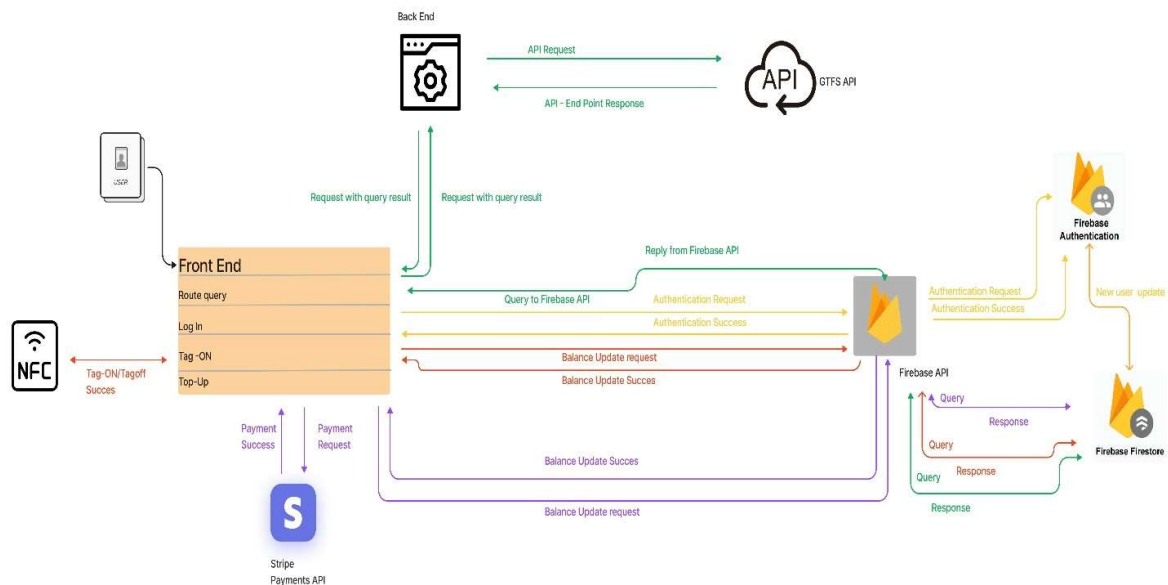


Figure 15: Technical High-Level Architecture

In Figure 28, we can see the High-Level Overview of the system, based on 4 functionalities. In this section a written description is provided for each of the functions and how the data flows.

3.2.1.1 Route Process

This functionality is used to search for routes and the live update of the time on each stop. The user will search will be able to search for a starting and ending bus stop name and based on that the user will get the route details and the live bus updates on each of the bus stops.

First, the users will select starting station name and the ending station name will be passed onto the Firestore Database through the Firebase API endpoints. The Firestore Database will then check if both starting station name and ending station name contains in the “trip_headsign” and give out a collection of trip numbers as a response to the Front-End. The Front End will connect to the backend server which will be written in Python with the help of REST API. The server will take in the trip ID and request GTFS API. When a successful response is secured by the server it will take in the response and send in the “StopId” and the “Departure” and “Arrival” “Delay” details back to the Front-End which will then make connections to the Firestore Database and query the “StopId” taken in from the back-end

server. The Firestore Database will search for “StopId” and “stop_names” and pass these results to the Front end which will display them in the UI.

3.2.1.2 Log In

For login, New user, and authentication all those tasks will be carried out by the built-in authentication process and database that the Firebase API provides for Android Studio.

The User will provide their email address, and password to the Front-end which will then establish a connection to the Firebase authentication system which will then secure and login based on the email and password.

The same data flow will happen when a user tries to create a new account but when the authentication is completed the Firebase authentication backend will connect to the Firestore database and add the user details.

3.2.1.3 Top-Up

When a user top up their account the Front-End will connect to the Stripe Payment Gateway which will send a response to the application if the connection is successful, when the connection is successful the Front-End will then send in a payment request with the amount that the user has selected to the Stripe payment UI where the user will fill in their card details and pay. When the payment is successful the Stripe API will send in a success token to the Front-End, the application will then again connect with Firebase API and request to Firestore Database to update the balance of the user. The Firestore Database will update the user info and the Firebase API will send a successful response back with the user account balance which will be then displayed in the UI.

3.2.1.4 Tag On

This feature allows the User to tap a preconfigured NFC tag and pay for their fares through their Android application.

When the Android application is launched and the user taps the NFC Tag, it will check for its content. The application will check if the NDEF message/content of the NFC tag matches the BUS agency which can be Bus Éireann, Dublin bus based on that the Front-end will connect to the Firebase API and update Firestore Database accordingly, which will be then sent back from Firebase API to the Front-End for the user to see.

3.2.2 Front End

3.2.2.1 Use- Case Diagram

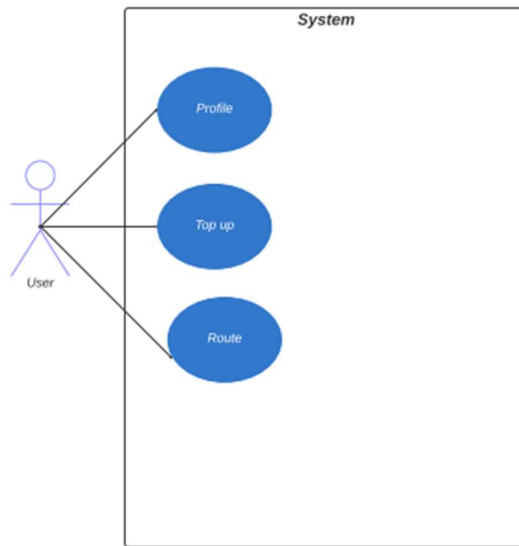


Figure 16: Menus User Case Diagram

Figure 14: Gives us the Use Case diagram of the Menu of the Application and relationship with the user.

