



ASHESI

ASHESI UNIVERSITY

CONTACTLESS PAYMENT FOR MOBILE MONEY IN GHANA

CAPSTONE PROJECT

B.Sc. Computer Engineering

Isaac Benjamin Clement-Owusu

2020

ASHESI UNIVERSITY

CONTACTLESS PAYMENT FOR MOBILE MONEY IN GHANA

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Isaac Clement-Owusu

2020

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:



Candidate's Name: Isaac Benjamin Clement-Owusu

Date: May 29, 2020

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:



Supervisor's Name:



Date:

Acknowledgments

I would like to thank my supervisor, Mr. Francis Gatsi, whose encouragement and academic advice helped me undertake this project.

Abstract

The use of Unstructured Supplementary Service Data (USSD) applications in conducting monetary transactions is an unsafe and slow means of payment. Research has shown that USSD applications utilize outdated encryption techniques, which make it susceptible to hackers during data transmission; they are unreliable and have long processes before payment is conducted. This study aims to explore how Near Field Communication (NFC) technology can be used to conduct payments. Comparing the two payment techniques, it asks: How does NFC payment compare USSD payment and what advantages does one present over the other. In this context, these payment methods are tested with Mobile Money, a mobile banking platform.

Based on a review of literature on NFC and USSD as technologies and payment techniques, a system was built to allow payments to be made via NFC technology. A survey was then conducted to collect the duration of the already existing USSD payment and the NFC payment. Analysis of the data collected showed that payment using NFC technology was substantially faster than that of the USSD. The results indicate the efficiency of NFC over USSD. On that basis, it is recommended that mobile money providers implement this form of payment. Further improvements must be made to the system to improve it in terms of security and user experience.

Contents

DECLARATION	i
Acknowledgments	ii
Abstract	iii
Chapter 1: Introduction	1
1.1 Background	1
1.2 Motivation	2
1.3 Problem Description	3
1.4 Project Overview	3
Chapter 2: Literature Review.....	5
Chapter 3: Design	13
3.1 System Design.....	13
3.1.1 System Flow.....	14
3.2 System Architecture	15
3.2.1 Software Architecture	15
3.2.2 Client-Server Architecture.....	17
3.3 Functional Requirements	17
3.4 System Users.....	18
3.4.1 Use Case Diagram.....	19
3.5 Non-Functional Requirements.....	20
3.6 Hardware Requirements	21
Chapter 4: Implementation	25
4.1 Overview.....	25
4.2 Backend Development	25
4.2.1 Programming Language – JavaScript ExpressJS	25
4.2.2 API Structure	26
4.3 Frontend Development.....	27
4.3.1 Administrator Portal	27
4.3.2 Mobile Application	30
4.4 Database Development	31
4.4.1 Mongo Database.....	31
4.5 Card Reader Programming	35

4.6 Screenshots	36
Chapter 5: Testing and Results	39
5.1 System Testing	39
5.1.1 Unit Tests	39
5.1.2 Feature Testing	41
5.1.3 User Testing.....	43
Chapter 6: Conclusions and Future Work	45
6.1 Conclusion	45
6.2 Challenges.....	46
6.3 Future Work.....	47
References.....	48
Appendix.....	50
Appendix A: Survey data.....	50

Chapter 1: Introduction

1.1 Background

Transactions between consumers and suppliers have seen growth from the days of barter (a system where commodities are exchanged) to the cash and the electronic money society we are in today. Mobile money was first introduced to the Ghanaian market in 2009 and since then has been one of the prime means of conducting transactions. In 2019, Ghana became the fastest growing mobile money market in Africa [1]. This growth is primarily because most people without bank accounts prefer to use mobile money to conduct most, if not all, of their transactions. The process of operations, a USSD application, makes it easy for the average Ghanaian to have access to funds directly and at their fingertips. Mobile banking has introduced the concept of a cashless economy to the Ghanaian market.

A cashless economy is an economic system where transactions are done electronically, through the use of credit cards, debit cards wallets, or other digital modes[2]. Countries such as China that produced the first recognizable coins are now conducting payment using facial recognition on their phones. The possibility of a cashless economy is made possible by the convergence of the telecommunication, banking, and retail industries. There are many advantages to living in a cashless economy, the elimination of middlemen, and a tool to fight corruption are just but a few benefits to this system[3]. A cashless economy is within reach, however for this economy to be productive, it requires security and efficiency, attributes that USSD applications lack, but contactless payment technology possesses.

Contactless payment is a secure method for consumers to purchase products or services using a debit card, credit, or smartcard – also known as chip card – by using Radio Frequency Identification (RFID) or near-field communication (NFC) [4]. Both NFC and RFID

technologies make use of radio waves to transfer data between the two devices. The difference between these two technologies is the range of the signals. RFID covers a distance ranging from inches to feet, while NFC covers a distance between millimeters and centimeters. Contactless payment with NFC is a seamless process where the customer makes payments by either tapping or getting their smartcard in proximity with the card reader. It is a fast and secure [5] medium and has seen significant implementation in visa cards and popularity in the UK since 2014 [6].

1.2 Motivation

Ghana plans to eliminate paper from most services and transactions to become the most digitalized economy in Africa. At the end of 2018, it was found that there were 456 million unique mobile phone subscribers in Africa; this number is projected to reach 623 million by the year 2025 [7]. Ghana already has different means of performing financial transactions electronically; these systems rely on the use of biometrics or USSD applications, such is the case with mobile money [8]. Implementing a different payment method allows consumers to have a more straightforward and secure way of conducting transactions while promoting the use of modern technology and a cashless economy. Using these systems allows the Ghanaian people to have diversity in their conducting transactions.

The current generation is more abreast of rising technology available due to global exposure. The contactless payment provides a new way of conducting transactions that will make it easier for people to not only adopt new technology but also to offer a modern, safer, and more secure form of using the mobile money service. Using this mode of transactions can allow users to use applications and services provided by Apple and Google through ApplePay and GoogleWallet.

1.3 Problem Description

Mobile money has increased the overall number of transactions that occur within the Ghanaian economy. This form of trade is common in retail shops, so are the use of systems such as EcobankPay [9], which makes use of a QR code and the mobile app, both of which are convenient ways of conducting transactions. A limitation of applications such as EcobankPay is the target market; these applications are limited to smartphone users even though they provide more secure payments relative to USSD applications. USSD applications provide access to both feature phone and smartphone users, where it makes up for in accessibility it lacks in security, thereby putting all users at risk of hacking. To improve the security of the mobile money platform, service providers have created a web-based system and a mobile application to accompany the USSD application.

Low security in USSD applications puts users at risk of being susceptible to breaches. When dealing with finances, it is paramount that information moving in and out of the system is secure to prevent interception and manipulation of data during transmission. If security measures are breached, it affects the integrity of the data – data integrity refers to the reliability and trustworthiness of data throughout its lifecycle [10]. In finance, sparse data integrity can lead to massive losses for both the service provider and users. Users that have their payments tampered with can lose a significant amount of money and will hold the service provider accountable for not ensuring the safety of the system.

1.4 Project Overview

This project aims to explore a more secure and fast means of conducting payments on the mobile money platform that will encompass both feature phone users and smartphone users. This system should possess features that allow the checking of balance, the ability to pay

merchants using a smartphone or smartcard, provide notifications, register users unto the system, get transaction history and connect with the mobile money API most importantly. This system should meet the requirements of contactless payment, which means it should be easy to use, secure, and fast to encourage the migration of users unto the platform.

Chapter 2: Literature Review

Communication in society has seen many evolutions due to rising technology; the introduction of the mobile handset has allowed people to communicate on a global scale. In 2007, the global number of mobile users reached 2.83 billion people, with 2.28 billion using the Global Service for Mobile Communication (GSM) [11]. GSM has been identified to have some security flaws. However, many operators in developing countries still use the traditional GSM network [11]. The most significant flaws of GSM include vulnerability to the man-in-the-middle attack, flaws in the implementation of A3/A8 algorithms, – a validation process used in a GSM network for authentication of the mobile user that is requesting service [12] – vulnerability to the Denial of Service (DoS) attack and absence of integrity protection [11]. All these security risks can be resolved using different measures, such as end-to-end encryption. The introduction of GSM has sprouted services such as Unstructured Supplementary Services Data (USSD), a real-time session-oriented technology used to provide mobile-based network and banking services with or without internet using USSD codes over a GSM channel [13]. USSD applications are interactive menu-driven applications that are cheaper, faster, and better than using SMS. USSD applications are platform-independent and require no software download. They are usually preferred when creating applications for feature phone users. Although USSD is a better alternative to SMS, it also possesses major security threats that can be exploited. USSD uses numeric codes with * prefixes and # suffixes. USSD works over GSM, the architecture for USSD is shown in figure 2.1.

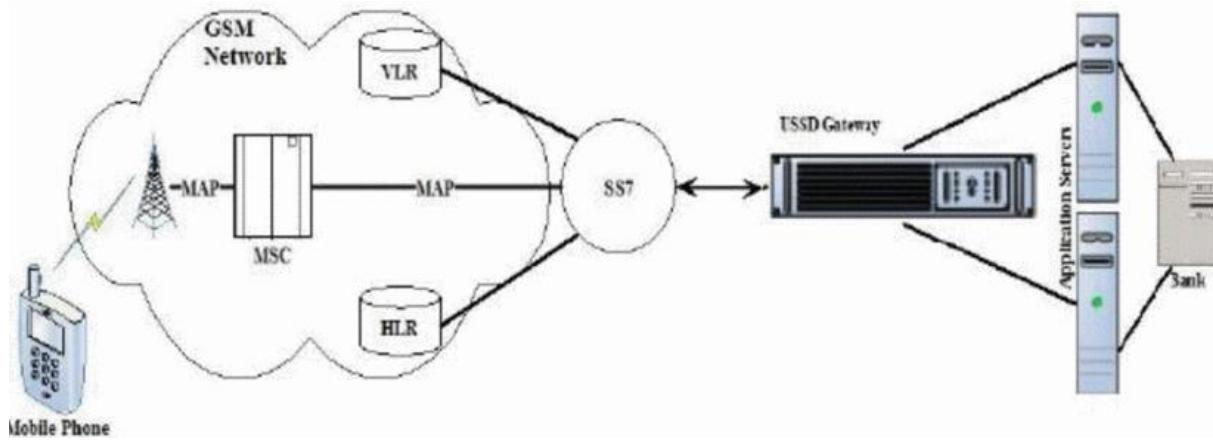


Figure 2.1: USSD Architecture for banking services

Source: “USSD — Architecture analysis, security threats, issues and enhancements”

From figure 2.1, the mobile device connects to the cell tower or base station of the service provider. It is forwarded to the mobile switching center (MSC) responsible for mobile authentication, routing, and call forwarding. The MSC forwards the data to the visitor location register (VLR), which contains dynamic details about subscriber information; the VLR forwards the data to the Home location register that has subscriber details before requesting the USSD gateway. USSD has some key benefits when it comes to storage, cost, implementation, and platform dependency. However, there are security threats to USSD that include the ability to tamper with request and response messages, the lack of end-to-end encryption, and the use of security algorithms such as the A5 algorithm that can be broken [13].

Wireless technology is gradually replacing wired technology. Most smartphones shipped out are equipped with different wireless technology. Smartphone users expect that their single device can be used to access a variety of services, including communication, entertainment, and commerce. This high demand has led to advancements in a variety of wireless technologies such as Near-Field Communication (NFC), which has many applications,

including contactless payment, commonly known as NFC Payments. These payment methods are being accepted by retailers in developed and developing countries, which have proven to be a convenient payment method for customers. The concept of NFC is derived from Radio Frequency Identification (RFID) and the concept of magnetic induction when two NFC-enabled devices are in proximity. RFID is a form of wireless communication that makes use of radio waves to track and identify objects. An RFID system consists of three components, an antenna, a transceiver, and a transponder or a tag. A signal is transmitted from the antenna to the transponder, which activates the transponder to respond. Any action with data triggers the programmable logic controller, which performs the task as simple as opening the door or as complex as performing a transaction. The reader emits a magnetic field generated by a small electric current flowing through a coil, the client device turns the field into electric pulses for communication of data, as shown in figure 2.2.

