Design and Implementation of a Co-operative System for Baze University Staff

BY

Anikwenze Olisa Harry
BU/21C/IT/5648

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING, FACULTY OF COMPUTING AND APPLIED SCIENCE, BAZE UNIVERSITY, ABUJA.

SEPTEMBER 2024

DECLARATION

I hereby declare that this research project has been written by me under the supervision of Dr. Usman Bello Abubakar. The work has been presented in any previous research for the award of B.Sc degree to the best of my knowledge. The work is entirely mine and I accept the sole responsibility for any errors that might be found in the work, while the reference to publish material have been duly acknowledge.

_____

_____

Anikwenze Olisa Harry Date

BU/21C/IT/5648

APPROVED BY

_____

Head of Department,

Department of Computer Science

CERTIFICATION

This project entitled "**Design and Implementation of a Co-operative System for Baze University Staff"** meets the requirements governing the award of Bachelor of Science in Software Engineering in Baze University, Abuja.

APPROVAL

This is to certify that the research work title "**Design and Implementation of a Co-operative System for Baze University Staff"** by Anikwenze Olisa Harry with BU/21C/IT/5648 has been approved by the Department of Computer Science, Faculty of Computing and Applied Science, Baze University, Abuja, Nigeria.

By
Dr Usman Bello Abubakar
Supervisor Date

Dr Usman Idris Abubakar
Head of Department
Date

Prof Helen Negbenebor
Dean, Faculty of Computing and Applied Science
Date

Prof Choji Davou Nyap
External Examiner
Date

DEDICATION

I am only where I am today and can only move forward by God's Grace who this project is dedicated to firstly, he has imbued me with the strength to overcome all the difficulties I have faced, allowed me to attend a university like Baze and blessed me with my friends and family. For this and more, I will forever be grateful and dedicated to him. I would also like to thank my family members, my parents Mrs. UzoAmaka Anikwenze and Engr. Bartholomew Ndubisi Anikwenze, they have made a lot of sacrifices for me, and I would not be able to complete this project without their love and support. Big thanks go to my sisters next, Jennifer, Sophie and Sonia. They have always believed in me and told me I can do anything I set my mind to even in my lowest moments, thank you for your love and always staying by my side. Thanks also go to my Uncle Mr. Emeka, thank you for your unwavering support.

I would also like to express my gratitude to my friends for the support I have received from them including but not limited to Ussah Nuhu-kwache, Emerald Akhaumere, Ammar Baba-Ahmed, Fadimatu Musa, Eldad Akhaumere, Bethel George-Nwaeke, Pochita Nerat Nwajah, Samuel Chimebuka Okorie,Collins Garba, Nafisat Muhammad, Hafsat Abbas Jega, AbdulWahab Uthman Mohammed, Iwuanoruo Osinachi, Tobenna Oguebie, Munira Bodoo, Remi Bakare, Abdallah Ibrahim, Maryam Umar, Aliyu wakama,Teslim Kayode, Zayyad Rumah lawal, Mohammed Dayyib Abdullahi, Favour Onyekachi, Blessing Onyekachi, Mujahid Adebisi, Oghenetega Ighomuaye Alvin, Maryam Saidu.

And to my wonderful friends outside school, Fajimi Wuraola, Angel Akpoveso, Demilade Olagungu, Daniel Asuquo. Thank you for listening to me during trying times.

ACKNOWLEDMENT

My appreciation goes to whole faculty of Computing and Applied Science, especially the Software Engineering and Computer Science department. The lecturers and students have driven me to always strive for academic greatness.

Special thanks go to me supervisor, Mr. Usman Abubakar Bello, thank you for guiding me through this process and not giving up on me. Thanks also go to my former supervisor Dr. Moses Ubaru for guiding me up to the point where he did, I am grateful for all his efforts.

I would also like to express my gratitude to my faculty's Head of Department, Dr. Usman Idris Abubakar for always listening to our complaints as students and encouraging a conducive learning environment.

I would like to thank the members of my department with whom I have completed this degree.

Next, I would like to thank the staff of Baze University who have looked out for me in some way or the other, both those in the computing and Applied sciences faculty, and other faculties, some include- Mrs. Mercy Johnson, Miss Khadijah Saad Mohammed, Mr. Gilbert George, Dr. Julius Makinde, Dr. Daniel, Dr. Mrs. Esther Omanayin, Dr. Lawrence Moralake, Dr. Charles Isah Saidu, Prof. Peter Ogedebe, Mr. Muhammad Shafi'u Shehu, Mrs. Mubarakah, Mrs. Ruqayya Muhammad, Mr. Tijani. Every one of my lecturers have supported me through my stay in this institution and I am eternally grateful, thank you for your support, love, and faith.

ABSTRACT

In a time where technology is rapidly advancing, co-operative societies need to be carried along by digitalizing their activities, transforming their manual activities to follow a more technological based system. The applications being developed include a web application and a mobile application. The mobile application is s user friendly system that allows members to perform their regular activities on the go like requesting for loans, always viewing their loan status and balance. The web app will allow the admin to oversee activities performed on the system and swiftly act in the occurrence of certain events. Our developed system is tailored to meet the needs of the co-operative society managed by Baze University to streamline their activities.

Table of Contents

DECLARATION ii
CERTIFICATION iii
APPROVAL iv
DEDICATION v
ACKNOWLEDMENT vi
ABSTRACT vii
LIST OF TABLES xi
LIST OF FIGURES xii
DEFINITION OF TERMS ..... **Error! Bookmark not defined.**
CHAPTER 1 1
INTRODUCTION 1
1.1 Overview 1
1.2 Background and Motivation 1
1.3 Statement of the Problem 2
1.4 Aim and Objectives 2
1.5 Significance of the Project 3
1.6 Project Risks Assessment 4
1.7 Project Organization 5
CHAPTER 2 7
LITERATURE REVIEW 7
2.1 Introduction 7
2.2 Historical Overview 7
2.3 Related Work 11
2.4 Summary 15
REQUIREMENTS, ANALYSIS, AND DESIGN 18
3.1 Overview 18
3.2 Adopted Methodology 18
3.2.1 Data Gathering Method 1 (interview) 18
3.3 Tools and Techniques 19
3.4 Ethical Consideration 19
3.5 Requirements Analysis 20
3.5.1 Functional Requirements Specifications 20

3.5.2 Non-Functional Requirement Specifications 22
3.5.3 System Requirements Specifications 22
3.6 System Design 23
3.6.1 Application Architecture 24
3.6.2 Use case diagram 26
3.6.3 Entity-Relationship Diagram (ERD) 29
3.6.4 Data Design 30
3.6.5 Activity Diagrams 33
3.8.5 Data Flow Diagram 37
3.6.6 Control Flow Diagram 38
3.6.7 User Interface Design 39
Chapter 4 43
Testing and Implementation 43
4.1 Overview 43
4.2 Main Features 43
4.3 Implementation Problems 62
4.4 Overcoming Implementation Problems 62
4.5 Testing 63
4.5.1 Test Plans (for Unit, Integration and System Testing) 63
4.5.2 Test Suite (for Unit Testing, Integration Testing, and System Testing) ....**Error! Bookmark not defined.**
4.5.3 Test Traceability Matrix (for Unit Testing, Integration Testing, and System Testing) **Error! Bookmark not defined.**
4.5.4 Test Report Summary (for Unit Testing, Integration Testing, and System Testing) **Error! Bookmark not defined.**
4.6 Use Guide 68
4.7 Summary 70
CHAPTER 5: DISCUSSION, CONCLUSION, AND RECOMMENDATIONS 71
5.1 Overview 71
5.2 Objective Assessment 71
5.3 Limitations and Challenges 71
5.4 Future Enhancements 72
5.5 Recommendations 72
5.6 Summary 72

REFERENCES.....73
Appendix A - Project Document 76
Appendix B – Interview.....77
Appendix C- Source Codes **78**

LIST OF TABLES

Table 1.1- potential risks faced by the system ……4

Table 2.1 related works summary ……14

Table 3.1 functional requirements of the members ……20

Table 3.2 functional requirements of the executives …… 20

Table 3.3 functional requirements of the admin ……20

Table 3.4 Non-functional requirements ……21

Table 3.5 systems requirements ……22

Table 3.6 Data dictionary for member table ……30

Table 3.7 Data dictionary for admin table ……30

Table 3.8 Data dictionary for loan table ……31

Table 3.9 Data dictionary for activities table ……31

Table 3.10 Data dictionary for loans table ……32

Table 3.11 Data dictionary for registration table ……32

LIST OF FIGURES

FIGURE 2.1- user interface of Co-opify ……..**12**

FIGURE 2.2- user interface of Co-opco …….. **13**

FIGURE 3.1- APPLICATION DIAGRAM …….**24**

FIGURE 3.2 Use case Diagram ……..**26**

FIGURE 3.3 Entity Relationship Diagram ……..**29**

FIGURE 3.4 Activity diagram for login of a user ……..**34**

FIGURE 3.5 Activity diagram for the registration of a user ……..**35**

FIGURE 3.6 Activity diagram for the loan request of a user ……..**36**

FIGURE 3.7 Activity diagram for changing of passwords by a user……..**37**

FIGURE 3.8 Data flow diagram for the registration of a user…….. **38**

FIGURE 3.9 Data flow diagram for the login of a user …….. **38**

FIGURE 3.10 Data flow diagram for the loan request of a user…….. **38**

FIGURE 3.11 Control flow diagram for the registration of a user……..**39**

FIGURE 3.12 loading page …….. **40**

FIGURE 3.13 login page …….. **40**

FIGURE 3.14 registration page …….. **41**

FIGURE 3.15 dashboard page …….. **41**

FIGURE 3.16 loan request page…….. **42**

FIGURE 3.14 Activities page…….. **42**

CHAPTER 1

INTRODUCTION

1.1 Overview

The System once fully developed, will provide a simple interface for the co-operative members to carry out their numerous activities -activities which will be properly covered later in the chapters- providing a secure, easy to use and efficient system with modern software development technologies to promote the saving of funds (at least monthly) among members. In this chapter, we will look at the expectations from the proposed system and a general overview of the problem which required the development of this system.

1.2 Background and Motivation

A co-operative society follows a system where a group of individuals come together and gather resources to reach mutually shared economic and social goals. It has become common practice for such societies to be found in the hearts of workplaces, allowing staff who decide to be members of the co-operative to help each other financially maybe by starting businesses together where profits will be split, or by contributing a certain amount over a certain time frame from which loans can be taken out of.

The organization concerned with this project is Baze University. It is a tertiary educational institution -located at Plot 686 Cadastral Zone C00, Kuchigoro, Abuja- and just like numerous organizations, has a properly functioning co-operative society where any of the staff whether academic or non-academic may join the society to enjoy the benefits it offers its members.

This system will greatly lessen the burden carried by the society's executive team in the case of validating registration or even approving loans; and even those levied upon its members like in the case of requesting for loans.

1.3 Statement of the Problem

At Baze University, we are still trying to fuse technologies with our everyday activities to enhance efficiency and this case is no different; the co-operative society faces a problem since they currently lack a digital platform to perform their numerous activities. The prior statement shouldn't lead to the conclusion that there is no technological structure in place now as there is a digital database that stores the staff-member details. But this is not enough. Members have no way to quickly see an accurate representation of their loan or savings balance at any given time or from any location including from bed when they wake up first thing in the morning; Or even when loans they've taken out are due for payment.

There is also the persistent issue of some requests, be it registration requests, loan requests etc. Not being attended to in the order of which they were made. Members have made complaints of their loan requests remaining unattended to for extended periods while others have already been sorted out; This shouldn't lead to the assumption that staff are incompetent but rather should enforce the idea that there is a crucial need for the system to promote the efficiency and integrity of the co-operative.

Looking at the issues that the executive board face, they aren't always able to stay on top of who has/hasn't made their monthly contributions or keep track of all members and their status. They could be given real time notifications through emails about most activities on the system that would require their attention.

1.4 Aim and Objectives

The aim of this project is to create a system to support the operations of the Baze University Co-operative society; thereby improving its efficiency, reliability, accessibility, and organization.

The objectives of the system-

1. To Allow members of the Co-operative to request loans.
2. To allow executive members to approve/decline loan requests and registration requests.
3. To allow the admin to suspend or activate members.

1.5 Significance of the Project

The completion of the co-operative society system will revolutionize the existing system now for all members of the co-operative society by employing modern technologies to give a safe and enjoyable experience.

Below, the benefits have been outlined-

1. **Faster decision making among the executive board**: an equal amount of voting power is given to each member of the co-operative, and they will be able give their individual votes at any time and from any location on their devices so that when the votes are accumulated, a task may or may not be carried out. The system looks at removing the constraints that may require the executive members to meet physically to give vote (to authorize a process), and the execution of that process (once votes are given, activity is carried out).

2. **Streamline the loan request process:** instead of requiring all members to visit the office to make loan requests, with the new system, members will request loans on the application. Not just can they make requests, but they will also be able to manage their loans (see the loan status which may have been approved, declined or pending. They can also see the deadline for approved loans and how much is left to upset the loan) upon approval or decline, they will be notified through mail.

3. Provide a webapp that allows the admin to have an overview of all contributions, withdrawals, registrations etc. that have been carried out on the system given a certain period. The admin will be able to oversee all the necessary information and activities on the system. The information gathered can be used to make decisions for the growth and improvement of the co-operative.

1.6 Project Risks Assessment

After careful analysis, the possible risks faced by the system are outlined below-

Table 1.1- Risk Assessment

RISK	**RISK MITIGATION**	**IMPACT**
Equipment failure hindering the software development process	The entire folder containing the projects documentation has been backed up to Microsoft's OneDrive. As of this time, the system is yet to be developed; but once the process in ongoing, it will also be backed up to the cloud.	MODERATE
Equipment Loss potentially causing loss of work	The security of the device which the app is being stored on will be ensured; in addition to backing up work both to the cloud and hardware devices (flash drives etc.)	MODERATE
Unavailability of Essential software	During the requirement gathering stage, required software will be identified and its availability will be considered. If not available, alternative means will be explored.	HIGH
Data Security breaches	measures to ensure security of the system and information is accessible only to the individual who owns it will be implemented. The admin does not have access to information that may be used to	HIGH

	identify a particular member, and neither can they modify a member's details.	
Coding issues (low quality code**)**	Appropriate tests will be carried out to ensure usability and efficiency of the system.	HIGH
Risks associated with Law and Compliance	Proper referencing and citation will be made to any work that has contributed in any-way to the completion of this project.	HIGH

1.7 Project Organization

Within this document, we will find material detailing the activities and the processes that led to the development of a web and mobile application to support the activities of Baze University's co-operative society.

The later chapters will cover the designated topics identified below:

Chapter 2: Literature Review

Looks at existing works related to a co-operative society's management system.

Chapter 3: Methodology

This chapter will discuss the tools, techniques and frameworks that have been used in the development of the project. Including the system architecture, flow of activities, system requirements etc.

Chapter 4: Implementation and Testing

In this chapter, details of the application development are defined. The applications code for the front-end and back-end including test cases and how the system performed in these cases will be outline.

Chapter 5: Conclusion

Summary of the project including its areas for potential improvement and key findings.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, we will find the literature review of the techniques used in this thesis. Details on past works that have been reviewed in relation to the project development, we will also look at literature on the technologies use i.e. NodeJS, React Native, MongoDB etc. and how they were used to address the issues raised in the previous chapter. This chapter will also delve into the impact technology has on co-operative societies.

2.2 Historical Overview

Humans have always made major breakthroughs or achieved incredible feats due to the collective effort of some kind of community where each sector plays a crucial role and delivers accordingly, even for large scale systems, the final product is a result of a well working or managed community/society. This can be linked to the popular saying that "No Man Is an Island" coined by John Donne in 1624 and just tries to say that no man is better 4off on his own completely as everyone is a piece of the greater whole which is humanity itself (Seffusatti, 2024).

A co-operative society is built on similar principles, giving its members room to grow and carry out personal projects only possible with the combined efforts of the society. To give the members of the co-operative a digital platform to carry out their activities, this system makes use of proper cloud-based technologies to ensure appropriate features will be available to all members. Over some years, great effort and work have gone towards research on how the use of computers and ICT technologies may be leveraged by co-operative societies in their day-to-day activities including registration of their members, monitoring of loan requests and active loans and all other operations performable within or outside the society. Some researchers aim to have a centralized system that keeps track of the operations of all co-operatives in Nigeria using a centralized system which will reduce the number of hours spent on manual computations of their activities (Olorunlomerue et al, 2017).er

2.2.1 Earliest record of the Co-operative movement (1700s)

Record on co-operative societies dates to 1761 in Fenwick, Scotland during a time when there was a decline in handloom weaving attributed to the sudden rise in addition of machines in the textile industry, the society was established for weavers to support each other to regulate price for the work, with the terms for being in the society including being honest to each other and to their employers (National Library of Scotland, 2023). While the initial cause for the co-operatives establishment was to give a society for weavers to support each other, in November 1769, the members of the co-operative wanted non-members to also benefit from their activities by using funds generated by the society to buy food in bulk that will be sold to both members and non-members but in smaller quantities and a slashed price. This gave non-members a cheaper alternative and gave another income source to the society members as the savings would be split amongst them (Carrell, 2007). This was during a time long before the first computer was introduced, the activities of the society was greatly impeded. The printing press was used to spread information across wide areas while most kinds of transactions would have to be carried out manually, thereby slowing down their operations.

2.2.2 The Technology Boom (1800s)

After a long time of co-operatives running their operations manually, slowly they started to make use of modern-day technologies; like with the use of transport networks to cover wider ground and grow their member base, they also made use of the telegraph to better communicate within the society or facilitate better communication with other societies allowing them to share good practices (Muller & Tworek, 2015). With less need for the use of manpower for some operations, the co-operatives were able to invest more time in their societies to increase productivity and value to both members and non-members

2.2.3 Emergence of Online Platforms (20th Century)

With the introduction of web-based technologies in the 20th century, the co-operatives were only strengthened further. With online applications making it easier for members of the societies to collaborate and communicate by abolishing the barrier of geographical location (Wegner et al., 2023).

2.2.4 Interactive Collaboration

This advancement also meant that the societies were able to cut costs in some area by leveraging on the use cases that the platforms come with. There was less need for paper works, physical meetings as they could be hosted online, and other dated methods of communication and forms of management (Atanasova et al, 2024).

2.2.5 Increased Technology Accessibility (Late 2000s - Present)

With the launch of the Apple app store and the ever-growing popularity of web applications, it marked a turning point (Goray, 2024). Co-operatives are now more equipped with technology that allows all their members to perform operations on the go. Smartphones have now become a core part of our everyday lives with almost everyone owning one, with about 7.2billion smartphones in circulation (Howart, 2024) there is a high chance each member has access to a smart phone. This means systems that support the operations of the co-operative can easily be loaded and accessed by members on their smartphones at almost any time. The systems are also more usable by members with special needs with features like text-to-speech, screen reading, and even language translation.

Another area where the use of technology shines is with the administrator's management of the society. The admin may now better track the activities within the cooperative thereby improving transparency and accountability from all members (Sobolev et al, 2023).

All these advancements in technology have made it possible to develop mobile and web applications that use cutting-edge cloud computing to offer interactive and responsive applications available across different platforms.

The technologies used in this work include:

1. **React:** React is a popular JavaScript framework that employs Webpack to compile React, JSX and ES6 code and handles CSS file prefixes. It is mostly used because of its easy creation of dynamic applications which greatly supports code reusability with their components that can be custom made and reduces overall development time, it's improved performance with Virtual-DOM that updates related items when states in the browser are updated (Deshpande, 2024).

2. **Cascading Style Sheet (CSS)**: CSS was developed by the world wide web consortium(W3C) and it's used to style elements written in markup languages like HTML. This allows developers to create visually appealing interfaces (Domantas, 2023).

3. **React Native:** React Native is a JavaScript mobile app framework that allows development of mobile applications for IOS and Android. React Native was built on React with several similarities allowing developers who are familiar with JavaScript and React to quickly pick it up (Budzinski, 2024). It has been used for the development of mobile applications and has been hosted on expo go which is a cross-platform application for building mobile applications for IOS and Android.

4. **MongoDB:** MongoDB is a NoSQL database management program that is used as an alternative to traditional databases (Gillis, 2023). MongoDB is popular among developers because of the NoSQL format that appears very similar to JavaScript objects therefore giving developers a familiar feel as the same concepts can be applied to them.

Some features of MongoDB –

1. **Replication:** replicas in MongoDB are set of processes that are used to reduce redundancy and to increase availability by creating copies of the data onto a server (Gillis, 2023).
2. **Scalability:** with horizontal scaling which allows you to add servers to your cluster, and vertical scaling which aims to add more power to a machine, MongoDB benefits large systems with high traffic (Saini, 2024).

3. **Load balancing.** MongoDB uses software to dissipate network traffic without a hardware load balancer because of its support for vertical and horizontal scaling.
4. **Schema-less:** MongoDB uses a scheme-less database meaning that one collection may hold different kinds of documents in it, the documents stored in the same collection must not have the same properties or size (Saini, 2024).
5. **Document Oriented:** MongoDB stores records in key-value pairs as it's a No-SQL database. This is a more familiar with developers as records are represented inn JSON which many developers are more familiar with this structure.

2.3 Related Work

Some individuals have been able to complete the development of digital systems to support co-operative societies in Nigeria as most existing systems support foreign accounts. Some of these systems include, (Olorunlomerue et al, 2017) who proposed a web-application system for handling records of cooperative societies in Nigeria. They used Java and MySql to build the system which was meant to be used to register co-operative societies in Nigeria to enable the Government to collect data on co-operatives for proper planning.

Still in 2017, (Onyeama et al, 2017) completed the development of a system designed to manage short-term and long-term loans. The system also kept tract of the inflow and outflow of cash within and outside a cooperative society. To build the system, they used CSS and HTML at the front end and PHP and MySQL for the backend. The project was said to allow investors to easily retrieve relevant information about the cooperative. In 2020, (Yusuff et al, 2020) embarked on a study to investigate how ICT in its appraised form was used in co-operative societies. The study included some tertiary institutions in Osun State but did not include the development of any solution or application for the societies. Research that was conducted in Indonesia by (Andreansyah & Rizkiana, 2020) took on the development of a web application that was to keep track of employees' salaries in the co-operative using the incremental model.

(Oluyombo, 2013) took interest in how co-operative society loans met the financial needs of their members in rural areas, while his research did not seek to find ways ICT would solve co-operative loan management, it further highlighted the issues with managing the loans manually and can be used to see the crucial need of a digital revolution in the area.

Meanwhile, (Mbam & Igboji, 2013) took on the project of developing a system that will aid a co-operative society's loan management process including short-term and long-term loans. MySQL server was used for the database with Visual Basic net framework on the frontend. Overall, the system failed because investors failed to see an area where the system was operational.

A potent research and project were conducted by (Caroline, 2018) where a loan cooperative information system in the form of a web application was developed for Cempaka cooperative. The project aimed to build an information system using a rapid prototyping, PHP and MySQL were used on the backend with HTML on the front-end. The finished system consisted of numerous capabilities including, savings and loan transactions, loan instalment transactions, cash withdrawal transaction etc. Another notable project was completed in 2021 (Moechammad et al, 2021) where the authors developed an android application to manage the savings and loans of a cooperative to enable its members to save and raise capital for the cooperatives business capital. They took on the project because it was found that while technology plays a crucial role on storing and processing data, cooperatives in Indonesia were yet to leverage on this advantage. The society the application was built for "Permata Ngijo Savings and Loans Cooperative" consisted of 30 active members and was run by several executives. The development of the project followed a prototype system developer method where the Unified modelling language (UML) was used to design the system, MySQL was used for the database structure with a black-box testing for system testing and PHP for the backend.

(Pane, 2019) took on a project to create an information system for a savings and loan cooperative system in the form of a web-application for the Pancuran Hidup Credit Union. The system employed the waterfall model to create a savings and loan information system that allowed the employees to manage their loan and savings data making it easier to report more accurate data to them.

Co-opify

Coopify is a web-based application that aims to provide a system that meets the needs of different co-operatives societies

Features

1. There are numerous features on the system. from adding members, requesting for loans and adding savings.
2. The admin has an overview of the system activities and directly manages the system.
3. Broadcasts can be made to third party apps like WhatsApp to alert members on any events.

Limitations

1. There is no login for the members as the admin must perform actions on the members behalf.
2. There are a lot of features that are very niche and unnecessary to most co-operatives, making the system seem clustered with unnecessary features.
3. The admin must manually send messages on the supported third-party applications to send messages to other members.
4. Nigerian Prominent banks do not seem to be supported as some didn't come up during registration.
5. Since the system requires you to fill in all bank information relating to the co-operative, it poses great financial risks.

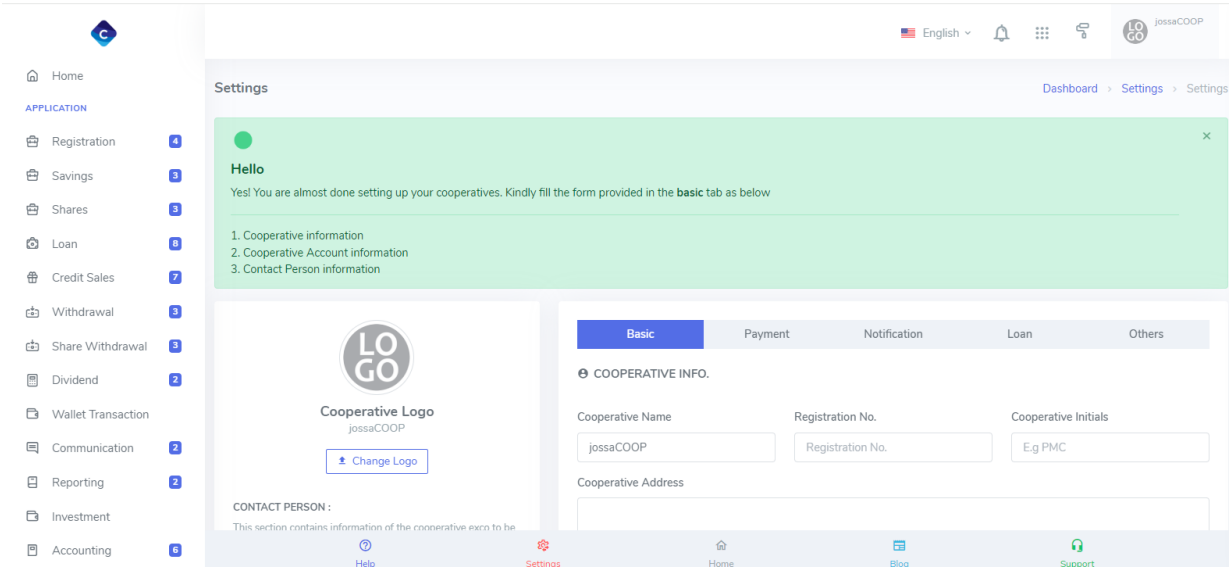


Figure 2.1 User Interface of Co-opify

COOPco

Features

1. Loan Requests
2. Said to support member registration

Limitations

This system does not support any free trials with the lowest tier starting at $10 (US Dollar)/month. The registration process for a live demo kept falling through and some input fields accepted wrong inputs e.g. a phone number field accepting string characters. All these show signs of a poorly managed system.