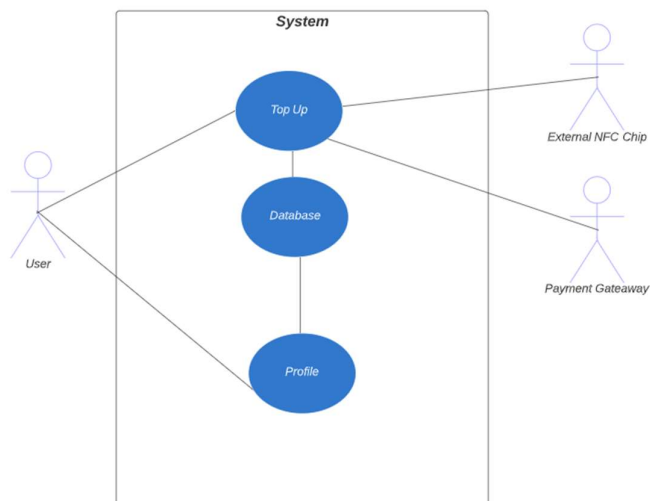


Figure 17: NFC Tap User Case diagram

Figure 15, show case the NFC- Tap Use case of how user makes a tap to the configure NFC tag and upon receiving the conformation the System will make use of Payment Gateway (Stripe) to carry out the payment. Then the application will connect to the database (Firebase) and update users' information which can be accessed by the user in the profile section.

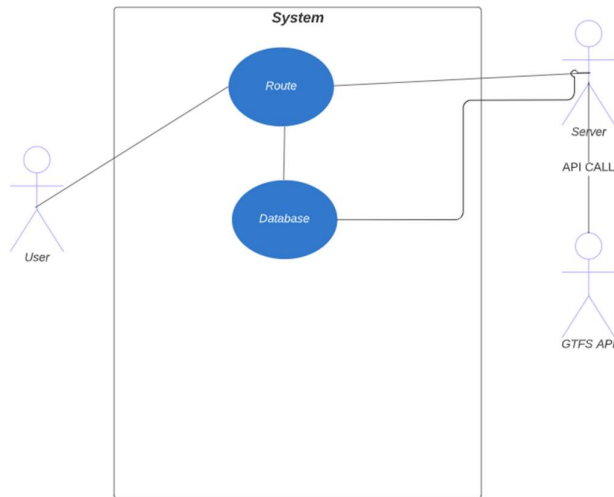


Figure 18: GTFS API User case diagram

Figure 16 shows the use case diagram of how the system interacts with real time API(GTFS) through the server to retrieve info based on users queries and connects with the server which in turns connects to the database to match the relevant information and provide relevant route info.

3.2.2.2 High Level Wireframe

High Level Wire frame examples of the Android application is given below, which may subject to change.

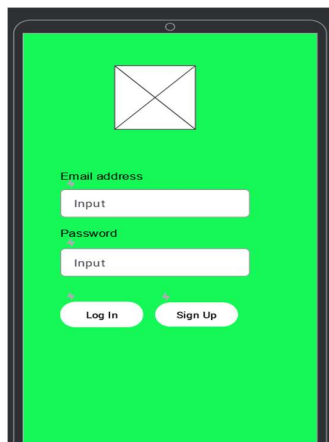
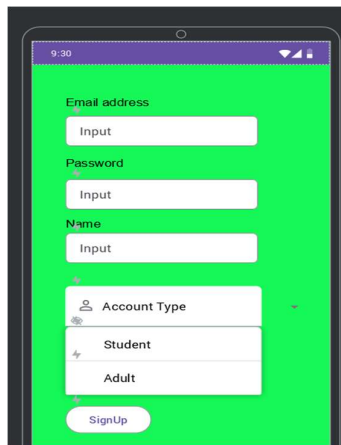


Figure 19: Log in Page



9:30

Email address
Input

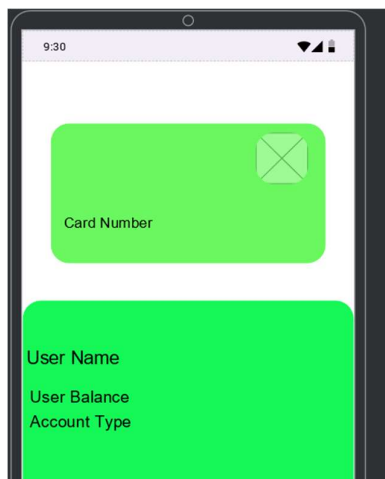
Password
Input

Name
Input

Account Type
Student
Adult

SignUp

Figure 20: Sign Up page

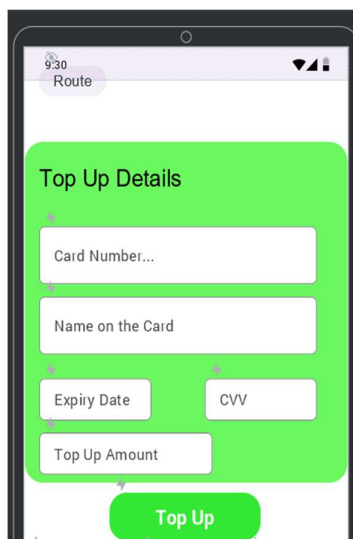


9:30

Card Number

User Name
User Balance
Account Type

Figure 21: Home page



9:30

Route

Top Up Details

Card Number...

