Cadmus Financial Corporation

A Project submitted to the

Yashwantrao Chavan Maharashtra Open University, Nashik.

ज्ञानगंगा घरोघरी

In partial fulfilment of the requirements for the degree of

Bachelor Of Computer Application (B.C.A)

Atharva Institute of Information Technology

ATHARVA
INSTITUTE OF
INFORMATION TECHNOLOGY

Under the guidance of

Mrs. Shilpa Mistry

By

SHARVIL RAKESH RANE **PRN: 2020017000507847**

2022-23

CERTIFICATE OF EVALUATION

This is to certify that the undersigned have assessed and evaluated the project work titled **"Cadmus Financial Corporation"** submitted by the following student.

Student's Name	**PRN Number**
SHARVIL RAKESH RANE	**2020017000507847**

The project report has been (accepted) for the partial fulfilment of the Bachelors of Computer Application Program.

Signature of the examiner: _____

Name of the examiner: _____

Stamp of the study centre

CERTIFICATE OF COMPLETION

This is to certify that the team of the following students of **B.C.A** have completed the Project Work titled **"CADMUS FINANCIAL CORPORATION"** under my guidance and supervision. The project report has been written according to the guidelines given by the Yashwantrao Chavan Maharashtra Open University.

Student's Name	**PRN Number**
SHARVIL RAKESH RANE	**2020017000507847**

Signature of the Study Centre Coordinator

Signature of the Guide

Name of the Study Centre Coordinator

Name of the Guide

Stamp of the study centre

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the people who directly or indirectly enabled to work on this project.

At the outset, I would like to thank the entire staff of Atharva Institute of Information technology, both the teaching and non-teaching staff for their continuous and unrelenting efforts in helping us with our project. Without them, these efforts would have not given us expected results.

I would like to thank to my project internal guide **Mrs**. **Shilpa Mistry** for constant support and the knowledge that she has shared with me for her guidance and support in completing my project.

I would also like to thank **Mrs. Aarti Oberoi (Centre Head)** and **Mr. Ashok Yadav (Co-ordinator)**. Their unwavering dedication and profound experience have played a pivotal role in motivating and inspiring me to achieve greater heights.

Last but certainly not least, I would like to express my sincere and heartfelt gratitude to all the teachers and friends who have played a significant role in assisting me throughout this project. Their support, guidance, and contributions have been invaluable.

I express my honest gratitude. Thank you.

SHARVIL RAKESH RANE.

SYNOPSIS

Cadmus Financial Corporation is an optimized banking management system. This Banking Management System project is a comprehensive software solution designed to streamline and automate various banking operations, ensuring efficiency, accuracy, and enhanced customer experience. This project aims to provide a robust platform for managing banking activities, such as account management, transaction processing, customer relationship management, and employee management, loan application processing and much more.

The primary objective of the Banking Management System is to develop a user-friendly and secure system that enables banks to optimize their operations, improve customer service, and maintain effective management of financial transactions. The project aims to create a centralized platform that integrates various banking processes, ensuring smooth workflows, reducing manual efforts, and minimizing errors.

The solution enables banks to improve services, cut costs, and consolidate processes. By integrating various banking operation into a single platform, the bank can simply automate manual processes and eliminate redundant tasks. With minimum administrative overhead, banks can instead focus on their customers and demands.

By offering a comprehensive software solution that improves productivity, accuracy, and customer satisfaction, this project seeks to completely revamp banking operations.

DECLARATION

I hereby declare that the project entitled, **"CADMUS FINANCIAL CORPORATION"** done at **ATHARVA INSTITUE OF INFORMATION TECHNOLOGY,** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF COMPUTER APPLICATION (B.C.A)** to be submitted as final semester project as part of our curriculum.

TABLE OF CONTENTS

Contents

1.0 Introduction

The Banking Management System is a complex software solution developed to help banks and financial institutions manage and optimise their operations. With ever-increasing client demands and the complexity of banking processes, an efficient and integrated system is critical for the seamless operation of banking operations. The following section provides a quick overview of the Banking Management System.

1.1 Background

A bank is a type of financial institution that accepts deposits, disburses loans, clears checks, pays interest at set rates, and frequently serves as a middleman in financial transactions. Additionally, it offers its clients other financial services.

Earlier, banking operations relied heavily on paper-based documentation, manual record-keeping, and physical branch visits for transactions. However, with the emergence of computer technology, banks began exploring ways to leverage technology to improve their operations.

The advent of computer networks, database systems, and software applications enabled the automation of various banking processes. Banks started adopting core banking systems to handle their daily needs. These early systems laid the foundation for the more sophisticated banking management systems we see today.

Over time, advancements in computing power, data storage, and networking capabilities facilitated the development of more comprehensive and integrated banking management systems. Today, banking management systems have become vital tools for banks to streamline their operations, improve customer service, and achieve operational efficiency.

1.2 Problem Statement

The current banking system is slow because every task is carried out by a person, making it unfair to compare a computer's processing speed to that of a human. Since there will be more transactions as there are more clients, the complexity of the system will expand as well. Everything must be logged into a file for future reference, which is difficult in the case of an offline banking system.

Every day, millions of transactions are carried out by hundreds of banks, and thousands of people use the banking system in their daily lives. We know that as the number of users rises, we will need more banks and personnel, which will result in more manual labour and an increase in dangerous and less safe bank deposits. A sophisticated computerised banking system would eliminate the need for more branches, minimise labour requirements, and allow for the automatic storage of the vast majority of data in the banking server.

With the banking management system in place, any virtual transaction can be completed by a user at any time, any place, and within the shortest amount of time possible.

Flaws of the existing banking system:

- Less security of customer and bank information.
- Demands additional manpower and physical labour.
- It will take much longer time and extra resources to do all the manual entering and editing.
- There is no disaster protection for paper documents.
- There is no information backup.

Optimization of the proposed banking management system:

- Increased security by integrating a layer of authentication and authorization.
- Requires minimal manpower.
- Ease the data entry and editing concerns.
- Minimal focus on resources necessary for the maintenance of data.
- Comparatively reliable, adaptable and efficient.
- There will be an automatic backup mechanism in place for the client and bank information, ensuring that the data is protected from many types of disasters.

1.3 Project Scope

The scope of a Banking Management System is extensive, encompassing various aspects of banking operations and providing a comprehensive solution for efficient management.