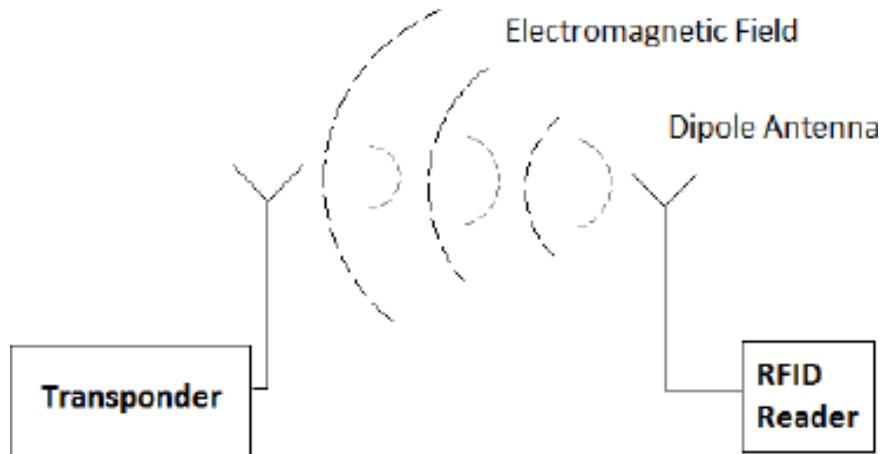


Figure 2.2: Magnetic induction diagram

Source: “NFC and NFC payments: A review”

In smartphones, the electric current going through the coil is generated by the battery sending current through the NFC chip built into the phone. NFC is a wireless short-range

communication technology that covers about ten centimeters and generates a radio frequency in 13.56MHz spectrum supporting data rates of 106 Kbps, 212 Kbps, and 424 Kbps. NFC has two different ways of communicating: two-way communication, which involves devices that are capable of reading and writing to each other, and one-way communication, where reading and writing is done by a powered device. NFC can exist in any of the three modes at a time; these modes are reader/writer modes, peer-to-peer mode, or card emulation mode. Card emulation mode the NFC enabled device acts as an NFC card placing the device in a passive communication mode suitable for payment. Like every wireless communication medium, there exist standards that govern the functionality of the technology. NFC has three primary standards namely ISO 14443 A/B, – a well-known standard was initially developed for contactless chip card communication over a 13.56 MHz radio – Sony ‘FeliCa’, created by Sony and commonly used in payment and transportation applications in Asia, and ISO/IEC 18092 that defines both the active and passive communication modes of Near Field Communication Interface and Protocol to realize a communication network using NFC devices. The NFC device architecture consists of two integrated circuits in a mobile device, namely Secure Element, and NFC interface. The NFC interface is composed of an NFC antenna, and an IC called the NFC controller to enable transactions via NFC, as seen in figure 2.3.

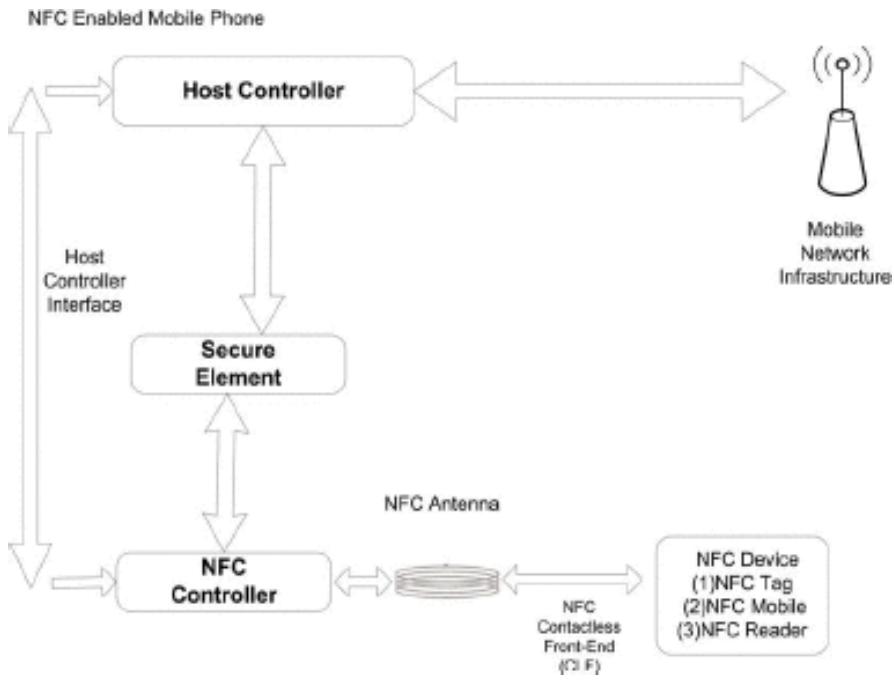


Figure 2.3: NFC device architecture

Source: “NFC and NFC payments: A review”

NFC utilizes the tap-to-pay technology where consumers can load their payment information into their NFC-enabled smartphones. In NFC payment, two points of data are associated with every transaction, RFID, and an encrypted password. The smartphone sends a radio signal with a unique code to the retailer's payment system, which sends the user transaction details. The user enters a PIN to approve the specific transaction. When conducting a payment, the NFC radio is restricted to one app on the phone and isolated from the operating system to protect against viruses and hackers [14]. Figure 2.4 explains the flow of how NFC payments occur.

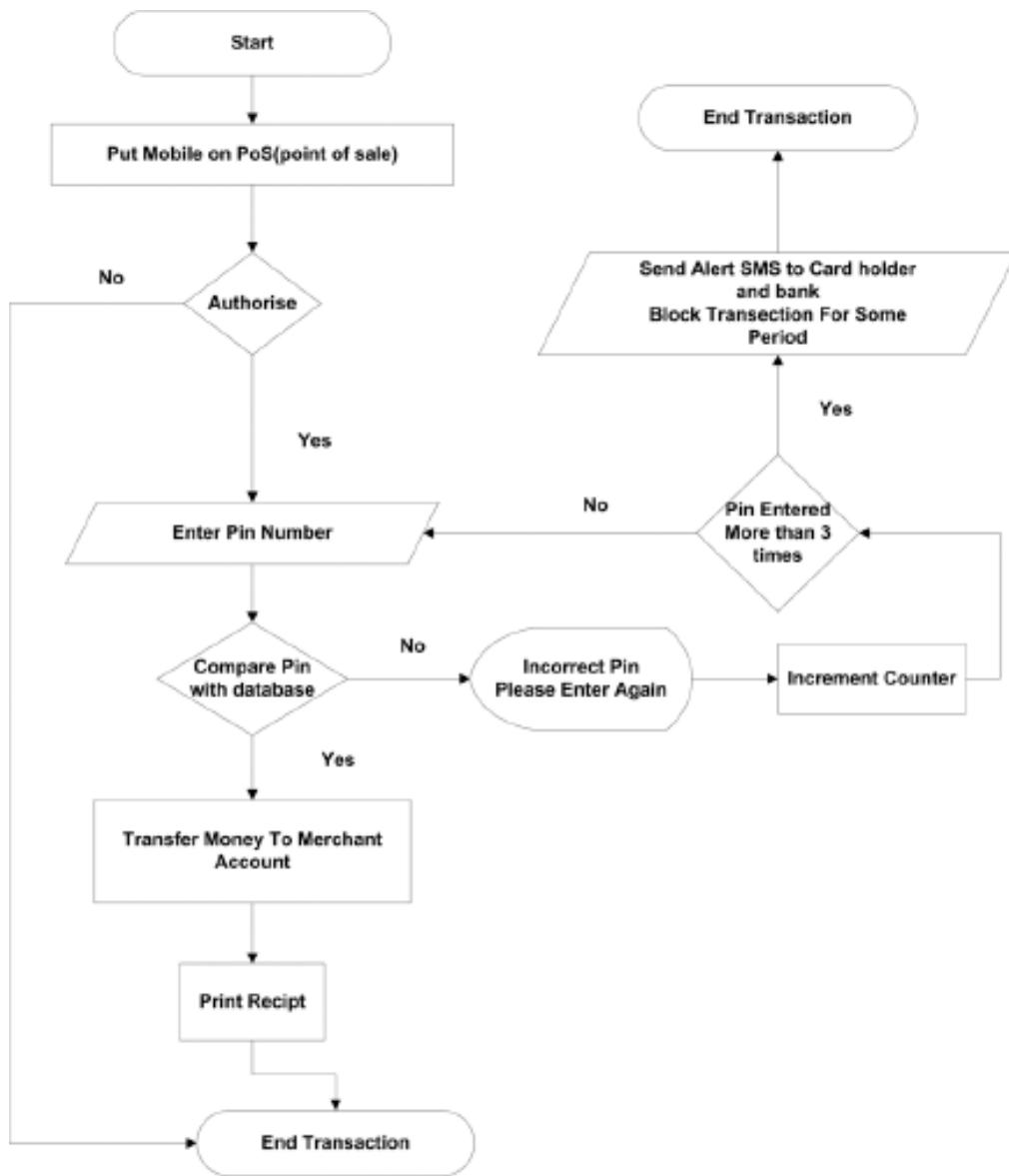


Figure 2.4: NFC payment flow chart

Source: “NFC and NFC payments: A review”

There are different areas where contactless payment has been implemented, such as transportation. An article by Noer et al. [15] sees the use of RFID technology to pay for transportation services at a terminal. Chen et al. [16] create a payment system integrated with the Citizen Digital Certificate. The difference in application areas does not affect the

comparison of these two articles. In both articles, the security of the system is prioritized to prevent system breaches and corruption of data. The articles present different modes of payment system. Noer et al. prioritize the offline mode of the system, meaning more data is stored on the card to ensure transaction execution at the terminal, while Chen et al. utilize the online mode of the system, making requests back and forth within the system. Both systems are effective in their tasks however one requires storing user's encrypted information on the card which makes it susceptible to misuse if stolen. However, storing all the information on the card makes for a faster system because the terminal has access to all information in one tap. One contrast between these two articles is the scenario, in transportation, a fixed price is being deducted at the terminal. This allows the system to rely on speed and use of on-card storage. However, in integrating the Citizen Digital Certificate with contactless payment, continuous security checks must be made to ensure that the card holder is indeed making the transaction, at this stage the user has to be authenticated.

User authentication in both systems is the second most crucial stage. The cardholder should be able to prove their identity to the system as a form of confirmation, ensuring the transaction goes through. Chen et al. make use of the manual mode of authentication which requires the use of a Personal Identification Number (PIN) from the user, this may slow the process by a few seconds or act as an inconvenience to the customer; however this manual check is relevant for the system to confirm identity. In the transportation scene, two authentication keys are added to the user information allowing the system to have access to information and validate transactions without manual interference. This type of system is more convenient for customers. Adopting the tap-and-go form of payment in a dynamic payment environment requires additional information from the user to prove identification. Such data includes location, which

can be used as a determining factor. Applications such as Apple Pay and Google Wallet use this approach to create a tap-and-go environment; however, they also make use of the manual PIN input to ensure the user when something unexpected is occurring, such as a substantial payment [17]. There is no set standard on the implementation of a contactless payment system, and it varies with the scenario. Contactless payment systems can be categorized as Touch and Go, Touch and Confirm, Touch and Connect, or Touch and Explore. Chen et al.'s system is within the Touch and Confirm category as it deals with varying amounts of money.

Per the research, it is recognized that contactless payment can be used in different areas where payment is required for services. In the context of Mobile Money (Momo) in Ghana, the use of USSD applications by telecommunications for finances puts users at risk. It jeopardizes the integrity of the data coming into the system. The development of wireless technology has opened the doorway to NFC payment, which can be used by retailers within the country. In order to achieve this, the preferred payment category is that of the Tap and Confirm. A payment system of this nature would be new to the Ghanaian market, making this prototype the first of many implementations. The system should be able to authenticate and validate users before ensuring a transaction. Though an offline system like the transportation payment system would be preferable and faster, an online system would allow for a better initial implementation of the system.

Chapter 3: Design

3.1 System Design

The system is meant to interact with the user and the mobile money service provider, allowing users to make requests to the provider. In order to do this, the system must be connected to the Service provider. In designing the system, continuous interaction with the service provider is implemented to ensure the success of every event. When a request is made, the system processes the data and forwards that request to the Service provider. Based on the feedback of the provider, a response is sent to the client should any additional information, such as authorization, be required, the system requests it from the user, as seen in figure 3.1. This design ensures that both the service provider and the system have the same data at every point in time, thereby increasing the integrity of data.

