UPSA

UNIVERSITY OF PROFESSIONAL STUDIES, ACCRA

Scholarship with Professionalism

FACULTY OF INFORMATION TECHNOLOGY AND COMMUNICATION STUDIES

DEPARTMENT OF INFORMATION TECHNOLOGY STUDIES

UNDERGRADUATE WORK

MICROFINANCE MANAGEMENT SYSTEM

BY

Caleb Okley Tetteh

Josephine Yirrah

Benedicta Richlyn Akita

May 2023

BLANK PAGE

UPSA

UNIVERSITY OF PROFESSIONAL STUDIES, ACCRA

Scholarship with Professionalism

MICROFINANCE MANAGEMENT SYSTEM

THIS PROJECT REPORT IS SUBMITTED TO THE DEPARTMENT OF INFORMATION TECHNOLOGY STUDIES OF THE FACULTY OF INFORMATION TECHNOLOGY AND COMMUNICATION STUDIES OF THE UNIVERSITY OF PROFESSIONAL STUDIES, ACCRA IN PARTIAL FULFILLMENT FOR A BACHELOR OF SCIENCE DEGREE IN INFORMATION TECHNOLOGY MANAGEMENT

MAY 2023

CANDIDATES' DECLARATION

We the undersigned declare that this project is our work and that no part of it has been submitted for assessment in another University for a Degree. We are confident that no one else plagiarized this project. All sources of information have however been acknowledged with due respect.

Name	**Student ID**	**Sign**	**Date**
Caleb Okley Tetteh	10283691	…….	……
Josephine Yirrah	10284039	…….	……
Benedicta Richlyn Akita	10273295	…….	……

SUPERVISOR'S DECLARATION

I declare that the preparation and presentation of this Dissertation conformed with the standards and guidelines on the supervision of the Dissertation established by the University of Professional Studies, Accra (UPSA).

Supervisor's Name: Dr. Ocquaye Elias Nii Noi

Sign……. Date…….

DEDICATION

This project denotes the conclusion of a period that began when we initially started our undergraduate studies and we would like to dedicate this project work to Almighty God for giving us the strength to study and work hard and for the determination that went into making our undergraduate degree a reality. We are also grateful to the lecturers, departmental officers, and Faculty of Information Technology and Communication Studies for investing their time and energy into our learning. We also want to show our appreciation to family and friends who provided endless support and encouragement, and to our colleagues who helped to make it possible for us to obtain this degree, we dedicate this work to you all.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to the Management of the University of Professional Studies, Accra for making it possible for students to gain knowledge in their respective programs of study throughout the four years of academics at the University.

Similar appreciation goes to Department of Information Technology Management researcher Dr. Ben Ocra who devoted his time to providing us with the opportunity to undertake this project by making sure each student has a thorough understanding of the project by providing us with the guidelines needed to complete the work successfully

Special thanks go to our supervisor Dr. Ocquaye Nii Noi for his invaluable guidance, good heart, and assistance that contributed to our understanding and development of the project work. His patience and encouragement have been instrumental in helping to develop both physically and professionally.

Finally, we wish to express our thankfulness to our family members who showed unconditional love and support as far as the completion of this course is concerned and provided for us financially.

ABSTRACT

Microfinance is a crucial tool for alleviating poverty and promoting economic growth, especially in developing countries. However, managing a microfinance institution can be a challenging task, especially when it comes to maintaining accurate records of customer accounts, and transactions, and generating reports for monitoring performance.

To address this challenge, we have developed a microfinance management system to aid the staff of microfinance institutions in their day-to-day operations. The system will assist staff in creating customer accounts, storing and managing account data securely, processing transactions, and generating reports for monitoring product performance. Additionally, customers will have the option to subscribe to SMS alerts for account activity notifications.

By providing a secure and efficient solution for managing microfinance operations, our system will help microfinance institutions to better serve their customer and achieve their goals of promoting inclusion and economic growth.

Keywords: Microfinance Management System, Microfinance Institutions, Microfinance Operations, and Economic Growth.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Microfinance has evolved as an economic development strategy aimed at assisting low-income individuals. This word refers to the provision of financial services to low-income customers, especially self-employed individuals. Savings, loan acquisition, and other associated services such as insurance are all examples of financial services. (Ledgerwood, 1998). In contrast to conventional financial institutions like banks, where the lender is primarily concerned with whether the borrower has sufficient collateral to cover the loan, many microfinance organizations are devoted to assisting business owners in becoming successful (Kagan, 2022). A study by Hulme and Mosley (1996) found that microfinance has helped women start their enterprises, boost their income, and raise their social position.

Hence, we consider that combining the usage of technology is critical in boosting the microfinance industry's growth and efficiency. Although many Microfinance institutions have realized this potential, the majority of microfinance institutions in Ghana are yet to integrate technology into their operations. As people who are enthusiastic about technology, we recognize the importance of technology in reaching new audiences and adopting good risk management, which leads to cost-effectiveness and sustainability. This chapter will highlight the background of the project, the problems some microfinance institutions are facing, and the need to design and implement a robust microfinance management system to help solve these problems.

1.2 Background of the study

Microfinance also known as microcredit is the offering of a variety of financial services including savings, insurance, and loans to the underprivileged to eradicate the poverty rate. (Addae-Korankye, 2020).

According to Microfinance Info (n.d), the origin of microfinance can be found as far back as the middle of the 19th century when theorist Lysander Spooner wrote about the advantages of small loans to farmers and entrepreneurs as a means of lifting people out of poverty. The word "Microfinance" first appeared in the 1970s when institutions like the Grameen Bank of Bangladesh founded by Mohammad Yunus began to pioneer and shape the current microfinance sector (Watkins,2018). The term microfinance has gained popularity in modern times.

Through this, other businesses started providing loans to the underprivileged population. Today, approximately 7000 microfinance institutions assist more than 16 million people worldwide according to the World Bank. According to Consultative Group to Assist the Poor (CGAP,2012) analysts, these small loans enable the creation of about 2.5 million new businesses. Globally, Microfinance has emerged as a significant liberating force in societies where the poor have to struggle against repressive social and economic conditions.

Ghana is no exception when it comes to providing financial services to individuals in society with a low-interest rate. Ghanaians have historically saved with individuals and groups and obtained loans from individuals and groups under the self-help philosophy to start businesses or engage in farming. Ramatu (2020) indicated that Microfinance institutions in Ghana are made up of numerous businesses that increase savings, promote investment, promote economic growth, and alleviate poverty through financing small and medium-sized businesses.

1.3 Problem statement

Management of the microfinance institution does not have an efficient system to store and keep track of the records which results in making difficult and inaccurate decisions. The lack of a

database system has also led to the duplication of data. Due to the manual process used, microfinance institutions cannot store and retrieve data efficiently. This results in the need for multiple copies of records to be kept leading to confusion and inaccuracies when trying to access important information. The lack of a proper database also makes it difficult to generate accurate reports and make informed decisions, resulting in lost opportunities and managing potential risks for the institution.

In addition, the lack of a system for managing leads to inefficiency in the operations of microfinance. Their operation is time-consuming and increases the workload of the employees leading to increase costs and decreased productivity. The inability to also keep track of the performance of each product makes it difficult to identify which areas need improvement and implement the necessary changes. This can lead to financial issues for the management of the microfinance institution. Similarly, to this, management may find it difficult to manage their portfolio efficiently and spot potential credit concerns if they are unable to precisely track the progress of susu applications or loan repayments.

1.4 Scope of the Project

The system will assist the microfinance institution's employees in creating staff accounts and customer accounts by securely storing and managing the account data in a database for the processing of service requests. The employees, as the main users of the system, will have access to all functionalities of the system including the transaction module for credit and debit of a customer account. The system will also aid in generating statement reports in either Excel or pdf to monitor product performance by displaying transaction activity for a specific day or period of days. Various account types can be created for customers with the institution offering more than one product for sign-up. The initial account which will be available for customers is the susu and savings accounts. The system will also feature a dashboard displaying the total number of customers and those with specific account types. During the account creation, the customer will have the option to enable SMS alerts for account activity notifications.

1.5 Limitations of the Study

Time is one of the limitations of the study because the system will require other functionalities and must be reliable. Properly creating all the functionalities and testing the Microfinance management system within the required timeframe will be challenging because of the project's schedule constraints. It is critical to thoroughly test the system to find and fix any potential problems before implementing the system to make sure it is of the highest caliber and fits the needs of the microfinance organization. It is possible that the system will not contain all the functions needed or will not work at the level that is expected if there is not enough time to thoroughly test it.

The study's budget is yet another limitation that will have an impact. To reduce the risk of a data breach, a robust and secure microfinance management system must be developed. To do this, there must be an adequate budget to dedicate cash for the purchase of the necessary resources. Controls will be necessary for the microfinance management system to ensure that there are no financial misstatements.

1.6 Objective of the study

1.6.1 General Objectives

The main objective of the system is to provide a robust secure and efficient system for the microfinance institution’s employees to manage customer accounts, including account creation, data storage and management, transaction processing, and generating reports for monitoring product performance. In addition, for customers to receive notification activity on their accounts.

1.6.2 Specific Objectives

1. To design a user management module to aid clients and employees to register and manage their accounts.
2. To implement an SMS notification system to notify customers of events.
3. To create a statement report module that generates a variety of reports on financial data.

4. To design and implement a database management system to securely store and manage customer data.
5. To design and implement a transaction module to aid employees in crediting and debiting customer accounts.
6. To implement an account module to show different types of products provided by the microfinance institution.
7. To implement a dashboard module to summarize the total number of customers, account types, etc.

1.7 Methodology

Extreme Programming (XP) is a method of software development that emphasizes teamwork, great application of programming skills, and clear communication, enabling us to achieve things we previously could not even fathom. Software development using the XP methodology takes risk into account as it is being developed. Utilizing brief development cycles, often known as iterations is one of the main ways that XP addresses development risk. For the team to tackle problems during the cycle, XP advocates for short-release cycles and plans with short tasks (Beck, 2000). The stages involved in Extreme Programming include Planning, Designing, Coding, Testing, and releasing modules.

1.8 Organization of study

Five (5) chapters will make up the study.

Chapter 1: This is the first chapter of the project which focuses on the study background, the general and specific objectives, the methodology which will be used, and the organization of the study.

Chapter 2: This chapter focuses on the literature review, which examines the current system and compares it with the suggested system.

Chapter 3: This chapter discusses the system design, and the requirements specification needed to run the system. It also includes the diagram representation of the system and tools used.

Chapter 4: This chapter focuses on the testing of the new system, implementation, and documentation.

Chapter 5: This is the final part of the project. It gives a comprehensive summary and recommendation of the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will go through the general background and context of the Microfinance management system. We will also look at the current condition of the field, including existing methods for managing microfinance operations. Finally, this study will enable us to evaluate and contrast the proposed Microfinance management system implementation with existing systems, highlighting its unique features and capabilities.

2.2 General Background of the Study

Microfinance management systems are computer programs that assist microfinance institutions in managing their financial and administrative tasks, such as loan origination, repayment tracking, client administration, and financial reporting. Recently, the usage of Microfinance management systems in microfinance has increased as microfinance institutions look to increase their efficacy and efficiency in serving their clients and attaining their social and financial goals.