Following are some key components that fall within the scope of a banking system:

a) **Creating New Bank Accounts:** Customers can use the application to open 2 types of accounts: Savings Accounts and Business Accounts. It alleviates the need for the customer to physically visit the bank to open/use these accounts.
b) **Transaction Processing:** The system enables seamless and secure processing of various banking transactions, such as deposits, withdrawals, fund transfers, payments. It ensures accurate and efficient transaction handling while maintaining transaction history for record-keeping purposes.
c) **Loan Management:** The system provides functionalities for loan mechanism. The customer can apply for 3 different types of loan i.e., Home Loan, Personal Loan, Gold Loan.
d) **Back-Office Operations:** The system supports back-office operations, including customer management, employee management, account management, analytics, document repository, application processing, loan ledgers, etc. It automates routine tasks, reduces manual errors, and improves operational efficiency.

1.4 Objective

When it comes to handling money or valuable assets, it eventually becomes a critical issue for the service provider, the client, and their degree of trustworthiness. The bank management system comes into the picture here as it is one of the most intricate systems that cover all of the aforementioned duties.

This lessens the need for manual labour and ensures that automated operations are error-free because they can only operate in accordance with programming, as opposed to manual tasks where human error is always a possibility.

Because there won't be any physical files or data sheets to manage, the work will not only be easier but also completed at a much faster rate thanks to the platform and system. With increased customer transparency and quick service, it will naturally win the trust of customers. The information about the customer will be just a few clicks away.

Instead of waiting in long queues, the customer can benefit from the seamless transaction of the online banking system. The consumer can benefit from the smooth transactions of the online banking system where they have constant access to their bank accounts rather than standing in long queues. Additionally, because all transactions are processed quite swiftly, you can access your money more quickly.

The system is designed to integrate with existing banking infrastructure in the future, such as core banking systems, payment gateways, and third-party applications. It aims to provide a scalable and flexible platform that can adapt to changing business requirements, accommodate future growth, and integrate with emerging technologies.

1.5 Features

The following are the main components of the banking management system:

a) **Customer:** Customers are the bank's primary source of income. As a standard procedure, banks investigate potential clients to make sure they are trustworthy and credible individuals with whom to enter into relationships. When someone opens a bank account, begins making deposits, or uses another service, they are considered a customer. The account enables the client to utilise the services offered by the bank. Every customer has a unique account number, and the bank will use an email and password to identify the customer. Customers may have the following types of accounts:
   - Savings Accounts
   - Business Account

   Utilities of the customer:

   - Register with the bank
   - Login with email and password
   - Enter and upgrade personal information
   - Deposit & withdraw money
   - View their account balance
   - Apply for a loan
   - Transfer money and much more

b) **Employee:** A qualified employee will always be needed for the proper management of events. Employees are the foundation of any bank. The employee on the premises will handle any issues or difficulties faced by the customers. The employees will also guide and assist the customers in fulfilling their needs.

c) **Admin:** A bank administrator will have complete administrative rights and powers over the system. The bank's administrator can set restrictions and rules, grant access to services, and oversee numerous branches. A bank manager is responsible for overseeing all aspects of the business, including customer management, employee management, branch management, transaction management, loan application management, etc.

   Utilities of the admin:

   - Login with an identification number, email and password
   - Review transactions of the customers
   - Approve loan applications
   - Monitor the system

2.0 Tools & Technologies Used

The creation of the Banking Management System involved a wide array of tools and technologies that played an instrumental role in shaping the success of the system. The following section explores all the tools and technologies used in order to implement this project.

2.1 Languages Used

FRONT END

HTML, short for Hypertext Markup Language, is a cornerstone of the World Wide Web. It is a markup language that web developers use to create the structure and layout of web pages. HTML uses tags to define and organize the various elements within a webpage, such as headings, paragraphs, images, links, forms, and more. It uses a straightforward syntax that consists of opening and closing tags to enclose elements.

HTML is often used in conjunction with **CSS** (Cascading Style Sheets) and to create visually appealing websites. CSS is responsible for styling the elements defined in HTML. CSS allows web developers to separate the design and structure of a website, making it easier to maintain and update. With CSS, you can change the background colour of a page, adjust the size and font of text, create borders around images and much more.

JavaScript is a versatile programming language that adds interactivity and dynamic functionality to web pages. It is commonly used for client-side web development, meaning it runs directly in the web browser of the user, allowing websites to respond to user actions and provide a more engaging experience. It allows developers to create dynamic elements that can update in real-time, respond to user clicks, validate forms, display pop-ups, and much more.

Bootstrap is an open-source tool kit for developing with HTML, CSS and JS. It helps your quickly prototype your ideas or build entire apps with attractive CSS, responsive grid systems, extensive prebuilt components, etc.

BACK END

Spring Boot is a powerful framework that simplifies the process of building robust and scalable web applications. It's like a toolbox filled with helpful tools and features that make developing web applications easier and more efficient. It provides a set of preconfigured components and sensible defaults, allowing you to focus on building your application's unique features. It automates many tedious tasks that developers often encounter, such as setting up the web server, managing dependencies, and handling database connections.

JSP (Java Server Pages) is a technology that allows developers to create dynamic web pages using Java. It's like having the ability to embed Java code directly into your web pages, making them more interactive and flexible. With JSP, one can write Java code directly within the HTML pages to handle these dynamic aspects.

Java is a widely used programming language known for its versatility, reliability, and platform independence. It was first introduced by Sun Microsystems (now owned by Oracle) in the mid-1990s and has since become one of the most popular languages in the world. One of the key strengths of Java is its object-oriented programming (OOP) approach.

The **Java Mail API** is a powerful tool that allows Java developers to send, receive, and manipulate email messages programmatically. It provides a set of classes and methods that simplify the process of integrating email functionality into Java applications. The JavaMail API abstracts the underlying email protocols, such as SMTP (Simple Mail Transfer Protocol) for sending emails and POP3 (Post Office Protocol) or IMAP (Internet Message Access Protocol) for receiving emails.

MySQL is a widely used open-source relational database management system (RDBMS) that allows you to efficiently store, manage, and retrieve structured data. It provides a reliable and scalable solution for applications that require persistent storage of data. MySQL supports the SQL (Structured Query Language) language, which is a standard language for managing relational databases. SQL allows to create, modify, and query databases using commands such as SELECT, INSERT, UPDATE, and DELETE.

2.2 Tools Used

Visual Studio Code (VS Code) is a lightweight and versatile code editor developed by Microsoft. It's designed to provide a rich and productive environment for developers to write, debug, and manage their code. It is mainly used to produce front end content.

IntelliJ IDEA is a powerful and widely used integrated development environment (IDE) created by JetBrains. It is specifically designed to enhance developer productivity and streamline the software development process. IntelliJ IDEA provides a comprehensive set of tools and features to support various programming languages and frameworks. It is mainly used to work with Spring Boot.

MySQL Workbench is a powerful visual tool for database design, development, and administration. It provides a comprehensive and user-friendly interface for working with MySQL databases. MySQL Workbench is developed by Oracle and is available for Windows, macOS, and Linux. It is mainly used to carry out RDMBS operations.