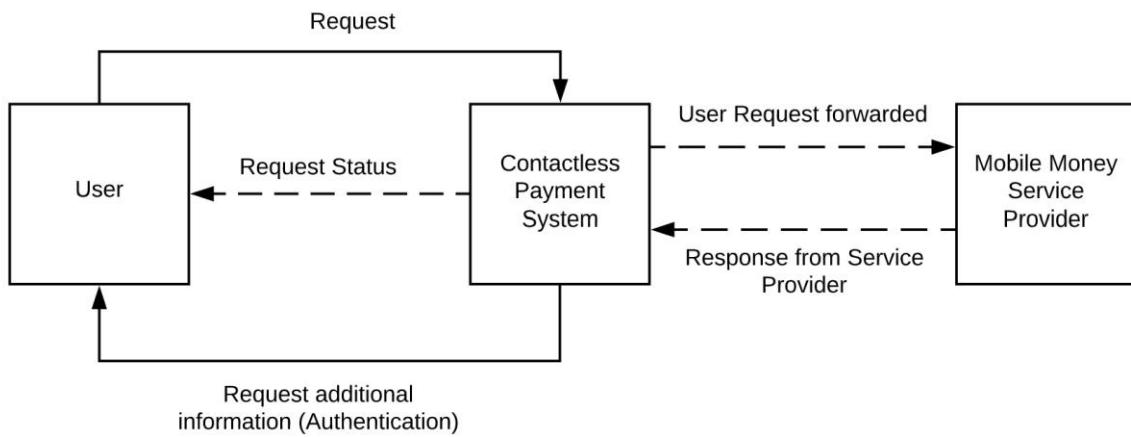


Figure 3.1: High-level design of overall payment system

3.1.1 System Flow

The system flow refers to how users and the system interact with each other and conditions that affect the output. The flow of the system is represented in the flow chart diagram in figure 3.2.

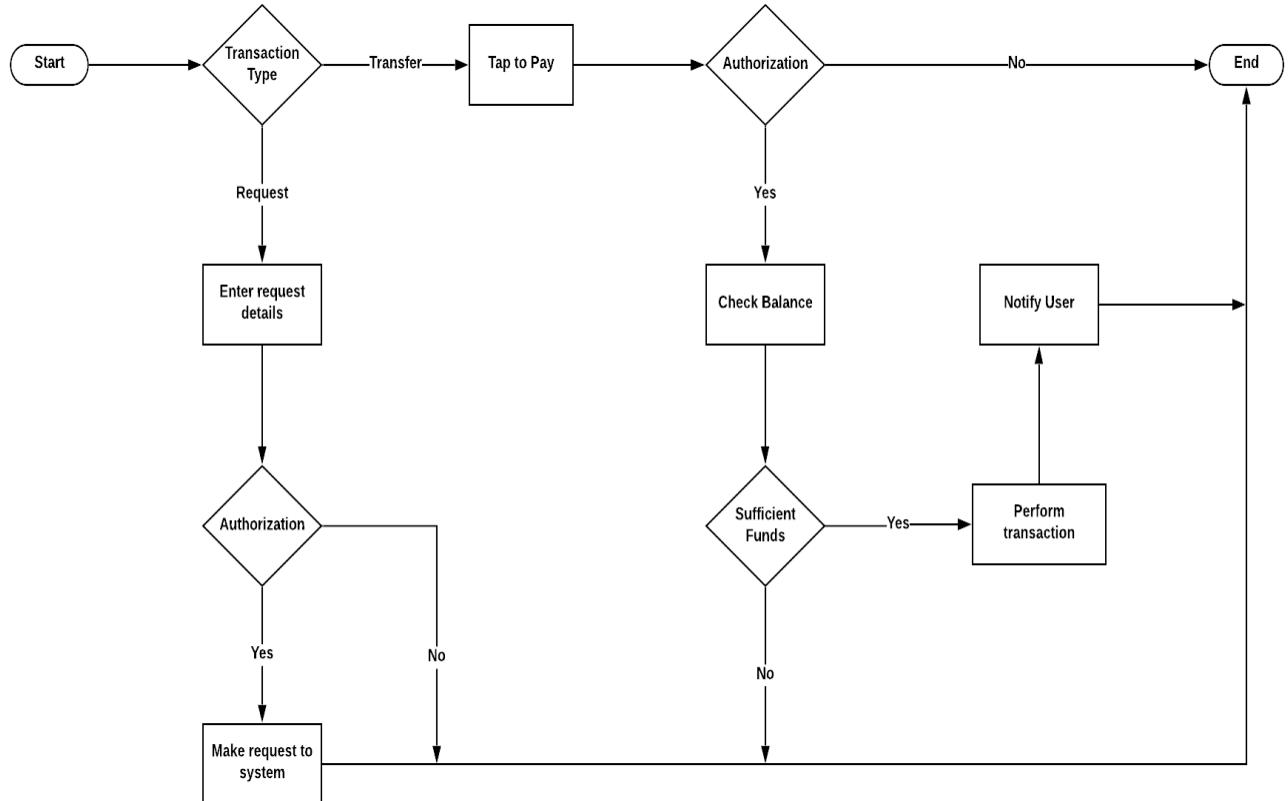


Figure 3.2: User Flow chart diagram

The flow diagram depicts a user can either request for payment or transfer money to a merchant for providing a service, and depending on which path is taken, the user has a different flow in the system. Without authorization by PIN or confirmation message, the transaction will end, and the process will have to be restarted. In the case of payment, insufficient funds can trigger the termination of the process.

3.2 System Architecture

The system is a web-based system that has many features and components. The backend of the system, which can be developed using a JavaScript framework, a Java framework, or a Python framework. A frontend for the admin portal can be built using any of the JavaScript frameworks such as React, Vue, or Angular. A microcontroller to perform the task of reading card data during transactions and authorization for the transaction. A mobile application that can be created using the native operating system language or by a framework. There are different frameworks for mobile applications, some of which include Flutter, Ionic, and React Native. A database is also required to store the user data. There are two categories of databases SQL and NoSQL. Authentication services may be utilized when considering the security of the system. The two main architecture designs will be **client-server architecture** and **software architecture**.

3.2.1 Software Architecture

The software architecture depicts the various aspects of the system and the relationships that exist between them. The system has an API that allows it to feed data from the database and the mobile money service provider. The aspects of the system are broken down in the diagram below.

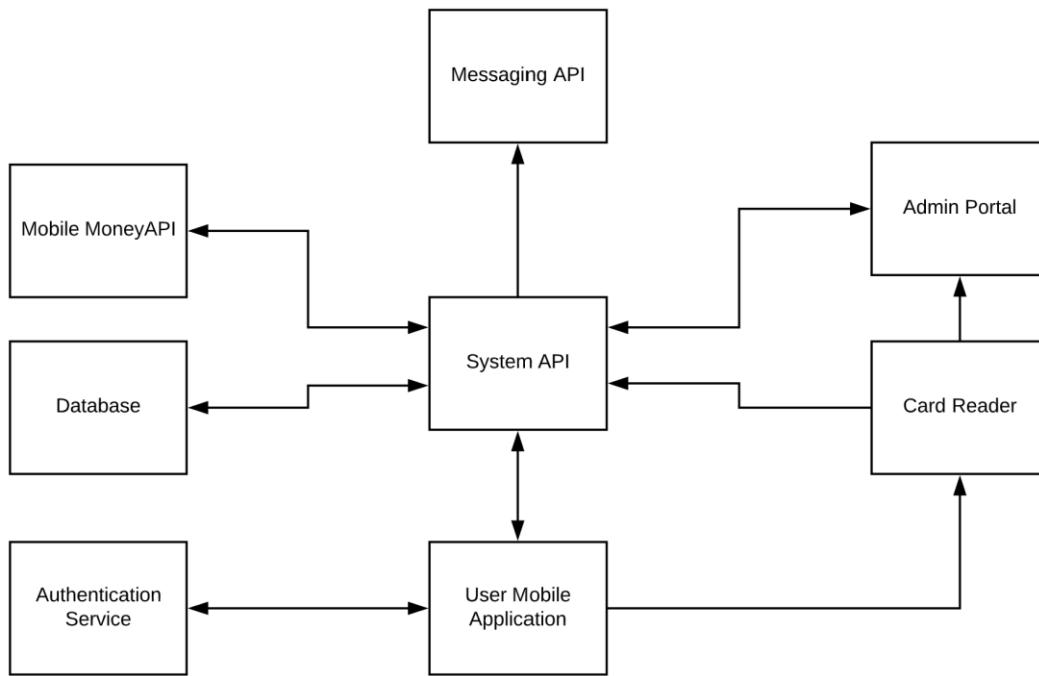


Figure 3.3 Software Architecture

Each component passes data to the system API, which returns relevant data needed by the user. For example, the mobile application passes the user's data on a specific transaction to the System API, which connects to the Mobile Money API and performs the transaction returning the output to the user. All relevant data concerning a user is stored in the database.

From figure 3.3, it can be observed that requests go through the System API for processing before it is distributed to the other sections of the system. The third-party APIs used will be accessed by the System API to complete specific actions such as mobile money-related actions and messages that will provide notification of the transaction completion. The system architecture shows the different parts of the system; however, it does not explain how the user is going to interact with the system, and the client-server architecture seeks to highlight that.

3.2.2 Client-Server Architecture

The client-server architecture displays the user interaction with the system. The system is hosted on a server and connected to the database. The Administrator and users interact with the portal (Web App) and mobile application, respectively, to send requests to the server. The server will connect to the third-party APIs and the database to retrieve information to be displayed to the user.

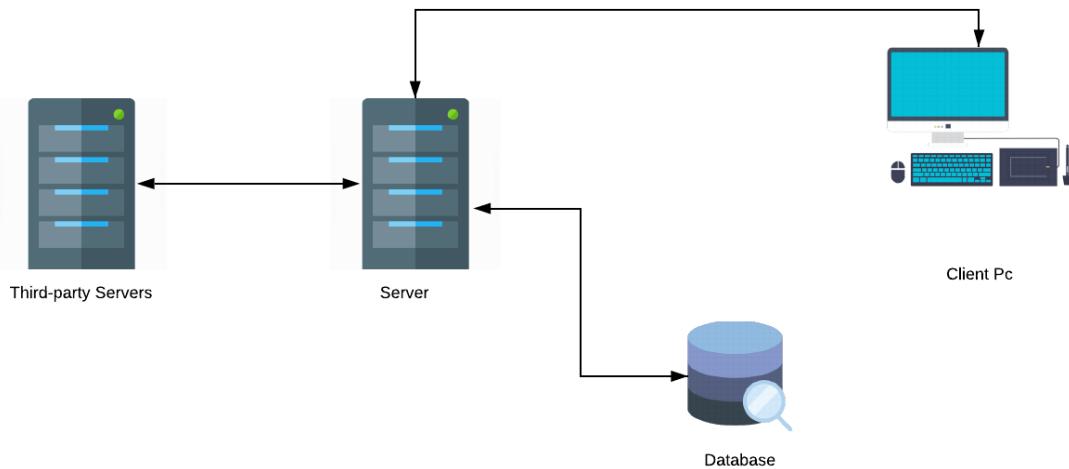


Figure 3.4 Client-Server Architecture

3.3 Functional Requirements

The following functional requirements require implementation:

1. Registration
2. Authentication
3. Perform transactions

These technical requirements each have their needs which are as follows:

1. **Registration**

- a. Administrators should be able to register users unto the platform
- b. Administrators should be able to activate user accounts

2. Authentication

- a. Users should be able to log in to the system
- b. Users should be able to validate transactions

3. Perform Transactions

- a. Users should be able to conduct transactions using NFC card
- b. Users should be able to conduct transactions using the mobile app
- c. Users should be able to request receipts of past transactions

3.4 System Users

The system is designed for the average user of mobile money within the Ghanaian market. However, mobile money is not used by consumers only; there are suppliers and merchants involved in the mobile money ecosystem. Below are some significant roles that directly interact with the system:

- a. **MTN:** These users are the administrators of the system; they register end-users unto the mobile money platform and will oversee distributing the smart cards that will ensure contactless payment. They are also responsible for the distribution of smartcard readers to merchants to ensure smooth and safe transactions. MTN is also a significant user because its API is open-source, allowing the testing of the system in its early stages.
- b. **Mobile money users:** The users are registered on the platform to ensure the simple, fast disbursement of money via mobile money. These users can be subdivided into

two, merchants and regular. Merchants will have access to the card reader together with the mobile app of users.

