UFODIAMA MIRACLE NMESOMA
2018/242949

COMPUTER SCIENCE
PHYSCIAL SCIENCES
UNIVERSITY OF NIGERIA, NSUKKA

AUGUST, 2O23

TITLE

DEVELOPING AN AUTOMATED LOAN PROCESSING SYSTEM FOR COMMERCIAL BANKS

CERTIFICATION

This is to certify that "**UFODIAMA MIRACLE NMESOMA**" with the Reg number "**2018/242949**" of the final year of Computer Science has submitted a project report entitled "**Developing An Automated Loan Processing System For Commercial Banks**" in partial fulfilment for the award of Bachelor of Science Degree of University of Nigeria, Nsukka in session 2021-2022. It has been found to be satisfactory and hereby approved for submission.

……. ……..

Ufodiama Miracle Nmesoma Date

APPROVAL

This research project was approved by the Department of Computer Science for an award of the Bachelor of Science (B.Sc.) degree in Computer Science, University of Nigeria Nsukka.

By

…… Date ……

Mr Raymond Nworgu

(Project Supervisor)

…… Date ……..

Prof. F.S. Bakpo

(Head of Department)

DEDICATION

This project report is dedicated first and foremost to my parents, for their love and support, and to my brother, Somto, for all the help. I also dedicate this report to Martins Victor Onuoha, his excellent mentoring contributed immensely to my being able to build this project.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the many people and organizations who helped me throughout my undergraduate studies. First and foremost, I want to thank my supervisor, Mr. Raymond Nworgu, for his insight, and valuable information, which have benefited me during my research and writing for this project. Without his assistance and instruction, this endeavour would not have been feasible. During my studies, I could not have asked for a better supervisor.

In the same vein, I would like to thank Mr. Daniel Anomfueme (DanEl) for always answering my many questions and reassuring me when I needed to make decisions about my project.

I would also like to thank the members of the tech communities on campus for their contribution to my growth, especially GDSC (Google Developers Students Club, UNN.)

In addition, I will be eternally thankful to my parents for their prayers, love, and unwavering support during my academic years. I was able to get to this point because of their encouragement and the financial aid I received.

Finally, I would like to thank me for believing in me.

ABSTRACT

An automated loan processing system tailored specifically for commercial banks. With the ever-increasing demand for efficient financial services, the need for streamlined loan processing has become paramount in the banking sector. This project aims to address this challenge by leveraging modern technologies resulting in reduced wait time and fewer Non-performant Loans (NPLs.) This system uses an Object-oriented Analysis and Design Methodology (OOADM) to solve the stated problems. The project utilizes React-Native, a versatile framework for building cross-platform mobile applications, to create a user-friendly and accessible front-end interface. This ensures that both bank employees and customers can interact seamlessly with the system. Postgres is employed on the database side because its reliability and scalability are crucial for handling the vast amounts of data associated with loan processing in commercial banks. This system can also be implemented easily in any Nigerian commercial bank which provides loans to borrowers.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0 Background to the Study

In today's fast-paced world, convenience and efficiency are key, for this reason, commercial banks are continuously looking for innovative ways to improve client satisfaction and optimise operations.

The loan processing system is a crucial area of banking that has experienced tremendous changes in recent times. In the past, those in need of fiscal loans had to make repeated trips to a bank branch and deal with tedious paperwork, lengthy lines, and long processing times [3].

Currently and with the advent of more cutting-edge technology in the financial sector, a few platforms and organisations in Nigeria have tackled this problem and subsequently reduced the stress availing a loan used to entail in the past, some of these platforms include finance applications like the banking application Palmpay, Fairmoney, KudaBank, and Opay.

The platforms mentioned above are not without problems, one such is the dismal loan recovery that is the hallmark of these digital loaning applications, to be short people borrow money on these platforms and never pay. According to the Central Bank of Nigeria (CBN), one of the most significant challenges to commercial bank stability lies in the effective recovery of such loans [1].

Recognizing the need for a more customer-centric approach, this project aims to develop an Automated Loan Processing System for commercial banks in the form of a mobile application that will increase the loan recovery rate for the banks and improve the satisfaction of their customers resulting in a win-win situation for both parties.

This system (mobile application) proposes to leverage Optical Character Recognition (OCR) to process identification documents and asset declarations in order to provide easier extraction of information in the documents provided by borrowers (this is tackling the fact that in a survey conducted, up to 56% of bankers say the biggest challenge in loan processing is “the manual collection of data and subsequent back and forth with the client.” [3]); personalize loans based on the customer's financial goals, risk tolerance, and repayment capacity via questionnaires within the application; and smooth loan recovery processes via reminders (mobile notifications,) and offering affordable repayment plans.

1.1 Statement of the Problem

The existing lending system in Nigerian commercial banks comes with severe limitations and shortcomings. These are some of those shortcomings:

i.) Tedious Paperwork: Traditional lending processes often involve extensive paperwork, which requires borrowers to fill out numerous forms and provide physical copies of documents. The manual documentation process is time-consuming and prone to errors [3], leading to delays in loan approvals and long wait times.

ii.) Accessibility Problems: Physical visits to the bank are often required for loan applications, making it inconvenient for individuals residing in remote areas or with limited physical mobility. This restricts access to financial services and excludes a significant portion of the population from loans and financial aid.

iii.) Lack of Personalization of Loan Plans: Traditional lending processes provide limited flexibility and personalization in loan offers. Borrowers may not have access to loan options that align with their specific needs, resulting in mismatched loan terms, interest rates, and repayment structures. This also holds true for modern, online finance applications.

iv.) Ineffective Loan Recovery Process: Traditional loan recovery methods often lack proactive strategies and rely heavily on manual follow-ups which leads to low recovery rates and challenges in tracking and managing delinquent accounts [3].

1.2 Aim and Objectives of the Study

This project aims to design and develop a fast, user-friendly runnable on mobile smartphones (both Android and iOS devices,) enabling individuals to conveniently access loans from a bank, and banks to adopt intelligent repayment enforcement methods. The application will provide a seamless and efficient borrowing experience, empowering users and banks to manage their loans more efficiently.

Specific objectives of this project include but are not limited to:

i.) To develop a mobile application runnable on a mobile smartphone or laptop for customers of Nigerian Commercial Banks using one bank as a case study.

ii.) To allow users immediately connect to the mobile application by registering and creating their credit profiles on login.

iii.) To allow users to be fitted with loans suited to them after filling out questionnaires on the app.

iv.) To promote loan repayment by providing affordable payment plans and payment reminders from application notifications as well as allowing only users with enough assets that can be used as collateral to take certain loans.

1.3 Significance of the Problem

The development of an Automated Loan Processing Application for Nigerian Commercial Banks is significant for a variety of parties, including banks, customers, and the government.

Banks will benefit from accelerated loan processing as a result of streamlining various stages, such as document collection, verification, and evaluation, allowing them to serve more customers effectively and increase their lending capacity. Furthermore, the application's smarter loan recovery mechanism is of great importance to banks because it protects their financial stability [3].

For the government, this project aligns with its agenda to promote financial inclusion and stimulate economic growth. By streamlining loan processes and reducing barriers to access, the application will enable a broader segment of the population to get loans, which in turn fosters entrepreneurship, investment, and overall economic development. The government can benefit from increased financial inclusion, higher tax revenues, and a more robust financial sector [2][4].

The Automated Loan Processing Application will significantly improve the loan application experience for customers. With a user-friendly interface and simplified procedures, borrowers can easily initiate loan applications, submit required documents electronically, and receive prompt responses. The application's personalized loan recommendations based on advanced algorithms will ensure customers find suitable loan options tailored to their financial needs and repayment capabilities.

The significance of the Automated Loan Processing Application for Commercial Banks in Nigeria lies in its ability to address the challenges of traditional loan processing, bringing efficiency and convenience to both banks and customers.

Overall, this project has the potential to provide significant benefits to its target audience, as well as contribute to the advancement of the field of computer science.

1.4 Scope of the Study

The created mobile application is primarily intended for end-users/customers of Nigerian Commercial Banks located geographically in Nigeria, using one bank as a case study.

In this study, the researcher will cover and develop an innovative loan processing application for commercial banks, providing a convenient and efficient digital platform for customers to initiate loan applications, receive personalized loan recommendations, and facilitate loan recovery, streamlining the overall lending process.

1.5 Limitations of the Study

Due to the sensitive nature of data collected and used by commercial banks about their customers, this study will not attempt to utilise real customer information in building the interface of the application, it will instead create custom use-cases and placeholder customers with varied information, and retain the ability for new users to add their data by signing up on the application.

Chapter 2

Literature Review

2.0 Introduction

This section presents an overview of the research project and the methods used to complete it. The goal is to identify and analyze relevant materials on the topic of automated loan processing systems. This includes a thorough examination of scholarly publications that provide current knowledge, substantive findings, and theoretical and methodological contributions to the subject matter. The focus is on understanding the relevance of this study and determining the optimal approach to meeting this project's requirements.

The theoretical background explores the ideas that foster the project and demonstrates a comprehensive understanding of relevant theories and concepts. This involves describing and comparing various loan processing automation strategies to provide an overview of the field.

The chapter also examines the different technologies and concepts used in this area of research to help readers understand the importance of a functioning, efficient and unbiased loaning and underwriting system in the lives and/or devices of end-users of banks, banks, and bankers. Additionally, it discusses the scope of related publications and their impact on the subject matter.

2.1 Theoretical Background

Research previously carried out on Loan processing automation systems has been overly focused on the ethics of Artificial Intelligence in credit decision-making, overly technical details on neural networks, and the general merits of Artificial intelligence in the finance sector; all of which are important

details. The needs and perspectives of the user which includes everyone who owns and uses a bank account and who may need a loan now or in the future are left out of the research, when in fact they and their needs are the primary targets of a loan processing automation system in the first place.

The other research papers that do not dwell on the details of neural networks, and the ethics and demerits of using AI in the finance sector versus the merits of using AI in making financial decisions instead list and explain how automation/Artificial Intelligence (AI), and inclusive credit systems will affect the African (or International) banking sector and the government which again does nothing to tackle the specific problems faced by the end-users of these systems, particularly in terms of fitting of loan plan that suit the user's need and convenient loan recovery process that work for the users as well as the banks.

Also, existing banking applications that provide loans and automate their loan automation processes do not pay much mind to underwriting and to how they can recover their loans back. For instance, the popular finance application, Palmpay, is now colloquially known by young people as the bank application to take easy loans without the pressure or need to pay back. In their process it would seem the need to pay back is merely a suggestion and not an important part of any lender-borrower dynamic. This project aims to tackle the problem of recovering loans in a way that works for both the lender and the borrower.