According to Ashta (2015), the use of technology and information systems can help to present low-cost opportunities, and improving the operations of microfinance organizations can help create new opportunities and growth. Microfinance institutions having a computerized system will help to identify market opportunities, and also reach out to potential borrowers with low credit risk which will enable them to enhance the penetration of markets and improve their loan portfolio. Due to the use of technology, large banks have been able to expand over the years to become global institutions delivering a multitude number of products due to certain reasons such as (1) to achieve a reduction in operational cost by streamlining the process and eliminating manual interventions. (2) To also provide a better customer experience by offering

new products and services or by enriching the existing products and services. (3) To be able to respond to rapidly changing market conditions and regulatory environment (Ashta, 2015)

According to Alter et al. (2013), the use of technology in banking to deliver services directly to the poor in their communities can help to increase the reach and impact of microfinance. The use of information systems can help scale up the customers due to how easier it is to perform operations such as Loan acquisition, loan disbursement, repayment, etc as compared to the manual system. (Waterfield & Ramsing, 1998) indicates that the use of microfinance management systems in microfinance institutions is essential for generating reports for decision-making. This will help the management be able to find new trends and insights.

Due to the increasing number of people using the services of microfinance institutions, Ashta et al. (2010), indicated that software system scalability is an important requirement to handle high traffic levels without adversely affecting user experience. The use of technology can help prevent corruption by making information more transparent but also protecting the user's private data. In terms of organizational change, Information Technology has a positive impact on human resources and socio-organizational perspectives in microfinance
organizations by reducing the laborious and risky nature of jobs and increasing the professional skills among staff. (Badruddozza & Ramage, 2018). Ajah & Inyiama (2011), also argues that when microfinance institutions use IT in their business operation, it can greatly benefit microfinance institutions or banks by reducing the incidence of non-performing products such as Loan types and increasing the efficiency of the loan review process. By using systems, management can have access to valuable information that can help them make better decisions, it will also help them collect and process borrowers' credit information quickly by using automated credit risk management systems. This will provide microfinance institutions with a competitive advantage and lead to higher loan performance.

2.3 Review of Existing Systems

2.3.1 Manual System

Currently, the microfinance institution does not have an existing automated system to manage the business processes, etc. All their activities are done manually. This method entailed managing the operations of microfinance organizations using paper-based records and procedures. The usage of physical records like ledgers and paper forms was one of the key characteristics of a manual system. Financial transactions such as susu and savings were recorded in ledgers and later manually typed into an office productivity tool such as Microsoft Excel. Potential customers were asked to fill up paper applications with information about their next of Kin, personal details, etc. Although the manual system might have worked well for small-scale activities, it had several drawbacks.

Problems with the Manual System

- Paper application process is prone to errors and labor intensive.
- Manual systems are inadequate for managing high transaction volumes, lack security, and control, and pose challenges in maintaining customer confidentiality and preventing fraud.

2.3.2 Ezee Microfinance management system

The Ezee microfinance system developed by Wisdom Agbenu is a platform created to help microfinance institutions manage different processes including loan origination, loan distribution, and loan repayment. Additionally, it is designed to assist microfinance institutions in tracking and managing loan requests and approvals from their clients as well as producing reports. The system was created using VB.Net to lower the risk of fraud and mistakes compared to the manual way of handling requests from clients. The system has the following features client module where clients can be registered into the system, a Loan form that will enable clients to fill out and request a loan which will then be approved by the staff, and a report module that will generate reports to assist managers to make decisions. The system will also

allow the clients to withdraw money from their account which will debit and show the available balance in their account etc. But the system has several flaws that might harm the efficiency and user experience.

Problem with Ezee Microfinance Management System

- The system does not notify clients through SMS about the progress of their loan requests which results in missed deadlines or other problems and inconveniences.
- The usage of four message boxes after a client completes the entry of their information results in a bad user experience and negatively impacts the business.
- One significant drawback of the system is its unfriendly user interface which is complex, and poorly organized, and makes it difficult to find the desired information. In particular, the client registration interface of the Ezee system calls for entering a lot of data which might be overwhelming for users and increases the risk of data entry errors.

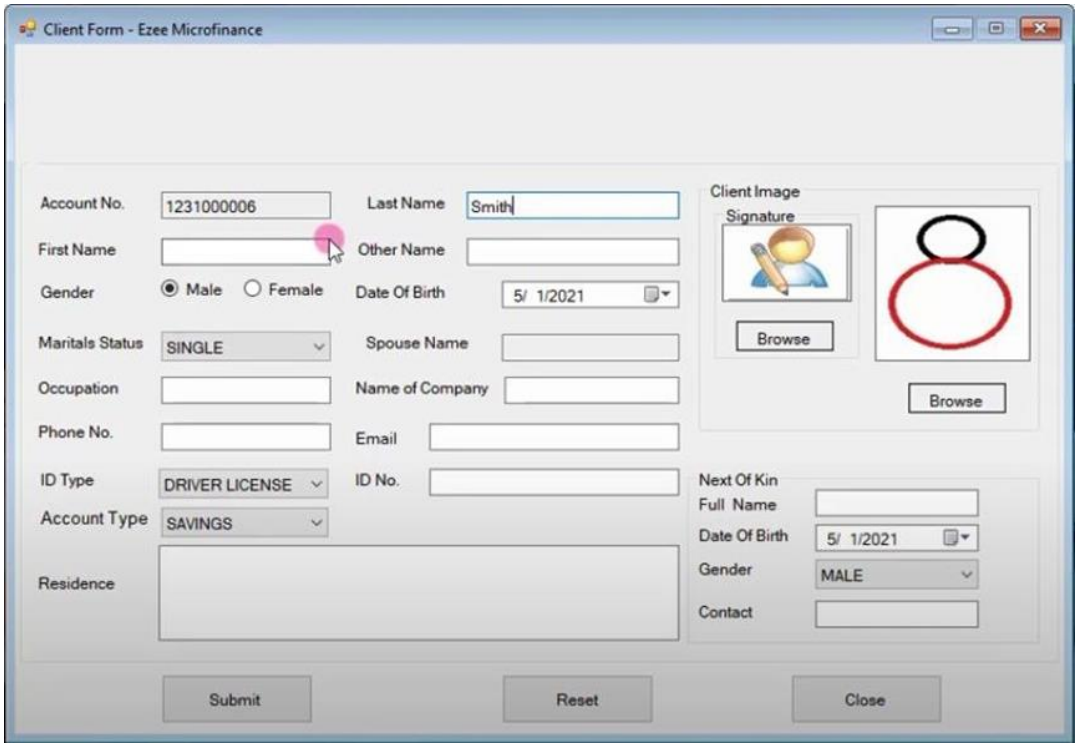


Figure 2.3.2 ***Ezee client registration form***

2.3.3 LetsGo

LetsGo is a financial application powered by LetsheGo. This application allows customers of LetsheGo around the Pan-African countries such as Ghana, Nigeria, Namibia, etc to manage their finances. The customers of LetsheGo can access products and services at any time wherever they are. The features of the system include a user management module that enables customers to manage their account profile and also create an account with LetsheGo after a user has registered into the system. The system also allows customers to pay bills, send and receive funds, add funds, and view their transaction history. The LetsGo app also provides products such as Loan acquisition whereby the customer can request loans and track the progress of the Loans requested. Aside from the Loan acquisition, they also provide the following products in the system, LetsGo insures, wellbeing, etc. The system has a notification module that notifies the users. The app has also been integrated with all types of contact platforms which aid the customers to reach out to them without needing to visit their office.

Problem with LetsGo

The system keeps on spamming the user with the “welcome on board” notification every time there is a click on a module. This makes it annoying as user experience is a concern.

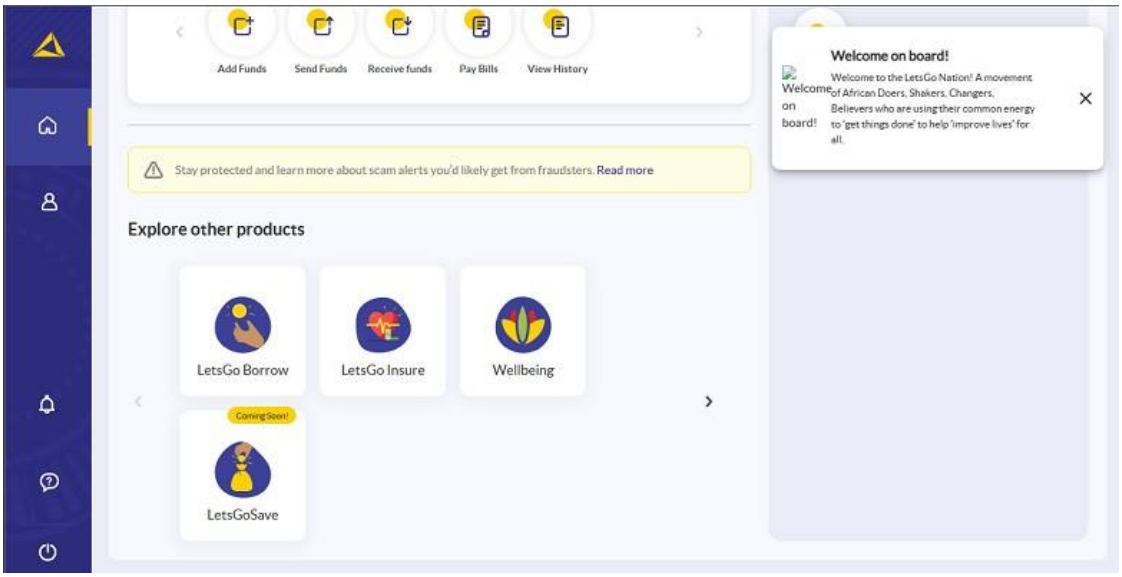


Figure 2.3.3 *LetsGo*

2.3.4 The Proposed System

Due to the rapid advancement of technology, everything will need to be done online. In this project, technology will be used in the field of microfinance so that employees can create staff accounts and also create customer accounts. The system will be hosted on the web to make it easier for the field operation team of the microfinance to also register customers remotely. It will prevent the susu team from carrying a ledger book to record transactions performed in the field of work. The system will have an SMS service alert to notify customers of activities on their account such as money credited or debited from their susu or savings account. This will lessen the money that microfinance institutions waste on ledger books and paper forms and lower the possibility of incorrect data values. During the account creation for customers, the system will require the agent or employee to provide their names. This will be done to assist management to identify which employees are working hard to bring in customers. The system will have a statement report module that will help generate reports on transactions for a specific day or period to assist management make informed decisions. Implementing input controls to stop the system from accepting incorrect data values for a text field will ensure data entering the system is accurate. The proposed system would have an extremely user-friendly interface that makes it simple for staff to interact with the system.

