



New era in payments
Smart Wallet
by
Stelian Claudiu
Supervisor: Renuga Jayakumar

Project submitted as part of the requirements for the award of BSc (Hons)
Cloud Computing

March 2022

DECLARATION FORM AND FORM OF CONSENT

DECLARATION

I, Claudiu Stelian declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Claudiu Stelian

Date: 16.03.2022

FORM OF CONSENT

I Claudiu Stelian hereby consent that my Project, submitted in candidature for the B.Sc. (Hons) Cloud Computing degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Claudiu Stelian

Date: 16.03.2022

Table of content

1	Introduction.....	7
1.1	Background	7
1.2	Aim.....	7
1.3	Objectives.....	7
1.4	Report Structure	8
2	Review of Literature	9
2.1	Introduction	9
2.2	Research on the problem domain	9
2.3	Evaluating existing examples.....	11
2.3.1	APPLE WALLET.....	11
2.3.2	GOOGLE PAY	13
2.3.3	SAMSUNG PAY	16
2.4	Analysis on Tools required	18
2.4.1.1	ANDROID STUDIO.....	18
2.4.2	FireBase	19
2.5	Mobile Application development.....	20
2.1	Software Development Life Cycle.....	21
2.1.1	Phase 1: Concept.....	21
2.1.2	Phase 2: Project Planning.....	21
2.1.3	Phase 3: Development.....	22
2.1.4	Phase 4: Closing.....	22
3	Analysis.....	23
3.1	Requirement Analysis:	23
3.1.1	Functional Requirements	23
3.1.2	Non-functional Requirements.....	23
3.2	WorkFlow.....	24
3.3	System Requirements.....	25
3.3.1	Hardware Requirements.....	25
3.3.2	Software Requirements	25
3.4	Risk Analysis.....	26
4	Design	28
4.1	App interface	28
4.2	Database design.....	33

5	Implementation	34
5.1	Main module explanation.....	34
6	Testing.....	50
7	Evaluation	56
7.1	Personal evalation	56
7.2	Project evaluation.....	57
8	Further Enhancement.....	58
9	Conclusion	60
10	References.....	61
11	Appendices.....	63

Table of figures

Figure 1 - Traditional database vs Firebase database	20
Figure 2- UML Diagram.....	24
Figure 3- Create Account and Sign in interface.....	28
Figure 4-SmartWallet action bar.....	28
Figure 5-Payment Fragment	29
Figure 6-Google Pay installation process	29
Figure 7-Pdf files upload and display feature	30
Figure 8- Calendar Fragment.....	30
Figure 9- Business card Fragment	31
Figure 10 Settings fragment.....	32
Figure 11 Database Structure.....	33
Figure 12 - Layout xml scripts.....	34
Figure 13 - Layout Script for Register Page	35
Figure 14 - Register Page Display	35
Figure 15 - Java libraries.....	36
Figure 16 - Register code in Java.....	37
Figure 17 - User Details with database connection	37
Figure 18 - Checking an existing Account	38
Figure 19 - Login redirect for account verification	38
Figure 20 - Password check	39
Figure 21 - Verification e-mail	39
Figure 22 - Firebase email verification in Java.....	39
Figure 23 - Navigation Bar inserted with MeowButtonNavigation.....	40
Figure 24 - Fragment activities	40
Figure 25 - Fetch current user data	41
Figure 26 - Get pdf files for current user	41
Figure 27 - ListView Frame.....	41
Figure 28 - Button for Upload Pdf.....	42
Figure 29 - Upload Pdf	42

Figure 30 - Class for selected pdf files	42
Figure 31 - Creating link for pdf files	43
Figure 32 - Check the pdf file existence	43
Figure 33 - Payment Fragment class.....	44
Figure 34 - Text field variables.....	45
Figure 35 - Fetching user's Business Card details.....	45
Figure 36 - Business User Details.....	46
Figure 37 - Open Pdf class.....	46
Figure 38 - Method to Create Pdf	47
Figure 39 - Method for screenshot.....	47
Figure 40 - Method for Popup Window.....	48
Figure 41 - Buttons in the Settings Fragment	48
Figure 42 - Sing out listener.....	49
Figure 43 - Sign-out onClick Allocation to logout button.....	49

Table of tables

Table 1 - Risk analysis table	27
Table 2 - Application Features Testing table	55
Table 3 - Personal Evaluation	56
Table 4 - Project Evaluation	57
Table 5 – Administrator Further Enhancement	58
Table 6 - User Further Enhancement	59
Table 7 - Pros and Cons of multiple eWallets applications	63

Abstract

This report is presenting a new era in payments: the eWallets. The most known digital wallets on the market: Apple Wallet, Google Pay, Samsung Pay are analysed from multiple perspective: the technologies that they use, the benefits and the drawbacks and the features of each application.

On this purpose, a new wallet application has been created. The design and implementation of this application is being discussed.

Another topic that is covered in this report talks about the tools that are being used to develop a new application: Android Studio and Firebase. These technologies are presented because they are a core part of the wallet application development.

This report also contains the evaluation, testing and future recommendations for the new application.

Other topic of this report is the testing and evaluation of the project. Another point of this project is to underline the importance and convenience of using an eWallet.

1 Introduction

1.1 BACKGROUND

In 1997, Coca-Cola owned a few vending machines that allow the customers to pay via SMS. They were placed in Helsinki and is believed that this the origin of a new transaction era – mobile transactions

Five years later, around 95 million phones were used to buy movie tickets, hotel bookings or food ordering.

Google launch in 2011 a mobile wallet. The NFC (near field communications) technology was used to pay and earn loyalty points. It was working on a single model of mobile phone, and it was accepted by a few merchants.

In 2012 Apple launched Apple's Passbook meant to the store the tickets, boarding passes, and coupons. [1]

PayPal is another major company that had an impact on the e-commerce. PayPal allowed the Users to link the back account or debit/cards to their PayPal account and to shop online with a single tap. Since then, multiple digital wallets appeared on the market: GrabPay, Lazada Wallet and this type of payment became popular.

1.2 AIM

The aim of this project is to present the usefulness and the importance of having a smart wallet application.

1.3 OBJECTIVES

The objectives of the project are:

1. To analyse the products and
2. To evaluate the market of the products that are already on the market
3. To develop an app e-type wallet
4. To underline the importance and easy handle of an eWallet
5. To point the negative part of the project

1.4 REPORT STRUCTURE

1. **Chapter 1 – Introduction:** This chapter is providing information about the first steps of digital payment era, the aim and the objectives of this project
2. **Chapter 2 – Review of Literature:** The main points of this chapter are: a problem domain discussion and an analyse of the existing applications on the market
3. **Chapter 3 – Analysis:** This chapter will provide an analysis on the functional requirements, non-functional requirements
4. **Chapter 4 – Design:** This chapter will provide details about Smart Wallet project design.
5. **Chapter 5 – Implementation:** This chapter will provide details about Smart Wallet project implementation.
6. **Chapter 6 – Testing:** In this chapter the Smart Wallet application is being tested
7. **Chapter 7 – Evaluation:** This chapter evaluates the project and provides a conclusion from the data collected.
8. **Chapter 8 – Further Enhancements:** This chapter is presenting future enhancements for the SmartWallet application
9. **Chapter 9 – Conclusion:** This chapter is a brief of the project achievements
10. **Chapter 10 – References:** This chapter presents the references that are being used.
11. **Chapter 11 – Appendices:** Supplementary information on the subjects discussed on the project.

2 Review of Literature

2.1 INTRODUCTION

This literature review evaluates four common mobile development frameworks, identifying common features between them that can be used into a personalized-build framework, which will be used as a frame to develop mobile applications.

2.2 RESEARCH ON THE PROBLEM DOMAIN

Contactless payments have become the most common payment system. Electronical wallets have been successfully not only for consumers but also for companies. Banks and businesses from various fields, believe that the contactless payments will replace the physical wallet in the long run.

Before getting into details, e-wallet can store user's financial information and transaction details that is based on smartphone secure design. It is also used to make and receive payments.

The amount of electronical wallet applications increased over the time. The expectations to increase consumer use of digital wallets are due to the availability of numerous mobile wallets apps. The first major companies that developed or used personal digital wallet are the technology companies such as Google, Facebook, and Alibaba.

Digital wallet became very successful for both, users, and companies. The transactions made by using an e-wallet are more frequent today and known a growth over the past years. Since this an easy way to execute transactions, it is predicted that more customers will use this way of payment.

In 2020, the digital wallet market exceled to 100 billion dollars and over 100 billion transactions, that cost almost 6 trillion USD. It is estimated that between 2021 and 2021 the compound annual growth rate will reach 20%.

Developing an eWallet software can be difficult and the developer can face multiple issues, that are not just technological.

1. Changing the old habits can be difficult

One of the most common difficulties is the changing the old payment habits of a customer. It is hard to convince people to use new technologies, especially when their moneys are involved. To gather more customers to use digital wallets there are market strategies that offers discounts and benefits for using this technology. Making people aware that this is a safety and convenient method to pay, needs a change in customer mindset.

2. Trust Issues

Based on research made by YouGov, 40% of mobile phone users have trust issues with digital wallets. One of the main concerns is their phone get stolen and they will lose their money. Confidentiality is a key factor on this customer trust. More than 55% of people feel more safer if they have the credit cards with them, rather than adopting the eWallet payment method.

3. No training or awareness programs

Customer usually think that using cash is simpler and the digital wallets are hard to use and are not designed for them. One solution to this problem is to train and create awareness programs for customers to use voice commands.

4. Fraud Detection & Mitigation

One of the major and significant problem are the frauds that can create problems in developing a digital wallet as people who are using electronical wallet can be easily scammed. The risk of digital wallet payment is associated with malware attack, hacking, data leak and data theft. This is one of the main reasons why the people avoid using eWallets so they can avoid a cyberattack. Since the technology has always been advancing the developers started to avoid this sort of threat by increasing the security system of the eWallets. Recently most digital payments applications are compliant with European Union General Data Protection Regulation (GDPR) compliance.

5. The Need for Regulatory Compliance

Regular standards apply when it takes using digital payment services. Finances and money are securely regulated by the banks and the eWallets services cannot be provided if the government does not allow them. Choosing the right eWallet is crucial as each of the country has different norms and regulations. A clear set of instructions must be listed before developing an eWallet as Emizen Tech's Wallet mobile application development team will review them.

6. Technological Problems

Hiring digital wallet developers with experience is important. The application should be easy and safe to use. It is also important to have a reliable application, that doesn't crash while a transaction is executed. Machine Learning and Artificial Intelligence can play a crucial role in the application development. It can gather information about the UI design that customers are familiar.

7. Lack of Rewards

Customers complain that they would like to have more benefits while using the mobile wallets, because coupons and discounts are offered when using credit cards or cash. In order to increase this way of payment, the digital wallets should have more advantages. [5]

Conclusion

With the advanced eWallets technology credit/debit cards and virtual currency will become a primary payment mechanism. Thanks to the convenience of mobile devices eWallet transactions are one of the trendiest systems. The key in launching an eWallet application are the challenges and trends that can be encountered. For developing an eWallet is recommended to choose an eWallet development company with professionals who are familiar with the rules and regulations.

2.3 EVALUATING EXISTING EXAMPLES

The pandemic has not only altered consumers' shopping behaviours, but it's also changed how they pay for goods and services.

The pay for goods and service and consumer's shopping behaviours has been change drastically since the pandemic situation from 2019 where around 72 million people enhanced the electronic payment system. [6]

Most known digital wallets are:

2.3.1 APPLE WALLET

Apple Wallet priorly named Passbook, is a mobile application developed by Apple for iOS and watchOS that allows users to store personal documents such as boarding passes, student ID cards, business credentials, resort passes, car keys, government ID cards, credit/debit cards, prepaid cards, loyalty cards and many others. On September 19, 2012, Wallet was introduced to the public as Passbook which three years later in 2015 when the release of iOS 9 took place the Passbook was changed into Wallet.