- c. **Developers:** Developers should be able to integrate the system into any mobile money distributor within the country.

3.4.1 Use Case Diagram

Below is a use case diagram of the system.

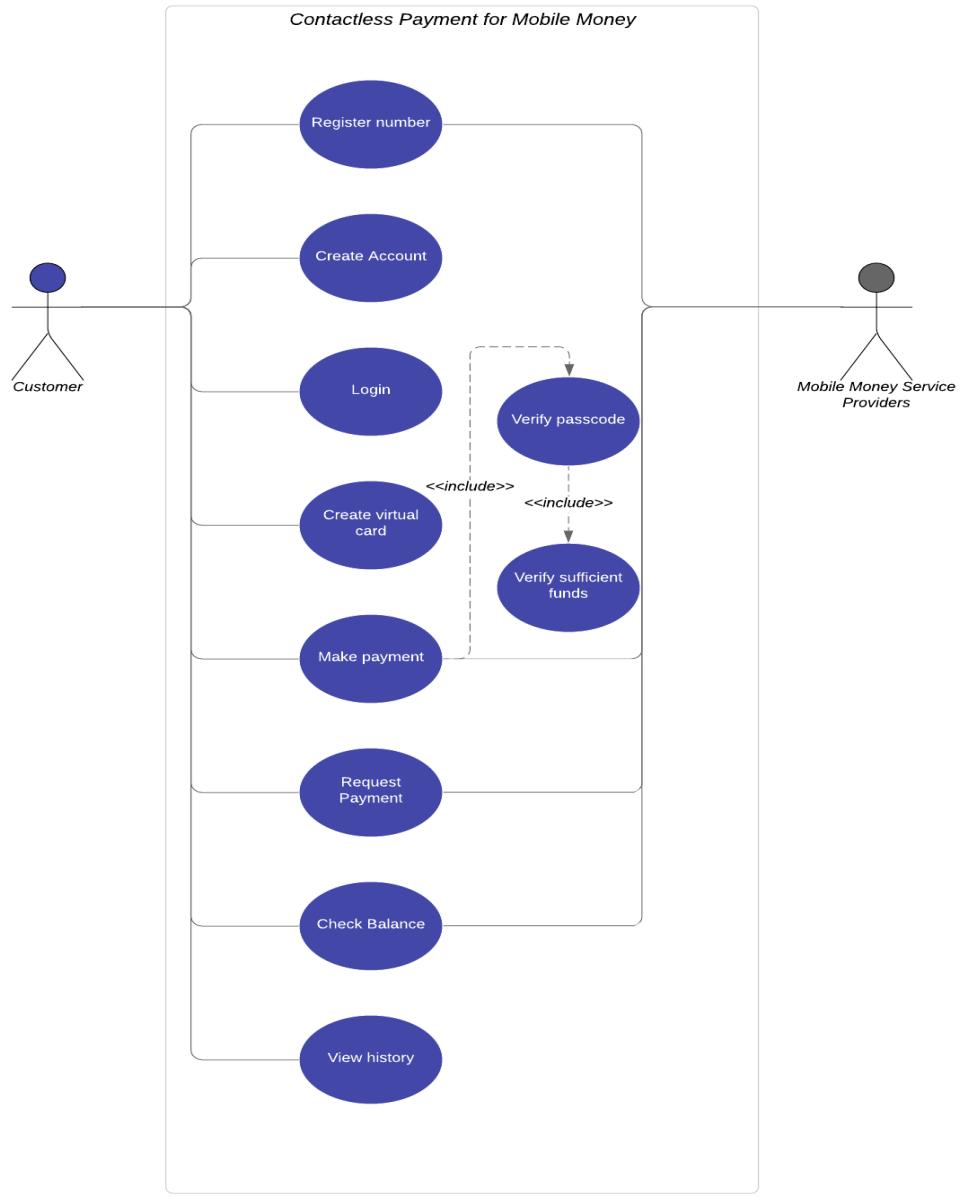


Figure 3.5 Use case diagram

3.5 Non-Functional Requirements

Security: Sensitive data is going through the system; hence it is paramount to protect the system from external attacks. This requirement is considered from the start to the completion of the order. All data on the system will use POST requests to prevent easy access.

Sensitive data per user will be encrypted before being saved on the database, and an encryption module will be used so that there is no fixed pattern generated in the encryption process that will make it easy to decipher. The system will implement two-factor verification for first-time users of the app; this form of attestation will ensure that the user number is correct and corresponds to the registered credentials in the database.

Performance and load-balancing: An increase in users comes to a need for balancing the requests made on the server. Heroku is a cloud hosting platform that allows developers to host applications on a server [18], it has an in-built load balancer that allows it to manage the client requests made to the server in an instance or period. The system must be efficient and handle all requests in quick time to provide the user with real-time response and be fast enough to conduct the transactions.

3.6 Hardware Requirements

The system has minimal hardware involved, and the equipment should be capable of communication via NFC. A microcontroller will be required to make the requests to the API via the internet. The available options for microcontrollers include but are not exclusive to Arduino and Raspberry Pi. Comparatively, the raspberry pi serves as a better option due to its overall performance and memory capability; however, the Atmega chip is not a bad option for creating a cost-effective solution. An additional criterion to be considered was connectivity. The raspberry pi has an in-built Wireless Fidelity (Wi-Fi) card. At the same time, the Atmega chip, however, will require a Global System for Mobile communication (GSM) or an ethernet module to allow connectivity. The Pugh matrix below shows the factors considered and the weight associated with each.

Table 3.1: Pugh Matrix

Criteria	Arduino (Baseline)	Weight	Raspberry Pi	ATMEGA
Cost	0	2	-1	+1
Size	0	1	+1	0
Ease of Use	0	3	+1	-1
Performance	0	4	+1	0
Speed	0	5	+1	+1
Total			+3	+1

The hardware components include the following:

- Raspberry Pi



Figure 3.6.1 Raspberry Pi 3b

- PN532 NFC NXP RFID Module



Figure 3.6.2 PN532 NFC module

- NFC smart card



Figure 3.6.3 PVC NFC card

The NFC sensor used is the Pn532 module, which has a range of about 2 -7 cm. The range of the sensor means proximity is required for the card details are read. Modern smartphones have NFC modules built into them, making it easier to operate on this hardware via mobile apps. The smart card has a specific set of hexadecimal characters it transmits when in proximity to the reader; this allows each user's card to be mapped to their profile on the platform. During the registration process, a user's card identification number will be mapped to their account and

stored in the database, allowing queries to be made when the card is scanned. The card reader given to merchants will have read-only ability to ensure that no card is overwritten during the transaction process.

Chapter 4: Implementation

The chapter describes the implementation of the system with regards to the languages of choice used on the backend, frontend, mobile app and card reader programming of the system, the frameworks, and the database used to store user data. The mechanisms and tools used to ensure usability and security will also be described in the chapter.

4.1 Overview

The primary language used for developing the backend is JavaScript using ExpressJs in tandem with NodeJs, which is a runtime environment. The language used for the frontend is the Cascading Style Sheet (CSS), Hyper-Text Markup Language (HTML) and JavaScript (Js). The framework used for the frontend portal is a JavaScript-based User Interface (UI) framework called ReactJS; this framework will be used together with a bootstrap framework. The mobile app will be built using a google developed cross-platform framework, Flutter, which is programmed in Dart. The database of choice is MongoDB for its flexibility in development and non-relational features. The microcontroller will be programmed using Python, which is the language of the raspberry pi. The tools used during the development process are Visual Studio Code, Postman, GitHub, and command line.

4.2 Backend Development

4.2.1 Programming Language – JavaScript ExpressJS

Express is a fast, minimalistic web framework for NodeJs. The minimalistic nature of Express allows the creation of REST APIs with full flexibility while utilizing minimum server space [19]. Express gives the developer full control of the system allowing the use of different dependencies such as MongoDB or MySQL and others. Express relies on the development environment, NodeJs, to provide the server as well as a debugger for the framework. Express

was chosen to allow uniformity across both ends of the web app and the excellent support community it possesses. The backend is responsible for processing and storing information input into the system from the frontend into the database, MongoDB, and vice versa. The system will notify users of the status of their transactions via SMS. Twilio is a third-party messaging service/platform that allows the developer to engage with customers on any channel – from text messages to emails, phone calls to video, intelligent chatbots, and back [20]. For the project, only text messages will be utilized. Administrators are verified on the system at two levels, user-level and server level. The backend verifies a user's access and capabilities via a JSON web token – a JSON web token is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed [21].

4.2.2 API Structure

The application programmable interface (API) structure used in the development of the system is generated using the express-generator, which can be called in the command line. This enables the entire project to be broken down into subsystems and communicate with each other, forming the complete platform. Express' template provides simplicity and modularity in the development process; these subsystems can be called by making requests to them, thereby calling their functionality into play. An example of the structure can be found below.

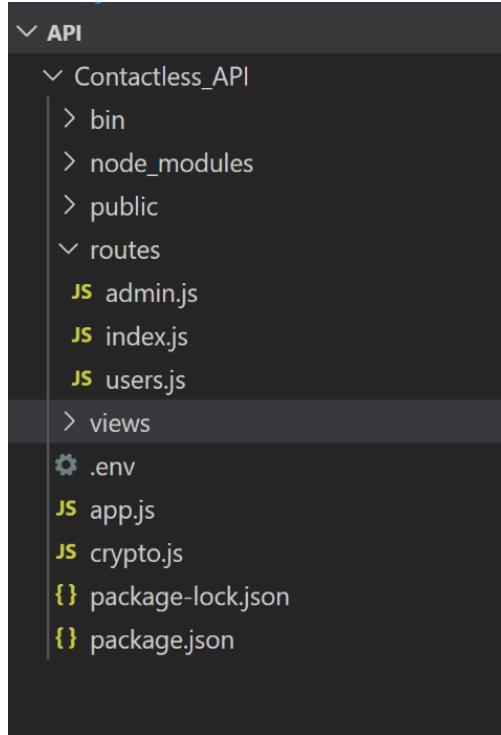


Figure 4.1: Structure of Express API

Figure 4.1 displays the assembly of folders and their respective files. *app.js* is the core of the template; this is where all function calls are made to all subsystems or modules. The functions or requests of the different user roles are separated and can be found in the routes folder and makes debugging much more straightforward because the codebase is structured. Although template comes with several folders, not all these folders will be used, for this system, the functions will be held in the route, and all rendering of webpages will be done independently of the API.