WampServer is a popular software stack that allows developers to set up a local web development environment on their windows computers. It provides an all-in-one solution for running Apache, MySQL, and PHP, collectively known as the AMP stack, on a Windows operating system. It is mainly used to connect to the MySQL server.

hMailServer is a free and open-source mail server software designed for Windows operating systems. It provides a reliable and customizable solution for setting up and managing email servers. hMailServer allows you to host your own email domain, enabling you to send, receive, and manage email accounts within your organization or for personal use. It is mainly used to create company domain.

GitHub is a web-based platform that provides a centralized repository for version control and collaboration on software development projects. It offers a range of features and tools designed to streamline the development process and enable collaboration among developers. It is mainly used as a backup storage.

2.3 Hardware & Software Requirements

Hardware Requirement

Processor	A multi-core processor (e.g., Intel Core i5 or higher) with a clock speed of 2.5 GHz or more
Memory (RAM)	Minimum 8 GB RAM, although 16 GB or more is recommended.
Storage (ROM)	1TB SSD recommended but HDD can also be used

Software Requirement

Operating System	Windows, macOS, or Linux.
Java Development Kit	JDK 11 or JDK 17
Build Tool	Maven

3.0 Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a systematic approach used by software development teams to plan, design, build, test, and deploy software applications. It provides a structured framework for managing the entire process of software development from conception to delivery. The following section explains the SDLC process.

3.1 Overview of Incremental Model

The Incremental Model is an iterative and incremental software development approach that breaks down the software development process into smaller, more manageable increments. It involves delivering functional subsets of the software in successive iterations, with each increment building upon the previous one. This model allows for early and frequent delivery of working software, providing benefits such as faster time to market and the ability to gather user feedback throughout the development process.

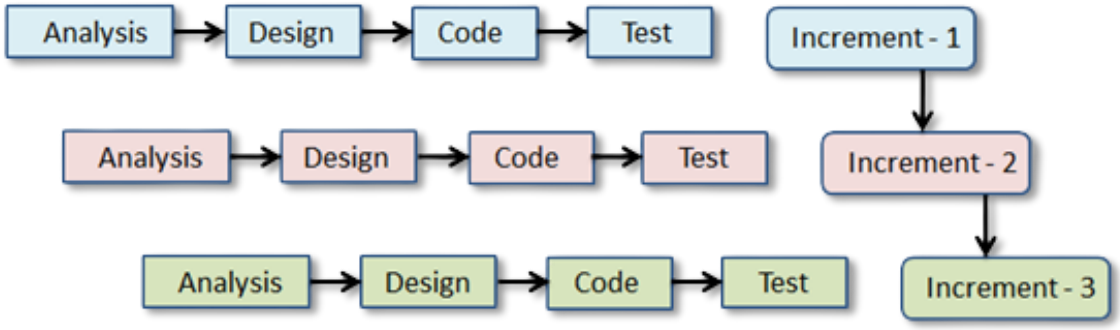
In the Incremental Model, the entire software development life cycle is divided into multiple increments or iterations. Each increment consists of requirements gathering, design, implementation, testing, and deployment activities, but focuses on delivering a specific set of functionalities. The development team completes these activities for each increment before moving on to the next one.

The Incremental Model is particularly useful in projects where requirements are not fully known or may evolve over time. It allows for flexibility and adaptability to changing requirements, while still providing early deliverables and opportunities for feedback. However, it requires effective planning, coordination, and collaboration among team members to ensure smooth integration of the increments and maintain overall system integrity.

Advantages of Incremental Model:

- Errors are easy to be recognized.
- Easier to test and debug.
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

3.2 Phases of Incremental Model



Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

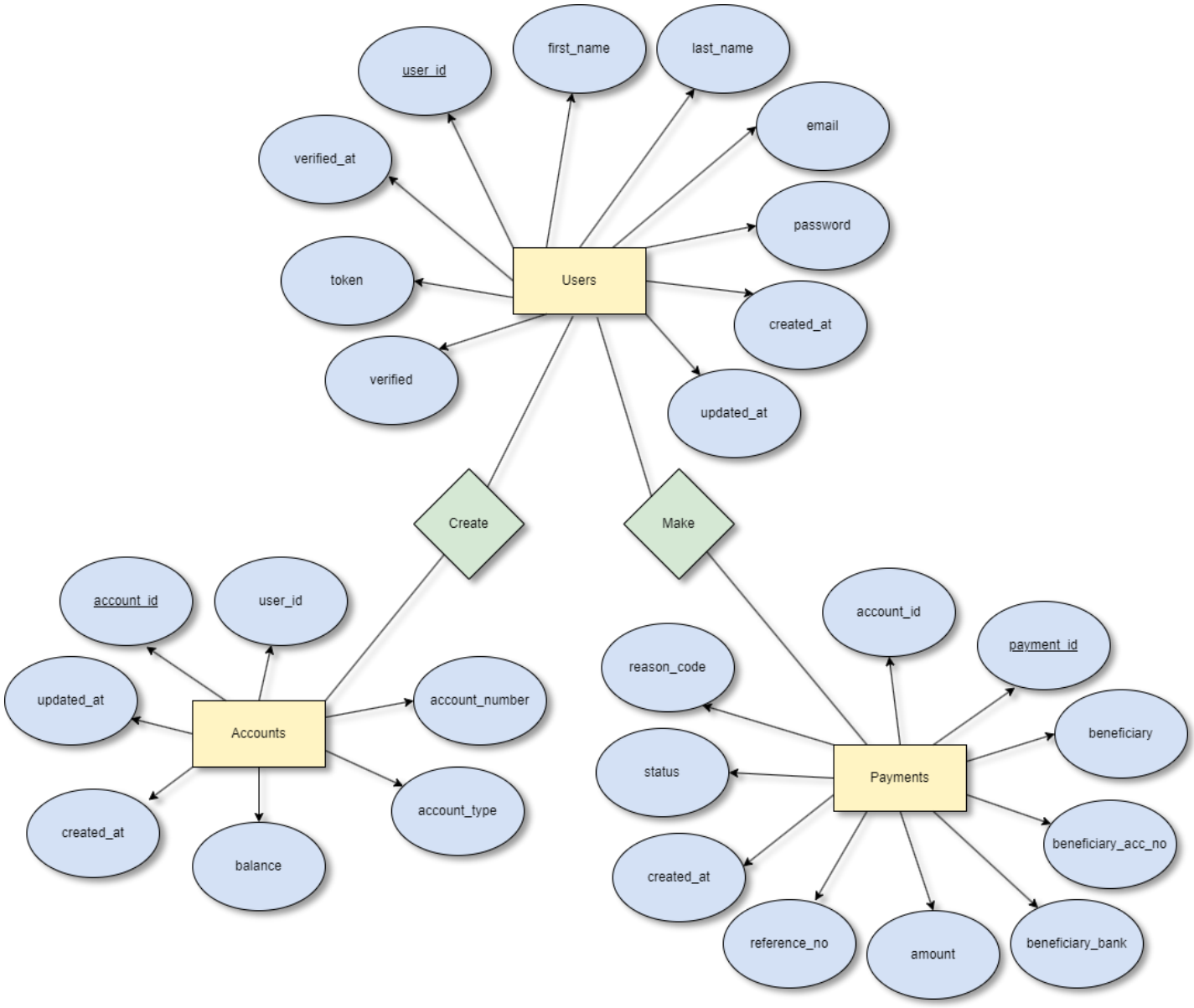
Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4.0 Design Phase