2.3.1.1 How it works?

The customer needs to scan a QR code that is generated by a system and to interact with a pass. Apple wallet is using a PassKit framework that allow the application and system to communicate with passes from the Wallet.

Passes are organized in a package; the package is a pass template and has a private key. The pass is represented by a JSON file called pass.json that will define the pass. JSON stores all the information of the pass.

The first known installations of iBeacon wireless geofencing in retail establishments started in the United States in late 2014. The business was able to send an unrequested lock-screen notice to cell phones within Bluetooth range thanks to the iBeacon technologies. [13]

Apple added the option to use NFC to display a loyalty card in Wallet to a point-of-sale terminal in 2015. [14] Customers can add their card to Wallet through the Walgreens or Duane Reade mobile applications and tap their phone to the terminal when prompted for their rewards card, which was originally done by Walgreens with their Balance Rewards loyalty programme. In 2018, support for contactless student IDs was implemented for a few universities. Such credentials can be tapped into supported readers in the same way that real RFID cards can.

When Express Mode is activated, the power reserve feature on select iPhones and Apple Watches allows the card to be used even when the battery is empty.

Car keys were made available for the Apple Wallet in 2020, starting with BMW vehicles that were supported. Friends and relatives can use these keys, and permissions can be specified for each member, such as how fast they can drive and whether certain vehicle-specific functions can be enabled or disabled. In addition to NFC, such passes can be used with UWB. [17]

Apple announced in 2021 that Wallet would be able to store government ID cards, resort passes, home keys, and business credentials.

2.3.1.2 Features

Add Unsupported Passes to Wallet

It can transform physical cards with barcodes into digital cards that that user can add it to Apple Wallet using third-party app.

The system is called Pass2U and by scanning a physical barcode can convert it to an Apple-compatible digital barcode. That can be inserted directly into the Wallet app. Pass2U won't work with every barcode scanned, but it can work with few loyalty cards that Apple doesn't officially support.

Rearrange Passes

The Apple Wallet offers the feature of rearrange the passes in any order the user choose to for a faster and easier handling as in most of the cases some passes are used more often than the others.

To use a pass, simply tap and hold it. Users may move it up or down and set it anywhere they want it once it has slid up ever so slightly. This is especially beneficial if they have a lot of passes and find themselves scrolling through them to find the ones they need. If they want to rapidly change their default credit card with Apple Pay, they can simply drag the one they want to the front.

Share Passes with Friends & Family

To access this feature the user will have to open a pass and hit the I button in the bottom-right corner. Then select "Share Pass" from the drop-down menu.

Sending a pass via iMessage, AirDrop, or the default Mail app are all options for sharing a pass. Users can also use any other third-party software that integrates with Apple Wallet.[9]

2.3.1.3 PROS AND CONS

PROS:

- NFC payment solution that is quick and simple.
- Credit card data is kept secure via an anonymous, single-use token mechanism.
- Gift cards, movie tickets, boarding permits, and other items are all managed by this programme.
- Option for a high-reward credit card

CONS:

- Apple gadgets are the only ones that work.
- Purchasing with facial recognition in stores is more difficult than paying with Android.
- More difficult to set up than other competitors.

2.3.1 4 Conclusion

Since the outbreak, Apple Pay has gained a significant lead over competing digital wallets, and it's anticipated that the total amount of eligible in-store Apple Pay transactions will rise 74 percent this year to \$91.7 billion. [19] This success is owed in large part to the fact that the business has grown: more retailers are accepting digital wallets, and more people are using Apple Pay-enabled iPhones, resulting in increased retail spending. Consumers, on the other hand, use it for a small percentage of eligible transactions, and Apple has failed to enhance this capture rate over time. In the post-pandemic new normal, contactless credit and debit card purchases are becoming the preferred method of payment. To avoid becoming obsolete, Apple Pay will need to improve quickly.

2.3.2 GOOGLE PAY

All of Google's digital payment tools are referred to as Google Pay. In 2018, Google Wallet (introduced in 2011) and Android Pay (released in 2015) merged to form Google Pay. Google Pay is officially supported by all Android phones running version 5.0 (Lollipop) or later, as well as iOS 12.0 or later. The app works with Android Wear 2.0 smartwatches as well. Google Pay or G-Pay, like other mobile payment programmes, stores debit card information so it may make contactless payments throughout app, online or in-person on devices such as smartphones, tablets, or smartwatches. The payment app wants to be a one-stop shop for all the financial management that needs to be done daily. To make purchases, it's not necessary to bring a wallet or carry any cards. All that must be done is to download the app, sign in with Google account, and choose any chosen payment method.

2.3.2.1 How it works?

To transfer your card information at any contactless payment terminal, Google Pay uses Near Field Communication (NFC) technology, also known as tap-to-pay. It allows the user to pay without opening the app by unlocking the phone with their fingerprint or a security passcode.

2.3.2.2 Google Pay Features

Loyalty cards, gift cards, bank accounts, coupons, college IDs, movie tickets, boarding passes, event tickets, and even public transportation tickets can all be loaded into Google Pay. The attractive feature of Google Pay's e-wallets is that the user may eliminate the trouble of carrying cards, passes, or tickets in your physical wallet, as well as fumbling through the wallet or purse to find them. After you've loaded a scanned copy on the phone, the user won't have to worry about leaving them at home.

Peer-to-Peer Payments

It can send/request and receive money from and to other Google Pay users via the Google Pay P2P Payment service. The money received is promptly put into bank accounts, unlike other digital wallets.

Users can pay with existing Google Pay balance or a direct transfer from their checking account. Users need to transfer money from their bank account, debit/credit card (fees will apply), or other Google Pay-accepted payment methods to their Google Pay balance.

Taping on one of the contacts in the app, a chat-like view of your transaction history between the two of users appears. Users can also form a group where members can send and request money from one another or split a bill or rent. It keeps track of who has or hasn't paid their share, and with a press of a button it reminds them to do so.

To assist international money transfers, Google Pay has formed partnerships with international money transfer businesses such as Western Union and Wise.

Enhanced Security

Google Pay requires identification such as fingerprint ID, pin, or pattern to launch the app or make a payment, ensuring that only you have access to your account.

To give an additional degree of security against fraudulent transactions, Google Pay employs a tokenization technique. During any purchase, the software replaces the debit card or credit card number with a randomly created virtual account number or tokens. When making payments online, the information is secured to protect customers details. When money transfer is made to someone who isn't in the contact list, Google Pay also sends an alert.

If the phone is stolen or lost, user can use Google's Find My Device tool to remotely lock it.

Privacy Controls

By turning on "Personalization" it allows Google's algorithms to curate and collect bargains and offers for user's based on their transaction history, payment methods, and loyalty cards. This isn't the default option. Users can test it out for three months to determine if they like it. After three months, users can opt out and only receive generic offers. Users' data will not be sold to third parties or shared with retailers, according to Google. If they do decide to take advantage of the incentives, the merchants involved will be notified. In addition, your translation data will not be shared with the rest of Google.

Extra Features

Google Pay offers multiple possibilities, from ordering food, buying gas to pay for parking stations. It has the availability to search for gas station based on your location. For raising the popularity, Google Pay offer multiple rewards if a friend invites someone to join and use Google Pay. [7]

2.3.2.3 PROS and CONS

PROS:

- Supports credit/debit cards, gift cards, loyalty cards and passes.
- In-store, in-apps, online purchases system.
- Supported by major U.S. & foreign banks.
- Protect user data.
- Send & request money

CONS:

- Not accepted at all stores.
- Not supported by all banks.
- Need eligible devices to work.
- 1.5% fee for debit card transactions.
- Can send money only to Google Pay users.

2.3.2.4 Conclusion

Google Pay has a large array of features in a single place. It can be a good option for customers that have to do mobile payments or need money management together. On the other hand, some of the Google Pay functionalities are not that good compared to other applications.

2.3.3 SAMSUNG PAY

Apple Pay and Google Pay are identical to Samsung Pay. It's a platform for Samsung smartphones and wearables that lets the user pay for products and services without using cash or credit cards by tapping their Samsung device near a contactless payment terminal or authorising an online payment.

2.3.3.1 *How it works?*

When using a phone, open the Samsung Pay app by swiping up from the bottom of the screen. The default card will appear, along with a message asking to validate a payment using a PIN, fingerprint, or iris scanner on compatible devices. Users can also use the app to launch Samsung Pay.

After the payment has been biometrically approved, the phone instructs the users to touch the phone against the contactless payment scanner, and a payment is performed using NFC (near field communication).

In some places, such as the United States, Samsung Pay goes beyond NFC by including MST (Magnetic Strip Technology), a mobile wallet technology which is used when LoopPay is taking place.

MST payments is used for terminals that do not have NFC readers (primarily outside the UK), that opens the payment technology to a far larger number of shops.

The following is how a two-step payment process works: On a mobile device, LoopPay's software manages and securely keeps all the payment cards (including credit, debit, loyalty, and gift cards), while the LoopPay device (LoopPay Fob, Card, ChargeCase, or CardCase) processes the payment at the checkout.[8]

2.3.3.2 *Samsung Pay Features*

Immediate Activation

Most Samsung handsets come pre-installed with Samsung Pay service, which allows users to make cardless payments. If the app isn't available, consumers may always download it from Google Play and enter their card information to use it.

Affordable Digital Wallet

Other digital wallet services are only compatible with certain terminals and card readers. Samsung Pay, on the other hand, works with systems that accept magnetic stripe, EMV, and NFC chip cards, allowing companies to offer contactless payment to in-store customers without having to replace their registers or card readers.

Fast and Secure Online Payment

Samsung Pay allows online shoppers to make quick and safe purchases. Instead of entering their card details for each purchase, customers can just use their fingerprint to authorise the following actions. For authentication, they can even use iris scans or passcodes.

Rewarding Application

Samsung Pay allows users to use any card that has a barcode. This means they can easily update their loyalty programmes by adding barcodes of certificates, cards, and discount offerings, allowing customers to easily store them in their digital wallets. Even if they don't have their real card with them the users can still earn points and benefits this way.[8]

2.3.3.3 PROS and CONS

PROS:

- The app is secure and easy to use with just of a fingerprint scan.
- The user-friendly interface as the payments can be made with a mobile device rather than physical card.
- It is easy, good, and reliable to use.
- It can be used on the majority Samsung devices such as smartwatches, phones and tablets.

CONS:

- It works only on Samsung devices
- Not all the banks are supported by Samsung Pay.

2.3.3.4 Conclusion

Samsung Pay is a global payment system that may be used anywhere. The users should be able to make a payment with Samsung Pay despite they can use their physical card in the country they are visiting. Samsung Pay is a quick and easy way to pay without having to go through your wallet or handbag. The credit and debit cards number will never be kept on mobile device.

2.4 ANALYSIS ON TOOLS REQUIRED

Developing an app takes a series of tasks such as its main functionality, platform that will run on and what framework will suit it. Choosing the framework is the most important task as each of them are using different programming languages and here is some information about them. [2]

2.4.1.1 ANDROID STUDIO

Android is an open-source operating system that gives the freely customisation to the manufacturers without any fixed hardware or software configuration. The base Android OS has many features that includes:

1. Storage – a lightweight relational database (SQLite) for data storage.
2. Connectivity – GSM/EDGE, CDMA, EV-DO, IDEN, UMTS, Bluetooth, Wi-Fi, LTE and WiMAX.
3. Messaging- SMS and MMS.
4. Media support -MP3, AMR, AAC, HE-AAC etc.
5. Hardware support – Accelerometer sensor, camera, digital compass, proximity sensor and GPS.
6. Multi-touch – screen that offer multi-touch feature.
7. Multi-tasking – the ability of running multiple application at the same time
8. Tethering – the ability of sharing internet connection via wireless or wired hotspot.[3]

Android Studio based on IntelliJ IDEA (Integrated Development Environment for Android) provides features that enhance the productivity when building android apps and some of them are:

- Fast mobile emulator
- Unified environment where the apps can be developed for all the Android devices
- Flexible build Gradle-based system
- Apply changes to push code and resource changes to the running app without restarting it
- GitHub and templates integration for easy code implementation
- Extensive software development system (frameworks) and testing
- Tools to catch usability, performance, version compatibility and other problems
- NDK and C++ support
- Built-in support for Google Cloud Platform for an easy Google Cloud Messaging and App Engine integration [4]

2.4.2 FireBase

Firebase is a set of tools that let the user to "create, improve, and expand an app," and the tools provides a set of services that developers would usually have to build themselves, but they don't want to because they'd rather focus on the app experience. Analytics, authentication, databases, configuration, file storage, push messaging, and so on are all part of this. Firebase offers cloud-based hosting which helps the developers to focus on other application aspects.