4.3 Frontend Development

4.3.1 Administrator Portal

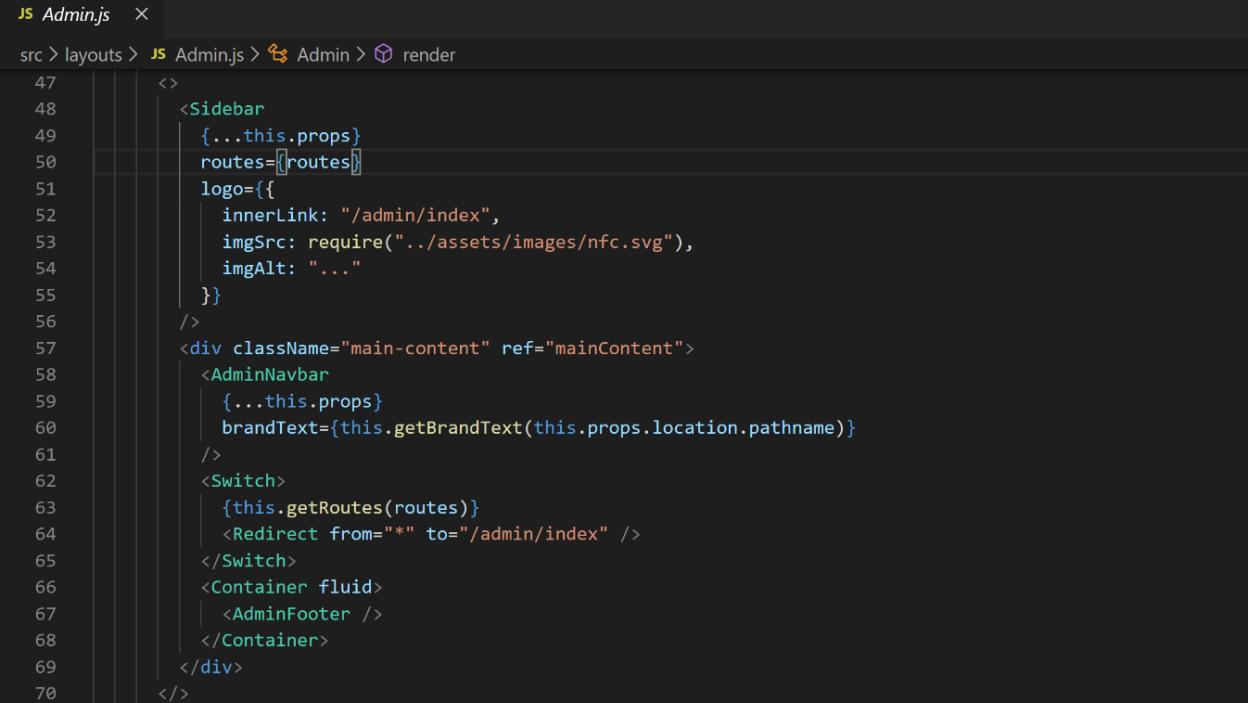
The Administrator portal is one of the vital facets of the system. This is where the service provider registers users unto the system and can monitor transactions, users, and daily statistics

via the system. Due to its relevance, this interface should be simple enough to operate by employees; it should have graphs and data tables to structure the information appropriately. Per the requirements of the portal, the frontend is developed using a JavaScript framework. Different open-source JavaScript-based frameworks could be used, **Angular**, **Vue**, or **React**, however for the portal, React will be used. React was chosen for its lightweight in terms of server space because it is a JavaScript library for building user interfaces meaning unused functionality is not incorporated and its component-based structure. **React** is component-based, encapsulated components are built and manage their state, components can be combined to form complex interfaces [22]. User authentication is handled using Auth0, upon authorization, a valid access token with an expiry will be generated and passed to the backend to prove accessibility. In creating the Administrator portal, I could either develop the entire structure from the ground up or use a template and mold it to my need, of which the template was preferred. The template of choice was the Creative-Tim Argon dashboard template, a user-friendly, open-source design system based on Bootstrap 4 and is licensed under MIT License [23]. The template comes with login and register pages, a dashboard with graphs and data tables, which display data in an organized table with added functionality such as querying, deletion, and updating. This is important because it allows easy manipulation of the data in the database by the user.

4.3.1.1 Template Customizations

The frontend is developed using a framework; hence it is not rendered from the server. To acquire data from the backend, HyperText Transfer Protocol (HTTP) requests must be made, and the response is rendered. As such, the template only needs customization in the content displayed. Due to the component-based nature of React, code is not repeated because the

component can be used and reused. The template comes with two layouts, the Administrator layout, and the authentication layout. These layouts are the encapsulating components of the template, the Administrator layout, for example, encompasses the header, footer, and sidebar, as shown in Figure 4.2. Figure 4.3 shows how the layout is used when rendering pages. The layout being rendered is dependent on the route path and defaults to the login page when loaded. Every page loaded appears in the body of the layout meaning only the content changing will change instead of the whole page being loaded, thereby reacting relatively fast.

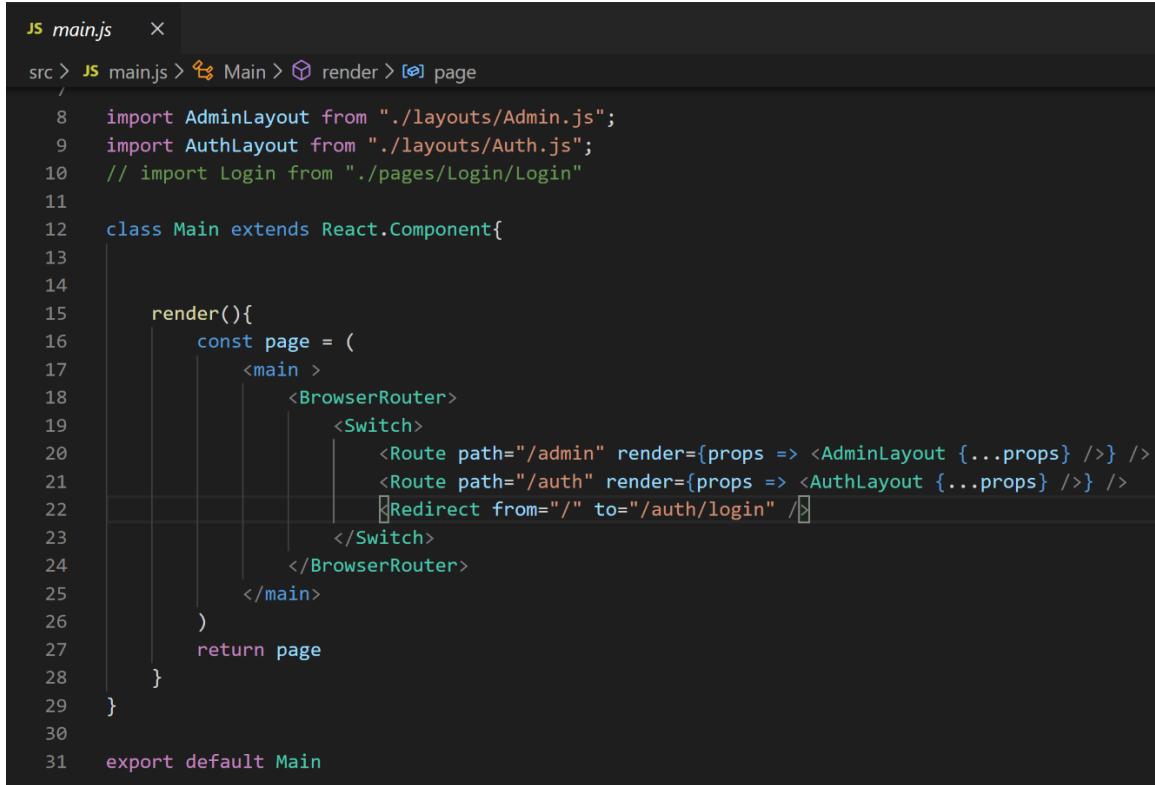


```

JS Admin.js ×
src > layouts > JS Admin.js > Admin > render
47   <>
48     <Sidebar
49       {...this.props}
50       routes={routes}
51       logo={{
52         innerLink: "/admin/index",
53         imgSrc: require("../assets/images/nfc.svg"),
54         imgAlt: "..."
55       }}
56     />
57     <div className="main-content" ref="mainContent">
58       <AdminNavbar
59         {...this.props}
60         brandText={this.getBrandText(this.props.location.pathname)}
61       />
62       <Switch>
63         {this.getRoutes(routes)}
64         <Redirect from="/" to="/admin/index" />
65       </Switch>
66       <Container fluid>
67         <AdminFooter />
68       </Container>
69     </div>
70   </>

```

Figure 4.2: Admin.js showing the layout of the template



```

JS main.js   ×
src > JS main.js > ⚙️ Main > 📂 render > 🏠 page
/
8 import AdminLayout from "./layouts/Admin.js";
9 import AuthLayout from "./layouts/Auth.js";
10 // import Login from "./pages/Login/Login"
11
12 class Main extends React.Component{
13
14
15     render(){
16         const page = (
17             <main >
18                 <BrowserRouter>
19                     <Switch>
20                         <Route path="/admin" render={props => <AdminLayout {...props} />} />
21                         <Route path="/auth" render={props => <AuthLayout {...props} />} />
22                         <Redirect from="/" to="/auth/login" />
23                     </Switch>
24                 </BrowserRouter>
25             </main>
26         )
27         return page
28     }
29 }
30
31 export default Main

```

Figure 4.3: main.js showing how the layout is selected and rendered

4.3.2 Mobile Application

End-users of the system will have two main ways of interaction, using the NFC enabled cards or a mobile device with NFC capabilities. The mobile application will be built using a cross-platform framework. A cross-platform framework is a better option because it allows a single codebase to run on the different mobile operating systems, iOS and Android, thereby reducing the amount of time spent developing the app and allowing deployment for both operating systems at the same time. There are different frameworks, each with their unique way of building and deploying the application; these include React Native, Ionic, and Flutter; out of the options, Flutter was the preferred choice. Flutter is Google's User Interface (UI) toolkit for building beautiful, natively compiled (dart programming language allows the operating system to compile the source code into code that is executable by the operating system) applications

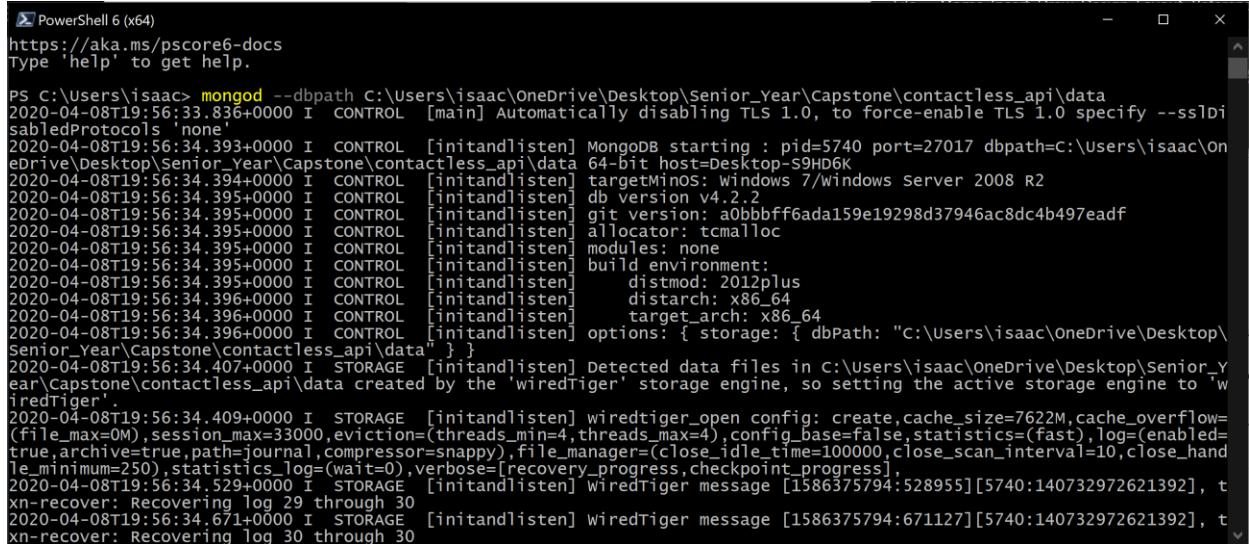
on mobile, web and desktop from a single codebase [24]. Flutter has fast development, expressive and flexible UI coupled with native performance, making it faster than the other frameworks. The development process allows developers to build on a device or emulator to give a hands-on experience of the application. The application will be used to provide the users with information on their account and allow them to perform transactions, and this is made possible by scanning the card, which will request the data from the database for that account. All user authentication and verifications will be done in-app using third-party services, such as Firebase authentication. The application will be created from the ground up because there is no template available for mobile applications. The application is structured in folders that separate the image files from the pages and other relevant files. Communication between the application and the backend will be done using HTTP requests.

4.4 Database Development

4.4.1 Mongo Database

MongoDB is a general-purpose, document-based, distributed database built for modern application developers and the cloud era. This means data is stored in JavaScript Object Notation-like (JSON) documents. The nature of the structure makes it more expressive and more powerful than the traditional row/column model. Mongo provides a powerful query language, queries are in JSON format, making them composable. The advantages of using a database with this structure include the ability to promote agile development and the ease of manipulating the data in the database. MongoDB has simple ways of performing CRUD (create, read, update, and delete) operations that are developer friendly. Node has a driver that allows it to connect to MongoDB, either running locally or on a server. In the developmental stages of the project, the database is run locally using the command line, as shown in Figure 4.4. Figure

4.5 shows how data is structured in the database (collections and documents) using a graphical user interface (GUI), Mongo Compass. Figure 4.6 shows the data structure of the database showing the simplicity and flexibility of the database.



