ADMINISTRATION SYSTEM INTERFACE FOR APIKKO INC REMITTANCE SERVICES

KERJA PRAKTEK REPORT

UBAYA

UNIVERSITAS SURABAYA

Edited by:

Lukita Iswara

NRP: 160919005

INFORMATION TECHNOLOGY DUAL DEGREE PROGRAM
DEPARTMENT OF INFORMATICS ENGINEERING
UNIVERSITY OF SURABAYA
SURABAYA

July 2023

LEMBAR PENGESAHAN

ADMINISTRATION SYSTEM INTERFACE FOR

APIKKO INC REMITTANCE SERVICES

KERJA PRAKTEK

Diajukan Sebagai Syarat Untuk Pelaksanaan Mata Kuliah Wajib

Kerja Praktek University of Surabaya Information Teknology

Diterima dan Disetujui oleh:

Dosen Pembimbing	**Pembimbing Perusahaan**
(Andre S.T., M.Sc.)	(Dr. Drs. Haryanto M.T.)

PREFACE

I am pleased to deliver this research titled " Administration System Interface For Apikko Inc Remittance Service." This report results from an in-depth investigation, data analysis, and careful consideration of the importance of the admin user interface. This research aims to develop the administration system interface for Apikko Inc. remittance services.

I want to take this opportunity to express my sincere appreciation to the following individuals, without whom the completion of this Internship Report would not have been possible:

1. Mr. Dr. Joko Siswantoro, S.Si, M.Si., is the Head of the Department of Informatics Engineering, Faculty of Engineering, University of Surabaya.
2. Andre S.T., M.Sc., as the supervisor, generously dedicated his time to guide and provide assistance and valuable input to the author in completing this internship.
3. Dr. Drs. Haryanto M.T, the owner of APIKKO., willingly guided me and allowed me to complete this system.
4. My family supports me throughout this journey.

I appreciate you taking an interest in this report. I trust you will find the information informative and help you make well-informed choices.

Surabaya, July 2023

Author

Lukita Iswara

CONTENTS

CHAPTER I INTRODUCTION

This chapter provides an overview of the research background, question, objective, Implication, scope, activity plan, and systematic of the report. The background section outlines previous studies and gaps in the literature. The research question and objective guide the study and specify the intended outcomes. The implication section discusses potential practical or theoretical contributions. The scope clarifies the boundaries of the research. The activity plan outlines specific research activities. The systematic of the report outlines the structure and organization of the research report.

1.1 Background

Apikko Inc (Indonesian name) or Haoti (Hongkong name) is a money transfer service from Hong Kong to Indonesia. Its main business is in remittance, in other words, it transfers money from one party to another. Most of their customers work in Hongkong and transfer their money from Hongkong to Indonesia. Apikko Inc was founded in 2012 by Haryanto from Surabaya. It has a branch in Hongkong which they named Haoti. The size of the company is about a small enterprise, so it has about 40 employees working in the company.

Apikko Inc needed a new user interface for their administrator since they had a problem doing their administrator system manually. Apikko Inc proposed to add a database management system (DBMS) user interface (UI) for their administrator. A database management system (DBMS) is a computer's organization and processing system. This DBMS can also help with data maintenance and processing by employing a DBMS so that it does not cause confusion and can be utilized by users as needed. For example, corporations often use DBMS to find which customer has given a recommendation and to check any suspicious activity in their customer data. But not many small

enterprises had the chance to have the technology. As a result, numerous micro-enterprises heavily relied on human labor, which slowed the administration process. Concurrently, human labored administrator tasks may take hours for the customer's input to proceed, and in some cases, the administrator may not process some customers' requests.

Currently, Apikko Inc operates its database administration by human labor. Apikko Inc relies on programmers to get specific data they need. If they wanted to get some detailed data, they needed to make a request often to their programmer. As a result, their database management system is processed slowly because of the limited capacity of programmers for handling the data. Apikko Inc planned to reduce this error by suggesting a UI which are easy to understand for a non-programmer to get detailed case data in their databases and accommodate administrator needs. The administrator interface will have a database management system interface that helps the administrator to process the database and give immediate data in specific circumstances. The database will also use charts to support their data analyst to analyze some data. The Apikko's administrator works on the Transaction and Customers database.

Apikko inc previously did not have any DBMS system for its nonprogrammer, so everything is done by using phpMyAdmin SQL. The new system that APIKKO will be built is going to help many non-programmers to be able to edit the data from the database. It also lets them search data that does not need SQL if the users want to find data in a specific condition. The users find the data by editing the setting in the menu. From here, it may help many non-programmers to use the database without any interaction from the programmer. As a result, it reduces the workload in APIKKO and makes the work faster.

1.2 Problem Statement

The problem statement is, "how to support administrators tasks that are related with data management efficiency?"

1.3 Research Objective

The objective is to develop the administration system interface for Apikko Inc. remittance services

1.4 Implication

The Implication that is expected from the project is as follows:

- Provide an easy-to-understand admin interface.
- Provide an interface to find data in specific circumstances and needs.
- Provide the administrator an interface to help the admin get customer insight from system-generated reports.

1.5 Scope

The scope of the project:

- The system only accessible to the administrator
- The system can only record customer and transaction data.
- The system provides customer transaction reports that have custom filter functionality and some exporting options to pdf or excel files.

1.6 Activity Plan

The plan of activities that will be carried out in making the final project can be structured as follows

1. **Preparation**

   This step includes strategies for data collection, such as surveys, interviews, or experiments, as well as methods for data analysis and interpretation.

2. **Analysis**

At this stage, there is a need to analyze the fields needed for the database. There's also a need to explore features that are required to be used by the administrator.

3. **Design**

   The designs that will be created in this area include system design, database design, and user interface design.

4. **Implementation**

   At the implementation stage, we will ensure that the available features can function as expected and provide no bugs in the application. The application is web-based on PHP.

5. **Experiment and Evaluation**

   At this stage, we will try to experiment and evaluate the application. We will also alpha and beta test the application.

6. **Report**

   This section prepares the report based on the steps carried out in the previous activity plan.

1.7 Systematics of The Report

The chapters will be separated into 7 chapters, as follows:

CHAPTER I: INTRODUCTION

The chapter outlines the issues that will be covered in later chapters. This chapter is divided into 7 major sections: research background, research question, research objective, implication, scope, activity plan and the Systematics of the Thesis Report.

CHAPTER II: LITERATURE REVIEW

The chapter discusses the theories employed in the system's design and implementation, particularly the DBMS.

CHAPTER III: SYSTEM ANALYSIS

The chapter explains similar systems In the DBMS and analysis the current similar system that is well known.

CHAPTER IV: SYSTEM DESIGN

The chapter explains the design of the system. The chapter also discusses the design of the data, process, and user interface.

CHAPTER V: IMPLEMENTATION SYSTEM

The chapter explains the transformation of the system design into a program. The implementation of the system consists of Data implementation, process implementation, and user interfaces.

CHAPTER VI: EXPERIMENT AND EVALUATION

The chapter discusses the experimentation on the performed system. The experiment is carried out through the verification step, and the experiment results are then analyzed to see whether the established system agrees with the system's goal.

CHAPTER VII: CONCLUSION

The chapter covers the results of the implementation and experiment of the system. The chapter also gave additional suggestions for ongoing system development.

CHAPTER II LITERATURE REVIEW

The Literature Review will give theoretical foundations or references supporting the application study's algorithms. The chapter's goal is to improve comprehension of the ideas. This literature study is built on the foundations of native PHP as a programming language, Apache as a web server, and MySQL DBMS as a database management system. Bootstrap is a CSS Framework that aids in the improvement of user interfaces.

2.1 Database Management System

A database management system (DBMS) is software that stores, manages, and ensures data security (Silberschatz et al, 2020). Before delving more into a DBMS, it's essential first to grasp what a database is. A database is a collection of information saved on a computer device. A database's data is structured to be controlled using specific commands. A database management system (DBMS) is software that connects databases to users for the data management process to function effectively. A DBMS is primarily responsible for managing data, the database engine, and the database schema so that the data management and organization process runs smoothly. In other words, a database management system (DBMS) is a visual intermediate that allows users to quickly access, edit, organize, and remove data in a database.

2.1.1 DBMS Functions

A DBMS performs numerous critical operations that ensure the database's integrity and consistency. Most of such functions are invisible to end users and can only be accomplished with the help of a DBMS.

1. **Data Dictionary Management:**

   The DBMS uses a data dictionary to hold definitions of data components and their relationships (metadata) in a data dictionary (Pratt et al, 2015). Any applications that access

database data do so via the DBMS. The data dictionary helps DBMS search for the appropriate data component structures and relationships, saving you from coding such complicated associations in each program.

2. **Data Storage Management**

The DBMS develops and manages the complex structures necessary for data storage, freeing you of the onerous process of specifying and programming the physical data properties. Data storage management is also critical for optimizing database performance.

3. **Data Transformation and Presentation**

The database management system (DBMS) converts entered data to correspond to specified data structures. The DBMS relieves you of the burden of differentiating between logical and physical data formats. That is, the DBMS prepares the physically obtained data to comply with the user's rational expectations (Reeve, 2013).

4. **Multi User Access Control**

The DBMS utilizes advanced algorithms to ensure that several users can access the database concurrently without compromising its integrity to offer data integrity and consistency (Bernard & Bachu, 2015).

5. **Backup and Recovery Management**

The DBMS provides backup and data recovery to assure data safety and integrity. Current DBMS systems have utilities that enable DBAs to execute ordinary and exceptional backup and restoration processes (Coronel & Morris, 2023). Recovery management is concerned with restoring a database following a failure, such as a faulty

sector on a disk or a power outage. This functionality is crucial for maintaining the database's integrity.

6. **Data integrity management**

The DBMS supports and enforces integrity standards, which reduces data redundancy and increases data consistency (Sumathi & Esakkirajan, 2007). Data integrity is implemented using the data relationships recorded in the data dictionary. The integrity of data is especially crucial in transaction-oriented database systems.

7. **Database access languages and application programming interfaces**

Data contained within the database can be retrieved and altered using a query language by the DBMS. By acting as a nonprocedural tool, this query language enables users to communicate their desired activities without explicitly specifying the step-by-step processes. Users can concentrate on outlining their information needs and the desired outcome rather than giving the exact instructions or algorithms for data retrieval or processing. (Bush, 2020).

8. **Data Security**

The database engine allows certain parties to access, modify, and lock data in the database (Vishwajit, 2020). As a result, the DBMS can restrict what end users view. Users are also not required to know where the data is kept or to be concerned about changes to the data structure. If each current software uses the DBMS's application programming interface (API) for the database, they can avoid tinkering with the program every time the DB changes.

2.1.2 DBMS Types

The following explanation will explain the types of DBMS:

1. **Hierarchical Database**

   A hierarchical database, also known as a parent-child relationship structure, is a database management system (Kahate, 2004). The data in the hierarchical type management system include information about the parent-child's relationship within their group.

2. **Network Database**

   A network structure builds relationships between entities in a network database. A network database is technically a subset of a hierarchical database (Yannakoudakis & Cheng, 1988). In a network database, however, an entity can have parent/child relationships with more than one other. Still, in a hierarchical database, an entity can only have parent/child relationships with one other entity.

3. **Relational Database**

   The links between data in a relational database management system (RDBMS) are relational, and the data is kept in tables with columns and rows (Harrington, 2016). Columns have characteristics, whereas rows hold records or data. Users must use Structured Query Language to operate RDBMS, such as adding, subtracting, deleting, and manipulating data.

2.1.3 MySQL DBMS

MySQL is a widely used open-source Relational Database Management System (RDBMS) for storing, organizing, and managing structured data. It was first launched in 1995, and Oracle Corporation owns and maintains it (Grippa, 2021).

MySQL is built on the Structured Query Language (SQL), a standardized language for handling data in relational databases. MySQL is widely utilized in numerous areas, including banking, healthcare, and e-commerce, because of its excellent performance, scalability, and stability.

SQL has four basic functions: Create, Read, Update, and Delete. Here are the basic SQL command operations:

a. Create

This action is used to add new data to a database. In the context of a relational database, this involves inserting a new record into a table. This procedure is carried out with the SQL INSERT command.

b. Read

Read functioned to retrieve data from a database. It involves picking data from one or more tables in the context of a relational database. The process is carried out using the SQL SELECT command.

c. Update

This action is used to alter existing data in a database. In the context of a relational database, this entails changing the values of one or more columns in a table. This procedure is carried out with the SQL UPDATE command.

d. Delete

This action is used to delete information from a database. In a relational database, the Delete operation removes records from the table. This procedure is carried out using the SQL DELETE command.

2.2 Application Architecture

The Presentation layer, Logic layer, and Data layer are the three primary levels that comprise an application's design in software development. Each layer serves a particular purpose and uniquely works with the others (Oussalah, 2014).

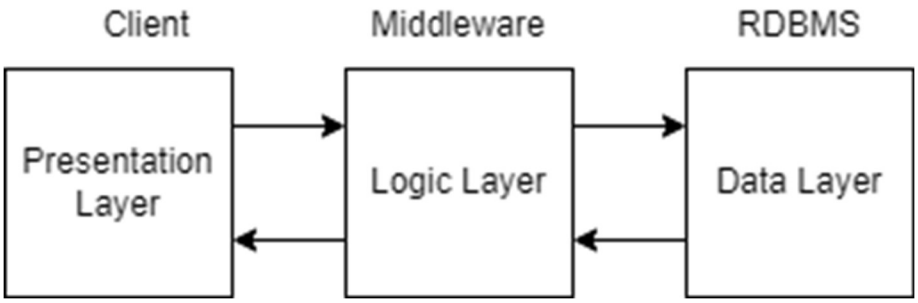


Figure 1 Application Architecture

1. **Presentation Layer**

The Presentation layer is the application's user interaction component, which is responsible for showing info and communicating with the user. This layer contains all user-visible components, such as online sites, forms, buttons, and menus. The Presentation layer gets human input and forwards it to the Logic layer for processing.

2. **Logic Layer**

The Logic layer controls the program's operation. It manipulates data by performing computations and actions on it. This layer includes the principles and algorithms that control the program's behavior. The Logic layer communicates with the Presentation layer to accept human input and show results to the user.

3. **Data Layer**

The Data layer manages data storage and access. This layer communicates with the database to receive and update data, and it is in charge of keeping the data's integrity and ensuring that it is securely kept. The Data layer communicates with the Logic layer to provide data for processing and with the Presentation layer to show data to the viewer.

The three levels collaborate to deliver a comprehensive software solution. The Presentation layer handles the application's user interface, the Logic layer manages the application's functionality, and the Data layer oversees data storage and retrieval. Separating these levels provides better application maintenance and scalability because each layer may be upgraded or replaced independently of the others.

2.3 PHP and MySQL Connect

PHP (Hypertext Preprocessor) is a popular server-side scripting language for computer creation. PHP is referred to as "native" because it is a computer language that is compiled and performed by the server rather than operating in the user's web browser (Williams, 2004).

Native PHP refers to the PHP language's fundamental features, which are included in the standard PHP release and do not require extra tools or modules. Native PHP has many built-in methods and features for dealing with arrays, characters, variables, and other data. It also supports several online networks, such as HTTP and FTP, and databases like MySQL.

MySQL Connect is the process of connecting a client to a MySQL database. A client might be a human, a program, or a device that needs to communicate with a MySQL database. To connect to a MySQL database, the user needs to input these data:

1. Hostname or IP address: This is the network address of the MySQL server to which the user connects.

2. Port: This is the network port that the MySQL server is listening on. MySQL's default port number is 3306.
3. Username and Password: These are the MySQL account credentials with access to the database to which the user connects.
4. Database name: This is the database name to which the user connects.

After user input from the data above, The user may connect to the database using a MySQL client or a programming language library like PHP, Python, or Java.

When a user connects to a MySQL database, the user may conduct various activities, including creating tables, adding data, querying data, and updating it. The user may also utilize the connection to manage the database's security, such as adding or deleting users and giving or canceling their rights.

It should be noted that connecting to a MySQL database necessitates adequate configuration and security procedures to protect the data contained in the database.

2.4 Bootstrap 5

Bootstrap 5 is a free, open-source front-end development framework that allows developers to create responsive, mobile-first websites and web apps. It was created by Twitter and is currently maintained by a developer community (Foreman, 2015).

Bootstrap 5 comes with pre-built HTML, CSS, and JavaScript components, such as forms, buttons, navigation menus, and models, that can be quickly altered and incorporated into a website or application. The framework also offers a responsive grid system, which allows developers to design flexible and adaptive layouts that look excellent on all screen sizes, from desktop to mobile.

Several components are used in bootstrap; some of the following components are:

1. **Tables**

Bootstrap includes a set of classes for creating and styling HTML tables. These classes make it simple to add basic table features to any table in a web page, such as striped rows, hover effects, and responsive behavior. Bootstrap tables can be categorize as follows:

a. Basic Table

This will result in a simple table with no additional design.

b. Striped Table

This will add stripes to the table's alternating rows, making it easier to read.

c. Hover Table

This will give the rows in the table a hover effect, making it simpler to tell which row is being interacted with.

d. Responsive Table

This makes the table responsive, which means it will automatically alter its size and layout to meet the screen size of the device on which it is being viewed.

2. Forms

Bootstrap includes a set of classes for creating and styling HTML forms. These classes enable it simple to add basic form functionality to any form on a web page, such as input validation, style, and layout.

Bootstrap forms class can be categorized as follows:

e. Vertical Form

The vertical form is the basic or default form of the bootstrap. The vertical form indicates that the labels are next to the input field (vertical) on a large and medium screen.

f. Inline Form

All components of an inline form are inline, and left-aligned, and the labels are alongside.

g. Horizontal Form

The horizontal form indicates that the labels are next to the input field (horizontal) on a large and medium screen.

h. Validation Form

This will result in the creation of a form with input validation. The invalid-feedback class will display an error message if the user inputs incorrect input.

CHAPTER III SYSTEM ANALYSIS

This chapter will go through the details of the current Apikko Inc system and analyze other similar systems. The discussion in this chapter will center on studying the current situation and analyzing a similar system. The last part will discuss an analysis of the problem and the system requirement, which will be implemented later in the program.

3.1 Analysis of The Current System

APIKKO Corp was founded in 2006 in Surabaya and received a KUPU operator license from the Bank Indonesia Surabaya Branch on January 18, 2012. Apikko was founded as a supplier of money transfer services from Hong Kong to Indonesia. Apikko is linked to over 20 Indonesian banks and PT Pos Indonesia, providing customers with simple and quick money transfer options. Apikko has over seven years of experience, is backed by a committed staff, is recognized with international security certifications, and is overseen by Bank Indonesia. Apikko admin had around 40 employees working in its company.

Currently, APIKKO Inc's system mostly lacks in its administration. The design sometimes needs to be clarified for its administrator because some menu options have the same icon design. The corporation requested an improvement in the design and additional supervisor options on the menu. The supervisor menu is intended for the supervisor to supervise the branch and cashier SQL tables.

Apikko Inc depended too much on using spreadsheets which can be hard to manage. Based on the current observation, Apikko Inc may experience a number of problems with spreadsheets. First, since spreadsheets frequently lack full data visualization tools, it might be challenging to see patterns or trends in the data. Second, Spreadsheets require more advanced automation capabilities, which could lead to time-consuming manual processes and increase the likelihood of errors. As a

result, even if spreadsheets can be a useful tool for data analysis, there might be more appropriate solutions for tasks requiring sophisticated functionality and automation or challenging datasets.