2.4 Comparative Study of Reviewed System

In several industries, including microfinance, there has been an increase in demand for systems that are more effective and efficient in recent years. As a result, several platforms and technologies have been created to assist firms in managing their operations and providing better client service. However, choosing the right system for a certain business can be difficult given the wide range of solutions available. Comparing and contrasting the characteristics, capabilities, and limits of several systems can yield insightful information about the reviewed systems. The proposed system and the existing systems are the two systems being compared

in this essay. We aim to provide a more knowledgeable perspective on each system by examining its advantages and disadvantages.

The table below shows the comparative study of the reviewed systems.

Existing systems	Proposed system
The paper data entry process leads to duplication of data.	Less paperwork because all necessary information will be entered directly into the system and is less prone to errors due to input controls.
Ledger books are inadequate for managing high transaction volume of data, lack security, and control, and pose challenges in maintaining customer confidentiality.	Customer information will be securely stored in a database to maintain customer information confidential.
The system does not notify clients through SMS about the progress of their loan requests which results in missed deadlines or other problems and inconveniences.	The provision of an SMS module will aid in notifying clients about the progress of loans and other related information.

Table 2.4 Comparative study of the reviewed system

2.5 Conclusion

Several areas require improvement after reviewing the existing system and according to the comparative study of the existing systems. So, the existing system should be upgraded or changed for a more effective system that can accommodate the demands of microfinance institutions.

CHAPTER 3

LIFE CYCLE DESIGN OF THE PROPOSED SYSTEM

3.1 Introduction

Presented in this chapter is the proposed system's life cycle design. In addition to the analysis and system design, this also refers to the problem's crystallization. Along with the hardware and software requirements for the system, we will also cover the system's requirements for its features and functions in this chapter. We will also give flow chart diagrams, context diagrams, entity relationship diagrams, dataflow diagrams, and use case diagrams so that you can understand how the system is designed. In addition, we will talk about the tools utilized in the system design.

3.2 Crystallization of the Problem

The problem is that the current management system of a microfinance institution is manual and inefficient, leading to inaccuracies in data storage, retrieval, and reporting, as well as difficulties in making informed decisions and managing operations. The proposed system is to implement a web-based system that utilizes technology to automate record-keeping, allows for remote work, and includes SMS alerts and a user-friendly interface. This aims to improve efficiency, accuracy, and decision-making while reducing costs and workload for employees.

3.3 Analysis and Design of the System

3.3.1 System Requirements

System requirements are the qualities and limitations that a system must meet to perform properly. They outline the hardware, software, and other resources that the system needs to run successfully and efficiently within a specific environment. The table below shows the system requirements of the system.

Operating System version & architecture	Processor	Memory	Screen Resolution
Windows 8.0 64-bit or later	Intel core i3 or later	4 GB minimum, 8 recommended or more	1024x680 or larger
macOS 10.12 Sierra or later	Apple M1 or later		
Linux 64-bit	AMD A-Series or later		

Table 3.1 System Requirements

3.3.2 Function Requirements

Functional requirements define the actions or features that a system should be able to accomplish. They are critical for defining the system's behavior and capabilities. Prioritizing functional requirements is critical for ensuring that the system meets the needs and expectations of users. The following are the functional requirements of the proposed system;

- **User Management module:** The system will allow employees to register and create staff accounts and also register customers with the microfinance institution and provide the necessary personal information. The employees can manage their accounts such as changing passwords and other related information.
- **SMS alert service:** The system will provide SMS notifications to customers regarding their account balance, and money credited or debited on their accounts.
- **Transaction Module:** The transaction module is responsible for ensuring that transactions are processed correctly and atomically, meaning that they are either completed in full or not at all.
- **Statement Module:** The statement module will also provide the functionality to export the financial statement in different file formats, such as PDF or Excel.

- **Data Entry and Validation form:** It will allow the employees to input and submit data into the system and the validation feature will validate the data before it is stored in the database.
- **Account module:** The account module will list all different products such as susu or savings accounts offered by the microfinance institution and which products a customer can sign up for.
- **Dashboard module:** The dashboard module of the system will summarize the total number of customers registered with the microfinance, and the number of customers signed up for savings or susu accounts.

3.3.3 Non-function Requirements

Non-functional requirements are system features that do not correspond to a specific function or action that the system is capable of performing. They determine the system's overall characteristics and qualities. The non-function requirement of the microfinance management systems includes;

- **Security:** By encrypting data and authenticating users, the system will be able to safeguard client data and guarantee the confidentiality of sensitive information.
- **Maintainability:** The system will be easy to maintain and update over time with minimal disruptions.
- **Reliability:** The system will operate consistently without experiencing unexpected errors and failures.
- **Usability:** The system will have a user-friendly interface that will be easy to use for employees.
- **Performance:** The system will be able to handle a high volume of data without glitches or making it unavailable.

3.3.4 Hardware Requirements

The physical components and devices that a system needs to work effectively are referred to as hardware requirements. The specifications outline the system's hardware, such as CPUs, memory, storage, and input/output devices. The table below shows the hardware requirements for the proposed system.

Processor	Intel or AMD 1GHz minimum 2GHz recommended or more.
Memory	DDR3 or later, 4GB minimum, Recommended 8GB or more.
Ethernet	Wired (LAN) or Wireless (wi-fi)
Hard disk	HDD or SSD 50GB minimum, recommended 256GB or more.
Input	Keyboard and Mouse.
Output	Monitor or any display system.

Table 3.2 Hardware Requirements

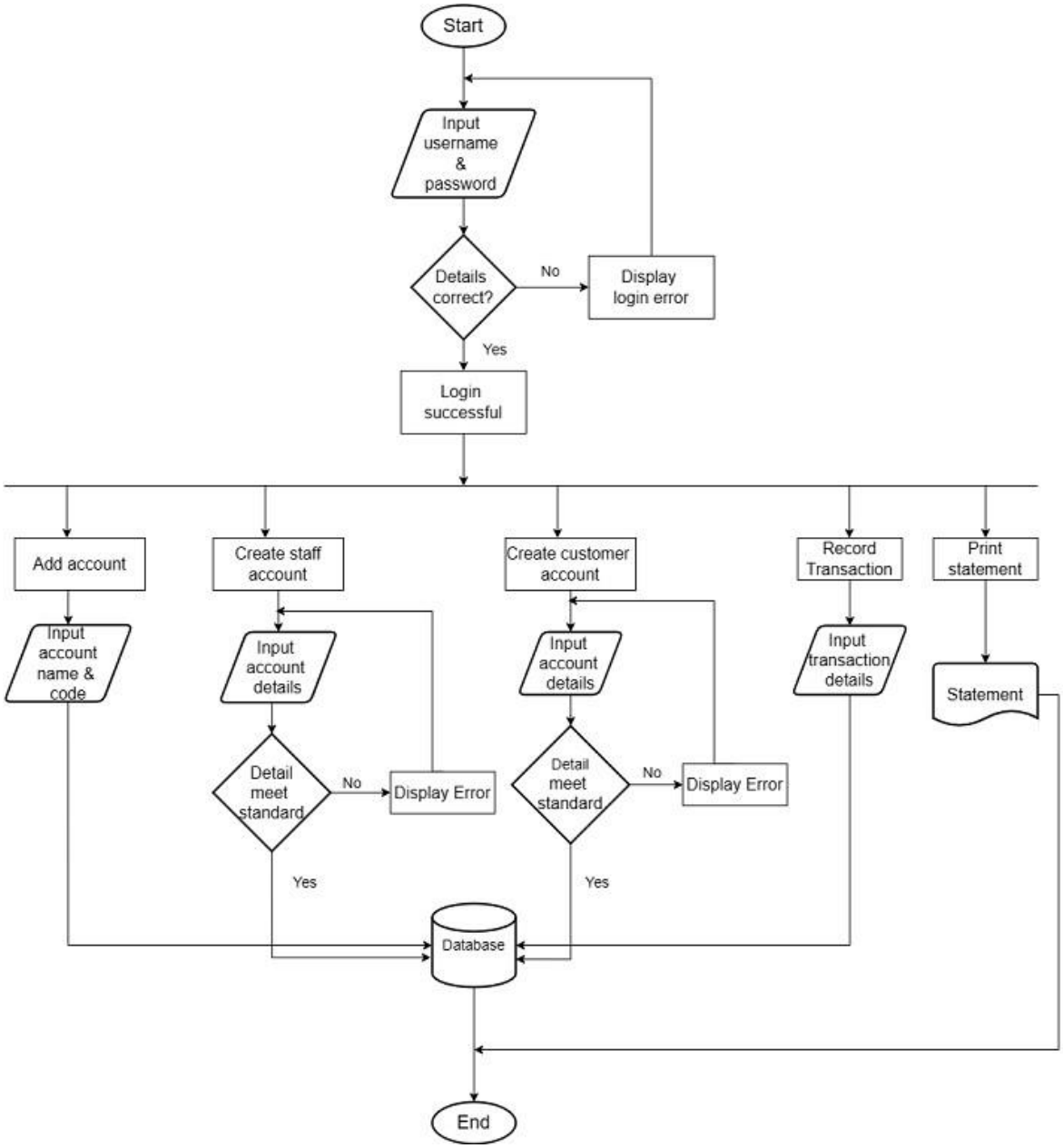
3.3.5 Software requirements

The software requirement is the particular software, such as operating systems, databases, and application software that a system requires to run successfully. The table below shows the software requirements for the microfinance management system.

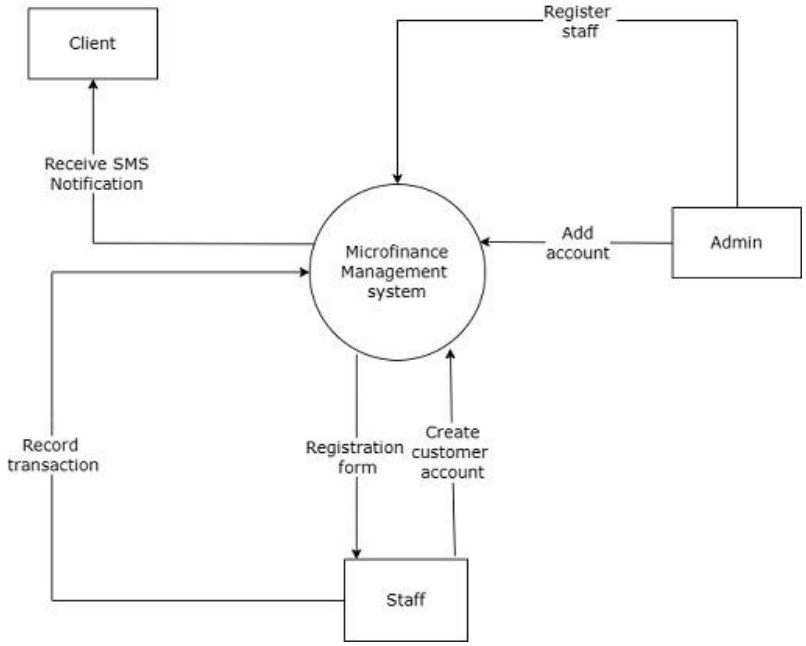
Operating systems	Windows 8.0 or later, macOS 10.12 Sierra or later, Linux.
Database and Server	MySQL and Apache (XAMPP)
Application software (Browser)	Google Chrome, Microsoft Edge, Safari, Firefox, Brave.