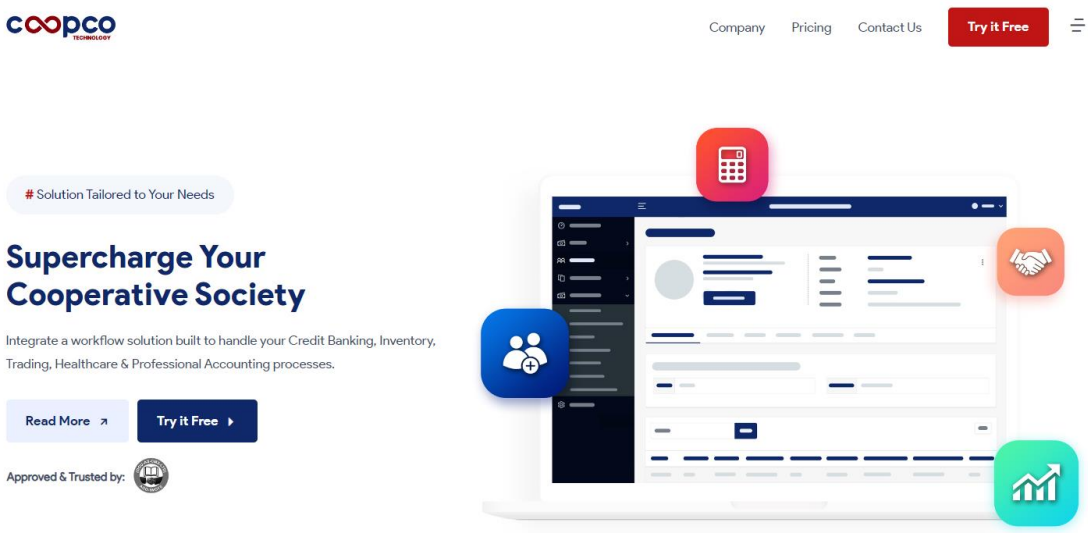


Figure 2.2 User Interface of COOPco

Electric Cooperatives FCU (mobile application)

Features

1. Allows users to see their account balance at any time.
2. Users can view and pay bills
3. Users can see recent transactions

Limitations

1. Only foreign banks seem to be supported on the application.
2. It is more of a personal banking application than suited for a co-operative society.

2.4 Summary

This chapter showed the review process of the literature regarding the numerous technologies and resources used through the development process. The Historical overview shows how co-operative societies have been able to evolve over the years mainly to the rapid development of technology.

The cooperative society being developed takes these issues into account and strives to provide solutions that are tailored to the needs of the co-operative society of Baze University.

Chapter 3 will delve into the requirements analysis and the methodology that has been used to solve the stated issue, including the provision of diagrams like the Use case, Activity. ERD etc.

Table 2.1 related works summary

S/N	App Name/Authors name	Features	Limitations
1	Co-opify	Allows, requesting for loans and adding savings. Admin directly manages the system. There is provision for communication with members.	Admin performs activities on behalf of members. Oversaturation of features that are not necessary to a cooperative. Communication with members could be easier.

2	Co-opco	Loan Requests Said to support member registration	Poorly managed system Does not offer a free version to test the app
3	Electric Cooperatives FCU	users can see their account balance at any time. Users can view and pay bills Users can see recent transactions	The system is more suited as a personal banking application and doesn't allow collaboration among members.
4	(Olorunlomerue et al, 2017)	Registration of co-operatives on a national level	The system wasn't scalable enough to store data on the whole country
5	(Onyeama et al, 2017)	Management of short- and long-term loans The developed system will monitor the cash outflow and inflow of the co-operative.	There was high Complexity in monitoring all loans
6	(Yusuff et al, 2020)	Research on impact of ICT to co-operative societies	The developed system didn't provide any solution to the raised issues
7	(Mbam & Igboji, 2013)	Management of short- and long-term loans	A good Use case for system was failed to be seen by investors

8	(Caroline, 2018)	Development of a web app to monitor savings and loan transactions, cash withdrawals	The project could only be used by the specific organization
9	(Pane, 2019)	Allows members to track their loan and savings information for easier data reports	Scope creep led to the project to not be finished
10	(Andreansyah & Rizkiana, 2020)	Developed system kept track of employee's salaries	The developed system did not provide enough technical features to meet the demands of the co-operative
11	(Moechammad et al, 2021)	The system kept track of the savings and loan activities for the cooperative	The system could not perform any other activities apart from keeping track of the financial activities. Meaning that other operations like registrations were still manual.

CHAPTER 3

REQUIREMENTS, ANALYSIS, AND DESIGN

3.1 Overview

This chapter will give an overview of the analysis carried out on the system to gather the requirements for the system (including the methods to gather these requirements). The methodology with its required steps will also be discussed. A general view of the design of the proposed system should be expected.

3.2 Adopted Methodology

The development of the proposed project adopted the Rapid Application Development model. This methodology was chosen due to its huge emphasis on user involvement during the software development process, which helps to reduce the risk of project failure. The requirements will be properly gathered, but during the development process, the developer doesn't follow a definite set of requirements, instead they create multiple prototypes as fast as they can and show it to the client to identify what they like and what they don't ("What is Rapid Application Development (RAD)? An Ultimate Guide for 2024", 2024).

3.2.1 Data Gathering Method 1 (interview)

Interviews with some staff of Baze university were conducted; not limited to those who are in the co-operative society but also those who are not but may be considering joining. This can help me identify areas which if improved will encourage them to join the institutions co-operative.

The details and findings on the interview and questionnaire will be covered properly in the Appendix.

3.3 Tools and Techniques

To ensure the successful development of the application, the use of various existing programming tools and techniques have been employed; they will help in various stages including but not limited to the creation, testing and maintenance of the application. These tools include-

1. **React JS** - this is a JavaScript framework that is used for front-end web development. It has been used for the web application which provides an interface for the admin to oversee certain activities performed in the co-operative.
2. **Node JS** - this is an open-source and cross-platform JavaScript runtime environment that allows developers to build both front-end and backend applications (Semah, 2022). For this application, it will provide functionality and connection to our database and some API calls.
3. **React Native -** is also a JavaScript framework that supports front-end development. It will be used to develop the mobile which all members of the co-operative will interact with.
4. **MongoDB-** it is a NoSQL database manager, NoSQL in the sense that it does not have relational databases as opposed to a database manager like MySQL (Gillis,2023).

A combination of these various technologies allowed me to develop an efficient and reliable app that significantly improves the lives of Baze University staff (especially those in the co-operative).

3.4 Ethical Consideration

To guide the research design and patterns, the project development will follow some set of principles which will enforce some trust and reliability in the application.

1. **Integrity**: All finding, and facts stated in this document will be researched on to allow users -or anyone related to the project- have confidence in the facts stated. Also, effort will be made to fulfil promises made concerning the project and reasons will be given if some features can't be implemented. Data displayed from the system will not be tampered with to avoid any form of false representation; this reflects the transparency of the system. Proper technologies and tools will be implemented to ensure the security and integrity of data to protect it from outsider attacks; measures will also be taken to minimize the risk of insider techniques.

2. **Intellectual Property**: all algorithms will belong to the developer unless stated otherwise; in which case appropriate recognition will be made to the authors in form of referencing and Citations.
3. **Confidentiality**: All records stored on the system will only be available to authorized individuals. Each member only has access to their own data and the admin will only see certain data across-board. Data that is considered highly sensitive isn't shown to the admin. No data will be involved in any form of external data mining and will only be used in analysis that can improve the co-operative society at large. In the occurrence of an event where user data is required, the consent of every member involved will be requested for which is in-line with the Nigerian Data Protection Regulation Act.
4. **Non-discrimination**: All members are treated equally and enjoy the same rights and privileges. Certain activities will only be performable only by the executive and the admin which is in line with their activities/responsibilities within the society and in no way will they be referred to in higher regards than other members within the system. There will also be no bias in any area of the application development or decision making; taken for example the system design or data analysis. In fact, research will be made to implement features that will provide a pleasurable experience for members with some disabilities.
5. **Legality**: all activities carried out on the system are legal regarding the 1999 Nigerian Constitution and other Law-making bodies. An example is the protection of users' data governed by NDPR (Nigerian Data Protection Regulation).

3.5 Requirements Analysis

3.5.1 Functional Requirements Specifications

Table 3.1 Functional requirements of the members

Req. No	**Description**
R-1	The system should allow users to view their savings balance and loan balance.
R-2	The system should allow users who have taken out loans should see the deadline for the loan anytime.

R-3	The system should allow users to request for loans; they should always see the status of the loans and get an email in a case where the loan is either approved or denied.
R-4	The system should allow users to log in/Sign out.
R-5	The system should allow users to register; giving only minimal information.
R-6	The system should notify users when their request to join the co-operative (after registering) has been approved or denied.
R-7	The system should notify users when a payment is verified or not.
R-8	The system should allow users to modify SOME details related to their account; like a password.
R-9	The system should allow users to see all activities performed on their account.

Table 3.2 Functional requirements of the executives

Req. No	**Description**
R-101	Each executive should be given equal right to vote in cases like -approving loans, approving registration, exemption from a month's contribution etc.
R-102	Executives should be notified immediately requests are made by other members.

Table 3.3 Functional requirements of the admin

Req. No	**Description**
R-201	They should be able to see all members in the co-operative including active, suspended, and ex-members.
R-202	They should be able to suspend a members account
R-203	They should be able to see the total amount contributed
R-204	The admin should be able to update payment made by members.

3.5.2 Non-Functional Requirement Specifications

Non-functional requirements are those requirements that are necessary to be met to enhance the end-user's experience. To do this, the following criteria will be met-

Table 3.4 Non- Functional requirements

Req. No	**Description**	**Type**
R-301	The system shouldn't quit randomly after the user opens it and must stay active until closed or some hardware failure.	performance
R-302	Users should be able to use the system at any time of the day if they have internet connectivity.	Availability
R-303	The system should give results quickly if the result doesn't rely on user input.	Performance
R-304	The system should only provide data and prompts associated to an account to whoever has authorized access to the account.	Authorization
R-305	The system should be properly run across numerous smartphones (for the mobile app) and laptops/desktops (web application).	Portability
R-306	The system should CPU and memory resources in an effective manner.	Efficiency

3.5.3 System Requirements Specifications

Table 3.5 system requirements

Req. No	**Description**	**Type**
R-403	The system should safely store members data	Security/data Encryption

3.6 System Design

The components of the system are:

1. **ADMIN:** the admin is a key player in this system. They oversee most of the activities performed by the members. They can see the active members, ex-members, and suspended members, and payments under a members account, and they can suspend members or even activate a suspended members account.
2. **REGISTER:** this page could be seen as a sign-up page, but the staff are really creating a whole new account; instead, they will provide minimum information that may be used to identify them including - the staff ID, first name and last name, and their email address- and they would've successfully applied when this process is completed.
3. **LOGIN**: Existing members of the co-operative will be able to login with their existing record if the data given exists.
4. **HOME PAGE:** the home page will give a brief overview of any updates to the user. The user's name will be displayed, the savings and loan balance will be showed in the same field. Information about any upcoming/on-going events or any deadlines (for loan applicants) will be shown below the user's balance. And finally, a summary of the last few activities will also be shown with the option to see the full list of activities.
5. **NAVIGATION BAR:** at the bottom of the screen there is a navigation bar that gives quick access to key features of the app from almost every page. There are 4 quick links available, and they are- home, loan requests, activities, and profile.
6. **DROP-DOWN MENU:** the drop-down menu should provide links to other pages that were not included in the navigation bar. Some of the links will be to see your loan status, for the executives; they'll be able to see loan and registration requests made by other members, logout etc.
7. **PROFILE PAGE:** members will be able to see an immutable list of details related to them. The only detail they'll be able to change is their password which can be changed from settings.
8. **LOAN PAGE:** the results on this page will differ depending on if the user is a loan applicant or not. If the user has an active loan, then they will see details on the current loan

including the deadline, amount paid etc. but if not, then the user is shown the page to request for a loan.

9. **SETTINGS:** in the settings, the user can change their password where they must provide their current password correctly and then confirm their new password before the change is effective. This change will be reflected under the account's activities.
10. **ALL ACTIVITIES/TRANSACTIONS: t**hese pages should show details on all the verified activities and transactions carried out on the account.

3.6.1 Application Architecture

The application diagram shows the assembly of an application and how the different components of the system interact with each other with the end goal to meet users' needs (Gavin & Ferguson, 2024). Refer to figure 3.1 to find the application diagram.

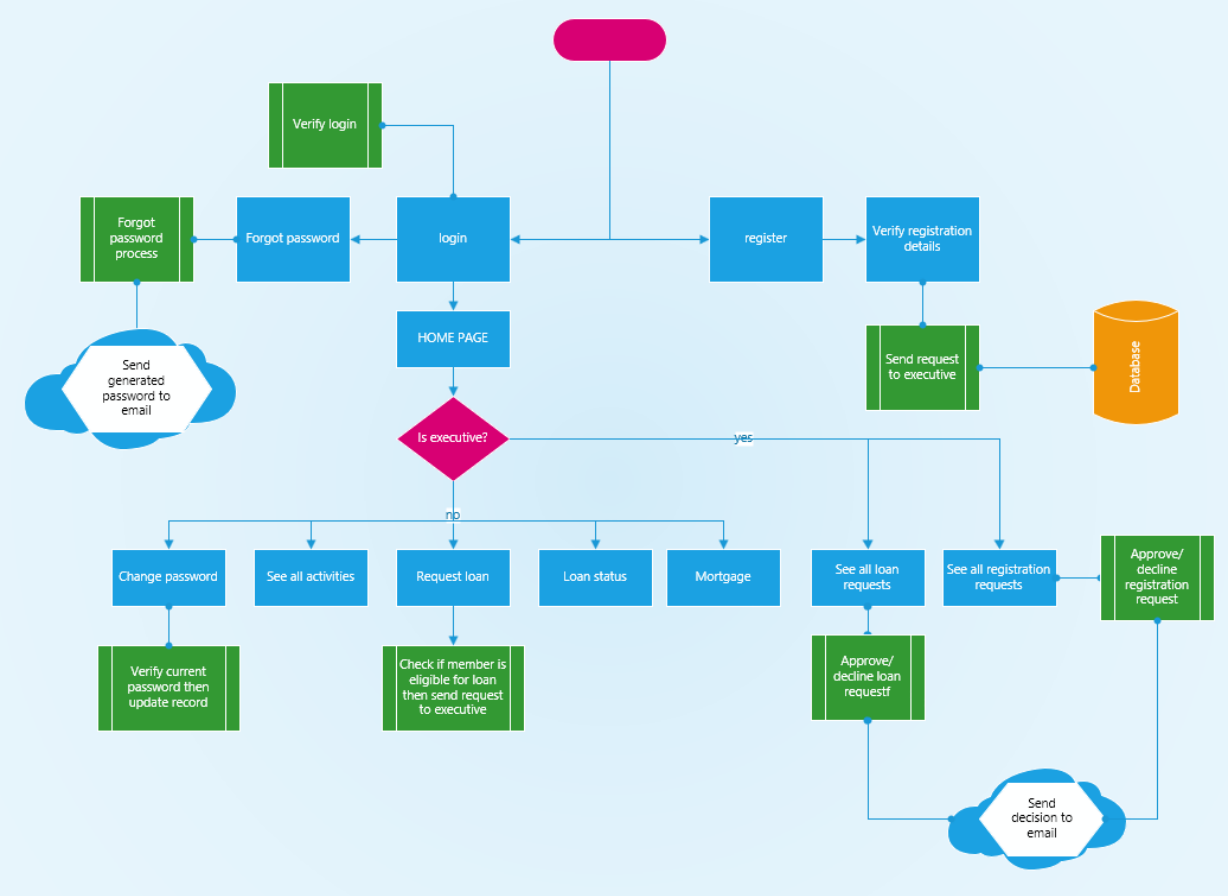


Figure 3.1- Application Diagram for mobile application

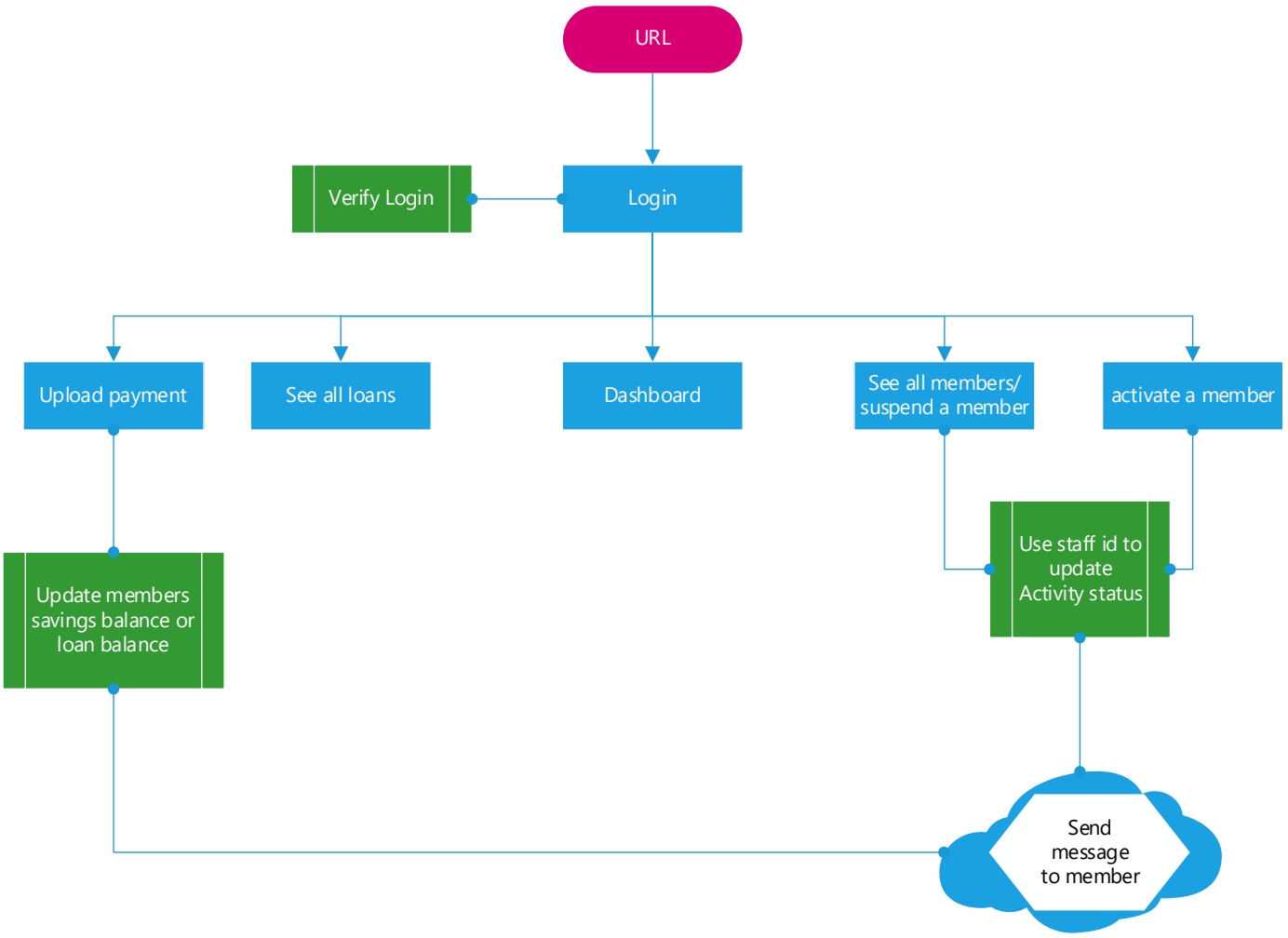


Figure 3.2- Application Diagram for web application

3.6.2 Use case diagram

From figure 3.3, that the actor labeled ‘executive’ is also a member but can perform an extra set of functions that other members cannot perform.

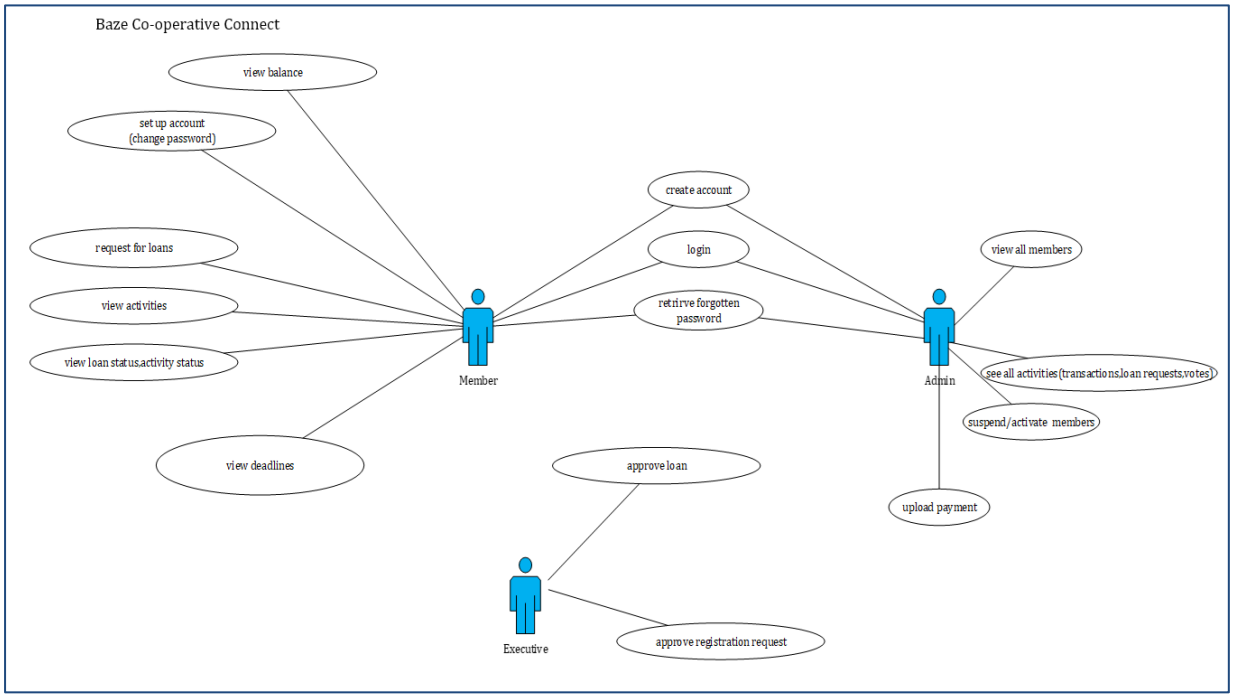


Figure 3.3- Use case diagram for members and admin

Table 3.6 - Use case description for loan Request

Use case	Request loan
Description:	Entails the loan request process performed by members
Actor	Member
Preconditions	Member is logged in Member has at least 5000 in savings Member has paid part of their active loan
Post Conditions	Confirmation message or failure message is displayed

Main flow	**User** 1) user selects loan requests 2) user enters loan details 3) user submits form	**System** 4) system checks if the user balance is above 5000. 5)system checks if other conditions are met to request for a loan. 6)system either displays confirmation message and sends a request to the executive, or it displays a failure message.
Exception condition	No details entered in the form or invalid characters entered.	

Table 3.8.2 - Use case description for View Loan Status

Use case	View loan status	
Description:	Description of a member viewing their loan status	
Actor	Member	
Preconditions	Member is logged in	
Post Conditions	Details of active loan displayed	
Main flow	**User** 1) user selects loan status	**System** 2) system searches the active loans collection for loans with the staff-id 3)system displays all results

Exception condition	No details entered in the form or invalid characters entered.	

Table 3.8.3 - Use case description for uploading payments

Use case	Uploading payments	
Description:	Description of the use case for the admin uploading a member's payment	
Actor	Admin	
Preconditions	Admin is logged in Staff-id is valid	
Post Conditions	Savings or loan balance of Staff will be updated	
Main flow	**User** 1) Admin selects upload payments 2) Admin enters staff id and paid amount	**System** 3) system searches the database for if the member has a loan or not 4) If there is a loan, the loan balance is updated, else the savings balance is updated
Exception condition	Staff ID doesn't exist	

Table 3.8.4 - Use case description for registration approvals

Use case	Registration approvals
Description:	Description of the registration approval/rejection of members
Actor	Executive member
Preconditions	Executive member is logged in

Post Conditions	Registration of member is either approved or declined	
Main flow	**User** 1) Executive member selects registration Requests 2) member selects either approve or decline	**System** 2) system finds registration request, if approved, create a new member account. 3)send email to member
Exception condition	Member email is incorrect	

3.6.3 Entity-Relationship Diagram (ERD)

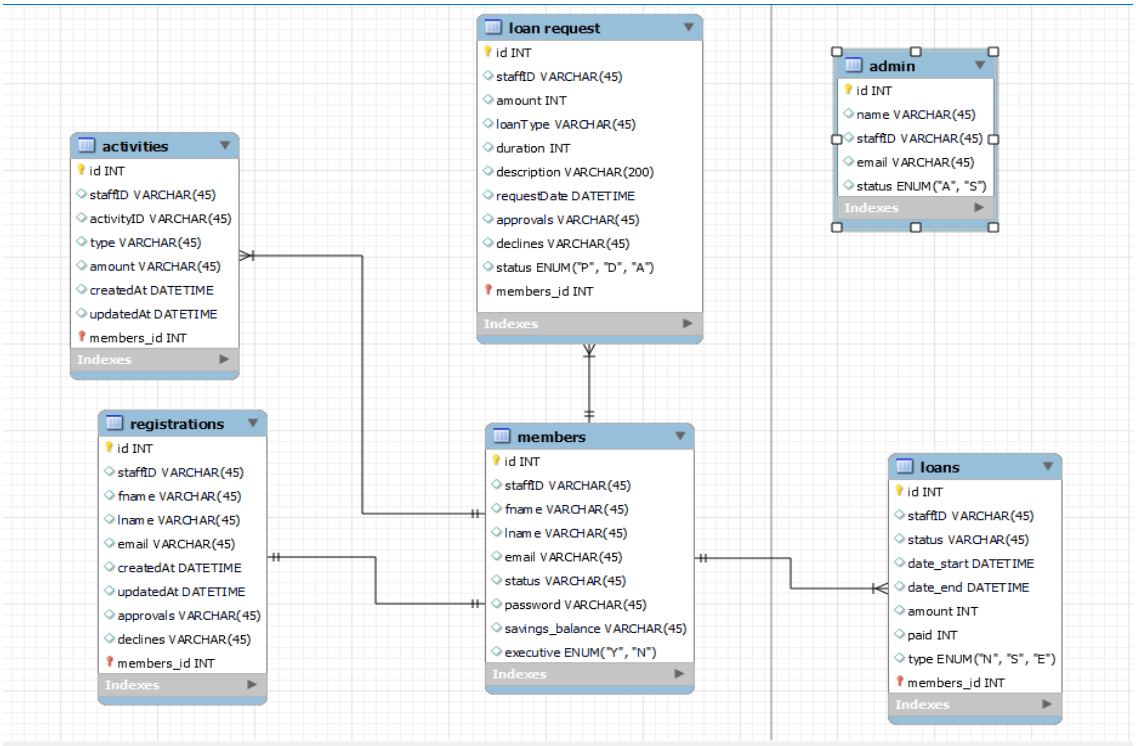


Figure 3.4 ERD diagram

3.6.4 Data Dictionary

Table 3.6 Data Dictionary for Member Table

Column	Type	Range	Required	PK/FK
staffId	INTEGER	45	Y	PK
fname	VARCHAR	45	Y	-
lname	VARCHAR	45	Y	-
email	VARCHAR	45	Y	-
status	ENUM('S','A','E')	-	Y	-
password	VARCHAR	45	Y	-
savings_balance	INTEGER	-	Y	-
phone_number	VARCHAR	45	Y	-

For the 'status' attribute, its type is an ENUM which is short for enumeration and allows you to the specify certain values that the column can take as values. Here there is the ENUM 'S','A','E'. where-

1. 'S'- signifies that a member has been suspended.
2. 'A'- signifies that a member is active.
3. 'E'- signifies that a member is now an EX-member.

NOTE: This set of values will be used in the structure of other tables.

Table 3.7 Structure of data collected for Admin.

This table collects relevant data on the Admins of the Co-operative.

Column	Type	Range	Required	PK/FK
staffId	INTEGER	45	Y	PK, FK
email	VARCHAR	45	Y	-
status	ENUM('A','T')	-	Y	-
password	VARCHAR	45	Y	-

The ENUM with values 'A','T' is described as-

'A'- the admin is currently active in the co-operative.

‘T’- the admin has been terminated.

Table 3.8 Structure of data collected for Loan.

Column	Type	Range	Required	PK/FK
id	INTEGER	-	Y	PK
amount	INTEGER	-	Y	-
status	ENUM(‘A’,’C’)	-	Y	-
paid	INTEGER	-	Y	-
date_start	DATETIME	-	Y	-
staffID	INTEGER	-	Y	FK
duration	INTEGER	-	Y	-
Date_end	DATETIME	-	Y	-

the attribute “status” has type ENUM(‘A’,’C’) where-

1. ‘A’- means active, meaning that a loan is active (it has been approved and has not been paid off).
2. ‘C’- means completed, meaning that a loan is completed (it has been fully paid off).