The technologies used in this project include:

i.) React Native: React Native is an open-source UI software framework invented by Meta Platforms, Inc. (formerly Facebook) in 2015. It is used to create Android applications, iOS applications, Web applications and a host of other applications that can be run on various platforms (cross-platform). It is used by Facebook,

Microsoft, and Shopify to create Android and iOS applications. Oculus is also using it to create virtual reality applications [8].

ii.) Tailwind CSS: Tailwind CSS is a free and open-source CSS framework created by Adam Wathan in 2017 [9]. The key between this software and other CSS frameworks (like Bootstrap) is that it does not include a set of predefined classes for elements such as buttons or tables. Instead, it generates a list of "utility" CSS classes that may be mixed and matched to style each element.

iii.) StyledComponents: Styled-components is a popular library for styling React applications. It enables you to create bespoke components by incorporating CSS into JavaScript and reusing the components thus styled [10].

iii.) Structured Query Language (SQL): Introduced in the 1970s, SQL is a domain-specific language developed for managing data in a relational database management system (RDBMS) or stream processing in a relational data stream management system (RDSMS). It is used to deal with structured data, which includes relationships between entities and variables [11].

iv.) Postgres: The robust, free PostgreSQL object-relational database system combines the SQL language with several capabilities to reliably store and scale even the most challenging data demands [13]. The POSTGRES project at the University of California, Berkeley is where PostgreSQL first emerged in 1986, and the basic platform has undergone more than 35 years of active development.

v.) NodeJS: Node.js is an open-source JavaScript runtime environment developed by Ryan Dahl in 2009 to run server-side JavaScript applications using the Chrome browser's V8 JavaScript engine. It is mostly used to create the server side of full-stack applications [12].

vi.) Azure Cognitive Services: Developed by Microsoft, Azure Cognitive Services are artificial intelligence (AI) cloud-based services that assist developers in integrating cognitive intelligence into applications without needing specific AI or

data science expertise [14]. In this project, the services are used to extract data from documents provided by users using OCR and to verify said documents.

2.2 Review of Related Literature

The findings of Beck and de la Torre provide valuable insights [2], the authors investigate the issue using a variety of data sources and analytical methodologies, including cross-country comparisons, econometric analysis, and case studies with the population being measured as deposit accounts in commercial banks in developing countries. In the second section of their publication, it is explained that they gathered information from a variety of sources, including surveys, publications, and existing datasets.

The research highlights the importance of financial inclusion and its impact on economic development. They examine various dimensions of access to financial services, such as physical proximity, affordability, and suitability of financial products. The paper presents empirical evidence from different countries and emphasizes the significance of financial infrastructure, regulatory frameworks, and institutional factors in determining access to financial services. The analysis of access to financial services can inform the design and implementation of a loan automation project, particularly in ensuring inclusivity and broadening the reach of financial services improving accessibility of financial services to those who need it the most.

As noted by Atoi in the CBN Journal of Applied Statistics [1], one of the key benefits of efficient loan recovery and credit assessment processes is that they help banks maintain liquidity, capital, and overall stability. This is particularly relevant in the Nigerian banking industry, where banks are many times not properly regulated and/or managed and fold, are closed by the government or are acquired by other banks. Automation can facilitate real-time monitoring of loan

portfolios and enable banks to proactively address emerging issues, such as potential Non-performing Loans (NPLs), thereby contributing to overall banking stability. In [1], for each licensed category, a "restricted" dynamic Generalized Method of Moments (GMM) is used to assess the macroeconomic and bank-specific drivers of Non-Performing Loans (NPLs.) In a panel vector autoregressive framework, the Z-Score is built to proxy banking stability. The findings show that the determinants of NPLs differ between the two types of banks (ones with national licenses and ones with international licenses,) but that the weighted average loan rate is an important macroeconomic driver of NPLs.

Peterson's whitepaper becomes highly relevant in this research. Their study underscores the potential of automation in transforming the loan origination process [3]. It utilises polls to gather data from bankers in commercial banks in the United States. It concludes with the idea that by leveraging automation technologies, repetitive and manual tasks can be automated, reducing the need for extensive paperwork and manual data entry. This not only saves time but also minimizes the chances of errors and enhances data accuracy. Furthermore, the literature highlights how automation can facilitate faster loan processing and decision-making. With automated workflows and intelligent algorithms, loan applications can be evaluated swiftly, allowing for quick loan approvals or rejections. This expedites the overall lending process and enhances customer satisfaction.

By implementing automation technologies, a loan automation project can streamline operations, reduce costs, enhance decision-making, and improve the overall customer experience [3].

In a similar vein as the work of Beck in [2], the paper: “Financial Inclusion and Economic Growth in Nigeria” by J.B. Obayori. [4] provides a Nigerian Business perspective, using an econometric approach and the ARDL model to

study mid-size businesses in the country, this study investigates the impact of financial inclusion on economic growth in Nigeria; emphasising the large role financial inclusion plays in promoting economic growth and reducing poverty in Nigeria. By automating loan processes, lenders can reach a wider population, including individuals and businesses that may have limited access to traditional financial services. Automation of loan processing can also help in creating more inclusive loan offerings, providing flexible terms and repayment plans, and enabling quicker and more efficient loan approvals.

S. Sachan [7] highlights the significance of loan underwriting and the need for efficient and accurate decision-making in this domain. The paper uses the AI method known as the Belief-rule-base (BRB) procedure to quantitatively and qualitatively analyse data from lending institutions in the United Kingdom stored in a credit bureau server. The system proposed in this paper utilizes machine learning algorithms and interpretable models to analyze borrower data, assess creditworthiness, and generate explainable decisions. By adopting an explainable AI decision-support system, lenders can automate various aspects of the underwriting process, reducing manual effort and improving efficiency. The system's use of interpretable models ensures transparency in the decision-making process, allowing lenders to understand the factors influencing loan approvals or rejections. Moreover, the paper highlights the importance of data analysis and feature selection in constructing reliable credit models.

The research in [5] addresses the ethical dilemmas that arise when machines make decisions traditionally performed by human professionals. It is a qualitative content analysis, and hence analyses other papers and theoretical perspectives in a narrative style, summing up about 2,472 articles/publications. The paper explores issues such as transparency, accountability, bias, privacy, and the impact on professional judgment. This paper emphasizes the need to address potential biases

embedded in AI models. Lenders must ensure that their automated loan processing systems do not perpetuate biases related to factors such as gender, race, or socioeconomic background. Loan automation projects should integrate human expertise and ethical oversight to ensure that important contextual factors and ethical considerations are appropriately considered.

Also, by considering the Four-component model postulated in [6] a loan automation project can ensure a comprehensive approach to designing and implementing an effective, unbiased system. It emphasizes the importance of aligning people, procedures, data, and software to achieve the desired goals of streamlining loan processing, improving accuracy, enhancing decision-making, and increasing overall efficiency.

Examining existing literature on systems related to mine, I find a flaw that is existent in most, the automated loan processing ideas and systems proposed by them come from the perspective of the lenders (banks, investors, loan sharks, and the government) and not the borrowers. This project proposes to create a system that enables borrowers to be fitted with loans that suit their credit profiles and risk capacity and provides affordable and convenient loan repayment plans as well as reminders to empower them to hold their own end of the lender-borrower contract resulting in better lender-borrower relationships, better credit profile for the borrower and improved banking stability for the banks.

Chapter 3

System Analysis and Design

3.0 Introduction

System analysis is a component of system development that deals with data collection and interpretation, problem identification, and breaking down a system into its constituent elements [15].

The researcher has observed that in the realm of commercial banking, the intricacies of loan processing pose distinct challenges to the seamless flow of operations. Factors such as application submissions, credit assessments, and loan approval delays necessitate a comprehensive analysis. To delve deeper into this, the researcher plans to meticulously dissect the proposed Automated Loan Processing System. By systematically analysing the system, the researcher aims to shed light on potential bottlenecks and inefficiencies, paving the way for a more streamlined and efficient approach to loan processing within commercial banks.

In the pursuit of revolutionizing loan processing within commercial banks, this project aims to construct an Automated Loan Processing System. Through a deep examination of the existing system's intricate parts, the researcher intends to streamline and enhance the efficiency of loan processing operations in Nigerian commercial banks via a mobile application.

System design is the process of planning a new system or modifying an existing one by specifying its components to fit the system's specific criteria. This section discusses how to achieve the system's stated objectives. To properly analyze and create a system, different software design methodologies, such as the one used in this project, must be used. The Object-Oriented Analysis And Design Methodology (OOADM) was employed in this project. This methodology

evaluates the system using the conceptual model by identifying the elements, their relationships, and behaviours [16]. For example, the Asset Model is used to construct an instance for the application database and is used by other processes in the code base whenever Asset instances are needed.

The benefits of the Object-Oriented Analysis and Design Methodology to the researcher are varied and important. These benefits include the ability to reuse program code through inheritance, the ability to arrange objects for enhanced maintainability, and the ability to impose limits on data and actions through information hiding. These combined aspects considerably improve the application's security and reliability.

For example, the code base's server package contains a number of classes that are responsible for fitting accounts to loan plans, verifying documents, and tracking accounts and their credit scores, among other things. In order to accomplish the purpose of object-oriented programming, these classes are utilized to generate instances through inheritance, and their methods are then implemented in various areas of the code base along with an adaptation of the information-hiding procedure.

The design phase describes these objects by creating class diagrams, which diagrammatically represent these objects' (class instances') characteristics, behaviours, and interactions. A system's data (fields) and processes (methods) are stored in objects (instances of classes.) Various technologies used to develop the system like React Native [8] have classes like the Event class which has members like onClick, and onMouseOver, and the State class with methods like useState and useContext as well as the JavaScript inbuilt Date class used to keep track of loan repayments dates.

System modelling is the process of creating system models, each of which represents a particular viewpoint on the system [17]. These models can be

modified to fulfil the aims of the system without having to build the full system in order to close the gaps between the problem and the solution.

The study underscores the difficulties faced by commercial banks in streamlining their loan processing systems using the chosen approach, including issues like the dynamic nature of loan approval timelines, alterations in administrative procedures, occasional oversights, loan repayments and more. Furthermore, the investigation delves into the various components and interfaces housing the operational functions (methods) of these processes.

3.1 Description of the Existing System

Analyzing the components and interactions within a specific system can be defined as system analysis. The prevailing approach used by commercial banks for loan processing is predominantly manual and paper-based, lacking a centralized point of access. This results in labour-intensive activities that are prone to errors [3], which leads to delays in loan approvals and extended wait times. In the context of loan processing, various departments handle different aspects of the process, each with its own set of procedures and documentation.

The loan officer is frequently burdened with the task of physically visiting different departments involved in the loan approval process. This manual effort is not only time-intensive but is also compounded by the geographical dispersion of these departments across the bank's premises, adding to the overall resource expenditure. Loan plans are also not as carefully fitted to customers by using algorithms introducing personal bias of the loan officer(s) in issuing loans.

Additionally, because this process is done manually and physically at the bank, customers are required to jot down their loan application details and submission deadlines, relying solely on their memory to keep track of these critical dates.

Upon closer analysis of the existing loan processing system, the following deficiencies were identified:

i.) Significantly prolonged duration for visiting multiple departments to complete loan processing.

ii.) Changes in loan application status or terms are often conveyed verbally, lacking proper documentation.

iii.) Loans are usually not personalised and suited to the borrower's risk appetite and capacity.

iv.) Loan repayments are difficult to navigate because the process is manual and comes with a lot of red tape leading to Non-Performing Loans (NLP's.)

v.) Banks must be visited physically in order for a loan to be obtained setting up a barrier for people living with disabilities from having access to loans and financial services.

3.2 Analysis of the Proposed System

The proposed system is a smartphone mobile application that allows borrowers to obtain loans suited to their credit profiles and risk capacity. The user can access the application through any device with access to the internet, once it connects to the online database where the real-time data is tracked and the user logs in it shows the user's account information and loan plans suited to them.

The user experience and user interface of the system must be streamlined. The system greets the user by presenting them with a login screen where they are asked to provide their login information and, if they are a first-time user, the option to register with their email, employment information, and identification documents.