Table 3.3 Software Requirements

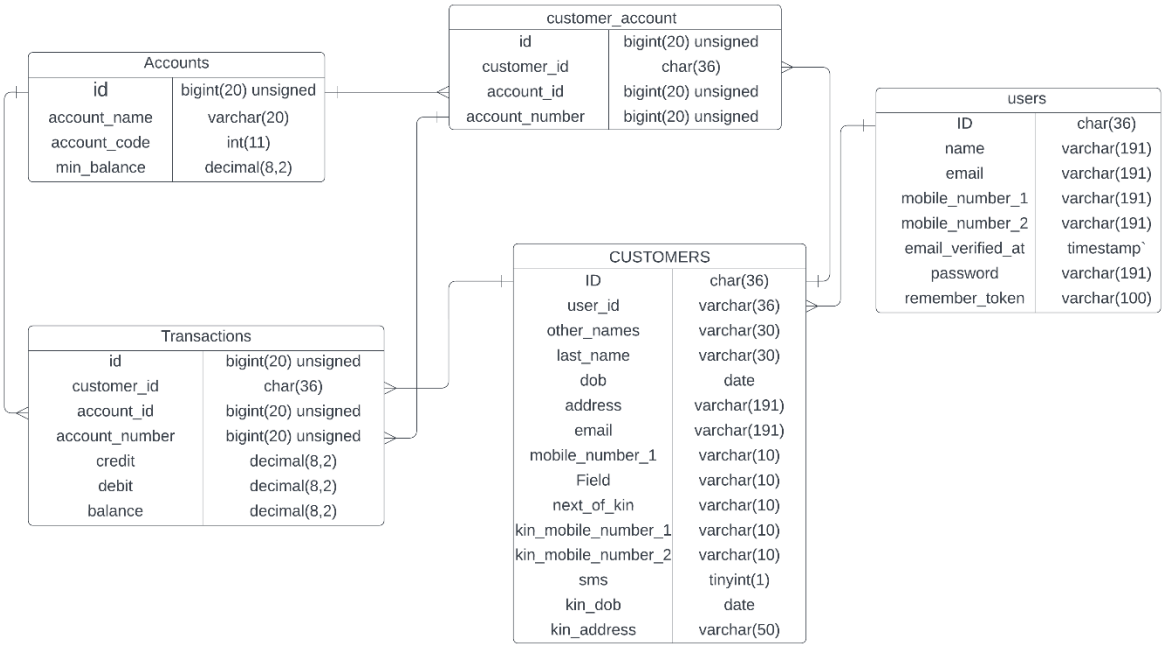
3.4 Flowchart Diagram



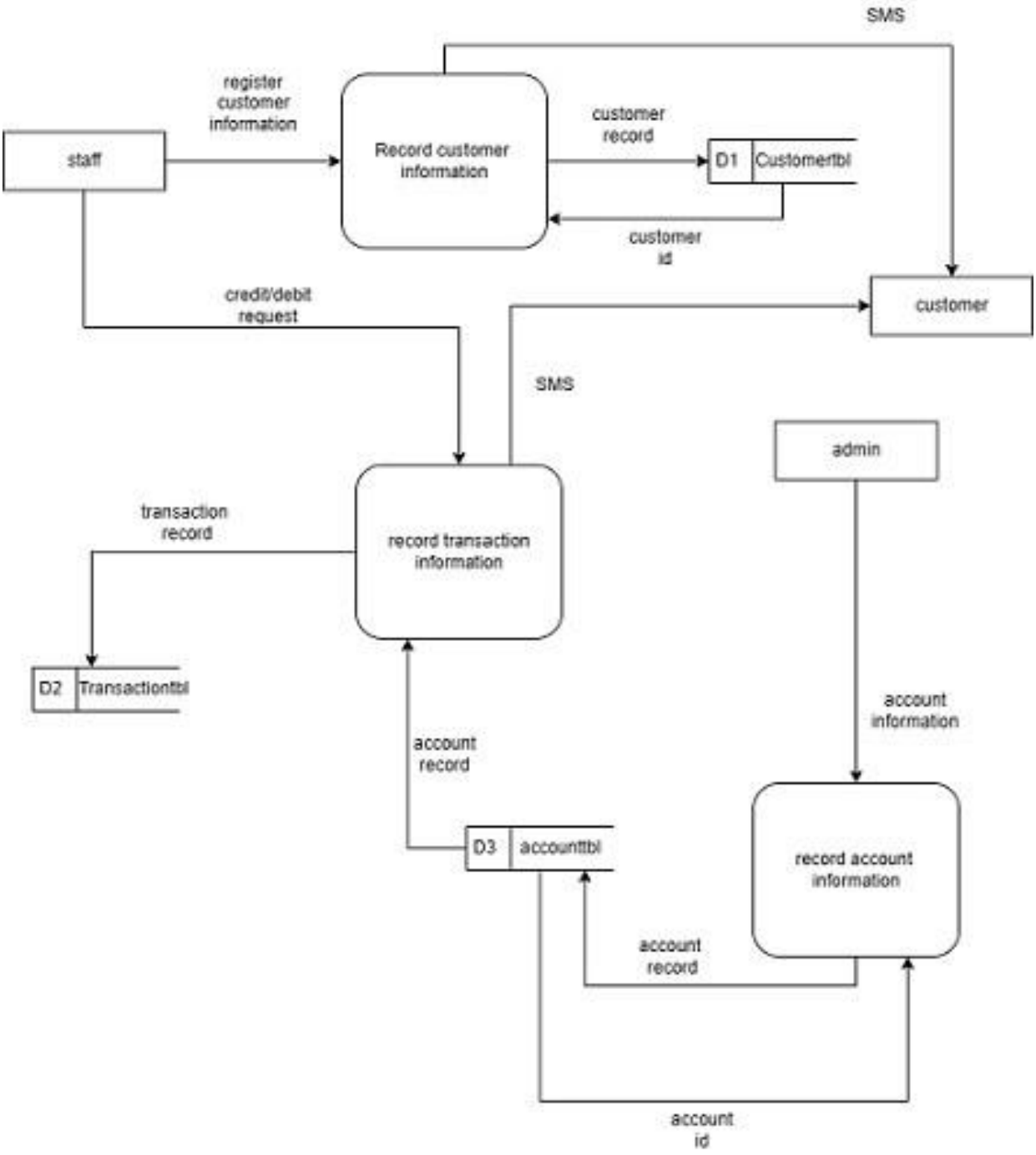
3.4.1 Context Diagram



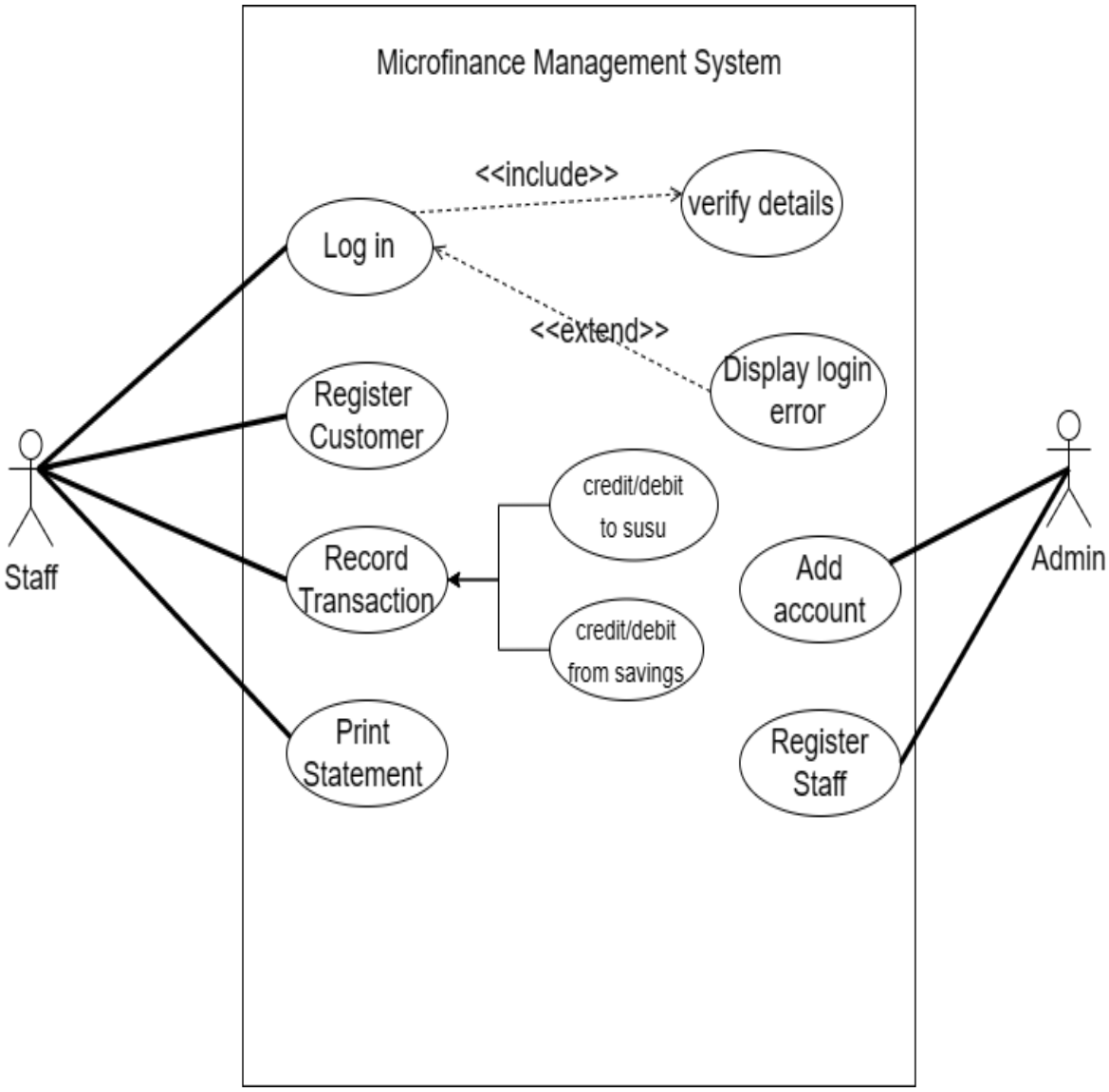
3.4.2 Entity Relationship Diagram



3.4.3 Dataflow Diagram



3.4.4 Use Case Diagram



3.5 Tools Used

The tools selected for the implementation of this project were chosen based on their capability to facilitate rapid web app development while simultaneously enhancing the overall system quality. The tools utilized in this project include:

Laravel: Laravel is a fast application development framework that emphasizes a short learning curve and cuts down on the number of steps needed to get from creating a new project to releasing it. The components offered by Laravel make it easier to do many of the most frequent activities in developing a web application, such as database interaction, authentication, etc. For the development and release of apps, Laravel offers a whole ecosystem of tools. Simplicity is a key component of Laravel. (Stauffer, 2019).