Name on the Card

Expiry Date CVV

Top Up Amount

Top Up

Figure 22: Top Up page

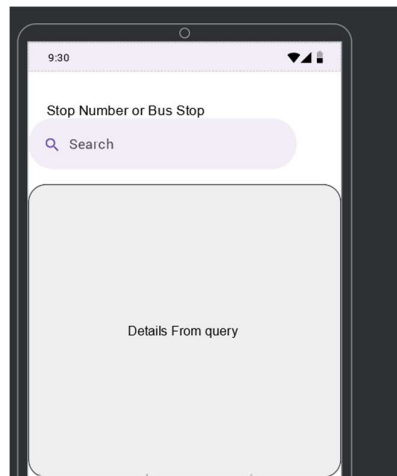


Figure 23: Route Page

3.2.3 Database

This section will go through the ERD that will be needed for this project, which may be subject to change in the future.

As Firestore Database, uses JSON objects and is stored in the form of collection, and there is no traditional way to depict it in a model, ERD will be used to closely depict what JSON object and the relation between the object may look like.

Figure 23 depicts all the collection of the JSON data that will be used in a traditional ERD form which shows the relationship between each collection more clearly and concisely.

Tables named trips, routes, stops, and agencies are prepopulated data taken from the National transport of Ireland authority website (Transport For Ireland, 2022). And the data that GTFS API gives in the form of a JSON object is represented in the form of tables GTFS_API_TripUpdate and GTFS_API_StopTimeUpdate in the figure below.

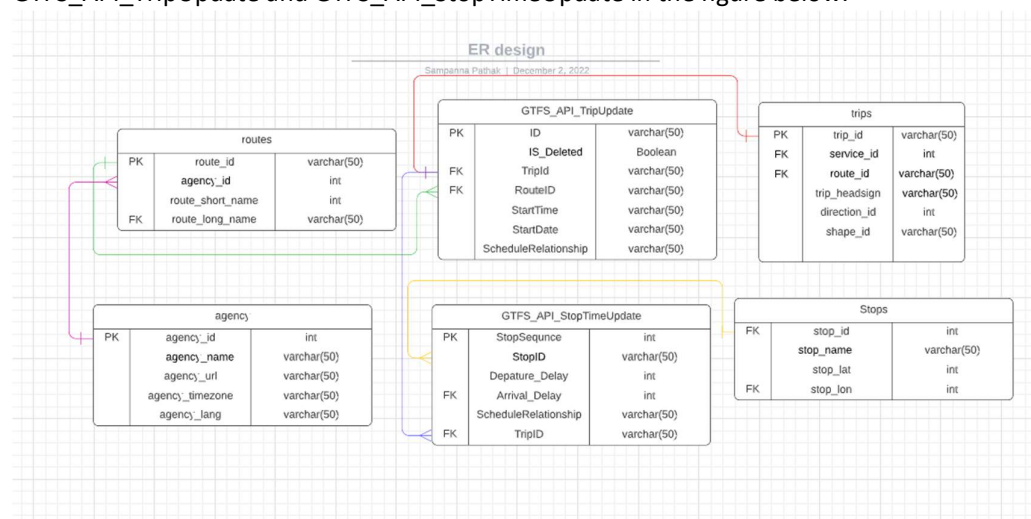


Figure 24: ERD

User information is stored in the following table

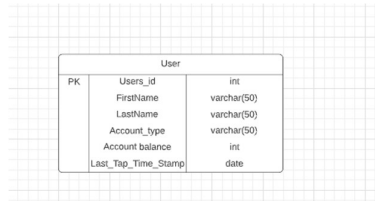


Figure 25: User table

3.2.4 Back- End

3.2.4.1 Sequence Diagram

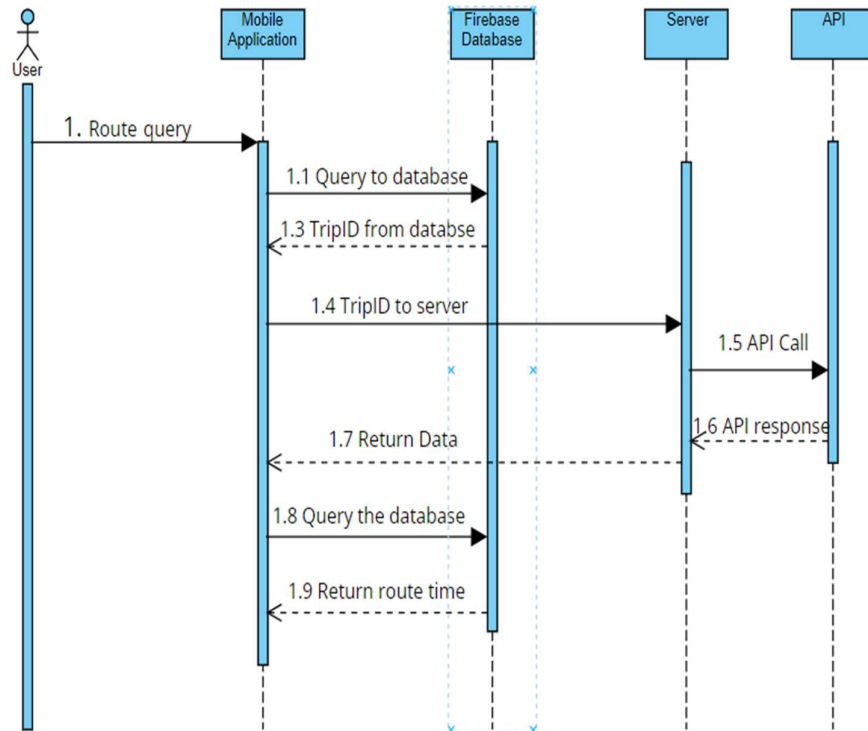


Figure 26: Sequence Diagram for Routing process

Figure 26 depicts the Sequence diagram for the routing process, this will be a rough workflow of how the system, the server and the API will communicate with each other.