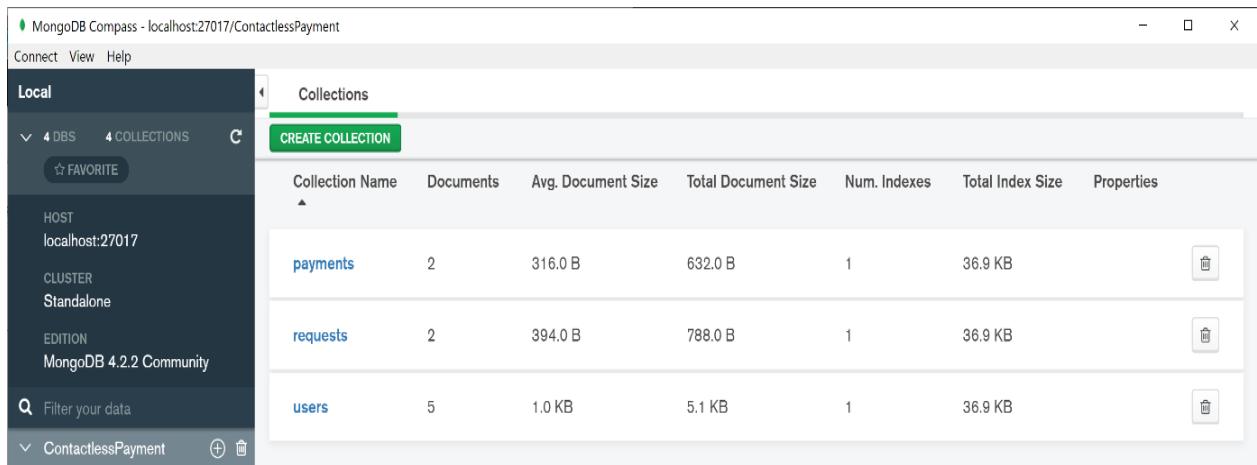
```

PowerShell 6 (x64)
https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\isaac> mongod --dbpath C:\Users\isaac\OneDrive\Desktop\Senior_Year\Capstone\contactless_api\data
2020-04-08T19:56:33.836+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-04-08T19:56:34.393+0000 I CONTROL [initandlisten] MongoDB starting : pid=5740 port=27017 dbpath=C:\Users\isaac\OneDrive\Desktop\Senior_Year\Capstone\contactless_api\data 64-bit host=Desktop-S9HD6K
2020-04-08T19:56:34.394+0000 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] db version v4.2.2
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] git version: a0bbbff6ada159e19298d37946ac8dc4b497eadf
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] allocator: tcmalloc
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] modules: none
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] build environment:
2020-04-08T19:56:34.395+0000 I CONTROL [initandlisten] distmod: 2012plus
2020-04-08T19:56:34.396+0000 I CONTROL [initandlisten] distarch: x86_64
2020-04-08T19:56:34.396+0000 I CONTROL [initandlisten] target_arch: x86_64
2020-04-08T19:56:34.396+0000 I CONTROL [initandlisten] options: { storage: { dbPath: "C:\users\isaac\OneDrive\Desktop\Senior_Year\Capstone\contactless_api\data" } }
2020-04-08T19:56:34.407+0000 I STORAGE [initandlisten] Detected data files in C:\Users\isaac\OneDrive\Desktop\Senior_Year\Capstone\contactless_api\data created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'
2020-04-08T19:56:34.409+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7622M,cache_overflow_low=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress]
2020-04-08T19:56:34.529+0000 I STORAGE [initandlisten] wiredTiger message [1586375794:528955][5740:140732972621392], txn-recover: Recovering log 29 through 30
2020-04-08T19:56:34.671+0000 I STORAGE [initandlisten] wiredTiger message [1586375794:671127][5740:140732972621392], txn-recover: Recovering log 30 through 30

```

Figure 4.4: local MongoDB server running in the command line



MongoDB Compass - localhost:27017/ContactlessPayment

Local

HOST: localhost:27017

CLUSTER: Standalone

EDITION: MongoDB 4.2.2 Community

Collections

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
payments	2	316.0 B	632.0 B	1	36.9 KB	
requests	2	394.0 B	788.0 B	1	36.9 KB	
users	5	1.0 KB	5.1 KB	1	36.9 KB	

Figure 4.5: Structure of collections in MongoDB

```

_id: ObjectId("5e444b97af1e350a0436cbf")
name: "Otema Yirenkyi"
account: "user"
msisdn: "0241237654"
id_type: "voters"
id_number: "2445345223"
balance: 1449.5
currency: "GHS"
collection_reference: Object
  iv: "2479d2e320c34dadc95952de01673ead"
  encryptedData: "a63c7984f0a37700cb0a1ab134af04d515773f14aa21f1f749f6f0ed264b4a6aaf3420..."
disbursement_reference: Object
  iv: "2479d2e320c34dadc95952de01673ead"
  encryptedData: "96341f6407cab08ac4c571101c6b18dd225e00c77c998daeb2d32549c47cb50a270512..."
date: 2020-02-12T19:01:43.797+00:00
status: "active"
lastModified: 2020-03-04T20:50:47.897+00:00
collection_Key: Object
  iv: "d5f3e94dc0efe741636b4ecd9ee4dbb4"
  encryptedData: "961d9dcc5aa695aede194b1c68cc87aafc3d08861a2dc7514b492f0e53de8962151..."
disbursement_Key: Object
  iv: "d5f3e94dc0efe741636b4ecd9ee4dbb4"
  encryptedData: "aed641f1261b7e28c0ced27c0839a8e0c6527f8f0cc88bc7b0b5361078a8ae8834d0bf..."
requests: Array
  0: Object
    request_id: ObjectId("5e60147012c66e4238c95a0f")
payments: Array
  0: Object
    payment_id: ObjectId("5e6014a712c66e4238c95a10")

```

Figure 4.6: structure of data in a document

The above figure shows that data has been encrypted to ensure the security of the system; this meets the non-functional requirement of the system. MongoDB, although non-relational, allows for relationships to exist between tables; an example is the *requests* key in Figure 4.6, which holds an array of objects containing the unique ids for request in another document. Auxiliary functions were created to simplify the manipulation of data when interacting with the database, and these functions are then called when a specific route is hit. Figures 4.7 and 4.8 show the functions created in order to increase efficiency and reduce duplicate code.

```

saveOne : function (collection, data, callBack){
    MongoClient(process.env['DB_URL'], {useUnifiedTopology:true}, (err,db)=>{
        if (err) throw err
        var dbo = db.db(`${process.env['DB_NAME']}`)
        dbo.collection(collection).insertOne(data, (err,res) => {
            if (err) {
                console.log(err)
                return;
            }
            db.close;
            // console.log(res.ops)
            callBack(res)
        })
    })
},
updateOne : function (collection, query, data, arrData){
    MongoClient(process.env['DB_URL'], {useUnifiedTopology:true}, (err, db) => {
        if (err) throw err
        var dbo = db.db(`${process.env['DB_NAME']}`)
        if (arrData != null && data !== null){ ...
        }if (arrData === null){ ...
        }
        if (data === null){ ...
        }
        db.close
    })
},

```

Figure 4.7: Saving and Updating documents

```

findMany : function (collection,query,callBack){
    let resp =[]
    MongoClient(process.env['DB_URL'], {useUnifiedTopology:true}, (err, db) => {
        if (err) throw err
        var dbo = db.db(`${process.env['DB_NAME']}`)
        dbo.collection(collection).find(query).toArray((err, res) => {
            if (err) callBack({status:400,data:err.message})
            db.close
            res.forEach(element => {
                resp.push({"_id":element._id,
                           "name":element.name,
                           "currency":element.currency,
                           "msisdn":element.msisdn,
                           "status":element.status})
            })
            callBack({status:200,data:res})
        })
    })
},
findOne : function (collection,query,callBack){

    MongoClient(process.env['DB_URL'], {useUnifiedTopology:true}, (err, db) => {
        if (err) throw err
        var dbo = db.db(`${process.env['DB_NAME']}];
        dbo.collection(collection).findOne(query,(err, res) => {
            if (err) throw err
            db.close
            callBack(res);
        })
    })
}

```

Figure 4.8: Finding documents

4.5 Card Reader Programming

The programming of the card reader is one of the crucial components of the project. The card reader uses a sensor that runs on a Raspberry Pi model 3B. The script to read and write the content of the card is written in Python programming language, which is the official language of the pi. The card reader is used in two scenarios; these scenarios need to be prepared for; these scenarios are: (1) when the Administrator is registering a user, and (2) when a user is making payment. In the first scenario, the card reader reads the card data and sends it to the frontend to be added to the user data stored in the database. The data will be transmitted through HTTP requests and will be in a read-only input field to prevent the Administrator from overwriting the content. The second scenario requires the read content to be sent to the API to perform the payment transaction. After the card details are read, the reader will request authorization via a Personal Identification Number (PIN) before sending the data to the backend API to perform the transaction for the user, and this simplifies the payment process.