MySQL: The most widely known standardized language for accessing databases is known as MySQL or "Structured Query Language". The Oracle Corporation is the creator, distributor, and supporter of MySQL a well-known open-source SQL database management system. MySQL helps to add, access, and process data saved in the database. The logical model provides a flexible environment for programming due to elements like databases, tables, views, rows, and columns. (Christudas,2019).

vscode: With a built-in debugger and integrated support, it is a very highly effective code-focused development environment that was specifically created to make it simpler to write web, mobile, and cloud applications in languages that are supported by various development platforms. It also supports the entire application development life cycle. (Del Sole, 2021).

Vue.js: Vue is a contemporary, open-source JavaScript framework that is used to build quickly changing user interfaces. It is simple to integrate with a wide variety of libraries and preexisting web apps. (Cherckesova et al, 2021).

Xampp: Xampp developed by Apache is a free open-source program that runs on multiple platforms. It encompasses an HTTP server and PHP interpreter, making it an effortless,

versatile, and lightweight solution for developers to establish a local web server for testing web apps and development. (Rainu et al, 2017).

3.6 Conclusion

The proposed system's life cycle design has been thoroughly planned and executed with the use of various tools and diagrams. The use of Laravel as the primary development framework in conjunction with the reliable database MySQL ensures the stability and functionality of the system. The diagrams provide a comprehensive overview of the system and interaction. All functional requirements have been taken into consideration to guarantee that the proposed system meets its intended purpose.

CHAPTER 4

SYSTEM TESTING, IMPLEMENTATION, AND DOCUMENTATION

4.1 Introduction

This chapter represents a crucial point where the proposed system is implemented and made usable. The system is coded, integrated, and tested throughout the implementation phase to make sure it complies with the requirements and it’s error-free. This is an essential step. This chapter's main goal is to present a comprehensive study of the system's documentation, testing procedures, and implementation process. The chapter will provide an overview of the various testing methods used to make sure that the system works as planned and satisfies user needs. In addition, for future upgrades and maintenance, this chapter will include thorough documentation of the system's design, and functionality.

4.2 Testing of The New System

Software testing is a methodical process of scrutinizing and evaluating software to find and isolate any glitches or flaws in the system. Software testing's main goal is to assess the qualities or capabilities of the program and make sure that they adhere to the established quality standards. Software testing can assist in determining additional quality factors, such as reliability, usability, integrity, security, efficiency, functionality, and compatibility, in addition to identifying mistakes. Software developers can enhance the quality of their products and make sure the program functions as intended by conducting tests as part of the development process. (Sawant et al., 2012).

4.2.1 Unit Testing

Unit testing is the testing of individual units or groups of related units following the development of a system. (Runeson, 2006). In performing the unit testing of the Microfinance Management System, test cases were designed to cover the requirements and expectations of the login form to verify the authentication mechanism. The test was performed in an isolated

environment. During the unit testing for the login form, we needed to ensure that when a person is registered in the system and logs in, the system automatically redirects them to the system's dashboard successfully. The following steps were taken to verify the functionality of the authentication mechanism.

1. A Function was created for the login to redirect an authorized user who has been registered into the system to the dashboard.
2. A user object with an email and password was established.
3. A post function was then created to simulate the login form submission with the test credentials.
4. Next, a redirect function was developed to guarantee that the user gets redirected to the correct page.
5. To ensure that the user was authenticated as the authorized test user object, an authentication method was also developed.
6. The test was carried out.

The test's outcome showed that the login authentication functionality was working correctly.

An additional test was run to verify if an unauthorized person could access the dashboard. The system's ability to prevent unauthorized users from accessing the home page was tested using the procedures below.

1. A function test login with an incorrect credential was generated.
2. Following that, an Error function was written to check the error message for incorrect credentials.
3. A Guest function was then passed to ensure that the user was not authenticated.
4. The test was carried out.

As a consequence of conducting this test, the system returned the user to the login page to be authenticated.

4.2.2 Functional Testing

Finding faults in the functional behavior of the system being tested is the goal of functional testing. To achieve a desired function for the system's user, the intended functional behavior is determined by the functional requirements of the system. When the system behavior does not adhere to the functional requirements of the system, errors are present. (Buhler 2008). In testing the functional behavior of the Microfinance Management System to identify errors in the system, the Black box method was used. The black box testing technique identifies flaws or faults in software based on how they appear in the output. Black box testing disregards the internal workings of the system and the operations carried out. (Galin, 2004). It only cares about if the system is outputting accurate results based on the input. The following test cases were created in verifying the login form of the system.

Functionality: Login form authentication.

Test case: To verify that the login form correctly authenticates users and grants access to the appropriate page and invalid credentials are not granted access.

Test steps

1. Open the Microfinance Management System.
2. Enter a valid username and password for a staff member and click the “Login” button.
3. Follow step 2 once more using a legitimate username and an incorrect password combination, then click the login button.
4. Repeat step 2 for invalid username and invalid password combination and click login.

Expected Results

Step 2 directed the user to the system dashboard displaying the Key Performance Indicator (KPI) of the Microfinance Management System.

Steps 3 and 4 produced error messages suggesting that the credentials were invalid and that a valid login and password were required.

To determine whether customer accounts could be successfully established and whether any erroneous data would prevent the system from saving the customer details, a different test case was built. The following test scenarios were developed, along with the anticipated outcomes of the test.

Functionality: Create a customer account

Test case: To ensure that a new customer may be successfully added and their information is saved in the database and the system prevents saving details if required fields are not completed.

Test steps

1. Log in to the Microfinance Management System.
2. Navigate to “User” and click “Customer list”.
3. Click on “Create new customer” enter valid details and data types in all the fields and click the save button to save the customer details.
4. Repeat steps 1 to 3 with invalid customer details, and missing required data fields.

Expected results

Step 3 successfully stored all client details in the database, and the customer was presented on the customer list. In step 4, the system prohibited the information from being stored in the

database and presented an error message noting that required fields must be filled in and not left blank.

The final functional test was performed on the transaction module, which allows employees to credit and debit a customer account. To run this test, test cases were written to validate the output after any transaction was performed.

Functionality: Record Transaction

Test case: To guarantee that the transaction module appropriately handles various sorts of transactions done by employees and that crediting and debiting amounts are correctly recorded, and SMS notification is sent to the customer.

Test steps

1. Log in to Microfinance Management System.
2. Navigate and click on transaction.
3. Click on record transaction.
4. Click on the new transaction.
5. Enter the account number of the customer and select the type of transaction applicable (credit/debit).
6. Enter the amount and click save.

Expected result

Following this test, the system logged the amount and kind of transaction completed. The system then promptly issued an SMS notice to the customer stating that a particular amount had been deducted or credited to her account indicating the date on which the transaction was performed. In addition, after entering the customer's account number, their profile is presented for verification reasons before the transaction is completed.

4.2.3 Usability Testing

Usability testing is a dynamic procedure that may be applied throughout the development of interactive software. The goal of usability testing is to identify flaws and give recommendations to improve a software's functionality. Learnability, user satisfaction, and performance effectiveness are the three dimensions of usability testing. (Lee, 1999). Participants were chosen to complete a task on the Microfinance Management system as part of the usability test. Participants had to complete transactions, make customer accounts, generate reports, and add staff members to the system. The idea was to be able to judge how simple or complex the system is to operate based on the arrangement and design of the modules. Performing the aforementioned exercise also assisted us in identifying flaws and improvements. Following the completion of the required task by the participants, the following inquiries were made:

1. How difficult or simple was it to locate the record transaction module and complete the customer registration form?
2. What do they think of the module layout?
3. How long did it take them to understand the system completely?
4. On a scale of 1 to 10, how will they grade the system’s performance?

After carrying out the test, most participants described that it was quite simple to locate the record transaction module and complete the customer registration form. The majority of participants also mentioned that the modules' simplicity made it simple to navigate through them. Instead of having to open a new layout form and navigate back and forth each time a user wants to conduct a different activity, placing the modules on the left side of the system was a useful method to provide users with a fast overview of the tasks to be accomplished on the other side of the system. Because of this, the whole design seems user-friendly, which will improve the user experience and boost productivity. The majority of participants claimed that

after their first attempt at utilizing the system, they were able to comprehend how it operated, and by their second attempt, they had become familiar with it. The majority of the participants gave the system a 6 out of 10 rating on their initial use due to errors they experienced. They gave the system a 9/10 when the mistakes were addressed.

4.2.4 Acceptance Testing

A test kind that reflects a customer's interest is acceptance testing. It instills trust in the end user that the program has all the necessary functionality, achieves its goals, and behaves properly overall. (Miller & Collins, 2001).

4.2.5 Visual (Live) Testing

Visual testing is a method of software testing that assesses the visual look and behavior of a software application's Graphical User Interface (GUI) or User Interface (UI). Its goal is to ensure that the application's visual features, including colors, pictures, fonts, and layout, are presented accurately across various devices, operating systems, and browsers. The significance of doing this test also aids in the identification of visual flaws that may negatively impact the user experience and usability of the system. (Bose, 2023). During the visual testing of the Microfinance Management System, the program was hosted on Xampp, allowing us to access the system from another device to the local host. A few people were entrusted with accessing the system using tablets, Android devices, and iOS devices. Throughout the test, the arrangement of the modules was correctly positioned regardless of the device from which the system was accessed.

4.3 Implementation of New System

The goal of system implementation is to place continuous system support and maintenance inside the Performing Organization (the transition) and make the new system accessible to a

ready group of users (the deployment). In more specific terms, deploying the system entails carrying out all procedures required to inform users on how to use the new system, putting the newly developed system into production, ensuring that all data necessary for the start of operations is accessible and accurate, and validating that business processes that interact with the system are operating as intended. (George et al., 2003).

4.3.1 Parallel Implementation

According to Motiwalla & Thompson (2011), the system is implemented and utilized concurrently with the legacy system in a parallel implementation. When the danger of system failure is a major issue, this form of system implementation is preferable.

The parallel implementation strategy will be used for implementing the Microfinance Management System. As a result, if the system fails or one of the modules fails, the business may continue to operate since there is redundancy. This will allow us to work on resolving the issue that was encountered. This also aids staff in learning the new system while conducting training.

4.3.2 Phased Implementation

Modules of the system are released into production when they have been thoroughly tested and found to be functional. (Motiwalla & Thompson, 2011). The complete system will not be functioning until each module has been utilized independently. Staff training is gradual with this form of implementation since they just need to be trained on the current module being utilized. It also aids in the detection of faults in a certain system module. The disadvantage of employing this strategy is that it is more expensive because each module must be tested and validated that it is operating properly, as well as offering training on each module, as compared to other implementation methods.

4.3.3 Pilot Implementation

Pilot implementation involves progressively transitioning from the existing system to the new system. This way of deployment takes a long time but is also the least detrimental to the business. (Motiwalla & Thompson, 2011). The majority of the groups in the firm continue to use the old system while a small number of them are chosen by the company's management to utilize the new system. The few groups using the system are the only ones affected by any problems encountered while using it; the company as a whole is not affected. The drawback of this approach of implementation is that it takes a very long time until the new system is fully operational. This implementation strategy is very costly because of the system's limited implementation budget as well.