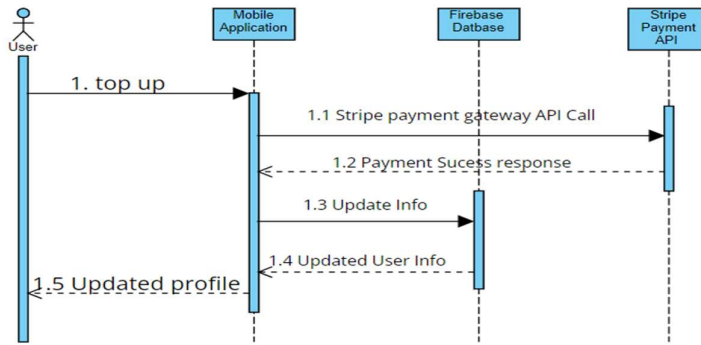


Figure 27: Sequence Diagram for Top Up process

Figure 27 shows the sequence diagram for the top up process and depicts the interaction between the user, the mobile application, the Firebase and the API during a top up process.

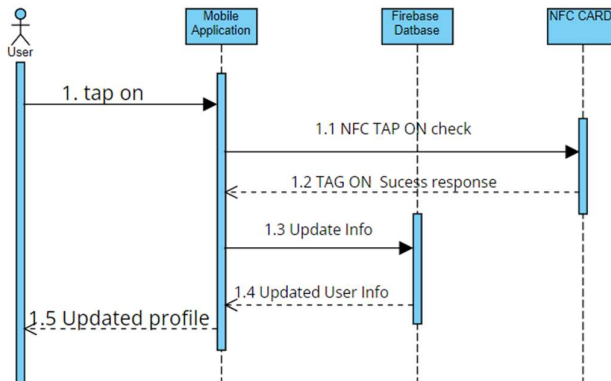


Figure 28: Sequence Diagram for Tag on Functionality

Figure 28 depicts the sequence diagram for the Tag on Functionality and shows how user, mobile application, Firebase and the NFC Card/Tag interact when the user taps on an NFC tag with their Android device.

3.3 Conclusions

This chapter has investigated different software methodologies and has given why the Agile methodology was selected for this project. An overall dataflow of the project was given, and high-level architecture of the system was broken down into 4 parts and a textual description of the data flow for each of the 4 functionalities was provided

In the end. The system diagram for the front-end, back-end, and database was given for this project namely, High-level wireframe for the Front-end, ERD for the Database, and the Sequence diagram for the Backend of this project.

The next chapter will go through the various testing plans and evaluations needed for the project.

4. Testing and Evaluation

4.1. Introduction

This section will introduce the testing and different type of tools that will be used to test the application as well as different testing techniques for Front-End, Back-End.

Application testing is a form of software testing that is carried out using scripts with the goal of identifying software bugs. Application testing in software engineering can be divided into several categories, including GUI, functionality, database (backend), load testing, etc.

4.1.2 System Testing Methodology

4.1.2.1 Black-Box Testing

Black box testing, also referred to as functional testing, looks at the software's functionality without needing to understand its core code structure.

The tester provides input, and the process for testing software considers the output.

This makes it easier to figure out faults with reliability, response speed, usability, and the system's reaction to expected and unexpected user activity.

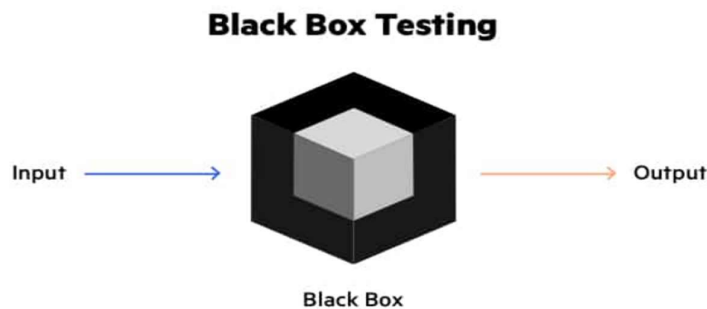


Figure 29: Black Box Testing

There are three types of Black-box testing namely functional testing, non-functional testing, and regression testing.

1. Functional Testing
2. Non-Functional Testing
3. Regression Testing

Some of the Techniques of Black Box testing are:

1. Boundary Value Testing
2. State Transition Testing
3. Error Guessing

4.1.2.1 White-Box Testing

White-box testing is used to test a software's architecture, design, and programming techniques. It mainly focuses on verifying the flow of inputs and outputs within the application.

White-box testing is usually on the internal workings of an application. The term “white-box” is used because it symbolizes the ability to see through the software’s outer “box” into its internal structure.

WHITE BOX TESTING APPROACH

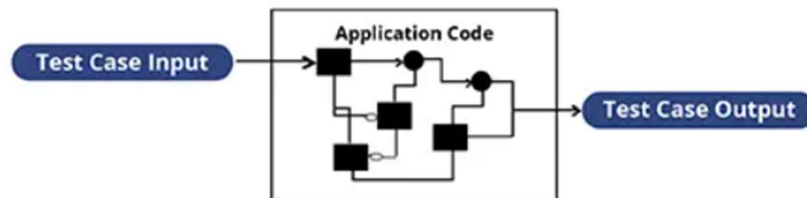


Figure 30: White- Box Testing

There are 2 types of White-box testing namely

1. Unit Testing
2. Testing for Memory Leaks

Some of the Techniques of White-Box testing are:

1. Statement Coverage
2. Branch Coverage

4.2 Plan for Testing

We will discuss the different types of tools and techniques that we will use to test this project.