Table 3.9 Structure of data collected for activities.

Column	Type	Range	Required	PK/FK
activity_id	INTEGER	-	Y	PK,
date	DATETIME	-	Y	-
staffId	INTEGER	-	Y	FK
type	ENUM(‘A’,’L’,’R’,’P’)	-	Y	-
amount	INTEGER	-	Y	-

the attribute “type” has type ENUM(‘T’,’L’,’R’,’V’) where-

1. L- means loan request.
2. R- means registration.
3. P- means password changed.
4. A- means

Table 3.10 Structure of data collected for loan Requests.

Column	Type	Range	Required	PK/FK
activity_id	INTEGER	-	Y	PK, FK
amount	VARCHAR	45	Y	-
status	ENUM('A','D','P')	-	Y	-
description	VARCHAR	45	Y	-
date_end	DATETIME	-	Y	-
staffId	INTEGER	-	Y	FK
loanRequestcol	INTEGER	-	Y	-
duration	INTEGER	-	Y	-
approvals	INTEGER	-	Y	-
type	ENUM('N','E')	-	Y	-

For the "status" column, it takes values 'A','D','P' where-

1. 'A'- means that the loan has been approved.
2. 'D'- means that the loan has been declined.
3. 'P'- means that the loan is pending.

The "type" column shows what kind of loan is being requested either 'N','E' which represent "Normal" and "Emergency".

Table 3.11 Structure of data collected for registration Requests.

Column	Type	Range	Required	PK/FK
activity_id	INTEGER	-	Y	PK, FK
staffID	INTEGER	-	Y	FK
Date_end	DATETIME	-	Y	-
approvals	ARRAY	-	Y	-
declines	ARRAY	-	N	-
status	ENUM('D','P','A')	-	Y	-

For the "status" column, it takes values 'A','D','P' where-

1. 'A'- means that the members registration has been approved.

2. ‘D’- means that the members registration has been declined.
3. ‘P’- means that the members registration is pending.

3.6.5 Activity Diagrams

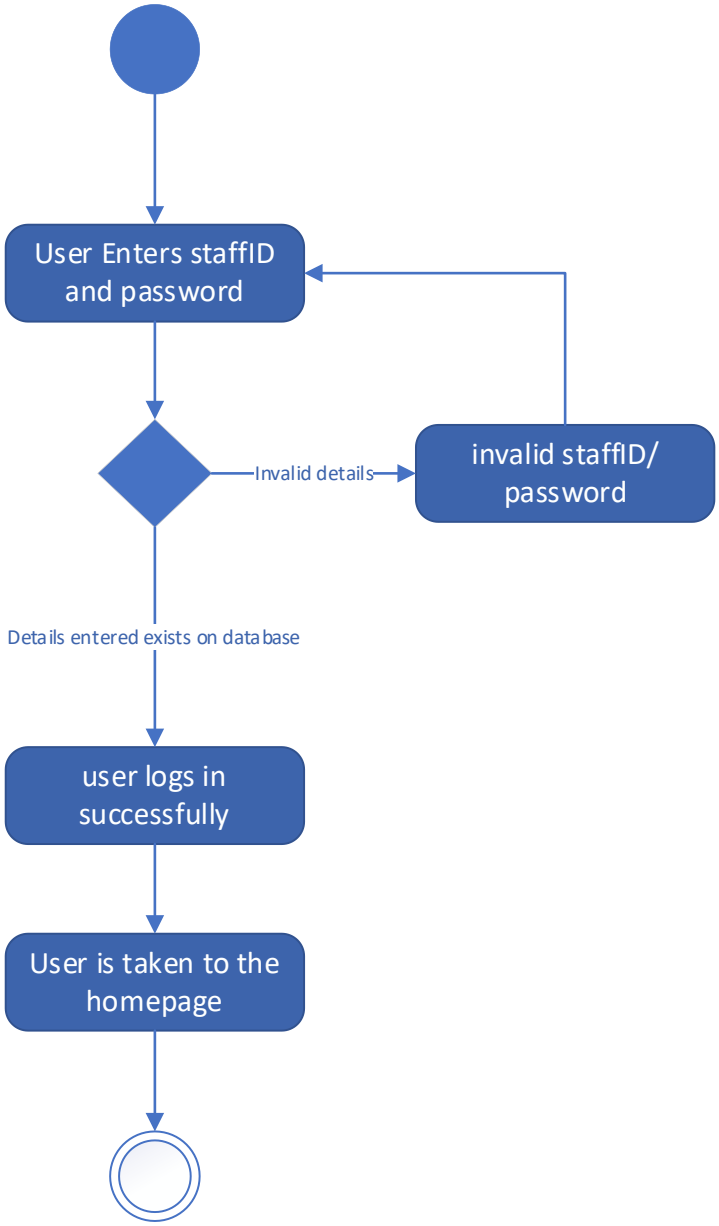


Figure 3.5 - activity diagram for a member’s login.

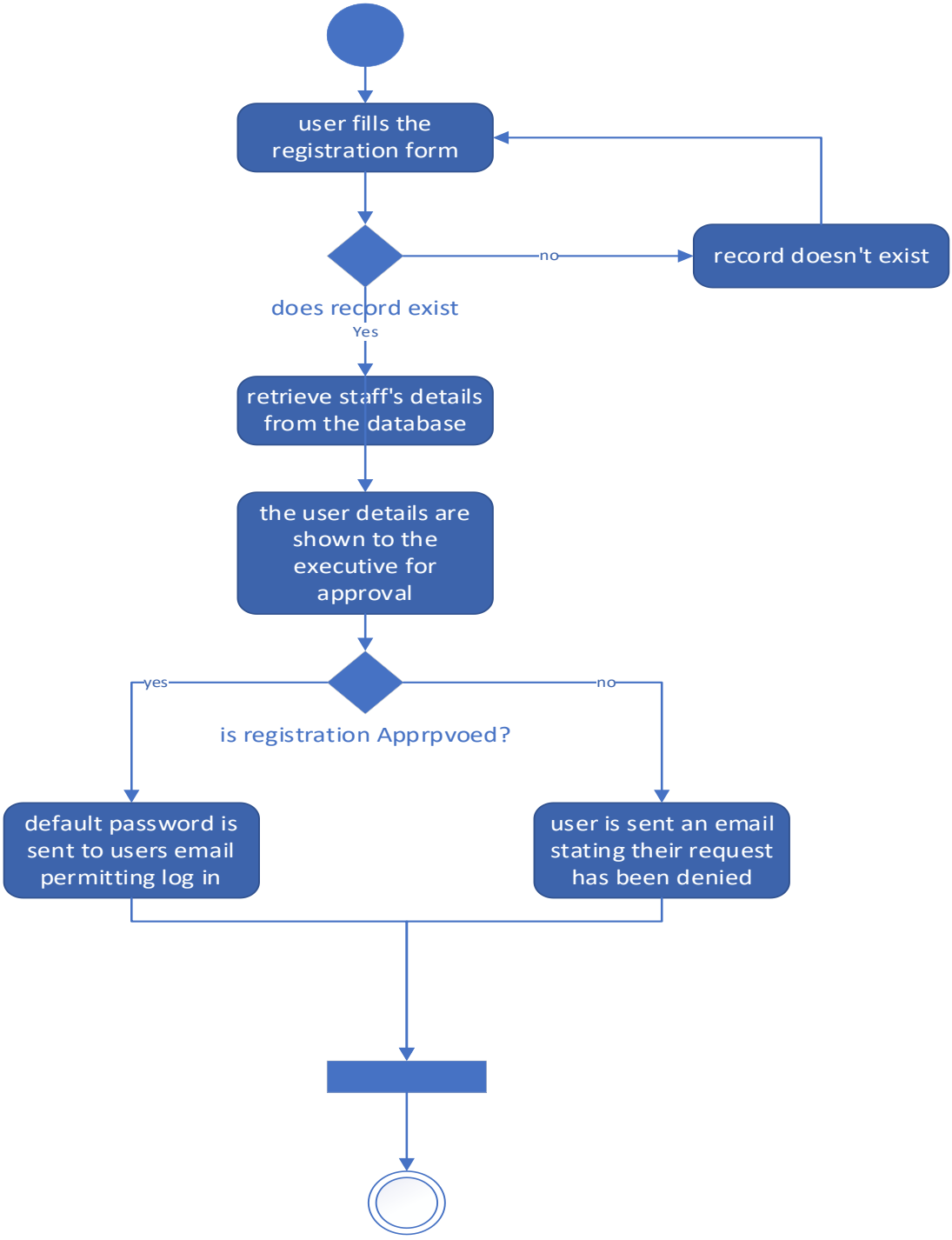


Figure 3.6- activity diagram detailing the registration process of a user.

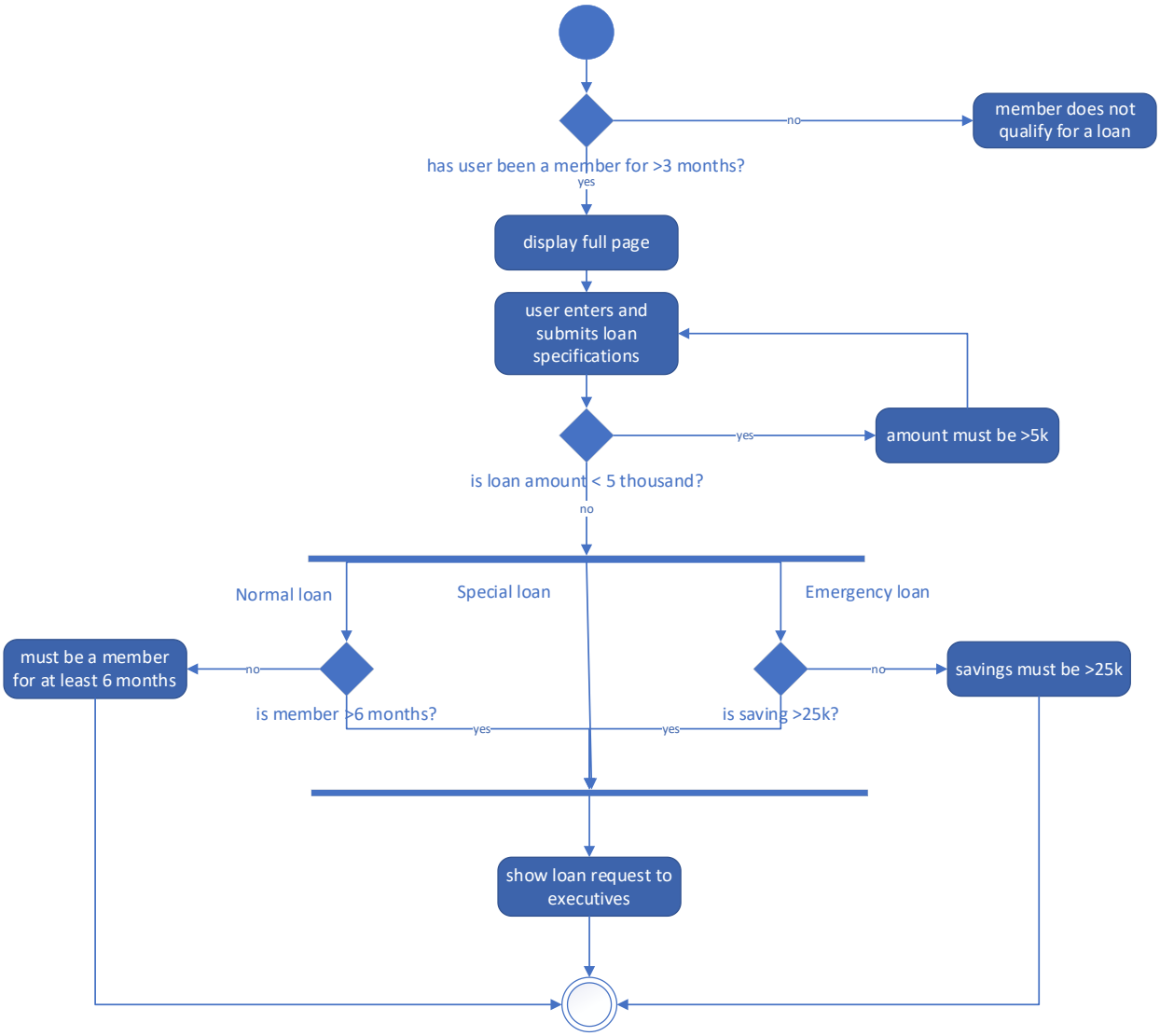


Figure 3.7 activity diagram detailing the loan request process by members.

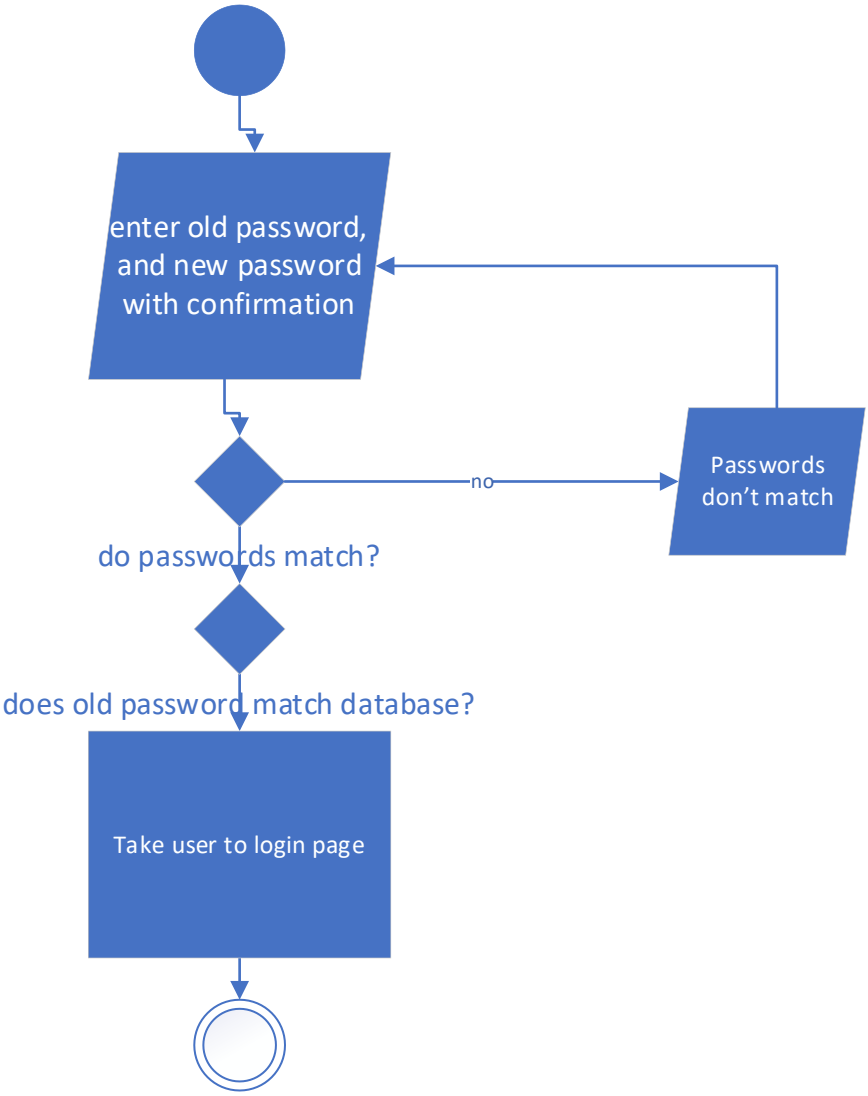


Figure 3.8 activity diagram detailing the changing of passwords by users.

3.8.5 Data Flow Diagram

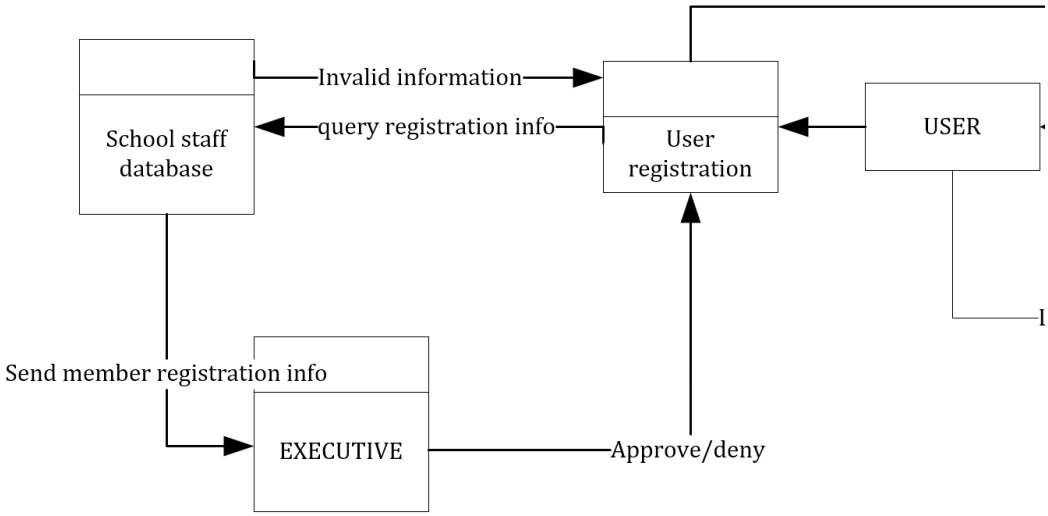


Figure 3.9- data flow diagram for user registration

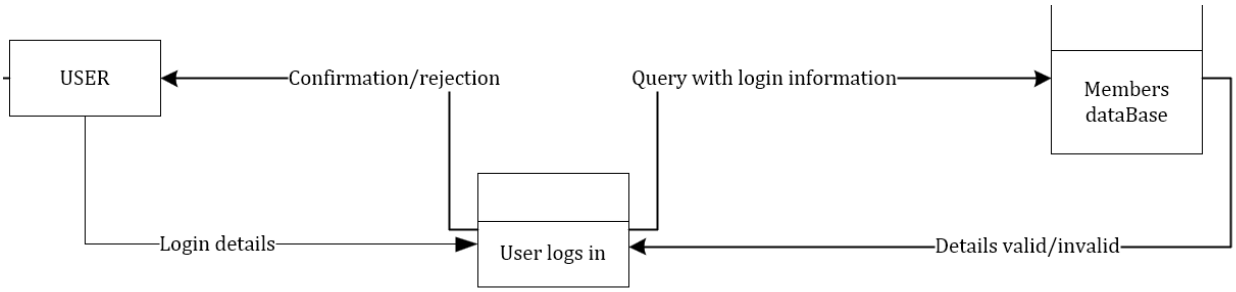


Figure 3.10 – data flow diagram for user login

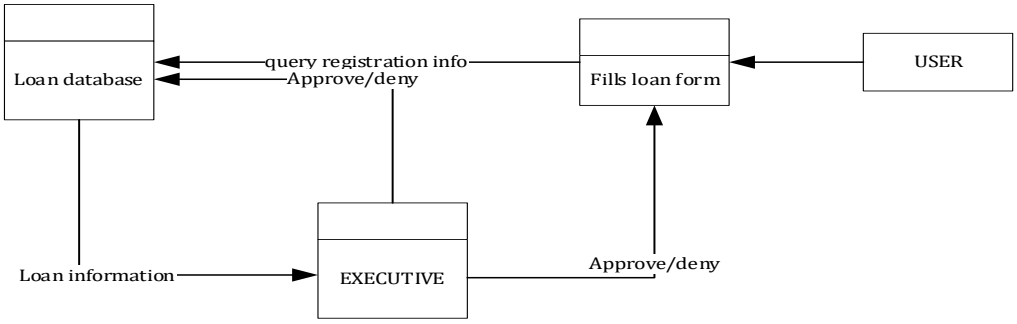


Figure 3.11- data flow diagram for loan request process

3.6.6 Control Flow Diagram

The control diagram should depict how certain actions alter the application's flow of execution.

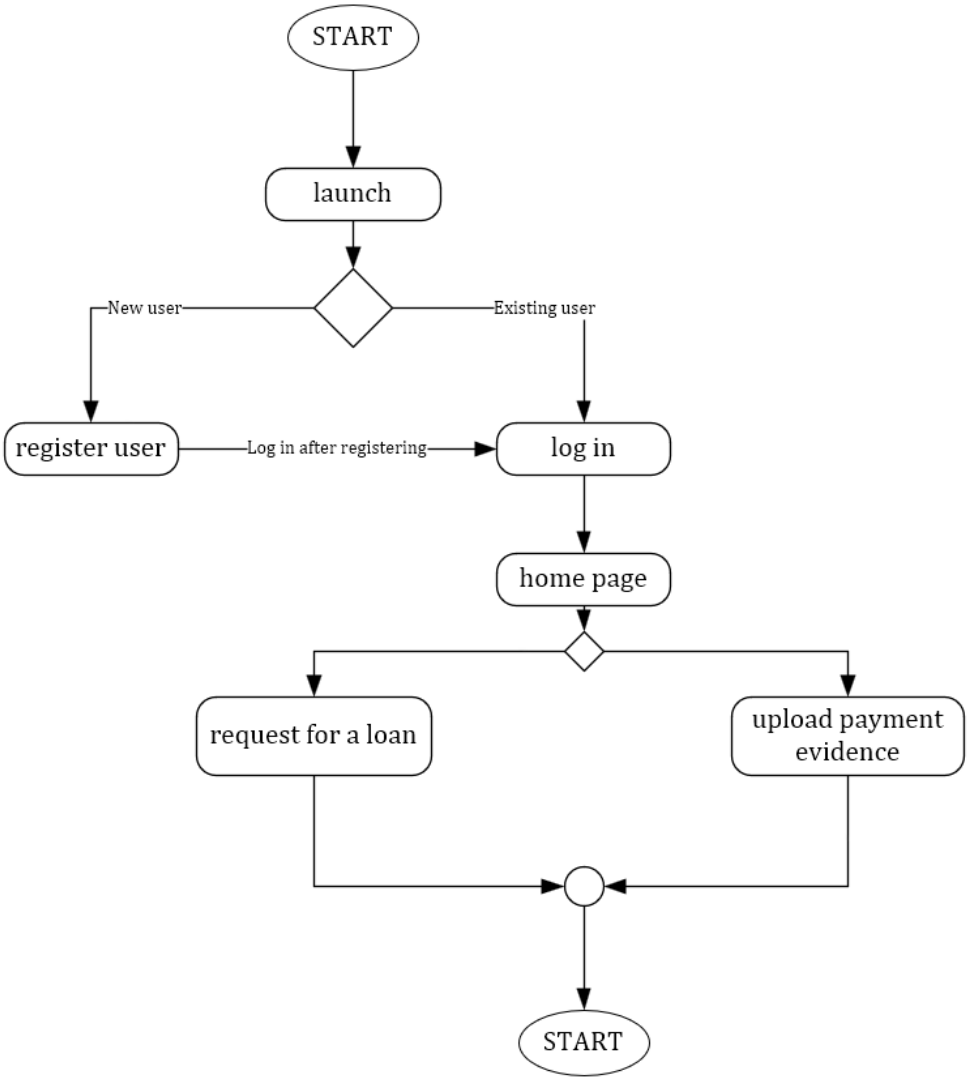


Figure 3.12- the figure below shows the control flow diagram.

3.6.7 User Interface Design

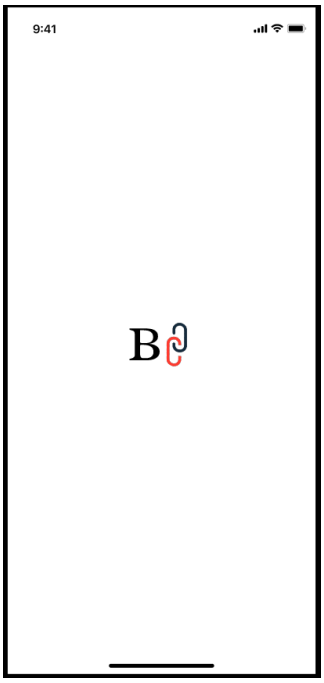


Figure 3.13 mobile application loading page below.

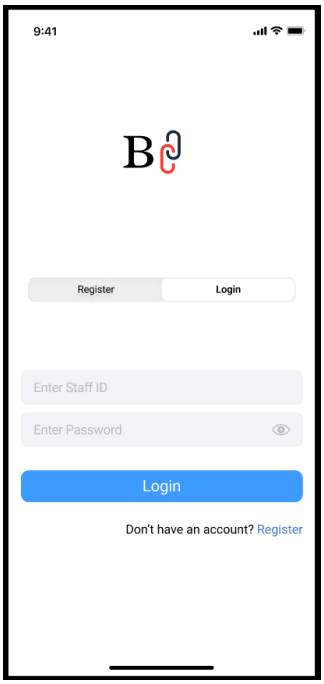


Figure 3.14 mobile application log-in page.

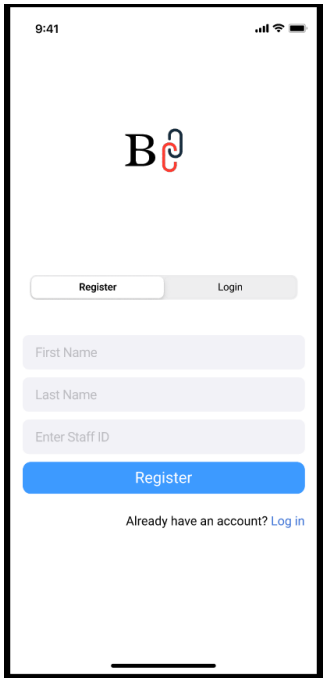


Figure 3.15 mobile application registration page.

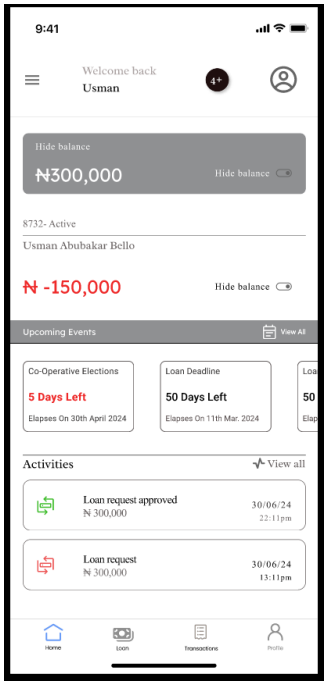


Figure 3.16 mobile application dashboard page.

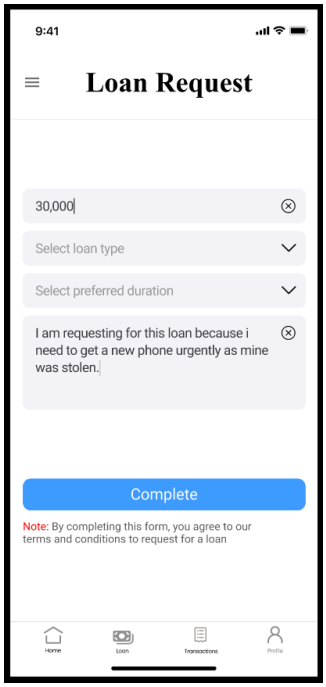


Figure 3.17 mobile application Loan Request page.

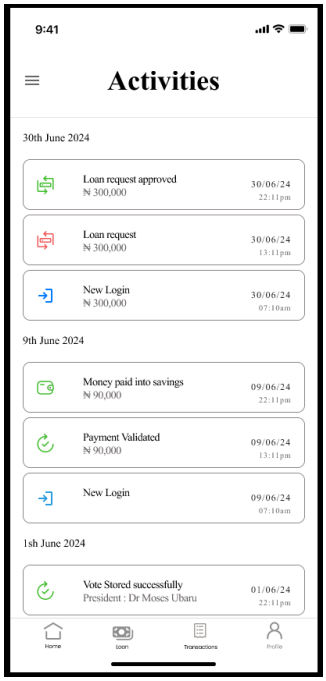


Figure 3.18 mobile application Activities page.