Once the user is signed in they are guided by alerts to learn how to navigate the application and understand the user flow, they can see their initial credit score

based on their employment information, ongoing loans (which will be none for a new user) as well as loan plans fitting their credit score which they can request. This proper guidance leads to the application being highly accessible and usable by users. The user can also increase their credit score by updating their job (yearly salary) and asset information (these assets are also used as collateral for loan repayment.)

The proposed system has several benefits compared to the existing systems which include:

i.) It is easily accessible from any location, and the borrower does not need to visit the physical bank to obtain a loan.

ii.) It fits borrowers to loan plans using algorithms based on their risk capacity and earnings without the bias of a loan officer coming into play.

iii.) The proposed system would be a mobile application, allowing for more mobility and ease of usage.

iv.) Repayment is encouraged as defaulting accounts are suspended from requesting any more loan plans and it can affect their credit scores. Users are also reminded via push notifications when it is time to pay an instalment.

A typical technique to represent the design of a system is provided by the Unified Modelling Language (UML), a general-purpose, developmental modelling language used in the field of software engineering [18]. UML offers a diagram that contains features like activities (jobs), system components and their interactions, user interface, and more to provide a visual representation of a system's architectural blueprints. To further analyze the suggested system, which includes use case diagrams, class diagrams, and activity diagrams, a tool for the Unified Modeling Language (UML) would be employed.

3.2.1 Use Case Diagram

Use case diagrams are dynamic representations of behaviour that use actors and use cases to show how a system works [18]. In this context, a system is a growing program, and the actors are the individuals or objects that take on different roles within the system. This diagram shows the various system members and their contributions to the accomplishment of the system's objectives. The activities that are essential to the system's success are shown in the use case diagram. The Use Case diagram for the system is displayed in Figure 3.1.

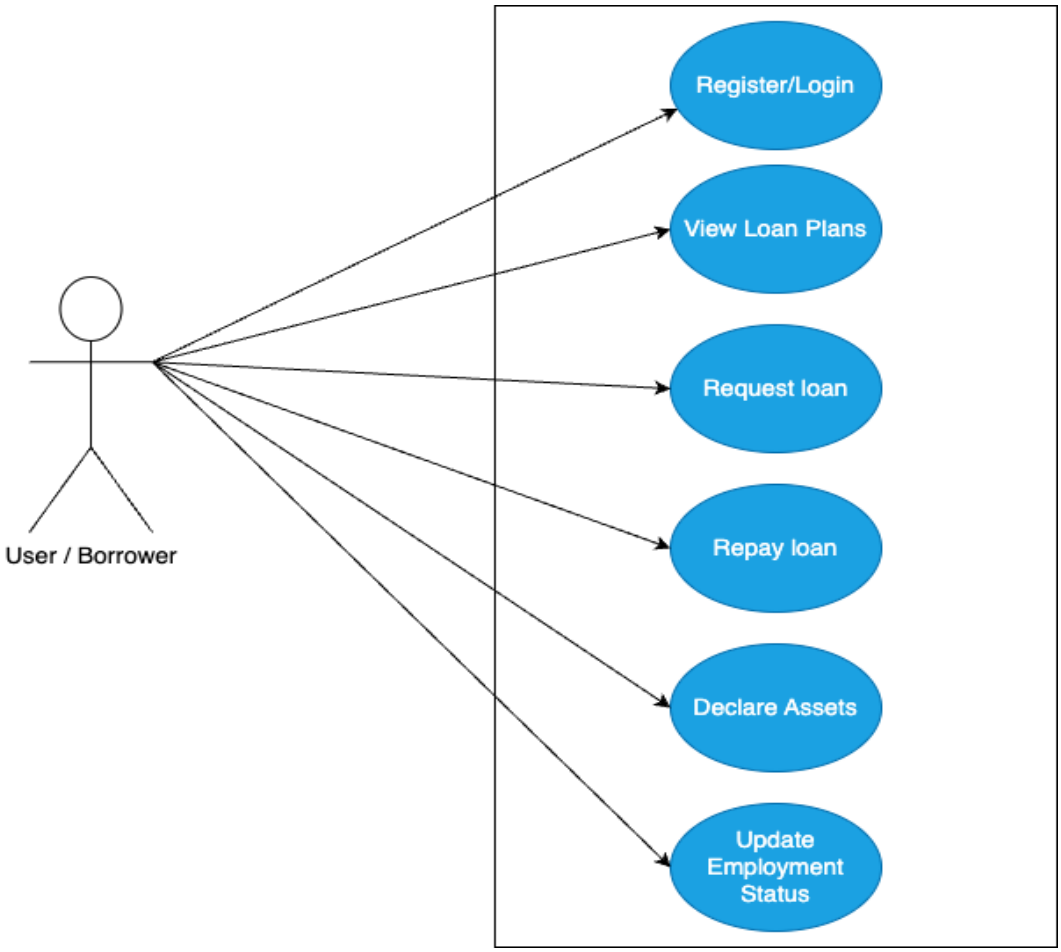

Figure 3.1: Diagram Showing the Use Case of the Proposed System

3.2.2 Class Diagram

A class diagram is a UML tool used to show all of a system's major parts and their connections [19]. A class diagram depicts a group of classes, interfaces, affiliations, and limitations. The classes in the diagram are as follows:

Account Class: This denotes the major actors in the system, namely the Borrower/User.

LoanPlan Class: These are the loan options available to the borrower/user based on their credit score.

Loan Class: Denotes and is used to keep track of the loans obtained by the system’s users.

FrequencyType Class: Denotes how often the borrower must make instalments, it is also used to show the duration of the loan.

RepaymentHistory Class: Denotes the tracking/history of all the user’s repayments/instalments paid.

Asset Class: Denotes the user’s declared assets used to increase their credit score.

AssetType Class: Denotes the type of the assets declared by users.

Job Class: Describes the employment information provided by users.

JobType Class: Denotes the type of employment from a fixed number of jobs.

Document Class: Denotes supporting documents.

DocumentType Class: Denotes the type of document uploaded by the user.

The Class diagram for the system is shown in Figure 3.2 below.

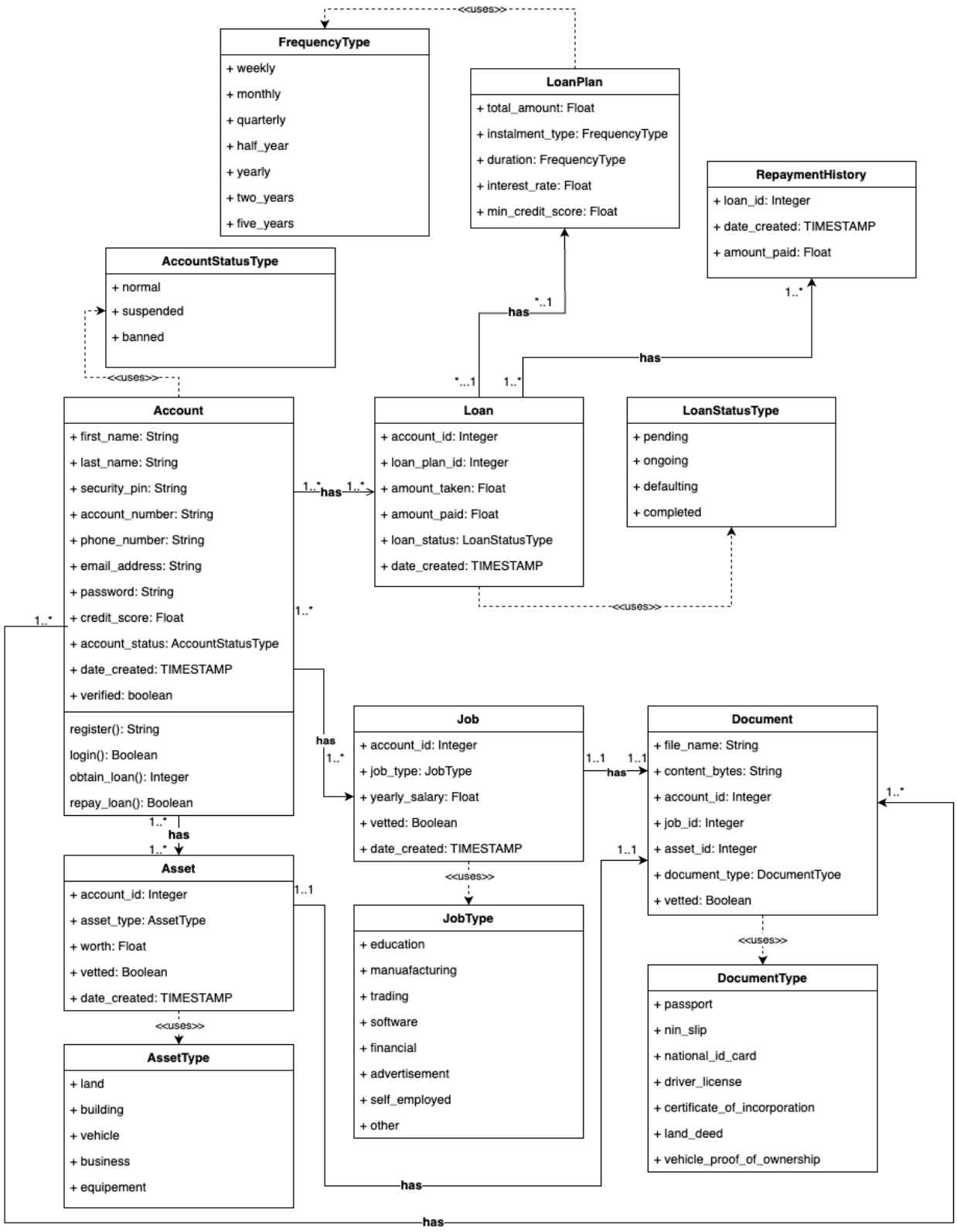

Figure 3.2: Class diagram of the proposed system.

3.2.3 Activity Diagram

The activity diagram is yet another crucial UML diagram for outlining the dynamic properties of the system. In essence, it is a flowchart that shows how information moves from one activity to the next. A system operation might be used to describe the action. Each actor in the system has its own activity diagram since each actor can do actions. The main actor in this system is the User, who represents the borrower.