3.2 Analysis of Similar System

This stage contains an analysis of similar pieces of literature. The research results at this stage will be used in system requirements. The application are chosen based on its similarity to the standard administration system. The results of the literature analysis are as follows:

3.2.1 OVO App

OVO is an Indonesian digital banking application. OVO allows customers to do various activities, including paying bills, purchasing plane tickets, obtaining credit, and transferring money to other OVO members. OVO also provides its customers with a variety of appealing promotions and rebates. The feature of OVO that will be explained is as explained below.

- **Homepage**

  The Homepage format begins with showing the balance and transaction of the top page. Many buttons below the balance section then follow it. The page has five main buttons: home, finance, pay, inbox, and profile.

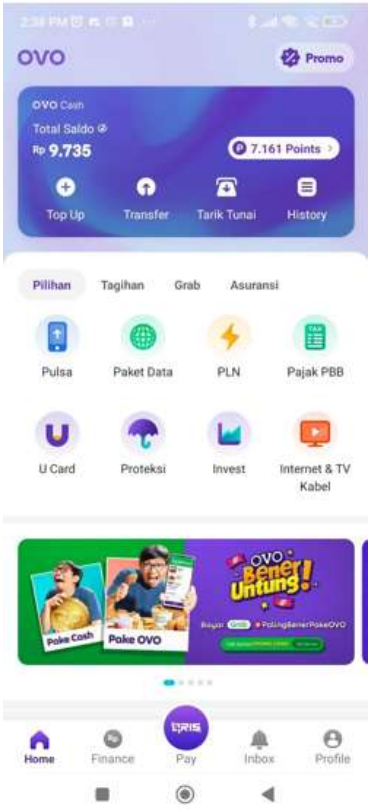


Figure 3.1 OVO Homepage

- **Messages Page**

The page function delivers a message to the users from the admin. There were two choices on the message page: notice and message. The admin can send messages to the user using the notification or message menu.

Figure 3.2 OVO Messages Page

- **Transfer Page**

The transfer page's primary function is to transfer money to the bank or a person. The transfer page has several options; users can transfer money through phone numbers or to the bank. After users transfer, they will be added to the final transaction history list, and users can also add to favorites from their transactions.

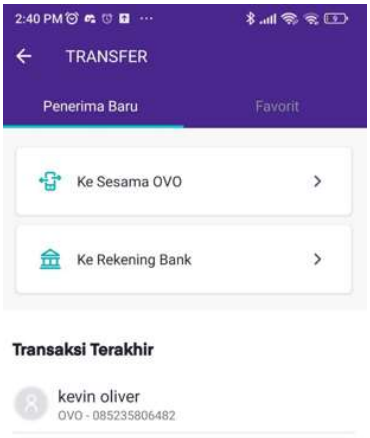


Figure 3.3 OVO Transfer Page

When the user chooses to transfer money through a phone number, the user needs to fill in the phone number of the direct transfer and the amount of money the user is willing to pay. If users transfer money through a bank then user need to choose the list of banks in the combo box and the account number. The user must also choose the payment method to transfer to the bank and the transfer amount. After filling in the form, the user needs to click continue to transfer the money.

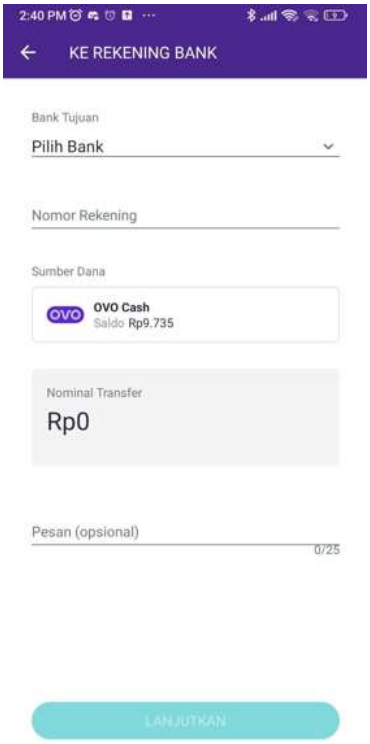


Figure 3.4 OVO Transfer Destination

- **History Page**

The history page shows the list of the user's transactions. The history will show the transaction date, the transaction destination, the method of payment, and the total payments. The user can also get the detail of the transaction by selecting the section of the transaction.

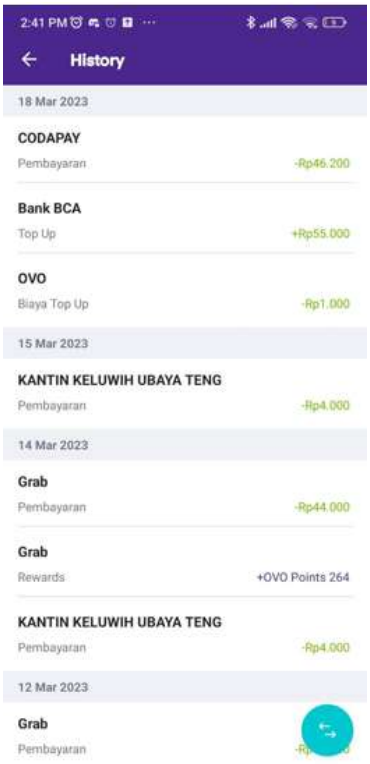


Figure 3.5 OVO History Page

3.2.2 Mandiri App

Livin' by Mandiri is a digital financial app provided by Bank Mandiri, one of Indonesia's leading banks. This app gives users access to various banking goods and services, such as money transfers, bill payments, credit purchases, and aircraft ticket purchases. Moreover, Livin' by Mandiri provides other options such as investing, insurance, and online shopping. The feature of Livin' by Mandiri that will be explained is as explained below:

- **Homepage**

  Many buttons show the screen's main page on the initial page, and the user balance on the page is put at the bottom of the buttons page. The page has five main buttons: home, promo, message, setting, and log out.

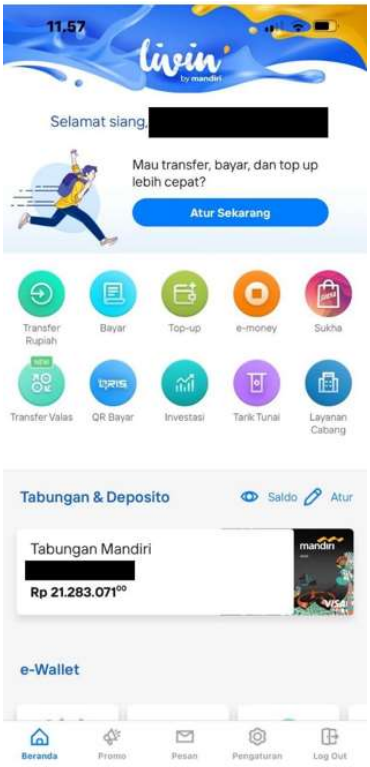


Figure 3.6 Mandiri Homepage

On the right side of the main button there a button to log out. The button logs out when pressed will deliver a pop-up message that is used to verify if the users logged out.

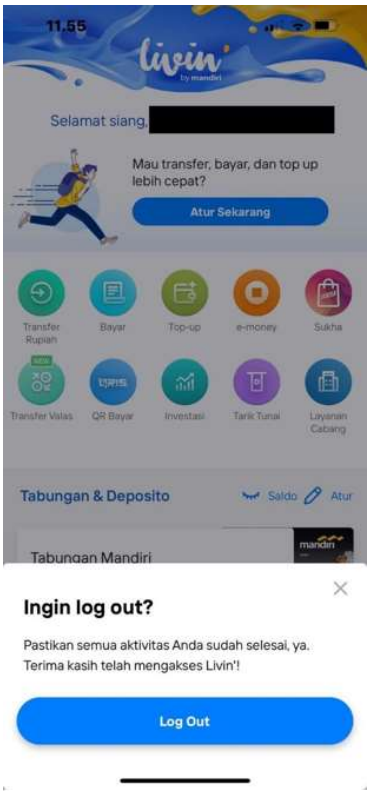


Figure 3.7 Mandiri Menu

- **Foreign Exchange Transfer Page**

The user uses the foreign transfer exchange page to exchange currency from other countries for the Indonesian rupiah. The user needs to select the country in the accepted nominal textbox on the page. After that, the user needs to input the nominal in the box. When the user inputs the nominal, the total transaction in rupiah is automatically shown to the user.

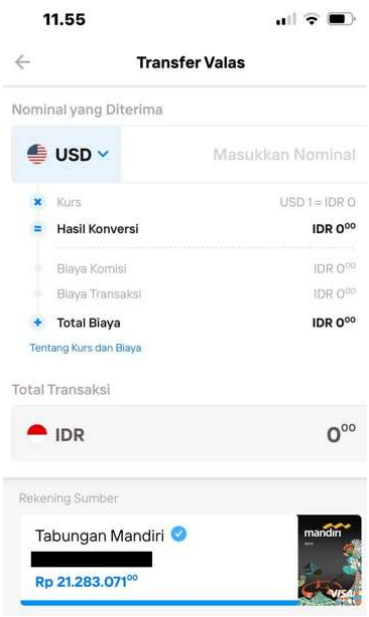


Figure 3.8 Mandiri Foreign Exchange Transfer Page

- **Message Page**

The message page had two menus, notification and status transaction. The notification menu is for the admin to send messages to the user. The status transaction is to show the history of the user's transaction. The status transaction will show if the user transaction is failure, pending, or successful. The user can also filter the status message by clicking the filter button. The status shows the transaction date, the sender, the sender id, and the user payment.

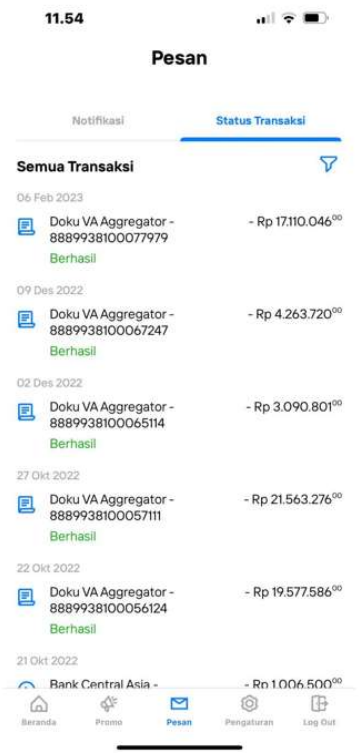


Figure 3.8 Mandiri Message Page

- **Transfer Page**

The transfer page has a basic design; the page only has one combo box, textbox, and button below the page. The page function is to transfer money to the bank destination. To use the page, the user first needed to select the bank that it plans to transfer and the bank's account number.

Figure 3.9 Mandiri Transfer Page

- **Payment Page**

The Payment page shows the user list of payments that the user wants to choose. After the user selects one of the payment options, the user will be directed to the page where the user needs to choose the payment destination. After that, the user must fill in the virtual account number and the payment amount.

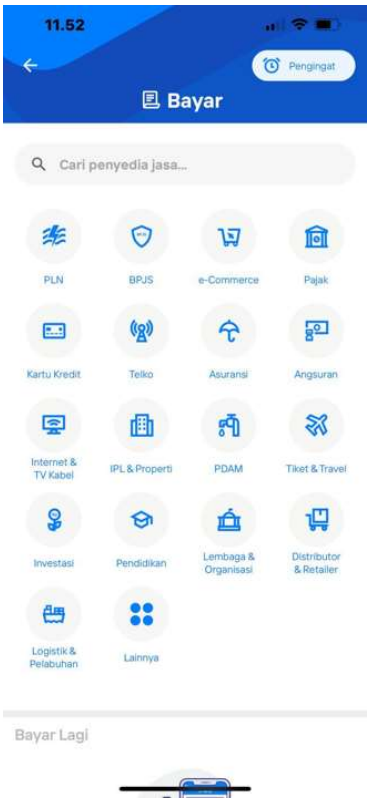


Figure 3.10 Mandiri Payment Page

3.2.3 Klik BCA website

KlikBCA is an internet banking service offered by Bank Central Asia (BCA), one of Indonesia's leading banks. This service enables BCA clients to conduct different financial operations online, including money transfers, bill payments, credit purchases, and aircraft ticket purchases. Apart from that, KlikBCA provides additional capabilities such as creating new accounts, purchasing banking goods, and applying for credit. KlikBCA may be accessed via the BCA official website or the KlikBCA mobile banking application, available on the Google Play Store or the App Store. The feature of Klik BCA that will be explained is as explained below:

- **Homepage**

The home page of the screen is started by welcoming the user name to the screen. In the above of welcome sentence, the user is shown the last time the user login into the form of Datetime. The menu of the page is placed on the left of the screen. The user can logout from the page by clicking the log out button on the top left corner.

Figure 3.11 Klik BCA Homepage

- **Money Transfer Page**

The page for transferring money is on the page of a form. The page design is simple without any complicated design. In the "Input 8 number" section, the user is warned for inputting the user to give the correct input format. The warning sign is given in bold and red color font. After the user finishes to fill the form, the user will be transferred to the next page; if not, the user will be warned to fill the form in.

Figure 3.12 Klik BCA Money Transfer Page

- **Status Page**

The status page shows whether the user has opened their account number. When the user hasn't opened their account, it will be given a warning if they haven't opened their account numbers.

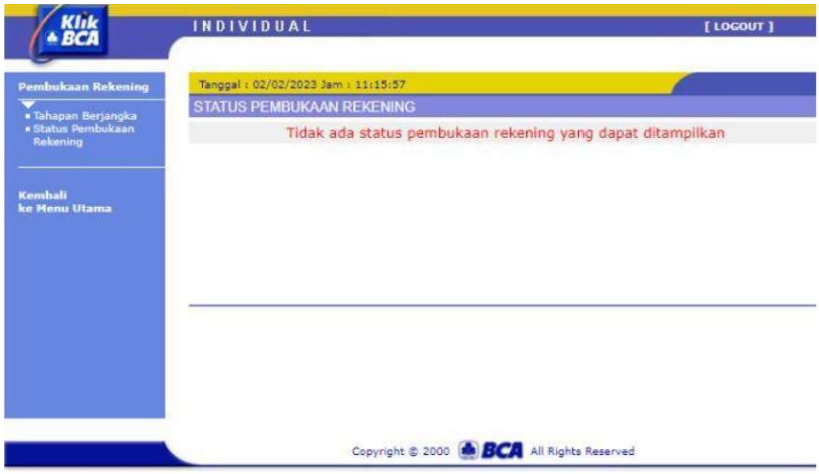


Figure 3.13 Klik BCA Status Page

- **Credit Card Transaction Page**

The credit card transaction page checks the user history of transactions of credit cards. If there's no transaction, the form will be replaced with the warning that the user did not have any transaction. The warning will be shown in red text.

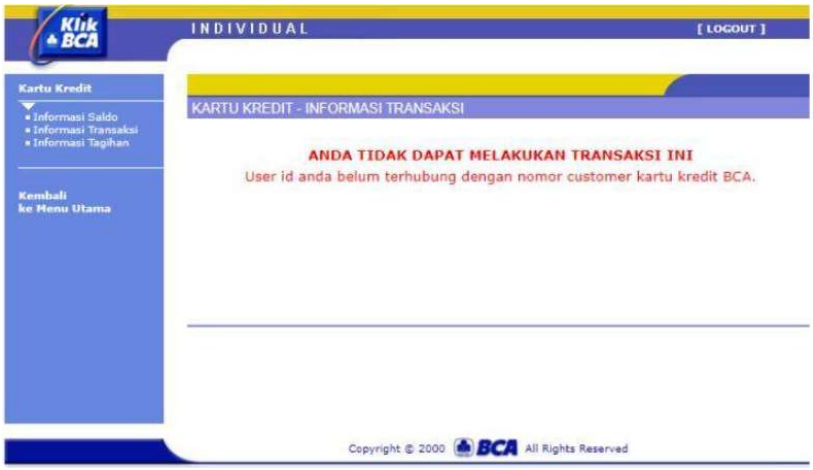


Figure 3.14 Klik BCA Credit Card Transaction Page

- **Account Information Page**

The account information page gives users information about their balance, account mutation, deposit, future stage, and fund account. Users who did not apply for one of the account information. The user will be blocked from information until the user apply it. The warning on the page is the same with the red color font.

Figure 3.15 Klik BCA Account Information Page

- **Failure Transaction Page**

The user transaction can fail if there's no transaction happening. The user can only see the page after the user did the transaction. The warning will be shown with the red font bold text.

Figure 3.16 Klik BCA Failure Transaction Page

- **Transaction Information**

Information transaction is used to find transactions of the users. From the form, we can see that the user can search type of transaction, the status transaction, and the transaction date. After the user fills out the form, the user will be shown a list of the user transaction.

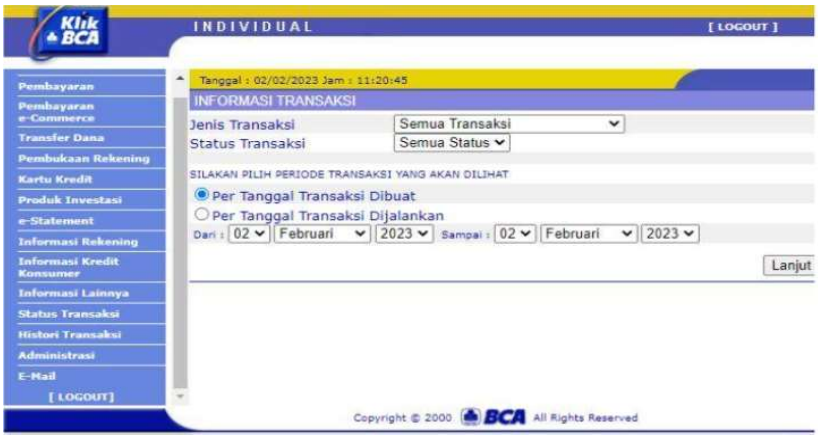


Figure 3.17 Klik BCA Information Transaction

If there's no transactions of the users, the users will be shown that there's no users transaction like in figure 3.18

Figure 3.18 Klik BCA Missing Transaction History

- **Administration Page**

  The administration page is used to edit user transaction, information, and language. The user also deletes a transaction, as shown in figure 9. If the users do not delete a transaction, it will show "List Empty" in red bold font.

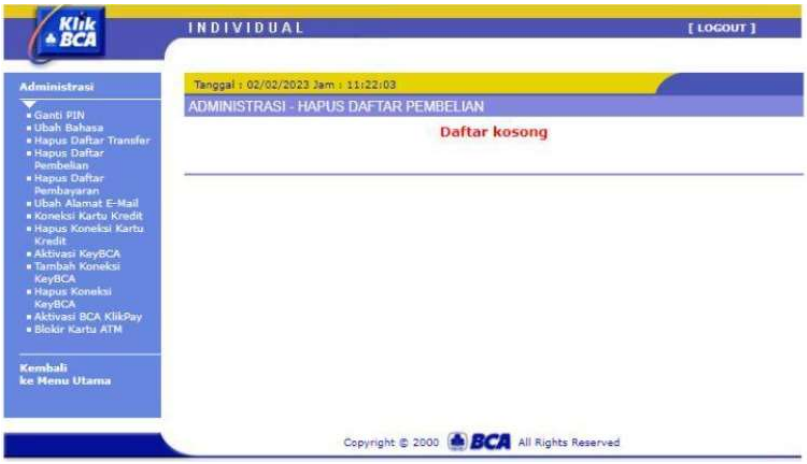


Figure 3.19 Klik BCA Administration

3.3 Analysis of The Problem

This section will discuss the issues arising from the previous system analysis. Based on the findings of the problem-solving analysis, namely:

- **Security Risk**

  When sensitive or private data is involved, the spreadsheet is usually kept locally on a computer, which creates security issues. Also, the system is vulnerable to viruses and malware, which might compromise the integrity of the data. Spreadsheets do not have any encryption features either. As a result, if the file is compromised, the saved data is susceptible to being viewed or taken. Furthermore, even though the spreadsheet has password security, it is simple for someone viewing the file to go pass it.