Figure 3.19 mobile application loan Status page.

Chapter 4

Testing and Implementation

4.1 Overview

This chapter covers the development of the system as well as other relevant aspects that have helped with the development like the database, frontend and backend. As well as difficulties faced throughout the development process and how they were solved.

4.2 Main Features

The main features of the system are-

1) **AUTHENTICATION**: the backend server contains logic that allows us to authenticate users on the system. New members may sign up on the Register page that is shown in **figure 4.1** below so that the registration data will be stored in the database.

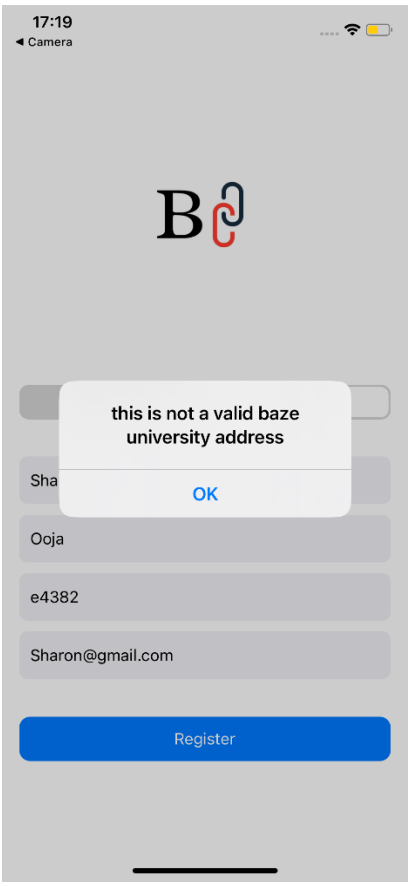


Figure 4.1 invalid registration depicting a user trying to register with an email that isn't provided by Baze university.

Since only the staff of Baze university are allowed to be a part of the cooperative, the registration form will filter out registration requests that don't have the valid format for Baze university's staff ID or requests that without the Baze university email.

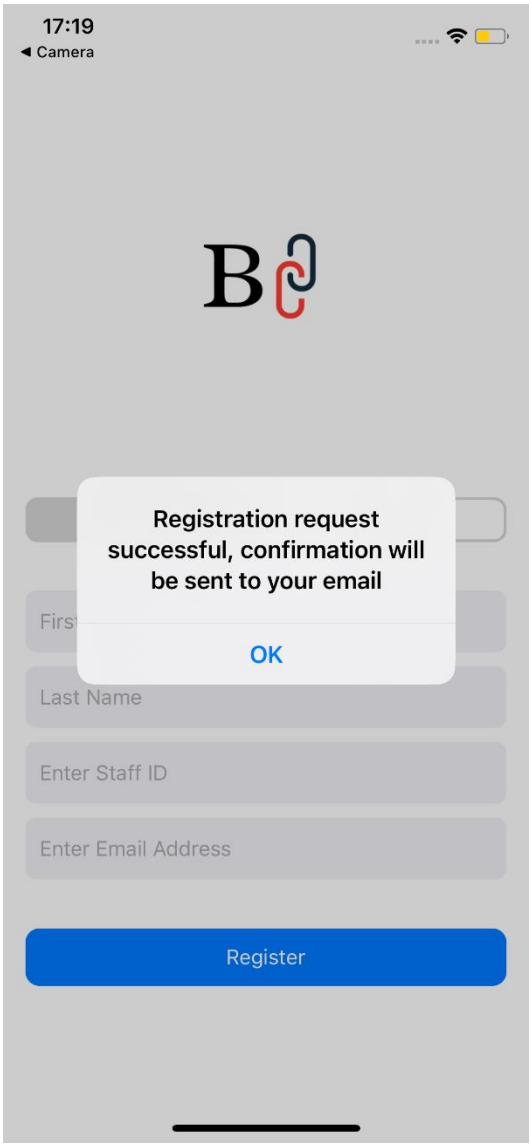


Figure 4.2 showing successful registration

If the provided information has been verified, then it will be stored in the database under the "registrations" collection shown in figure 4.3. with the staff-ID and email address acting as the primary keys

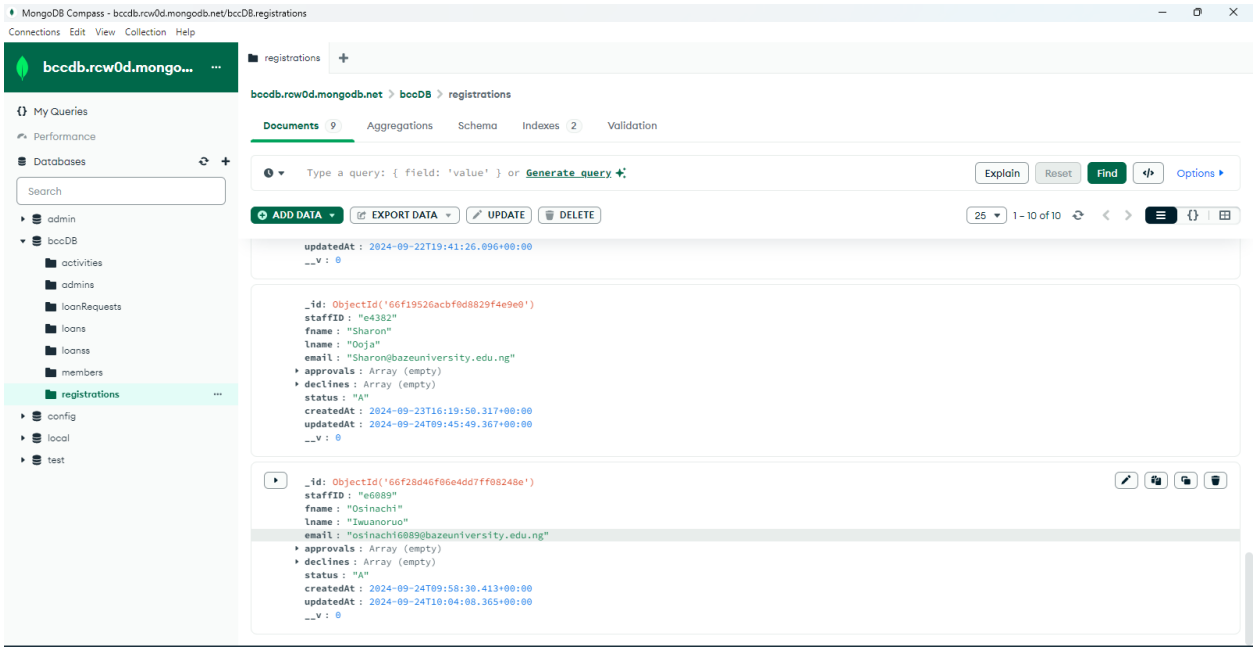


Figure 4.3 Member registrations Collection

The above figure shows how a user's registration is stored in the database to be shown to the executive for the executive's rejection or rejection.

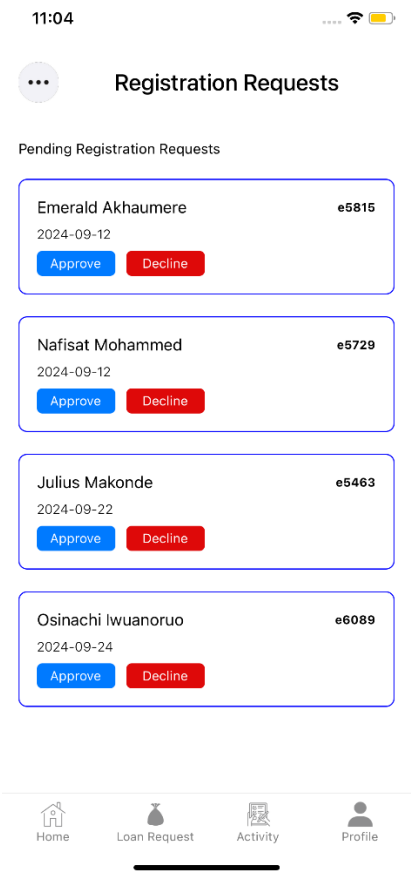


Figure 4.4 registration requests page

Shows the executives view for registration requests, where the last request is the most recent request made.

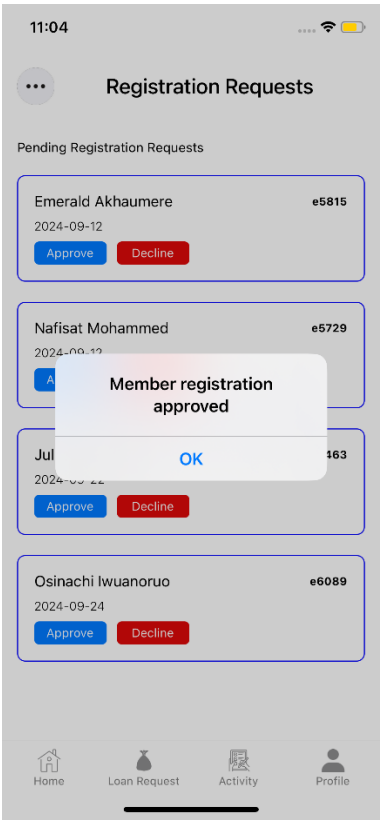


Figure 4.5 registration approved

The figure above shows the confirmation message that the user's registration has been approved, and then an email containing the default password should be sent to the appropriate member. A rejection email will also be sent in the case of an executive rejecting the request.

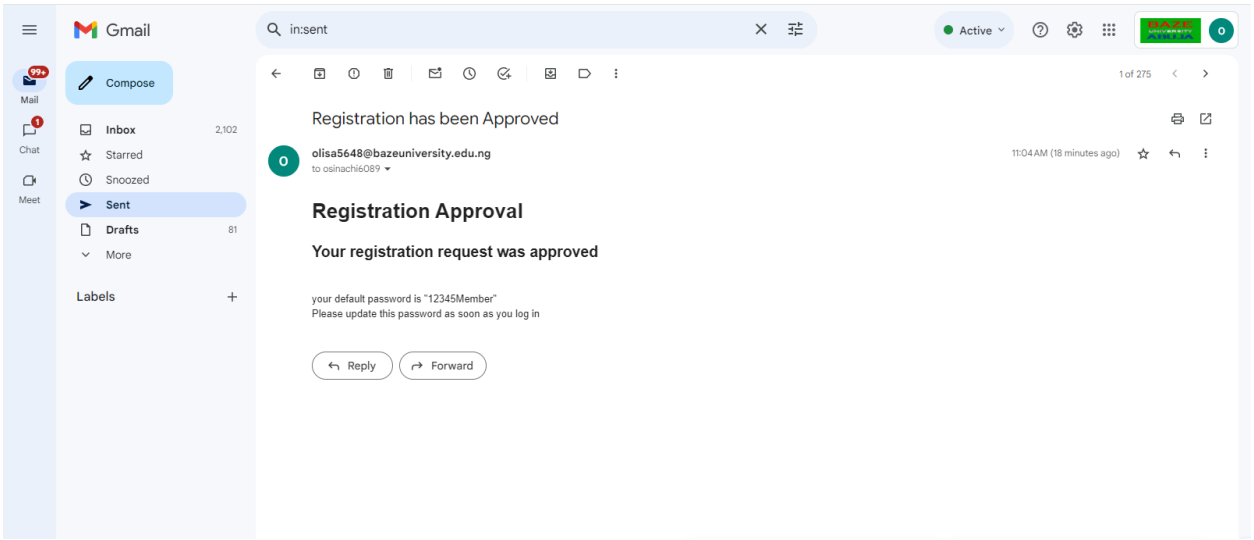


Figure 4.6 Approval sent to member

Figure 4.7 New member login

The recently approved member can now login on the system (**Note**: password does not appear in screenshot because of secure text feature). If the entered details don't match a record in the database, or the entered details don't match a specified pattern, the system will not accept it and show an "invalid details" alert.

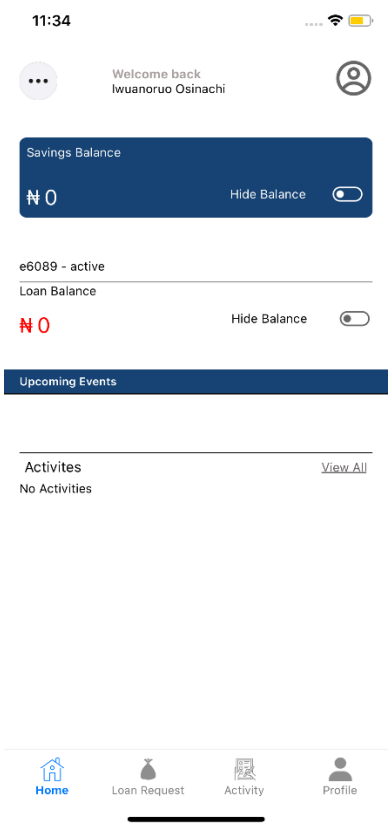


Figure 4.8 new member homepage

Above, we can see the homepage of a member who has recently had their account approved. They cannot perform financial activities until they have a certain amount in their savings.

3) **loan Request:** the new member has no balance until the admin from the web app updates their balance, the figure below depicts the admin updating the new member's balance.

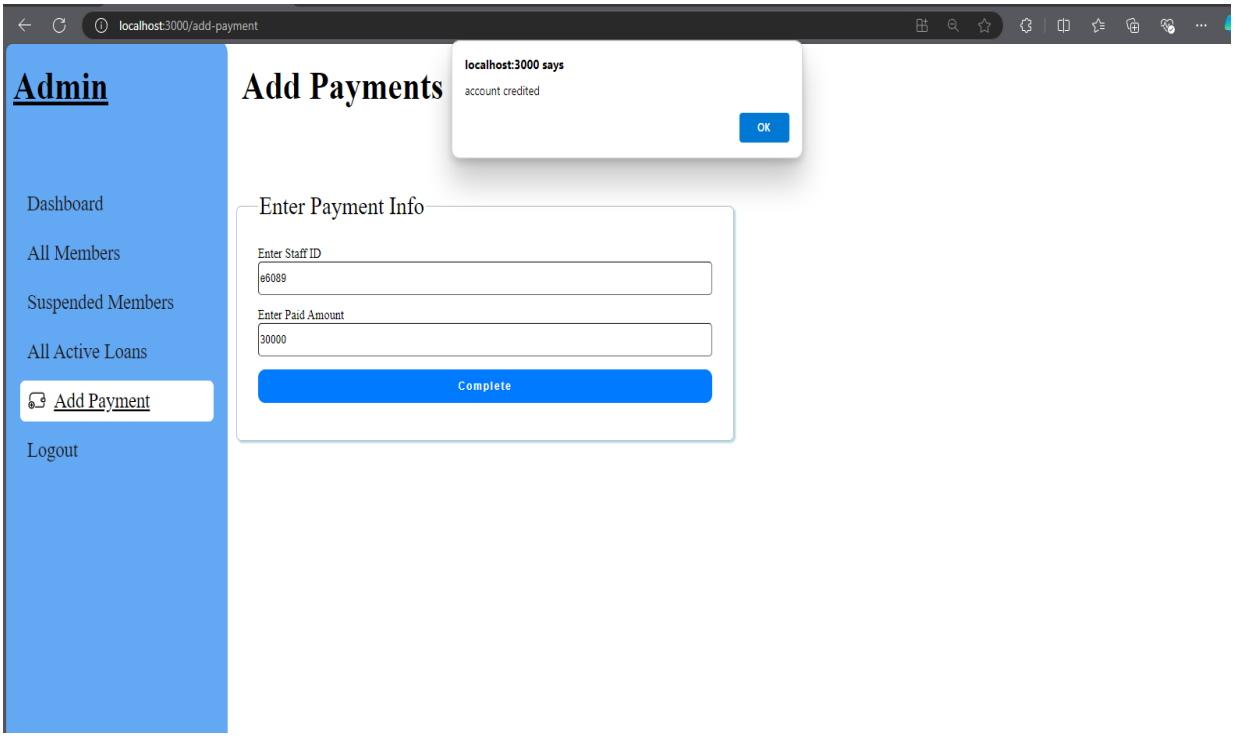


Figure 4.9 admin adding members payment

In the figure above, the admin has successfully added an amount to the members account, the system will check if the current staff has an active loan, if they do then the money is sent towards their loan. Else it will be sent towards their savings. In the database, we should see the member's savings balance updated since they don't have an active loan.

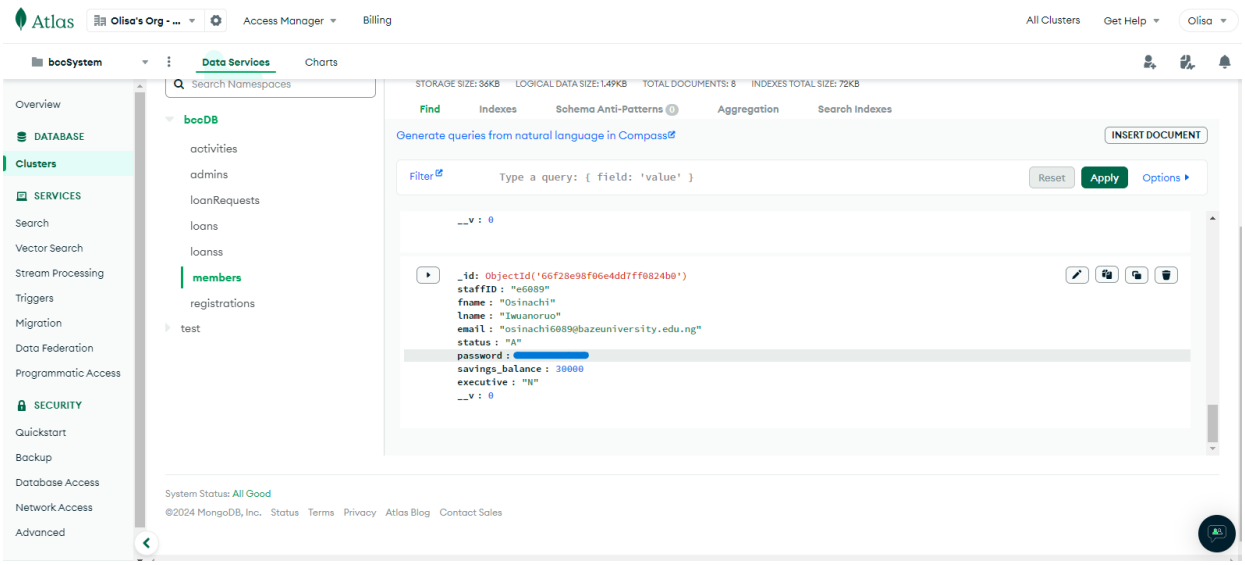


Figure 4.10 The savings balance has been updated

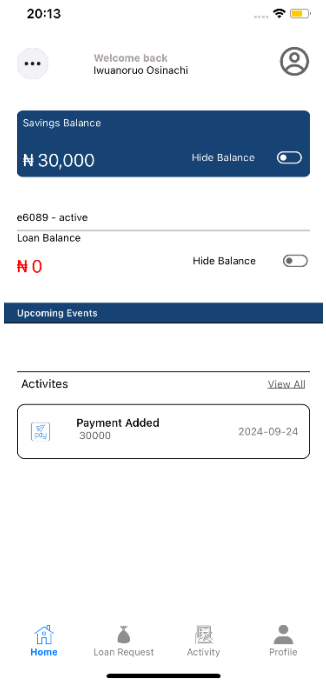


Figure 4.11 Members View

Now the member is eligible to request for a loan, below is the loan request page. It should be noted that the recent payment that was made has been added under the recent activities section and a list of all activities can be found under the “Activities” page.

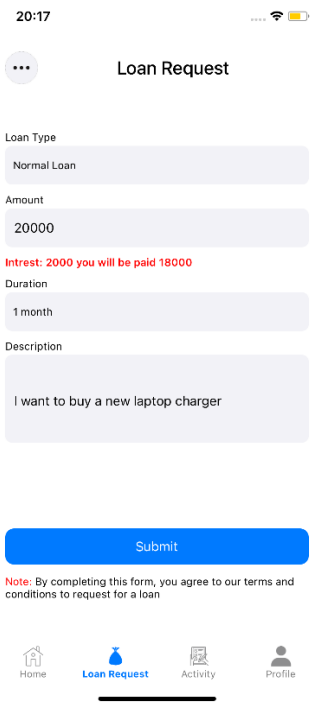


Figure 4.12 Loan Request

For the different kinds of loans, there are different criteria's to be met to be eligible for the loan. The system will consider the members savings balance to know how much they can request for, if they have any pending loans and how much they have paid towards it, if the duration for the selected loan type is within the predefines maximum limit. If these criteria are met, then the request is sent to the executive for approval or rejection.

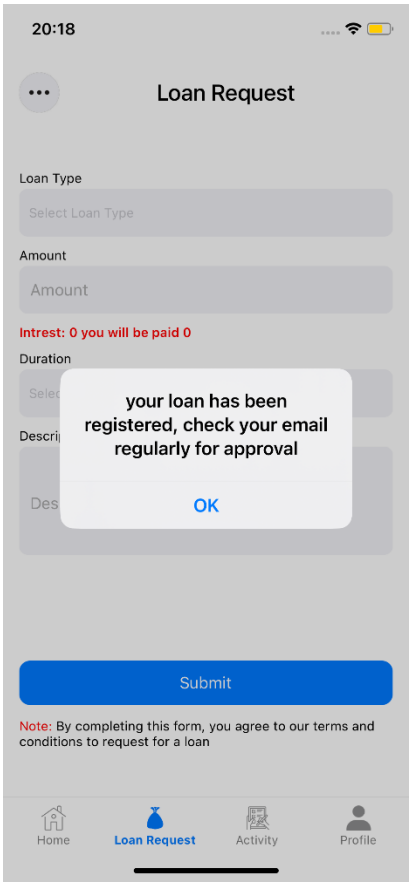
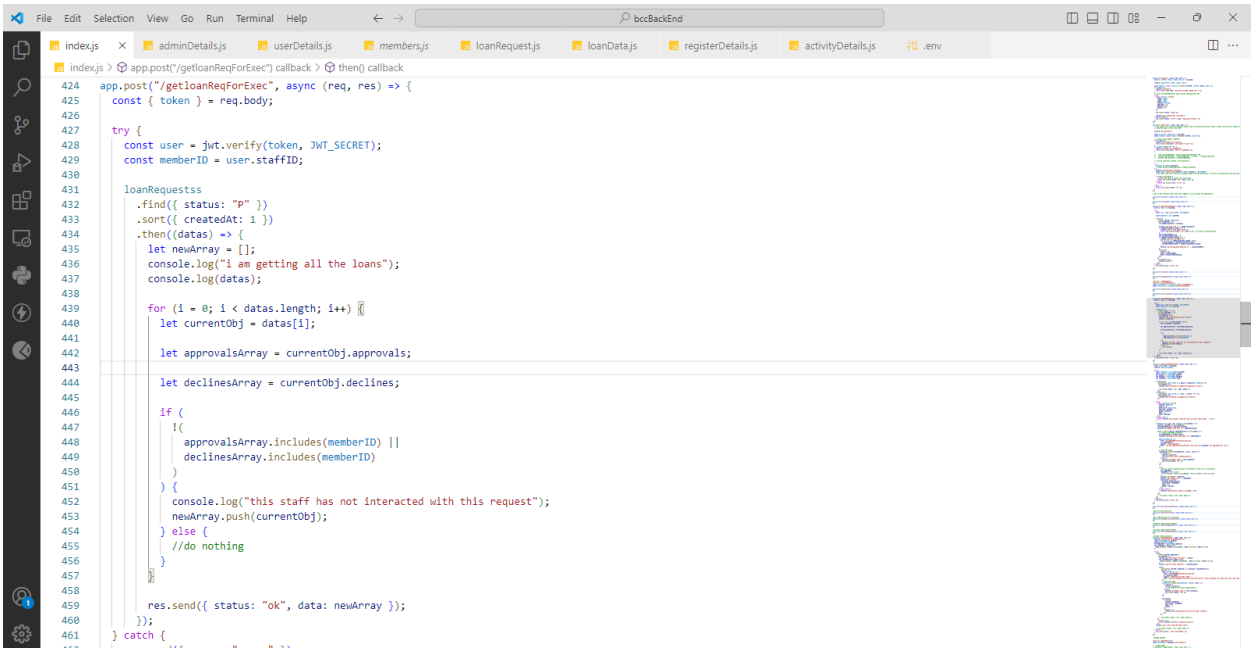


Figure 4.13 loan approval alert



The above figure shows the logic for showing loan requests to the executive. Firstly, all pending loan requests are gotten from the database, then for each record, it is checked if the staff-id of the executive is found under the approvals, or declines array because if it does, then the staff has already either approved or declined the registration request before and should not be shown the request again,

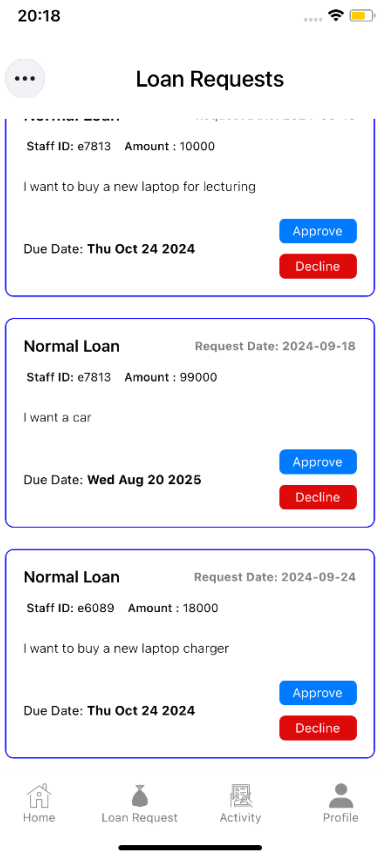


Figure 4.14 Executive view for loan requests

The last loan request on the page is the most recent one and the executive can either decline or approve. If they decline, they are shown a prompt to give a reason, and this reason is sent along with the rejection email to the address of the designated member. If the loan has been approved, then the changes will be reflected on the members dashboard.

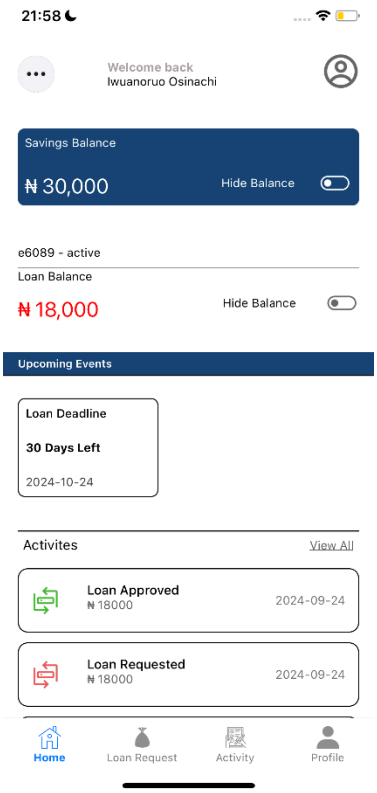


Figure 4.15 Updated dashboard

now that the loan has been approved, we can see the changes with the loan balance being updated, the loan deadline appearing under the events section and the activity appearing under the activities section. More information on the loan will be found in the loan status page.

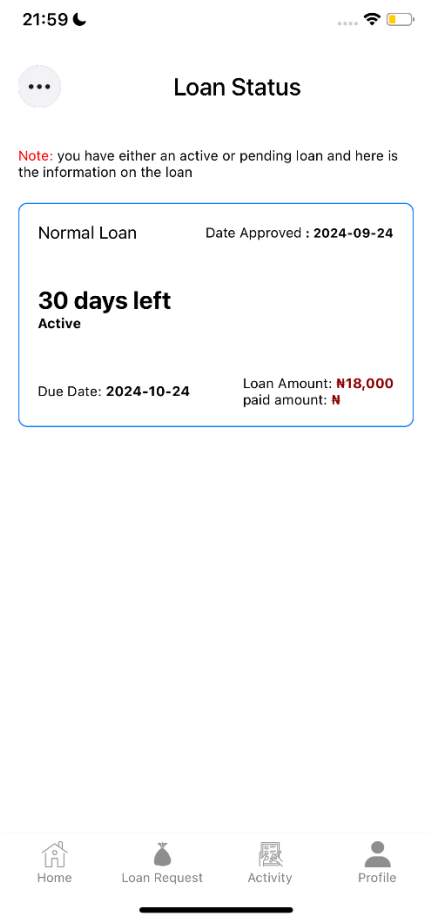


Figure 4.16 Loan Status page

In the figure above, for the active loan, we can see the loan type, how many days till the deadline, the approved date and the actual due date, the loan amount and how much has been paid towards the loan.

