03/02/2022

I began to create the database for the application using cloud firestore. Using the entity model diagram that I’ve designed previously, I created two collections. The first collection named ‘available seats’ will have a document titled ‘vehicle’. This will contain every individual vehicle associated with the transport operator. Each vehicle document will have a field for each seat with a Boolean value of true or false, which will represent whether a seat is available or not

The second collection is named ‘users’. It will have a document containing all the users that have signed up to the application. Each user will have a collection for tickets and a collection for travel cards, as well as a field for address\_line\_1, address\_line\_2, city, county, phone number and post code.

The ticket collection will contain a document that will store multiple tickets, with each ticket having a field for activated, amount, date, operator, price, route, transaction ID and type. Every ticket will have one transaction associated with it as the tickets are single use only.

The travel card collection will also contain a document which will store multiple travel cards. Each travel card will have a field for balance, bus cap, multimode cap, train cap, tram cap and type. A collection for the transactions associated with the travel card is included. Similar to the ticket collection, each transaction will contain the amount, date, operator, and route.

I’ve setup the pre-requisites for firebase in the flutter environment by following the instructions from <https://firebase.flutter.dev/docs/overview> and watching some YouTube tutorials <https://youtu.be/sfA3NWDBPZ4>

04/02/2022

I’ve set up the file structure in my application as follows:

* Config – Any application specific configurations e.g. colours
* Models – Groups of data
* Pages – Screens
* Services – Logic
* Widgets – Reusable widgets
* Providers – State management

I began the implementation of the sign-in and sign-up pages. My goal is to provide a form allowing the user to sign up or sign in with their credentials. For now I will be only implementing sign in functionality using email/password to test it out. In the future, I would like to integrate 3rd party services such as Google and Apple sign in.

On the sign in form, I’ve added a basic form with a field for the email and password. On the sign up form, I’ve added a basic form with a field for the email, password, first name, last name, phone number, address line 1 & 2, city, county and postcode.

Next I connected the application to my firebase authentication service using the firebase auth package. I created a file called firebase\_auth.dart. This file will contain functions relating to the interaction of authenticating with firebase i.e. Sign in, sign up, sign out and obtaining specific data relating to the user.

I created a user model object names user.dart. This file has 2 classes. The first is userModel. This class will provide us with the uid (User Id) only when we want to authenticate with firebase instead of the various parameter we can receive. The second class is UserInformation. This class will provide us with additional information that isn’t provided by firebase but is obtained through the sign up page. The parameters that will be accessed is the email, first name, last name, phone number, address line 1 & 2, city, county and postcode.

My next step was to create a wrapper for the application. If the user is already signed in, then they don’t need to go through the whole phase of signing in again, they’ll just be redirected to the home page. Similarly, if a user isn’t logged in, then we want to guide them to the sign in or sign up page.

I've added the provider dependency to use in our application. Using streams, it will allow us to access the incoming data across our application and handle its state i.e. display the sign in/sign up page or the home page

Later as the application develops, I will add a splash screen and walkthrough/onboarding screen before the sign up/sign in pages. For now I just want to make sure the basic functionalities are working and implemented.

So far, I’ve been able to successfully create a user with the sign up page and sign in to the application. What I now want to achieve is connecting to the database and creating a user collection with the document id equating to the uid of the user. This will mean that I can store information about the user with their specific id like the parameters in the sign up page and the ticket/travel card information.

06/02/22

As my demonstration for my interim report is tomorrow, I need to create my presentation slide on PowerPoint. I obtained a slide template from Slidesgo <https://slidesgo.com/theme/how-to-code-workshop#search-Editor%27s+Choice&position-7&results-1865>. My slides were divided into the following chapters

* Project Idea
* Research
* System Architecture
* System Design
* Technologies used
* Project Complexity
* Testing and Evaluation
* Demonstration
* Future Work
* Questions

To save time, I’ve pre recorded the demonstration of the Sign-In/Sign-Up Page, OpenTripPlanner Instance and View Stop On Map.

08/02/22

I’ve managed to achieve the goals set at the end of 04/02/22. What I want to work on now is to install and set up the QR Code package and have a simple instance of it working. My thought process is to attach the document id of the ticket in the QR code and create a query from there to access it.

For the NFC part of the ticket/travel card, I’ve ordered the NFC NTAG213 tags from ShopNFC - [NFC Stickers NTAG213 Round ø22mm - Shop NFC](https://www.shopnfc.com/en/nfc-stickers/115-nxp-ntag213-nfc-tags.html). I’ve also added the additional service of obtaining the tags unique id in a CVS file to save time on looking for it. They will be in my possession within a weeks’ time – more to follow!

13/02/2022

Over the past 5 days I’ve built the functionality for creating a QR code for the ticket. First, I created a variable with a firebase query to create a ticket document in the ticket collection based on the users id. By creating this variable, I can access the document id of the ticket, which I will be passing into the QR code alongside the uid. To make the QR code secure and protect the sensitive information of the document id and the uid, I’m going to be encrypting it with the AES Encryption technique using the encrypt package. I created a class that will contain the variables needed to encrypt/decrypt such as key, iv and encrypter. This class will also contain functions to encrypt, decrypt and retrieve the code.

When a QR code is encrypted, when scanned by a QR Code Scanner which doesn’t have the key that we’ve created, it’ll be unintelligible. However if the QR code scanner has the key, then it will be able to decrypt and understand the data we’ve passed. The QR Code scanner will be implemented for the admin/operator in authenticating the validity of the QR code.

What I need to do is to save this ticket document into firebase – however, I want to implement the stripe payment first before I save the ticket into firebase to give it the right flow.

I’ve also been successful in displaying the tickets/travel cards that is stored in the users account. I created a ticket class which will contain the structure of the ticket including the ticketID, activated, amount, date, transportOperator, price, route, transactionID and type. A list of tickets was also created, which will, be populated with the tickets from firebase. A firebase query to get all the tickets from the uid is mapped to the empty list of tickets. Using the structure from this website <https://medium.com/flutter-community/how-to-create-card-carousel-in-flutter-979bc8ecf19>, I’ve modelled the card carousel to display the tickets/travel cards. I’ve also created a ternary operation to change the colour of the ticket based on the type of ticket. For now I’ve displayed the id, status and type of the ticket.

19/02/22

|  |  |
| --- | --- |
| Live Stripe Key | sk\_live\_51KSjRkBYgfObXZIJ1ULs3Chf0F7byCOzmiy7Ey5BauVd2F3QNPV1DxYxHCW5mjSK9c3nCAG9xXiS0hB7yRgd8gE2009WqIyeHs |

I’ve finished the implementation of paying for the ticket. I first began by importing the http and flutter\_stripe packages. I had to make some alterations to the androidManifest.xml, MainActivity.kt. styles.xml, gradle-wrapper.properties and the build.gradle files to accommodate the flutter\_stripe package. I also added Firebase Functions to my application using the firebase cli and NodeJS. The installation process concluded with a folder named functions where our functions will reside. The function contains code to grab the customer details, creating a secret key and initialising a payment intent. The response from the function will then by used to start the payment in add\_ticket.dart. At this point a new document for the ticket will be created with some details filled in using the initUserTicket function in the class FDB. Its document id will be passed alongside the user id to use\_ticket.dart where the encrypted QR code will be created.

The NFC tags have also arrived too so I will begin programming it in the next coming days.