4.6 Screenshots

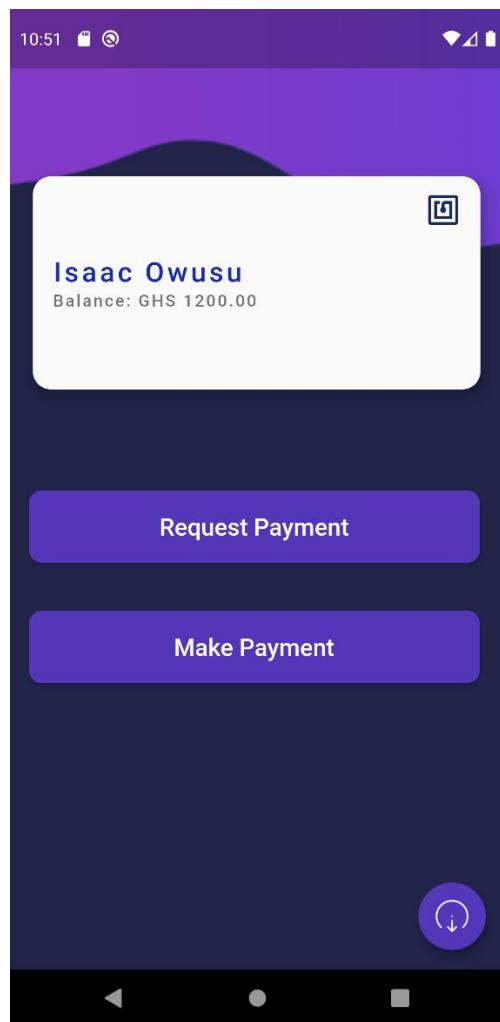


Figure 4.9.1: Homepage of the mobile app

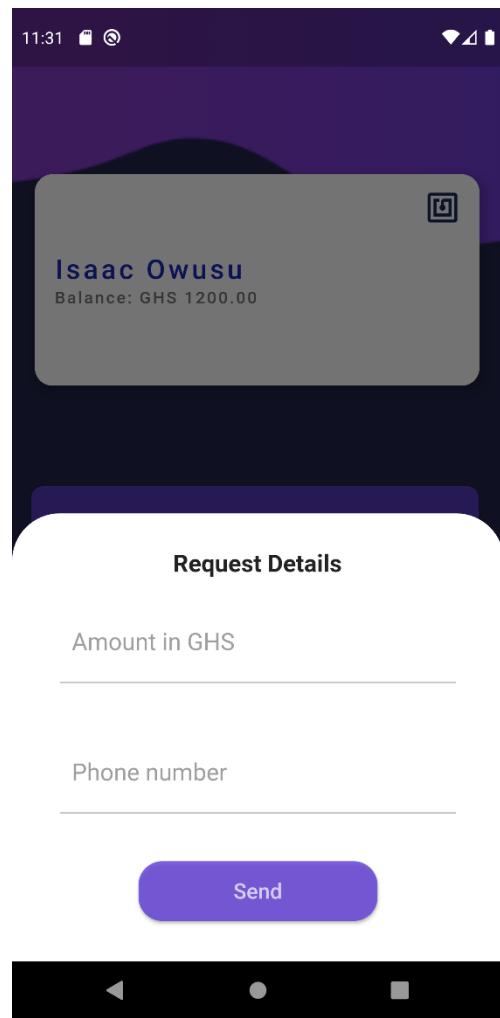


Figure 4.9.2: User Payment request page

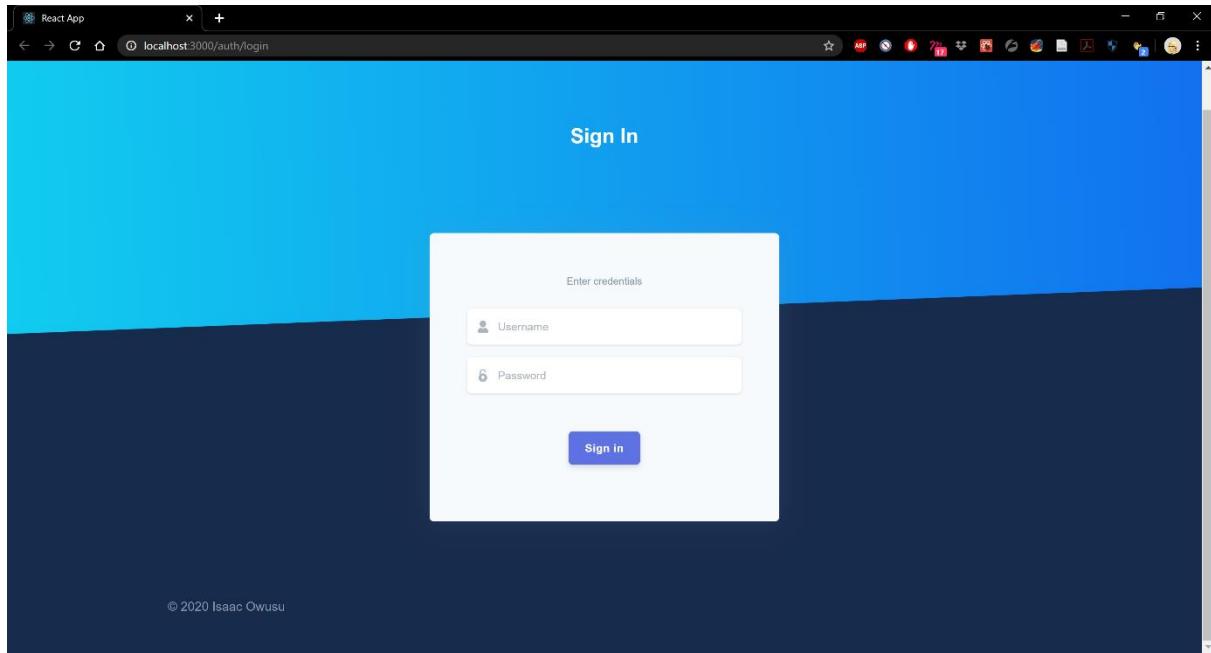


Figure 4.9.3: Login page of the Administrator portal

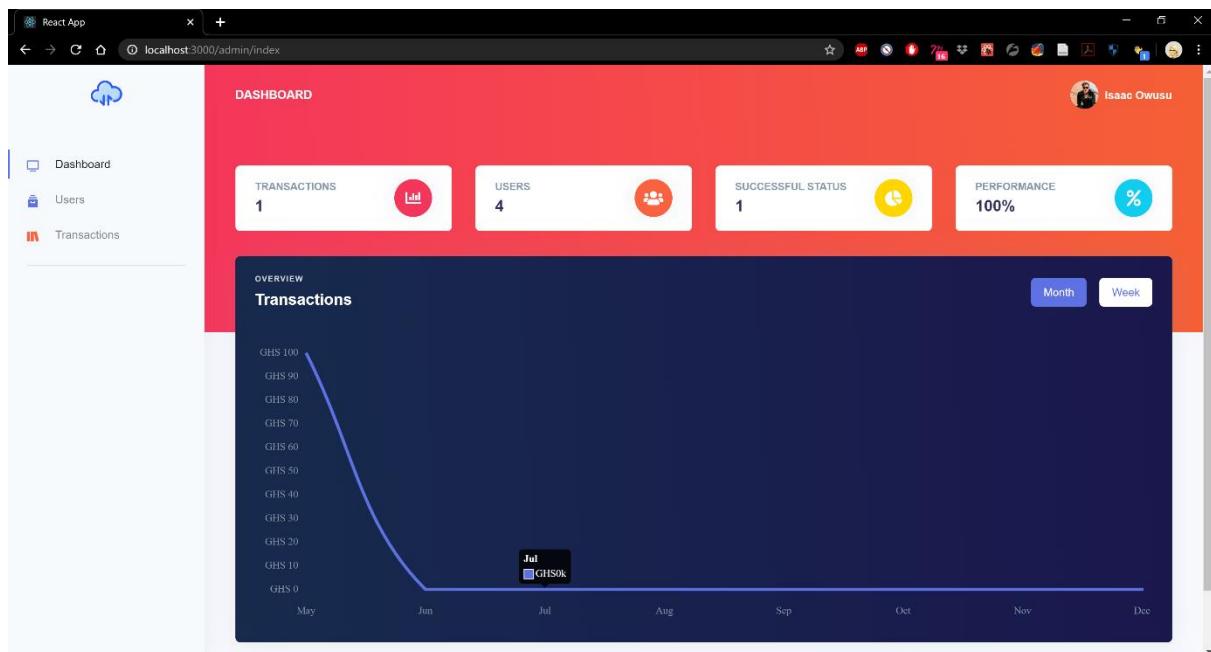


Figure 4.9.4: Dashboard of the Administrator portal

Chapter 5: Testing and Results

In software development, testing of the system is as necessary as its creation. It is useful in assessing the different facets of the system, finding flaws and bugs in the system, and resolving those flaws. The testing of this system was categorized into two, the system tests and the user tests.

5.1 System Testing

The system test can involve testing the functionality of the system independent of the user. The conventional means of testing systems are unit tests, feature tests, and full functionality tests.

5.1.1 Unit Tests

Unit tests involve the testing of individual components of a software. In this test, only vital components that are the bedrock of the system were tested. The authentication of the system, albeit a vital component, was not tested using this kind of test because it was being done by a third-party service. Registering a new user unto the platform, activating the account, having access to all registered parties, and a user having access to his/her account via the mobile application are some of the vital components to be tested in the system.

In order to perform unit tests in Express, a library known as Jest was added to the project, and a script file was written in order to test specific routes. A new database was created to allow the test to deposit and retrieve the dummy information without affecting the final database. The format of the test file can be seen in Figure 5.1.

```

JS routes.test.js ×

tests > JS routes.test.js > ⚡ describe('Post Endpoint') callback
  1  const request = require('supertest');
  2  const app = require('../app');
  3
  4  describe('Post Endpoint', () => {
  5    it('Add a new user', async ()=>{
  6      const res = await request(app)
  7        .post('/api/admin/register/user')
  8        .send({
  9          "name": "Samara Morgan",
 10          "account": "user",
 11          "msisdn" : "0248632941",
 12          "id_type": "passport",
 13          "id_number": "G2717",
 14          "balance": 50,
 15          "currency": "GHS"
 16        })
 17        // console.log(res)
 18        expect(res.body).toMatchObject({code: '01', message: 'User created', data: []})
 19      },10000)
 20    })

```

Figure 5.1: Unit testing in ExpressJS

In the *package.json* file, I configured and set a script command to execute Jest when called. In the *routes.test.js* file, the function tests the adding of a user to the database with the data seen. Upon the first attempt, the test should succeed and, when tried again, should fail because the user already exists in the database. The results of the test can be seen below:

```

> jest

  PASS  tests/routes.test.js (9.818s)
    Post Endpoint
      ✓ Add a new user (4448ms)

    POST /api/admin/register/user 200 4245.845 ms - 48
    Test Suites: 1 passed, 1 total
    Tests:       1 passed, 1 total
    Snapshots:  0 total
    Time:        9.916s, estimated 16s
    Ran all test suites.

```

Figure 5.2: successful unit test to register user

```

● Post Endpoint > Add a new user

expect(received).toMatchObject(expected)

- Expected - 2
+ Received + 2

Object {
-   "code": "01",
+   "code": "02",
   "data": Array [],
-   "message": "User created",
+   "message": "This number is already registered",
}

```

Figure 5.3: failed unit test because the number was registered

The other routes of the system are tested in the same fashion.

5.1.2 Feature Testing

Upon completing unit tests of the routes/subsystems came the testing of the features from the user point of view. This involved testing the features available in the Administrator Portal and mobile application to ensure the right flow of data and user interactions were occurring. Testing included features like login and registering a card on the application. The mobile application should trigger the phone to activate the NFC sensor when trying to add a card to, and this should resemble that of Figure 5.4. The feature tested in the Administrator portal was the registering of a user unto the platform; this is the first and most crucial stage of the user journey in the system. Figures 5.5.1 and 5.5.2 show the view of a user being added to the system and the data requested, respectively. Upon success, the user will be added to the list of registered users. The other features were tested accordingly.

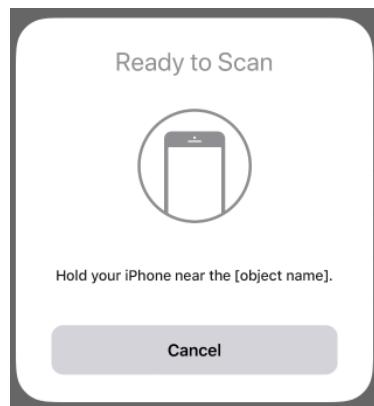


Figure 5.4: application requesting to scan card

Source: “www.apple.com/support”

The dashboard has a red header with the title 'USERS'. On the left, there's a sidebar with icons for Dashboard, Users, and Transactions. The main area shows four summary cards: 'TRANSACTIONS 1' (red), 'USERS 4' (orange), 'SUCCESSFUL STATUS 1' (yellow), and 'PERFORMANCE 100%' (blue). Below this is a table titled 'Registered Users' with the following data:

ACCOUNT HOLDER	ACCOUNT TYPE	PHONE NUMBER	STATUS	ACTIVATION	DATE	Actions
Sarah Owusu	merchant	0248732941	Active	<input checked="" type="checkbox"/>	Sat Apr 25 2020	<i>i</i>
Ben Owusu	user	0247283429	Active	<input checked="" type="checkbox"/>	Sat Apr 25 2020	<i>i</i>
Ashesi	merchant	0241234567	Active	<input checked="" type="checkbox"/>	Sat Apr 25 2020	<i>i</i>
Isaac Owusu	user	0240227269	Active	<input checked="" type="checkbox"/>	Sat Apr 25 2020	<i>i</i>

At the bottom left is a copyright notice: © 2020 Isaac Owusu.

