Ticket Tapper

3rd Year Project Technical Specification
John Duffy: 15424962
Michael Solan: 15470992
Supervisor: Mark Roantree.

1 - Introduction.

1.1 - Overview of Project.

The main goal of our project was to design an app that allows people to buy train and bus tickets on their phone rather than looking for change or taking out their wallet.

The TicketTapper app will function on both android and ios. This is possible because of the new framework called flutter which was released by google. The user signs into their account on TicketTapper which then in turn interacts with the firebase servers. We use a firebase authentication server which saves the users information and credentials when they login. Then once the user has already registered then they can just simply log in. If the customer hasn't already registered they would go to the new account page where the user would enter their email, name and password. We also added in the feature of google login. When the user clicks the google login button they will be greeted with the regular google login page where they can sign in through their google account. Once the user has logged in they will go onto the home page where they will see options for payment and to plan their route. On the home page there is also a list of attractions that they the user can see in Dublin as that is the center area of our app. When they click on pay they will see a page where they can push a raised button to start the nfc connection. Once the nfc connection is made then it connects with the square payment API once payment is received then the customer will receive their qr code which acts as their ticket. This animation is shown to the user. NFC chips are important to this as we use these to show the NFC connection. After the user has bought the ticket they are sent back to the home page where they have the choice of clicking on maps or looking at their previous qr code. In the plan route section of our app the user is greeted with an open map where they can get their current location. Then the maps interact with the google maps api so they can use google maps feature that allows you to see what Dublin Bus route will take you to your location. The design of the app was kept consistent and by using the same theme for the UI it has a clean and minimalist look.

1.2 - Glossary of Terms.

NFC - Is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a smartphone, to establish communication by bringing them within 4 cm (1.6in) of each other.
Firebase Authentication - provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app.

Flutter - Flutter is Google’s mobile UI framework for crafting high-quality native experiences on iOS and Android.

Firebase Database - The firebase database provides the app a realtime database and backend as a service. The service provides the application developers an API that allows application data to be synchronized across clients and stored on Firebase’s cloud.

Application Programming Interface (API) - An application programming interface is a set of subroutine definitions, communication protocols and tools for building apps and software.

Flutter Plugin - A flutter plug-in is a software component that adds a specific to an existing computer program. When your program accepts a plugin in, it enables customization.

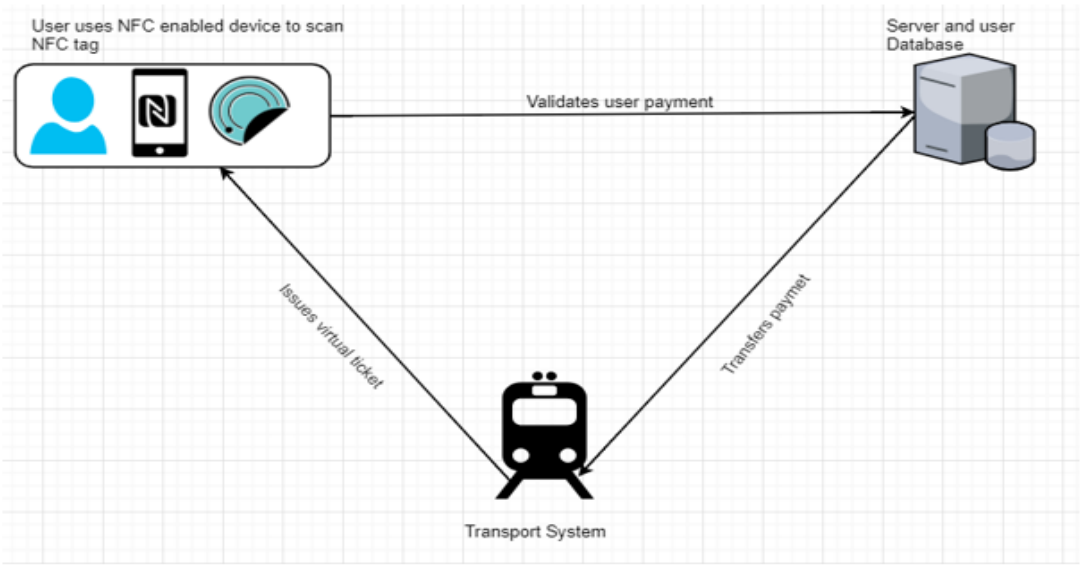
QR code - A barcode is a machine-readable optical label that contains information about the item to which it is attached.