- **Data Integrity Error**

  Inaccuracies and inconsistencies in the spreadsheet are common, mainly when several users are engaged. Compared to more capable data management tools, the spreadsheet is more prone to data input errors, erroneous computations, and formatting issues. Human errors can lead to inaccurate data that is challenging to identify and correct, and these errors can occur during manual data entry, calculations, and formulas. Furthermore, the spreadsheet makes it easy to create duplicate data, which might result in errors and inconsistencies. By accidentally copying and pasting data many times, users, for instance, might create a large amount of the same content.

- **Limitation of Data Reporting**

  The major problem with creating custom spreadsheet reports is that they take a long time and are prone to mistakes. Although the spreadsheet has reporting tools, creating customized reports is a significant labor-intensive task, such as data extraction, formatting,

and analysis. The spreadsheet must be revised to create complex reports using various data sources and formats. The spreadsheet can be difficult to manage and maintain as the report's complexity and data volume increase, leading to errors and inconsistencies.

- **Limitation Automation**

  Simple chores can be automated using a spreadsheet, but more complex jobs may benefit from using anything else. Most users need to gain coding skills to automate sophisticated spreadsheet operations. Finding flaws and issues in automated processes may be challenging due to spreadsheets' limited capabilities for error detection. Inaccurate findings that are challenging to uncover and correct might result from the restricted error-checking capability. The spreadsheet is designed to function with ordered data but needs to be more flexible when working with unstructured data or intricate data linkages. Automating complex data processing operations, such as data cleansing, merging, or transformation, can be difficult.

- **Limitation Collaboration**

  Working together on the same spreadsheet might be time-consuming because a spreadsheet is designed for use by a single person. Inconsistent data may occur from conflicts between concurrent users attempting to view or amend the same sheet. Moreover, the spreadsheet lacks granular access restrictions, making monitoring who may access the file and make modifications difficult. This can be a security risk, especially when sensitive data is involved. A single individual may work on the spreadsheet at a time because it does not permit concurrent editing. There may be delays and bottlenecks if many people need to work on the same file at once

3.4 Analysis of System Requirement

From the Analysis of the system requirement, better DBMS management is needed for a more advanced approach. Based on the results of several analyzes above, the system requirements needed are as follows:

- **Security Solution**

  The new system must incorporate security features such as user authentication and access restrictions to protect sensitive data from unauthorized access or alteration. Data security measures are required in the new system to help safeguard the privacy of sensitive data, including trade secrets and personal data. The new technology assures the data is private and safe from unauthorized access.

- **Force Data Integrity solution**

  The new system should offer a user-friendly graphical user interface that enables data entry without the need for programming knowledge. The system can now more effectively enter data by non-programmers. The new system eliminates data duplication by ensuring data is stored just once and in the correct format. Inconsistent data is less likely with the new approach, which can also help with storage space reduction.

- **Custom Reports solution**

  Users of the new system should be able to create custom reports without programming knowledge, thanks to reporting features. It is simple for non-programmers to build the necessary reports using the reporting tools. Thanks to access to custom reports, users can create reports specific to their needs. The report's structure, desired level of detail, and material to include are all selectable by users. Creating direct custom reports might assist in standardizing reporting throughout a company. Comparing data from different

departments will be made more accessible by ensuring that reports adhere to a standard format.

- **Easy Data Entry Solution**

  The new system has to include an intuitive graphical user interface that enables data entry without programming knowledge, which will help it overcome its automation limitations. Now, data entry by non-programmers can be done more quickly. Users must be able to save time by rapidly and accurately inputting data into the new system owing to simple data entry. This helps when entering substantial amounts of data. When data entry is easy, users are more likely to offer correct information. The new approach could improve the data's general quality.

- **Data Sharing Solution**

  The new system should provide smooth data exchange across apps and users to address the cooperation issue. This might facilitate team collaboration and data sharing among users. Duplication of effort and data may be eliminated by data sharing, which will save time and money. Productivity may be increased by allowing several users to access and work with the same data simultaneously. The data is kept consistent and accurate because changes one makes are instantly apparent to others.

CHAPTER IV SYSTEM DESIGN

This chapter will discuss system design results based on the results requirements. The system designs that will be explained include Database Design, Process Design, and User Interface Design.

4.1 Database Design

This database architecture is broken into the Entity Relationship Diagram (ERD) and Data Dictionary sections. The database structure utilized in the system will be shown in the sub-chapter Entity Relationship Diagram. In the meantime, the data dictionary subchapter will explain the contents of the completed ERD.

4.1.1 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) is a high-level representation of data modeling for constructing a system. Each table has a relationship, and it serves a particular purpose. Some are used to store user data, while the administrator may use others to edit and delete data. There is also a table that keeps data transactions. The ERD design for the system is illustrated in Figure 4.1.

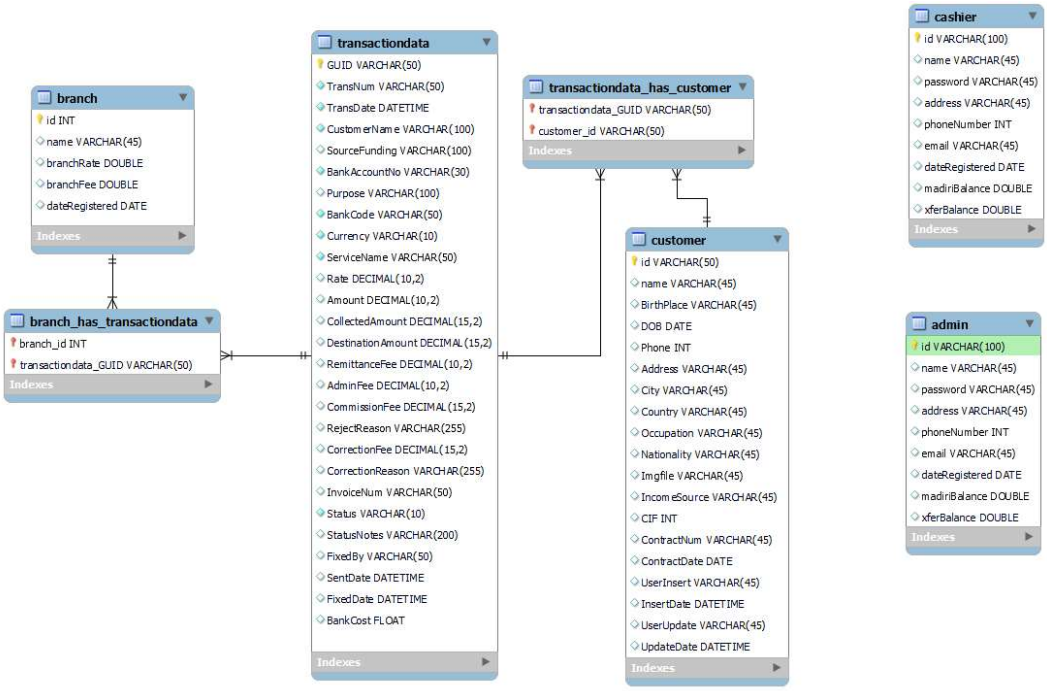


Figure 4.1 ERD Diagram

Beginning with the transaction table it contains many-to-many relationships with the branch and the customer Table. A user can be in the cashier or admins table, and the cashier and admin table can edit the transaction, customer, and branch table. Moreover, it is also used for users to log in or register into the admin interface. The main difference is that only the admin table can edit the cashier, while the cashier cannot edit the admin table.

4.1.2 Data Dictionary

Each table in the ERD will be discussed in this data dictionary sub-chapter. This section will review each table's attributes, data types, and explanations. The following is an explanation of the ERD tables.

- **Branch Table**

Information on branch data is kept in a table called the branch table. As indicated in Table 4.1, the information in this branch table is categorized by data types and descriptions. Branch Table had many-to-many relationships with the transaction table.

Table 4.1 Branch Table

Data Name	Data Type	description
id	INT	Primary Key, AUTO_INCREMENT . NOT NULL.
name	VARCHAR(50)	The field name of the branch. NOT NULL.
Branch Rate	double	The rate of the branch. It can be Null.
Branch Fee	double	The fee of the branch. It can be Null.
Date Registered	date	The first registration to the branch table. NOT NULL.

- **Admin Table**

Admin information is kept in a table called the admin table, and the table serves as the system login page. As demonstrated in Table 4.2, the information in this admin table is categorized into several data types and descriptions. The admin table is necessary for user login. This table is important for matching usernames and passwords so that unauthorized users do not gain access to the program.

Table 4.2 Admin Table

Data Name	Data Type	description

id	INT	Primary Key, AUTO_INCREMENT . NOT NULL.
Password	VARCHAR(255)	The field name secures the cashier password. The password is encrypted. NOT NULL.
Name	VARCHAR(100)	The field name that stores the unique name of the cashier. NOT NULL.
Address	VARCHAR(200)	The field name that stores the address of the cashier. It can be Null.
Phone Number	Int(200)	Field name that represents the cashier phone number. It can be Null.
Email	VARCHAR(200)	The field name that stores the cashier's email. It can be Null.

- **Cashier Table**

The cashier table is a table used to keep track of customer data. The table serves as the system login page. As indicated in Table 4.3, the information in this user table is categorized into several data categories and descriptions. The cashier table is used for the admin can log in to the cashier interface. The cashier table is required for user login. For the purpose of preventing

unauthorized users from accessing the application, this table is crucial for matching usernames and passwords.

Table 4.3 Cashier Table

Data Name	Data Type	description
id	INT	Primary Key, AUTO_INCREMENT . NOT NULL.
Password	VARCHAR(255)	The field name secures the cashier password. The password is encrypted. NOT NULL.
Name	VARCHAR(100)	The field name that stores the unique name of the cashier. NOT NULL.
Address	VARCHAR(200)	The field name that stores the address of the cashier. It can be Null.
PhoneNumber	Int(200)	Field name that represents the cashier phone number. It can be Null.
Email	VARCHAR(200)	The field name that stores the cashier's email. It can be Null.

BranchId	INT	Foreign key from Branch Table

- **Customer Table**

Customer information is kept on the customer table. According to Table 4.4, the information in this customer table is categorized by data types and descriptions. The Customer Table had many-to-many relationships with the transaction table.

Table 4.4 Customer Table

Data Name	Data Type	description
ID	VARCHAR(50)	Primary Key, AUTO_INCREMENT . NOT NULL.
Name	VARCHAR(100)	The field name contains the unique Customer name. NOT NULL.
Birthplace	Varchar(100)	The Birthplace of the Customer. It can be Null.
DOB	INT	The date of birth of the customer. NOT NULL.
Phone	Varchar(20)	It contains the phone number of the customer. It can be Null.
Address	Varchar(200)	The field name stores the address of the customer. It can be Null.

City	Varchar(50)	The field name stores the city address of the customer. It can be Null.
Country	Varchar(50)	The field name stores the country address of the customer. It can be Null.
Occupation	Varchar(50)	The field name stores the job occupation of the customer. NOT NULL.
Nationality	Varchar(50)	The field name contains the customer's nationality. NOT NULL
imgfile	Varchar(255)	The field address of the customer images. It can be Null.
Income Source	Varchar(50)	The field source income of the customers. Not Null. It can be Null.
Contract Num	Varchar(50)	The contract code of the customers. It can be Null.
Contract Date	date	The date of the customer contract. NOT NULL.
User Insert	Varchar(50)	The name of the user who inserted it. It can be Null.
Insert Date	datetime	The insert date of the first time the customers inserted. It

		can be Null.
UserUpdate	Varchar(50)	The user who update the customer's column. It can be Null.
Update Date	datetime	The date of the customer's update. It can be Null.

- **Transaction Table**

The transaction table stores transaction data. Table 4.4 categorizes the information in the transaction table by data types and descriptions. The Transaction Table had many-to-many relationships with the Customer and Branch table.

Table 4.5 Transaction Table

Data Name	**Data Type**	**description**
GUID	VARCHAR(50)	Primary Key, AUTO_INCREMENT . NOT NULL.
TransNum	VARCHAR(50)	The field name contains the unique transaction number. NOT NULL.
BranchID	Varchar(10)	The field name store unique branch ID. NOT NULL.
TransDate	datetime	The date of the transaction. NOT NULL.
CustomerID	Varchar(50)	The field name store unique Customer ID. NOT NULL.

CustomerName	Varchar(100)	The field name stores the customer Name. Can be Null.
SourceFunding	Varchar(100)	The field name stores The source of the fund. NOT NULL.
BankAccountNo	Varchar(30)	The field name stores the customer bank numbers. NOT NULL.
Purpose	Varchar(100)	The field name stores the purpose of the transaction. It can be Null.
BankCode	Varchar(50)	The field name stores The code of the bank. NOT NULL.
Currency	Varchar(10)	The field name contains the types of currency being used. NOT NULL
ServiceName	Varchar(50)	The field name contain the name of the service. It can be Null.
Rate	Decimal(10,2)	It stores the rate of the transaction. It can be Null.
Amount	Decimal(10,2)	It stores the amount of the transaction. It can be Null.
CollectedAmount	Decimal(15,2)	It stores the collected amount of the transaction. It can be Null.

DestinationAmount	Decimal(15,2)	It stores the destination amount of the transaction. It can be Null.
RemittanceFee	Decimal(10,2)	It stores the remittance fee of the transaction. It can be Null.
AdminFee	Decimal(10,2)	It stores the admin fee of the transaction. It can be Null.
CommissionFee	Decimal(15,2)	It stores the commission fee of the transaction. It can be Null.
RejectReason	Varchar(255)	It stores the reason for the rejection of the transaction. It can be Null.
CorrectionFee	Decumal(15,2)	It stores the correction fee of the transaction. It can be Null.
CorrectionReason	Varchar(255)	It stores the reason for the correction being done. It can be Null.
InvoiceNum	Varchar(50)	It stores the Invoice Num of the transaction.
Status	Varchar(10)	It stores the status of the transaction. There are two types of status in the transaction: Pending or sent.
StatusNotes	Varchar(200)	It stores the explanation of the status of the transaction.

FixedBy	Varchar(50)	It stores the person who fixed the transaction. It can be Null.
SentDate	datetime	It stores the date of the sent in the transaction.
FixedDate	datetime	It stores the date of the transaction being fixed.
BankCost	float	It stores the cost of the bank.

4.2 User Interface Design

This User Interface Design will describe the interface's look on the created system. This system's user interface is separated into three sections: a page for administrators, and users.

- **Login Cashier Page**

The login Cashier page is the first page for Cashiers only. On the login screen, the user must select the branch name, and after that the user will be directed to the homepage. The login cashier page design contains a combo box and a sign-in button on the bottom of the combo box, which can be seen in Figure 4.2.

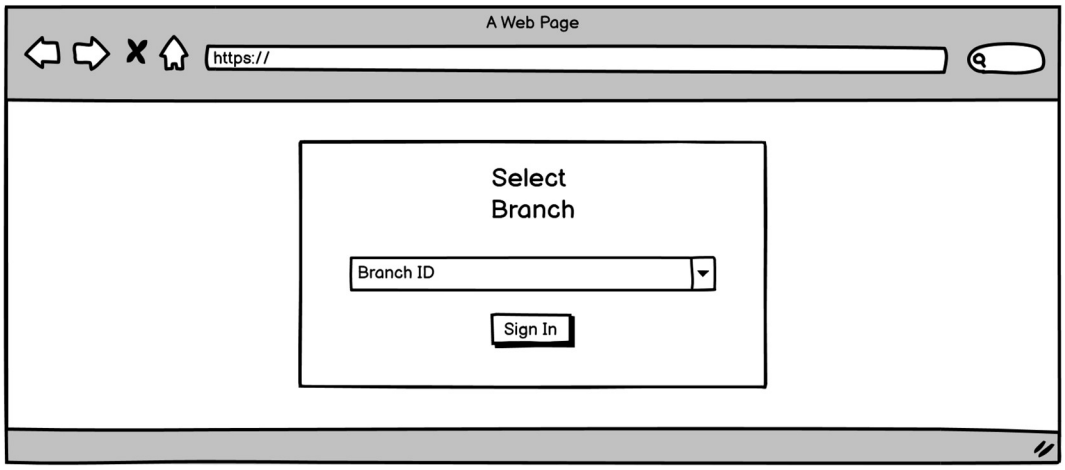


Figure 4.1 Sign-In for Cashier

- **Login admin Page**

The login admin page is the first page for the admin. On the login screen, the user must provide a username and password. On the login screen, two textboxes assist the user in entering the username and password, and submit button is provided, as shown in Figure 4.2. Users must enter the username and password to login to the system. The user must enter the correct username and password for it to be directed to the main page. If the user enters the wrong username and password, the system will display a notice that the username and password are wrong.

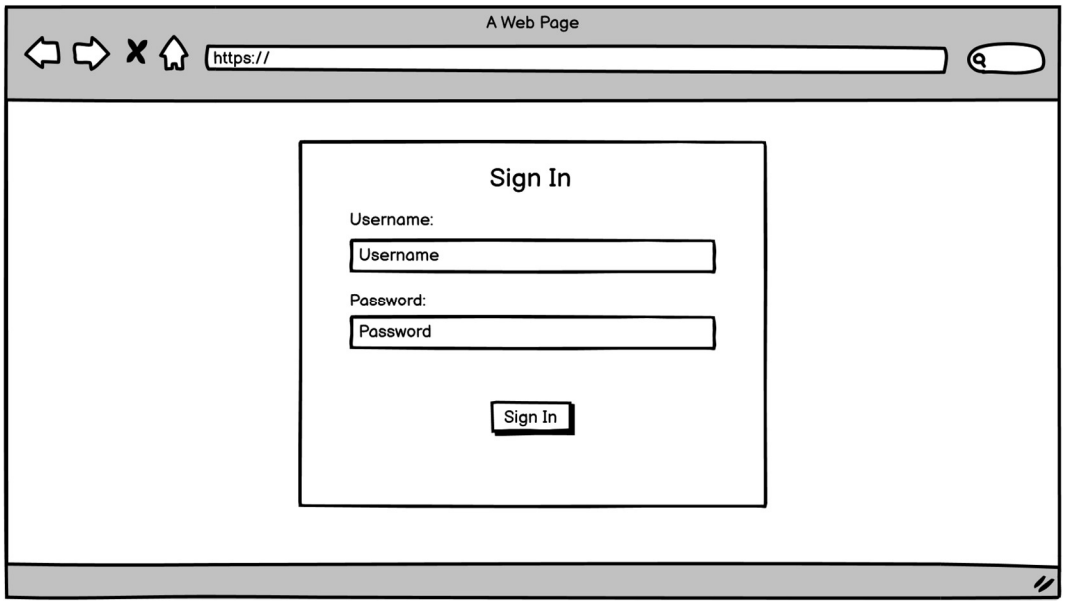


Figure 4.2 Sign-In for Admin

- **Transaction Page**

The transaction page is where the user sees the list of transactions done in the database. Users can search data from the transaction, sort the data, and limit the number of data in the transaction. Users can also edit the data on the transaction page by clicking the edit button. If the transaction date does not exist, the table will be empty. The page illustration can be seen in Figure 4.3.