Backend components are maintained and operated by Google's cloud hosting services. Firebase's client SDKs interacts with back-end services directly without establishing any middleware between the app and services. Using Firebase database options, the user can write code to query the database in the client app.

Traditional app development often entails writing both frontend and backend code. The frontend code just calls the backends' API endpoints, and the backend code takes care of the rest. The traditional backend is bypassed with Firebase products, putting the job on the client. The Firebase console provides administrative access to each of these products.

“Build” section offers a set of products:

Authentication — can store the user's login details

Realtime Database — offer service such as NoSQL database, cloud hosted in real-time

Cloud Firestore — offer service such as NoSQL database, cloud hosted in real-time

Cloud Storage — file storage that can be massively increased

Cloud Functions — backend events driven without server

Firebase Hosting — hosting globally web

ML Kit — software development kit for common meta language task

It's important to note that Firestore is a Google Cloud service, not a Firebase service. Why is it in Firebase's database? Firebase provides SDKs that can be used in the mobile app to provide direct data access, eliminating the requirement for a middleware component. Also, to mention that there are other items listed here that have a similar interaction with Google Cloud.[20]

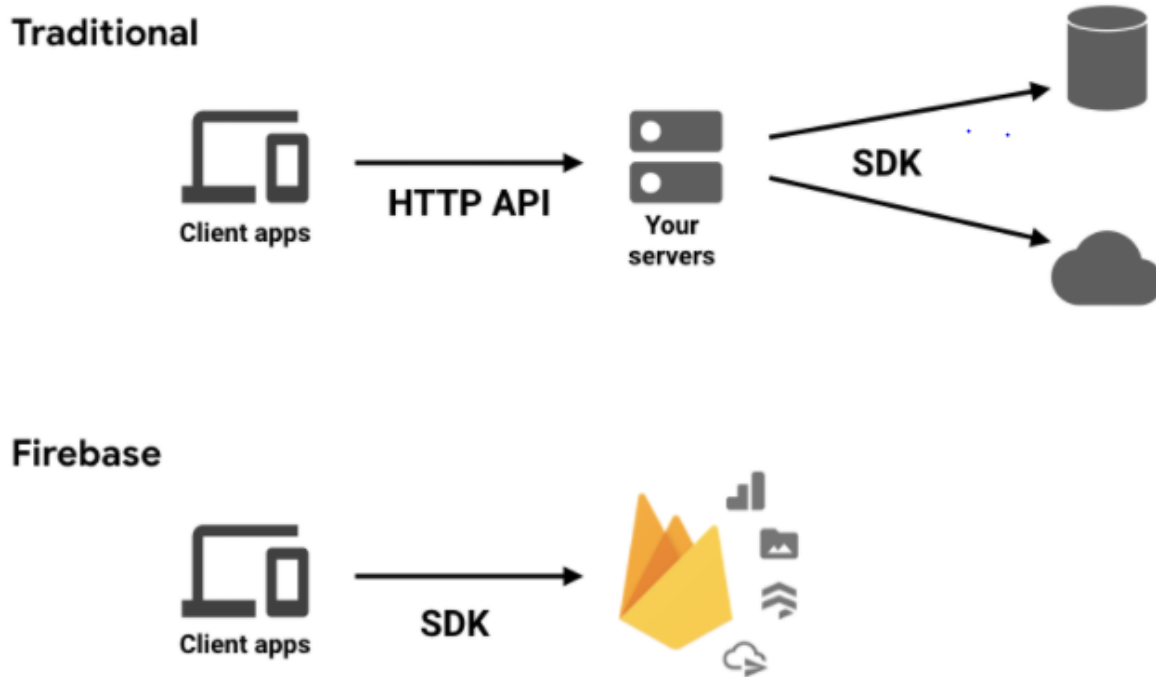


Figure 1 - Traditional database vs Firebase database

2.5 MOBILE APPLICATION DEVELOPMENT

Mobile application development is the process of creating a software application that can run on a mobile device. Mobile application development has its root in more traditional software development like web application development. In some of the cases some mobile apps are often written specifically to take advantage of some unique features of a particular mobile device. Good examples can be a mobile gaming app that can take the advantage of the I Phone's accelerometer or navigating a cursor on the Apple Watch's display through a series of hand gestures using a motion sensor like gyroscope.[2]

These days the most used mobile platforms are Android from Google and IOS from Apple that each of the devices comes with preloaded essential applications such as Apple App Store and Google Play Store which give the access to a larger variety of applications and games. [2] Back in the days of mobile applications the only way to ensure an application could perform optimally on any device it was to develop the app natively. The new code had to be written specifically for each device's specific processor as each device required its own mobile app development project with its own code base. [2]

Modern cross-platform tools use same program languages such as C#, and JavaScript to share the code across projects enabling the developers to use a single codebase for Google Android, Apple IOS and progressive we apps (PWAs). [2]

2.1 SOFTWARE DEVELOPMENT LIFE CYCLE

The development methodology is agile. It will contain 3 sprints and each sprint will have multiple steps: Plan, Design, Development, Test, Deploy, Review, Launch.

2.1.1 Phase 1: Concept

The main point of this project is to create a digital wallet mobile application.
The Application goals are:

- To enable mobile payment
- To store pdf files: flight tickets.
- To have scheduled events
- To create a customer business card

2.1.2 Phase 2: Project Planning

2.6.2.1 Task priority has been assigned.

High importance tasks are:

- A user-friendly design
- A simple and convenient way to use the application
- To find the tools that are required for the project development
- To create a related database
- To decide which platform is fitted to be used for mobile payment
- To choose the program language
- To create a register/login system
- To create a payment fragment in the application

Medium importance tasks are:

- To have a feature for uploading pdf files in the application
- To create a calendar with scheduled events
- To have an email account confirmation
- To have the ability of changing the account password

Low importance tasks are:

- To store multiple cards in the application
- Contact us page
- Administrator functionalities

2.1.2.1 *Deadlines*

- Establish the application design and features 3 weeks
- Establish the main development tools and frameworks: 1 week
- Login/Register page: 3 weeks
- Enable Payments: 3 weeks
- Business Card implementation: 3 weeks
- Pdf uploading files: 3 weeks
- Calendar with events: 3 weeks
- Setting implementation: 2 weeks

2.1.2.2 *Budget*

The allocated budget for this application is 200 pounds

2.1.3 Phase 3: Development

- Revision of development specification
- Critical design review
- Writing the code of the application in a development environment
- Programmer is testing the main functionalities
- The code is migrated to Unit Test Environment.
- Quality Assurance testing
- Deployment
- Launch
- User Testing
- Maintenance

2.1.4 Phase 4: Closing

- Continued existence
- Feasibility
- Performance

3 Analysis

3.1 REQUIREMENT ANALYSIS:

3.1.1 Functional Requirements

The eWallet application features are:

- User Register: The user can create an account
- User Login/ User logout/ Password reset
- E-mail account confirmation
- Attach and access pdf files to the application
- Contactless payments: Payments will be done via Google pay. On payments app the user will have the option of paying contactless by pressing on google pay button which will redirect into Google Pay 3rd party app. Add a notification on the calendar for a easy time management.
- Events and schedule management: You can set schedule reminders on the app's calendar that will notify the user the day before of the event.
- Business card: The app will have its own business card template based on each user's personal details that will allow to add other's people's business cards or share his one via email or any other message app. The user will be able to create a business card by simply pressing Edit button on Business card tab.
- Contact the developer in case of failure
- Access data from any mobile device

3.1.2 Non-functional Requirements

- Scalability: Firebase database permits unlimited amount of storage using the Firebase subscription. This makes the database scalable.
- Reliability: eWallet is using third party payment system, so called Google Pay. This future will be used through Google Pay application.

- Recoverability: The code is stored GitHub, and database is stored into a cloud database which can be accessed from any device and any place that has internet connection. Backup are executed daily. [GitHub repository link](#)
- Performance: The NO-SQL database offers a faster performance in data retrieve. This tool is making the application move faster.

3.2 WORKFLOW

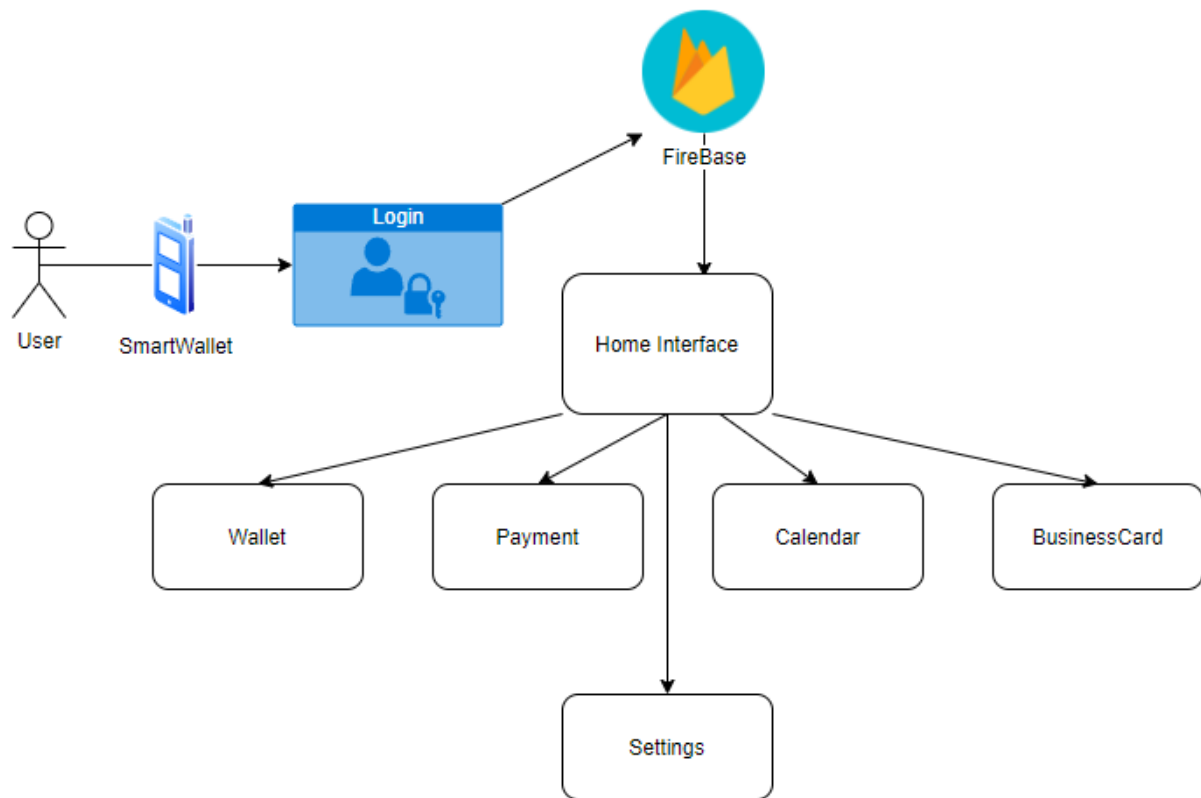


Figure 2- UML Diagram

3.3 SYSTEM REQUIREMENTS

3.3.1 Hardware Requirements

The application can run on any Android device and the recommended hardware requirement are:

- Chipset - Exynos 9825 (7 nm) - EMEA/LATAM, Qualcomm SM8150 Snapdragon 855 (7 nm) - USA/China
- Network technology - GSM / CDMA / HSPA / EVDO / LTE / 5G
- CPU - Octa-core (2x2.73 GHz Mongoose M4 & 2x2.4 GHz Cortex-A75 & 4x1.9 GHz Cortex-A55) - EMEA/LATAM, Octa-core (1x2.84 GHz Kryo 485 & 3x2.42 GHz Kryo 485 & 4x1.78 GHz Kryo 485) - USA/China
- GPU - Mali-G76 MP12 - EMEA/LATAM, Adreno 640 - USA/China
- Communications WLAN - Wi-Fi 802.11 a/b/g/n/ac/6, dual-band, Wi-Fi Direct, hotspot
- NFC near field communication

3.3.2 Software Requirements

- OS - Android 9.0 (Pie), upgradable to Android 12, One UI 4.
- Google Pay for payment system
- Pdf reader application

3.4 RISK ANALYSIS

Risk Event	Probability of the risk happening	Impact Rating	Mitigation
The risk of losing the mobile device	70%	Low impact as all the data stored by the user will be into the cloud server	The risk cannot be avoided
System failure	40%	low impact as the data will not change	Keeping the android and application updated always can avoid system failure
Database failure	15%	Low impact as the data is stored in cloud database	Regular database backups can be made to avoid database data loss
Connectivity issues	50%	Medium impact as the application needs internet connection	Use reliable internet providers this can be avoided
Password loss	5%	Low impact as the can be easily recovered	The application has a forgot password feature that can be used
Wrong information about the user	15%	Low impact as the data is stored in a no-sql database that is using column filtering	NA
Security breach	45%	Medium impact as phishing application is available and can retain the user credentials	User phishing knowledge is important and can play a key factor
Events not being triggered	80%	High impact because the feature is not fully tested	User experience can provide more details about a possible event issue

SmartWallet Project

Slow Performance	10%	Low impact because the database is stored in cloud and is using the no-sql database to gain interrogation performances	NA
Low scalability	10%	Low Impact because the database is scalable, and the Google pay is being used regularly by other mobile owners	Keeping the database clean and data reliable can avoid scalability issues
Development delay	60%	Medium impact as the one developer is implicated in product development	Hiring multiple employees can solve this issue

Table 1 - Risk analysis table

4 Design

4.1 APP INTERFACE

The User Interface of SmartWallet application has an intuitive design and User-friendly features. It has a simple and easy to use design formed out of motioned buttons and text fields designs.

The design of SmartWallet application is based on motion icon's bar menu and multicolour contrasts. SmartWallet has multifunctionalities and they will be presented in this chapter.

The first part of design represents the Login/Register step.

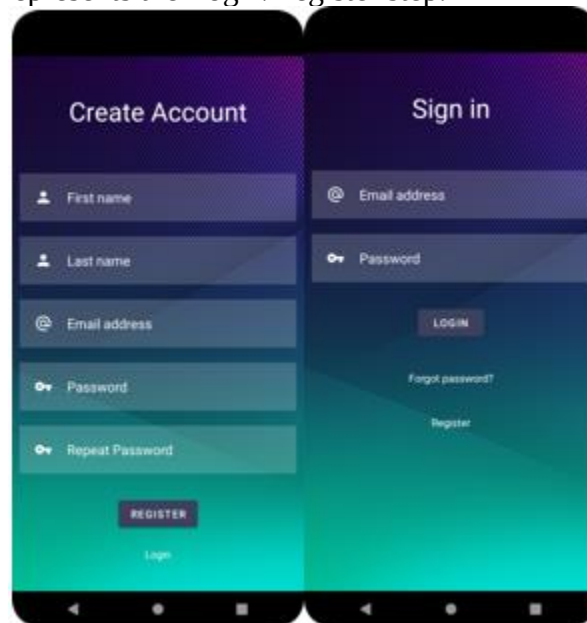


Figure 3- Create Account and Sign in interface

Register screen has multiple text filled that must be filled with User's personal information's such as First name, Last name, Email address, Password, and password confirmation. Once the field are populated the register button will create a user's account and it will redirect the User on the Login page.

After User inserts the login details and press the LOGIN button, he will receive an email for account confirmation. Once the email is confirmed the User will be able to access the main features of the SmartWallet application.

The main application activity contains of 5 different fragments that offers multiple functionalities.



Figure 4-SmartWallet action bar

SmartWallet Project

The first fragment offers the payment functionality which can be used by pressing the Google Pay button.

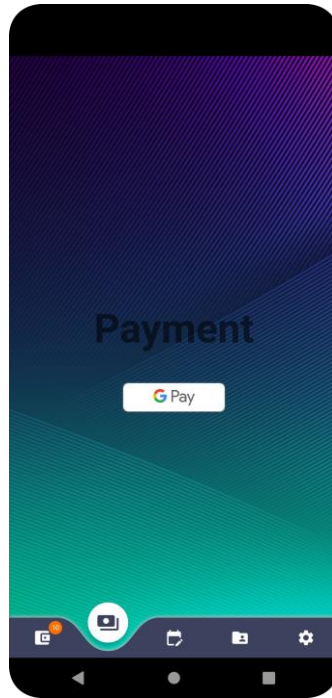


Figure 5-Payment Fragment

Google pay button will redirect the User to the Google Play Store to help the User to install or open the Google Pay application to add the credit or debit cards for an easy contactless payment.

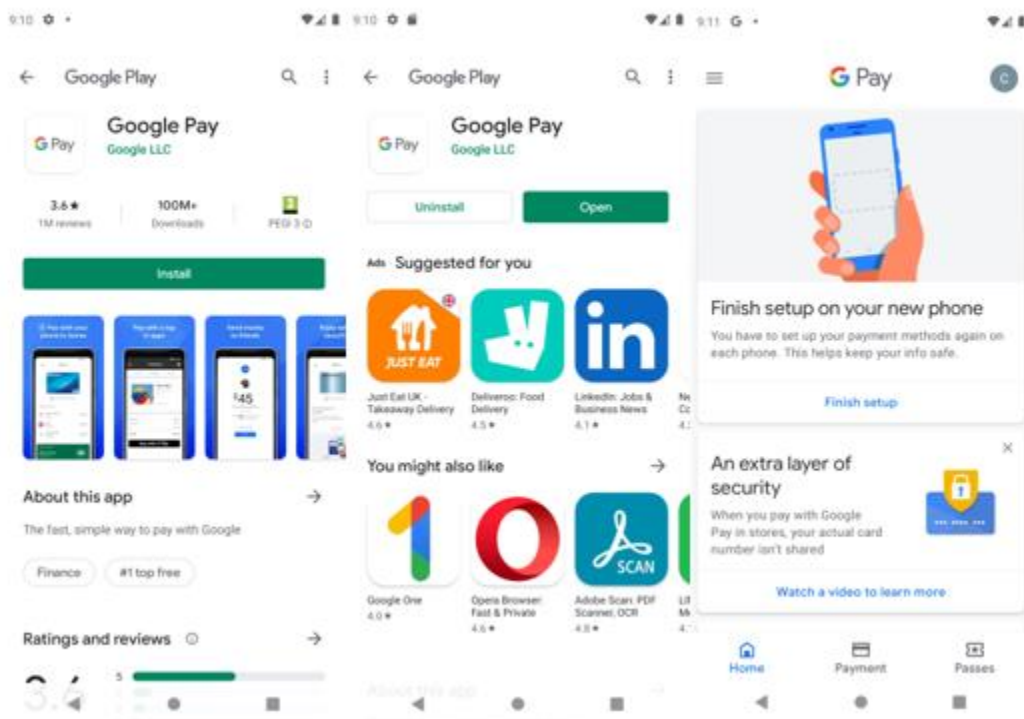


Figure 6-Google Pay installation process

The Wallet fragment helps the User to add existing flight tickets and gift cards to the database by simply pressing the + rounded button placed on the bottom right of the fragment. By pressing the + button a small adding menu will pop up. The pop up buttons will let the User to upload any pdf files. The small pdf pictured button will extend a Upload pdf page with a title fields. Once the title is entered and the Upload button is pressed, the phone will open the stored file.

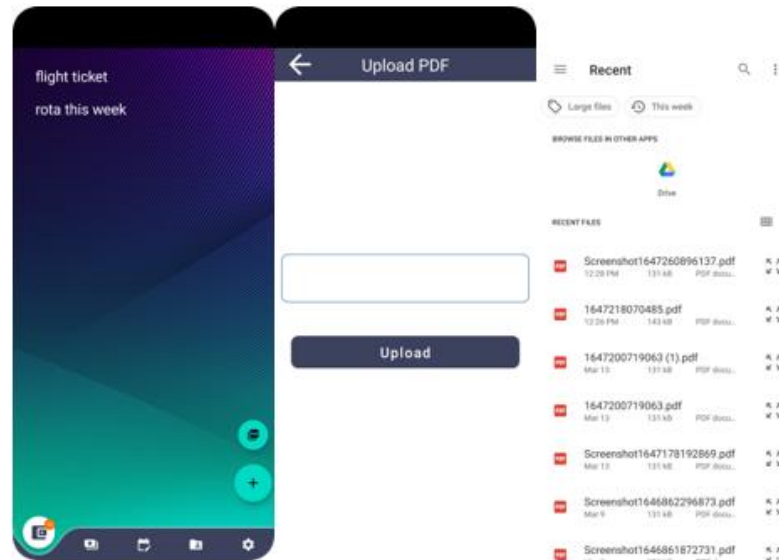


Figure 7-Pdf files upload and display feature

The Calendar fragment offers the function of adding future schedules by Pressing the + rounded button . The schedule page contains text fields that will store data such as date, time and event. All the schedules will be listed on the main calendar fragment as pe fololowing image.



Figure 8- Calendar Fragment

The Business card filed is one of the main features of the app which will help the User to store , edit it's own business card and also to share it via other communication system apps such as Facebook Messangers, Whatsap, Gmail etc. The Prin button will automatically create a .pdf file that will contain a screenshot of card's details. Once the .pdf file is created , it can be shared via .pdf file reade apps. The edit button will launch a pop up page with the user's details in order to change it for further updates.