4.3.4 Direct Implementation

The approach of system implementation with the highest risk is direct implementation. This method allows the new system to function after being installed. The older system is taken down, and all of the data from it is transferred to the new system. (Motiwalla & Thompson, 2011). As all users will quit using the old system and be educated on the new system, this strategy has the benefit of being less expensive. The drawback of this strategy is that business continuity issues arise if the new system fails. Due to their lack of comprehension of how the system functions, users' productivity is also hampered.

4.4 System Documentation

A description of what the system does, how it works, and how it should be used is provided in system documentation, which serves as a resource for users and developers. Moreover, it describes how users should communicate with the system. (Aghajani et al., 2019). According to Kipyegen and Korir (2013), system documentation improves the system's usability and ease

of the support. Quality documentation makes information easily available, allows new users to learn fast, and simplifies the product, lowering support costs.

4.4.1 About the System

The Microfinance Management System (DTSL) is a computerized web application system that will aid Divine Trusties Saving and Loan staff to register customers who want to save with the company. It will also enable the staff who go around the community to collect susu and perform transactions with just having an electronic device. This system was developed to address the issue of staff or the susu collectors carrying ledger books around to record transactions and register customers' accounts on paper before retyping the details in an Excel sheet. The system has the following features.

i Dashboard

The dashboard of the Microfinance Management System (DSTL) shows the KPI such as products or services being provided by DSTL which a customer can sign up for such as susu or savings, the total number of customers who signed up for a particular product, the total number of customers who have accounts with DSTL and the total number of employees as shown in the figure below.

Fig 1 – Dashboard

ii. Users

The customer list and the staff list are the two sub-modules under the users' module. The staff list module enables DTSL management to set up a user profile for newly hired employees. When a user creates an account, their information is presented among the personnel that are available, as illustrated in the image below.

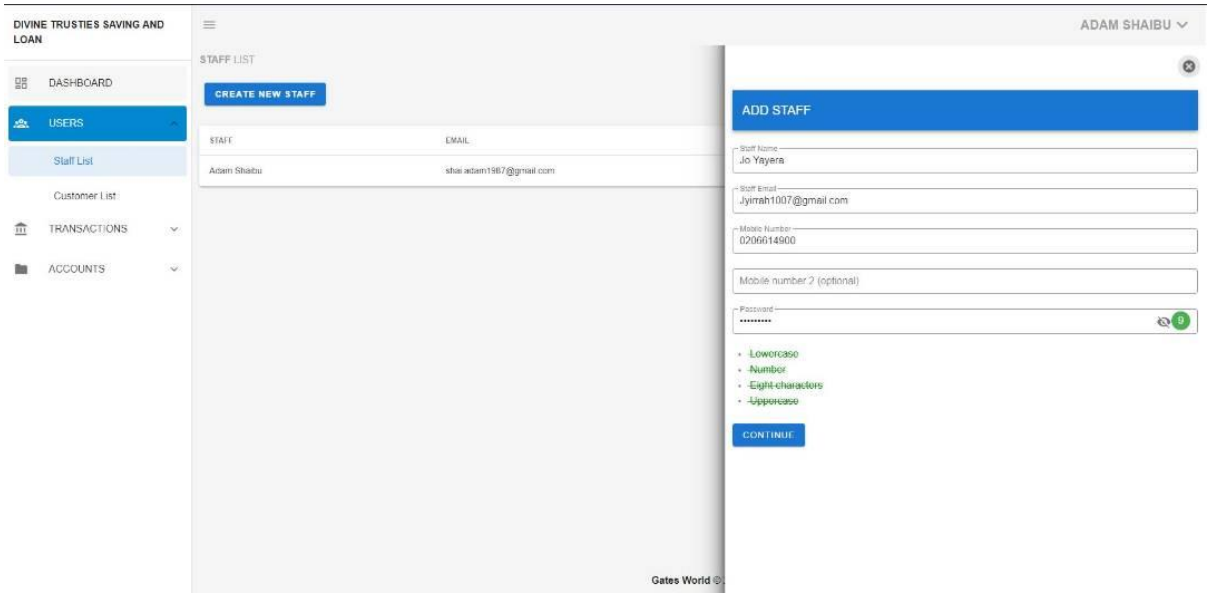


Fig 2 – Staff list

The customer list module enables DTSL personnel to register customers. Customers' information is presented on the customer list dashboard after they have been registered. When a member of staff clicks on the information of a certain customer, the system displays an overview of the customer's details and account details, as seen in the image below.

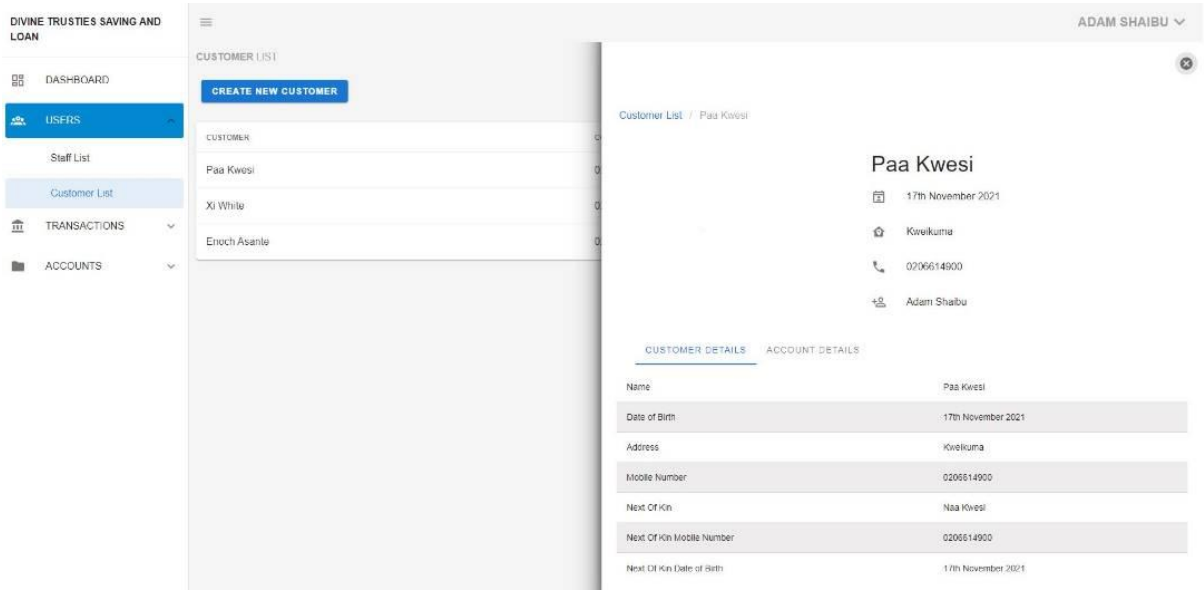


Figure 3 – Customer List

iii. Transaction

The transaction module facilitates various types of transactions, such as credit and debit. This module is divided into two sub-modules: Record transaction and Statement. All transactions will be performed and recorded using the Record transaction module. It contains the following functions: an account number field for entering the customer's account number, the amount to be credited or debited, and a transaction type category, as illustrated in the image below.

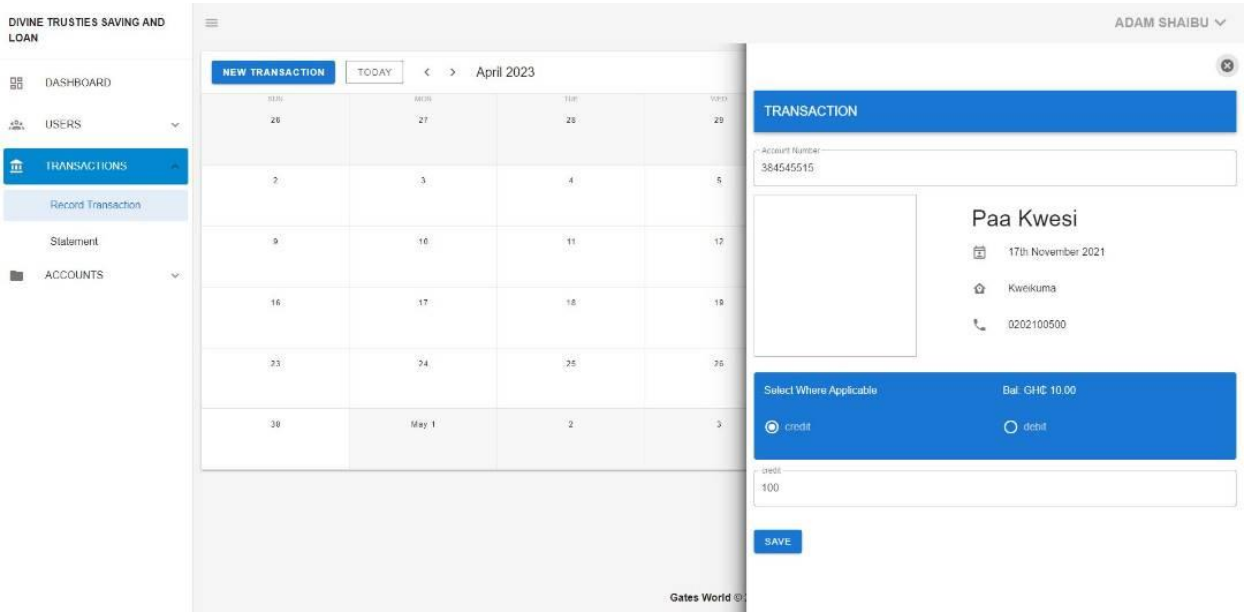


Fig 4 - Record Transaction

The statement module will assist in printing the statement or report of transactions completed within a specific period. The Start and End dates assist in selecting all transactions that happened within that period. The user will have the choice of generating the data in either an Excel file or a PDF file, as indicated in the picture.

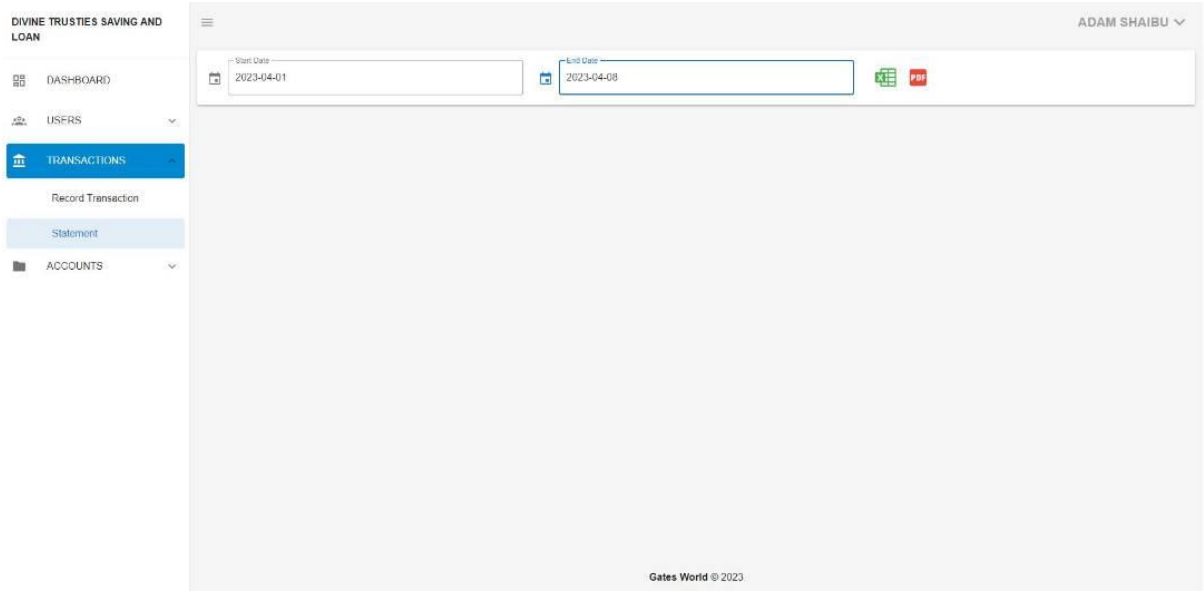


Fig 5 - Statement

iv. Account

This module enables DTSL to offer new items that customers may sign up for. As illustrated in the graphic below, each product has a unique code that identifies it.