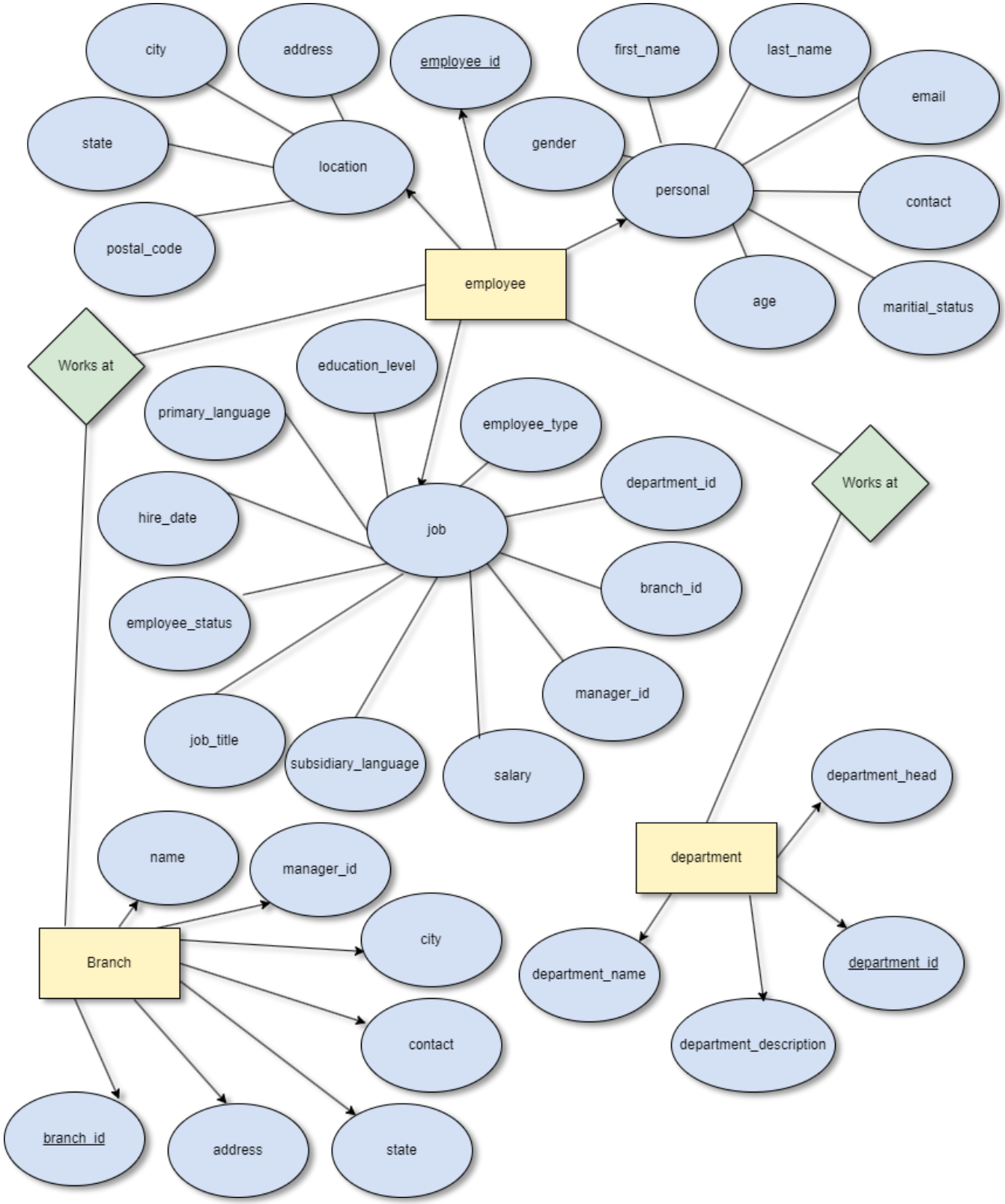
The design phase is a crucial stage in the software development life cycle where the overall structure and architecture of the software system are planned and defined. It involves translating the requirements gathered in the previous phases into a detailed blueprint for the software solution. The following section visualizes the schema of the system.