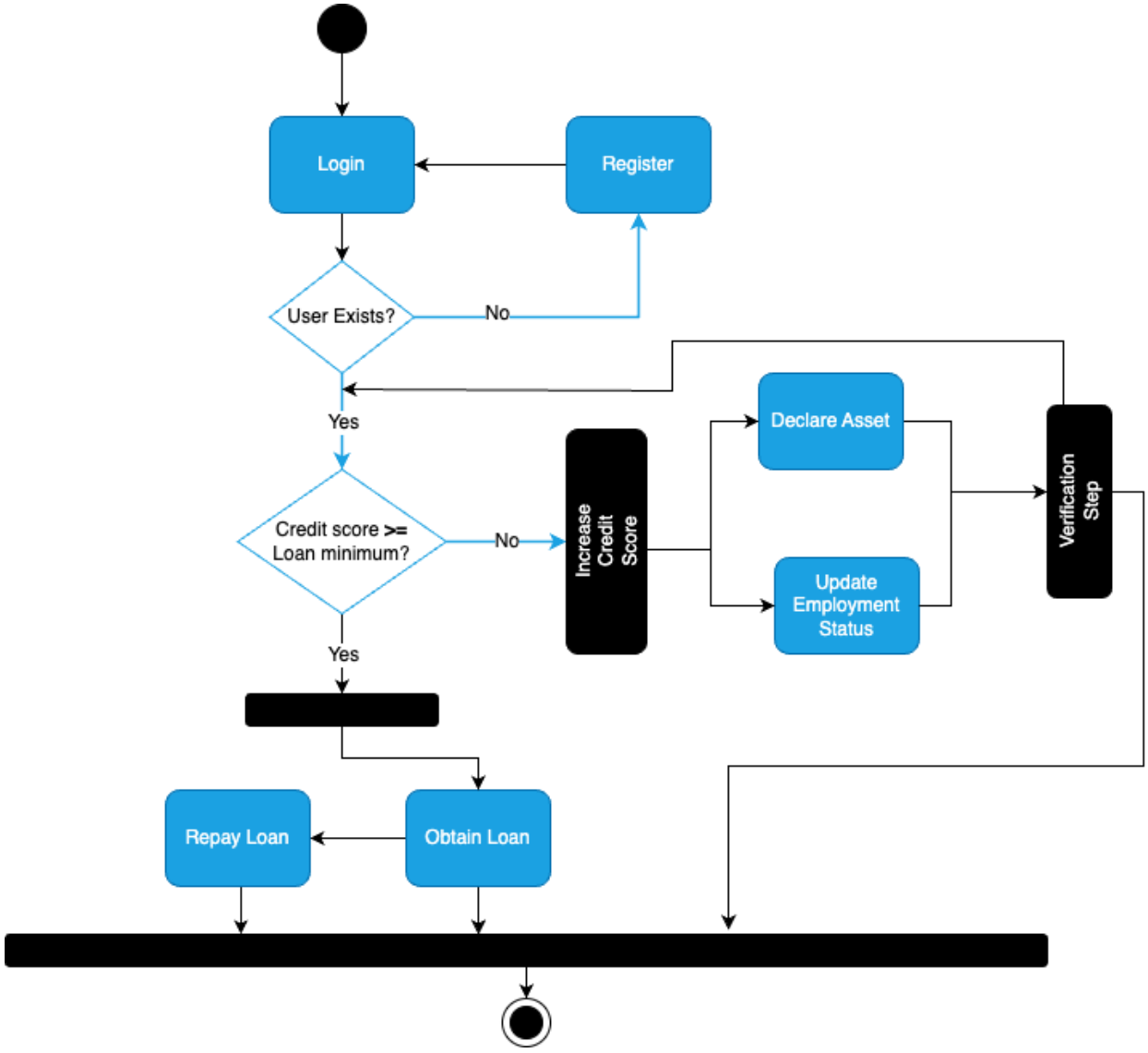

Figure 3.3: Activity Diagram of the Proposed System.

3.3 Design of the Proposed System

System design is the process of developing a system's architecture, modules, and components as well as the numerous interfaces that connect them to the data that flows through them [15].

In order for the implementation to be compatible with the architectural entities listed in the system architecture models and views, the system design process must provide sufficient particular data and knowledge about the system and its components. The following issues will be covered in this part, which describes how the system will operate:

- Database Design
- Input Design
- Output Design
- System Architecture

3.3.1 Database Design

Organizing data based on database models is referred to as database design in the context of PostgreSQL. In this process, the researcher determines the necessary data to be stored and establishes how different data components interact with each other. For the purpose of managing the database in this project, PostgreSQL, a relational database management system, was employed.

The researcher considered various storage options, such as shared preferences and internal/external storage. However, PostgreSQL was chosen as the optimal solution due to its comprehensive and self-contained relational database engine. Setting up the database is hassle-free as Android comes equipped with built-in support for PostgreSQL functionality. The database entities utilized in this research encompass.

Tables Defined in the Database:

Account: This represents a user of this application.

Loan: This represents a loan taken by a user on this application.

LoanPlan: This represents a loan a user can view and request on the application.

RepaymentHistory: This represents a repayment of a loan made by a user on the application.

Asset: An asset or thing of value added by a user to increase their credit score or serve as collateral.

Job: A job position added by a user on the platform.

Document: Asset and job employment documents for verification.

Account Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
account_number	BigInt	10	No	User 10 digit account no		
email_address	String	255	No	User email		
password	String	255	No	User password		
first_name	String	50	No	User first name		
last_name	String	50	No	User last name		
nin	BigInt	11	No	User NIN		

security_pin	String	4	No	User 4 digit pin	Change	
phone_number	String	13	No	User phone number	Change	
credit_score	Float	10	No	User credit score	Change	
account_status	AccountStatusType	11	No	User account status	Change	
verified	Boolean	5	No	Is User account verified	Change	
date_created	TIMESTAMP	50	No	Date user created account		

LoanPlan Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
name	String	255	No	Descriptive name for loan	Change	
total_amount	Float	150	No	Loan amount dtaken	Change	
duration	FrequencyType	50	No	How long loan is to held for	Change	
instalment_type	FrequencyType	50	No	How often one should pay the loan	Change	
interest_rate	Float	10	No	Interest rate to be paid (%)	Change	

min_credit_score	Float	10	No	Minimum credit score needed to take loan	Change	

Loan Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
account_id	Int	11	No	Id referencing Account Table		
loan_plan_id	Int	11	No	Id referencing LoanPlan table		
amount_taken	Float	150	No	Loan amount taken in Naira		
amount_paid	Float	150	No	Loan amount paid in Naira	Change	
loan_status	LoanStatusType	10	No	Status of the loan	Change	
date_created	TIMESTAMP	50	No	Date the loan was obtained		

RepaymentHistory Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment

loan_id	Int	11	No	Id referencing Loan Table		
amount_paid	Float	150	No	How much was repaid		
date_created	TIMESTAMP	150	No	When repayment happened		

Job Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
account_id	Int	11	No	Id referencing Account Table		
yearly_salary	Float	150	No	Yearly salary in Naira		
job_type	JobType	255	No	Type of job		
vetted	Boolean	5	No	Is job verified	Change	
date_created	TIMESTAMP	150	No	When job is created		

Asset Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
account_id	Int	11	No	Id referencing Account Table		

asset_type	AssetType	255	No	How much was repaid		
worth	Float	150	No	Worth of asset in Naira	Change	
vetted	Boolean	5	No	Is asset verified	Change	
date_created	TIMESTAMP	150	No	When asset is declared	Change	

Document Table

Field	Data Type	Size	Nullable	Description	Action	Extra
id	Int	11	No	Unique id		Auto_increment
file_name	String	255	No	Name of file		
content_bytes	String	65535	No	Relative path of file in device storage		
document_type	DocumentType	255	No	Type of document		
account_id	Int	11	No	Id referencing Account Table		
job_id	Int	11	Yes	Id referencing Job Table		
aseet_id	Int	11	Yes	Id referencing Asset Table		
vetted	Boolean	5	No	Is file verified	Change	
date_created	TIMESTAMP	150	No	Date document is uploaded		

Types Used in the Above Tables:

AccountStatusType: This is an enum of Strings that can take only values of ‘normal’, ‘suspended’, or ‘banned’.

FrequencyType: This is an enum of Strings that can take only values of ‘weekly’, ‘monthly’, ‘quarterly’, ‘half_year’, ‘yearly’, ‘two_years’, or ‘five_years’.

LoanStatusType: This is an enum of Strings that takes only values of ‘pending’, ‘ongoing’, ‘defaulting’, or ‘completed’.

AssetType: This is an enum of Strings that takes only ‘land’, ‘building’, ‘vehicle’, ‘business’, ‘equipment’, or ‘other’.

JobType: This is an enum of strings that takes ‘education’, ‘manufacturing’, ‘trading’, ‘software’, ‘financial’, ‘advertisement’, ‘self_employed’, or ‘other’.

DocumentType: This is an enum of strings that takes values of ‘passport’, ‘nin_slip’, ‘national_id_card’, ‘driver_license’, ‘certificate_of_incorporation’, ‘land_deed’, or ‘vehicle_proof_of_ownership’.

3.3.2 Input Design

The application's input architecture is shown in the input design, where the input is the data or information that the system is given for processing. This is accomplished by utilizing the React Native Text User Interface Library. Two of the inputs to the proposed system are registration and asset upload forms. These forms are depicted below.

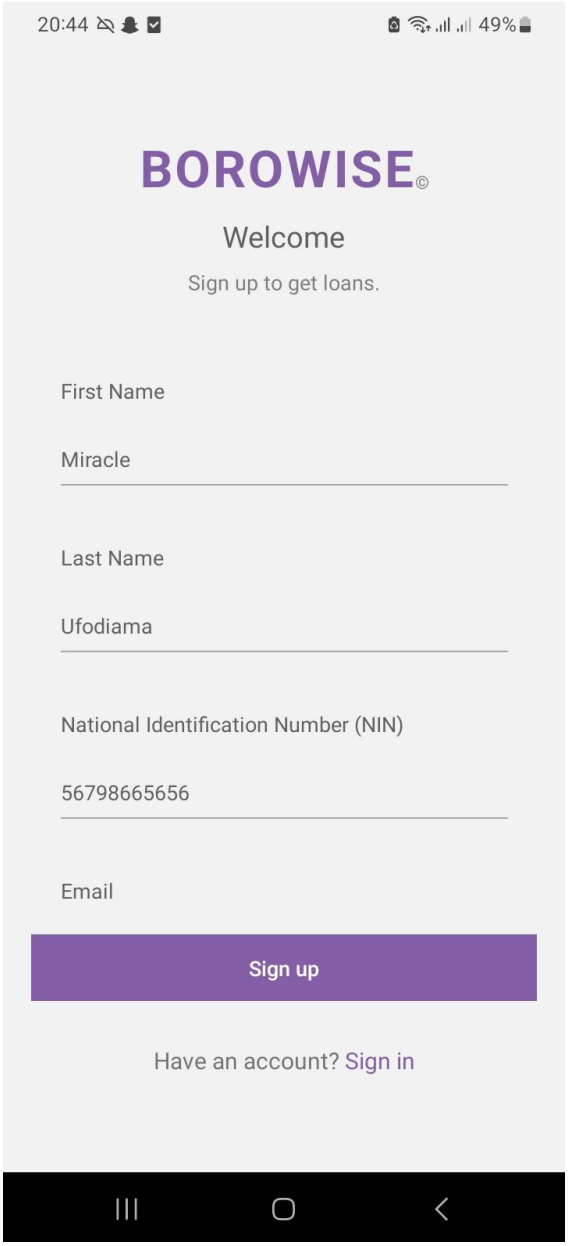

Figure 3.4: Input Design of the Proposed System using the Registration Form.

3.3.3 Output Design

The application's output display, which displays the loan plans and account information like credit score, is shown in the output design. The Text, Views, and Buttons used are from React Native's User Interface Library. The system's output, which includes the outcome of input processing, is what the system creates.