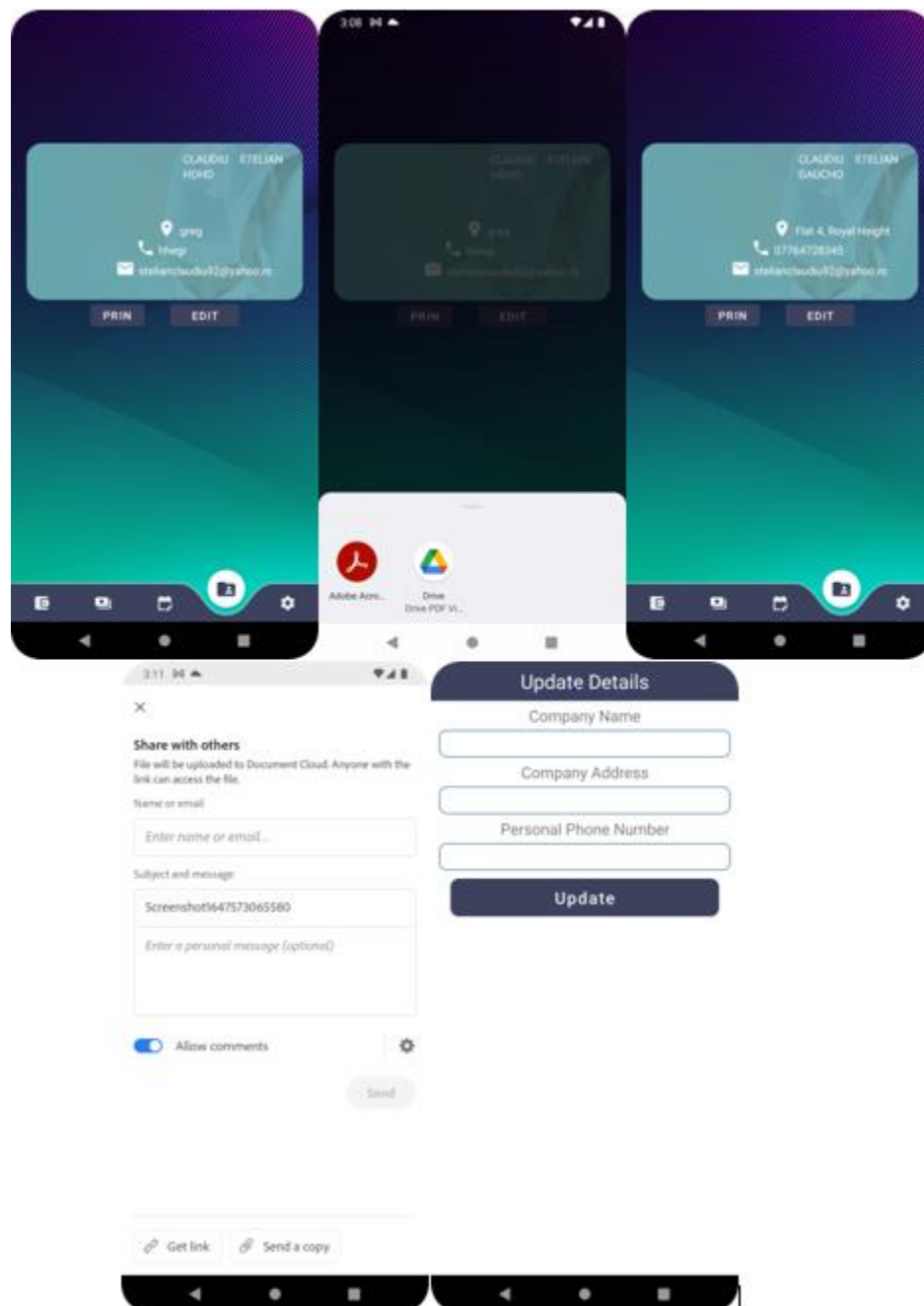


Figure 9- Business card Fragment

Settings page offer features that will help the User to change the password , get in touch with the development team and also logout. The Edit Password will open up a password reset where the User will have to fill in the text fields with the new password and the confirmation password. The old password will be changed straight after the Submit button is pressed . Once the Logout button is pressed the user will be redirected to the Login page.

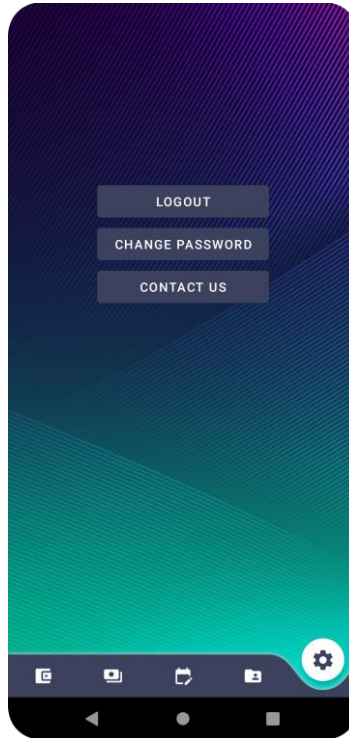


Figure 10 Settings fragment

4.2 DATABASE DESIGN

Database is based on Firebase Realtime Database and Firebase Authentication. Smartwallet's database is formed from a single class called Users that collect each of the user's ID's. Each user's ID can store details such as firstName, lastName, password, companyAddress, companyName, companyPhone, email and 2 other classes that are events and pdffiles. Events field gives the option to the user to store events details such as date, hour, event into an eventID. The pdffiles field can store each of the pdf files ID into the pdffileID. The pdffileID will also store data such as pdfnote and pdfurl.

Details such as firstName, lastName, password, companyAddress, companyName, companyPhone and email will be used into the business card's details as they will be printed on the business card fragment.

Details such as date, event, hour will be used to store the user's schedule details for future notice.

Pdf files will allow the user to store important data such as flight tickets or gift cards or other important documents which they are formed from a pdfnote and a pdfurl.



Figure 11 Database Structure

5 Implementation

5.1 MAIN MODULE EXPLANATION

The layout of all application's pages is using .xml scripts. The android studio schema has been imported to create the relevant tag: EditText, TextView, and MaterialButtons. Using the tags specified prior, the android: function gives the ability to set up the background, the visibility, the margin, etc.

This represents the front-end design of the application.

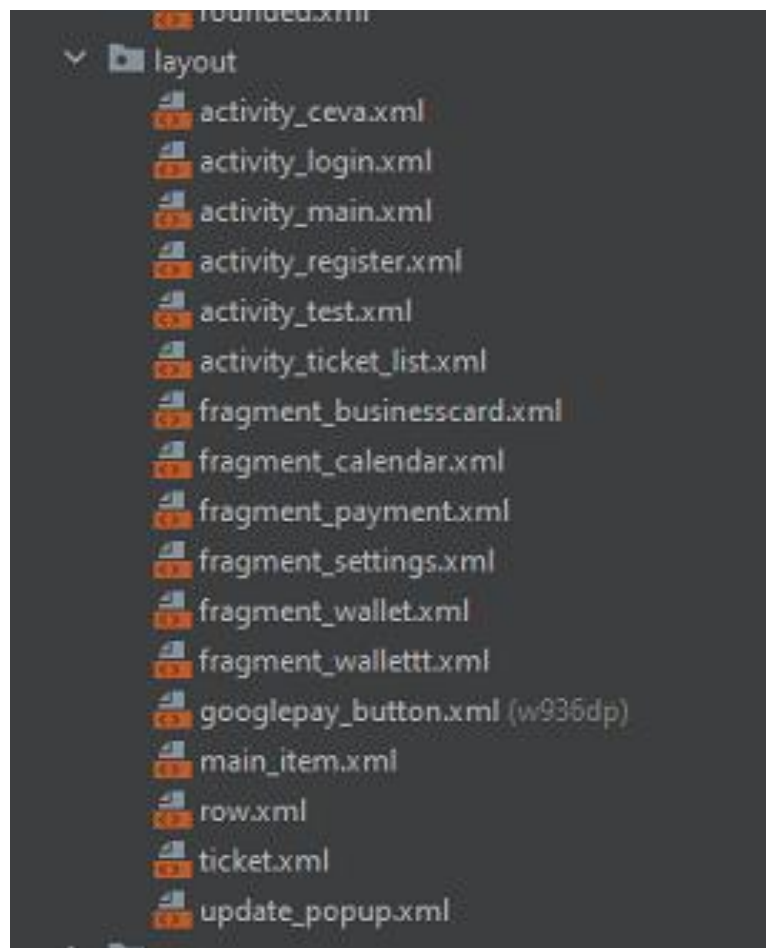


Figure 12 - Layout xml scripts

Below we have an example for the Register Page, where the First Name text box is positioned and coloured.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/ConstraintLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/background"
    android:visibility="visible"
    tools:context=".Register">

    <TextView...>

    <EditText
        android:id="@+id/firstname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/createaccount"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="10dp"
        android:background="#30ffffff"
        android:drawableStart="@drawable/ic_baseline_person_24"
        android:drawablePadding="20dp"
        android:hint="First name"
        android:padding="20dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />
```

Figure 13 - Layout Script for Register Page



Figure 14 - Register Page Display

The backend scripts are coded in Java.

Multiple Android libraries are imported

```
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
```

Figure 15 - Java libraries

Register.java is the backend script, of the activity_register.xml script.

- Checking the integrity of the inserted data using if conditions. These conditions can check empty fields, patterns matcher, passwords length and password similarity. In case of error the specific field will print an error using setError command.

```
if (TextUtils.isEmpty(lastName)){
    mLastname.setError("Last name is Required!");
    mLastname.requestFocus();
    return;
}
if (TextUtils.isEmpty(email)){
    mEmail.setError("Email is Required!");
    mEmail.requestFocus();
    return;
}
if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
    mEmail.setError("Please provide valid email");
    mEmail.requestFocus();
    return;
}
if (TextUtils.isEmpty(password)){
    mPassword.setError("Password is Required!");
    mPassword.requestFocus();
    return;
}
if (TextUtils.isEmpty(repassword)){
    mRePassword.setError("Please repeat the Password!");
    mRePassword.requestFocus();
    return;
}
if (password.length() < 6) {
    mPassword.setError("The Password must contain more than 6 characters!");
    mPassword.requestFocus();
    return;
}
if (!password.equals(repassword)){
    mRePassword.setError("Please repeat the Password!");
    mRePassword.requestFocus();
    return;
}
```

Figure 16 - Register code in Java

- Importing User classes that helps to store the user's details as per database.

```
public class User {

    public String firstName, lastName, email, password;

    public User(){

    }

    public User(String firstName, String lastName, String email, String password){
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
    }

}
```

Figure 17 - User Details with database connection

- Checking if the inserted email using task functions. If the email is not taken by another user all the inserted data will be stored into the Realtime Database under new UserID by displaying “User has been registered successfully!” toast otherwise the user will receive a toast with the actual error.

```

User user = new User(firstName, lastName, email, password);

FirebaseDatabase.getInstance().getReference( path: "Users")
    .child( FirebaseAuth.getInstance().getCurrentUser().getUid() )
    .setValue( user ).addOnCompleteListener( new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()){
                Toast.makeText( context: Register.this, text: "User has been registered successfully!", Toast.LENGTH_LONG ).show();
                mProgressBar.setVisibility( View.GONE );

                startActivity(new Intent(getApplicationContext(), Login.class));
                mProgressBar.setVisibility(View.INVISIBLE);

                //redirect to Login Layout!
            }else {
                Toast.makeText( context: Register.this, text: "Failed to register! Try again!", Toast.LENGTH_LONG ).show();
                mProgressBar.setVisibility( View.GONE );
            }
        }
    });
}

}else {
    Toast.makeText( context: Register.this, text: "Failed to register!" + task.getException().getMessage(), Toast.LENGTH_LONG ).show();
    mProgressBar.setVisibility( View.GONE );
}
}

```

Figure 18 - Checking an existing Account

- Redirecting the User to the Login page for account verification by using startActivity function.

```

startActivity(new Intent(getApplicationContext(), Login.class));
mProgressBar.setVisibility(View.INVISIBLE);

```

Figure 19 - Login redirect for account verification

1. Login Page

Login page is formed from one Login Button and two EditText fields that will ask to insert the user’s email and password details. Each of the field has an individual id for a storing the text input. All the inserted detail will be taken over by the Login.xml class which will have the following functions:

- To check if the EditText fields are empty and if the length of the password is not smaller than 6 characters otherwise an error will be displayed.

```
if (TextUtils.isEmpty(email)){
    mEmail.setError("Email is Required.");
    return;
}

if (TextUtils.isEmpty(password)){
    mPassword.setError("Password is Required.");
    return;
}

if (password.length() < 6) {
    mPassword.setError("The Password must contain more than 6 characters.");
    return;
}
```

Figure 20 - Password check

- To send a verification email using sendEmailVerification function otherwise the registered email is incorrect.

```
user.sendEmailVerification();
Toast.makeText( context: Login.this, text: "Check your email to verify your account!", Toast.LENGTH_LONG ).show();
progressBar.setVisibility(View.INVISIBLE);
```

Figure 21 - Verification e-mail

- When the Login button is pressed an email and password verification take place using Firebase Authentication system. Once the email verification is confirmed the user can successfully access the MainActivity of the application by being redirected to a Payment Fragment. To have a successful firebase database authentication an email verification is required and the java

```
if(task.isSuccessful()){
    FirebaseUser user= FirebaseAuth.getInstance().getCurrentUser();

    if (user.isEmailVerified()){
        startActivity( new Intent( packageContext: Login.this, MainActivity.class) );
    }
}
```

Figure 22 - Firebase email verification in Java

2. Main Activity

MainActivity page is divided in five different fragment activities and a bar navigation. Each of the fragments can be accessed individual as each of them contain different activities. All the fragments will be displayed into the MainActivity, and they can be accessed by using the bar navigation.

- The menu items are added with MeowBottomNavigation.
MeowBottomNavigation is using pictures to populate the items of the bar navigation.