4.2.1 Testing tools from Android Studios

Android Studio being a popular and widely used IDE for Android application development provides several testing tools that this project will be using, 2 mainly being:

a. Espresso Test Recorder

The Espresso Test Recorder tool allows you to develop UI tests for your app without writing any test code. You can document your interactions with a device and add assertions to verify UI elements in specific snapshots of your app by creating a test scenario. Espresso Test Recorder then takes the saved recording and automatically creates a corresponding UI test that you can run to test your app. (Android studio, 2017)

b. Exerciser Monkey Testing

UI/Application Exerciser Monkey (aka “Monkey”) is an open-source random testing tool included in the Android SDKUnit Testing. (Android Studio, 2017)

Exerciser Monkey is a tool that emulates a user interacting with an app, by generating and injecting pseudorandom gestures, e.g., clicks, drags, or system events into the app’s

event input stream. It also watches for exceptional conditions, e.g., the app crashing or throwing an exception. (Patel, et al., 2018)

4.2.1 Unit Testing

Unit Tests include sets of one or more programs which are designed to verify a unit of source code, such as a method or a class.

Android platform comes pre-integrated with Junit 3.0 framework. It's open-source framework for automating Unit Testing.

4.3 Plan for Evaluation

Nielsen Heuristic evaluation is a way to gauge a systems usability by going through a series of tests. These tests help to identify usability flaws in a system. Each check should test against specific design principals which have been altered to suit the needs of these types of apps. (Nielsen, 1995)

1. **Visibility of system status:**
The system's ability to keep users informed of the system's status with the appropriate outputs.
2. **Match between system and the real world:**
This refers to the system's ability to convey information in an understandable manner to the user. This includes using appropriate language as well as real-world phrases and terms.
3. **User control and freedom:**
Allows users to maintain control of the system without having to go through unnecessary steps; clearly labelled options should be available.
4. **Consistency and standards:**
The system's structure should adhere to its own conventions and be clearly structured so that users can understand it.
5. **Error prevention:**
A properly designed system is preferable to good error prevention but checking with users before committing to options can help prevent further errors.
6. **Recognition rather than recall:**
Users should be able to navigate the system intuitively rather than having to consult the manual. Following industry standards can aid in the recognition of system functions by users. This point is about the safety of the navigation stage.
7. **Flexibility and efficiency of use:**
The use of shortcuts or "Accelerators" can be a good way of increasing flexibility and efficiency for advanced users while not impeding the experience for new users.

8. **Aesthetic and minimalist design:**

The design of a system has a great effect on how users interact with it. The appropriate information must be displayed at the appropriate times.

9. **Help users recognize, diagnose, and recover from errors:**

Error messages should be presented in plain language so that any user can understand the problem and/or find a solution.

10. **Help and documentation:**

A system that does not require explicit documentation is always preferable, but proper and structured documentation can aid in assisting users with many specific problems.

Grading will be from 1 to 5 on each of the 10 Heuristics listed above, with a small explanation for each scoring.

The grading system scores:

1. Does not meet heuristic specifications.
2. Meets few of the heuristic specifications.
3. Meets some of the heuristic specifications.
4. Meets most of the heuristic specifications.
5. Fully meets the heuristic specifications.

4.4 Conclusion

This chapter went through the 2 types of testing tools methods namely white-box testing and black-box testing. Then we investigated several testing tools by Android Studio that will be used for the testing of the application. And then in the plans for evaluation section, Nielsen Heuristic evaluation was discussed, and how it works and will be used in this project.

5. Prototype Development

5.1 Introduction

This section outlines the design of a prototype that was developed for the interim presentation. It will give an example of each of the 3 main functionality that have been identified for this project.

5.2 Prototype Development

5.2.1 Firebase Authentication

For the user to log in the user needs to go through an authentication process which will be handled by the Firebase authentication backend. Firebase Authentication SDK, authenticate users with their email addresses and passwords. The Firebase Authentication SDK provides methods to create and manage users that use their email addresses and passwords to sign in. Firebase Authentication also handles sending password reset emails.

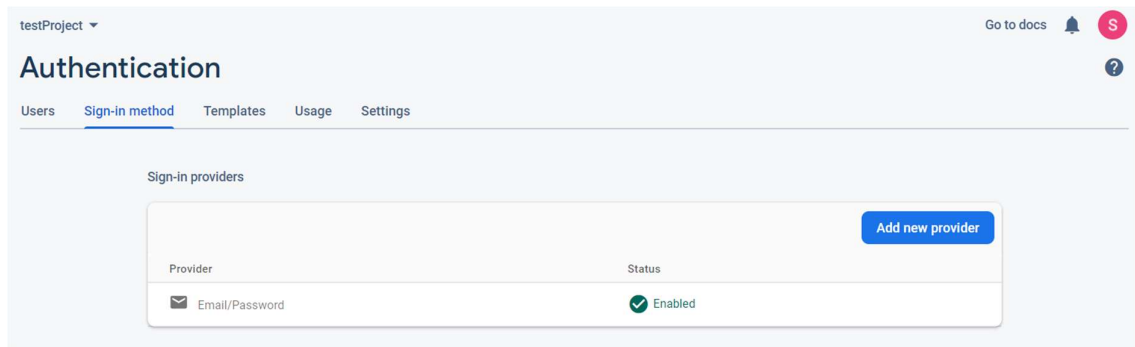


Figure 31: Firebase authentication enabled.