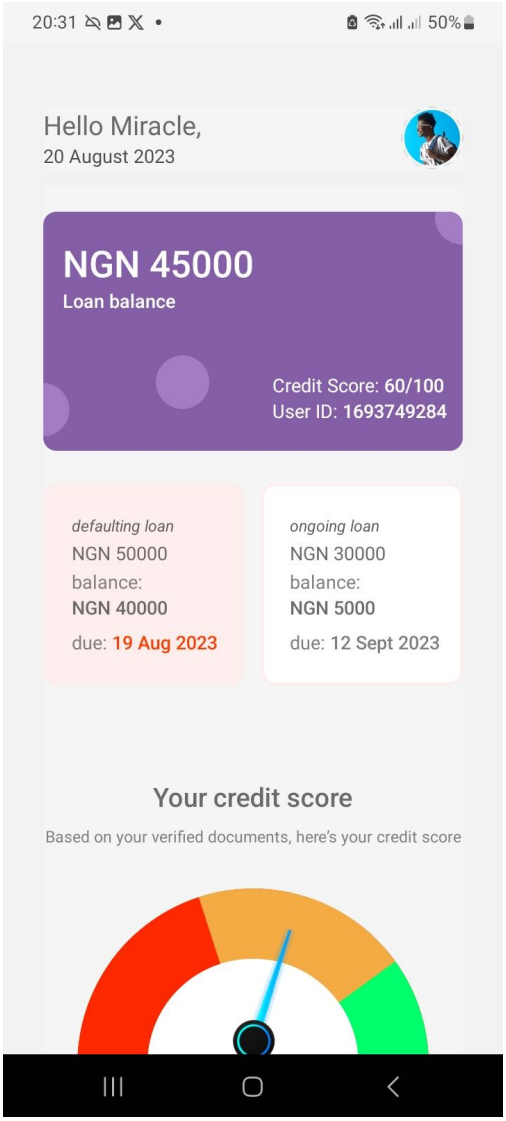

Figure 3.5: Output design of the proposed system showing the output of credit score.

3.3.4 System Architecture

The suggested method employs a three-tiered structure. the middle or business-logic tier, which is driven by the JavaScript programming language and runs the React Native Engine, the data or database server tier, and the presentation tier, which is made up of the mobile phone and its operating system.

The operating system of the user's mobile phone powers the user interface. The data tier is responsible for keeping the necessary data, while the presentation tier is linked to the middle tier's business logic. The architecture of the suggested system is shown in Figure 3.6 below.

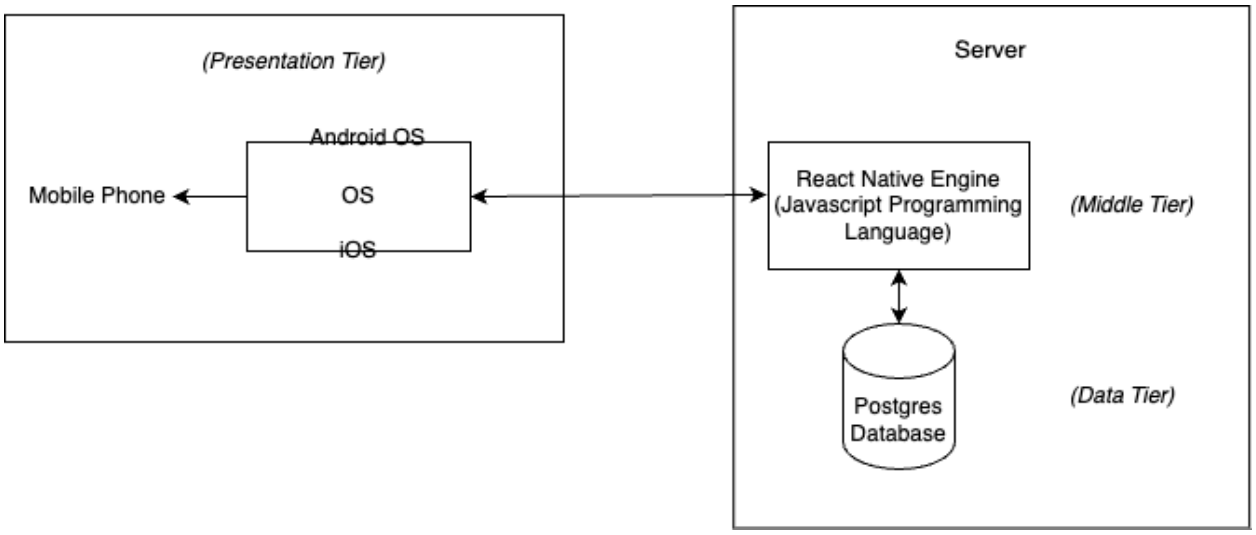

Figure 3.6: System Architectural Diagram of the Proposed System.

Chapter 4

System Implementation

4.0 Introduction

System implementation is the process of making a new system available to a specific group of users (also known as deployment.) It is a process that includes the development of the system, ongoing system support, and continuous maintenance [20]. Support and maintenance are important in order to guarantee that the information system is operational and meets the quality requirements (commonly referred to as quality assurance.)

This chapter discusses how the proposed system will be implemented and the prerequisites that must be satisfied to implement it. Additionally, it includes information about the architecture that was used to implement it, the software testing that was conducted, and the development environment that was chosen.

4.1 Choice of Development Environment

A development environment, also referred to as an integrated development environment (IDE), is a software development environment that enables programmers and software developers to work more efficiently by combining various tasks involved in software development into a single program. This environment includes tools for editing source code, creating executables, highlighting and completing syntax, and, most importantly, debugging, all within one environment. This project was developed within the Visual Studio IDE which provides linting and debugging capabilities during the development phase.

The proposed Automated Loan Processing System will be developed using React Native [8], a highly preferred choice for cross-platform mobile development

which works on both Android and iOS mobile devices. This decision was made due to the remarkable benefits that React Native brings to the table, enhancing the efficiency of the entire development journey while minimizing the required time investment. Noteworthy features of React Native include its robust code editor tailored for development, seamless integration with Postgres for database management, an advanced and adaptable build system, and a compatibility tool ensuring optimal performance across various device versions. React Native stands as the forefront environment for creating exceptional applications tailored for a wide range of platforms.

JavaScript, which is the default language for building React Native applications, was utilised in the development of the application. The JavaScript programming language was used for the application's logic and the React Native framework was used for the implementation of the user interface.

4.2 Implementation Architecture

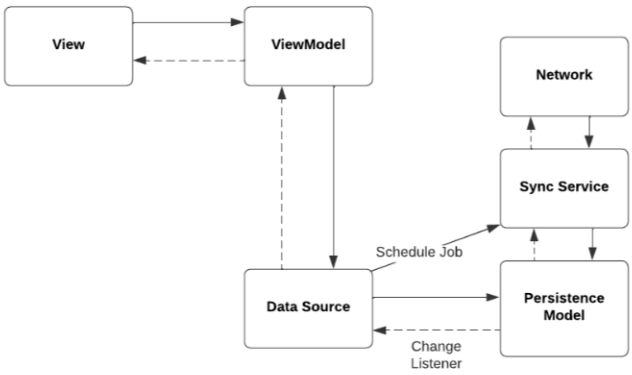

Figure 4.1: Implementation architecture of the proposed system.

The proposed system's implementation architecture describes the system's physical design as well as how it should be developed. The Automated Loan Processing program is in charge of a number of duties, which it completes with the aid of its essential parts. The accompanying diagram demonstrates how the system interacts with a variety of various components, each of which serves a specific purpose and plays a unique role.

The application offers aid in successfully exploring its View when the user is welcomed to the system. When the user is welcomed, they are guided on the next steps by alerts and intuition since the flow of the application is designed to be very intuitive. The system's ViewModel, which is always connected to the Application Logic, automatically redirects data that a user enters into the User Interface (UI). The persistent logic then interacts with this application logic, which causes it to write the user sign-in token as well as other non-sensitive data to the device disk. As a result, the View is refreshed to reflect updated information.

4.3 Software Testing

Software testing is the process of assessing a software program's functionality, and its main goal is to determine whether the generated software satisfies the specifications that were initially outlined.

Software testing is done to see if the finished product truly complies with the specifications that were initially set forth for the program. Additionally, it finds software problems so that they can be identified and fixed in order to improve the system's overall performance. Both human and automated testing methods are available.

For the purposes of this project, the application was tested at each level of development to identify and correct any faults as soon as possible. The examination was carried out in two stages:

- During the project's development phase, the first level of testing was performed with the purpose of eliminating difficulties caused by logical and semantic errors. Whenever the codebase was changed, the program was put through a series of tests to ensure that the output met the specifications.
- The application's accuracy, user interface, and user flow were validated during the second round of testing, which was performed on a number of Android devices with varying screen sizes and versions of the Android Software Development Kit (SDK) and on the Expo app.

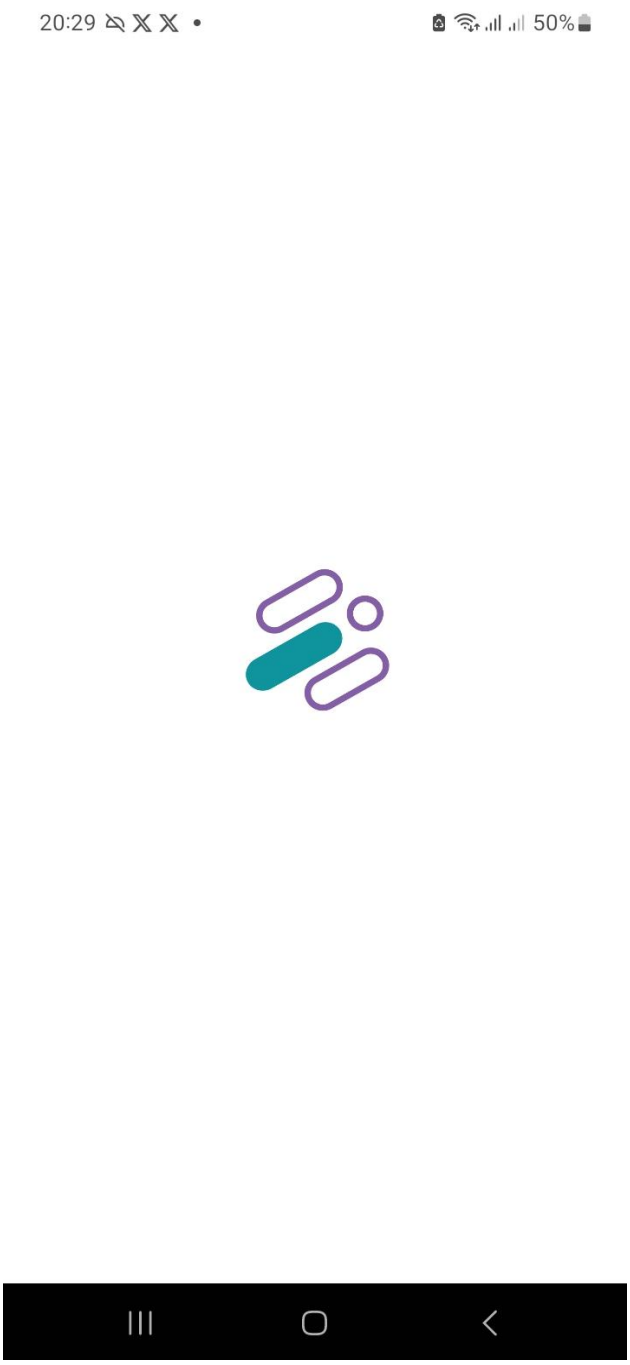

Figure 4.2: A screenshot of the splash screen of the application.