```
bottomNavigation.add(new MeowBottomNavigation.Model( id: 1,R.drawable.ic_wallet));  
bottomNavigation.add(new MeowBottomNavigation.Model( id: 2,R.drawable.ic_payment));  
bottomNavigation.add(new MeowBottomNavigation.Model( id: 3,R.drawable.ic_calendar));  
bottomNavigation.add(new MeowBottomNavigation.Model( id: 4,R.drawable.ic_businesscard));  
bottomNavigation.add(new MeowBottomNavigation.Model( id: 5,R.drawable.ic_settings));
```

Figure 23 - Navigation Bar inserted with MeowBottomNavigation

- Each of the picture from the navigation bar has assigned a fragment that will be displayed on click. Each fragment will contain different activities.

```
bottomNavigation.setOnShowListener(new MeowBottomNavigation.ShowListener() {  
    @Override  
    public void onShowItem(MeowBottomNavigation.Model item) {  
        //Initialize fragment  
        Fragment fragment = null;  
        //Check condition  
        switch (item.getId()){  
            case 1:  
                //When id is 1  
                //Initialize notification fragment  
                fragment = new WalletFragment();  
                break;  
            case 2:  
                //When id is 2  
                //Initialize payment fragment  
                fragment = new PaymentFragment();  
                break;  
            case 3:  
                //When id is 3  
                //Initialize calendar fragment  
                fragment= new CalendarFragment();  
                break;  
            case 4:  
                //When id is 4  
                //Initialize business fragment  
                fragment= new BusinesscardFragment();  
                break;  
            case 5:  
                //When id is 5  
                //Initialize business fragment  
                fragment= new SettingsFragment();  
                break;  
        }  
        //Load Fragment  
        loadFragment(fragment);  
    }  
});
```

Figure 24 - Fragment activities

3. Wallet fragment

Wallet fragment is a frame that is displayed into the main activity. It is also functioning as activity. In this chase the walletfragment has the function of displaying details of .pdf files from the Firebase Realtime database and forward the user to add pdf activity.

- To fetch the data of the current user a user variable has been declared to store the user's data from database which can be seen in the picture below.

```
user = FirebaseAuth.getInstance().getCurrentUser();
```

Figure 25 - Fetch current user data

- For fetching the pdffiles data from firebase a child function must be executed. This will store all the current user pdf files and pdf names which later will be used to be displayed into a ListView frame.

```
user = FirebaseAuth.getInstance().getCurrentUser();
String uid = user.getId();

databaseReference = FirebaseDatabase.getInstance().getReference().child( "Users" ).child( uid ).child( "pdffiles" );
```

Figure 26 - Get pdf files for current user

- To list the pdf files names (titles) a layout function must be executed and also implemented into ListView frame

```
Object mActivity = getActivity();

if (mActivity != null) {
    ArrayAdapter<String> adapter = new ArrayAdapter<>( getActivity().getApplicationContext(),
        android.R.layout.simple_list_item_1, Uploads ) {
        @NonNull
        @Override
        public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
            View view = super.getView( position, convertView, parent );
            TextView text = (TextView) view.findViewById( android.R.id.text1 );
            text.setTextColor( Color.WHITE );
            text.setTextSize( 22 );

            return view;
        }
    };

    listView.setAdapter( adapter );
}
```

Figure 27 - ListView Frame

- To add pdf files into the database, the walletfragment has the add function that can be used by pressing the x button and then the pdf picture button. This will redirect the user to a new activity called upload_pdf.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View v = inflater.inflate( R.layout.fragment_wallet, container, attachToRoot: false );

    FloatingActionButton add_flight = v.findViewById( R.id.add_pdf );
    add_flight.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new Intent( getActivity(), upload_pdf.class );
            startActivity( i );
        }
    } );

    return v;
}
```

Figure 28 - Button for Upload Pdf

4. Upload_pdf page

Upload_pdf activity has the functionality of storing the user's new pdf files into its database with a wished titles for easy accessibility.

- As before a user variable must be declared as a guid for the function where the new pdf file will be stored.

```
//Database
user = FirebaseAuth.getInstance().getCurrentUser();
storageReference= FirebaseStorage.getInstance().getReference();
String uid = user.getUid();
databaseReference= FirebaseDatabase.getInstance().getReference().child("Users").child(uid).child( "pdffiles" );
```

Figure 29 - Upload Pdf

- To access the pdf file from user's device a selectfile function needs to be created where an inbuild action with "Select PDF files..." message will be executed.

```
private void selectFiles() {
    Intent intent = new Intent();
    intent.setType( "application/pdf" );
    intent.setAction(Intent.ACTION_GET_CONTENT );
    startActivityForResult( Intent.createChooser( intent, "Select PDF Files..." ), requestCode: 1 );
}
```

Figure 30 - Class for selected pdf files

- When uploading the file, a URL link will be created for a future reference. This URL will contain all the pdf document details. The pdf file will be stored where the pdf files were referenced by creating a unique id as reference.

```
private void UploadFiles(Uri data) {

    final ProgressDialog progressDialog= new ProgressDialog( context: this );
    progressDialog.setTitle( "Uploading..." );
    progressDialog.show();

    StorageReference reference = storageReference.child( "Uploads/"+System.currentTimeMillis()+".pdf" );
    reference.putFile( data )
        .addOnSuccessListener( new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                Task<Uri>uriTask= taskSnapshot.getStorage().getDownloadUrl();
                while (!uriTask.isComplete());
                Uri url = uriTask.getResult();

                pdfClass pdfClass= new pdfClass(pdf_name.getText().toString(),url.toString());
                databaseReference.child( databaseReference.push().getKey() ).setValue( pdfClass );

                Toast.makeText( context: upload_pdf.this, text: "File Uploaded!", Toast.LENGTH_SHORT ).show();

                progressDialog.dismiss();
            }
        } ).addOnProgressListener( new OnProgressListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onProgress(@NonNull UploadTask.TaskSnapshot snapshot) {

                double progress =(100.0* snapshot.getBytesTransferred())/ snapshot.getBytesTransferred();
                progressDialog.setMessage( "Uploaded" +(int)progress+"%");
            }
        } );
}
```

Figure 31 - Creating link for pdf files

- Before uploading the file into the database, a function will have to check the pdf file existence will be executed otherwise the user will not be able to move further.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult( requestCode, resultCode, data );

    if (requestCode==1 && resultCode==RESULT_OK && data!=null && data.getData()!=null){

        UploadFiles(data.getData());
    }
}
```

Figure 32 - Check the pdf file existence

5. Payment fragment

As the name says the payment fragment will help the user to make payments using google pay application.

- The payment will be accessed from GooglePayButton by being allocated to a variable ImageView type. The variable will have the function of redirecting the user to its own android store in the selected Google Pay field to download it and in case the Google Pay application is installed the user will have to press on open button

```
public class PaymentFragment extends Fragment {
    View view;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.fragment_payment, container, attachToRoot: false);
        // Inflate the layout for this fragment
        ImageView payy = view.findViewById( R.id.GooglePayButton );
        payy.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Uri webpage = Uri.parse( "https://play.google.com/store/apps/details?id=com.google.android.apps.walletnfcrel" );
                Intent Website = new Intent( Intent.ACTION_VIEW, webpage );
                startActivity( Website );
            }
        });
        return view;
    }
}
```

Figure 33 - Payment Fragment class

6. Business fragment

Business fragments retrieve the data from user's details database and display them into a CardView layout. For sharing the business card, the application will take a screenshot to the CardView and convert it into a .pdf file for future sharing. Another functionality is to add and edit the user's data by simply pressing the edit button.

- For displaying the user's details a set of TextView must be declared to store the user's details. A TextView variable is assigned to each of the user's detail for the data to be displayed on which are called hooks.

```
firstname = rootView.findViewById( R.id.firstnameC );
lastname = rootView.findViewById( R.id.lastnameC );
email = rootView.findViewById( R.id.emailC );
companyName = rootView.findViewById( R.id.companyName );
companyAddress = rootView.findViewById( R.id.companyAddress );
companyPhone = rootView.findViewById( R.id.companyPhone );
```

Figure 34 - Text field variables

- A method used for Business Card user details is implemented. The hooks are used to get the user's details that have been fetched from user's Firebase database.

```
private void showAllUserData(String firstNameValue,
                             String lastNameValue,
                             String companyNameValue,
                             String companyAddressValue,
                             String phoneNumberValue,
                             String emailValue
) {

    firstname.setText(firstNameValue);
    lastname.setText(lastNameValue);
    email.setText(emailValue);
    companyName.setText(companyNameValue);
    companyAddress.setText(companyAddressValue);
    companyPhone.setText(phoneNumberValue);

}
```

Figure 35 - Fetching user's Business Card details

- User's data must be fetched by using database reference. The reference will fetch the current user's database details. Once the data is extracted a String variable will get each of the user details. The String variables later are assigned to the TextView fields by executing the ShowAllUserData.

```
public String getUserData(String callback) {
    FirebaseAuth user = FirebaseAuth.getInstance().getCurrentUser();
    if (user != null) {
        DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
        String uid = user.getUid();
        mDatabase.child("Users").child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (!task.isSuccessful()) {
                    Log.e( tag: "firebase", msg: "Error getting data", task.getException());
                }
                else {
                    Log.d( tag: "firebase", String.valueOf(task.getResult().getValue()));
                    String firstName = task.getResult().child( "firstName" ).getValue(String.class);
                    String lastName = task.getResult().child( "lastName" ).getValue(String.class);
                    String companyName = task.getResult().child( "companyName" ).getValue(String.class);
                    String companyAddress = task.getResult().child( "companyAddress" ).getValue(String.class);
                    String companyNumber = task.getResult().child( "companyPhone" ).getValue(String.class);
                    String email = task.getResult().child( "email" ).getValue(String.class);

                    if (callback == "showAllUserData") {
                        showAllUserData(firstName, lastName, companyName, companyAddress, companyNumber, email);
                    } else if (callback == "prefillFields") {
                        prefillFields(firstName, lastName, companyName, companyAddress, companyNumber, email);
                    }
                }
            }
        });
    }
}
```

Figure 36 - Business User Details

- openPdf method will help to access the created pdf file and open it into a pdf file reader that can be selected by the user.

```
private void openPdf(String path) {
    File file = new File(path);
    Intent target = new Intent(Intent.ACTION_VIEW);
    target.setDataAndType( Uri.fromFile(file), type: "application/pdf");
    target.setFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);

    Intent intent = Intent.createChooser(target, title: "Open File");
    try{
        startActivity(intent);
    }catch (ActivityNotFoundException e){
        Toast.makeText(getActivity(), text: "No Apps to read PDF File", Toast.LENGTH_SHORT).show();
    }
}
```

Figure 37 - Open Pdf class

- CreatePdf method will convert the screenshot into a pdf file. Once the pdf file is formed it will be automatically opened into an PDF file reader.