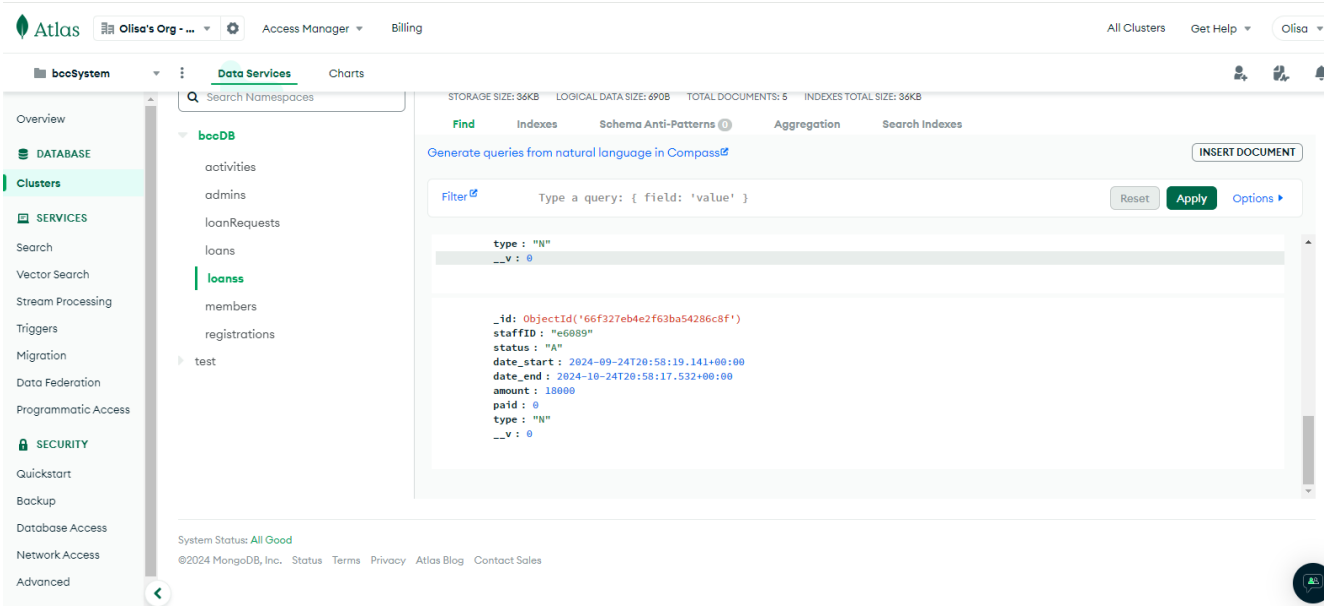


Figure 4.17 loan stored in database

The figure 4.17 above shows how data on active loans are represented in the database.

Admin Web App

Some features of the Administrators app have been outlined below-

1) LOGIN/ LOGOUT: the admin must provide details that match a record in the database.

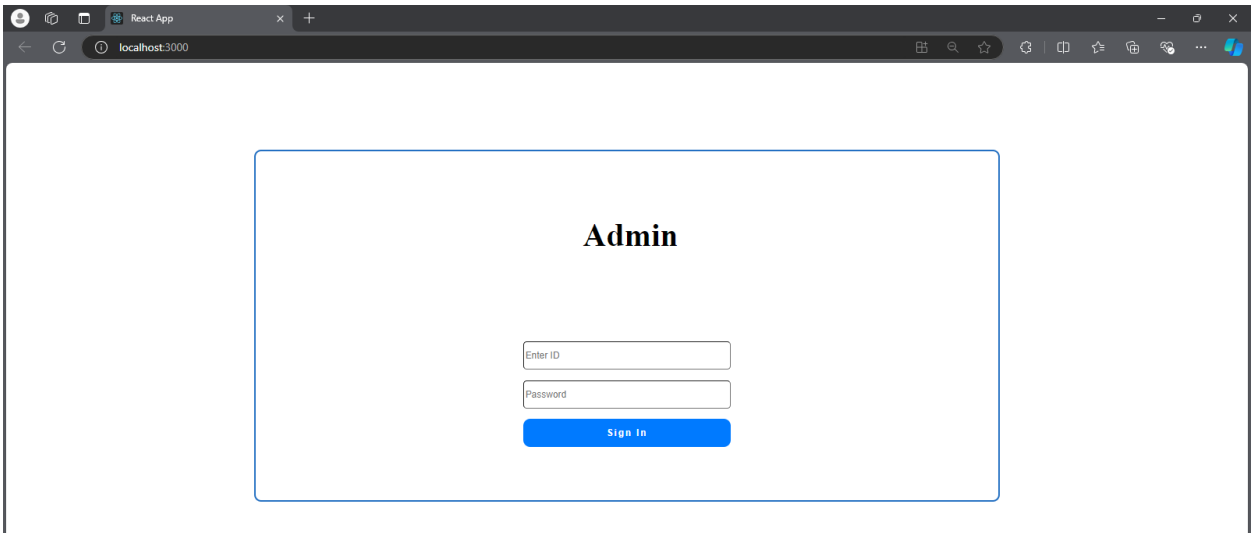


Figure 4.18 Admin login page

2) DASHBOARD: the admin sees how many total members there are, how many are suspended, and the total active loan amounts

3) SEE ALL MEMBERS: on the active members page, the admin finds all active members and may quickly suspend a previously active member. The member details provided are the name, staff-id, status, email. When the admin clicks the suspend button, they get a confirm alert to ensure that the button wasn't clicked accidentally, if confirmed, a suspension email is sent to the member and their activity status is updated.

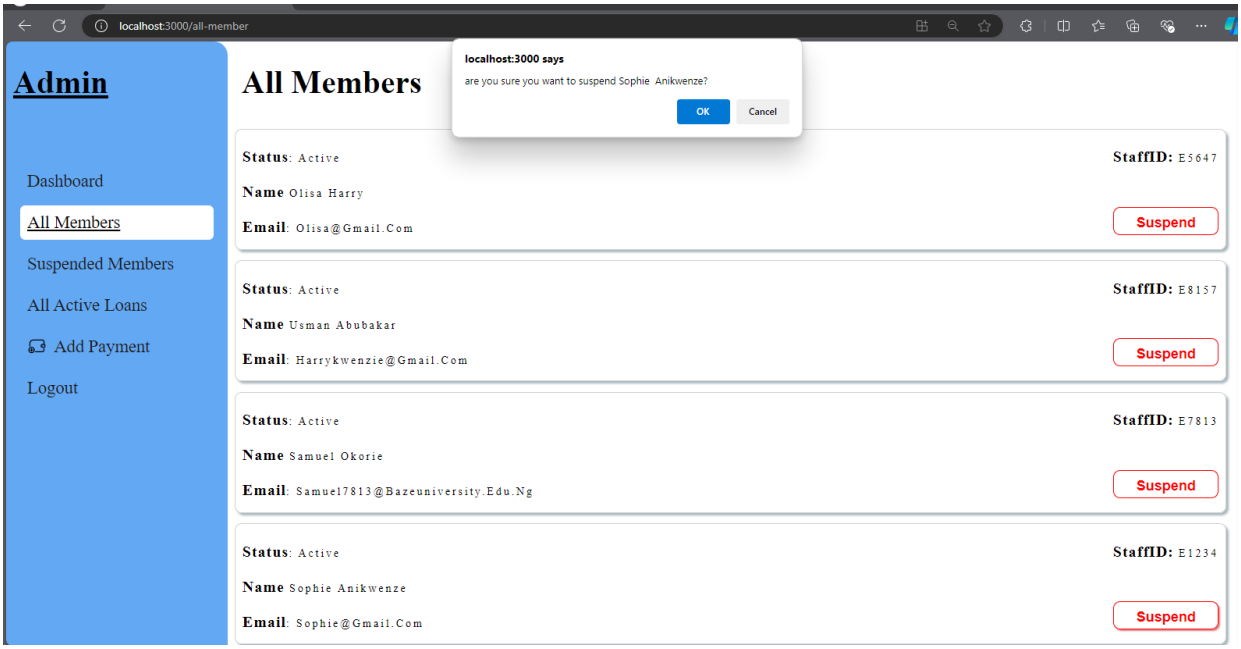


Figure 4.19 All member's page

4) ACTIVATE A MEMBER: the admin can also activate a suspended members account from the suspended members' page.

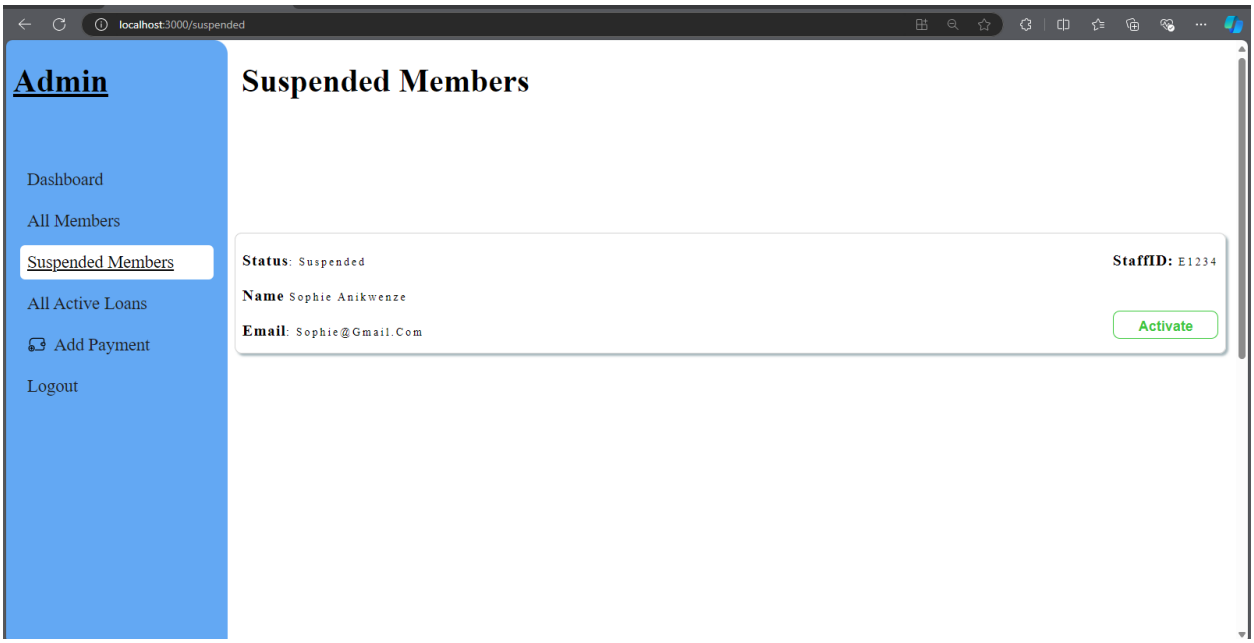


Figure 4.20 All suspended page

This page displays all the suspended members of the co-operative, clicking the activate button will show a confirm alert to reduce human error. The member details provided are the name, staff-id, status, email.

5) VIEW ACTIVE LOANS: all active loans may be viewed by the admin, and they can see who has the active loan, how much has been allocated and how much has been paid back.

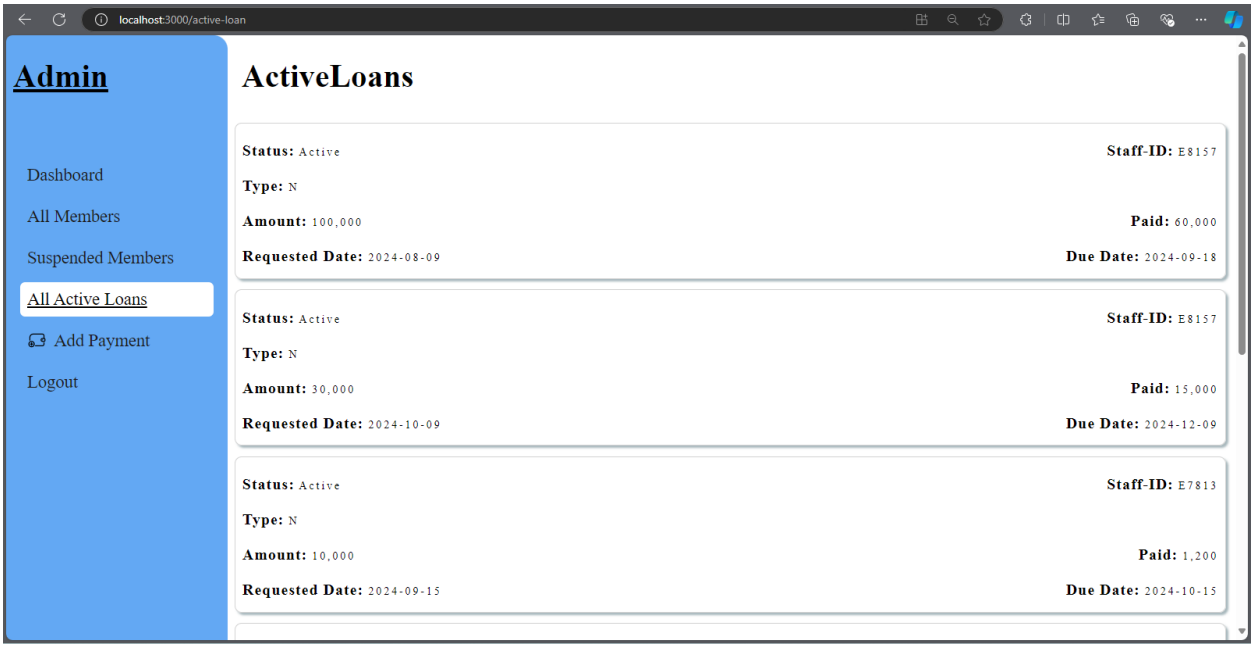


Figure 4.21 All Loans page

6) <u>UPDATE PAYMENT:</u> once the admin has confirmed a member's payment, they can update their account's balance on the "Add Payments" page. They don't need to bother about if it's towards the loan or savings. The system will check If there is an active loan for the member and if there is, then the amount will go towards it, else it will go towards their savings.

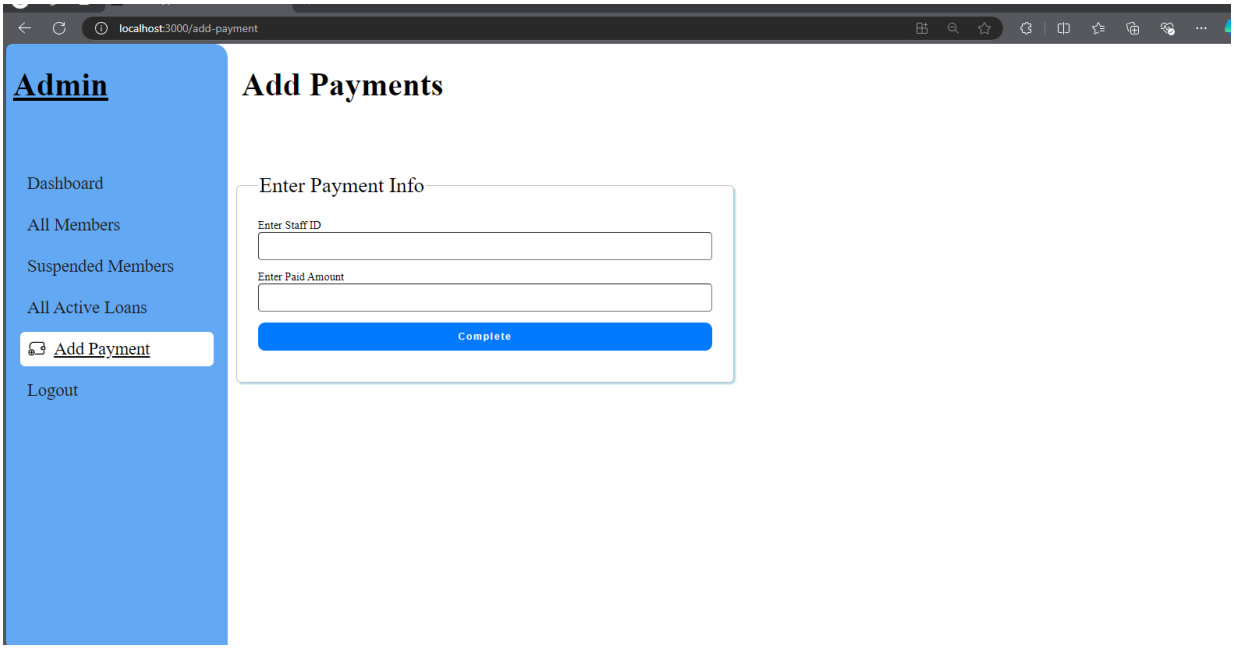


Figure 4.22 Add Payments page

4.3 Implementation Problems

Throughout the course of the systems development, there were a few hitches in the development that were solved with proper problem-solving skills, some have been outlined below-

1. **Learning a new language-** learning new technologies like React Native, Nodejs, MongoDB was quite challenging as the timeframe to learn all of them and implement the knowledge was quite small, but I was able to properly implement these different technologies to develop a set of apps that communicate with each other by God's Grace.
2. **React Native version-** the Latest React native version changes the file organization, and it was difficult to find documents and tutorials that were up to date.
3. **Relatively slow data retrieval-** as there are no servers in the country, the time taken to retrieve data from the cloud was with noticeable lag, but I was able to connect to a server within closer range.
4. Since I did not have access to the institutions database and neither am I a staff member, I had to build my own mock database on the information I was able to gather and perform some reasonable and logical assumptions.
5. There have not been a great number of active applications for co-operative societies to review.

4.4 Overcoming Implementation Problems

the problems mentioned above were overcome by performing extensive research on the related areas.

1. Watching several video tutorials and reading official documentations offered on the technologies made development easier.
2. Making sense of the new file structure made building the application speedy.

3. Interviews with members of the co-operative society helped me to understand their workings and clear some assumptions.

4.5 Testing

To ensure that the system worked properly and met the required specifications, the systems were tested from the point of view of the end users to depict how the systems perform in a real-world use case, to show if the applications can perform proper error handling, or if they are even usable at all. Testing methods such as unit testing, were used to analyze the systems from the frontend to the backend.

4.5.1 Test Plans (for Unit, Integration and System Testing)

Table 4.1 Test Summary for user registration

Test Case	User Registration
Related requirements	R-5
Prerequisites	User should be able to access the main login/register page
Test procedures	Click on the “Register” toggle button Enter valid signup details Click on “Register” button
Test Data	Users Valid details
Expected Results	Registration sent to executive
Actual Result	Registration information was sent to executives view
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024

Test environment	Hp laptop

Table 4.2 user login testing

Test Case	User login
Related requirements	R-4
Prerequisites	User should be able to access the main login/register page, member account must be active
Test procedures	Click on the “login” toggle button Enter valid staff-id and password Click on “login” button
Test Data	Valid staff id and password
Expected Results	Member should be taken to the home page where they can see their loan and savings balance
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024
Test environment	Hp laptop

Table 4.3 user loan request

Test Case	User loan request
Related requirements	R-3
Prerequisites	Member is logged in Member has paid over half of their current loans

	Member has an active account
Test procedures	Navigate to loan requests page Select loan type Enter loan amount and select duration Optionally enter reason for loan request
Test Data	Amount must be a number, duration must correlate with loan type,
Expected Results	Loan request should be stored in database and shown to the executive
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024
Test environment	Hp laptop

Table 4.4 Executive loan approval/rejection

Test Case	Executive loan approval/rejection
Related requirements	R-101
Prerequisites	Member is logged in Member is an executive
Test procedures	Navigate to loan requests page on executive drop-down menu Click either the approve or decline button Optionally give a reason for rejection if the decline button is clicked
Test Data	Loan request and staff id

Expected Results	Loan request should either be approved or declined and a confirmation or rejection email should be sent to the members email
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024
Test environment	Hp laptop

Table 4.5 Executive registration approval/rejection

Test Case	Executive registration approval/rejection
Related requirements	R-102
Prerequisites	Member is logged in Member is an executive
Test procedures	Navigate to registration requests page on executive drop-down menu Click either the approve or decline button
Test Data	Registration request, clicked button and staff id
Expected Results	Registration request should either be approved or declined and a confirmation or rejection email should be sent to the members email
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024

Test environment	Hp laptop

Table 4.6 user see loan status

Test Case	Member views loan status
Related requirements	R-2
Prerequisites	Member is logged in Member has an active loan
Test procedures	Navigate to loan status page on drop-down menu View all active loans
Test Data	Staff ID
Expected Results	Any active loans for the member should be shown on this page
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	23rd August 2024
Test environment	Hp laptop

Table 4.7 Admin suspends or activates a member account

Test Case	Admin activated/suspends user account
Related requirements	R-202
Prerequisites	Admin is logged in
Test procedures	Navigate to either all Members page or suspended members page Select the activate/suspend button

	Select okay from the confirmation alert
Test Data	Staff id and selected button
Expected Results	The members account should either be suspended or activated and an appropriate email should be sent to the email of the member
Status	Pass
Remarks	None
Created by	Anikwenze Olisa Harry
Date of creation	23rd August 2024
Executed by	Anikwenze olisa Harry
Date of execution	26th August 2024
Test environment	Hp laptop

4.6 Use Guide

Step1: Internet connectivity

Make sure your device has a stable internet connection.

Step2: Registration or Login

1. New members should be full in the registration form with valid registration details and check their emails regularly for confirmation with password or rejection emails.
2. Returning users should fill in the login form with valid login details to gain access to the system.
3. The admin should login on the web app with their valid login information.

Step3: Homepage/Dashboard

For all members (Executive Inclusive)

1. Click the three dots at the top left corner to get to mortgage, see your loan status or log out.
2. Click the hide icon to hide either loan or savings balance.
3. Click on the activities link to see all activities.

For Executives only (Regular members exclusive)

1. Click the three dots at the top left corner to get to see all loan requests and registration requests.

Step4: Loan request page

For members with at least 5000 in savings

1. Fill the loan request form with valid details
2. Check interest before filling out the form
3. Click the "submit" button to send form to executive

Step:5 All activities

1. Click "Activities" from the navigation bar to see all activities performed on the account
2. Scroll to see older activities sorted by date

Step6: Profile page

1. See your account related information
2. Click "Change password" to change your password

Step7: Loan status page

See all active loans sorted in order of the lowest days remaining

FOR EXECUTIVE ONLY

Step9: Loan requests page

1. Click approve to approve a member's loan request
2. Click decline to reject a member's loan request and optionally give a reason

Step10: registration requests page

1. Click approve to approve a member's registration request.
2. Click decline to reject a member's registration request.

FOR ADMIN ONLY

Step11: ADD payment

1. Select add payment to add member payment
2. Enter valid staff-ID and amount then click update

Step12: suspend member

1. Select all members' link
2. Click the suspend button and select "ok" from the prompt to suspend member's account

Step13: activate member

1. Select suspend members' link
2. Click the activate button and select "ok" from the prompt to activate member's account

4.7 Summary

The testing of a system is very important to the final version that will be rolled out to the public. Various kinds and stages of testing must be carried out to find out if the system is usable or meets the needs of the end-user. Proper testing was carried out on this system and any errors found were rectified accordingly. The next chapter covers an assessment of the project, the summary and any recommendations for future iterations.

CHAPTER 5

DISCUSSION, CONCLUSION, AND RECOMMENDATIONS

5.1 Overview

The content of this chapter evaluates the project, including the constraints faced during development and any features that will be incorporated to enhance its functionality in the future.

5.2 Objective Assessment

Most of the objectives set for the system were completed like allowing members of the co-operative to request for loans on a digital platform, keep track of their loan and savings balance, see all their activities, approve and decline requests quickly allowing members to use their free time more productively. As advancements in technology continue, the system will also be updated to not lack.

5.3 Limitations and Challenges

Developing the systems proved to be a greatly demanding task to accomplish and I was able to acquire necessary skills to see it through. Some challenges have been outlined below:

1. **Time management**: managing my time between demanding schoolwork, house chores and developing both a mobile and web application required I fine tune my time management skills to deliver a properly working app.
2. **New Technologies:** getting to understand these new technologies was not easy as I was not familiar with them before.

For members to access the applications' features, they must meet the following requirements,

1. A member should have internet connection to login or register.
2. A member should have internet connection to request for a loan, see all activities, change their password.

3. An executive must have internet connection to see loan/ registration requests and to approve or decline them.
4. An admin must have internet connection to suspend or to activate a members account.
5. An admin must have internet connection to update a member's balance.

5.4 Future Enhancements

With the given period for the project to be completed, some features were not incorporated, these features include:

1. **Voting system**: the members of the co-operative can vote for members during an election.
2. **Upload payments**: members can upload payment evidence for the executives to validate directly.
3. **Link system to school's bank records**

5.5 Recommendations

Some measures that can be followed to enhance the usability of the system include:

1. Set a reminder for payment of loans
2. Use an anti-virus on mobile phone or any device

5.6 Summary

This chapter concludes the project documentation. This document contains all phases of the software development process including the aims and objectives, the challenges and limitations faced and overcome, assessment of possible risks and how they may be mitigates, requirement analysis, project design and implementation and the methodology that guided the development process. Adequate testing was carried out to ensure that the systems were reliable and efficient. Finally, recommendations and suggestions for future improvements were analyzed.

REFERENCES

Andreansyah, A., & Rizkiana, R. (2020). “Implementation of Incremental Models on Development of Web Based Loan Cooperative Applications”. International Journal of Education, Science, Technology and Engineering, 3(1),26-34, https://doi.org/10.36079/lamintang.ijeste-0301.105

Atanasova, A., Eckhardt, G., & Laamanen, M. (2024). Platform cooperatives in the sharing economy: How market challengers bring change from the margins.

Budzinski, M. (2024). What Is React Native? Complex Guide for 2024. NetGuru. https://www.netguru.com/glossary/react-native

Carrell, S. (2007). Strike Rochdale from the record books. The Co-op began in Scotland. The Guardian. https://www.theguardian.com/business/2007/aug/07/retail.uknews

Deshpande, C. (2024). The Best Guide to Know What Is React. Simplilearn. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs#:~:text=Easy%20creation%20of%20dynamic%20applications,thereby%20creating%20web%20applications%20faster.

Domantas, G. (2023). What Is CSS and How Does It Work? Hostinger. https://www.hostinger.com/tutorials/what-is-css

Gillis, A., & Botelho, B. (2023). What is MongoDB. TechTarget. https://www.techtarget.com/searchdatamanagement/definition/MongoDB

Goray, S. (2024). History of Mobile Apps - The Past, Present and Future. Webandcrafts. The History of Mobile Apps and Evolution of Mobile Platforms (webandcrafts.com)

Gillis, S. (2023). What is MongoDB? TechTarget. https://www.techtarget.com/searchdatamanagement/definition/MongoDB [Accessed on 29/05/2024]

Howart, J. (2024). How Many People Own Smartphones? Exploding Topics. https://explodingtopics.com/blog/smartphone-stats

Mbam, E., & Igboji, O (2013). Enhancing Cooperative Loan Scheme through Automated Loan Management System, West African Journal of Industrial & Academic Research,6(1), 144-152

Moechammad, S., Devi-Khanthi, B., Putri-Elfa, M., Yunia, M., Nailul, Muna5., & Ekananda, S (2021). “Bulletin of Electrical Engineering and Informatics”:The development and implementation of an android-based saving and loan cooperative application. 10(6), 3482-3488

Muller, S., & Tworek, H. (2015). ‘The telegraph and the bank’: on the interdependence of global communications and capitalism. Cambridge University Press. https://www.cambridge.org/core/journals/journal-of-global-history/article/abs/telegraph-and-the-bank-on-the-interdependence-of-global-communications-and-capitalism-18661914/09E91AB90AE1274717589316F28B6E73

National Library of Scotland. (2023). The history of working people in Scotland. https://www.nls.uk/learning-zone/politics-and-society/labour-history/

Olorunlomerue, A., Ekuewa, B., Oyetunji, O., & Ramoni, A. (2017). Web Based Centralized Cooperative Information Management System. International Journal of Computer Trends and Technology (IJCTT). 54(2), 126-129.