Dart - Dart is an object-oriented, class defined, garbage-collected language.

Flutter Testing - Features unit, widget and integration testing of code designed in the framework flutter.

System Architecture

2.1 - System Architecture Diagram



The above image shows the architecture of our system in its whole. This diagram shows that our system is made up of these components. It starts off where the user who has an NFC enabled device interacts with the firebase system to authenticate their login credentials. Once the user has been authenticated then the user connects back to the server and user database which then validates the user payment. Once the payment is validated then a ticket is issued back to the user where their ticket is stored on the clients application. This is done when the API scripts react with the NFC chips to issue the ticket. In this scenario our NFC chips act as the transport system.

The clients application is how the user will use our app and our system i.e the phone. It is where the client will see all the front end implementation of our code. It will be installed on the person's own device. This is extremely important to the system architecture as it is in charge of connecting to the firebase database and it is where the client will receive their ticket after they have purchased it. The clients application also allows the customer to initiate the NFC connection.

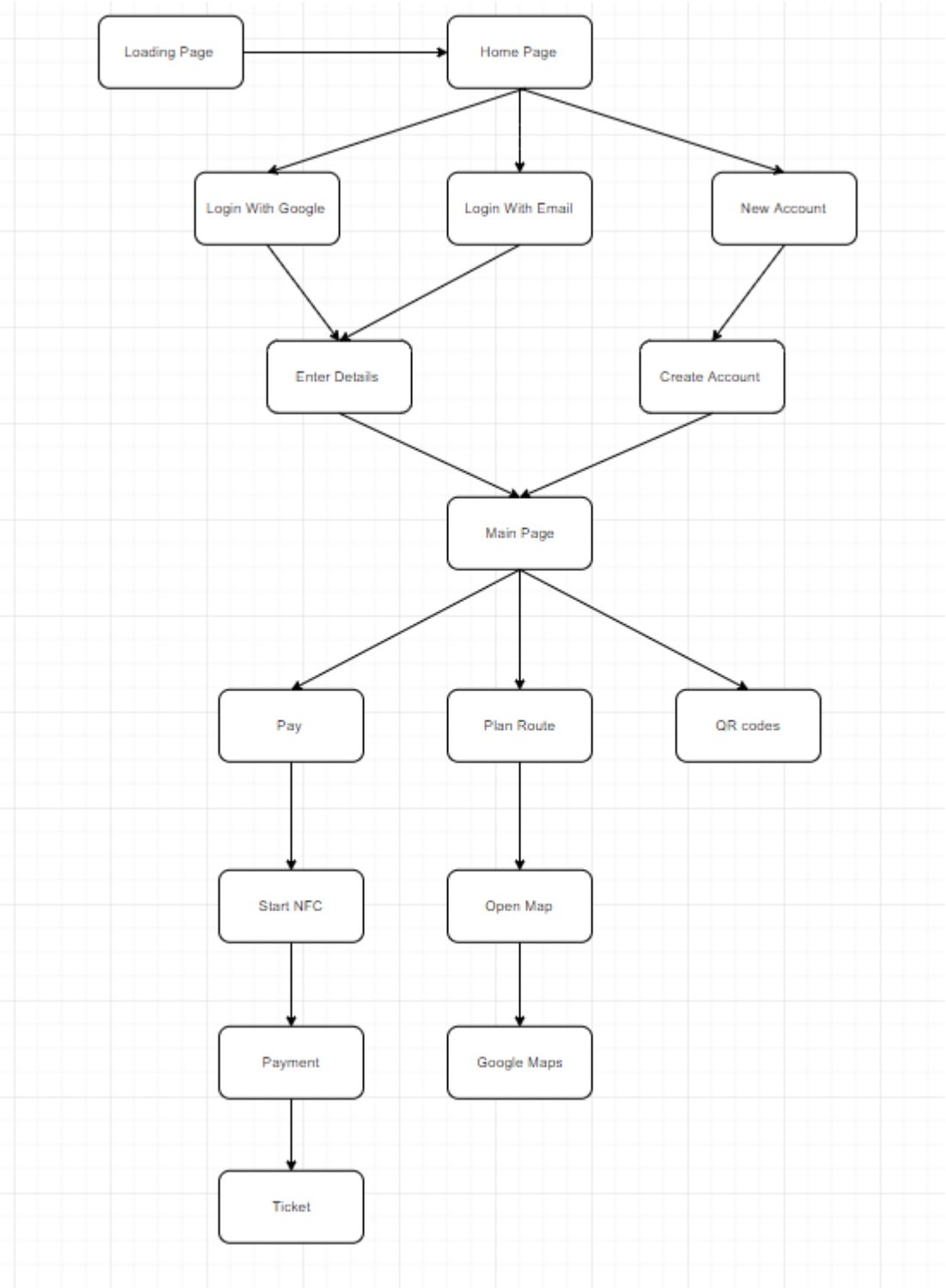
NFC connection. This allows the client to communicate and exchange information and data with the server. It is primarily used on system for payments of the tickets. When the connection is made it initiates the payment element of our system. This means that the user is brought to the payment screen where they enter their details and that in turn sends a request to the server.