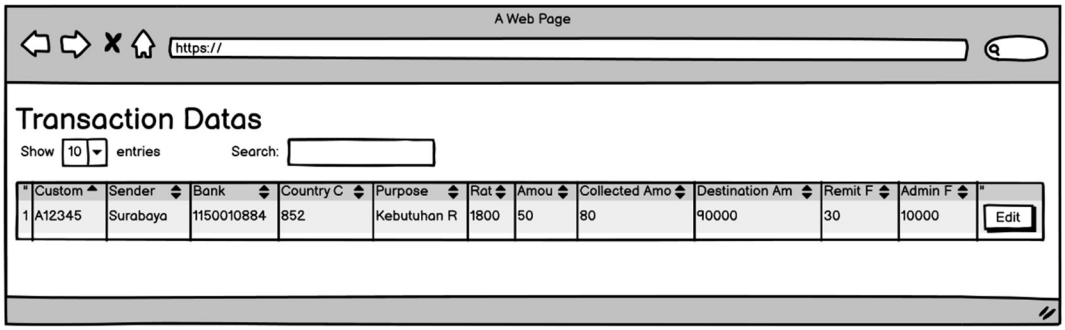


Figure 4.3 Transaction Page

- **Block Customer or Cashier Page**

The block customer or cashier page has the same layout. The cashier can block the customer for the customer not to be able to use the transaction anymore. For admin, they block the cashier access for entering the page. The block page layout consists of two combo boxes for id and two buttons. The customer who is blocked will not be listed in the transaction table. The admin or cashier can also unblock customers or cashiers by entering the id in the unblock box. The main difference is that it blocks the cashier from entering the admin page. The design of the page is illustrated in Figure 4.45

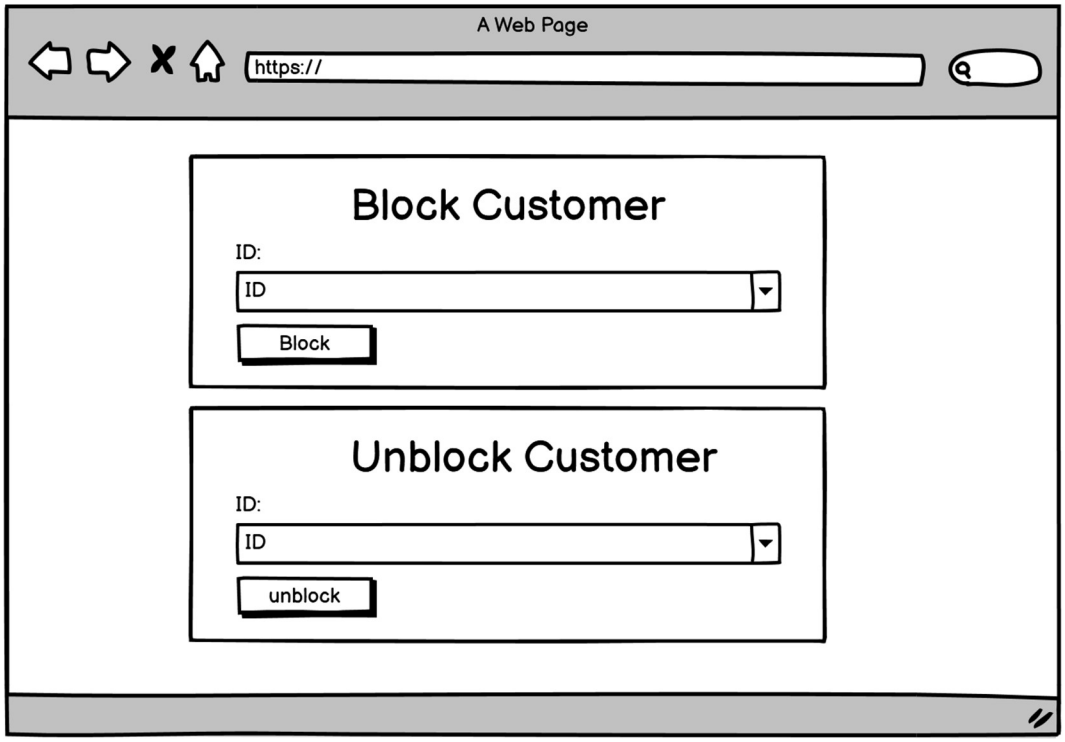


Figure 4.4 Block Customer Page

- **Change Password Cashier Page**

The user can change the password of the admin by inputting the admin id and the admin's old password. The user needs to have the correct old password in order for the user to be able to change the admin password. The user must give the same password with the confirm password textbox to ensure the password is correct. The page consists of four textboxes and one button. The page illustration can be seen in Figure 4.5

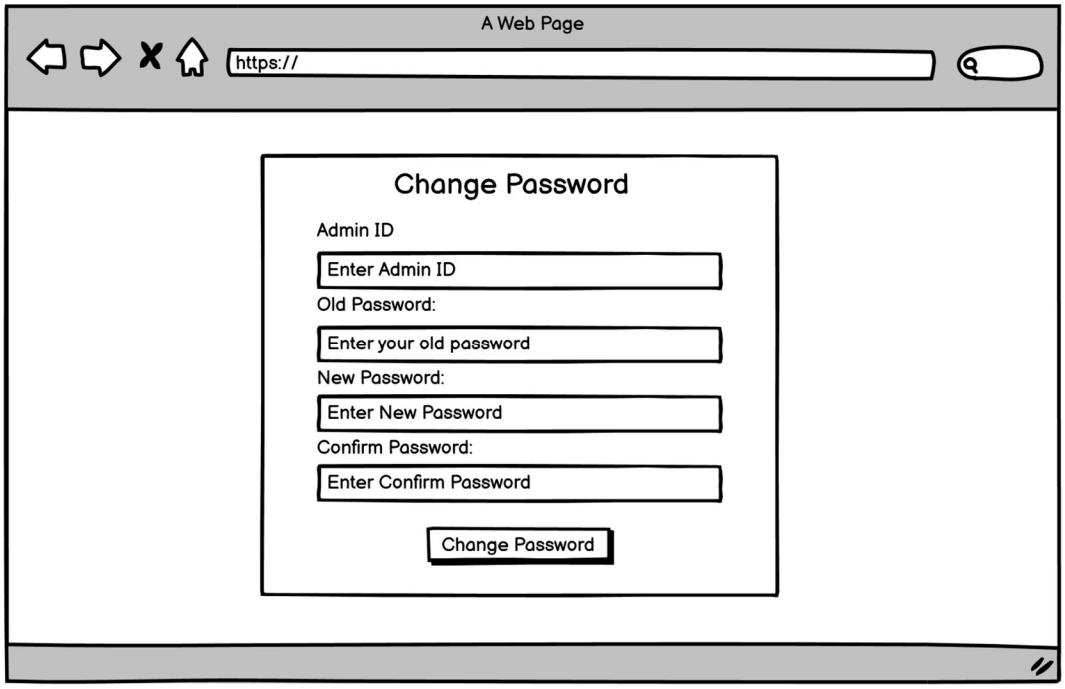


Figure 4.5 Change Password Cashier Page

- **Branch Rate and Fee Page**

Branch fee and rate pages have similar designs to each other. The user can change the branch's rate or fee by selecting the branch name and changing its rate or by inputting it in the textbox. The page consists of a single combo box, textbox, and button for the page. The page illustration can be seen in Figure 4.6.

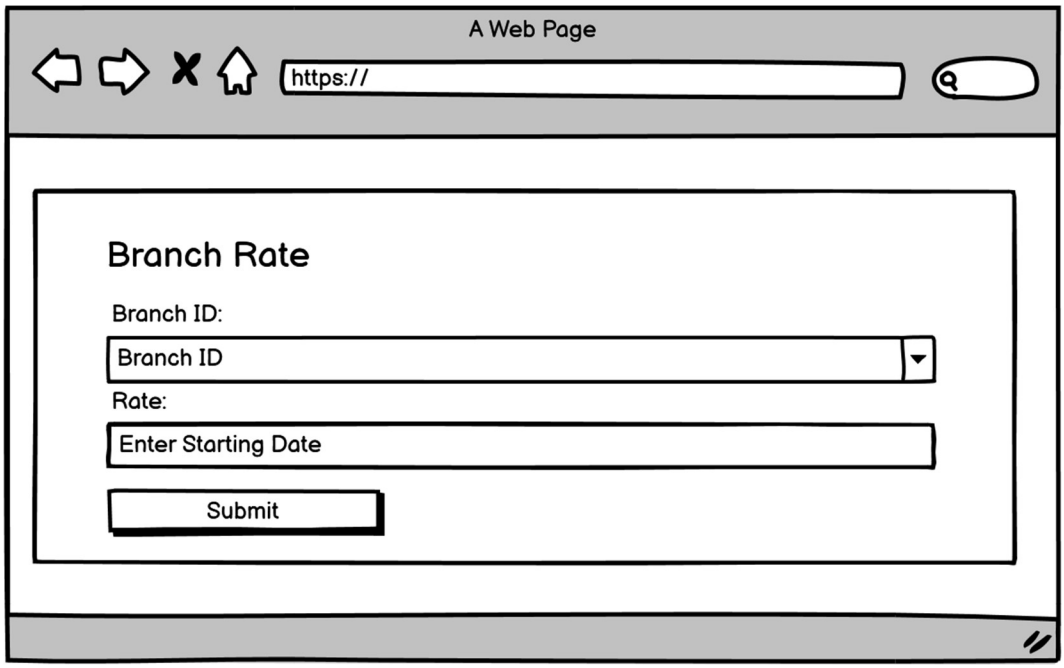


Figure 4.6 Branch Rate Page

- **Check Status Transaction Page**

Check Status Transaction pages is similar to trash page; their primary function is to permanently delete the transaction or undo the delete in the transaction. The page has a similar design to the transaction page and still has the same feature as the transaction page; the main difference is that there is no edit button on the page. The page layout can be seen in Figure 4.7.

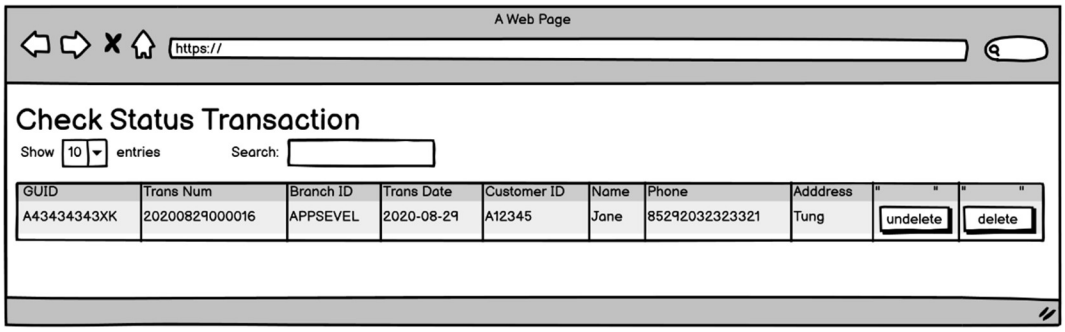


Figure 4.7 Check Transaction Page

- **Add Cashier Page**

The admin enters the new cashier into the database using the add cashier page. The admin must enter the ID, Password, name, address, phone number, and email. The page comprises four textboxes, four combo boxes, and one button—the page's design can be seen in Figure 4.8.

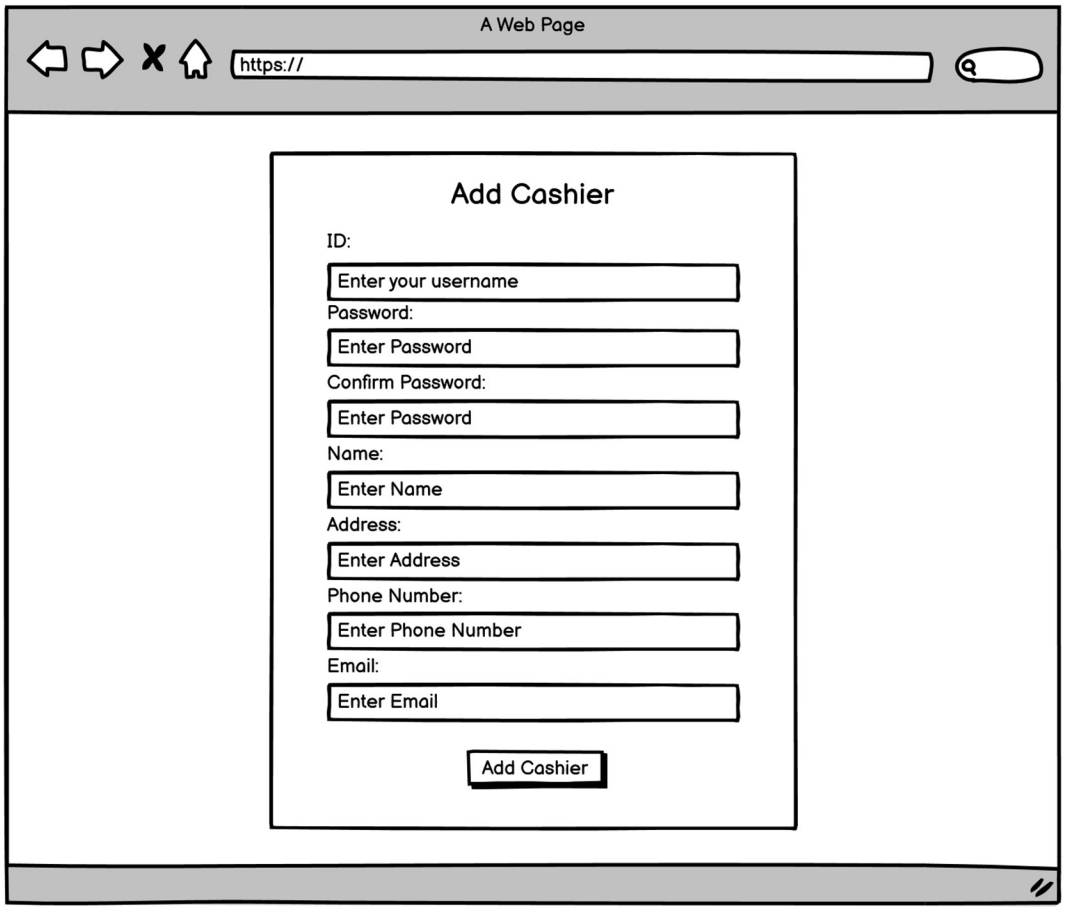


Figure 4.8 Add Cashier Page

- **Cashier Report Page**

The admin only uses the cashier report page to view, update, and remove the report's table. Id, name, address, phone number, email, date registered, and branch name are among the fields in the table. The page still have the feature in the transaction page. Figure 4.9 illustrates the page's layout. The edit page will open a pop-up box when the admin clicks the edit button. The pop-up box will resemble Figure 4.8 in appearance. The report on the page can also be exported to Excel by clicking the export Excel button.

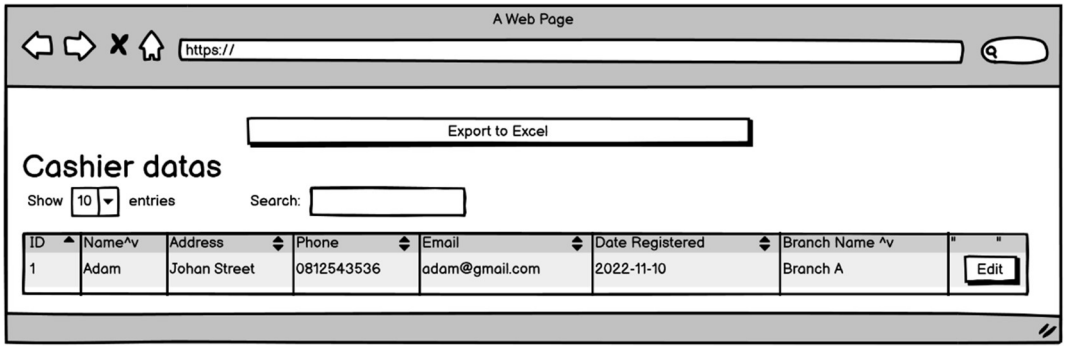


Figure 4.9 Cashier Report Page

- **Add Customer Page**

The admin adds the new customer to the table using the add customer page. Id, Name, BirthPlace, Date of Birth, Phone, Address, Country, Occupation, Nationality, IncomeSource, and ContractDate must all be entered by the administrator. The id in the textbox needed to be unique if not the user will be warn if the id of the customer is not unique. The layout of the page is shown in Figure 4.10.

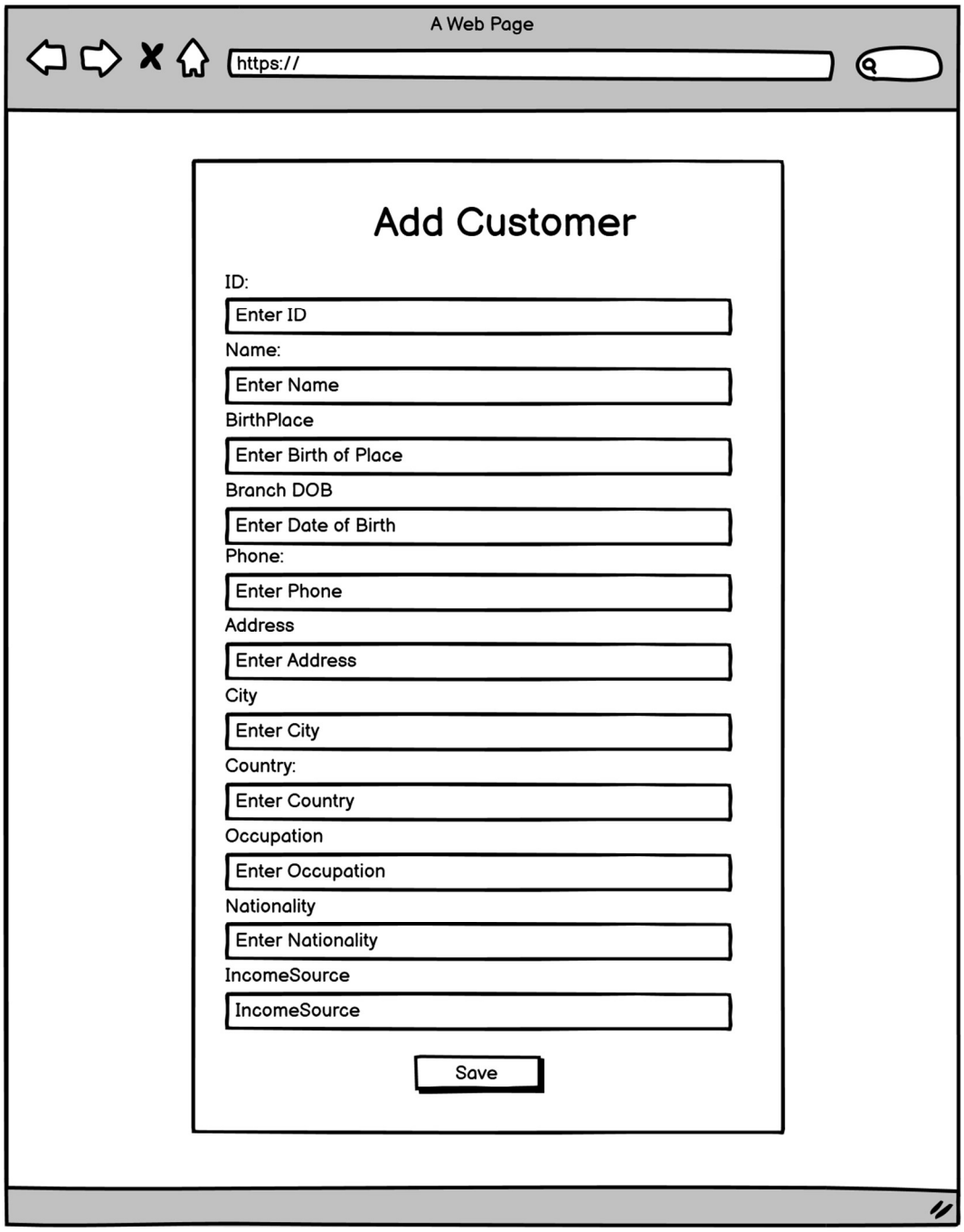


Figure 4.10 Add Customer Page

- **Customer Report Page**

The customer report page is mainly used for cashiers to edit, delete and read the table in the report. The table fields include id, name, birthplace, date of birth, phone, address, country, occupation, nationality, income source, and contract date. The page also has features such as search

data, limit data entries, and sort data. The design of the page is illustrated in Figure 4.11. When the cashier clicks the edit button, a pop-up box of the edit page will appear. The pop-up box will have a similar design to Figure 4.10.