Figure 5.5.1: Administrator view of registered users

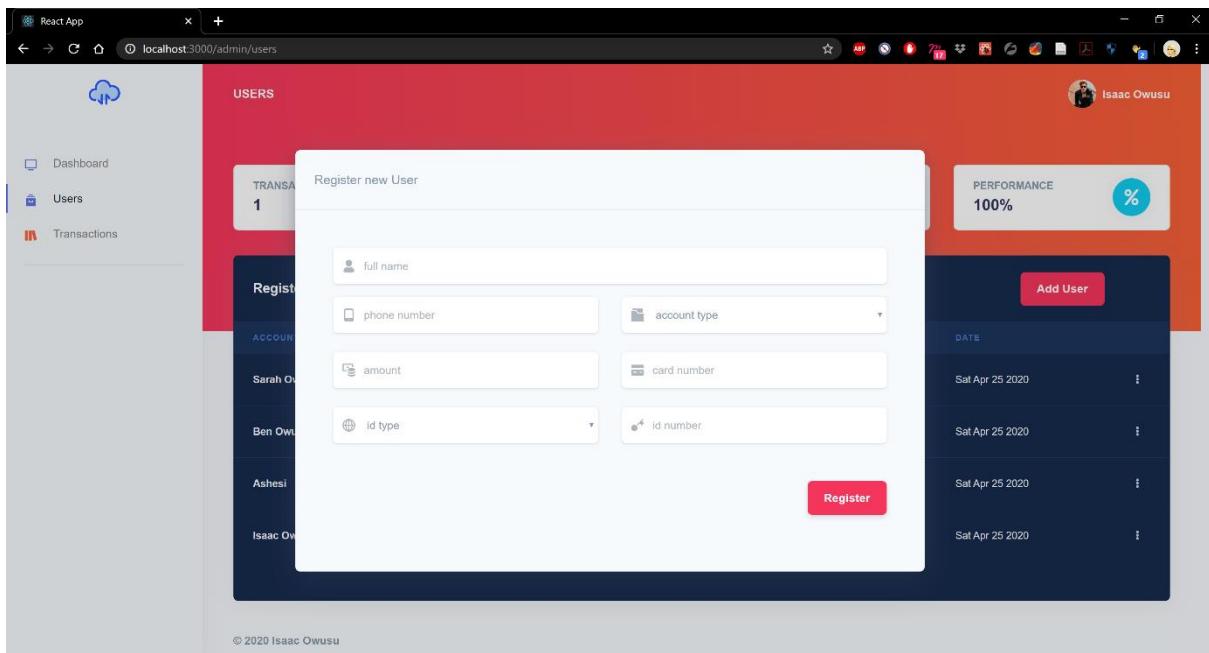


Figure 5.5.2: User registration form on the Administrator portal

5.1.3 User Testing

User testing involves a user simulating the entire process of the system with me as the Administrator. To conduct this test, the user had to be present when I was testing because the entire system depended on the custom card reader built, thereby making it challenging to conduct remote tests. The development version of the app was installed on the phone because hosting it would require getting certification from the application stores. This form of testing was impeded due to the presence of the global pandemic, COVID-19; however, a survey was conducted to attain the duration of conducting a transaction using USSD against the NFC enabled transaction. The data derived will aid the validation of the project's premise. Out of the 54 responses received it was found that 83% (45) of users had mobile money accounts already registered providing a basis for this project and increasing the chances of the system being implemented, Figure 5.6 shows a graph of the distributions of registered accounts (blue) and unregistered (orange).

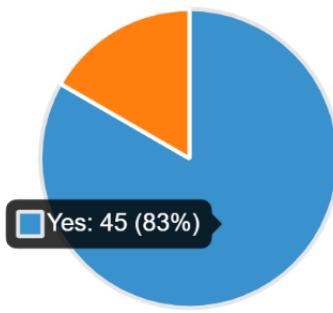


Figure 5.6: Graph of mobile money users

In testing the solution derived against the current system, the time taken to complete that process and not the entire transaction process because the solution is not live; hence a lag would alter the results. When analyzing the data, the One-Way ANOVA test was used with the null hypothesis that the mean of the two processes is equal. The significance level used was 5%. Figure 5.7 shows the results of the ANOVA test, and the values used can be found in the appendix; the p-value shown is less than the significance level, thereby disproving the null hypothesis. When proving the better solution, the average values were compared, and the average time for the solution was seen to be significantly less, thereby justifying the system is a better alternative.

Anova: Single Factor						
SUMMARY						
Groups	Count	Sum	Average	Variance		
NFC	45	124	2.755555556	2.461616162		
(USSD)	45	3458	76.84444444	1601.134343		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	123506.1778	1	123506.1778	154.036529	4.9018E-21	3.949321007
Within Groups	70558.22222	88	801.7979798			
Total	194064.4	89				

Figure 5.6: ANOVA test results

Chapter 6: Conclusions and Future Work

The purpose of this project was to implement a more efficient and secure means of conducting transactions with mobile money in Ghana. The system was broken down into three different parts, the user app, the portal, and the card reader. The functionalities of the portal include registering and activating user accounts and getting user, transactional, and statistical data of the system. The user app allows users to login, conduct transactions, and view details of the account. The card reader reads cards using a sensor and transmits the information to the system for processing.

6.1 Conclusion

USSD applications provide mobile banking service providers with an offline payment system that allows them to reach users in remote areas. USSD has many benefits and weaknesses. The major weakness of USSD applications is security issues. The security issues recognized in these applications pose a threat to the integrity of data, which may lead to loss of trust and money. NFC payment provides a more secure and fast means of payment. In creating an NFC payment system, an online model was opted for with continuous interaction with the service provider. In developing the system, the main objectives were achieved, the system could register and activate user accounts, users could request for transactions and conduct payments via NFC cards and mobile applications, and security measures implemented worked as expected. Some additional features were not implemented due to the priority assigned to them; this includes transaction history for users and generating reports by the Administrator. These features were not implemented because they do not affect the main objective of the system, which is to create a system that is a fast and secure payment system that can be used as a potential substitute to the USSD applications. Analysis of both systems proved that NFC

payment is faster than USSD payment, and applying current encryption algorithms ensures the safety and security of the system.

6.2 Challenges

Creating a project of this magnitude and intent usually takes a team of developers to complete with significant backing. Across the span of the project, I encountered several challenges that stalled the development and design process, thereby requiring critical thinking and the application of skills acquired during my time at Ashesi University.

The first challenge I faced was in the design phase. As stated, it takes a team of developers to complete a product of this nature; as a result, in designing the product, I had to consider the relevant features that would allow the idea behind the product to cut across without underlying the importance of certain features. I had to apply knowledge in areas of computer-human interaction to ensure the best possible user experience for the product. When designing the target market had to be considered in the process as if they refused to use the system, then it was of no use to the public. With the users being the priority, different services had to be employed to facilitate better user experience.

The second challenge faced dealt with third party services and platforms to use. Numerous third-party services perform the same task, and deciding on one proved a challenge. All third party services have documentation and different ways of integrating it with a system, and they had different subscriptions that needed to be factored in the designing of the system. I spent a good number of weeks deciding which services to use without incurring much cost. I had to watch video tutorials and consult with developers to help me in the decision and development process. Some of the third party services were designed to produce dummy results, and this created some challenges to the system as it affected the output being delivered to users.

Finally, the most significant challenge faced was due to the global pandemic, COVID-19. The pandemic caused the closure of the Chinese market first and resulted in the components ordered being significantly delayed. Although many components were not required for the project, they were essential to the system, getting these components were of high priority. The pandemic also affected the user testing of the system in its entirety; accurate feedback could not be derived because users had to remain in self-isolation during this period, and I required them to be present in order to conduct the full test.

6.3 Future Work

The product is far from complete and requires more features to ensure sustainability and longevity.

Firstly, getting access to all mobile money service providers is vital to the promotion of the product; the current system is managing with the MTN MoMo API, which limits and slows down the capability of the system. Collaboration with service providers allows expansion across all mobile money users.

Secondly, a more budget efficient model of the card reader should be designed and manufactured to allow widespread use of the product.

Thirdly, the mobile app should incorporate multiple card assignments, meaning multiple cards can be assigned to one user and allow users to view past transactions.

Fourthly, extra features such as monthly reports could be added to the portal to make it easier for telecommunication companies to compare results.

Finally, an offline version of the system should be created to allow transactions in remote locations where merchants lack a means of linking to the payment system in real-time.

References

- [1] S. Ozyurt, "Ghana is now the fastest-growing mobile money market in Africa," *Quartz Africa*. <https://qz.com/africa/1662059/ghana-is-africas-fastest-growing-mobile-money-market/> (accessed Dec. 21, 2019).
- [2] P. Maurya, "Cashless Economy and Digitalization," *SSRN Electron. J.*, 2019, doi: 10.2139/ssrn.3309307.
- [3] "The benefits of a cashless society," *World Economic Forum*. <https://www.weforum.org/agenda/2020/01/benefits-cashless-society-mobile-payments/> (accessed Apr. 15, 2020).
- [4] J. Kagan, "Contactless Payment," *Investopedia*. <https://www.investopedia.com/terms/c/contactless-payment.asp> (accessed May 08, 2020).
- [5] "Tap to pay with Visa." https://usa.visa.com/content/VISA/usa/englishlanguagemaster/en_US/home/pay-with-visa/contactless-payments/contactless-cards.html (accessed Dec. 21, 2019).
- [6] F. Campbell, "Contactless payments continue to grow in the UK," *Mobile Transaction*, Jan. 02, 2019. <https://www.mobiletransaction.org/contactless-payments-uk/> (accessed Dec. 21, 2019).
- [7] "Ghana working towards becoming cashless economy." <https://www.devere-vault.com/blog/Ghana-working-towards-becoming-cashless-economy> (accessed Apr. 15, 2020).
- [8] "Cashless Economy 2019: A Smart New Year Resolution," *Modern Ghana*. <https://www.modernghana.com/news/910974/cashless-economy-2019-a-smart-new-year-resolution.html> (accessed Apr. 15, 2020).
- [9] Ecobank, "Ecobank - EcobankPay." <https://www.ecobank.com/personal-banking/payments-transfers/ecobankpay> (accessed Dec. 21, 2019).
- [10] C. N. Updated: 3/29/2020, "What is Data Integrity and How Can You Maintain it?" *Inside Out Security*, Mar. 27, 2018. <https://www.varonis.com/blog/data-integrity/> (accessed May 08, 2020).
- [11] M. Toorani and A. Beheshti, "Solutions to the GSM Security Weaknesses," in *2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, Sep. 2008, pp. 576–581, doi: 10.1109/NGMAST.2008.88.
- [12] "A3 Algorithm." <http://www.wirelessdictionary.com/Wireless-Dictionary-A3-Algorithm-Definition.html> (accessed May 09, 2020).
- [13] K. K. Lakshmi, H. Gupta, and J. Ranjan, "USSD — Architecture analysis, security threats, issues and enhancements," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, Dec. 2017, pp. 798–802, doi: 10.1109/ICTUS.2017.8286115.
- [14] N. S. S. Shobha, K. S. P. Aruna, M. D. P. Bhagyashree, and K. S. J. Sarita, "NFC and NFC payments: A review," in *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, Nov. 2016, pp. 1–7, doi: 10.1109/ICTBIG.2016.7892683.
- [15] A. Noer, Z. B. Hasanuddin, and D. Djamaluddin, "Implementation of RFID based raspberry Pi for user authentication and offline intelligent payment system," in *2017 15th International Conference on Quality in Research (QIR) : International Symposium on Electrical and Computer Engineering*, Jul. 2017, pp. 251–255, doi: 10.1109/QIR.2017.8168491.

- [16] W. Chen, K. E. Mayes, Y. Lien, and J. Chiu, “NFC mobile payment with Citizen Digital Certificate,” in *The 2nd International Conference on Next Generation Information Technology*, Jun. 2011, pp. 120–126.
- [17] “Apple Pay,” *Apple*. <https://www.apple.com/apple-pay/> (accessed May 10, 2020).
- [18] “Cloud Application Platform | Heroku.” <https://www.heroku.com/> (accessed Apr. 15, 2020).
- [19] “Express - Node.js web application framework.” <https://expressjs.com/> (accessed Apr. 07, 2020).
- [20] “Twilio - Communication APIs for SMS, Voice, Video and Authentication,” *Twilio*. <https://www.twilio.com> (accessed Apr. 15, 2020).
- [21] “JSON Web Token Introduction - jwt.io.” <https://jwt.io/introduction/> (accessed Apr. 15, 2020).
- [22] “React – A JavaScript library for building user interfaces.” <https://reactjs.org/> (accessed Apr. 07, 2020).
- [23] “Argon Dashboard React by Creative Tim,” *Creative Tim*. <http://demos.creative-tim.com/argon-dashboard-react/#/admin/dashboard> (accessed Apr. 08, 2020).
- [24] “Flutter - Beautiful native apps in record time.” <https://flutter.dev/> (accessed Apr. 08, 2020).

Appendix

Appendix A: Survey data

NFC	(USSD)
1	60
5	55
2	40
3	60
4	60
1	45
2	50
4	50
5	50
1	40
6	120
2	180
3	120
5	100
2	45
6	62
2	55
3	180
1	55
3	70
3	120
5	50
5	64
2	180
3	60
1	50
4	45
2	90
5	64
2	100
1	50
4	50
2	64
4	120
2	90
1	60

5	120
2	60
1	180
2	50
1	50
1	64
1	60
3	60
1	60