The Firebase database. The firebase database is the primary database for our system. Not only does it store data about the users and their login details. It stores all the information about the users registration and implements firebase authentication. This means that the clients usernames and passwords are secured safely and is dealt with by the cloud.

Request Server. The request server is very critical to this system as it has the job of handling communication between outside parties and the client application. Its job is to receive a json message from the client which in turn the API scripts to carry out requests and form responses which the client can comprehend.

High-Level Design

3.1 High-Level Design Diagram



3.2 High-Level Design Description.

Stage 1 - Loading Page. This is the loading page that the user sees while they are waiting for our app to open. This loading is caused by our app so that the resources have time to set up.

Stage 1.1 - Main Page. This is the page that the user is greeted too after they open our application

Stage 2 - Login with email or google. This is where the user can register an account using Firebase Authentication while allows the user the ability to open the application. The user has two choices when they want to log in. they have the choice of logging in with their email or their google account.

Stage 2.1 - New Account . Enter in information about themselves like email, password and full name. This information is sent to the firebase servers where it allows the users to login with them credentials at a later date.

Stage 3 - Enter Details. Enter your email and password so that firebase authentication can grant you access to our application.

Stage 3.1 Create an Account. After the user has entered their details it is then stored in the firebase database. Then they will be asked to login with that informations then they will gain full access to our app.

Stage 4 - Main Page. This is the page that the user will be redirected to after they have logged in. From this area the user has a wide range of functionality in which they can choose. For instance they can choose pay, plan route or they can look at their qr code.

Stage 5.1 - Pay. The user is greeted with a page where they have the option to start the Near field communication connection.

Stage 5.2 - Plan route. User is greeted with a map where they can pick a location on the map where they want to go.

Stage 5.3 - Qr Code. This is where the user can see their most recently bought ticket.

Stage 6 - Start NFC. This is where the clients application communicates and exchanges data with the server. Once the connection is made then the user will be greeted with a payment screen where they can complete their transaction.

Stage 7 - Payment. After the NFC connection is complete and the server has return the information from the database. Then the user is presented with a square payment page where they can ender in their debit card details to continue the payment.

Stage 8 - Ticket. User clicks pay and then is presented with their recently bought ticket. This comes in the form of a qr code which is generated using a qr code generator plugin.

Stage 9 - Open Map. The user clicks where they want to go to on the map and then they are given the option to choose whether they want to be redirected to google maps.