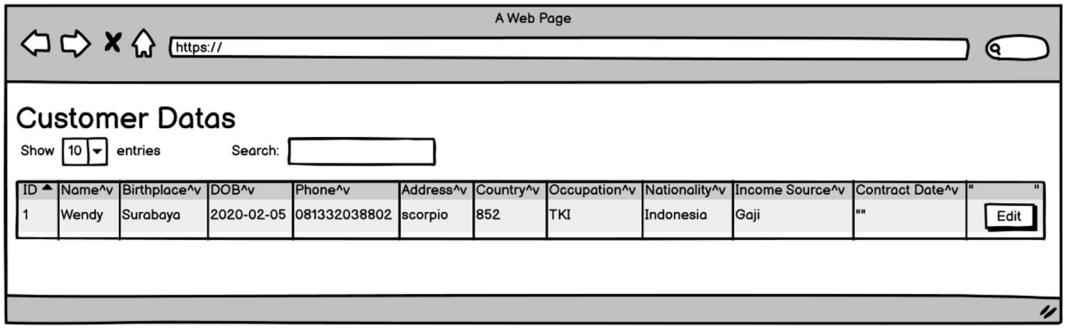


Figure 4.11 Customer Report

- **Add Branch Page**

The administrator uses the add branch page to add the new branch to the database. The administrator must enter the name, branch rate, and fee. There are three text boxes and one button on the page. The page design is illustrated in Figure 4.12.

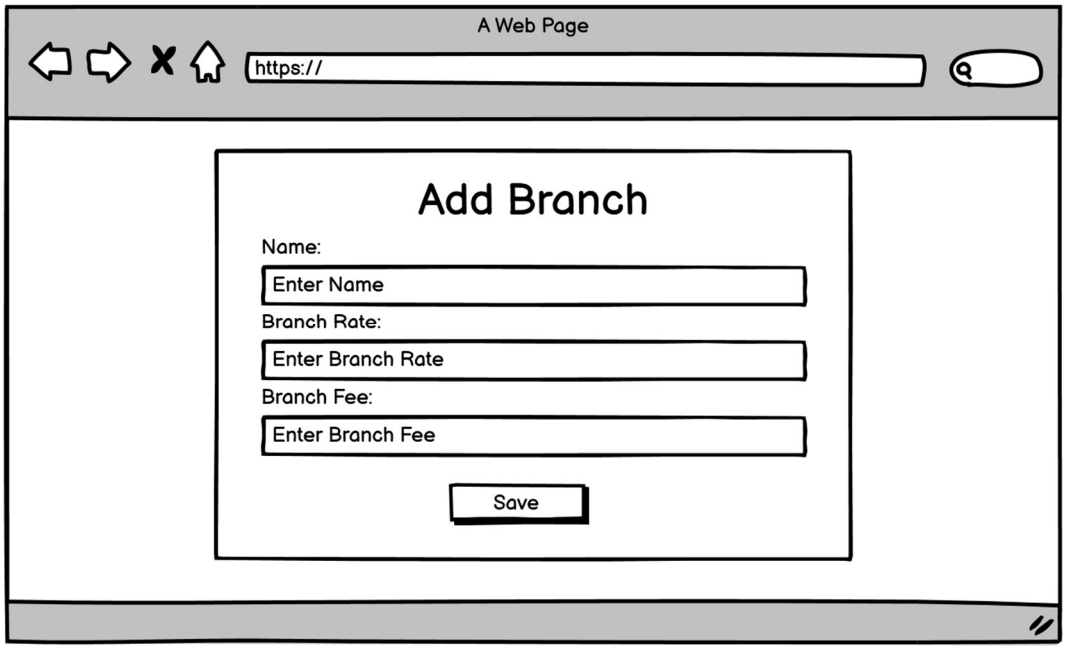


Figure 4.12 Add Branch page

- **Branch Report Page**

The branch report page is mainly used for the admin to edit, delete and read the table in the report. The table fields include id, name, branch rate, branch fee, and date registered. Users can search the rows in the table using the search box and can search and sort the page by clicking the up and down error. The design of the page is illustrated in Figure 4.13. When the admin clicks the edit button, a pop-up box of the edit page will appear. The pop-up box will have a similar design to Figure 4.12. Users can also export the report into Excel by clicking the export to Excel button.

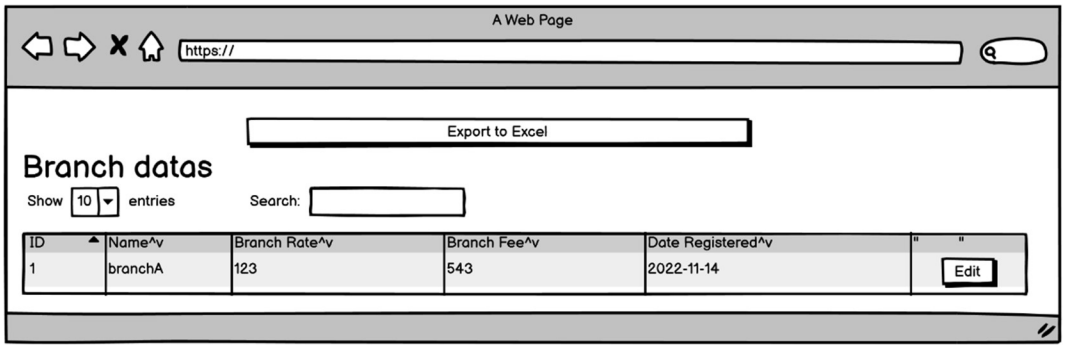


Figure 4.13 Branch Report Page

- **Add Transaction Page**

The cashier uses the add transaction page to add new transactions to the database. The cashier must input the branch id, user ID, Sender Name, Account Number, Currency, sent amount, deposit amount, and amount shipped. The page consists of four textboxes, four combo boxes, and a single button. The page illustration can be seen in Figure 4.14.

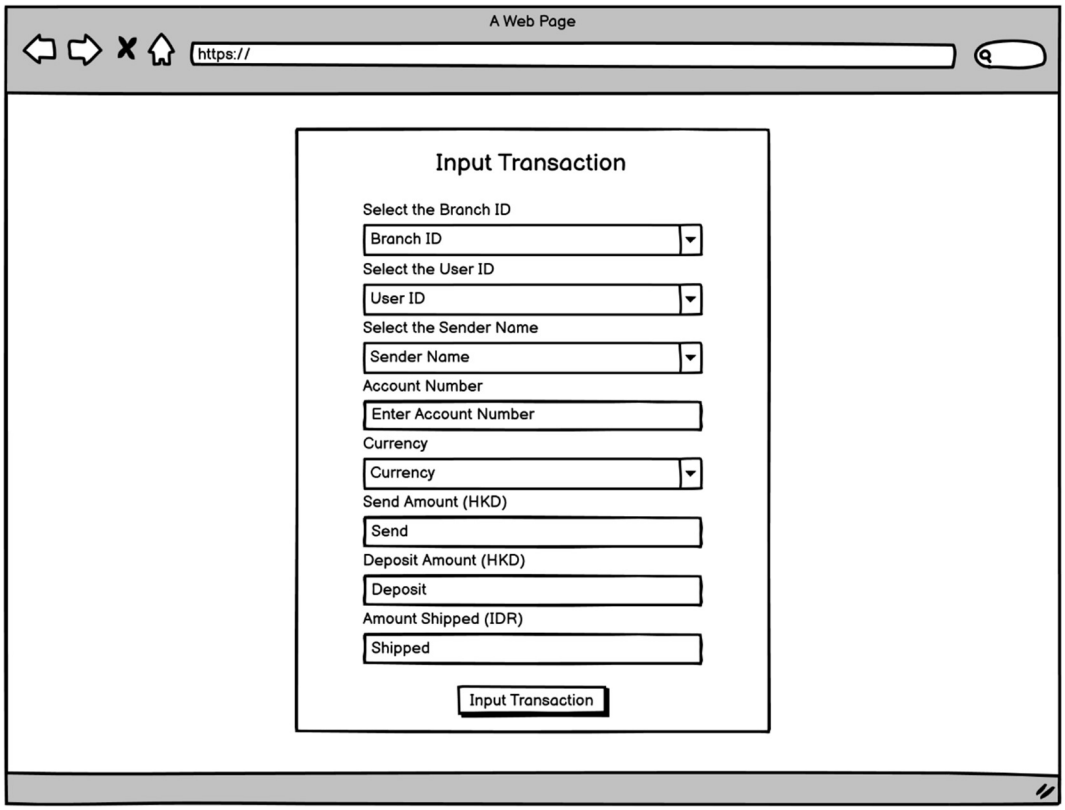


Figure 4.14 Add Transaction Page

- **Balance Page**

The balance page function is to show the user the statistic on the branch rate or fee. The page chart in page is bar charts and pie charts. The user can change the page by clicking the left and right arrow or swipe the page. The page illustration can be seen in Figure 4.15.

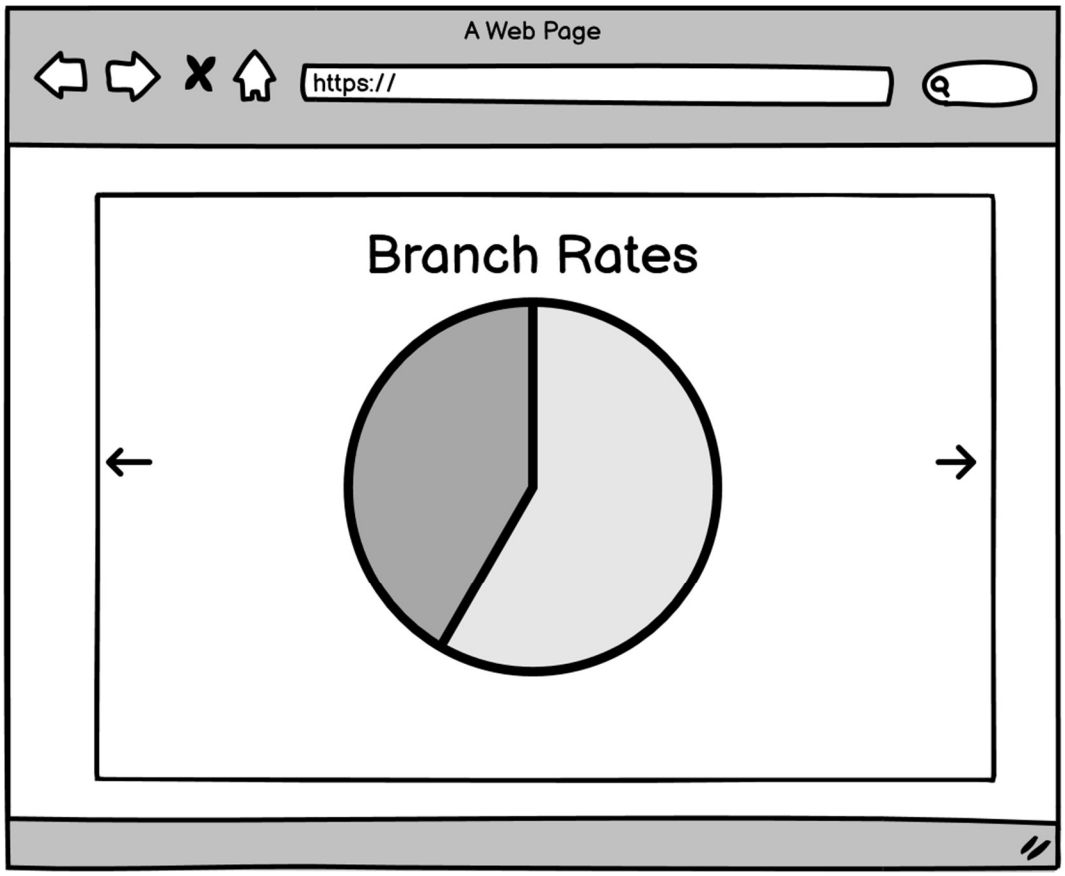


Figure 4.15 Balance Page

CHAPTER V SYSTEM IMPLEMENTATION

This chapter will discuss the design outcomes completed in the previous chapter. The backend is implemented using HTML, CSS, Javascript, and PHP.

5.1 Implementation of Database

The database design of the system will be covered in more detail in this subchapter. The system uses phpMyAdmin and MySQL to handle its database administration needs. A web-based administration tool called phpMyAdmin makes it possible to administer MySQL databases effectively by providing a navigable and user-friendly interface.

5.1.1 Admin Table

The administration data, which includes attributes such as id, password, name, address, phone number, email, and date registered, is stored in the admin table. The id attribute serves as the primary key in the admin table and has the properties of auto-increment and immutability. The administrator has the capability to update the password for the application. The organizational structure of the admin table can be visualized in Figure 5.1..

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
2	Password	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
3	Name	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
4	Address	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
5	PhoneNumber	int(200)			No	*None*			Change Drop More
6	Email	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
7	dateRegistered	date			No	*None*			Change Drop More

Figure 5.1 Admin Table

5.1.2 Cashier Table

The structure of the cashier table closely resembles that of the admin table, sharing similar attributes and data fields such as id, password, name, address, phone number, email, and date registered. The primary key in the cashier table, like the admin table, is auto-incremented and remains unchangeable. For a visual representation of the cashier table structure, please refer to Figure 5.2..

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**ID** 🔑	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	2	**Password**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**Name**	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	4	**Address**	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	5	**PhoneNumber**	int(200)			No	*None*			Change Drop More
☐	6	**Email**	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	7	**dateRegistered**	date			No	*None*			Change Drop More

Figure 5.2 Cashier Table

5.1.3 Transaction Table

The transaction table serves as a repository for storing customer transactions. The customer table exhibits a highly intricate structure, but only specific fields from the customer table are intended for utilization within the transaction table. It is imperative that the Id attribute of each transaction is not left null or empty. For a visual representation of the transaction table, refer to Figure 5.3.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	GUID	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
2	TransNum	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	BranchID	varchar(10)	latin1_swedish_ci		Yes	NULL			Change Drop More
4	TransDate	datetime			No	None			Change Drop More
5	CustomerID	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
6	CustomerName	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
7	CustomerPhone	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
8	CustomerAddress	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
9	SourceFunding	varchar(100)	latin1_swedish_ci		Yes	NULL			Change Drop More
10	ContractNum	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
11	ContractDate	date			Yes	NULL			Change Drop More
12	Occupation	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
13	Nationality	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
14	BirthPlace	varchar(100)	latin1_swedish_ci		Yes	NULL			Change Drop More
15	DOB	date			Yes	NULL			Change Drop More
16	SenderCity	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
17	SenderCountryCode	varchar(5)	latin1_swedish_ci		Yes	852			Change Drop More
18	BankAccountNo	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
19	ReceiverName	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
20	ReceiverPhone	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
21	ReceiverAddress	varchar(200)	latin1_swedish_ci		Yes	NULL			Change Drop More
22	Purpose	varchar(100)	latin1_swedish_ci		Yes	NULL			Change Drop More
23	BankCode	varchar(50)	latin1_swedish_ci		No	None			Change Drop More

24	**ReceiverProvince**	varchar(100)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
25	**ReceiverCity**	varchar(100)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
26	**ReceiverID**	varchar(50)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
27	**ReceiverCountryCode**	varchar(5)	latin1_swedish_ci	No	62	Change	Drop More
28	**Currency**	varchar(10)	latin1_swedish_ci	No	HKD	Change	Drop More
29	**ServiceName**	varchar(50)	latin1_swedish_ci	No	*None*	Change	Drop More
30	**Rate**	decimal(10,2)		Yes	*NULL*	Change	Drop More
31	**Amount**	decimal(10,2)		Yes	*NULL*	Change	Drop More
32	**CollectedAmount**	decimal(15,2)		Yes	*NULL*	Change	Drop More
33	**DestinationAmount**	decimal(15,2)		Yes	*NULL*	Change	Drop More
34	**RemittanceFee**	decimal(10,2)		Yes	*NULL*	Change	Drop More
35	**AdminFee**	decimal(10,2)		Yes	*NULL*	Change	Drop More
36	**CommissionFee**	decimal(15,2)		Yes	*NULL*	Change	Drop More
37	**RejectReason**	varchar(255)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
38	**CorrectionFee**	decimal(15,2)		Yes	*NULL*	Change	Drop More
39	**CorrectionReason**	varchar(255)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
40	**InvoiceNum**	varchar(50)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
41	**AgentID**	varchar(50)	latin1_swedish_ci	No	*None*	Change	Drop More
42	**AgentName**	varchar(50)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
43	**Status**	varchar(10)	latin1_swedish_ci	No	*None*	Change	Drop More
44	**StatusNotes**	varchar(200)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
45	**FixedBy**	varchar(50)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
46	**SentDate**	datetime		Yes	*NULL*	Change	Drop More
47	**FixedDate**	datetime		Yes	*NULL*	Change	Drop More
48	**PID**	int(11)		Yes	*NULL*	Change	Drop More
49	**BookingCode**	varchar(6)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
50	**Resend**	varchar(1)	latin1_swedish_ci	Yes	0	Change	Drop More
51	**CallbackCode**	varchar(20)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More
52	**BankCost**	float		Yes	*NULL*	Change	Drop More
53	**AccountingCode**	varchar(20)	latin1_swedish_ci	Yes	*NULL*	Change	Drop More

Figure 5.3 Transaction Table

5.1.4 Branch Table

The Branch table serves as a repository for storing branch-related data. It contains several fields, namely id, name, branch rate, branch fee, and date registered. The id field functions as an auto-incremented and immutable identifier for each branch entry. A visual representation of the table can be observed in Figure 5.4..

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id** 🔑	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**name**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**branchRate**	double			No	*None*			Change Drop More
☐	4	**branchFee**	double			No	*None*			Change Drop More
☐	5	**dateRegistered**	date			No	*None*			Change Drop More

Figure 5.4 Branch Table

5.1.5 Customer Table

In this system, customer data is stored in a structured format using the customer table. The table consists of various fields including id, name, birthplace, date of birth, phone, address, country, occupation, nationality, income source, contract date, user insert, insert date, user update, update date, and status ban. The visual representation of the table's structure is presented in Figure 5.5.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
2	Name	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
3	BirthPlace	varchar(100)	latin1_swedish_ci		Yes	NULL			Change Drop More
4	DOB	date			Yes	NULL			Change Drop More
5	Phone	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop More
6	Address	varchar(200)	latin1_swedish_ci		Yes	NULL			Change Drop More
7	City	varchar(50)	latin1_swedish_ci		Yes	Hong Kong			Change Drop More
8	Country	varchar(50)	latin1_swedish_ci		Yes	Hong Kong			Change Drop More
9	Occupation	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
10	Nationality	varchar(50)	latin1_swedish_ci		Yes	Indonesia			Change Drop More
11	Imgfile	varchar(255)	latin1_swedish_ci		Yes	NULL			Change Drop More
12	IncomeSource	varchar(50)	latin1_swedish_ci		Yes	Gaji			Change Drop More
13	ContractNum	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
14	ContractDate	date			Yes	NULL			Change Drop More
15	UserInsert	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
16	InsertDate	datetime			Yes	NULL			Change Drop More
17	UserUpdate	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
18	UpdateDate	datetime			Yes	NULL			Change Drop More
19	statusBan	int(11)			No	None			Change Drop More

Figure 5.5 Customer Table

5.2 Business Process Implementation

This chapter will explain the implementation of the main business process of the system. The process are: sign-up, change password, CRUD table, and show table.