Oluyombo, O (2013). Impact of Cooperative Societies Savings Scheme in Rural Finance: Some Evidence from Nigeria. Economic Review -Journal of Economics and Business,11(1):77-88

Onyeama, C., Ekeh, E., Onyike, G., & Onwubuariri, I (2017). Design and Implementation of a Web Based Saving Scheme Management System, IRJN Journal of Management, 2(1): 1-21

Saini, A. (2024). MongoDB – Working and Features. GeeksforGeeks. https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/

Seffusatti, N. (2024). “No Man Is an Island”: The Power of Community. GUIDE COLLECTIVE. https://www.guide-collective.com/gc-magazine/no-man-is-an-island-the-power-of-community . [Accessed on 30/06/2024]

Semah, B. (2022). What Exactly is Node.js? Explained for Beginners. freeCodeCamp. https://www.freecodecamp.org/news/what-is-node-js/

Sobolev, A., Bednyagin, S., & Yurmanova, H. (2023). Cooperative Platforms: The Practice of Digital Participation. Springer Link. Cooperative Platforms: The Practice of Digital Participation | SpringerLink

Sumana, G. (2023). 7 Leading Software Development Methodologies. DESIGNRUSH. https://www.designrush.com/agency/software-development/trends/software-development-methodologies .

Wegner, D., Borba, A., & Mitrega, M. (2024). A systematic review of collaborative digital platforms: structuring the domain and research agenda. 4(18), 2663-2695.

Wright, G., & Ferguson, K. (2024). Application architecture. Techtarget. https://www.techtarget.com/searchapparchitecture/definition/application-architecture [Accessed on 29/05/2024]

Yusuff, A., Folajin. O., & Oriowo. O (2020). Appraisal of Information and Communication Technology in Cooperative Societies Management: A Case Study of Cooperative Societies in Osun State Tertiary Institution of Learning, Nigeria. The International Journal of Business & Management,3(4):438

Appendix A - Project Document

IN-DEPTH PROJECT DOCUMENTATION

Full Candidate Name: Anikwenze Olisa Harry

Student ID: BU/21C/IT/5648

Title Design and Implementation of a Co-operative System for Baze University Staff.

Course of Study: B.Sc. Computer Science [Software Engineering].

Background and Motivation

The current state of the Nigerian economy has become almost unbearable for Nigerians with the price of food, fuel amongst other necessities skyrocketing beyond all reasonable levels. The need for properly functioning co-operative societies has never been so dire in the country.

With the existing cooperative society in Baze University, the staff already carry out these executives, but more manual based which takes significant time from their daily lives that could be spent towards more productive tasks. The development of a digital platform will greatly impact the society and increase their efficiency and accuracy while saving them time.

This system will greatly lessen the burden carried by the society's executive team in the case of validating registration or even approving loans; and even those levied upon its members like in the case of requesting for loans.

Statement of the Problem

The activities carried out by the members of Baze University is manual based and must be digitalized to match current trends and increase efficiency.

There the issue of some requests made by members not being attended to, so the system follows a FCFS approach to displaying requests on the system.

Appendix B – Interview

There was multiple interview sessions conducted with a member of the Baze Co-operative. Some questions asked were-

Me: what Exactly does Baze cooperative Need?
Answer: the members must always know their balances.

Me: what balance?
Answer: their loan balance, savings balance, how much they have left to pay, and Its due date.

Me: how does the current system work?
Answer: the current system follows works on an FCFS (first come first Serve) basis, because members have the issue of not being attended to, So the proposed system should follow that principle.

Me: what are some of the problems faced by the co-operative members?
Answer: → Newcomers may be attended to first before others
→ there is no way for the members to know their balance

Me: what are some of the problems faced by the co-operative executive?
Answer:
→ the debtors find it hard to pay back, making the creditors struggle to gather these funds.
→ keep track of all members, (current members, Ex-members, Suspended members)
→ the executive panel should be given real time notifications about activities on the app that is the activities that are necessary for them to know of.

Me: Is it open to only Academic Staff?
Answer: Not just academic staff, but non-teaching staff only at Baze university also.

Me: Will the System allow members to make in App payments?
Answer: while not a core function, it will be a fine feature

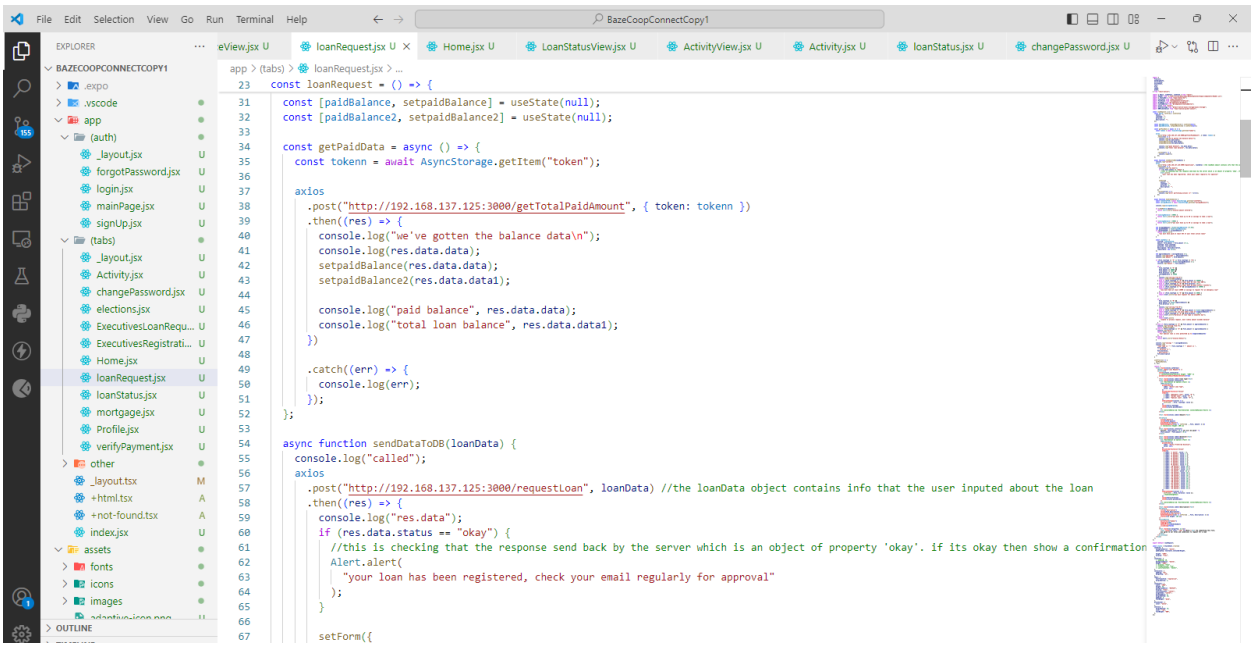
Me: is registration done in person?
Answer → that is the set up in place, where you go to someone maybe in an office, then upon Some sort of review they are added to the cooperative.

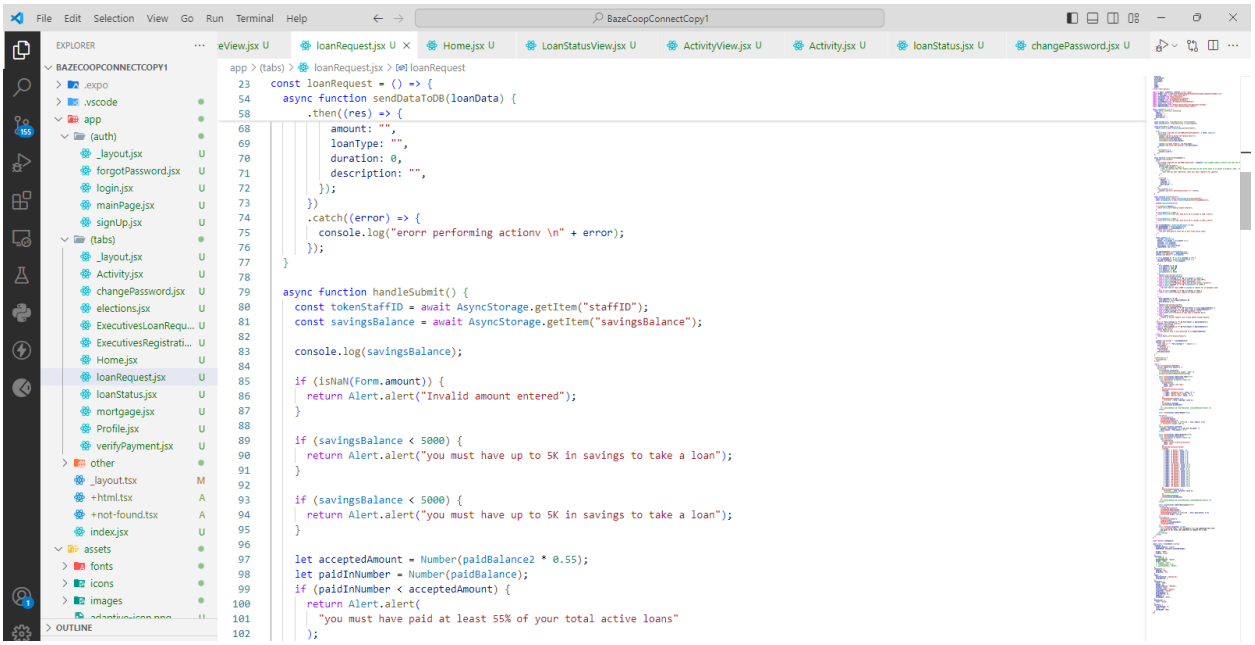
Me: Can a member request for multiple loans at once?
Answer: No, but in a special Condition, yes. If for a member's current active loans, they have paid at least 55% of their balance, then they Can request for a new loan.

Me: will a mobile application or web application be better for the society?
Answer: a mobile application can be developed for the members of the co-operative to perform their activities while a web-based application can be used by the admin to oversee activities of the co-operative.

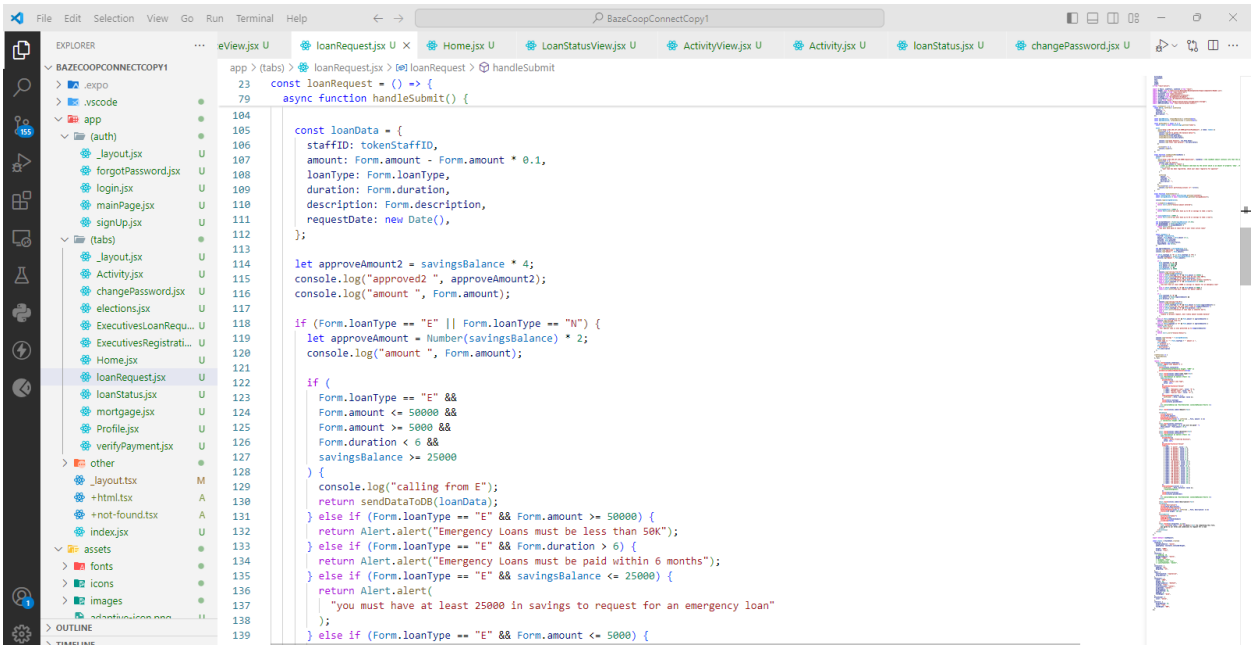
Appendix C- Source Codes

Loan request page

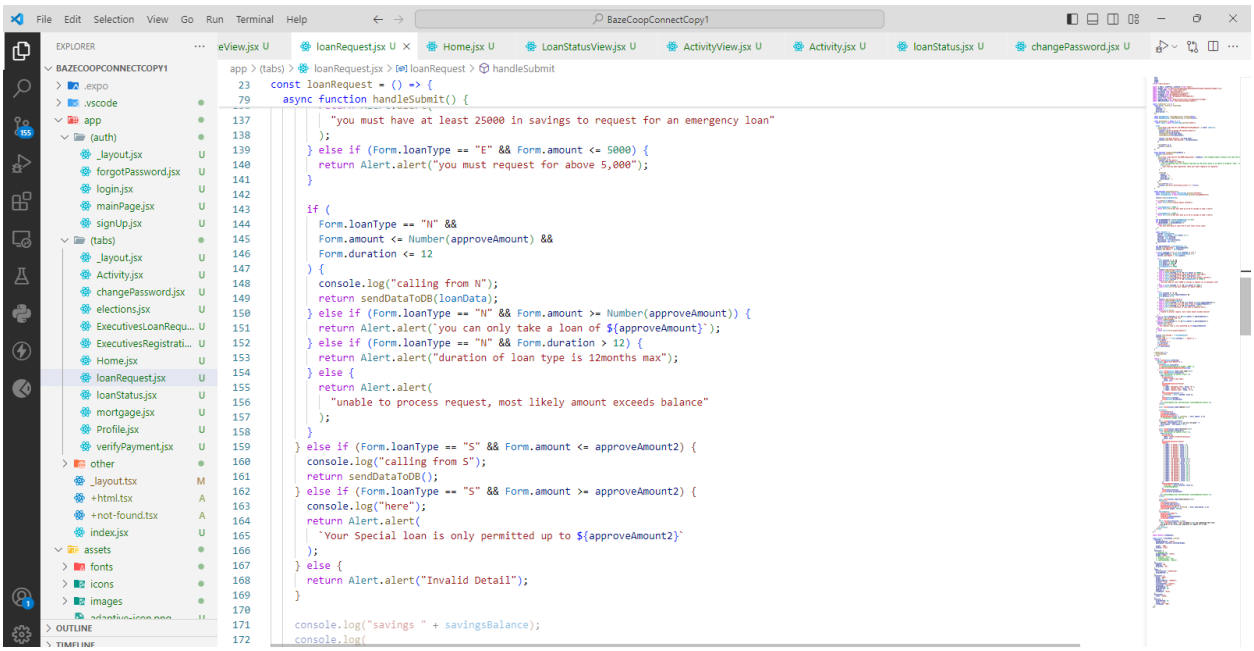




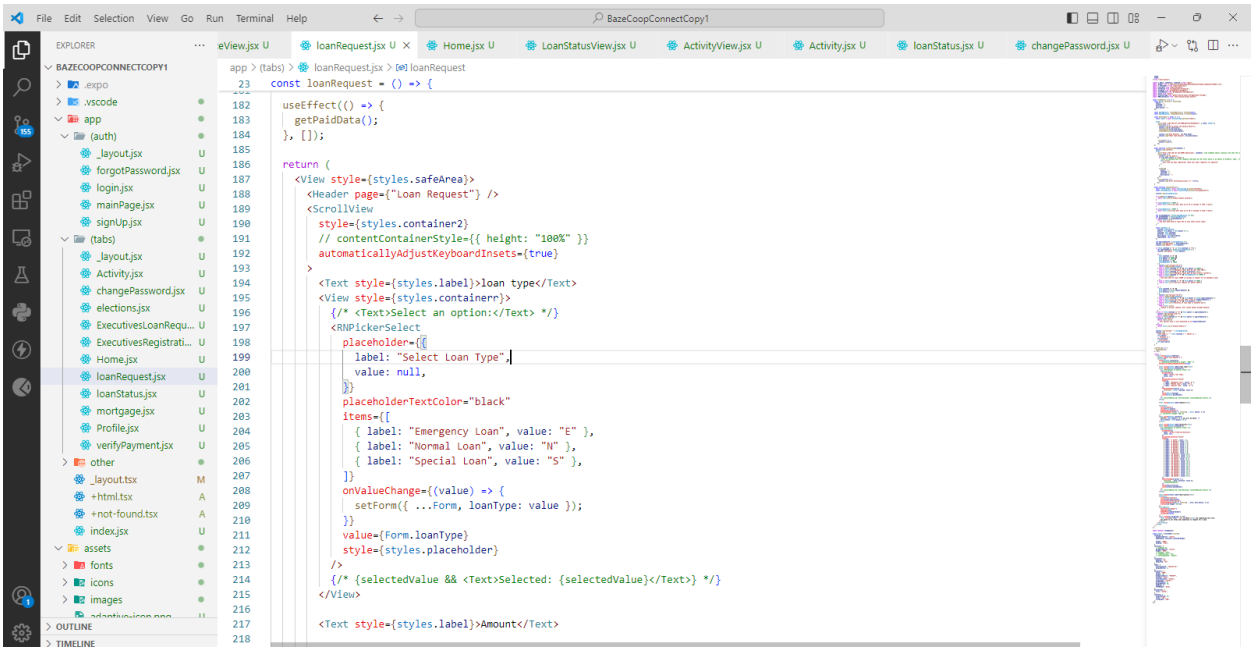
```
23  const loanRequest = () => {
54    async function sendDataToDB(loanData) {
58      .then((res) => {
68          amount: "",
69          loanType: "",
70          duration: 0,
71          description: "",
72        });
73      })
74      .catch((error) => {
75        console.log("erorr performing actionv \n" + error);
76      });
77  }
78
79  async function handleSubmit() {
80    const tokenStaffID = await AsyncStorage.getItem("staffID");
81    const savingsBalance = await AsyncStorage.getItem("savingsBalance");
82
83    console.log(savingsBalance);
84
85    if (isNaN(Form.amount)) {
86      return Alert.alert("Invalid amount entered");
87    }
88
89    if (savingsBalance < 5000) {
90      return Alert.alert("you must have up to 5K in savings to take a loan");
91    }
92
93    if (savingsBalance < 5000) {
94      return Alert.alert("you must have up to 5K in savings to take a loan");
95    }
96
97    let acceptedAmount = Number(paidBalance2 * 0.55);
98    let paidInNumber = Number(paidBalance);
99    if (paidInNumber < acceptedAmount) {
100     return Alert.alert(
101       "you must have paid at least 55% of your total active loans"
102       );
```



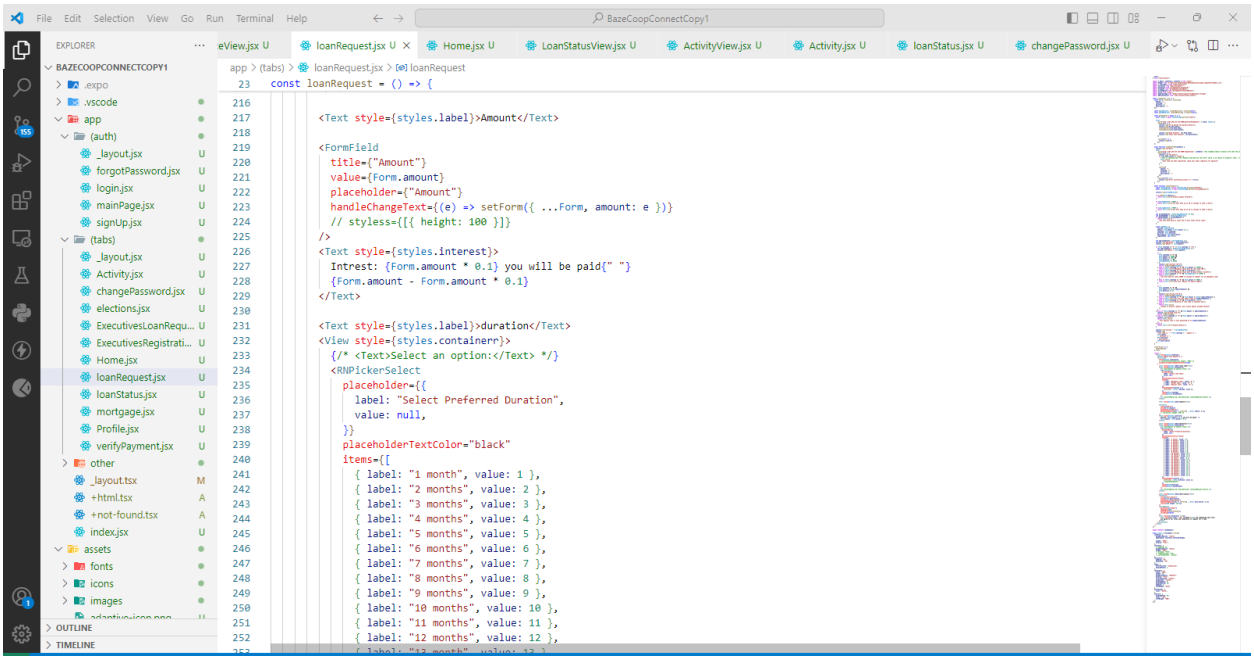
```
23  const loanRequest = () => {
79    async function handleSubmit() {
104
105     const loanData = {
106       staffID: tokenStaffID,
107       amount: Form.amount - Form.amount * 0.1,
108       loanType: Form.loanType,
109       duration: Form.duration,
110       description: Form.description,
111       requestDate: new Date(),
112     };
113
114     let approveAmount2 = savingsBalance * 4;
115     console.log("approved2 ", approveAmount2);
116     console.log("amount ", Form.amount);
117
118     if (Form.loanType == "E" || Form.loanType == "N") {
119       let approveAmount = Number(savingsBalance) * 2;
120       console.log("amount ", Form.amount);
121
122       if (
123         Form.loanType == "E" &&
124         Form.amount <= 50000 &&
125         Form.amount >= 5000 &&
126         Form.duration < 6 &&
127         savingsBalance >= 25000
128       ) {
129         console.log("calling from E");
130         return sendDataToDB(loanData);
131       } else if (Form.loanType == "E" && Form.amount >= 50000) {
132         return Alert.alert("Emergency Loans must be less than 50K");
133       } else if (Form.loanType == "E" && Form.duration > 6) {
134         return Alert.alert("Emergency Loans must be paid within 6 months");
135       } else if (Form.loanType == "E" && savingsBalance <= 25000) {
136         return Alert.alert(
137           "you must have at least 25000 in savings to request for an emergency loan"
138           );
139       } else if (Form.loanType == "E" && Form.amount <= 5000) {
```



```
const loanRequest = () => {
  async function handleSubmit() {
        "you must have at least 25000 in savings to request for an emergency loan"
      );
    } else if (Form.loanType == "E" && Form.amount <= 5000) {
      return Alert.alert("you must request for above 5,000");
    }

    if (
      Form.loanType == "N" &&
      Form.amount <= Number(approveAmount) &&
      Form.duration <= 12
    ) {
      console.log("calling from N");
      return sendDataToDB(loanData);
    } else if (Form.loanType == "N" && Form.amount >= Number(approveAmount)) {
      return Alert.alert(`you can only take a loan of ${approveAmount}`);
    } else if (Form.loanType == "N" && Form.duration > 12) {
      return Alert.alert("duration of loan type is 12months max");
    } else {
      return Alert.alert(
        "unable to process request, most likely amount exceeds balance"
      );
    }
  } else if (Form.loanType == "S" && Form.amount <= approveAmount2) {
    console.log("calling from S");
    return sendDataToDB();
  } else if (Form.loanType == "S" && Form.amount >= approveAmount2) {
    console.log("here");
    return Alert.alert(
      `Your Special loan is only permitted up to ${approveAmount2}`
    );
  } else {
    return Alert.alert("Invalid Detail");
  }

  console.log("savings " + savingsBalance);
  console.log(
```

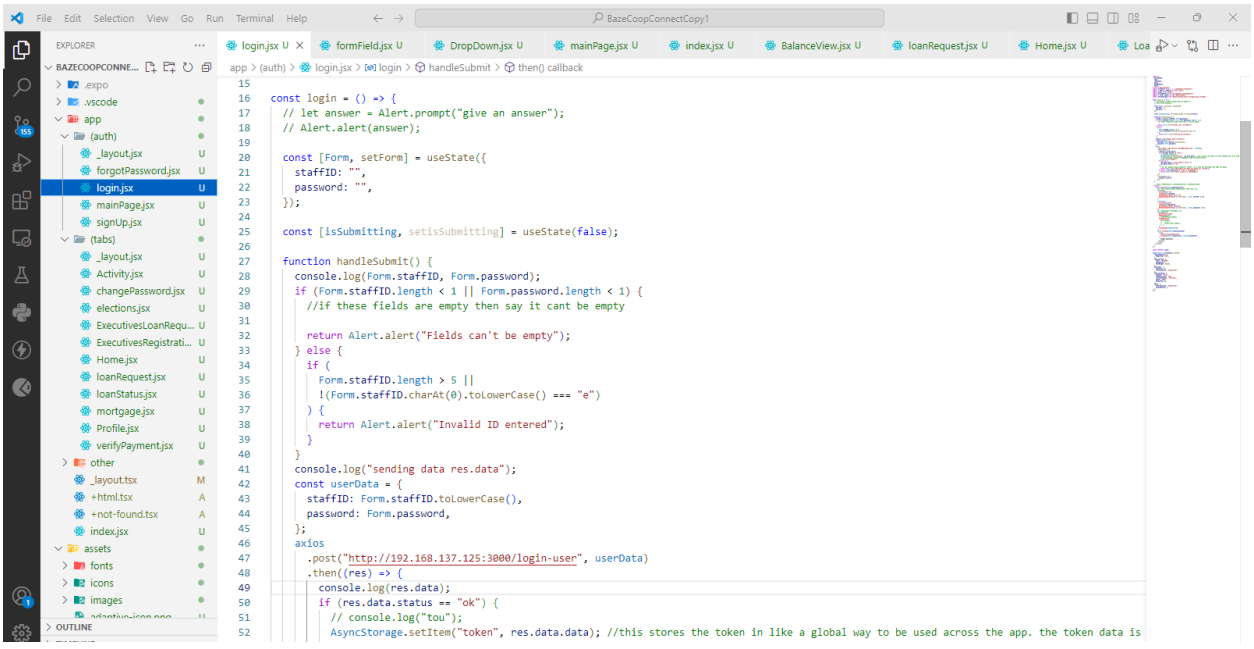


```
const loanRequest = () => {
useEffect(() => {
  getPaidData();
}, []);

return (
  <View style={styles.safeArea}>
    <Header page={"Loan Request"} />
    <ScrollView
      style={styles.container2}
      // contentContainerStyle={{ height: "100%" }}
      automaticallyAdjustKeyboardInsets={true}
    >
      <Text style={styles.label}>loan type</Text>
      <View style={styles.containerr}>
        {/* <Text>Select an option:</Text> */}
        <RNPickerSelect
          placeholder={{
            label: "Select Loan Type",
            value: null,
          }}
          placeholderTextColor="black"
          items={[
            { label: "Emergency Loan", value: "E" },
            { label: "Normal Loan", value: "N" },
            { label: "Special Loan", value: "S" },
          ]}
          onValueChange={(value) => {
            setForm({ ...Form, loanType: value });
          }}
          value={Form.loanType}
          style={styles.placeholder}
        />
        {/* {selectedValue && <Text>Selected: {selectedValue}</Text>} */}
      </View>

      <Text style={styles.label}>Amount</Text>
```