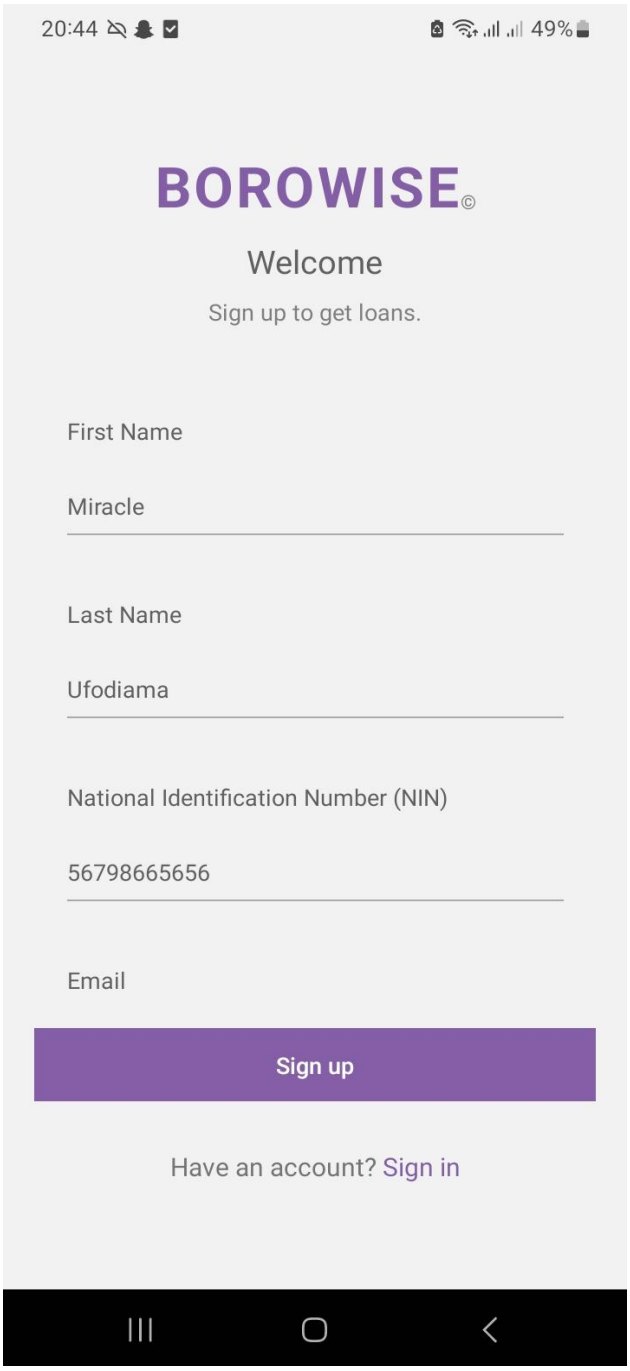

Figure 4.3: A screenshot of the account creation screen of the application.

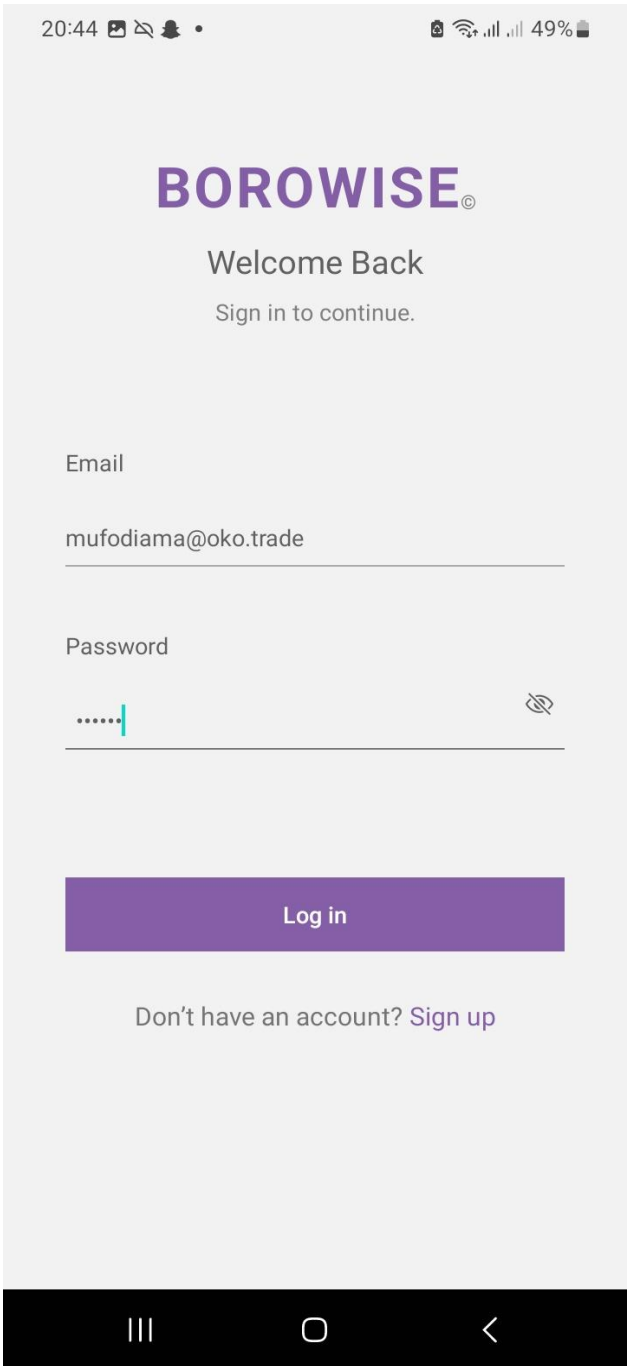

Figure 4.4: A screenshot of the sign-in screen of the application.

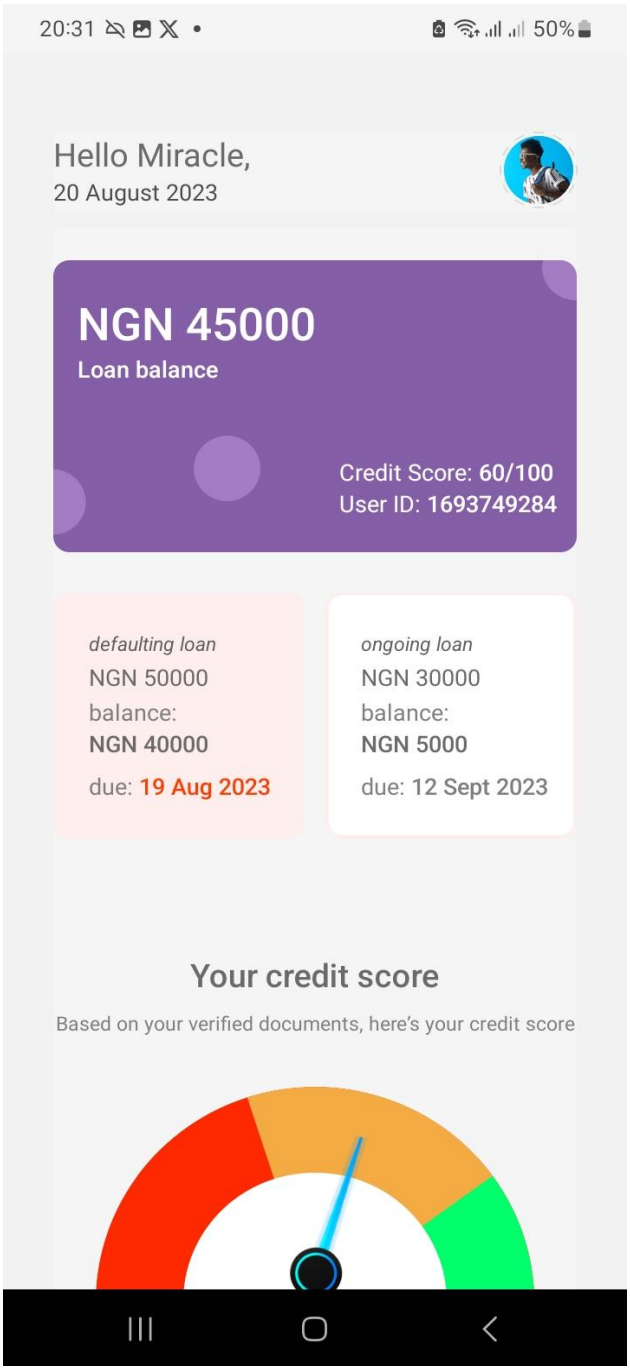

Figure 4.5: A screenshot of the home screen of the application.

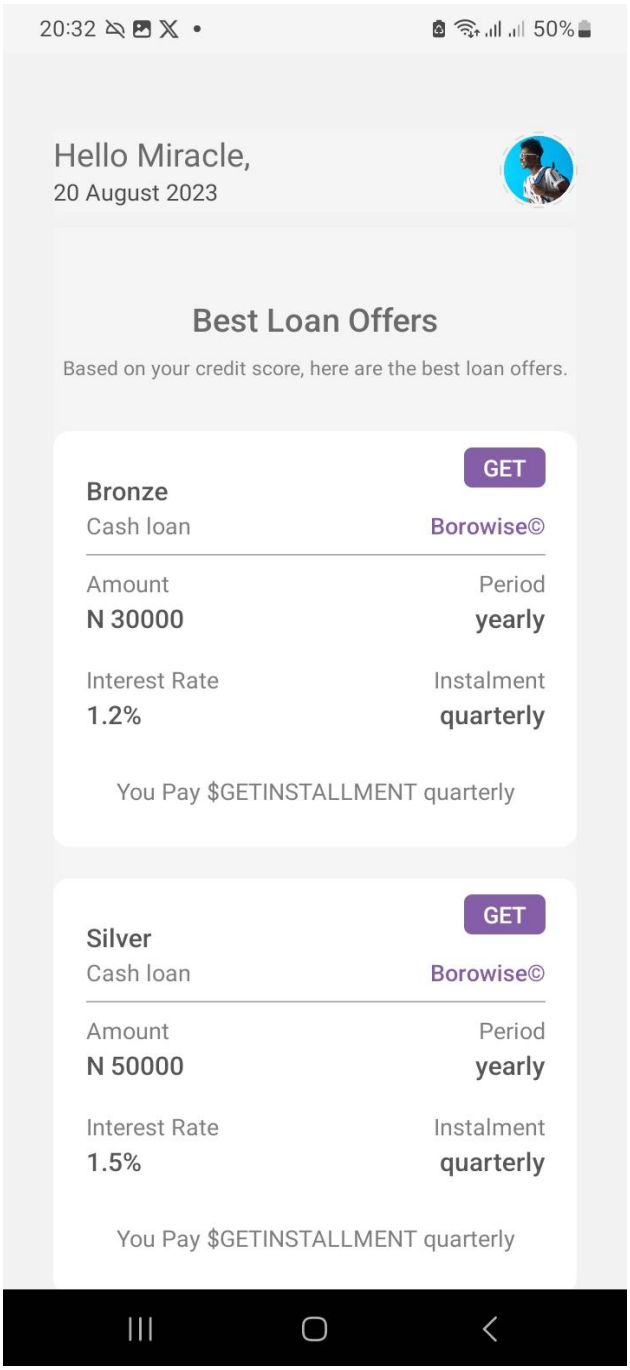

Figure 4.6: A screenshot of the loans screen of the application.