5.2.1 Change Password Process

Changing a password in PHP entails multiple stages, including confirming the user's previous password and updating the user's password in the database.

List 5.1 Change Password

```
if(isset($_POST['btn-save'])) {
    $refID = $_POST['IDuser'];
    $oldPass = $_POST['oldPass'];
    $newPass = $_POST['newPass'];
```

```
    $confirmPass = $_POST['confirmPass'];
    if($newPass == $confirmPass){
        $stmt = $conn->prepare("update admin set Password='$newPass' where
ID='$refID'");
        $stmt->execute();
        if(!$conn){
            echo "Prepare failed: (". $conn->errno.") ".$conn->error."<br>";}
    }
    else{
        alert("New Password and ConfirmPassword is not the same");}}
```

The change password page is triggered when the user selects the save button. On this page, the user is required to input their user ID, old password, new password, and confirm password. To proceed with the password change, the user ID and old password entered by the user are compared against the data stored in the admin table. If a match is not found, the password change process will not be executed. Furthermore, it is necessary for the new password and confirm password fields to contain the same value in order for the process to continue. If the new and confirmed passwords do not match, the user is alerted about the mismatch. Subsequently, the process involves inserting the query and establishing a connection with the database. The final step is to execute the query, which initiates the password change process.

5.2.2 Show Table

Use PHP to retrieve data from a database and present it in a table format to make reports based on that data. Users may easily view the data and see any trends or patterns. Show table code can be seen in List 5.2.

List 5.2 Show Table

```
$stmt = $conn->query("select * from branch;");
$res = $stmt->fetch_all(MYSQLI_BOTH);
foreach($res as $row) {
        $id = $row['id'];
        $name = $row['name'];
        $branchRate = $row['branchRate'];
        $branchFee = $row['branchFee'];
        $dateRegistered = $row['dateRegistered'];
        echo"<tr>
            <td class='d-none d-xl-table-cell'>$id</td>
            <td class='d-none d-xl-table-cell'>$name</td>
            <td class='d-none d-xl-table-cell'>$branchRate</td>
            <td class='d-none d-xl-table-cell'>$branchFee</td>
            <td class='d-none d-xl-table-cell'>$dateRegistered</td>
        </tr>";}
```

The show table in the system will be used to show the table for the branch. The features that will be shown in the branch are the id, name, branch rate, branch fee, and date registered from the branch. The first thing the system will do is select all the features in the table branch. The following process is to fetch all the data in the table. Then the id, name, branchRate, branches, and dateRegistered will fetch from the table. The final step is to show the table from the branch.

5.2.3 Login

A login mechanism is a need for the majority of web applications. Due to its user-friendliness, cross-platform portability, database support, robust community support, and built-in security measures, PHP is a great choice for creating login systems. Here is a PHP script that uses the login system:

List 5.3 Login

```
$ID ="";

$Password = "";

if(isset($_POST['login'])){

   $ID = $_POST['emailLink'];

   $Password = $_POST['pass'];}

$query = "SELECT * FROM cashier WHERE Email = '$ID' AND Password
='$Password'";

$result = $conn->query($query);

if ($result) {

   if (mysqli_num_rows($result) > 0) {

       header("Location:index.php");

   } else {

       echo "<script type='text/javascript'>alert('The ID or Password
wrong');</script>"; }}
```

In list 5.21, the user needed to input their ID and password; from the ID and password, it will continue to be checked by using SQL query to search the user id and password in the cashier table. When the ID and password are correct, it will be continued to the main page; otherwise, the user will be warned if the user uses the wrong password or ID.

5.2.4 Block Customer Account

The page's purpose is to tell the user that their access has been limited or denied for a particular reason and to offer any instructions or information that may be required. The website uses a database to hold client information, and when a customer is to be barred, it sets a "blocked" flag for their customer ID.

List 5.4 Block ID code

```
if(isset($_POST['btn-save'])) {

        $refID = $_POST['id'];

        $stmt = $conn->prepare("UPDATE `customers` SET `status` = '1' WHERE
`customers`.`ID` = '$refID'");

        $stmt->execute();}
```

Suppose the user wants to block the customer from accessing the transaction. In that case, the user can block it by inputting the customer id. When the ID enters the system, the customer status will be updated to 1, meaning that any customer with 1 in the transaction will be blocked.

5.2.5 Export Excel

Making data more accessible, shared, and usable may be accomplished by exporting data to Excel using PHP. Users can make better judgments and work more efficiently if the user deals with data using PHP.

List 5.5 Export Excel

```
<?php
    header("Content-type: application/vnd-ms-excel");
    header("Content-Disposition: attachment; filename=Cashier.xls");?>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Address</th>
            <th>PhoneNumber</th>
            <th>Email</th>
            <th>dateRegistered</th>
            <th>mandiriBalance</th>
```

```
            <th>xferBalance</th>
        </tr>
        <?php
        $data = mysqli_query($conn,"select * from cashier");
        $no = 1;
        while($d = mysqli_fetch_array($data)){?>
        <tr>
            <td><?php echo $d['ID']; ?></td>
            <td><?php echo $d['Name']; ?></td>
            <td><?php echo $d['Address']; ?></td>
            <td><?php echo $d['PhoneNumber']; ?></td>
            <td><?php echo $d['Email']; ?></td>
            <td><?php echo $d['dateRegistered']; ?></td>
            <td><?php echo $d['mandiriBalance']; ?></td>
            <td><?php echo $d['xferBalance']; ?></td></tr><?php }?>
    </table>
```

First, to export the SQL table into PHP, the first thing to do is to specify the format file in Microsoft Excel. The next thing is to name the file name of the exported file. List 5.5 in the table will consist of ID, name, address, phone number, email, date registered, Mandiri Balance, and xferBalance.

5.2.6 Update Fee

Cashiers mainly use the code to update the branch's main fee. The update code also had same function with update rate. The details of the code can be seen in List 5.7.

List 5.6 Update Fee

```
if(isset($_POST['btnSave'])) {
        $branch_listSelected = $_POST['branch_listSelected'];
```

```
        $refSaldo = $_POST['refFee'];

        $stmt = $conn->prepare("update branch set branchFee='$refSaldo' where
id='$branch_listSelected'");

        $stmt->execute();}
```

The update page starts when the user selects the branch from the combo box and inputs the page fee. After that, the php will use the library that allows the user to use SQL for updating the table in phpMyAdmin.

5.2.7 Insert Cashier

The admin uses the codes to add new cashiers to the database page. The only one who can access the code is the admin, and the insert code can be seen in List 5.7.

List 5.7 Insert Cashier

```
if(isset($_POST['save'])) {

        $idCashier = $_POST['idCashier'];

        $pasCashier = $_POST['pasCashier'];

        $pasConfirmCashier = $_POST['pasConfirmCashier'];

        $nameCashier = $_POST['nameCashier'];

        $cashierAddress = $_POST['cashierAddress'];

        $cashierPhone = $_POST['cashierPhone'];

        $emailCashier = $_POST['emailCashier'];

        if($pasCashier == $pasConfirmCashier){

            $stmt = $conn->prepare("insert into cashier(ID, Password, Name,
Address, PhoneNumber,Email,dateRegistered) values(?, ?, ?,? ,?,?,?)");

            $stmt->bind_param("ssssiss",    $idCashier, $pasCashier,
$nameCashier,$cashierAddress , $cashierPhone

            , $emailCashier, date("Y-m-d") );

            $stmt->execute();

        }
```

```
        else{
            alert("Password and ConfirmPassword is not the same");}}
```

The cashier code will input the field that the user has already inputted. Following that, PHP will use the PHP library to connect to the Apikko database. After connecting to the database, the system will insert the transaction using the preset SQL code.

5.2.8 Insert Transaction Function

The code is used to insert the transaction of the customers. The insert branch, cashier, and customer data function follow a logic pattern similar to the code insert transaction function. The principle of the code is illustrated in List 5.8.

List 5.8 Insert Transaction

```
if(isset($_POST['save'])) {
        $GUID = $_POST['GUID'];
        $TransNum = $_POST['TransNum'];
        $BranchID = $_POST['BranchID'];
        $CustomerID = $_POST['CustomerID'];
        $CustomerName = $_POST['CustomerName'];
        $CustomerPhone = $_POST['CustomerPhone'];
        $CustomerAddress = $_POST['CustomerAddress'];
        $dateTransaction = date("Y-m-d");
        $sql = "INSERT INTO `transaction` (`ID`, `GUID`, `TransNum`,
`BranchID`, `TransDate`, `CustomerID`, `CustomerName`,
         `CustomerPhone`, `CustomerAddress`, `SourceFunding`, `ContractNum`,
`ContractDate`, `Occupation`, `Nationality`, `BirthPlace`,
          `DOB`, `SenderCity`, `SenderCountryCode`, `BankAccountNo`,
`ReceiverName`, `ReceiverPhone`, `ReceiverAddress`, `Purpose`,
          `BankCode`, `ReceiverProvince`, `ReceiverCity`, `ReceiverID`,
`ReceiverCountryCode`, `Currency`, `ServiceName`, `Rate`,
```

```
          `Amount`, `CollectedAmount`, `DestinationAmount`, `RemittanceFee`,
`AdminFee`, `CommissionFee`, `RejectReason`,

          `CorrectionFee`, `CorrectionReason`, `InvoiceNum`, `AgentID`,
`AgentName`, `Status`, `StatusNotes`, `FixedBy`,

          `SentDate`, `FixedDate`, `PID`, `BookingCode`, `Resend`,
`CallbackCode`, `BankCost`, `AccountingCode`)

          VALUES (NULL, '$GUID', '$TransNum', '$BranchID', '$dateTransaction',
'$CustomerID', '$CustomerName', '$CustomerPhone', '$CustomerAddress',

          NULL, NULL,

          NULL, NULL, NULL, NULL, NULL, NULL, '852', '', '', NULL, NULL, NULL,
'', NULL, NULL, NULL,

          '62', 'HKD', '', NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, '', NULL,

          '', NULL, NULL, NULL, NULL, NULL, NULL, '0', NULL, NULL, NULL);";

        if ($conn->query($sql) === TRUE) {

            alert("New record created successfully");

        } else {

            alert("Hello");

            echo "Error: " . $sql . "<br>" . $conn->error;

        }

        $conn->close();}
```

The insert code will not insert all of it into the transaction table, and only a particular field will be added to the transaction table. In the transaction code, the user already fills the input that must be in the table. After that, PHP will use the PHP library that allows users to connect with the Apikko database. After connecting to the database, the system will insert the transaction using the predetermined SQL code.

5.2.9 Get Detail Transaction

The get detail code is used in the transaction table for the user can edit specific transactions. The function has a similar principle to get detailed branches, cashiers, and customer data. The code can be seen as illustrated in List 5.8.

List 5.9 Detail Transaction Input Box

```
if($_POST['function'] == "getdetailtransaksi") {
    $guid   = $_POST['guid'];
    $stmt = $conn->query("select * from transaction where GUID = '$guid'");
    $res = $stmt->fetch_all(MYSQLI_BOTH);
    foreach($res as $row) {
        echo json_encode($row);
    }
}
```

List 5.9 code is started when the transaction id is inputted into the database. The id will be used to read the table transaction in the database phpMyAdmin. After receiving the result, it will return it to the page that needs the table.

5.2.10 Update Detail Transaction

The code is mainly to edit the transaction of particular transaction IDs. The function has a similar design to update detailed customers, cashiers, and branch tables. The code of the Update detail transaction is illustrated in List 5.10.

List 5.10 Edit Transaction

```
else if($_POST['function'] == "updatedetailtransaksi") {

    $guid           = $_POST['guid'];

    $transnum       = $_POST['transnum'];

    $branchid       = $_POST['branchid'];

    $customerid     = $_POST['customerid'];

    $customername   = $_POST['customername'];

    $customerphone  = $_POST['customerphone'];

    $customeraddr   = $_POST['customeraddr'];

    $stmt = $conn->prepare("update transaction set TransNum = '$transnum',
BranchID = '$branchid', CustomerID = '$customerid', CustomerName =
'$customername', CustomerPhone = '$customerphone', CustomerAddress =
'$customeraddr' where GUID = '$guid'");

    $stmt->execute();

    echo "success";

}
```

In List 5.9, the code is shown to update the transaction table. The user needed to input the transaction number, branch id, customer id, customer name, customer phone, and customer address into the transaction. After that, the table will be updated by using SQL query. If the table is successfully updated, it will return "success."

5.2.11 Delete Transaction

The delete transaction function is to delete the transaction in the table transaction. A similar logic of the code function can be seen in the delete branch, cashier, and customer data functions. The process of the code is shown in list 5.11.

List 5.11 Delete Transaction

```
else if($_POST['function'] == "deleteDetailtransaksi") {

    $transnum       = $_POST['transnum'];

    $stmt = $conn->prepare("DELETE FROM transaction WHERE TransNum='$transnum
';");

    $stmt->execute();

    echo "success";

}
```

In list 5.11 is executed after the user inputs the transaction number. The code will run an SQL query to delete the transaction by locating the transaction number the user had inputted. It will return "Success" after the code is successfully executed.

CHAPTER VI Testing and Evaluation

This chapter discusses the comprehensive evaluation of the implemented system, encompassing the examination of its functionality and performance. In particular, focus on presenting the system's test results and thoroughly assessing its user interface (UI) to ascertain its seamless operation and user-friendliness. This evaluation aims to comprehensively understand the system's capabilities, offering valuable insights into its effectiveness and usability.

6.1 Verification

The implementation verification results for the system are presented in this sub-chapter. The verification will explain the UI and the quality of the system.

6.1.1 Login Verification

In the context of user login, when a cashier logs in, they are prompted to select a branch. Once the branch selection is made, the user is redirected to the login page. In contrast, for the admin user, the branch selection step is bypassed, and they are immediately directed to Figure 6.2. The user login page is visually represented in Figure 6.1.

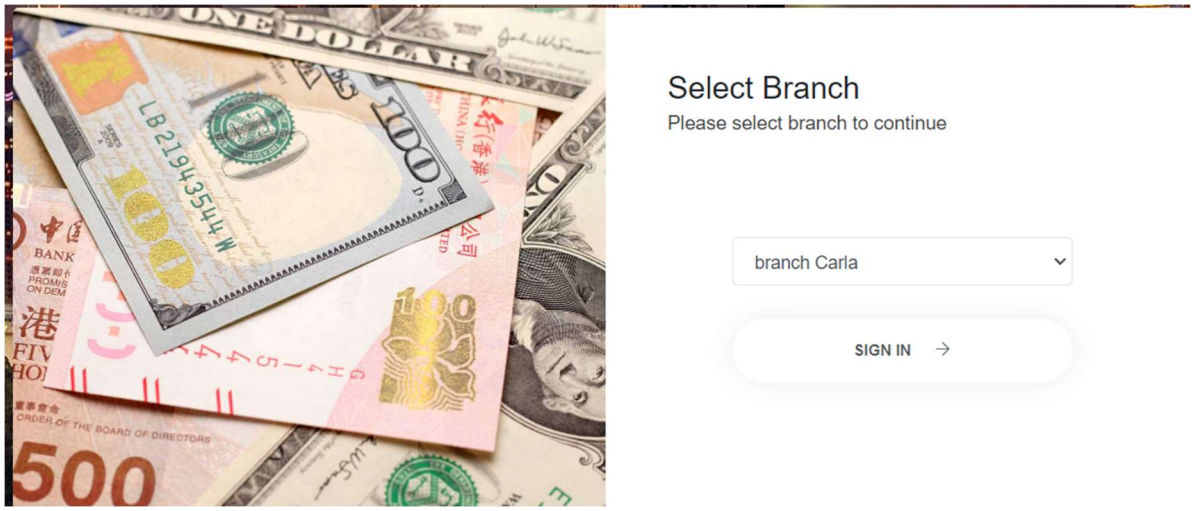


Figure 6.1 Login Cashier

By entering their email and password in the login box, users log in to the website, and Cashiers can log in from the selected branch. If the user puts the wrong id or password, the user will immediately be warned if the user uses the wrong id or password. If the user input correctly, it will immediately be directed to the cashier page; on the other hand, for admin, it will then be the admin page.

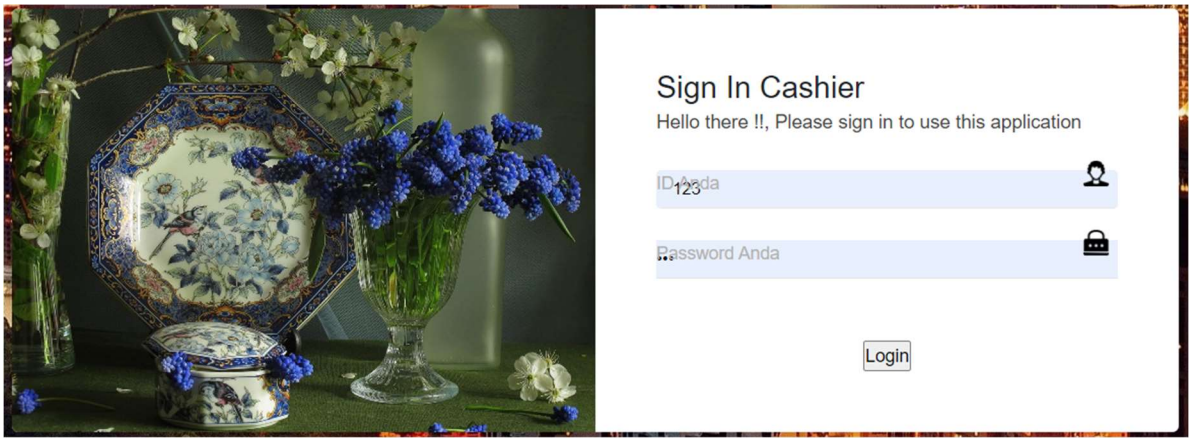


Figure 6.2 Login Form

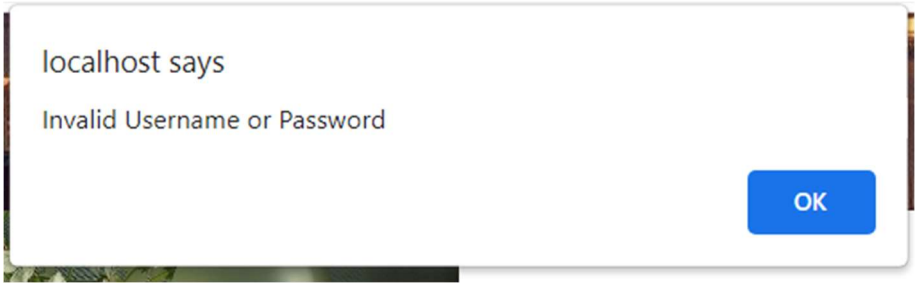


Figure 6.3 Warning for Invalid Username or Password

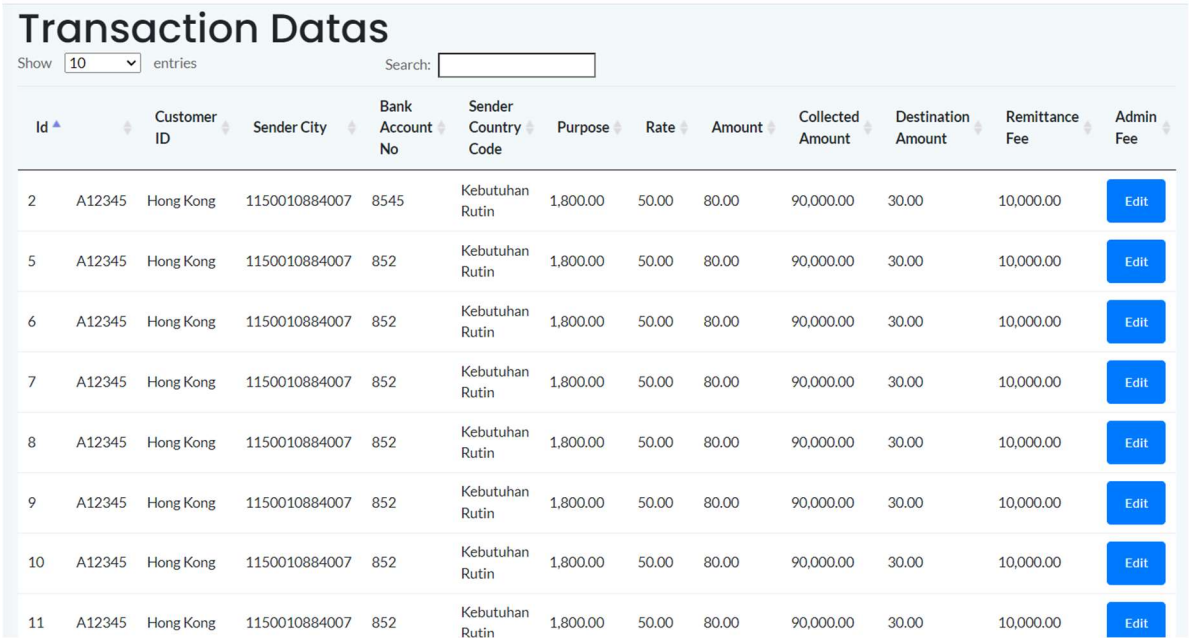
6.1.2 Log Out

The user can log out from the site by clicking the log out from the menu. The logout menu is illustrated in Figure 6.4.