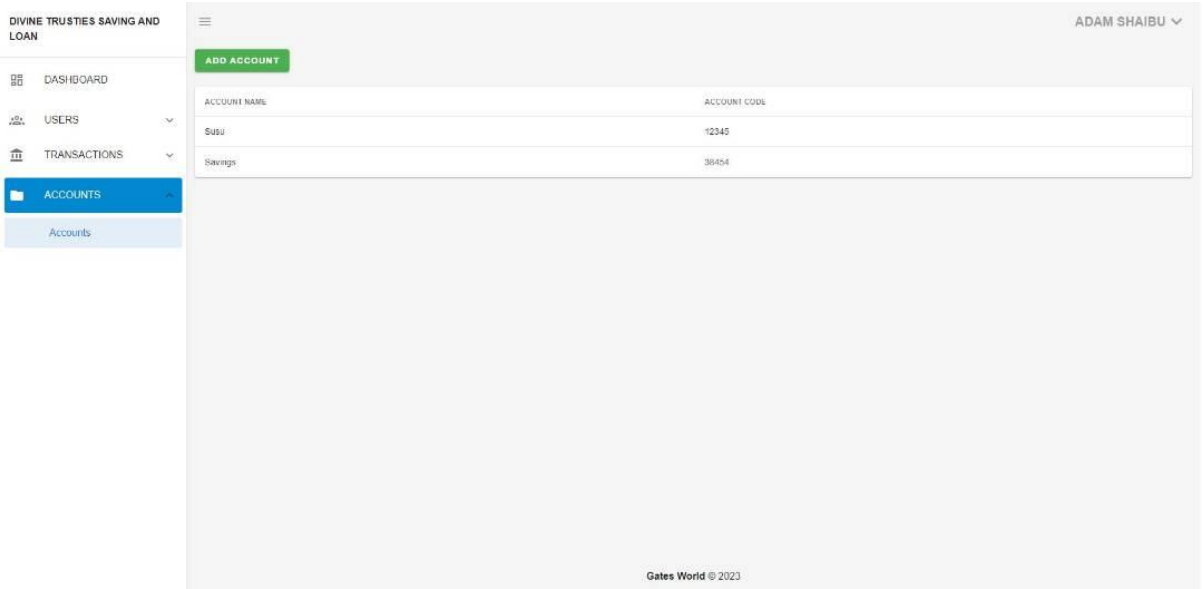


Fig 6 - Account

After all transactions have been completed, the system will notify the customer through SMS.

The image below depicts an SMS notice received after completing a transaction.

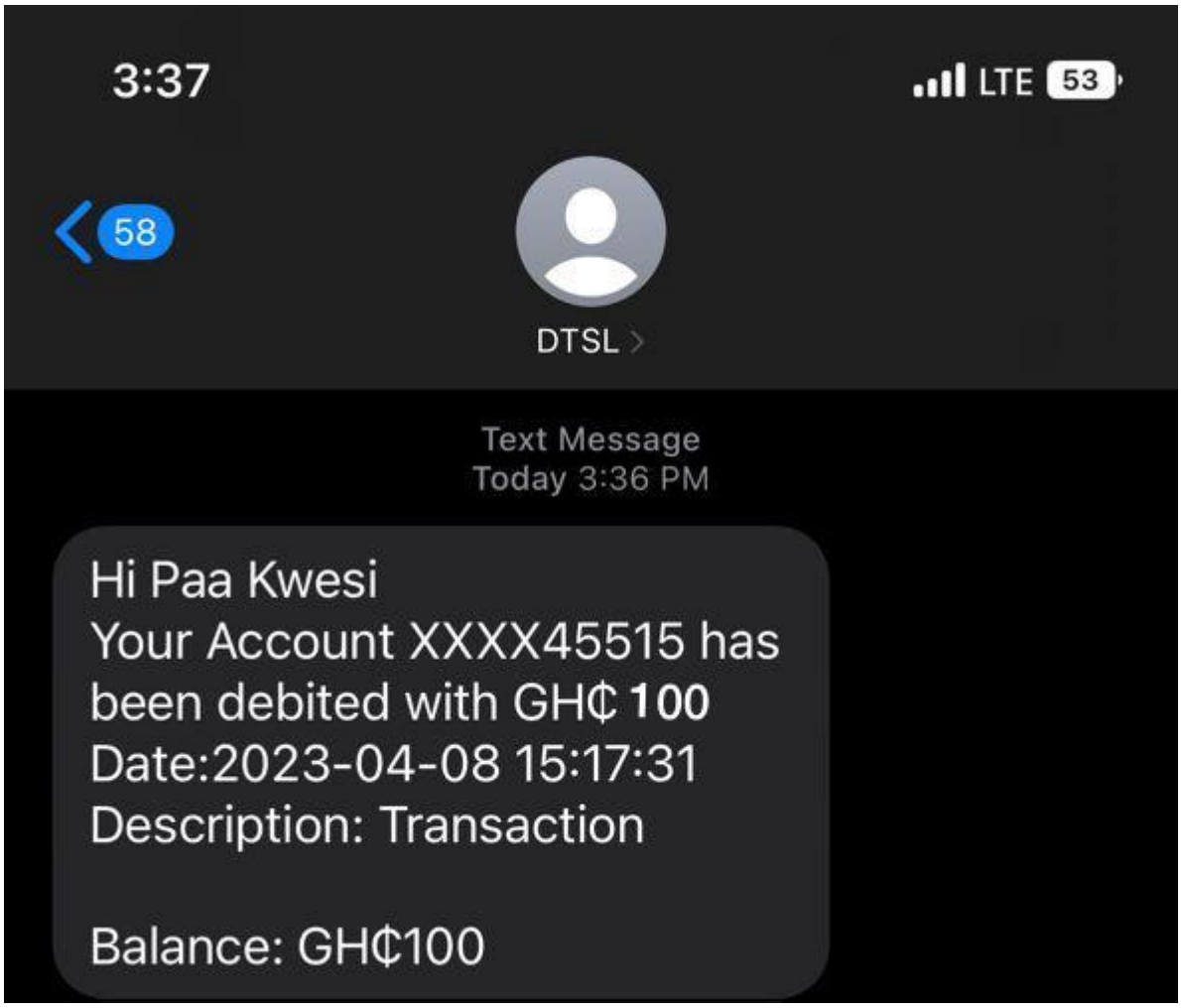


Fig 7 - SMS

4.4.2 User Access Level

The users have access to the following

1. Dashboard
2. User
    - Staff list
    - Customer list
3. Transaction

- Record transaction
- Statement

4. Account

4.4.3 Getting Started

The prototype of the system is currently hosted on the Xampp server. To access the system, use the following steps.

- Start the Xampp server **(start Apache and MySQL**) then close Xampp.
- Open Vscode and in the terminal, type **npm run serve** and press enter.
- Open the command prompt (CMD) and in the terminal type **php artisan serve.** NB: when you are notified with an error message after typing the command, type in this command **Set NODE_OPTIONS= - - openssl-legacy-provider** then press enter.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Introduction

This chapter presents the summary, recommendation, and conclusion of the Microfinance Management System developed in this project. The system was designed to provide a digital platform for managing microfinance transactions, staff, and customers. In Chapter 3, we described the system architecture and design, and in Chapter 4, we presented the implementation and testing details. In this chapter, we emphasize the key system objectives and capabilities, offer suggestions for further enhancements, and conclude consideration of the objectives and results of our project.

5.2 Summary

It was mentioned in Chapter 1 of this project that the origin of microfinance can be traced as far back as the middle of the 19th century, when Lysander Spooner wrote about the value of small loans to farmers and business owners, and that in the 1970s, Mohamed Yunus's Grameen Bank of Bangladesh, which he founded, started to influence the microfinance industry. There are now millions of microfinance organizations offering a variety of financial services including loans, savings accounts, Susu, etc. We also discussed some of the issues that microfinance institutions confront as a result of not having a secure and efficient digital platform on which to carry out their activities. We then discussed how our system will tackle these difficulties, as well as the system's scope. The system requirements, such as hardware requirements, software requirements, and tools used to construct the system, were described in Chapter 3.

To emphasize the system, the primary purpose of developing the microfinance management system was to provide a secure and efficient digital platform for microfinance institution

workers to perform transactions and register customers without the need for ledger books or paper registration forms. Furthermore, the technology was built to help the institution generate a report, evaluate product performance, and eliminate fraud by sending SMS to customers once they had been credited or debited. As stated in Chapter 1, management's overall goal is to conserve money while increasing productivity. Modules such as customer lists, record transactions, and accounts that will aid the institution in introducing new products were created to assist us in meeting our objectives.

5.3 Recommendations

We have extended the usage of a digital platform to microfinance institutions with the aid of this project, and their employees will no longer utilize ledger books to record transactions or carry paper registration forms to register new customers. Throughout this project, we would like to urge that small and medium-sized microfinance institutions do everything possible to implement such a system to boost productivity and minimize fraud.

5.4 Conclusion

In conclusion, the Microfinance Management System has been successfully developed to cater to the needs of small and medium-scale microfinance institutions. The system provides a seamless user experience for staff members to carry out transactions, manage their accounts, assist management in introducing new items, and monitor product performance. The project has successfully implemented key functionalities such as user authentication, transaction management, user management, and a message notification system.

As part of our plans for this project, future improvements would include multifactor authentication, a loan request module, the ability for customers to log in, and a data visualization module to help management get more insight into their product performance. This will provide additional convenience to the staff and customers and enhance the security of both the system and user accounts.

REFERENCES

Addae-Korankye, A. (2020). The historical emergence of microfinance and the microfinance sector in Ghana. Business Management Dynamics, 10(02), 1-11.

Ashta, A. (2010). Advanced Technologies for Microfinance: *Solutions and challenges*. Business Science Reference, USA.

Ashta, A., Barnett, B., et al. (2015). Management Information Systems for Microfinance: *Catalyzing Social Innovation for Competitive Advantage*. Cambridge Scholars Publishing.

Aghajani, E., Nagy, C., et al. (2019). Software Documentation Issues Unveiled. IEEE.