4.1 Entity Relationship Model

Customer side ERM

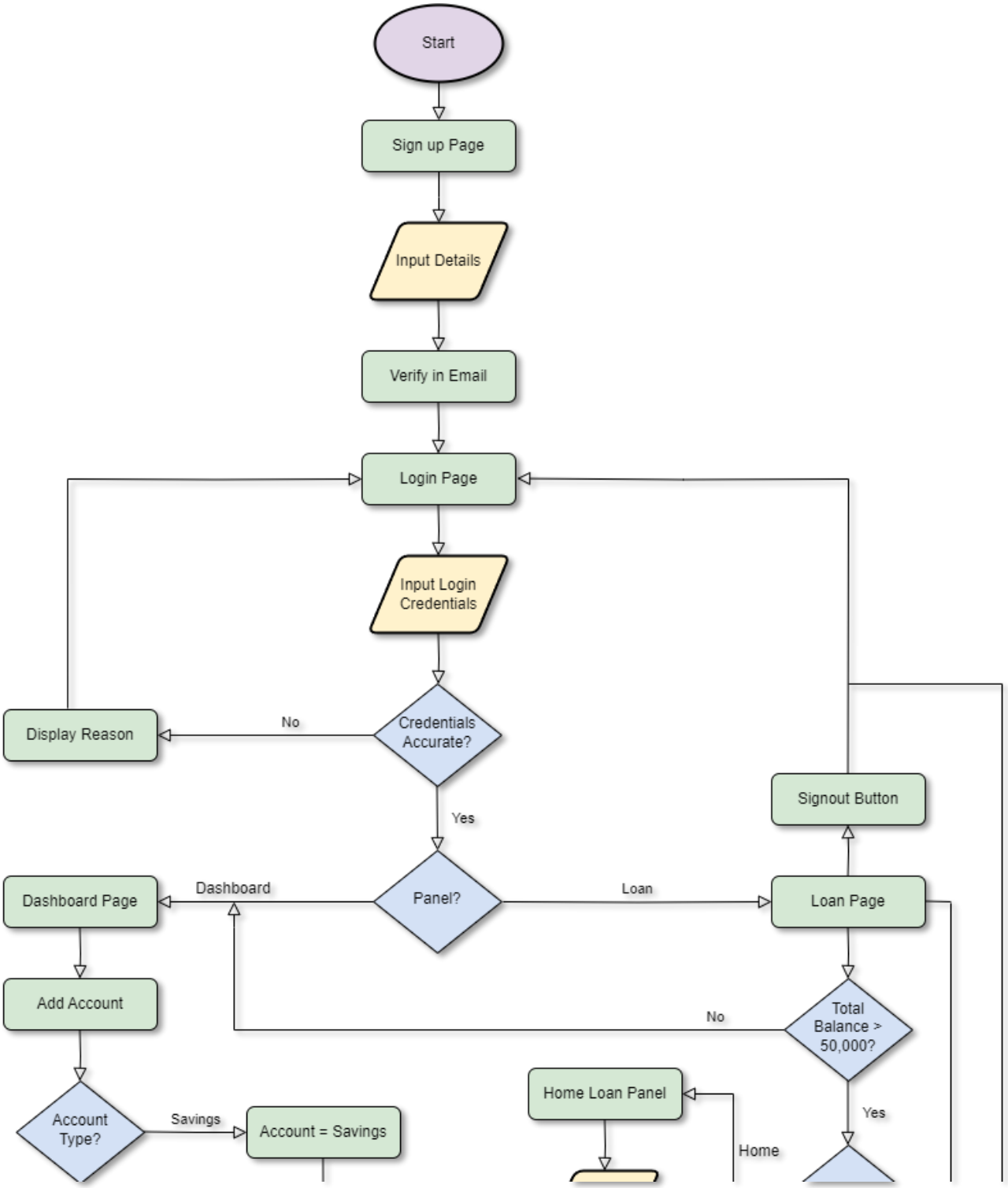


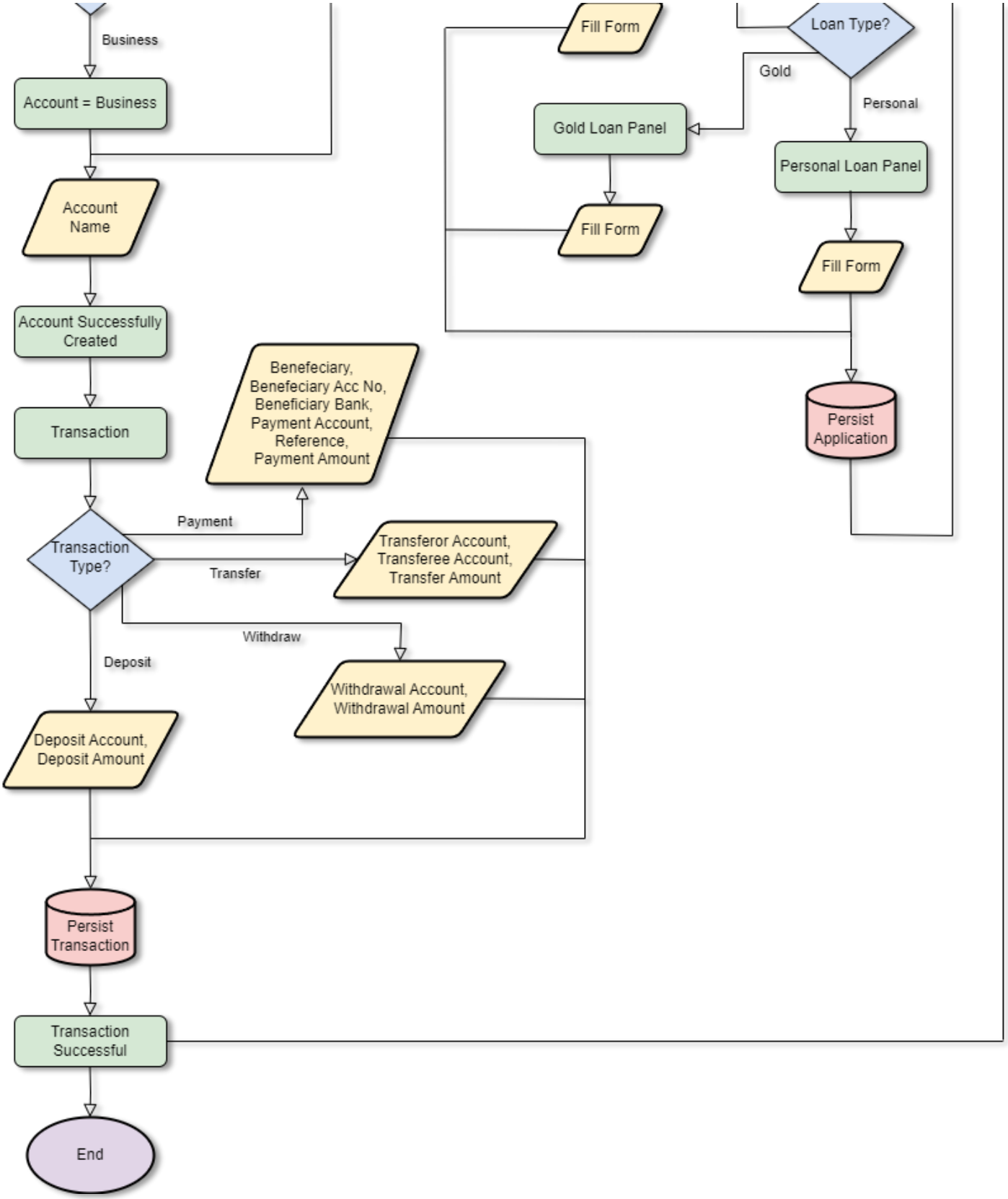
Admin Side ERM



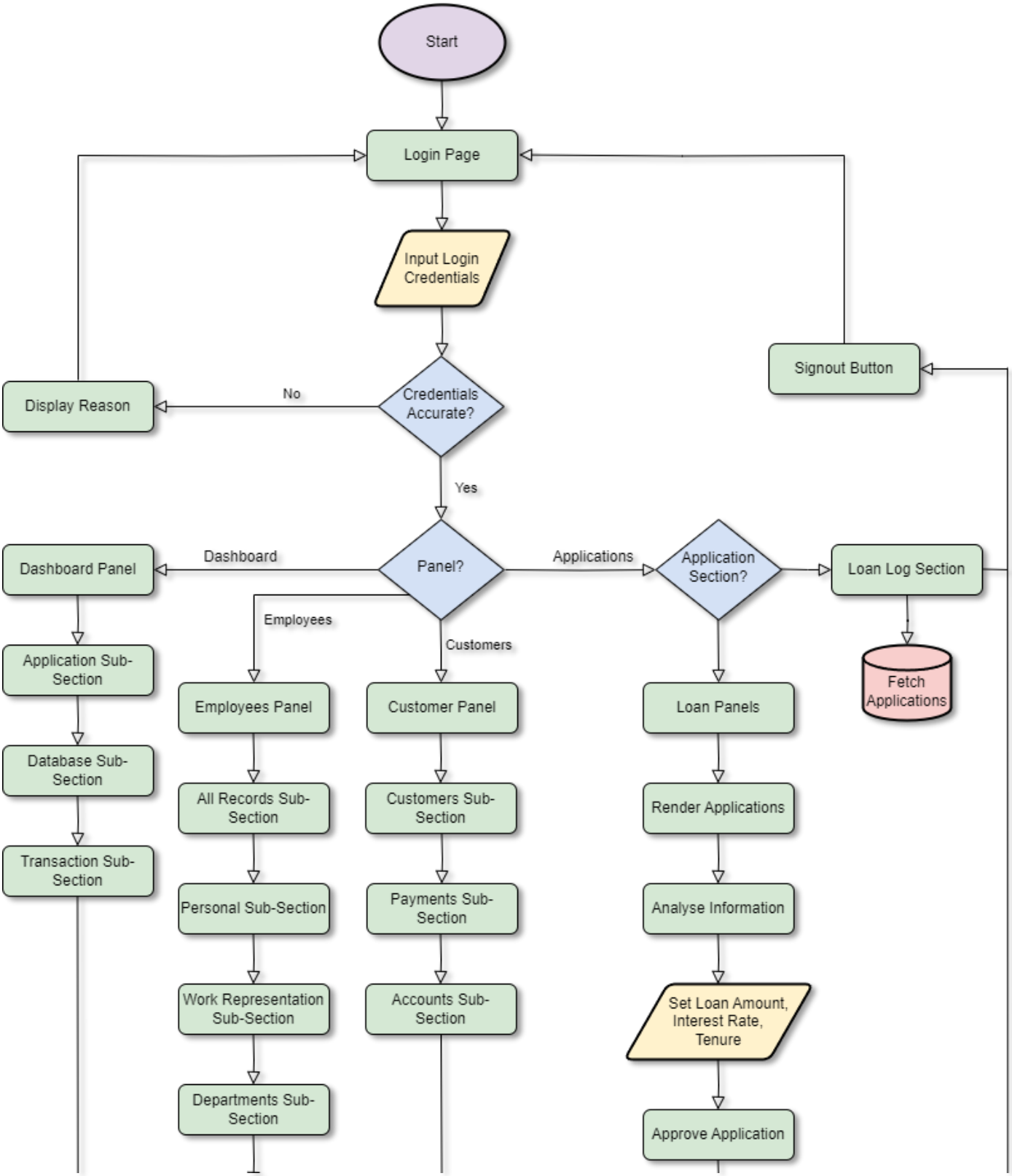
4.2 Flowchart

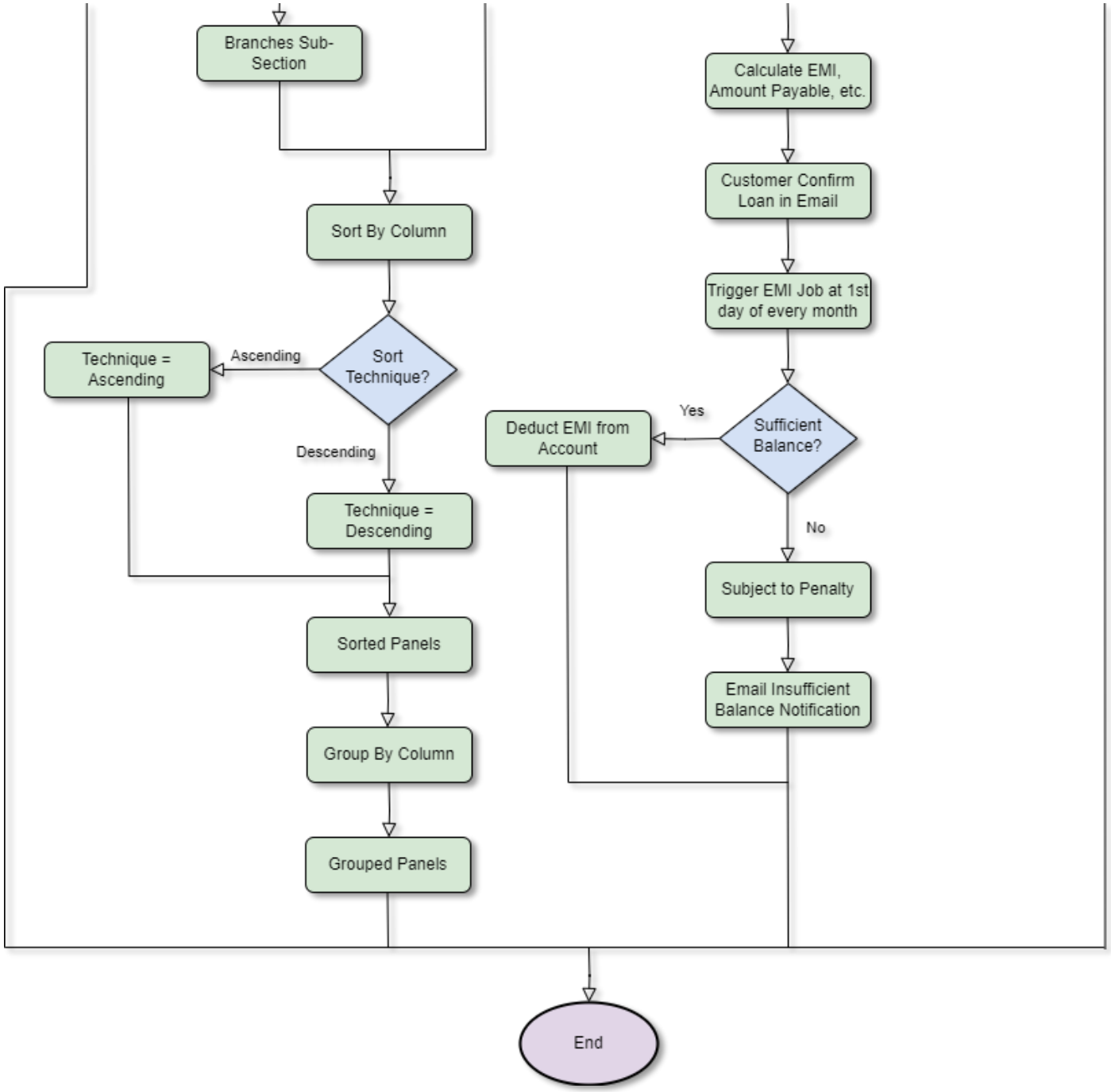
Customer Side Flowchart




Business
Account = Business
Account Name
Account Successfully Created
Transaction
Transaction Type?
Payment
Benefeciary, Benefeciary Acc No, Beneficiary Bank, Payment Account, Reference, Payment Amount
Transfer
Transferor Account, Transferee Account, Transfer Amount
Withdraw
Withdrawal Account, Withdrawal Amount
Deposit
Deposit Account, Deposit Amount
Persist Transaction
Transaction Successful
End
Fill Form
Loan Type?
Gold
Personal
Gold Loan Panel
Personal Loan Panel
Fill Form
Fill Form
Persist Application