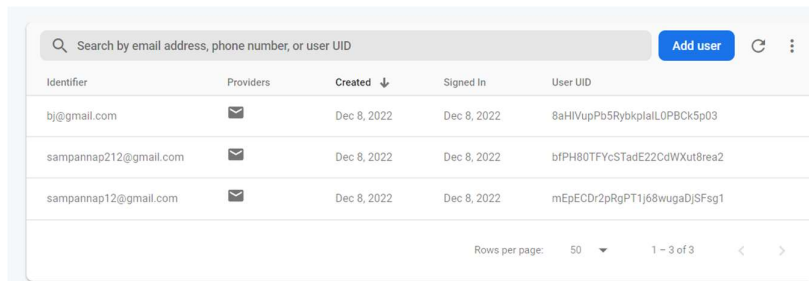


Figure 32: Authenticated user in the firebase console

```
//authenticating the user
 mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            Toast.makeText( context: Login.this, text: "User Signed In.", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
        }
        else
        {
            Toast.makeText( context: Login.this, text: "Error." + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
});
```

Figure 33: Authenticating user

```
// Register the user
 mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        // Checking the response from the firebase
        //if successful
        if (task.isSuccessful()){
            Toast.makeText( context: Register.this, text: "User Created.", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
        }
        else
        {
            Toast.makeText( context: Register.this, text: "Error." + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
});
```

Figure 34: Registering User


```
// To check if the user is already logged in if so sent them to the main activity
if (fAuth.getCurrentUser() != null)
{
    startActivity(new Intent(getApplicationContext(), MainActivity.class));
    finish();
}
```

Figure 35: Checking if user is logged in.

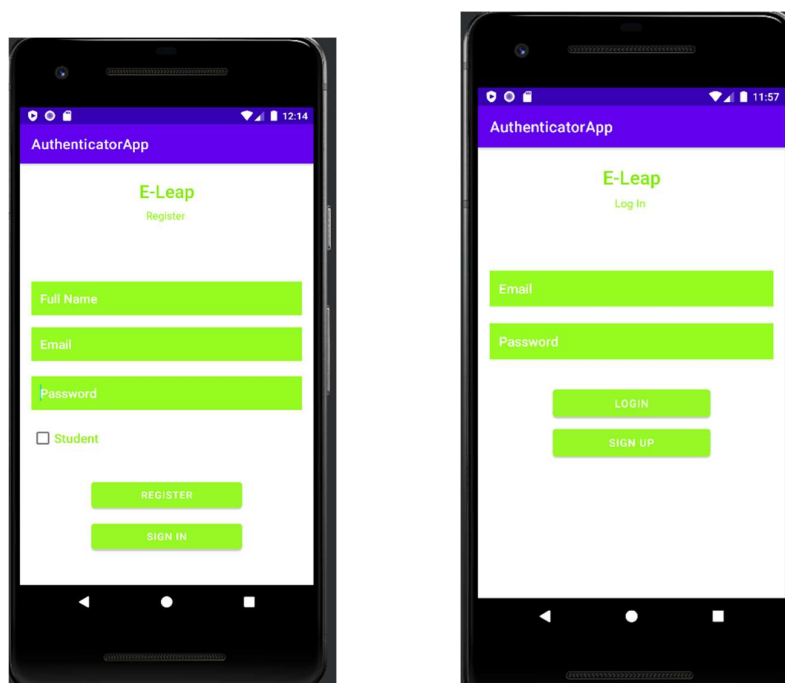


Figure 36: Register and Log In UI

5.2.2 NFC Tag Reader

To support the NFC card, a sample of NFC reader app is created as an example to show how the NFC tag will be scanned by the users' Android mobile phones and see the contents of the NFC tag.

```

// Reading TAG
private void readIntent(Intent intent){
    String action = intent.getAction();
    if(NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)
    || NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)
    || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action))
    {
        Parcelable[] rawMessage= intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
        NdefMessage[] msg =null;
        if (rawMessage != null){
            msg = new NdefMessage[rawMessage.length];
            for (int i=0; i< rawMessage.length; i++){
                msg[i] = (NdefMessage) rawMessage[i];
            }
            buildTagViews(msg);
        }
    }
}

```

Figure 37: Reading a Tag Android Studio-1

```

private void buildTagViews(NdefMessage[] msg){
    if (msg == null || msg.length== 0) return;
    String text ="";
    byte[] payload = msg[0].getRecords()[0].getPayload();
    //getting the text encoding
    String textEncoding = ((payload[0] & 128)==0) ? "UTF-8" : "UTF-16";
    //get Language code
    int languagecodelen = payload[0] & 0063; //0063 is "en"

    try {
        text = new String(payload, offset: languagecodelen + 1, length: payload.length - languagecodelen - 1, textEncoding);
    } catch (UnsupportedEncodingException e){

    }

    nfc_contents.setText("Content is ===== "+ text);
}