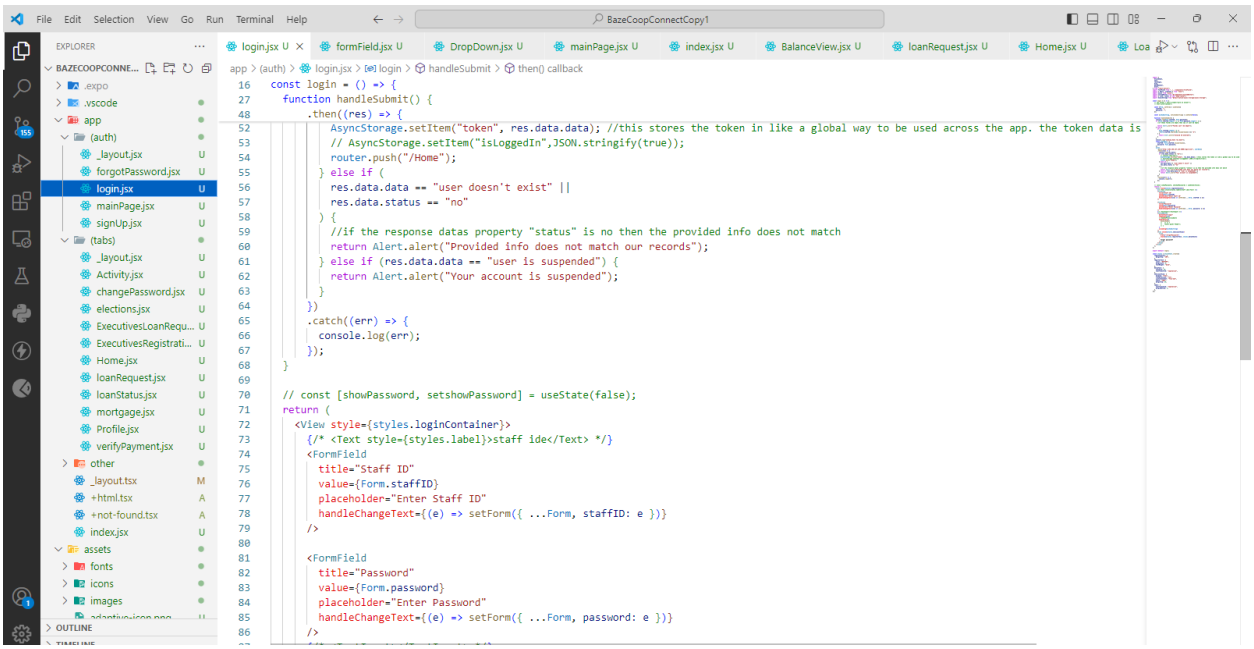


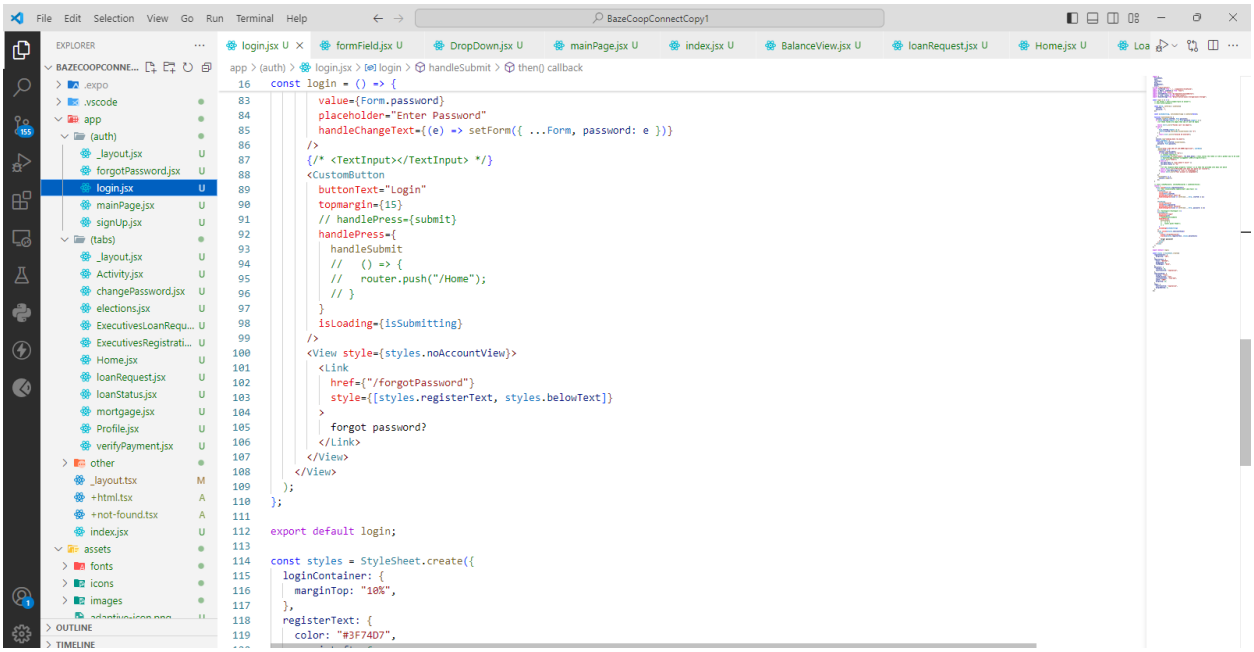
```
const loanRequest = () => {

  <Text style={styles.label}>Amount</Text>

  <FormField
    title={"Amount"}
    value={Form.amount}
    placeholder={"Amount"}
    handleChangeText={(e) => setForm({ ...Form, amount: e })}
    // styless={[{ height: 100 }]}
  />
  <Text style={styles.interest}>
    Intrest: {Form.amount * 0.1} you will be paid{" "}
    {Form.amount - Form.amount * 0.1}
  </Text>

  <Text style={styles.label}>duration</Text>
  <View style={styles.containerr}>
    {/* <Text>Select an option:</Text> */}
    <RNPickerSelect
      placeholder={{
        label: "Select Preferred Duration",
        value: null,
      }}
      placeholderTextColor="black"
      items={[
        { label: "1 month", value: 1 },
        { label: "2 months", value: 2 },
        { label: "3 months", value: 3 },
        { label: "4 months", value: 4 },
        { label: "5 months", value: 5 },
        { label: "6 months", value: 6 },
        { label: "7 months", value: 7 },
        { label: "8 months", value: 8 },
        { label: "9 months", value: 9 },
        { label: "10 months", value: 10 },
        { label: "11 months", value: 11 },
        { label: "12 months", value: 12 },
```

Login Page

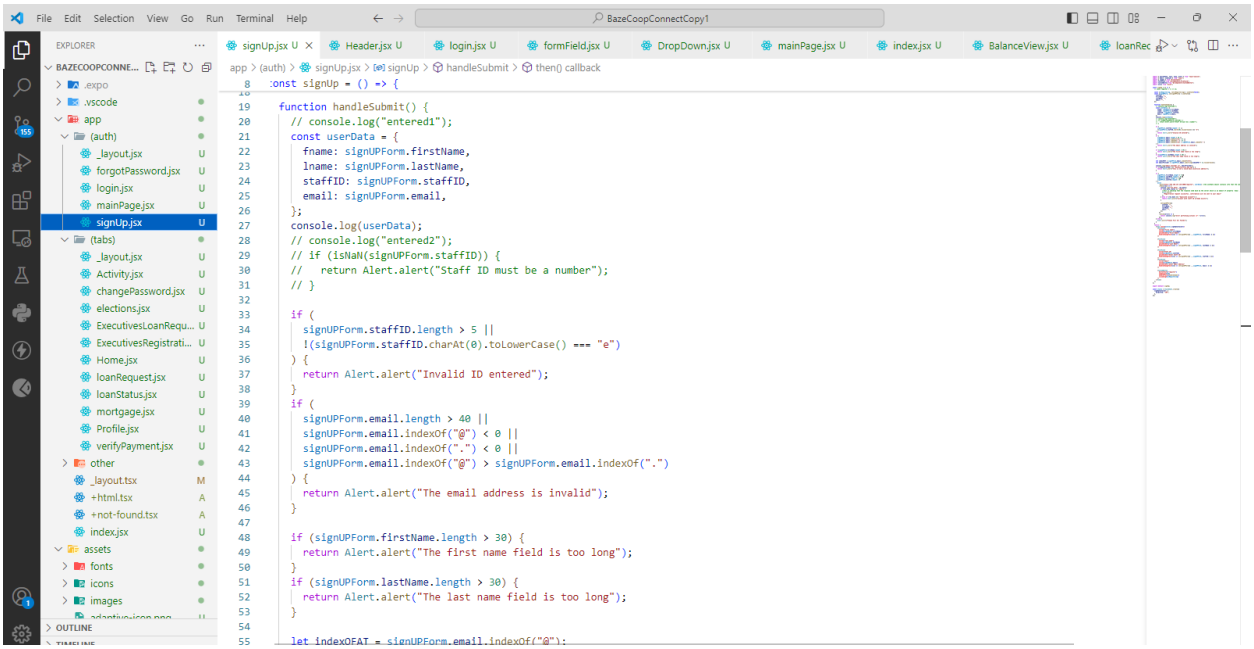


```
const login = () => {
  // let answer = Alert.prompt("give an answer");
  // Alert.alert(answer);

  const [Form, setForm] = useState({
    staffID: "",
    password: "",
  });

  const [isSubmitting, setisSubmitting] = useState(false);

  function handleSubmit() {
    console.log(Form.staffID, Form.password);
    if (Form.staffID.length < 1 || Form.password.length < 1) {
      //if these fields are empty then say it cant be empty

      return Alert.alert("Fields can't be empty");
    } else {
      if (
        Form.staffID.length > 5 ||
        !(Form.staffID.charAt(0).toLowerCase() === "e")
      ) {
        return Alert.alert("Invalid ID entered");
      }
    }
    console.log("sending data res.data");
    const userData = {
      staffID: Form.staffID.toLowerCase(),
      password: Form.password,
    };
    axios
      .post("http://192.168.137.125:3000/login-user", userData)
      .then((res) => {
        console.log(res.data);
        if (res.data.status == "ok") {
          // console.log("tou");
          AsyncStorage.setItem("token", res.data.data); //this stores the token in like a global way to be used across the app. the token data is
```

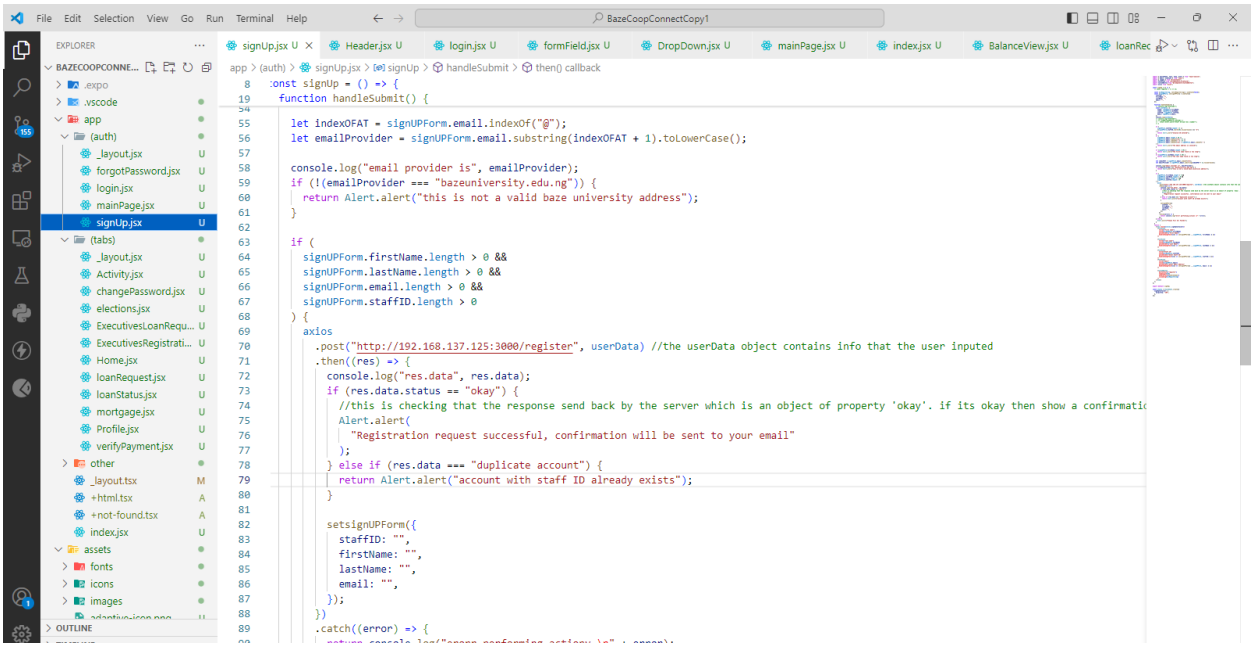




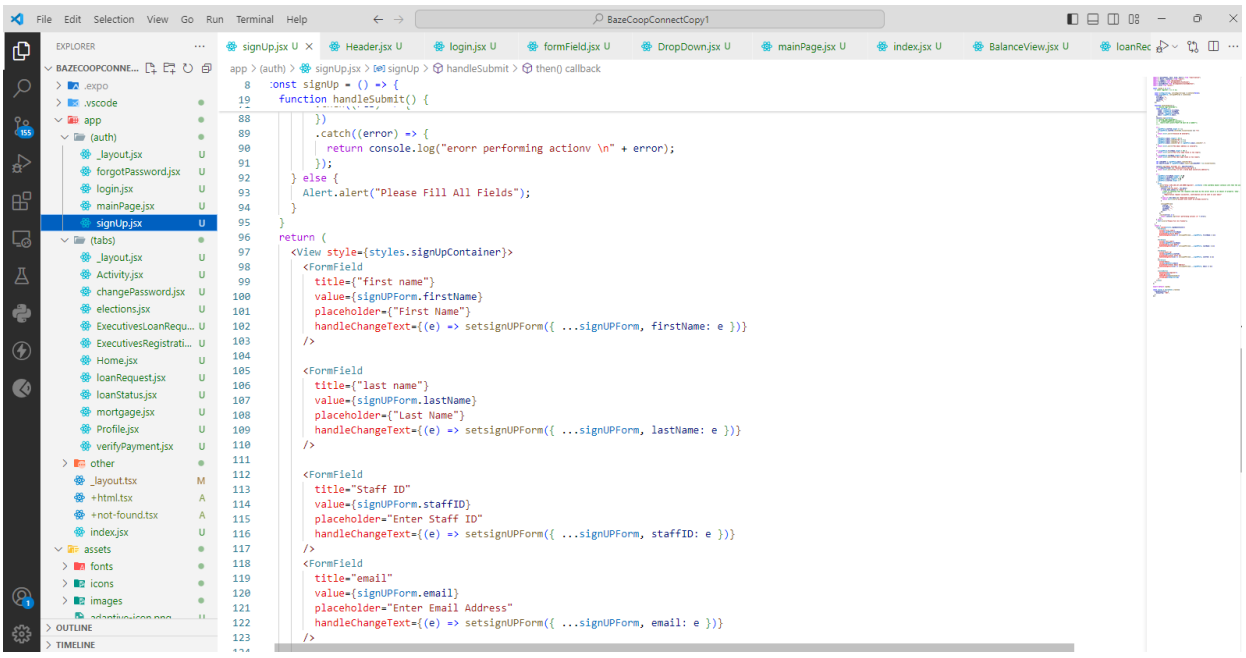
Sign Up Page



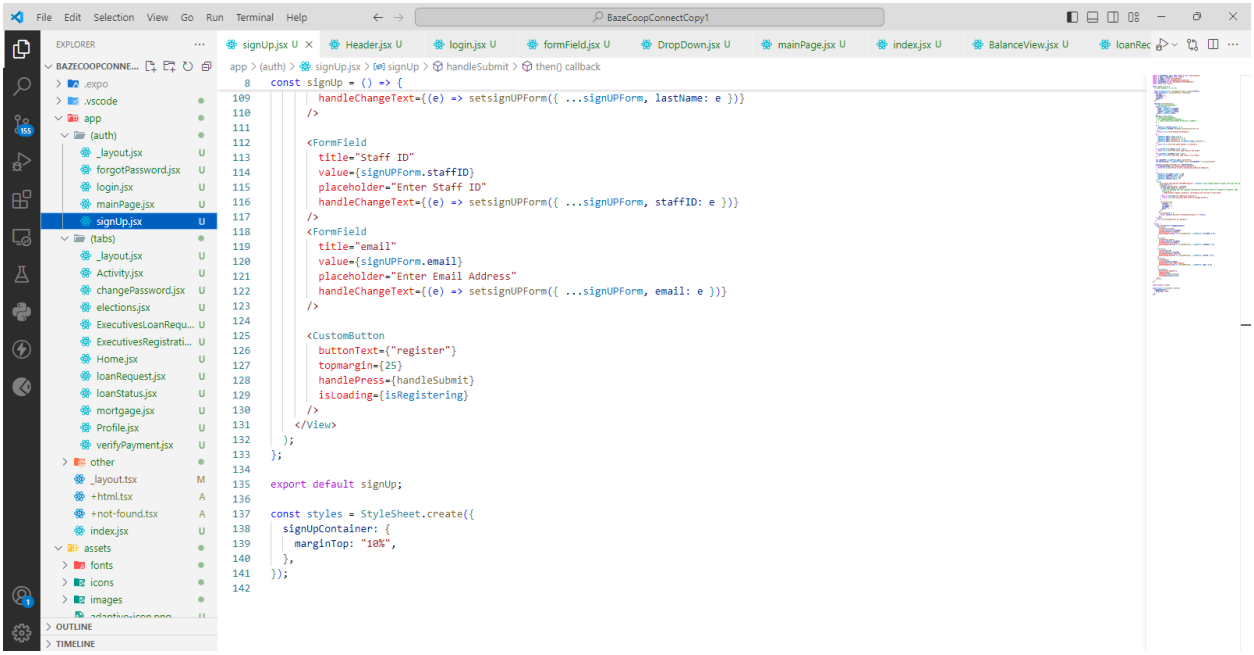
```
function handleSubmit() {
  // console.log("entered1");
  const userData = {
    fname: signUPForm.firstName,
    lname: signUPForm.lastName,
    staffID: signUPForm.staffID,
    email: signUPForm.email,
  };
  console.log(userData);
  // console.log("entered2");
  // if (isNaN(signUPForm.staffID)) {
  //   return Alert.alert("Staff ID must be a number");
  // }

  if (
    signUPForm.staffID.length > 5 ||
    !(signUPForm.staffID.charAt(0).toLowerCase() === "e")
  ) {
    return Alert.alert("Invalid ID entered");
  }
  if (
    signUPForm.email.length > 40 ||
    signUPForm.email.indexOf("@") < 0 ||
    signUPForm.email.indexOf(".") < 0 ||
    signUPForm.email.indexOf("@") > signUPForm.email.indexOf(".")
  ) {
    return Alert.alert("The email address is invalid");
  }

  if (signUPForm.firstName.length > 30) {
    return Alert.alert("The first name field is too long");
  }
  if (signUPForm.lastName.length > 30) {
    return Alert.alert("The last name field is too long");
  }

  let indexOFAT = signUPForm.email.indexOf("@");
```



```
function handleSubmit() {
  let indexOFAT = signUPForm.email.indexOf("@");
  let emailProvider = signUPForm.email.substring(indexOFAT + 1).toLowerCase();

  console.log("email provider is", emailProvider);
  if (!(emailProvider === "bazeuniversity.edu.ng")) {
    return Alert.alert("this is not a valid baze university address");
  }

  if (
    signUPForm.firstName.length > 0 &&
    signUPForm.lastName.length > 0 &&
    signUPForm.email.length > 0 &&
    signUPForm.staffID.length > 0
  ) {
    axios
      .post("http://192.168.137.125:3000/register", userData) //the userData object contains info that the user inputed
      .then((res) => {
        console.log("res.data", res.data);
        if (res.data.status == "okay") {
          //this is checking that the response send back by the server which is an object of property 'okay'. if its okay then show a confirmatic
          Alert.alert(
            "Registration request successful, confirmation will be sent to your email"
          );
        } else if (res.data === "duplicate account") {
          return Alert.alert("account with staff ID already exists");
        }

        setsignUPForm({
          staffID: "",
          firstName: "",
          lastName: "",
          email: "",
        });
      })
      .catch((error) => {
```

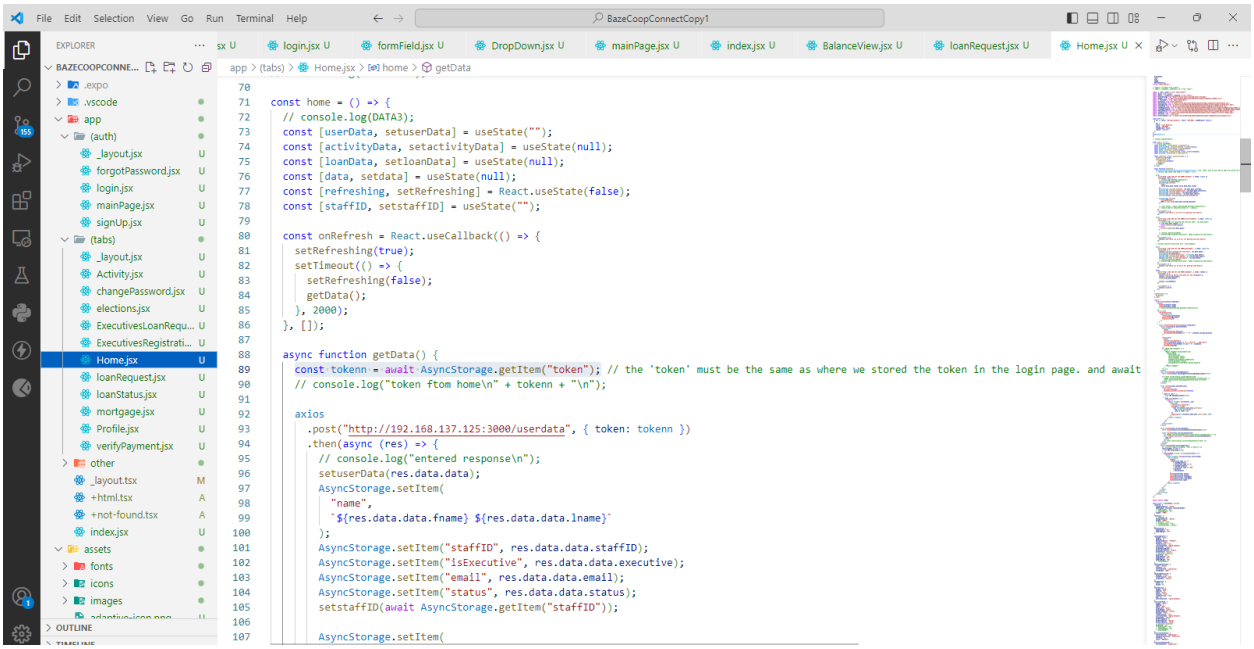


```
:onst signUp = () => {
 function handleSubmit() {
        })
        .catch((error) => {
          return console.log("erorr performing actionv \n" + error);
        });
    } else {
      Alert.alert("Please Fill All Fields");
    }
  }
  return (
    <View style={styles.signUpContainer}>
      <FormField
        title={"first name"}
        value={signUPForm.firstName}
        placeholder={"First Name"}
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, firstName: e })}
      />

      <FormField
        title={"last name"}
        value={signUPForm.lastName}
        placeholder={"Last Name"}
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, lastName: e })}
      />

      <FormField
        title="Staff ID"
        value={signUPForm.staffID}
        placeholder="Enter Staff ID"
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, staffID: e })}
      />
      <FormField
        title="email"
        value={signUPForm.email}
        placeholder="Enter Email Address"
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, email: e })}
      />
```

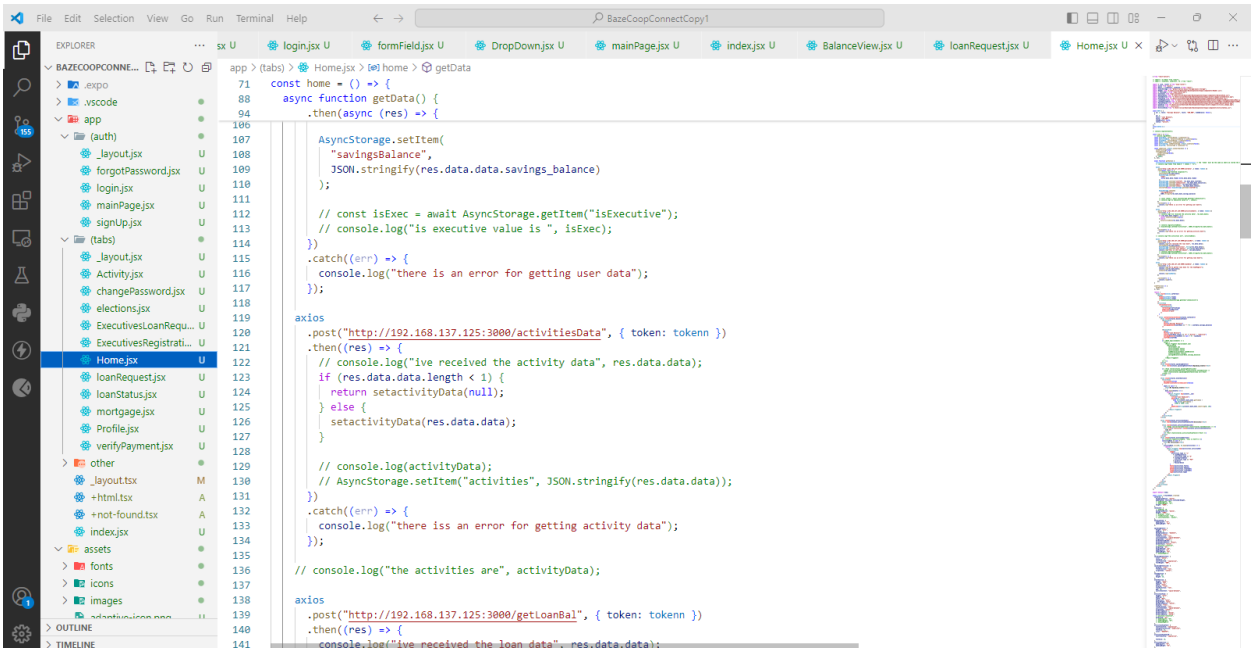


```
const signUp = () => {
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, lastName: e })}
      />

      <FormField
        title="Staff ID"
        value={signUPForm.staffID}
        placeholder="Enter Staff ID"
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, staffID: e })}
      />
      <FormField
        title="email"
        value={signUPForm.email}
        placeholder="Enter Email Address"
        handleChangeText={(e) => setsignUPForm({ ...signUPForm, email: e })}
      />

      <CustomButton
        buttonText={"register"}
        topmargin={25}
        handlePress={handleSubmit}
        isLoading={isRegistering}
      />
    </View>
  );
};

export default signUp;

const styles = StyleSheet.create({
  signUpContainer: {
    marginTop: "10%",
  },
});
```