Badruddozza, M. M., & Ramage, M. (2018). ICT-mediated organizational change in microfinance organizations: *a case study*. Business Perspectives Ltd.

Beck, K. (2000). Extreme programming explained: *Embrace change* (2nd ed.). Boston, MA: Addison-Wesley.

Bose, S. (2023, March 9). Visual testing: *A beginner's guide*. Retrieved from https://www.browserstack.com/guide/visual-testing-beginners-guide/

Bühler, O., & Wegener, J. (2008). Evolutionary functional testing. Computers & Operations Research, 35, 2951-2967.

Cherckesova, L., Boldyrikhin, N., Revyakina, E., et al. (2021). Development of a real-time document approval system. E3S Web of Conferences.

Christudas, B. (2019). Practical Microservices Architectural Patterns. Apress. Berkeley, CA.

Consultative Group to Assist the Poor (CGAP). (2012). Small loans, big impact: *How small loans can change the lives of the poor*. Washington, DC: CGAP.

Del Sole, A. (2021). Introducing Visual Studio Code. In Visual Studio Code Distilled: *Evolved Code Editing for Windows, macOS, and Linux* (pp. 1-15). B Apress.

Galin, D. (2004). Software quality assurance*: From theory to implementation*. Pearson Education Limited.

George, E. P., James, T. D., & McCormack, M. (2003). Project Management Guidebook: *System Implementation*. New York State Office for Technology. Retrieved from https://its.ny.gov/nys-project-management-guidebook-release-2/

Ifeyinwa, A., & Chibueze, I. (2011). Loan fraud detection and IT-based combat strategies. Journal of Internet Banking and Commerce, 16(2).

Hulme, D., & Mosley, P. (1996). Finance against poverty volume 1& 2. Routledge. London and New York.

Kagan, J. (January 4, 2022). Microfinance Definition: *benefits, history, and how it works.*

Investopedia. Retrieved from https://www.investopedia.com/terms/m/microfinance.aspa

Kipyegen, N. J., Korir, W. P. K. (2013). Importance of Software Documentation. IJCSI International Journal of Computer Science Issues, 10(5), 1.

Kumari, P., & Nandal, R. (2017). A Research Paper on Website Development Optimization Using Xampp/PHP. *International Journal of Advanced Research in Computer Science*.

Knologist. (2022). What is a User-friendly System? Retrieved from https://knologist.com/what-is-a-user-friendly-system/

Lee, S. H. (1999). Usability testing for developing effective interactive multimedia software: *Concepts, dimensions, and procedures*. Journal of Educational Technology & Society, 2(2). Retrieved from http://www.jstor.org/stable/jeductechsoci.2.2.4/

Ledgerwood, J. (1998). Sustainable banking with the poor. Microfinance Handbook: *An institutional and financial perspective*. The World Bank, Washington, D.C.

Microfinance Info. (n.d.). History of microfinance. Retrieved from https://microfinanceinfo.com/history-of-microfinance/

Miller, R., & Collins, C. T. (2001). Acceptance testing. *Proc. XPUniverse*, 238.

Mohan, L., Potnis, D., & Alter, S. (2013). Information systems to support "door-step banking*": Enabling scalability of microfinance to serve more of the poor at the bottom of the pyramid*. Communications of the Association for Information Systems, 33(1), 423-442.

Motiwalla, L. F., & Thompson, J. (2011). Enterprise Systems for Management (2nd ed.). Pearson.

Ross, J. (2014). The business value of user experience. Cranbury, NJ: 2 Commerce Drive.

Runeson, P. (2006). A survey of unit testing practices. IEEE Software, 23.

Stauffer, M. (2019). Laravel: Up and Running: *A Framework for Building Modern PHP Apps*. O'Reilly Media.

Sawant, A., Bari, P., & Chawan, P. (2012). Software Testing Techniques and Strategies.

Waterfield, C., & Ramsing N. (1998). Handbook for management information systems for microfinance institutions. Consultative Group to Assist the Poorest.

World Bank Group. (2021). Microfinance. Retrieved from https://www.worldbank.org/en/topic/financialinclusion/brief/microfinance

APPENDIX A: PROGRAMMING CODES

Login

```
<template>
  <v-container fluid fill-height>
    <v-row align="center" justify="center">
      <v-col md="4" sm="8" lg="3">
        <div class="mb-5" flat>
          <!-- <p><v-img :src="require('@/assets/gctu_logo.png')"></v-img></p> -->

          <p class="font-weight-bold text-center">
            DIVINE TRUSTIES SAVING AND LOANS
          </p>
        </div>
        <v-card class="align-center">
          <div class="text-center">
            <v-alert
              v-show="IsErrors"
              text
              prominent
              type="error"
              v-text="errors"
            />
          </div>

          <v-card-text>
            <v-form ref="form" @submit.prevent>
              <v-text-field
                outlined
                clearable
                v-model="emailAddress"
                prepend-inner-icon="mdi-account"
                label="Username"
                type="email"
                :rules="rules"
                required
                @focus="removeErrors"
                @keyup.enter="btnSignin"
              />

              <v-text-field
                outlined
                clearable
                ref="password"
                v-model="passwrd"
                prepend-inner-icon="mdi-lock"
                label="Password"
```

```
            :append-icon="show3 ? 'mdi-eye' : 'mdi-eye-off'"
            :rules="rules"
            :type="show3 ? 'text' : 'password'"
            name="input-10-2"
            class="input-group--focused"
            @keyup.enter="btnSignin"
            @click:append="show3 = !show3"
          />

          <v-card-actions>
            <v-spacer />
            <v-scale-transition>
              <v-btn
                color="primary"
                :loading="loading"
                @click="btnSignin"
                width="150"
                class="font-weight-bold"
              >
                Login
              </v-btn>
            </v-scale-transition>
          </v-card-actions>
        </v-form>
      </v-card-text>
    </v-card>
  </v-col>
 </v-row>
</v-container>
</template>

<script>
 import { computed, reactive, toRefs } from "@vue/composition-api";
 import { useActions, useGetters } from "vuex-composition-helpers";

 export default {
  setup(props, context) {
    document.title = context.root._route.meta.title;
    const { authenticated } = useGetters(["authenticated"]);
    const { signIn } = useActions(["signIn"]);

    const form = reactive({
      emailAddress: "",
      passwrd: "",
      device: "browser",
      disabled: false,
      loading: false,
```

```
  rules: [(value) => !!value || "Required."],
  errors: [],
  show3: false,
});

const { disabled, loading, emailAddress, passwrd, errors } = toRefs(form);

const IsErrors = computed(() => {
  return errors.value.length > 0;
});

const btnSignin = () => {
  let email = emailAddress.value;
  let password = passwrd.value;

  disabled.value = true;

  let v = context.refs.form.validate();
  if (v) {
    loading.value = true;
    signIn({ email, password })
      .then(() => {
        loading.value = false;
        if (authenticated.value) {
          context.root._router.push({
            path: context.root._route.query.redirect || `/home`,
          });
        }
      })
      .catch((error) => {
        loading.value = false;
        disabled.value = false;
        context.refs.form.reset();
        switch (error.response.status) {
          case 422:
            errors.value = "User Credentials Are Invalid";

            break;

          case 429:
            errors.value = "Too Many Requests";

            break;

          default:
            errors.value = "Try Again Later";
            break;
```

```
            }
          });
        } else {
          disabled.value = false;
        }
      };

      const removeErrors = () => {
        errors.value = [];
      };

      return {
        IsErrors,
        removeErrors,
        btnSignin,
        loading,
        ...toRefs(form),
        errors,
      };
    },
  };
</script>

<style></style>
```

Account Module Frontend

```
/** @format */

import Api from "@/apis/Api";

export default {
  state: {
    accounts: [],
    countAccount: []
  },
  getters: {
    getters_accounts: (state) => state.accounts,
    getters_countaccount: (state) => state.countAccount

  },
  mutations: {
    ACCOUNT_LIST: (state, accounts) => (state.accounts = accounts),
    ADD_ACCOUNT: (state, account) => state.accounts.push(account),
    COUNT_ACCOUNT: (state, account) => state.countAccount = account
  },
```

```
  actions: {
    async accountList({ commit },) {
      await Api()
        .post("/accountlist",)
        .then((res) => {
          commit('ACCOUNT_LIST', res.data.data)
        });
    },

    async addAccount({ commit }, account) {
      await Api()
        .post('/addaccount', { account })
        .then((res) => commit('ADD_ACCOUNT', res.data.data))
    },

    async countAccount({ commit }) {
      await Api()
        .post('/countaccount')
        .then((res) => commit('COUNT_ACCOUNT', res.data.data))
    }

  },
};
```

Transaction Module Frontend

```
/** @format */

import Api from "@/apis/Api";

export default {
  state: {
    accounts: [],
    monthlyTransaction: [],
    transaction: {}
  },
  getters: {
    getters_monthly_transactions: (state) => state.monthlyTransaction,
    getters_transaction: (state) => state.transaction
  },
  mutations: {
    MONTHLY_TRANSACTION: (state, transaction) => state.monthlyTransaction = transaction,
    SAVE_TRANSACTION: (state, transaction) => state.transaction = transaction
  },
  actions: {
```

```
  async monthlyTransaction({ commit }, date) {
    await Api()
      .post("/getmonthlytransaction", { date })
      .then((res) => {
        commit('MONTHLY_TRANSACTION', res.data.data)
      });
  },

  async saveTransaction({ commit }, data) {
    await Api()
      .post("/savetransaction", { data })
      .then((res) => {
        console.log(res)
        commit('SAVE_TRANSACTION', res.data.data)
      });
  },

 },
};
```

Users Module Frontend

```
/** @format */

import Api from "@/apis/Api";

export default {
 state: {
  staff: [],
  staffMeta: [],
  singleStaff: [],
  members: []
 },
 getters: {
  getters_staff: (state) => state.staff,
  getters_staff_meta: (state) => state.staffMeta,
  getters_single_staff: (state) => state.singleStaff,
  getters_members: (state) => state.members
 },
 mutations: {
  STAFF_LIST: (state, staff) => (state.staff = staff),
  ADD_STAFF: (state, staff) => state.staff.push(staff),
  STAFF_LIST_PAGINATE: (state, meta) => state.staffMeta = meta,
  SEARCH_STAFF: (state, staff) => state.singleStaff = staff,
  COUNT_MEMBERS: (state, members) => state.members = members
 },
```

```
  actions: {

    async staffList({ commit },) {
      await Api()
        .post("/stafflist",)
        .then((res) => {
          commit('STAFF_LIST', res.data.data)
          commit('STAFF_LIST_PAGINATE', res.data.meta)
        });
    },

    async addStaff({ commit }, staff) {
      await Api()
        .post('/addstaff', { staff })
        .then((res) => commit('ADD_STAFF', res.data.data))
    },

    async searchStaff({ commit }, staff) {
      await Api()
        .post('/searchstaff', { staff })
        .then((res) => commit('SEARCH_STAFF', res.data.data))
    },

    async countMembers({ commit }) {
      await Api()
        .post('/countmembers')
        .then((res) => commit('COUNT_MEMBERS', res.data.data))
    }

  },
};
```