Figure 6.4 Log Out

6.1.3 Transaction Report Page

The transaction report page encompasses various functionalities, while other pages, such as customer, branch, and cashier reports, exhibit a similar design structure to the transaction page. These pages share standard features, including search engines, a sorting tool, and an edit and delete tool. A visual representation of the page is provided in Figure 6.5. Users can utilize the sorting functionality by clicking the up or down arrow icons. The up arrow icon enables ascending sorting, while the down arrow icon facilitates descending sorting.



Transaction Datas

Show 10 entries Search:

Id		Customer ID	Sender City	Bank Account No	Sender Country Code	Purpose	Rate	Amount	Collected Amount	Destination Amount	Remittance Fee	Admin Fee
2	A12345	Hong Kong	1150010884007	8545	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
5	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
6	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
7	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
8	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
9	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
10	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
11	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit

Figure 6.5 Transaction Report

When the user presses the edit button on the page, a module form will appear in front of the user, like in Figure 6.6. The edit form gives the user three options: to cancel, edit or delete the row on the page. For example, in Figure 6.6, we want to edit the city name from Surabaya to Hongkong. After we edit it, figure 6.7 will appear. The only textbox that cannot be edited is the id textbox of the form. If we want to delete the row in the table, it will immediately be sent to the Status transaction page shown in Figure 6.7.

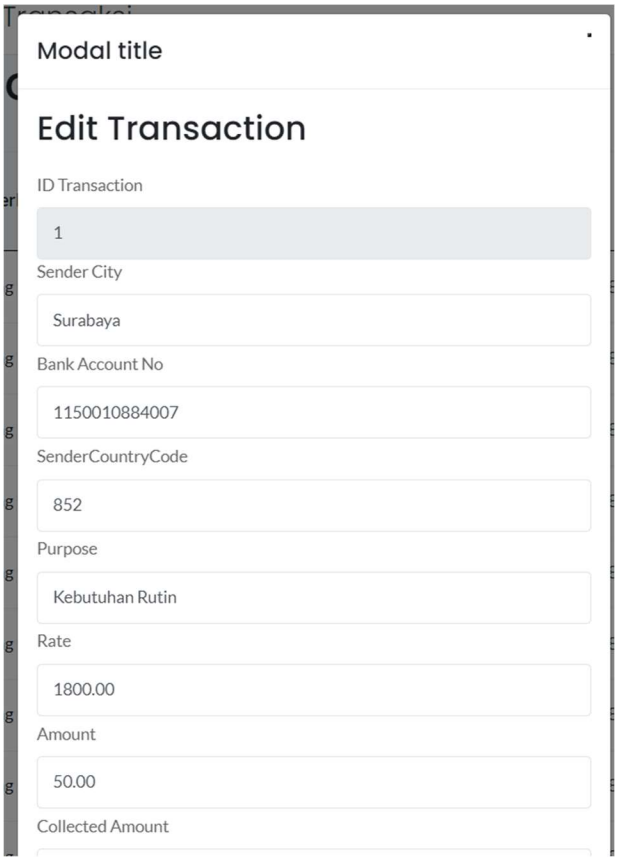


Figure 6.6 Form Module for Transaction

Transaction Datas

Show 10 entries Search:

Id	CustomerID	Sender City	Bank Account No	Sender Country Code	Purpose	Rate	Amount	Collected Amount	Destination Amount	Remittance Fee	Admin Fee	
1	A12345	Surabaya	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
2	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
3	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
4	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
5	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
6	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
7	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit
8	A12345	Hong Kong	1150010884007	852	Kebutuhan Rutin	1,800.00	50.00	80.00	90,000.00	30.00	10,000.00	Edit

Figure 6.7 Edit Sender city from Surabaya to Hong Kong

6.1.4 Status Transaction Page

The status transaction page allows the user to see the deleted transaction row. The user can undelete the transaction page by clicking the undelete button. The page also has the same table features as the transaction page. The page illustration can be seen in Figure 6.8. We either had two choices what we could do with this row to delete it permanently or return it to the transaction report. Either one of the buttons is clicked, a pop-up message if its success will appear like In Figure 6.9

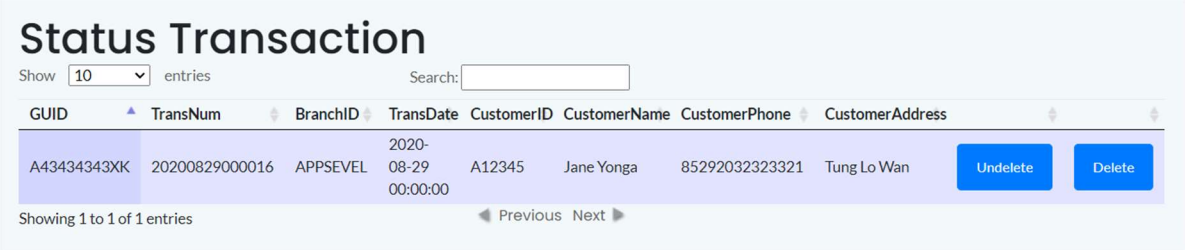


Figure 6.8 Status Transaction

Figure 6.9 Pop-Up Success Message

6.1.5 Change Password Page

This sub-chapter evaluates the change password functionality, allowing users to modify their passwords on the designated page. The process entails users providing their old password and desired new password. If the user enters an incorrect old password, a warning will be triggered to alert the user of the error. Moreover, the user must confirm the new password, ensuring that both the new and confirmed passwords match. A warning will be displayed in the event of a mismatch,

indicating that the passwords are different. The visual representation of the change password page can be found in Figure 6.10, providing a visual reference for the discussed functionality.

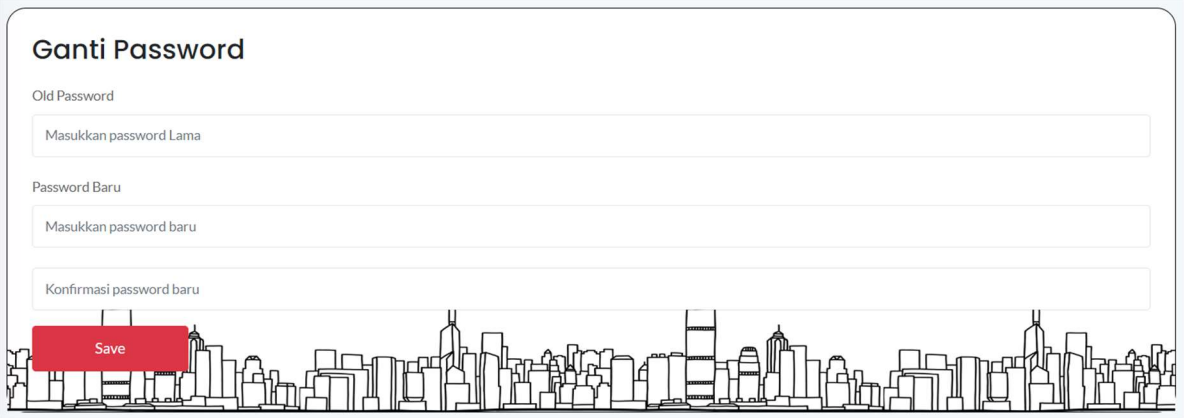


Figure 6.10 Change Password Page

6.1.6 Block or Unblock Customer Page

The block page is used by the user to block customers from transactions or to unblock them, customers. The user can block the customer by selecting the customer in the combo box and clicking the block customer button. For unblocking the user use the box below the block customer page. The page illustration can be seen in Figure 6.11. Suppose the user selects "Pengimaa" as the block customer. In that case, it will immediately transfer to the unblock customer box, like in Figure 6.12 The cashier will not be able to do more transactions.

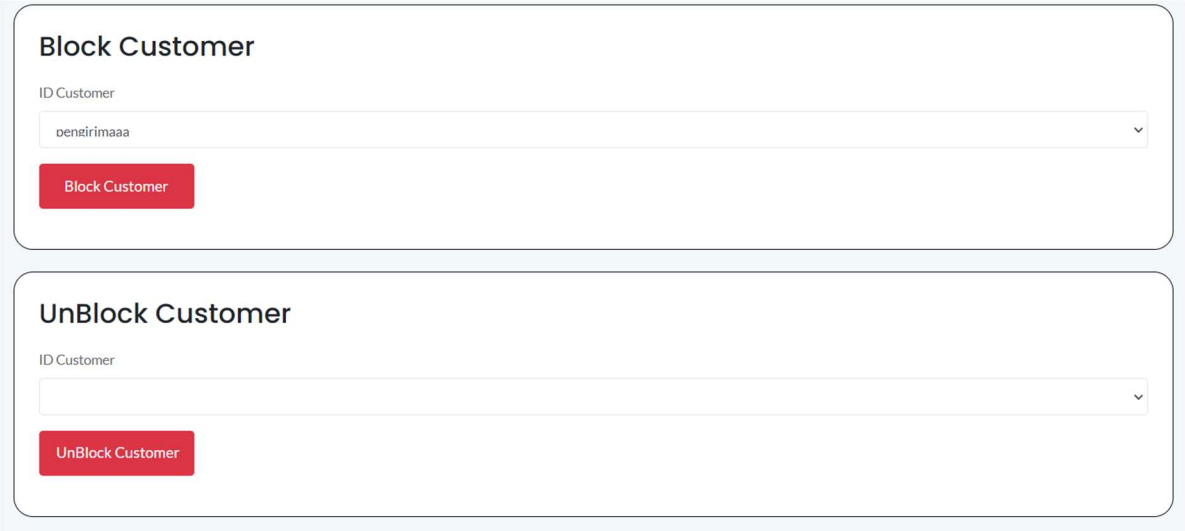


Figure 6.11 Block or Unblock Customer Page

Figure 6.12 Example of Blocking Customer

6.1.7 Add Branch Page

The user uses the add branch page to add a branch to the page. Other pages have the same concept as add branch pages, such as add transaction and cashier pages. Users only need to input the provided textbox on the page and click the add cashier button. The page illustration add branch can be seen in Figure 6.13. The added new branch can be seen in figure 6.14 shows the added branch "new."

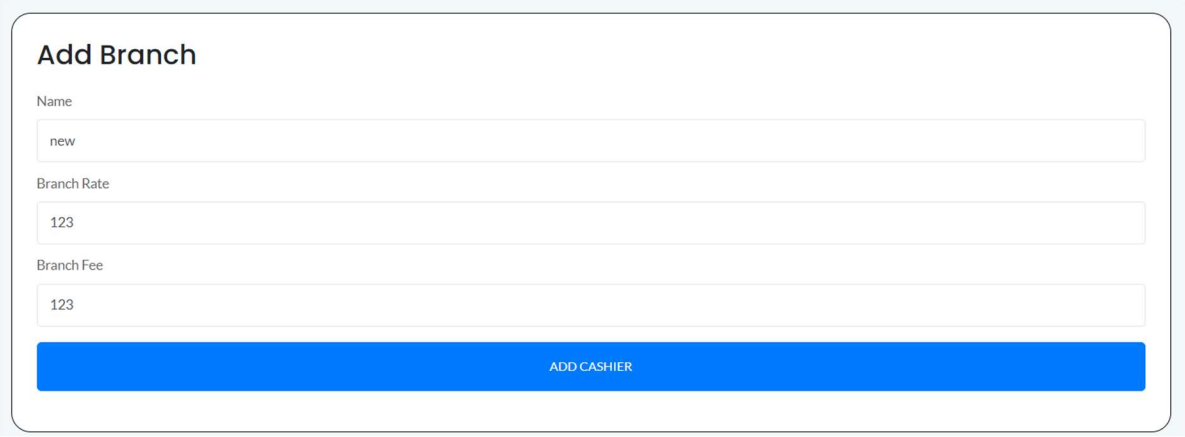


Figure 6.13 Add Branch Page

6.1.8 Cashier or Branch Report Page

This sub-chapter evaluates the Cashier or Branch Page, which shares the same design as the transaction page with a distinct feature—enabling users to export tables to Excel. Users can open the file explorer by clicking the "Export Excel" option, allowing them to upload the file to a designated location, as depicted in Figure 6.15. For a visual representation of the page, refer to Figure 6.14.

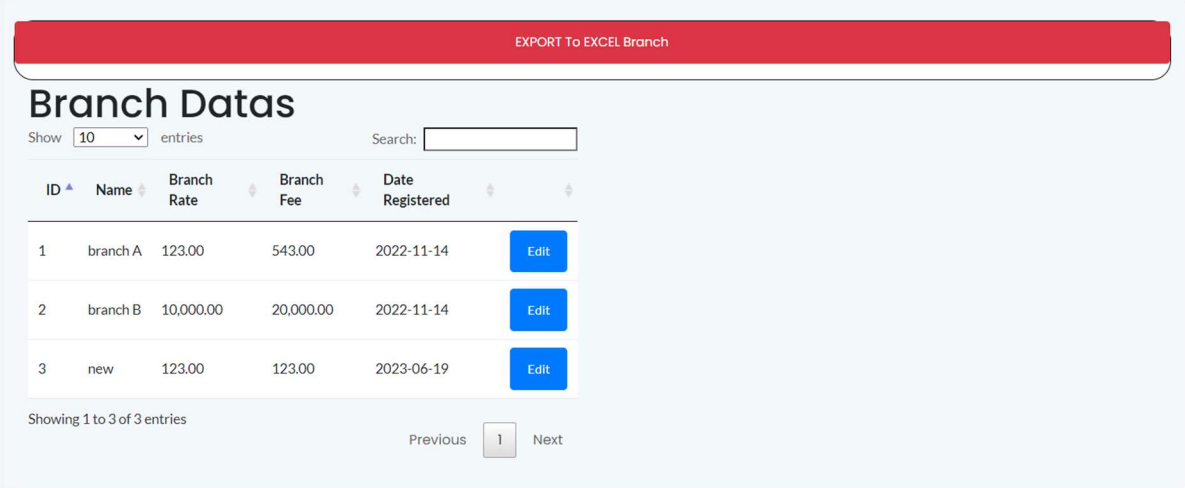


Figure 6.14 Cashier or Branch Report Page

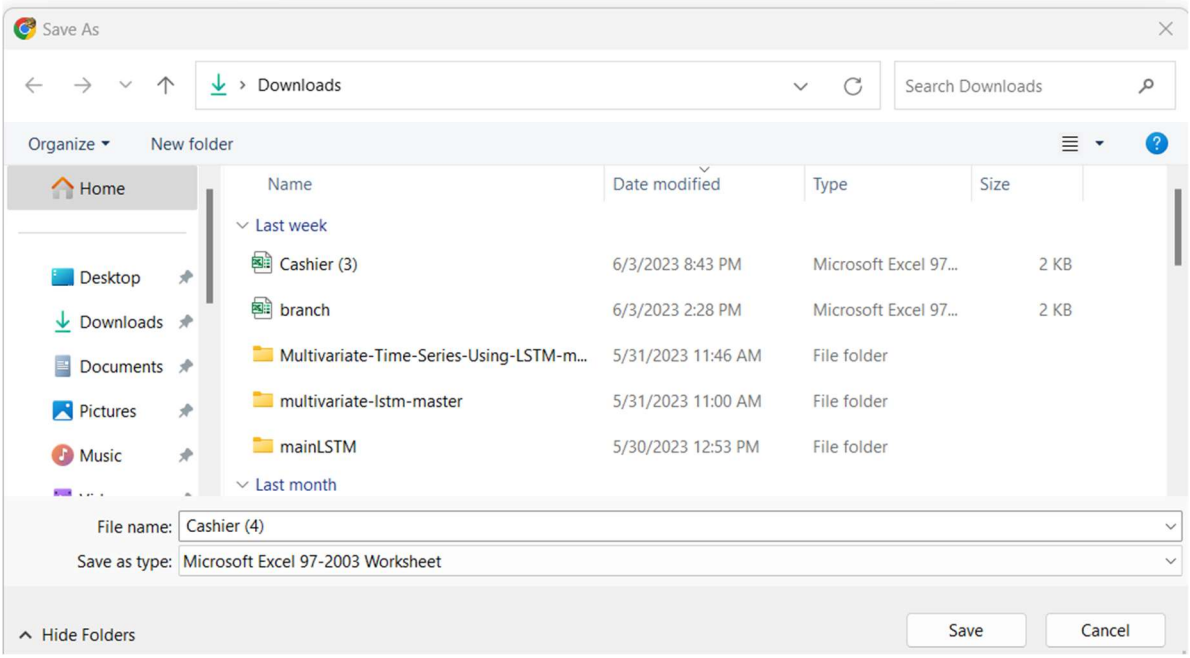


Figure 6.15 Upload Action

6.1.9 Balance Page

The user uses the balance page to see the branch fees or rate chart as a pie chart or bar chart. On the balance page, users can slide the page to see other charts in the model of the page by clicking the right or left arrow. The page is illustrated in Figure 6.16.