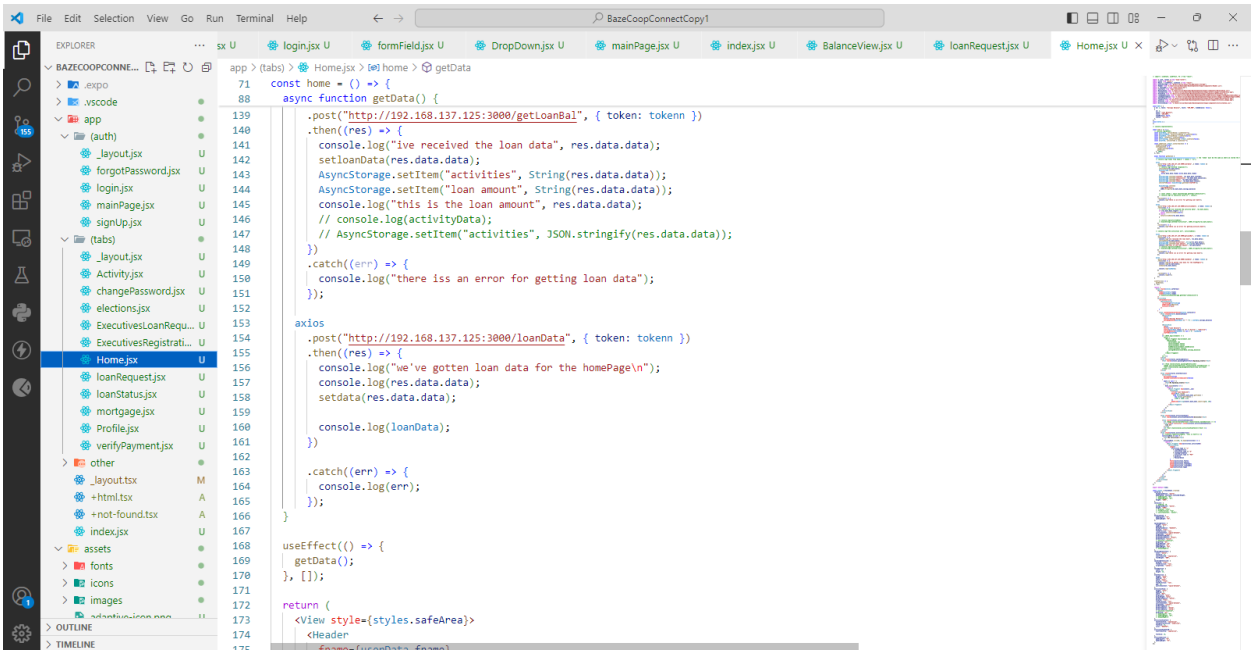
Home-Page



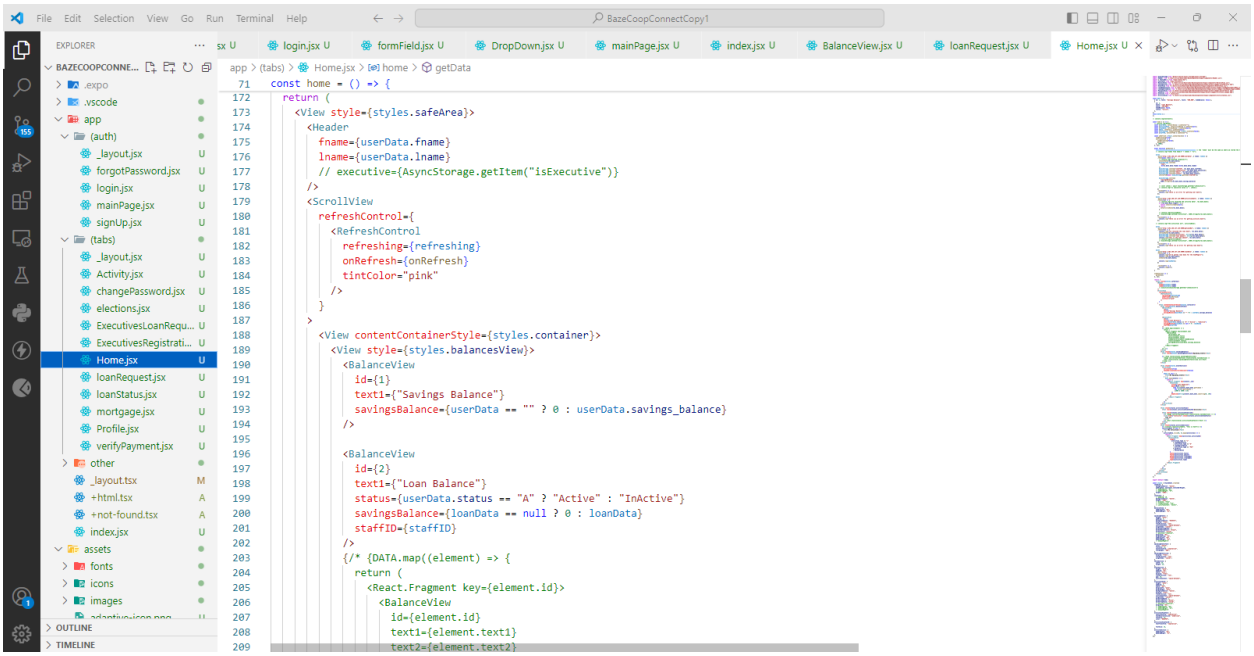
```
const home = () => {
  // console.log(DATA3);
  const [userData, setuserData] = useState("");
  const [activityData, setactivityData] = useState(null);
  const [loanData, setloanData] = useState(null);
  const [data, setdata] = useState(null);
  const [refreshing, setRefreshing] = React.useState(false);
  const [staffID, setstaffID] = useState("");

  const onRefresh = React.useCallback(() => {
    setRefreshing(true);
    setTimeout(() => {
      setRefreshing(false);
      getData();
    }, 2000);
  }, []);

  async function getData() {
    const tokenn = await AsyncStorage.getItem("token"); // the 'token' must be the same as where we stored the token in the login page. and await
    // console.log("token ftom home\n" + tokenn + "\n");

    axios
      .post("http://192.168.137.125:3000/userdata", { token: tokenn })
      .then(async (res) => {
        // console.log("entered response\n");
        setuserData(res.data.data);
        AsyncStorage.setItem(
          "name",
          `${res.data.data.fname} ${res.data.data.lname}`
        );
        AsyncStorage.setItem("staffID", res.data.data.staffID);
        AsyncStorage.setItem("isExecutive", res.data.data.executive);
        AsyncStorage.setItem("email", res.data.data.email);
        AsyncStorage.setItem("status", res.data.data.status);
        setstaffID(await AsyncStorage.getItem("staffID"));


        AsyncStorage.setItem(
```



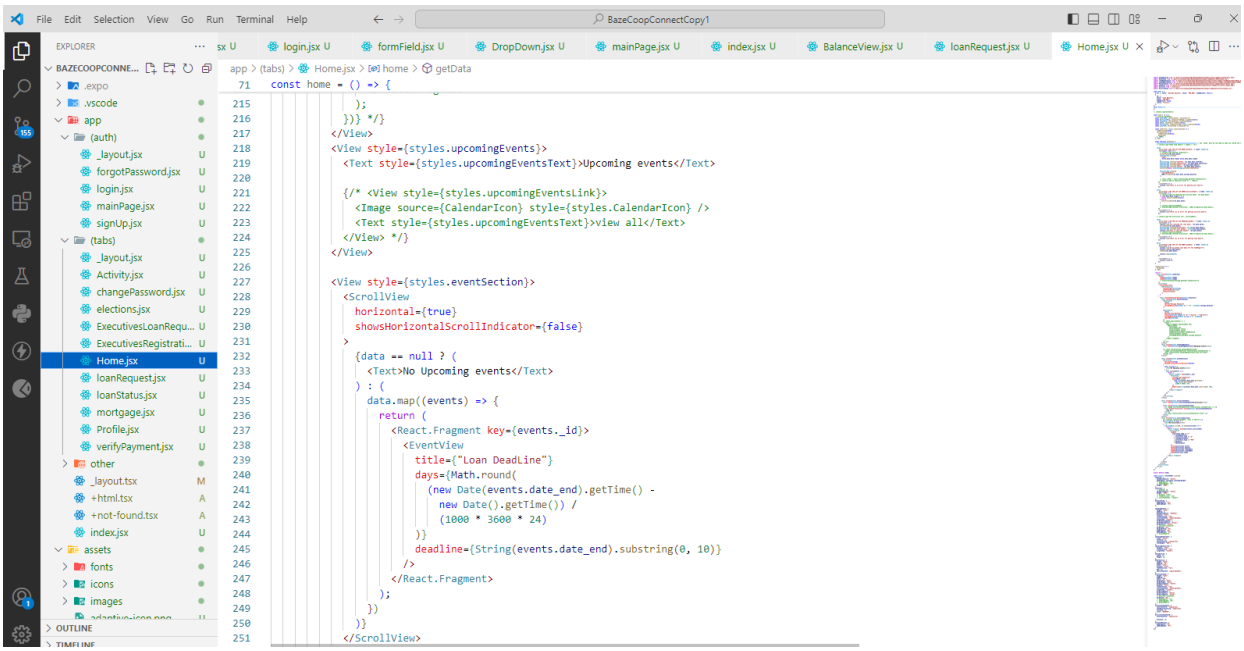
```
const home = () => {
  async function getData() {
      .then(async (res) => {

        AsyncStorage.setItem(
          "savingsBalance",
          JSON.stringify(res.data.data.savings_balance)
        );

        // const isExec = await AsyncStorage.getItem("isExecutive");
        // console.log("is executive value is ", isExec);
      })
      .catch((err) => {
        console.log("there is an error for getting user data");
      });

  axios
    .post("http://192.168.137.125:3000/activitiesData", { token: tokenn })
    .then((res) => {
      // console.log("ive received the activity data", res.data.data);
      if (res.data.data.length < 1) {
        return setactivityData(null);
      } else {
        setactivityData(res.data.data);
      }

      // console.log(activityData);
      // AsyncStorage.setItem("activities", JSON.stringify(res.data.data));
    })
    .catch((err) => {
      console.log("there iss an error for getting activity data");
    });

  // console.log("the activities are", activityData);

  axios
    .post("http://192.168.137.125:3000/getLoanBal", { token: tokenn })
    .then((res) => {
      console.log("ive received the loan data", res.data.data);
```



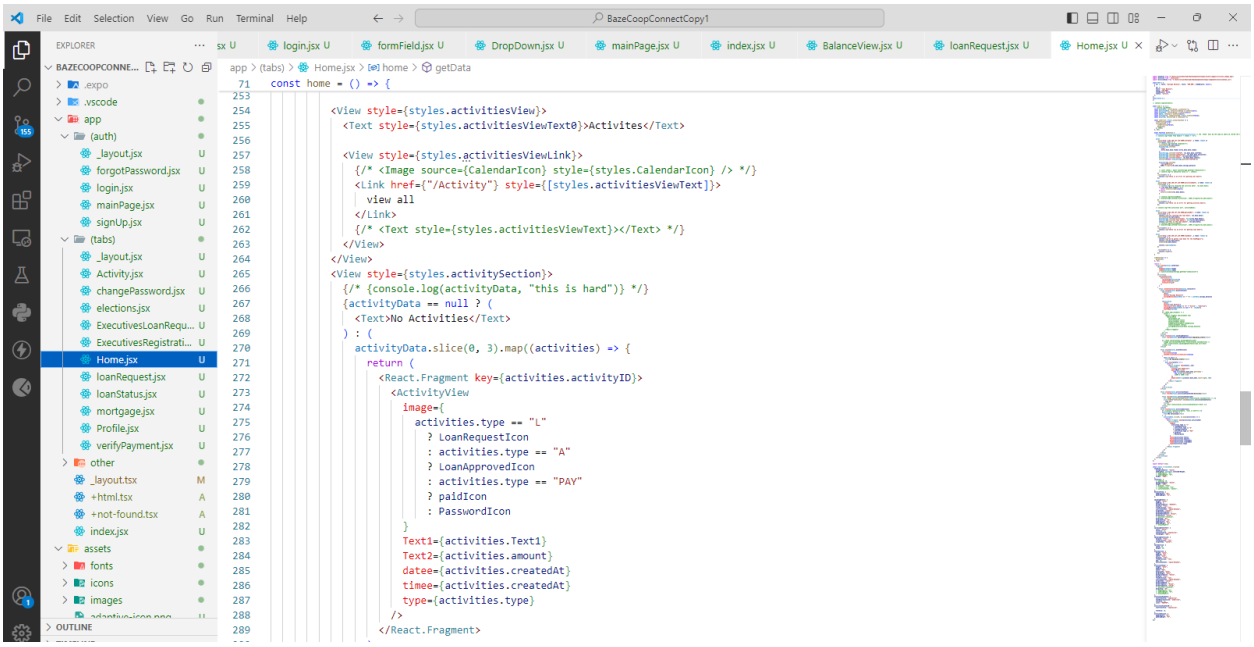
```
const home = () => {
  async function getData() {
    .post("http://192.168.137.125:3000/getLoanBal", { token: tokenn })
    .then((res) => {
      console.log("ive received the loan data", res.data.data);
      setloanData(res.data.data);
      AsyncStorage.setItem("activities", String(res.data.data));
      AsyncStorage.setItem("loan amount", String(res.data.data));
      console.log("this is the loan amount", res.data.data);
      // console.log(activityData);
      // AsyncStorage.setItem("activities", JSON.stringify(res.data.data));
    })
    .catch((err) => {
      console.log("there iss an error for getting loan data");
    });

  axios
    .post("http://192.168.137.125:3000/loanData", { token: tokenn })
    .then((res) => {
      console.log("we've gotten loan data for the homePage\n");
      console.log(res.data.data);
      setdata(res.data.data);

      console.log(loanData);
    })

    .catch((err) => {
      console.log(err);
    });
}

useEffect(() => {
  getData();
}, []);

return (
  <View style={styles.safeArea}>
    <Header
```



```
const home = () => {
return (
  <View style={styles.safeArea}>
    <Header
      fname={userData.fname}
      lname={userData.lname}
      // executive={AsyncStorage.getItem("isExecutive")}
    />
    <ScrollView
      refreshControl={
        <RefreshControl
          refreshing={refreshing}
          onRefresh={onRefresh}
          tintColor="pink"
        />
      }
    >
      <View contentContainerStyle={styles.container}>
        <View style={styles.balancesView}>
          <BalanceView
            id={1}
            text1={"Savings Balance"}
            savingsBalance={userData == "" ? 0 : userData.savings_balance}
          />

          <BalanceView
            id={2}
            text1={"Loan Balance"}
            status={userData.status == "A" ? "Active" : "InActive"}
            savingsBalance={loanData == null ? 0 : loanData}
            staffID={staffID}
          />
          {/* {DATA.map((element) => {
            return (
              <React.Fragment key={element.id}>
                <BalanceView
                  id={element.id}
                  text1={element.text1}
                  text2={element.text2}
```

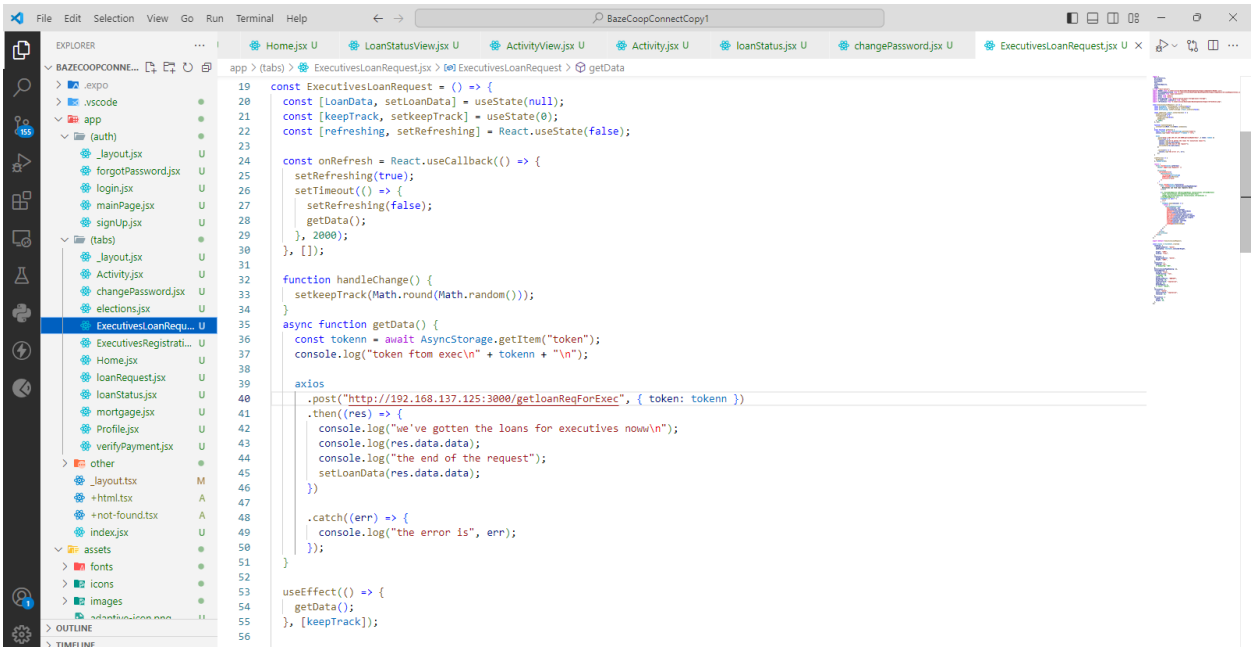


```
const home = () => {
                );
              })} */}
            </View>
            <View style={styles.upcomingEvents}>
              <Text style={styles.upcomingEventsText}>Upcoming events</Text>

              {/* <View style={styles.upcomingEventsLink}>
                <Image source={CalendarIcon} style={styles.CalendarIcon} />
                <Text style={styles.upcomingEventsText}>view all</Text>
              </View> */}
            </View>

            <View style={styles.eventSection}>
              <ScrollView
                horizontal={true}
                showsHorizontalScrollIndicator={false}
              >
                {data == null ? (
                  <Text>No Upcoming events</Text>
                ) : (
                  data.map((events) => {
                    return (
                      <React.Fragment key={events._id}>
                        <EventView
                          title={"Loan DeadLine"}
                          days={Math.round(
                            (new Date(events.date_end).getTime() -
                              new Date().getTime()) /
                              (1000 * 3600 * 24)
                          )}
                          deadline={String(events.date_end).substring(0, 10)}
                        />
                      </React.Fragment>
                    );
                  })
                )}
              </ScrollView>
```

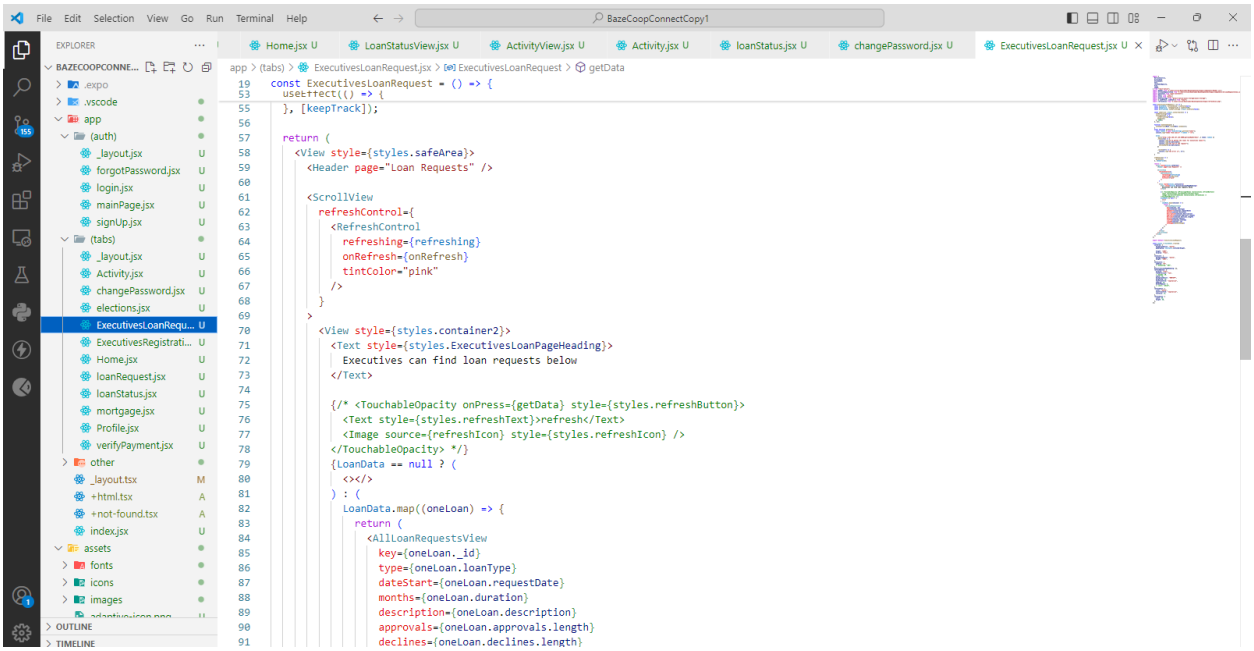


```
const home = () => {

            <View style={styles.activitiesView}>
              <Text style={styles.activitiesViewText0}>Activites</Text>

              <View style={styles.activitiesViewLink}>
                {/* <Image source={CalendarIcon} style={styles.CalendarIcon} /> */}
                <Link href={"/Activity"} style={[styles.activitiesViewText]}>
                  view all
                </Link>
                {/* <Text style={styles.activitiesViewText}></Text> */}
              </View>
            </View>
            <View style={styles.activitySection}>
              {/* {console.log(activityData, "this is hard")} */}
              {activityData == null ? (
                <Text>No Activities</Text>
              ) : (
                activityData.slice(0, 3).map((activities) => {
                  return (
                    <React.Fragment key={activities.activityID}>
                      <ActivityView
                        image={
                          activities.type == "L"
                            ? LoanRequestIcon
                            : activities.type == "A"
                            ? LoanApprovedIcon
                            : activities.type == "PAY"
                            ? paidIcon
                            : PasswordIcon
                        }
                        Text1={activities.Text1}
                        Text2={activities.amount}
                        datee={activities.createdAt}
                        timee={activities.createdAt}
                        type={activities.type}
                      />
                    </React.Fragment>
```

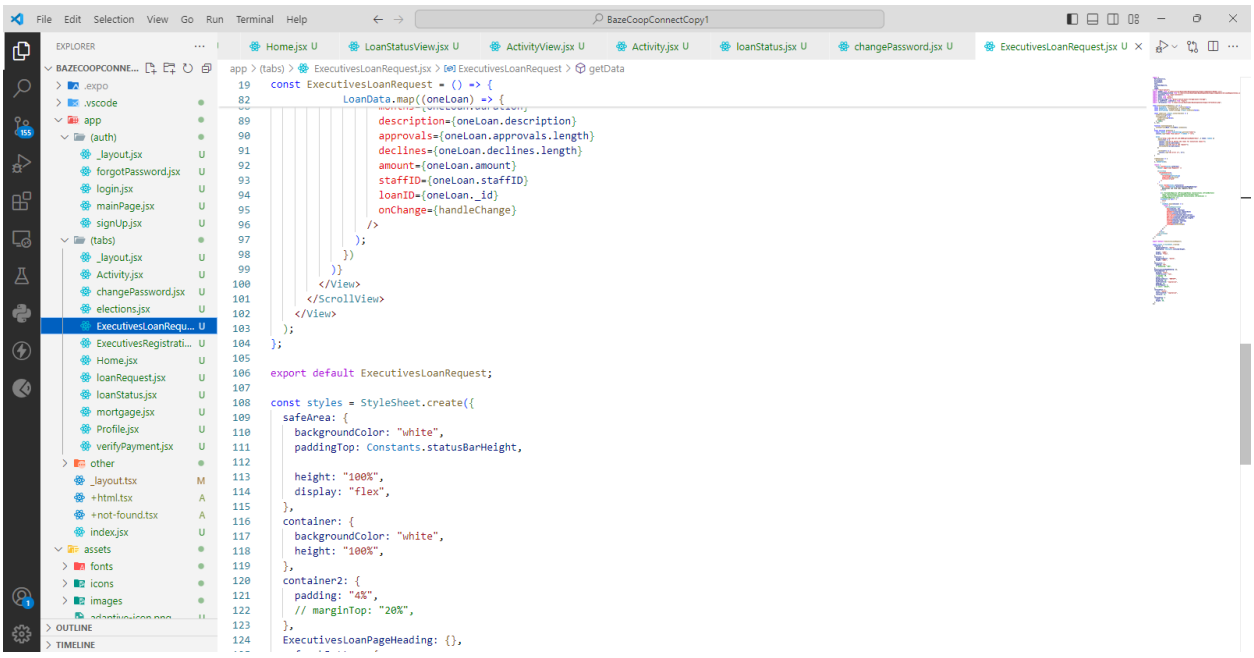
Executives loan request page



```
const ExecutivesLoanRequest = () => {
  const [LoanData, setLoanData] = useState(null);
  const [keepTrack, setkeepTrack] = useState(0);
  const [refreshing, setRefreshing] = React.useState(false);

  const onRefresh = React.useCallback(() => {
    setRefreshing(true);
    setTimeout(() => {
      setRefreshing(false);
      getData();
    }, 2000);
  }, []);

  function handleChange() {
    setkeepTrack(Math.round(Math.random()));
  }
  async function getData() {
    const tokenn = await AsyncStorage.getItem("token");
    console.log("token ftom exec\n" + tokenn + "\n");

    axios
      .post("http://192.168.137.125:3000/getloanReqForExec", { token: tokenn })
      .then((res) => {
        console.log("we've gotten the loans for executives noww\n");
        console.log(res.data.data);
        console.log("the end of the request");
        setLoanData(res.data.data);
      })

      .catch((err) => {
        console.log("the error is", err);
        });
  }

  useEffect(() => {
    getData();
  }, [keepTrack]);

```



```
const ExecutivesLoanRequest = () => {
  useEffect(() => {
  }, [keepTrack]);

  return (
    <View style={styles.safeArea}>
      <Header page="Loan Requests" />

      <ScrollView
        refreshControl={
          <RefreshControl
            refreshing={refreshing}
            onRefresh={onRefresh}
            tintColor="pink"
          />
        }
      >
        <View style={styles.container2}>
          <Text style={styles.ExecutivesLoanPageHeading}>
            Executives can find loan requests below
          </Text>

          {/* <TouchableOpacity onPress={getData} style={styles.refreshButton}>
            <Text style={styles.refreshText}>refresh</Text>
            <Image source={refreshIcon} style={styles.refreshIcon} />
          </TouchableOpacity> */}
          {LoanData == null ? (
            <></>
          ) : (
            LoanData.map((oneLoan) => {
              return (
                <AllLoanRequestsView
                  key={oneLoan._id}
                  type={oneLoan.loanType}
                  dateStart={oneLoan.requestDate}
                  months={oneLoan.duration}
                  description={oneLoan.description}
                  approvals={oneLoan.approvals.length}
                  declines={oneLoan.declines.length}
```


File Edit Selection View Go Run Terminal Help
BazeCoopConnectCopy1
EXPLORER
Home.jsx U
LoanStatusView.jsx U
ActivityView.jsx U
Activity.jsx U
loanStatus.jsx U
changePassword.jsx U
ExecutivesLoanRequest.jsx U
app > (tabs) > ExecutivesLoanRequest.jsx > ExecutivesLoanRequest > getData
const ExecutivesLoanRequest = () => {
LoanData.map((oneLoan) => {
description={oneLoan.description}
approvals={oneLoan.approvals.length}
declines={oneLoan.declines.length}
amount={oneLoan.amount}
staffID={oneLoan.staffID}
loanID={oneLoan._id}
onChange={handleChange}
/>
);
})
)}
</View>
</ScrollView>
</View>
);
};
export default ExecutivesLoanRequest;
const styles = StyleSheet.create({
safeArea: {
backgroundColor: "white",
paddingTop: Constants.statusBarHeight,
height: "100%",
display: "flex",
},
container: {
backgroundColor: "white",
height: "100%",
},
container2: {
padding: "4%",
// marginTop: "20%",
},
ExecutivesLoanPageHeading: {},
BAZECOOPCONNE...
.expo
.vscode
app
(auth)
_layout.jsx
forgotPassword.jsx
login.jsx
mainPage.jsx
signUp.jsx
(tabs)
_layout.jsx
Activity.jsx
changePassword.jsx
elections.jsx
ExecutivesLoanRequ...
ExecutivesRegistrati...
Home.jsx
loanRequest.jsx
loanStatus.jsx
mortgage.jsx
Profile.jsx
verifyPayment.jsx
other
_layout.tsx
+html.tsx
+not-found.tsx
index.jsx
assets
fonts
icons
images
OUTLINE
TIMELINE