```
private void createPdf() {  
  
    PdfDocument document = new PdfDocument();  
    PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(bitmap.getWidth(), bitmap.getHeight(), pageNumber: 1).create();  
    PdfDocument.Page page = document.startPage(pageInfo);  
  
    Canvas canvas = page.getCanvas();  
  
    Paint paint = new Paint();  
    paint.setColor(Color.parseColor("#ffffff"));  
    canvas.drawPaint(paint);  
  
    Bitmap bitmap = Bitmap.createScaledBitmap(this.bitmap, this.bitmap.getWidth(), this.bitmap.getHeight(), filter: true);  
  
    paint.setColor(Color.BLUE);  
    canvas.drawBitmap(bitmap, left: 0, top: 0, paint: null);  
    document.finishPage(page);  
    File filePath = new File(path);  
    try{  
        document.writeTo(new FileOutputStream(filePath));  
    }catch (IOException e){  
        e.printStackTrace();  
        Toast.makeText(getActivity(), "Something Wrong: "+e.toString(), Toast.LENGTH_SHORT).show();  
    }  
  
    document.close();  
  
    if (myPath.exists())  
        myPath.delete();  
  
    openPdf(path);  
}
```

Figure 38 - Method to Create Pdf

- The screenshot method will use internal android features and will convert it into a pdf file by using the createPdf () method. The pdf file will also be stored into a Download directory

```
void takeScreenshot() {  
    String pdfPath = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS).toString();  
    File folder = new File(pdfPath + "/ScreenShot/");  
  
    if(!folder.exists()){  
        boolean success = folder.mkdir();  
    }  
    path = folder.getAbsolutePath();  
    path = path + "/" + file_name + System.currentTimeMillis() + ".pdf";  
  
    View u = getView().findViewById(R.id.businessCard);  
  
    CardView z = getView().findViewById(R.id.businessCard);  
    totalHeight = z.getChildAt(0).getHeight();  
    totalWidth = z.getChildAt(0).getWidth();  
  
    String extr = Environment.getExternalStorageDirectory() + "/Flight Ticket/";  
    File file = new File(extr);  
    if(!file.exists())  
        file.mkdir();  
    String fileName = file_name + ".jpg";  
    myPath = new File(extr, fileName);  
    imagesUri = myPath.getPath();  
    bitmap = getBitmapFromView(u, totalHeight, totalWidth);  
    try{  
        FileOutputStream fos = new FileOutputStream(myPath);  
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);  
        fos.flush();  
        fos.close();  
    }catch (Exception e){  
        e.printStackTrace();  
    }  
    createPdf();  
}
```

Figure 39 - Method for screenshot

- The edit functionality is designed to launch a popup dialog with update_popup layout. The update_layout is formed from a variety of EditText fields where to each of them have been declared a variable that can hold the data from the firebase database.

Figure 40 - Method for Popup Window

```
@Override
public void onCreateView(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onCreateView( view, savedInstanceState );
    StrictMode.VmPolicy.Builder builder = new StrictMode.VmPolicy.Builder(); StrictMode.setVmPolicy(builder.build());

    btnPrint = getView().findViewById(R.id.btn_print);
    btnEdit = getView().findViewById( R.id.btn_edit );

    WindowManager wm = (WindowManager) getActivity().getSystemService(Context.WINDOW_SERVICE);
    mDisplay = wm.getDefaultDisplay();

    btnEdit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final DialogPlus dialogPlus = DialogPlus.newDialog(getContext())
                .setContentView( new ViewHolder( R.layout.update_popup ) )
                .setExpanded( expanded: true, defaultContentHeight: 1560 )
                .create();
            dialogPlus.show();

            View view = dialogPlus.getHolderView();

            getUserData( callback: "prefillFields" );

            Button btnUpdate = (Button) dialogPlus.findViewById( R.id.btnUpdate );//dialog instead of view
            btnUpdate.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) { Log.d( tag: "bnUpdate", msg: "clicked"); }
            });
        }
    });
}
```

1. Settings fragment

This fragment has 3 functions that are triggered by 3 different buttons.