```

Figure 38: Reading a Tag Android Studio-2

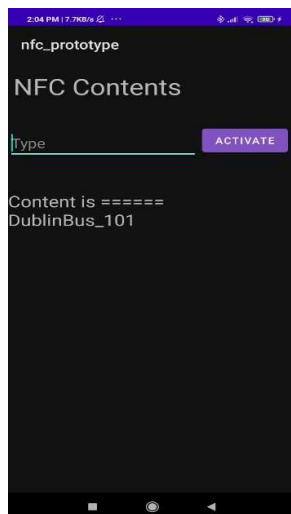


Figure 39: Example of the UI for the NFC tag reader

5.2.3 GTFS API

A simple python server has been created to showcase the connection between the GTFS Live feed and get the data as a form of JSON file using the GTFS API. This will be used in the project later to scan for queries from the user and integration of Route functionality.

```
try:
    url = "https://api.nationaltransport.io/gtfsr/v1?format=json"

    hdr = {
        # Request headers
        'Cache-Control': 'no-cache',
        'x-api-key': ' ',
    }

    req = requests.get(url, headers=hdr)
    data = req.json()
    #pretty = json.dumps(data, indent=4)
    ...
    # Writing Excel File:
    jsonFile = open("C:/Users/sampa/Desktop/FYP/data.json", "w")
    jsonFile.write(pretty)
    jsonFile.close()
    ...
    entries = data["Entity"]
    print (entries)

    pretty = json.dumps(entries, indent=4)
    jsonFile = open("C:/Users/sampa/Desktop/FYP/data2.json", "w")
    jsonFile.write(pretty)
    jsonFile.close()
```

Figure 40: Connecting to GTFS API and dumping the JSON data in a JSON file

5.4 Conclusions

There are 3 main functionality that need to be integrated in the project, NFC tag on, Routing queries, and a top up system. All the above functionality has been developed as a separate prototype which will all be integrated in single application later.

6. Issues and Future Work

6.1. Introduction

This section will go through the possible issues that may be faced during the development and implementation of code. And the amount and type of work that will be done in the future.

6.1.1 Implementation of NFC cards

This project's main aim is to make use of the NFC card on Android devices to pay for the bus fare. The integration of the NFC card and the code behind may pose some risks while integrating with other functionalities. Continue research will be going through the project to make sure that this won't pose a risk in the future.

6.1.2 Implementation of routing process

The routing process will be the second hardest challenge for this project. While the research has been, and part of the solution has been discussed with the stakeholder how the implementation will be carried out and the UI that will be designed around it and may be subject to change based on the circumstances which will be reviewed and researched and discussed with the stakeholders if any problem arises.

6.1.3 Time Constraints

With the project having multiple moving parts any unseen problem may delay some of the implementation and testing of the functionalities which will affect the development time. To minimize the loss of time and make sure that development goes on time. The project plan/ GNATT chart will be followed, and changes will be made after discussing with the stakeholders wherever possible.

6.2 Future Work

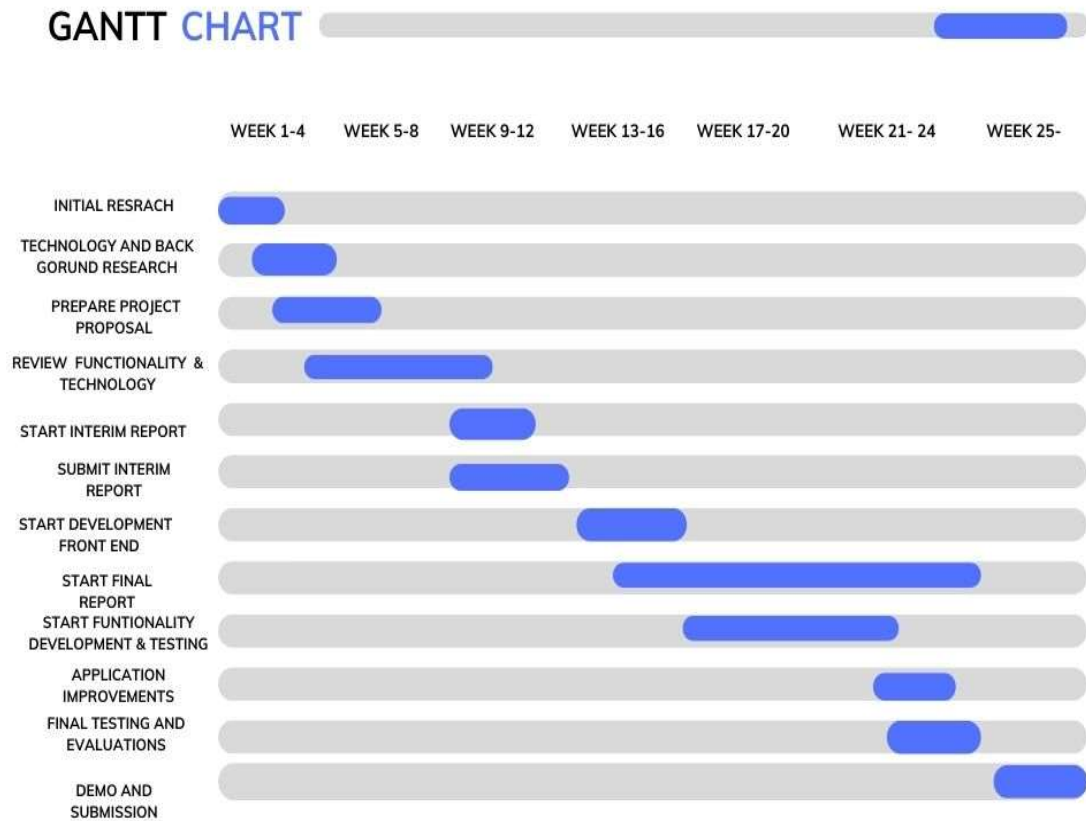
After the research was conducted and the key technologies identified. The coding and implementation of the functionality will start. The first process will be creating the User Interface for the login system and all the views needed as per the High-level wireframe in Figure above which may be subject to change. After the implementation of the User Interface, the integration of different functionalities will be carried out one at a time to make sure that one doesn't affect the other.