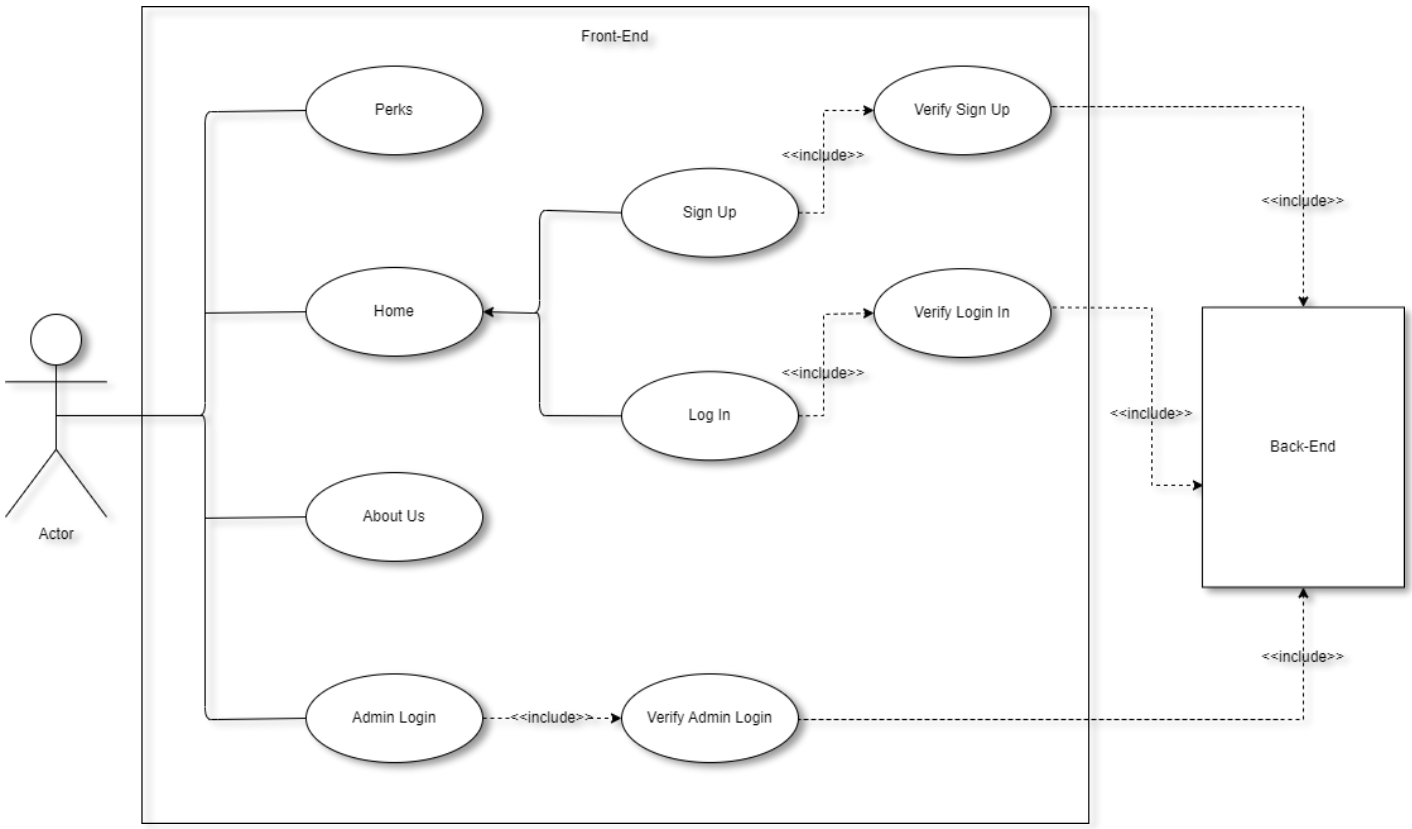
Admin Side Flowchart



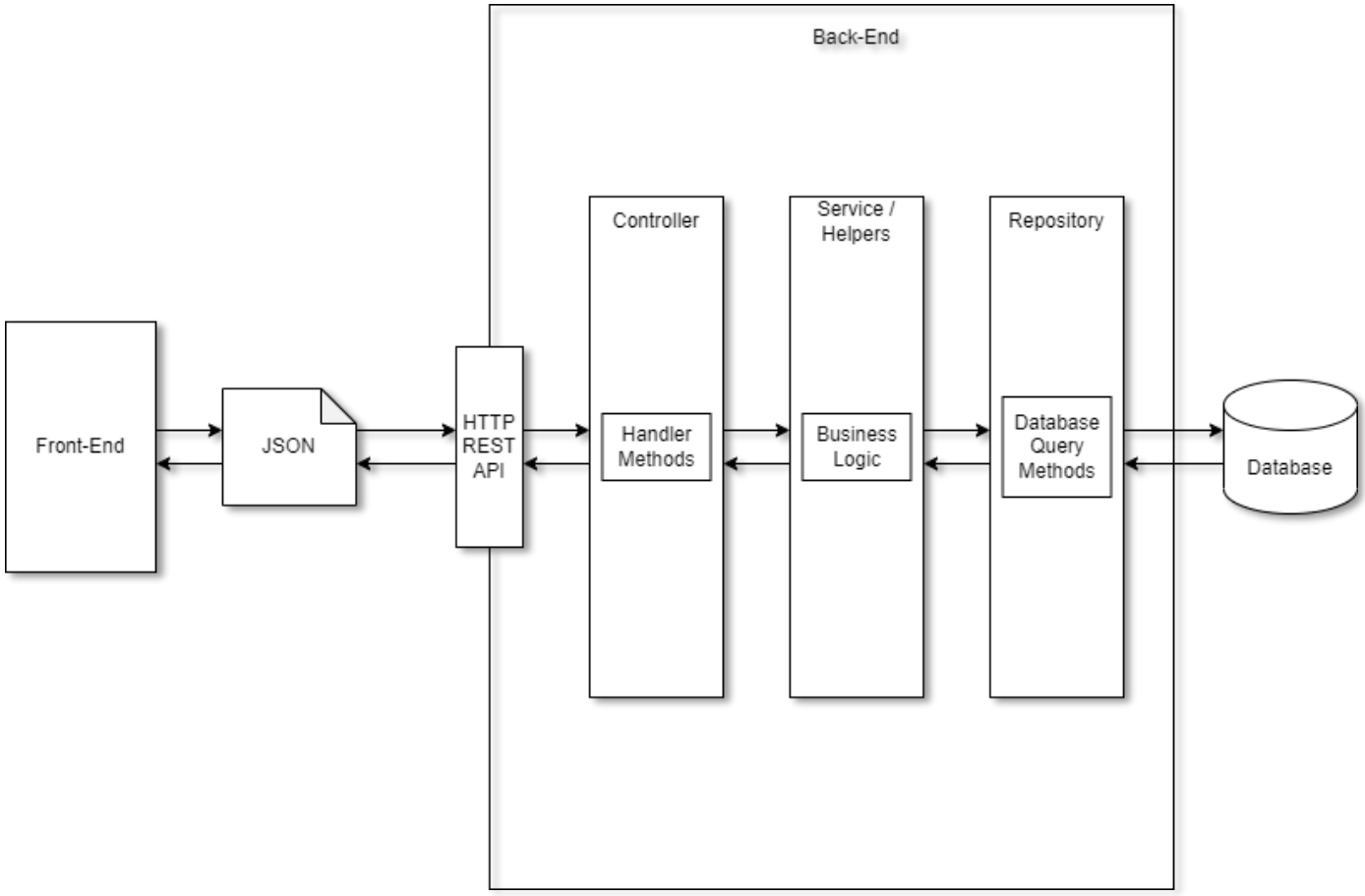

Branches Sub-Section
Sort By Column
Sort Technique?
Ascending
Technique = Ascending
Descending
Technique = Descending
Sorted Panels
Group By Column
Grouped Panels
Calculate EMI, Amount Payable, etc.