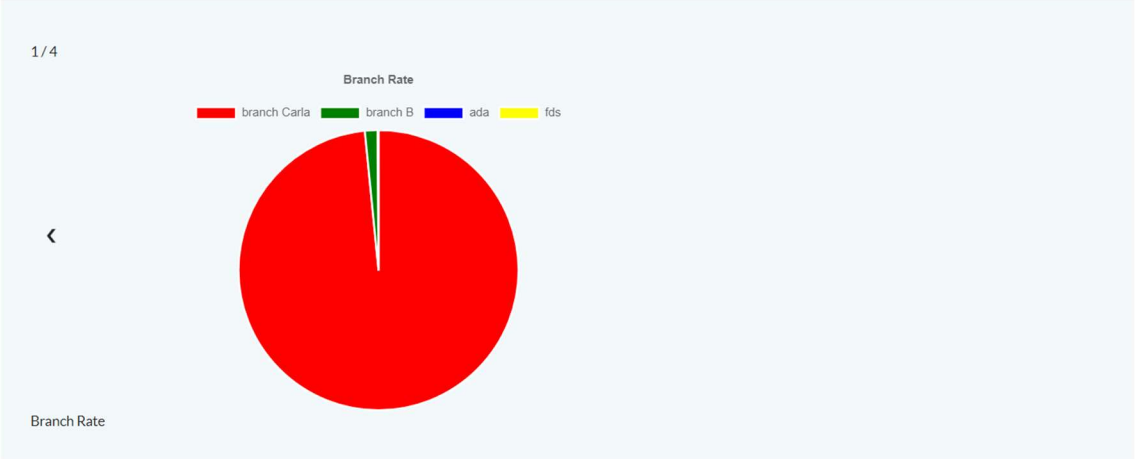


Figure 6.16 The Chart Page

6.1.10 Edit Branch Fee or Rate Page

The Edit branch fee or rate page can be used to edit the branch fee or rate. The user only needs to select the branch name from the combo box and input the fee or rate of the branch, and then the user needs to click submit to edit it. For example, branch new is edited from 123 to 1000, as shown in Figure 6.17, and will give the result as shown in 6.18.

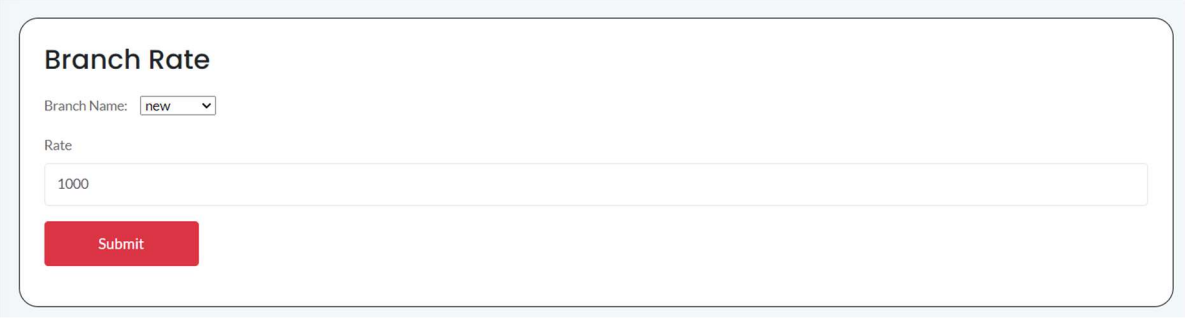


Figure 6.17 Edit Branch Fee or Rate Page

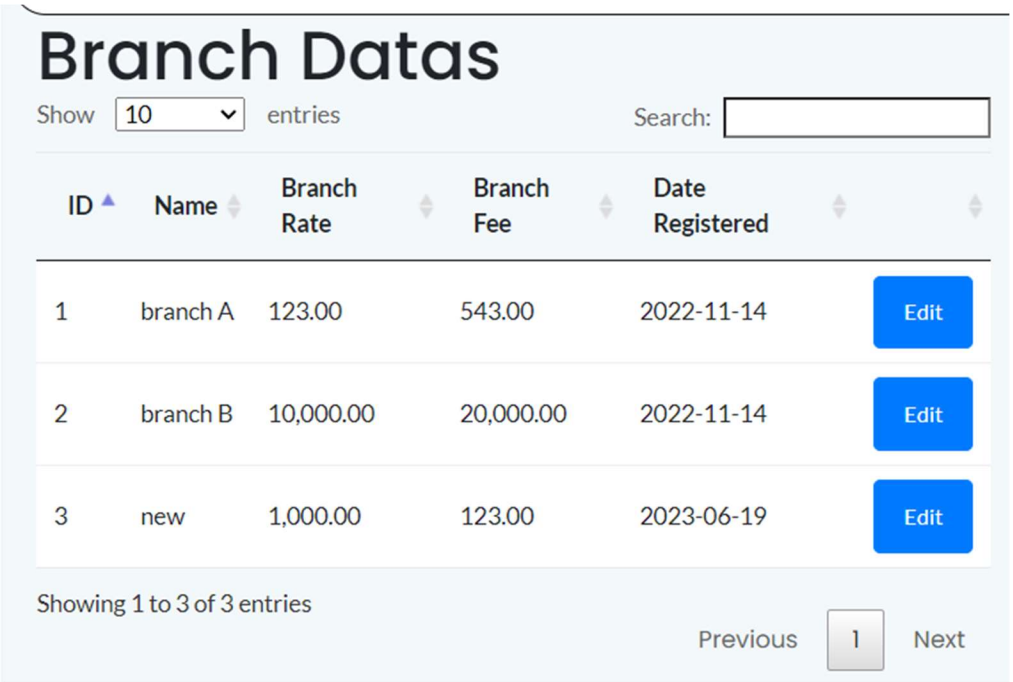


Figure 6.18 Edit Result of the Branch Rate

6.2 Questionnaire Responds

The collected primary survey was performed to understand the system's performance. Around four respondents answered, and the result is as follows:

1. Figure 6.19 illustrates the respondent's view of the quality of the admin system. The majority of respondents gave the admin system a rating of 4, which denotes high overall quality, according to the chart, showing that they had a generally favorable opinion of it. It is important to note, though, that the lack of ranks at the extremes (1 and 5) suggests that there may be an opportunity for improvement.

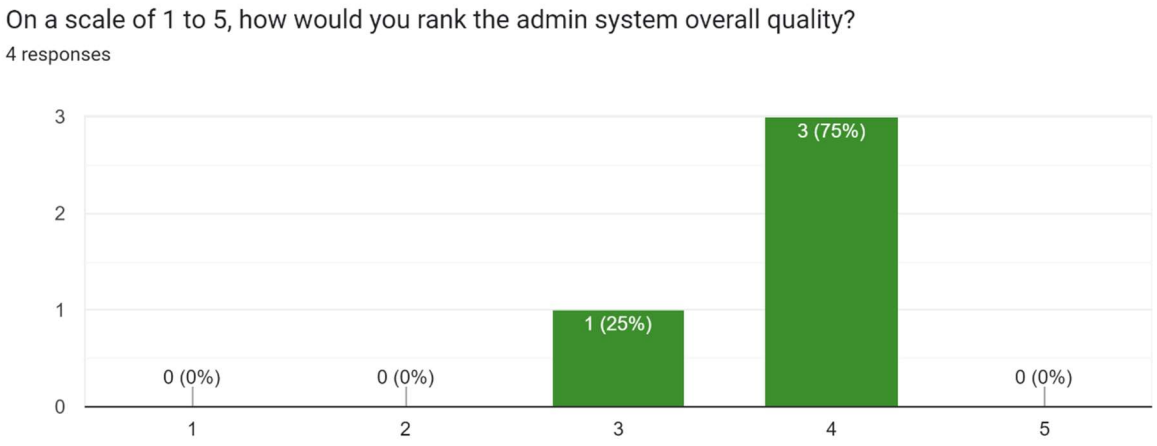


Figure 6.19 Chart of Admin system quality

2. The second questionnaire asks respondents, "Have there been any problems or flaws with the admin system? If so, kindly give further information?" The survey results show a variety of experiences with the administrative system. While some respondents mention minor concerns or have no problems, others highlight significant problems like glitches, delays, and insufficient alerts. The feedback also shows a desire for enhancement and objectivity regarding

the system's condition. These comments can offer insightful information for additional analysis and future admin system improvements.

3. Figure 6.20 describe the system that needed to be improved. The feedback provided by the respondents identifies several areas where the administrative system can be improved. These areas include the user interface and navigation, system dependability and stability, security and data protection, user permissions and access controls, performance and response times, reporting and analytics capabilities, compatibility with different browsers and devices, and data synchronization and integrity. These perceptions can direct efforts to raise the caliber of the system and cater to the particular worries and requirements of the consumers.

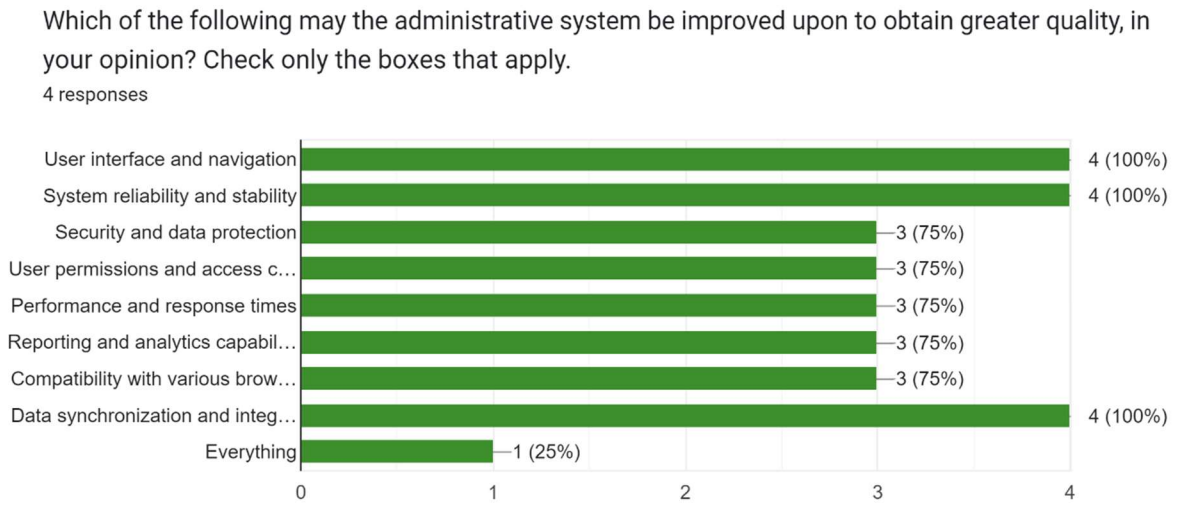


Figure 6.20 Questionaire about the need for the system improvement

4. Figure 6.21 The replies show the respondent's opinions on the system's capabilities—a range of opinions on whether the administrative system has all the features required for efficient administration. While most respondents think the system lacks specific crucial components, others think it fulfills their needs. The lack of understanding of the system's capabilities that

some respondents stated suggests a possible need for enhanced communication. This input can assist direct efforts to solve any functional gaps and increase the efficiency of the admin system for administrative activities.

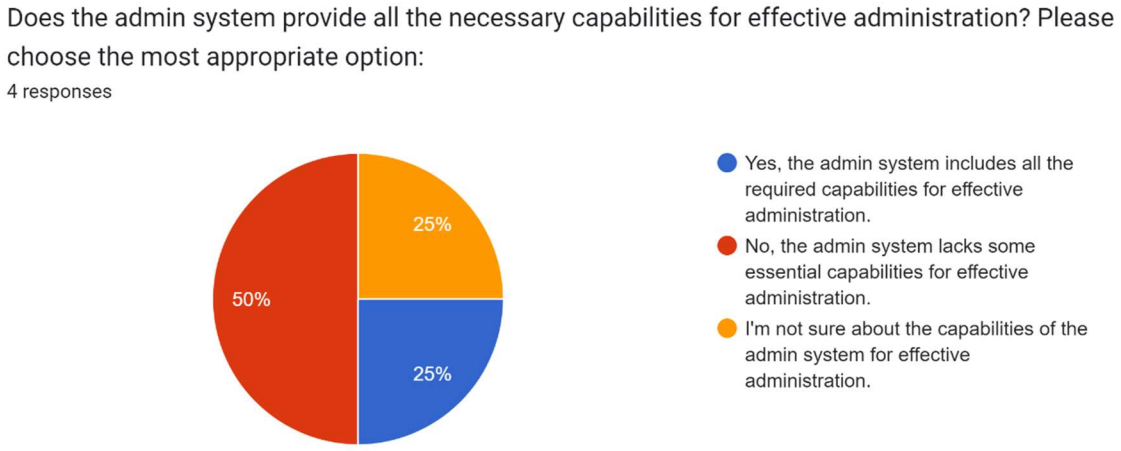


Figure 6.21 Chart Shows The Capability of The Administrative System.

5. The fifth questionnaire asked, "Are there any features that may be deleted because they are redundant or unnecessary?" The respondent's comments differ from each other. One person recommends adding additional features, while another considers all present functions beneficial. The third response is aware that features might eventually stop being applicable or required. These replies show differing opinions on the current feature set and suggest that ongoing assessments may be necessary to keep the administrative system efficient.
6. The sixth questionnaire asked, "Is the system's navigation easy to understand and logical?" The responses demonstrate the diversity of opinions on how logical and intuitive the system's navigation is. While some individuals have no trouble understanding it, others struggle with systematization, clarity, or simplicity. These observations can pinpoint areas where the system's navigation design needs to be enhanced to improve usability and user-friendliness.

7. Most respondents awarded the admin system's user interface (UI) a three out of five stars rating, as seen in the graph in Figure 6.22. Fewer respondents (rating of 4) thought it was more user-friendly. No respondents rated the user-friendliness of the UI a high or low grade. These rankings provide information on the admin system's present usability, which may help focus efforts on enhancing its usability and the user experience.

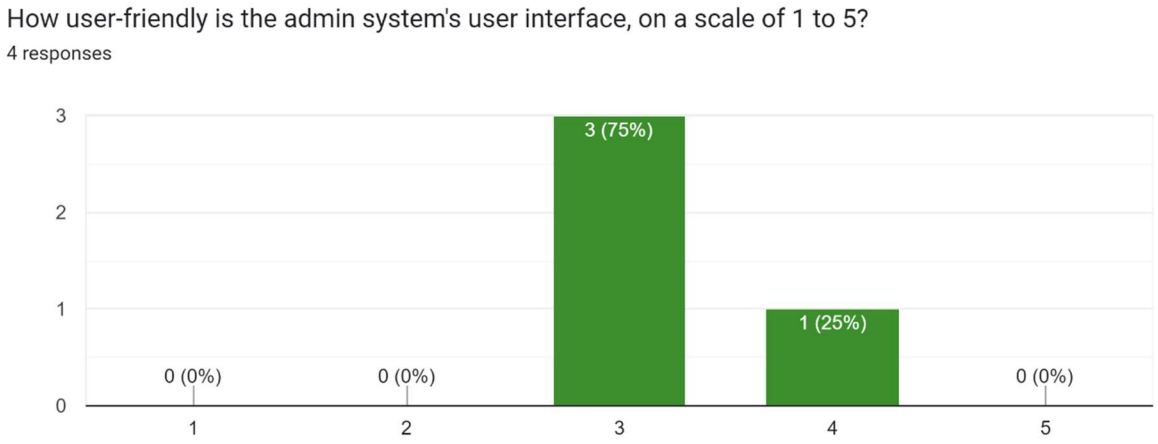


Figure 6.22 Chart Shows the Respondent' Opinions on the User-Friendly System

8. The eighth questionnaire asked respondents, "Is the admin system stable and free from unforeseen errors or crashes?" The feedback reveals that the majority of respondents concur that the system is error-free. While few individuals express concern about flaws and instability, some assert that the system is stable and devoid of errors. These arguments demonstrate that there is little need to address the system crash.
9. The nine questions asked respondents, "Is the system adaptable enough to support future additions of modules or functionality?" According to the survey's findings, all respondents agreed that the system is flexible enough to accommodate future additions of modules or capabilities. According to all respondents, the system is flexible enough to support future

additions. This may indicate that they think the system is adaptable and capable of incorporating new modules or features as necessary.

10. The last question asked the respondent, "How effectively can a system manage big data quantities without sacrificing performance?" The replies suggest that users assess the system's ability to manage large amounts of big data without degrading performance as moderate or reasonable but not extraordinary. It is decent, according to one respondent, good enough for another, and not highly effective, according to another. These divergent perspectives suggest that there could be potential for improvement in the system's capacity to efficiently manage and analyze vast amounts of data while preserving performance.

CHAPTER VII Conclusion and Future Works

This chapter tries to provide the conclusions drawn from the research findings and offers suggestions for improving the study's quality. It will also clarify the conclusions from the results gathered and offer ideas for improving the study strategy.

7.1 Conclusion

Considering the findings, the following conclusions may be drawn from the research that was done:

1. The system is able to successfully manage massive volumes of data without noticeably degrading speed. This shows that the system can manage data well and continue to operate at its best.
2. The system manages a good report. The reports that the system generates are accurate, well-organized, and offer pertinent data for analysis and decision-making.
3. The system design is pretty user-friendly, despite there still being some areas for improvement in usability. This means that, on the whole, consumers find the system simple to use and navigate, pointing to a well-designed interface that enhances the user experience.
4. The system doesn't have any significant errors or issues. As a result, users have access to a trustworthy and stable platform for data management and analysis, indicating that the administration system is largely error-free. If there are no reported problems, it is obvious that the system is working as intended and does not impede user processes.

7.2 Future Works

The following suggestions are made for future works:

1. The system security currently is only user access control. There is not enough security for illegal intrusion against the system. There is a need for an Intrusion detection and prevention system (IDP/IPS).
2. There is a need to enhance reporting and analytics tools capabilities inside the administrative system. Create improved features and capabilities to give users access to thorough and intelligent data analysis.
3. There is a need to enhance the understanding of the system's user. The system is easy to understand, but there is no guide for the user to understand the system mechanism. Ensure that users know all the features and functionalities offered by the admin system.

References:

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database system concepts*. McGraw-Gill Education.

Pratt, P. J., & Adamski, J. J. (2015). *Concepts of database management*. Course Technology.

Reeve, A. (2013). *Managing data in Motion: Data Integration, best practice techniques and technologies*. Morgan Kaufmann, an imprint of Elsevier.

Bernard, M., & Bachu, E. (2015). *Database systems with case studies*. PHI Learning Private Limited.

Coronel, C., & Morris, S. (2023). *Database systems: Design, implementation, & management*. Cengage Learning.

Sumathi, S., & Esakkirajan, S. (2007). *Fundamentals of Relational Database Management Systems*. Springer.

Bush, J. (2020). *Learn SQL database programming*. Packt Publishing Limited.

S., B., VISHWAJIT. ZANJAT, SHRADDHA N. KARMORE, BHAVANA. (2020). *Fundamentals of Database Management System*. LAP LAMBERT ACADEMIC PUBL.

Kahate, A. (2004). *Introduction to database management systems*.

Yannakoudakis, E. J., & Cheng, C. P. (1988). *Standard relational and network database languages*. Springer-Verlag.

Harrington, J. L. (2016). *Relational database design and implementation*. Morgan Kaufmann.

Grippa, V. M., & Kuzmichev, S. (2021). *Learning mysql: Get a handle on your data*. O'Reilly Media, Inc.

Foreman, D. (2021). *Bootstrap 5 foundations*. Independently published.

Williams, H. E., & Lane, D. (2004). *Web database applications with PHP and mysql*. O'Reilly.

Oussalah, M. (2014). *Software architecture*. ISTE/Wiley.