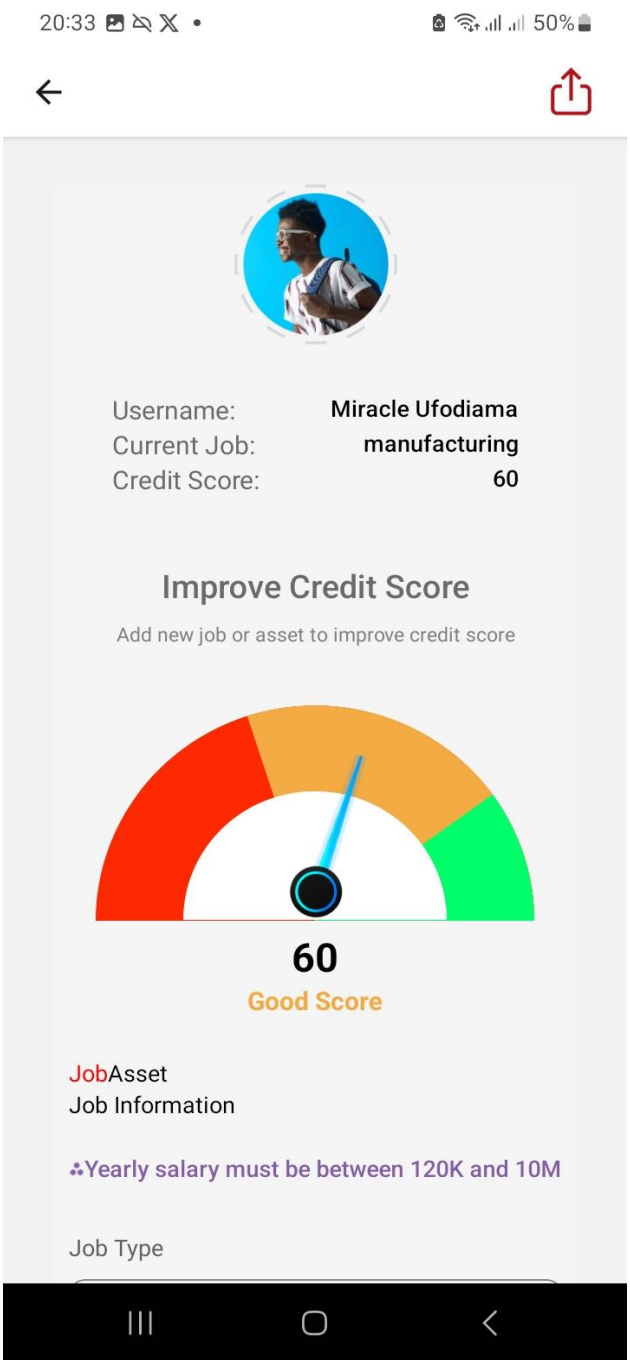

Figure 4.6: A screenshot of the user's profile screen.

4.4 Documentation

Software Documentation is a written text or image that accompanies or is present in the source code of computer software. The documentation's job is to explain how the software works or how it should be used to achieve its intended goal. The Software Documentation Process is an important aspect of the overall Software Development process.

The documentation of real programming components, such as algorithms and program codes, is known as "technical documentation." This portion of the project was completed successfully thanks to the use of comments. Comments are explanations or annotations in the source code of a computer program which are only visible to the program's developer and are skipped over during the compilation process.

Multi-line comments were used to describe large parts of code in this codebase, whereas single-line comments were used to explain minor sections of code. Both types of comments improve the code base's maintainability, readability, and comprehension.

4.4.1 User Manual

This project makes use of external documentation (documentation that is not within the source code) in the form of a user manual. This is a piece of technical documentation intended to act as a guide for anyone using the application.

The major goal of this program is to make it easier for students to track and manage the numerous activities that occur at school. The user guide, on the other hand, would be documented as a step-by-step way to engage with the application's capabilities as stated below.

- The user is welcomed with a series of onboarding screens and prompted to sign in or register if there's no existing account.
- The user is moved to the Home tab which has a menu showing their credit score and the loans available to them based on their credit score. It also shows their ongoing loans if any.
- The user can now choose to request a new loan or repay an existing one.
- The user can switch to the Profile tab where the account’s job and asset details can be updated and logout initiated. Also here the user can increase their credit score by declaring assets or updating their job positions.

4.4.2 SOURCE CODE LISTING

The source code listing for this project work can be found in the Appendix.

Chapter 5

Summary and Conclusion

5.0 Summary

The Automated Loan Processing Application introduces a revolutionary mobile experience aimed at simplifying and expediting the loan application process. This innovative Android application transforms the way users interact with loans, making it a seamless and efficient journey from start to finish.

By harnessing advanced automation technology, the application eliminates the need for cumbersome paperwork and manual submissions. This app enables users to submit loan applications digitally, saving time and reducing the complexities associated with traditional loan procedures.

Designed with user convenience in mind, the application offers an intuitive interface that can be easily navigated by users of all technological backgrounds. This user-centric approach ensures that managing loan applications becomes effortless and accessible, right from the users' iOS and Android devices.

The onboarding process is smooth and hassle-free, requiring users to sign up for an account within the app. During this process, users can upload non-sensitive financial information in order to be given a credit score reflective of their risk capacity. This tailored approach ensures that the app's services align with each user's unique needs.

Keeping users informed throughout the loan application journey is a priority. The app provides informative screens and guides to achieve this, as well as notifications. These timely notifications enhance user trust by providing transparency and clarity throughout the process.

Currently, the essential components of the Automated Loan Processing Application have been successfully integrated. These components encompass the incorporation of loan requests and processing workflows. React Native's Text Input component, renowned for its capabilities, was harnessed to establish a streamlined mechanism for inputting loan application specifics. Displaying and managing this data was seamlessly executed through the utilization of React Native's View and FlatList component, harmonized with layout management techniques, thus furnishing users with comprehensive insights into loan requests and processing statuses.

For data persistence, the React Native AsyncStorage was adeptly employed to manage local storage needs, while Postgres served as the robust cloud storage solution. As a final touch, the removal of data entries was realized by skillfully deploying the removeItem() function from React Native's AsyncStorage package.

5.1 Conclusion

In conclusion, the Automated Loan Processing App is a transformative solution that reshapes the loan management landscape. By integrating automation, user-friendly design, and instant communication, the app simplifies and accelerates loan processing. Users can now leave behind the complexities of traditional loan applications, embracing a solution where loans are accessible and efficient.

The current methods of loan processing are designed from the lender's perspective and do not adequately consider the needs of the borrower. This can lead to longer processing times, increased risk of errors, and lack of transparency.

An automated loan processing application can help to address these problems by automating the loan application process, providing borrowers with real-time updates, and making the loan process more transparent. This can benefit

both lenders and borrowers by saving time and money, and getting the money they need more quickly and easily.

In addition, an automated loan processing application can help to improve the accuracy of loan decisions by using artificial intelligence and algorithms to assess a borrower's creditworthiness more accurately. This can lead to fewer loans being denied, and more borrowers being able to access the credit they need.

Overall, an automated loan processing application can be a valuable tool for lenders and borrowers alike. It can help to speed up the loan process, reduce the risk of errors, and make the loan process more transparent.

5.2 Recommendation

Throughout the course of this project, this mobile application has been able to solve a small portion of the challenges that confront lenders and borrowers in today's rapidly evolving financial and technological landscape.

It would be beneficial to conduct additional research on the subject of adding more features, such as making it possible for administrators to regulate user behaviour on the application in a seamless manner. Additionally, additional work needs to be done to investigate better options that would enable these people to have a smooth and secure procedure of logging in with relation to the input of their passwords.

Borrowers would be able to use the majority of these functions on their Android devices without the need for additional assistance if they followed the steps outlined in the User Manual section in System Implementation.

REFERENCES

[1] N.V. Atoi, "Non-performing Loan and its Effects on Banking Stability: Evidence from National and International Licensed Banks in Nigeria," *CBN Journal of Applied Statistics*, vol. 9, no. 2, 2018, pp. 43–45.

[2] Thorsten Beck, Augusto de la Torre, "The Basic Analytics of Access to Financial Services", *World Bank Policy Research Working Paper*, 2006, pp. 1–5.

[3] Doug Peterson, "Maximize Efficiency: How Automation Can Improve Your Loan Origination Process", *Moody's Analytics, Inc.*, 2018. [Online]. Available: https://www.moodysanalytics.com/articles/2018/maximize-efficiency-how-automation-can-improve-your-loan-origination-process. [Accessed: 06-Mar-2023].

[4] J.B. Obayori, C.C. George-Anokwuru, "Financial Inclusion and Economic Growth in Nigeria", *Business Perspective Review*, vol. 2, no. 2, 2020, pp. 46–56.

[5] O.M. Lehner, Kim Ittonen, Hanna Silvola, Eva Ströom, and Alena Wührleitner, "Artificial intelligence-based decision-making in accounting and auditing: ethical challenges and normative thinking", *Accounting, Auditing & Accountability Journal,* vol. 35, no. 9, 2022, pp. 109–126.

[6] Four-component model definition, *www.accountingtools.com,* 2023. [Online] Available: https://www.accountingtools.com/articles/four-component-model. [Accessed: 20-May-2023]

[7] S. Sachan, J.B. Yang, D.L. Xu, D.E. Benavides, and Y. Li, "An Explainable AI Decision-Support-System to Automate Loan Underwriting," *Expert Systems with Applications*, vol. 144, 2022, pp. 1–42.

[8] "React Native (software) - Wikipedia", *En.wikipedia.org*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/React_Native. [Accessed: 15-June-2023].

[9] "Tailwind CSS (software) - Wikipedia", *En.wikipedia.org*, 2023. [Online]. Available https://en.wikipedia.org/wiki/Tailwind_CSS. [Accessed: 15-June- 2023].

[10] "Introduction to Styled-Components in React - Better Programming", *www.betterprogramming.pub*, 2021. [Online]. Available: https://betterprogramming.pub/introduction-to-styled-components-in-react-d84583c28dde. [Accessed: 18-June- 2023].

[11] "SQL (software) - Wikipedia", *En.wikipedia.org*, 2023. [Online]. Available:

https://en.wikipedia.org/wiki/SQL. [Accessed: 18-June- 2023].

[12] "About Nodejs - Nodejs", *nodejs.org*, 2023. [Online]. Available: https://nodejs.org/. [Accessed: 18-June- 2023].

[13] "What is PostgreSQL? - Postgres", *postgresql.org*, 2023. [Online]. Available: https://www.postgresql.org/about/.[Accessed: 20-June- 2023].

[14] "What are Azure Cognitive Services? - Microsoft Learn", *learn.microsoft.com*, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services. [Accessed: 20-June- 2023].