This will be done throughout every week from the 12th of December while also holding weekly meeting with stakeholder to discuss any problems and the progress made in the following week.

A rough time plan is given below through the GNATT chart for the future work.

6.2.1 GANTT Chart

GANTT chart gives a brief overview about the timing of the project so far and the estimated timing for the future works.



Bibliography

- Android Studio, 2017. *UI/Application Exerciser Monkey*. [Online]
Available at: <https://developer.android.com/studio/test/other-testing-tools/monkey>
[Accessed 2 12 2022].
- Android studio, 2017. *Create UI tests with Espresso Test Recorder*. [Online]
Available at: <https://developer.android.com/studio/test/other-testing-tools/espresso-test-recorder>
[Accessed 2 12 2022].
- Anon., n.d. *Create UI tests with Espresso Test Recorder*. [Online]
Available at: <https://developer.android.com/studio/test/other-testing-tools/espresso-test-recorder>
- ARYA, N., 2015. *BUILD AN API UNDER 30 LINES OF CODE WITH PYTHON AND FLASK*. [Online]
Available at: <https://impythonist.wordpress.com/2015/07/12/build-an-api-under-30-lines-of-code-with-python-and-flask/>
[Accessed 2 12 2022].
- Benharosh, J., 2018. *What is REST API? in plain English*. [Online]
Available at: <https://phpenthusiast.com/blog/what-is-rest-api>
[Accessed 12 December 2022].
- Bi, Z., 2013. *Near field communication system design with a circuit implementation*. Dallas: The University of Texas at Dalla.
- Blischak, J. D., Davenport, E. R. & Wilson, G., 2016. A quick introduction to version control with Git and GitHub. *PLoS computational biology*, 12(1), p. e1004668.
- Capan, T., 2013. *Why the Hell Would I Use Node.js? A Case-by-Case Tutorial*. [Online]
Available at: <https://www.toptal.com/javascript/why-the-hell-would-i-use-node-js>
[Accessed 09 December 2022].
- Clark, S., 2011. *China Unicom launches commercial NFC service in Beijing*. [Online]
Available at: <https://www.nfcw.com/2011/01/05/35551/china-unicom-launches-commercial-nfc-service-in-beijing/>
[Accessed 18 November 2022].
- Dhule, M., 2018. NFC Based Smart Urban Public Bus Transport. *2018 3rd International Conference for Convergence in Technology (I2CT)*, pp. 1-4.
- Dorgan, B., 2017. *NFC, Irish National Transport Authority Makes Leap Card Go Farther with NFC*, Dublin: nfc-forum.
- Dreissigacker, U., 2022. *Why Agile is so Popular in Project Management*. [Online]
Available at: <https://blog.ganttpro.com/en/why-agile/>
[Accessed 9 December 2022].
- Hughey, D., 2009. Comparing traditional systems analysis and design with agile methodologies. *Retrieved September*, Volume 20, p. 2016.
- Khawas, C. & Shah, P., 2018. Application of Firebase in Android App Development-A. *International Journal of Computer Applications*, 179(46), pp. 49-53.
- Lathiya, P. & Wang, J., 2021. *Near-Field Communications (NFC) for Wireless Power Transfer (WPT): An Overview*. Rijeka: IntechOpen.

Madlmayr, G., Langer, J., Kantner, C. & Scharinger, J., 2008. NFC Devices: Security and Privacy. *Third International Conference on Availability, Reliability and Security*, pp. 642-647.

Mandula, L., 2018. *Difference Between Source Code and Bytecode*. [Online]
Available at: <https://www.differencebetween.com/difference-between-source-code-and-vs-bytecode/>
[Accessed 09 December 2022].

Murray, S., 2018. *Using phones instead of physical Leap Cards part of NTA's plans to shake up commuter travel*, Dublin: thejournal.ie.

Nielsen, J., 1995. How to conduct a heuristic evaluation. *Nielsen Norman Group*, Volume 1, p. 8.

Patel, P., Srinivasan, G., Rahaman, S. & Neamtiu, I., 2018. On the effectiveness of random testing for Android: or how i learned to stop worrying and love the monkey.. *In Proceedings of the 13th International Workshop on Automation of Software Test*, pp. 34-37.

PM Partners, 2021. *The Agile Journey: A Scrum overview*. [Online]
Available at: <https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/>
[Accessed 09 December 2022].

Ragunath, P. et al., 2010. Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC). *International Journal of Computer Science and Network Security*, 10(1), pp. 112-119.

Singh, R., 2014. An Overview of Android Operating System and Its Security. *Int. Journal of Engineering Research and Applications*, 4(2), pp. 519-521.

Transport For Ireland, 2022. *PUBLIC TRANSPORT DATA*. [Online]
Available at: https://www.transportforireland.ie/transitData/PT_Data.html
[Accessed 8 December 2022].

Ulz, T. et al., 2017. SECURECONFIG: NFC and QR-code based hybrid approach for smart sensor configuration. *017 IEEE International Conference on RFID (RFID)*, pp. 41-46.

Wallis, J., 2021. *Why Stripe Is Arguably The Best Payment Gateway Built To Date*. [Online]
Available at: <https://webo.digital/blog/why-stripe-is-arguably-the-best-payment-gateway-built-to-date/>
[Accessed 05 12 2022].

Zapryanova, R., 2022. *Card and smartphone payments on the way for Dublin buses and trams*. [Online]
Available at: <https://www.dublinlive.ie/news/card-smartphone-payments-way-dublin-25483595>
[Accessed 18 November 2022].