```
Button btnchangepassword;
Button contactus;
Button logout;
```

Figure 41 - Buttons in the Settings Fragment

- Logout button is executing a method that is implemented into the MainActivity.class. this method is called logout and once is executed the user will be redirected back to the Login page and disconnect the user from the Firebase authentication. This method is assigned into Settings fragment to the Logout button as a onClick command.

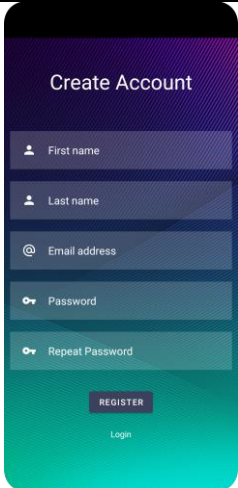
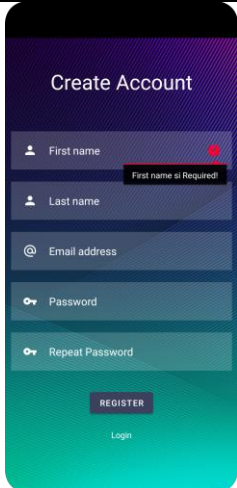
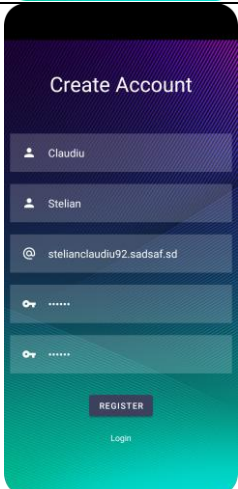
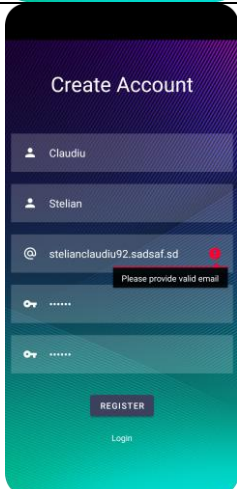
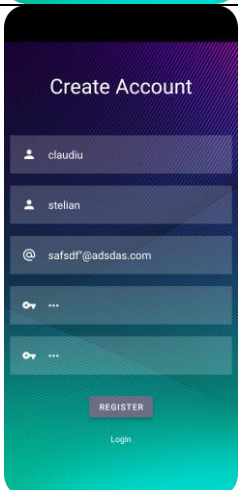
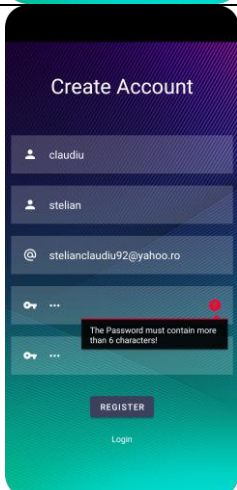
```
public void logout (View view){  
    FirebaseAuth.getInstance().signOut();//logout  
    startActivity(new Intent(getApplicationContext(),Login.class));  
    finish();  
}
```

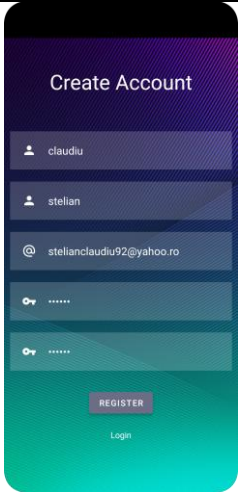
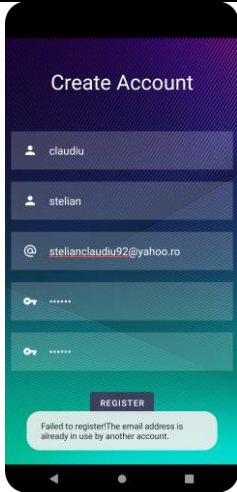
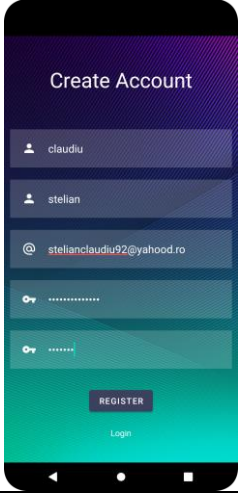
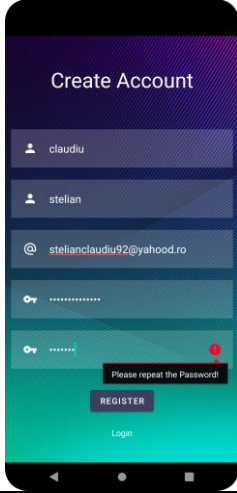
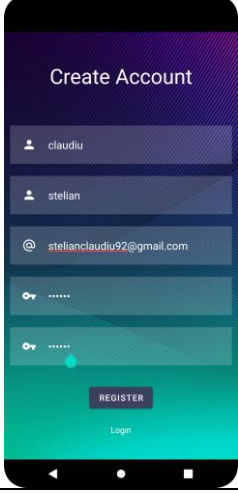
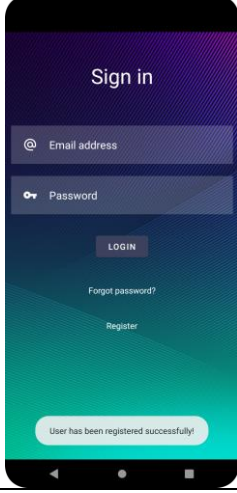
Figure 42 - Sing out listener

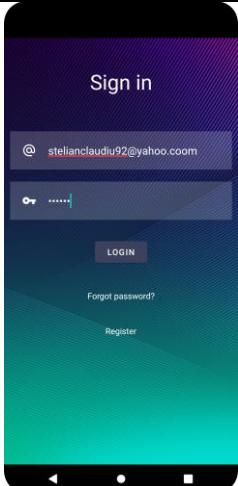
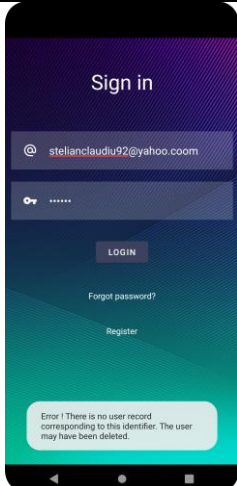
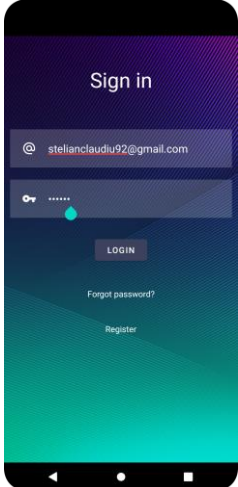
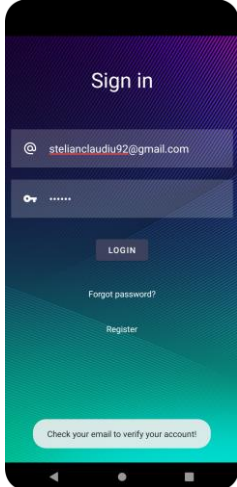
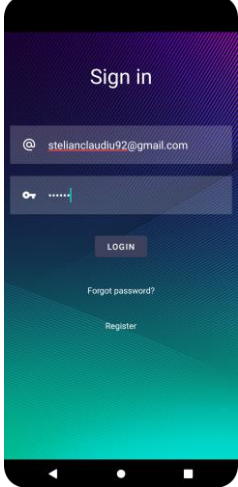
```
<Button  
    android:id="@+id/LOGOUT"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="200dp"  
    android:layout_marginLeft="100dp"  
    android:layout_marginRight="100dp"  
    android:onClick="logout"  
    android:text="logout" />
```

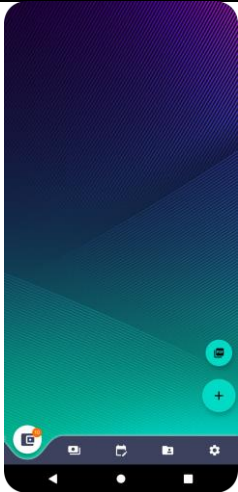
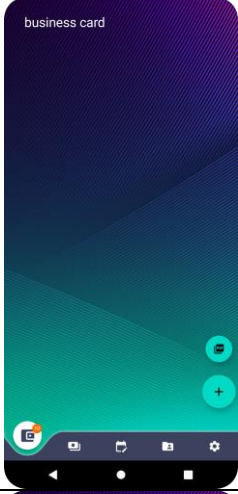
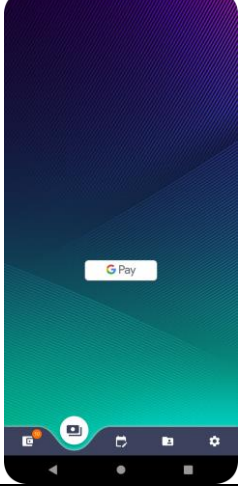
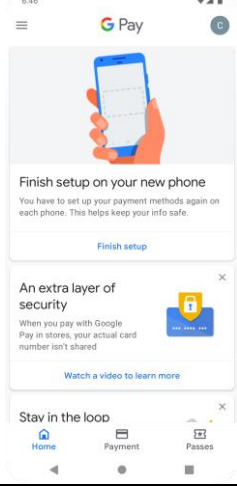
Figure 43 - Sign-out onClick Allocation to logout button

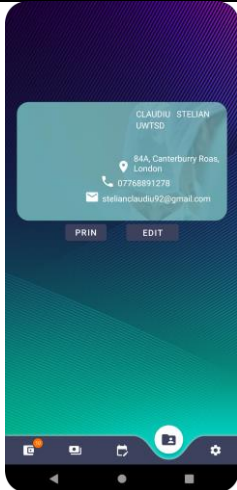
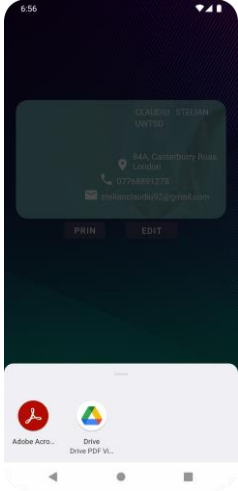
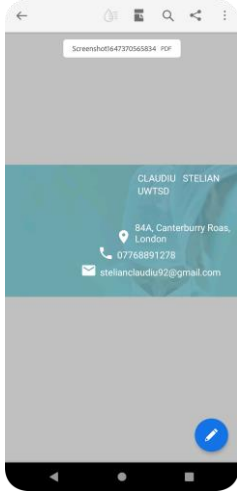
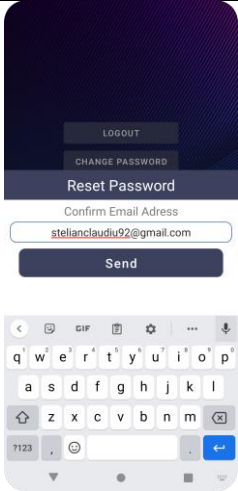
6 Testing

Purpose of testcase	Input	Output	Result
Register without inserting data			Pass
Register with wrong email address pattern			Pass
Register with a password smaller than 6 characters			Pass

Register with an existing registered email account			Pass
Register by inserting two different passwords			Pass
Register by inputting the correct details			Pass

Login with an unregistered email address			Pass
Login without email confirmation			Pass
Login once the registration email is confirmed			Pass

Upload Pdf file into Firebase database			Pass
Download the Pdf file from Firebase database			Pass
Make payments using Google Pay third- party application			Pass

Create a business card with the user's details			Pass
Print a business card			Pass
Change password			Pass

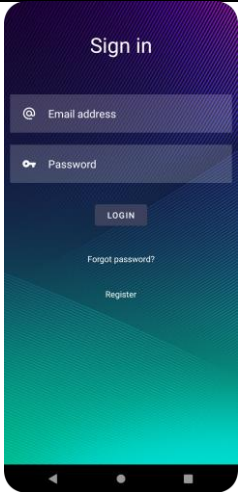
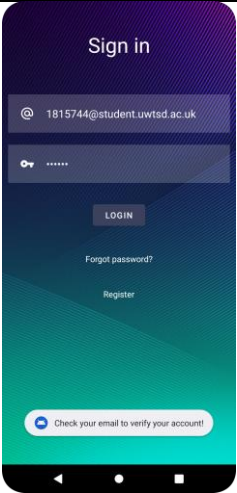
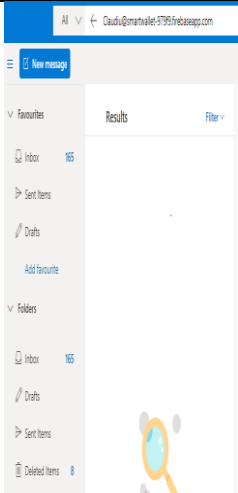
Logout			Pass
Login using Outlook student email			Pass (Firebase doesn't support outlook student email and the user will be able to Login by waiting 30 seconds or using another email services provider)

Table 2 - Application Features Testing table

7 Evaluation

7.1 PERSONAL EVALATION

Nr		Strongly Disagree	Disagree	Agree	Strongly Agree
1	I gain knowledge about the existing digital wallets			x	
2	I improved by programming skills				x
3	Now I have a strong base about mobile development			x	
4	I learned so much about project and report structure			x	
5	I learned to work agile			x	
6	I consider my project useful				x
7	I consider digital wallets convenient and secure			x	
8	I consider my research useful			x	
9	I learned more about project planning			x	
10	I know how to estimate the budget of a project		x		
11	I think there is one digital wallet application that is better developed				x

Table 3 - Personal Evaluation

7.2 PROJECT EVALUATION

Nr		Strongly Disagree	Disagree	Agree	Strongly Agree
1	The application is ready to be used			x	
2	The application is scalable and performant			x	
3	The features are useful			x	
4	The application has a user-friendly UI				x
5	The application can be used from everywhere				x
6	The application can be improved				x
7	The tools used to develop de application are reliable and fitted for the requirement				x

Table 4 - Project Evaluation

8 Further Enhancement

Administrator functionalities		
Feature	Description	Example
Filter users	The app will have the search function for easy data management	The admin will simply select the user by searching its name through app's data base.
Select user	The app will have the user select function from the app's data base	The admin will have the capability to go through the app's database by simply scrolling through user's name field.
Access and edit user's cards	Display the details of the selected user to edit or just check it.	If one user wouldn't be able to access his data, the admin will be able to go through that specific user details and change it.
Access and edit user's calendar	Display and edit all the selected user's calendar's schedule.	The admin will be able to change or delete the user's calendar schedules if the user cannot.
Access and edit user's Business card	Display and edit selected user's business card.	If the user cannot change the details of its business card the admin will be able to access and change it.
Access and edit user's personal details	Display and edit selected user's personal details.	If the user cannot change its personal details, the admin will have the ability of accessing and change it.

Table 5 – Administrator Further Enhancement

User's functionalities		
Feature	Description	Example
Storing multiple cards	Gives the opportunity to add important cards via QR code scan.	The user can add fidelity cards, gift cards into the app by scanning the QR code.
Contact details	Accessing the contacts tab, the user will be able to choose between adding a new phone number or search through existing phone's contacts	The user can simply Press ADD function that will redirect the user into the phone's contact details. The user can access the phones contact details by simply pressing on Search contacts.

Table 6 - User Further Enhancement

9 Conclusion

Online payment solutions, usually in the form of an app, are known as digital wallets. There is no need to enter the card data or carry a physical card to make payments because the wallet securely saves virtual versions of debit and credit cards. These types of application may allow to save digital tickets and e-vouchers, ensuring that a customer have all the documentation needed. There are several digital wallets on the market, but in this reach are examined three most known ones for different angles, and perspectives.

A new wallet application has been developed for this purpose. The features and functionalities are being tested, evaluated and discussed.

Mobile wallets are a convenient and secure method to replace the old ways of payment.

10 References

- [1] Navanwita Sachdev, "The evolution of ewallets: history, benefits and withdrawals" 2019 [Online] Available: <https://sociable.co/mobile/evolution-ewallets-history-benefits-withdrawals/#:~:text=The%20origin%20of%20digital%20payments,is%20believed%20as%20the%20origin>. [Accessed: 09-03-2022].
- [2] Matthew David, Amy Novotny, James Denman, "mobile application development" 2021 [Online] Available: <https://www.techtarget.com/searchapparchitecture/definition/mobile-application-development>. [Accessed: 09-03-2022].
- [3] Jerome DiMarzio, "Beginning Android Programming with Android Studio", October 24, 2016.
- [4] Android Studio, "Meet Android Studio", 2021, [Online] Available: <https://developer.android.com/studio/intro> [Accessed: 09-03-2022].
- [5] Amit Samdsukha, "The Challenges Of Developing eWallet Apps", 2021, [Online] Available: <https://www.emizentech.com/blog/challenges-of-developing-ewallet-apps.html> [Accessed: 11-03-2022].
- [6] Rimma Kats, " More than 100 million Americans will use proximity mobile payments ", 2021 <https://www.emarketer.com/content/more-than-100-million-americans-will-use-proximity-mobile-payments-2021> ,[Accessed: 11-03-2022].
- [7] [Online] <https://www.compareremit.com/money-transfer-guide/google-pay-review-pros-and-cons/>, "Google Pay Review: Pros And Cons", Aug 30, 2021, [Accessed: 11-03-2022].
- [8] Maggie Tillman, "What is Samsung Pay, how does it work, and which banks support it?", January 11, 2022, [Online] Available: <https://www.pocket-lint.com/apps/news/samsung/132981-what-is-samsung-pay-how-does-it-work-and-which-banks-support-it> ,[Accessed: 11-03-2022].
- [9] CRAIG LLOYD, "Six Apple Wallet Features You May Not Have Known About", AUG 9, 2017, [Online] Available: <https://www.howtogeek.com/320721/6-apple-wallet-features-you-may-not-have-known-about/> ,[Accessed: 11-03-2022].
- [13] Greg Sterling, "Beacon marketers have new ways around app download requirement", April 19, 2016, [Online] Available: <https://martech.org/beacon-market-matures-marketers-drop/>, [Accessed: 11-03-2022].
- [14] Caldwell, Serenity. "How to use rewards cards with Apple Pay and the Wallet app", April 16, 2018, [Online] Available: <https://www.imore.com/how-use-rewards-cards-apple-pay-and-wallet-app>, [Accessed: 11-03-2022].
- [17] [Online] www.press.bmwgroup.com, "BMW announces support for Digital Key for iPhone. A secure and easy way to use iPhone as a car key to lock, unlock, drive, and share keys with friends", May 18, 2021, [Accessed: 11-03-2022].
- [18] Adam Uzialko, "Mobile Wallet Guide: Google Pay vs. Apple Pay vs. Samsung Pay", Dec 03, 2021 , [Online] Available: <https://www.business.com/articles/google-pay-vs-apple-pay-vs-samsung-pay/>, [Accessed: 11-03-2022].
- [19] PYMNTS, "Apple Pay", August 10, 2021, [Online] Available: <https://securecdn.pymnts.com/wp-content/uploads/2021/09/PYMNTS-Apple-Pay-Turns-Seven-September-2021.pdf> [Accessed: 11-03-2022].
- [20] Doug Stevenson, "What is Firebase? The complete story, abridged.", Sep 24, 2018,

[Online] Available: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0> [Accessed: 11-03-2022].

11 Appendices

Digital wallet	Number of users	Pros	Cons
Apple Pay	43.9 million	85% of U.S. merchants accept it (according to Apple). 44% of U.S. consumers have an Apple Pay-enabled device. It has the fastest YOY growth in users: 24% from 2020 to 2021. Apple Pay users live in urban centres, are younger and earn more than users of other digital wallets. It accounts for nearly half of all in-store digital wallet purchases (45.5%).	Fewer than 2% of in-store purchases are made with Apple Pay.
Google Pay	25 million	It has a fast projected growth of 40% (2020 to 2025). You can add loyalty cards. Users can earn rewards. It's a good option for Android phone users.	It's much less used than Apple Pay (10.3% of digital wallet in-store purchases). It recently relaunched, requiring users to download a new app and reinstall. There are added fees for P2P. Its user base is growing more slowly than Apple's.
Samsung Pay	16.3 million	It offers a rewards program to users. It can be used with non-NFC credit card readers. You can use Samsung Money as a bank account and earn interest.	Its user growth is stagnant. You must have a Samsung device to use it. It has the smallest user base.

Table 7 - Pros and Cons of multiple eWallets applications