Stage 10 -Google Maps. the user clicks to be redirected to google maps where they can see what buses travel to their prefered destination. They can also see when the next bus will be

arriving along with how far away it is in terms of time and what distance it is away from their current location.

Problems and Resolutions.

4.1 Problems and Resolutions.

The first problem we encountered with our app was integrating our firebase plugins with android X. Due to the fact that flutter is relatively new it is always updating. In our case this meant that when we updated to the android x library it cause many problems to the compatibility of our plugins. Another big problem to android X was the lack of backwards compatibility and therefore we could not revert our changes. Somethings that had worked flawlessly before the update had stopped working altogether. To fix this it took a lot of research online to find out what versions of plugins were compatible together and how we would fix them. We also had to take out plugins that we were using and replace them with similar ones that worked just as well.

Payments were another problem that we faced when designing our app. When speaking to our advisor at the early stages we had both agreed that the best way to implement payments in our app was stripe. However when it came to implementing it there was massive problems between our framework flutter and stripe. After some research we realised that there is a problem with flutter when it comes to implementing stripe and that it wasn't going to be possible to implement stripe so that it could cater for our app. We decided that we would have to implement a different payment system so we decided to go with square payments. Even though there was some time wasted on stripe getting square implemented was a success.

Issues with plugins for maps. The compatibility of some versions of the plugins for maps cause problems when we were designing the plan route part of our app. This meant that we had to look up what plugin versions were compatible with our other plugins that we had already implemented.

Testing proved to be difficult because of flutter. When trying to test our code the test scripts we wrote were running through the language dart instead of flutter itself . This meant that the testing scripts wouldn't run and therefore wasn’t giving any results. However to solve this we had to change the configuration in flutter to show that we wanted the scripts to run through flutter and not dart.

Testing.

Testing was extremely important to our app as it was designed to work on both android and iOS. We used many different types of testing for example mockito/widget testing, unit testing and user acceptance testing.

5.1 Unit Testing.

With unit testing we tested single functions, methods and classes of our app. The goal of these tests is to verify the correctness of a unit of logic under a wide variety of different conditions.

5.2 Widget Testing.

We implemented widget testing to test single widgets that are in our code. Our goal was to verify that the widget's UI looks and interacts just as we expect. To test a widget we need multiple classes and requires a test environment that is suitable an appropriate widget lifecycle context.

5.3 Focus Group.

Another important aspect of testing is user testing. We created a focus group where each person were given dummy accounts and were told to try out our app. In which we could take feedback and figure out where we could make changes to our app to improve it. After the focus group we changed the design of the NFC screen increased the font size and added icons to the buttons on the main menu screen.

Installation Guide

6.1 Application installation and configuration

The application was designed with the framework flutter which means that this app is available with both the android operating system and the iOS operating system. At the moment the only way for a person to have the app on their phone is to connect their device to the computer that we are currently designing the app on. To get the app the person must first enable developer options, enable USB debugging and allow MTP for their connection. However we would love to upload an alpha version of our app onto the android store.

Appendices

7.1 Useful Resources

To keep track of our documentation we used google drive and google docs so we could add and change our files. This helped us keep on the same page when it came to our specification and details.

For finding different dart plugins we ended up using https://pub.dartlang.org. This was very useful as it laid out all all the plugins in a clear and concise manner. It also showed us top plugins so that we could choose the right one.

Udacity was very important to our project because it allowed us to do revision on different aspects of our project like UI and UX. The courses on Udacity really gave us a clear idea on how we were going to go about designing our user interface and user experience.

In regards to firebase and the setting up of firebase authentication. We used the firebase documentation that is on their website. Google really has done a great job in explaining the different ways you can use firebase in your project.

For testing we referenced flutter docs for information on testing. This is where we found out how to test units and widgets and make sure our app was running smoothly.

Blogging was a relatively new experience for both Michael and I so we decided to go with blogger.com. It is a free and easy to use online blog which made blogging simple.