[15] "System Analysis and Design: Overview", *Tutorials Point*, [Online]. Available: *https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_overview.htm* [Accessed: 22 January 2022]

[16] Omar Elgabry, "Object-Oriented Analysis And Design - Conceptual Model (Part 2)", blog, 19 March. 2017; https://medium.com/omarelgabrys-blog/object-oriented-analysis-and-design-conceptual-model-part-2-ce730ac4eb31

[17] "*CS 410/510 - Software Engineering class notes*", [Online]. Available: https://cs.ccsu.edu/~stan/classes/CS410/Notes16/05-SystemModeling.html#:~:text=System%20modeling%20is%20the%20process,Unified%20Modeling%20Language%20(UML) [Accessed: 9 February 2022]

[18] "*UML: Use Case Diagrams",* [Online]. Available: https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm [Accessed: 15 April 2022]

[19] Nishadha, "*UML: Class Diagram Relationships", blog, [Online]*. Available: https://creately.com/blog/diagrams/class-diagram-relationships/ [Accessed: 15 April 2022]

[20] John Snoderly, Alan Faisandier, "*System Implementation", blog, [Online]*. Available: https://www.sebokwiki.org/wiki/System_Implementation [Accessed: 2 June 2022]

APPENDIX

```
require('dotenv').config({path: './.env.local'});

const express = require('express');
const cors = require('cors');
const { Pool } = require('pg');

const app = express();

app.use(cors());
app.use(express.urlencoded({ extended: true }));


const PORT = 8080;

// Create a connection pool to the PostgreSQL database
const pool = new Pool({
host: process.env.PGHOST,
database: process.env.PGDATABASE,
user: process.env.PGUSER,
password: process.env.PGPASSWORD,
port: 5432,
ssl: false,
});

// Middleware to parse JSON requests
app.use(express.json());

// Route to handle POST requests and proxy them to the database
app.post('/sign-in', async (req, res) => {
try {
const { email, password } = req.body.params;
console.log(req.body, email);
if (!req.body || !email || !password) {
return res.status(500).json({ error: 'Email and/or password not present in the
request body'});
}
const { rows } = await pool.query(`SELECT * FROM account WHERE email_address =
'${email}' and password = '${password}'`);
```

```
console.log('SIGNED IN', rows);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route /sign-in'});
}
});
app.post('/sign-up', async (req, res) => {
try {
const {
accountNumber,
email,
password,
firstName,
lastName,
nin,
phoneNumber,
securityPin,
creditScore,
} = req.body.params;

if (!req.body || !email) {
return res.status(500).json({ error: 'Email is not present in the request body'});
}

const accountStatus = 'normal';
const verified = false;
const dateCreated = new Date().toISOString();

const { rows } = await pool.query(`
INSERT INTO account(first_name, last_name, security_pin, account_number,
phone_number, credit_score, account_status, verified, date_created, email_address,
password, nin)
VALUES('${firstName}', '${lastName}', '${securityPin}', '${accountNumber}',
'${phoneNumber}', '${creditScore}', '${accountStatus}', '${verified}',
'${dateCreated}', '${email}', '${password}', '${nin}') returning *`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route /sign-up'});
}
```

```
});
app.post('/add-job', async (req, res) => {
try {
const { accountId, yearlySalary, jobType, vetted } = req.body.params;
if (!req.body || !accountId || !yearlySalary || !jobType) {
return res.status(500).json({ error: 'Job is not correctly specified'});
}

if (!vetted) {
return res.status(500).json({ error: 'Job is not vetted'});
}
const dateCreated = new Date().toISOString();
const { rows } = await pool.query(`
INSERT INTO job(account_id, yearly_salary, job_type, vetted, date_created)
VALUES('${accountId}', '${yearlySalary}', '${jobType}', ${vetted}, '${dateCreated}')
`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route /add-job'});
}
});
app.get('/verify/:id', async (req, res) => {
try {
console.log(req);
const { id } = req.params;
if (!req.params || !id) {
return res.status(500).json({ error: 'Not verified.'});
}
const { rows } = await pool.query(`UPDATE account SET verified = true WHERE id =
'${id}' returning *`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route /verify'});
}
});

app.get('/get-loan-plans-by-credit/:minCreditScore', async (req, res) => {
try {
const { minCreditScore } = req.params;
```

```
if (!req.params || !minCreditScore) {
return res.status(500).json({ error: 'Minimum credit score not sent.'});
}
const { rows } = await pool.query(`SELECT * FROM loan_plan WHERE min_credit_score <=
${minCreditScore}`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-loan-plans-by-credit'});
}
});

app.get('/get-incomplete-loans-by-account/:accountId', async (req, res) => {
try {
const { accountId } = req.params;
if (!req.params || !accountId) {
return res.status(500).json({ error: 'Account id not sent.'});
}
const { rows } = await pool.query(`SELECT * FROM loan WHERE account_id =
${accountId} and loan_status != 'completed'`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-incomplete-loans-by-account'});
}
});

app.post('/add-loan', async (req, res) => {
try {
const { amountTaken, accountId, loanPlanId } = req.body.params;
console.log('BOOH', amountTaken, accountId, loanPlanId);

console.log(req.body.params)

if (!req.body || !req.params || !amountTaken || !accountId || !loanPlanId) {
return res.status(500).json({ error: 'Correct parameters not sent not sent.'});
}
const amountPaid = 0;
```

```
const dateCreated = new Date().toISOString();
const loanStatus = 'ongoing';
const { rows } = await pool.query(`
INSERT INTO loan(amount_taken, amount_paid, loan_status, account_id, loan_plan_id,
date_created)
VALUES('${amountTaken}', '${amountPaid}', '${loanStatus}', ${accountId},
${loanPlanId}, '${dateCreated}')
RETURNING id;
`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-incomplete-loans-by-account'});
}
});

app.post('/repay-loan', async (req, res) => {
// add to repayment history
// calculate and add payment to existing amountPaid in loan
});

app.get('/get-assets-by-account/:accountId', async (req, res) => {
try {
const { accountId } = req.params;
if (!req.params || !accountId) {
return res.status(500).json({ error: 'Account id not sent.'});
}

const { rows } = await pool.query(`SELECT * FROM asset WHERE account_id =
${accountId}`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-incomplete-loans-by-account'});
}
});

app.get('/get-current-job/:accountId', async (req, res) => {
try {
```

```
const { accountId } = req.params;

if (!req.params || !accountId) {
return res.status(500).json({ error: 'Account id not sent.'});
}

const { rows } = await pool.query(`SELECT * FROM job WHERE account_id = ${accountId}
ORDER BY date_created DESC LIMIT 1;`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-incomplete-loans-by-account'});
}
});

app.post('/add-job', async (req, res) => {
try {
const { yearlySalary, accountId, jobType } = req.body.params;
console.log('ADD JOB', req.body.params);

if (!req.body || !req.params || !yearlySalary || !accountId || !jobType) {
return res.status(500).json({ error: 'Correct parameters not sent not sent.'});
}
const dateCreated = new Date().toISOString();
const vetted = true;

const { rows } = await pool.query(`
INSERT INTO job (account_id, yearly_salary, job_type, vetted, date_created)
VALUES(${accountId}, ${yearlySalary}, ${jobType}, ${vetted}, '${dateCreated}')
RETURNING id;
`);
return res.json(rows);
} catch (error) {
console.error(error);
return res.status(500).json({ error: 'Internal server error on route
/get-incomplete-loans-by-account'});
}
});

app.listen(PORT, () => {
```

```
if (pool) console.log(`Server is running on port ${PORT}`);
});

app.use((req, res, next) => {
next('404RouteNotFound. You\'re falling off the earth 🌍.');
});

module.exports = app;

import FontAwesome from '@expo/vector-icons/FontAwesome';
import { DefaultTheme, ThemeProvider } from '@react-navigation/native';
import { useFonts } from 'expo-font';
import {
SplashScreen,
Stack,
} from 'expo-router';
import { useEffect } from 'react';

import { AuthProvider } from '@context/auth';
import LogoutButton from '@containers/LogoutButton';

export {
// Catch any errors thrown by the Layout component.
ErrorBoundary,
} from 'expo-router';

// eslint-disable-next-line @typescript-eslint/naming-convention
export const unstable_settings = {
// Ensure that reloading on `/modal` keeps a back button present.
initialRouteName: '/modal',
};

// Prevent the splash screen from auto-hiding before asset loading is complete.
SplashScreen.preventAutoHideAsync();

function RootLayoutNav() {
return (
<ThemeProvider value={DefaultTheme}>
<AuthProvider>
<Stack>
<Stack.Screen name="(tabs)" options={{ headerShown: false }} />
<Stack.Screen name="modal" options={{ presentation: 'modal', title: '', headerRight:
```

```
() => (<LogoutButton icon />) } } />
</Stack>
</AuthProvider>
</ThemeProvider>
);
}

export default function RootLayout() {
const [loaded, error] = useFonts({
// eslint-disable-next-line global-require
SpaceMono: require('../assets/fonts/SpaceMono-Regular.ttf'),
...FontAwesome.font,
});
// Expo Router uses Error Boundaries to catch errors in the navigation tree.
useEffect(() => {
if (error) throw error;
}, [error]);

useEffect(() => {
if (loaded) {
SplashScreen.hideAsync();
}
}, [loaded]);

if (!loaded) return null;

return <RootLayoutNav />;
}

import axios from 'axios';
import { BASE_URL } from '@constants/index';

export const getAssetsForUser = async (accountId: string | number | undefined) => {
if (accountId) {
const id = Number(accountId);
const { data } = await axios.get(`${BASE_URL}/get-assets-by-account/${id}`);
// eslint-disable-next-line no-console
console.log('Assets for account id', data);
if (data?.length > 0) {
return data;
}
```

```
return [];
}
return [];
};

// for profile display
export const getCurrentJob = async (accountId?: string | number) => {
if (accountId) {
const id = Number(accountId);
const { data } /*: { data: LoanFromDB } */ = await
axios.get(`${BASE_URL}/get-current-job/${id}`);
// eslint-disable-next-line no-console
console.log('Current job', data);
if (data?.length > 0) {
return data;
}
return [];
}
return [];
};

import axios from 'axios';
import { BASE_URL } from '@constants/index';

export const getLoansForCreditScore = async (
creditScore: string | number | undefined,
) => {
if (creditScore) {
const minCreditScore = Number(creditScore) * 100;
// eslint-disable-next-line no-console
console.log(`${BASE_URL}/get-loan-plans-by-credit/${minCreditScore}`);
const { data } = await
axios.get(`${BASE_URL}/get-loan-plans-by-credit/${minCreditScore}`);
if (data?.length > 0) {
return data;
}
return [];
}
return [];
};
```

```
export const getIncompleteLoans = async (
accountId: number | undefined,
) => {
if (accountId) {
const id = Number(accountId);
const { data } = await
axios.get(`${BASE_URL}/get-incomplete-loans-by-account/${id}`);
// eslint-disable-next-line no-console
console.log('Incomplete loans', data);
if (data?.length > 0) {
return data;
}
return [];
}
return [];
};

export const addLoan = async (
amountTaken: number | string | undefined,
loanPlanId: number | undefined,
accountId: number | undefined,
) => {
if (amountTaken && loanPlanId && accountId) {
const { data } = await axios.post(`${BASE_URL}/add-loan`, {
params: {
amountTaken: Number(amountTaken),
loanPlanId,
accountId,
},
});

if (data?.length > 0) return data;
return [];
}
return [];
};

const calculateCreditScore = (yearlySalary: number, totalAssetsWorth: number) => {
// Define maximum values for salary and assets
const maxSalary = 10000000; // 10 million Naira
const maxAssetsWorth = 100000000; // 100 million Naira
```

```
// Calculate normalized values (between 0 and 1)
const normalizedSalary = Math.min(yearlySalary / maxSalary, 1);
const normalizedAssetsWorth = Math.min(totalAssetsWorth / maxAssetsWorth, 1);

// Weighted factors (adjust these as needed)
const salaryWeight = 0.7;
const assetsWeight = 0.3;

// Calculate the credit score
const creditScore = (
normalizedSalary * salaryWeight + normalizedAssetsWorth * assetsWeight
) * 100;

// Ensure the score is within the 0-100 range
const finalCreditScore = Math.min(Math.max(creditScore, 0), 100);

return finalCreditScore;
};

export default calculateCreditScore;
```