Customer Confirm Loan in Email
Trigger EMI Job at 1st day of every month
Sufficient Balance?
Yes
Deduct EMI from Account
No
Subject to Penalty
Email Insufficient Balance Notification
End

4.3 Use Case Diagram

Front-End

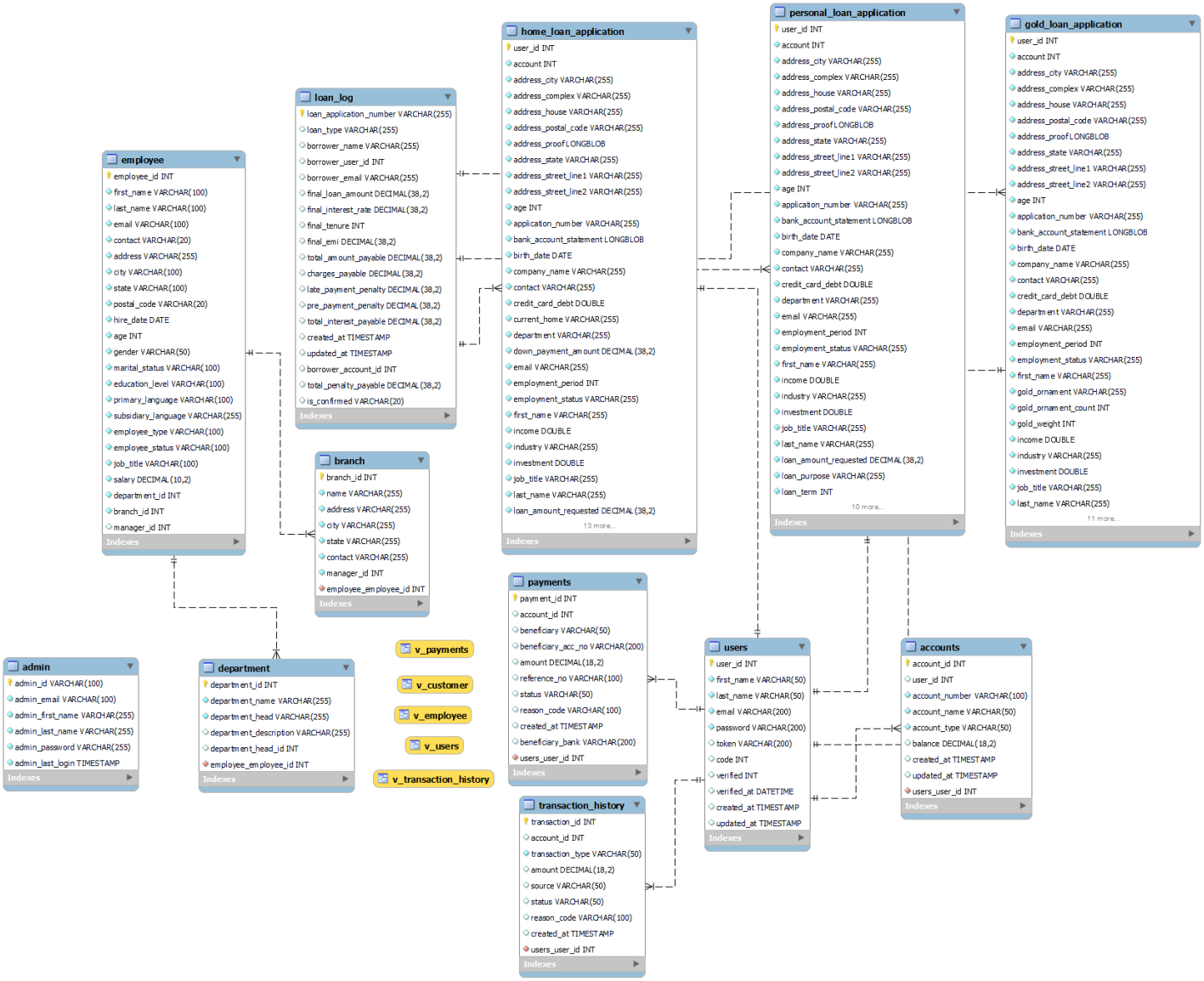


Back-End

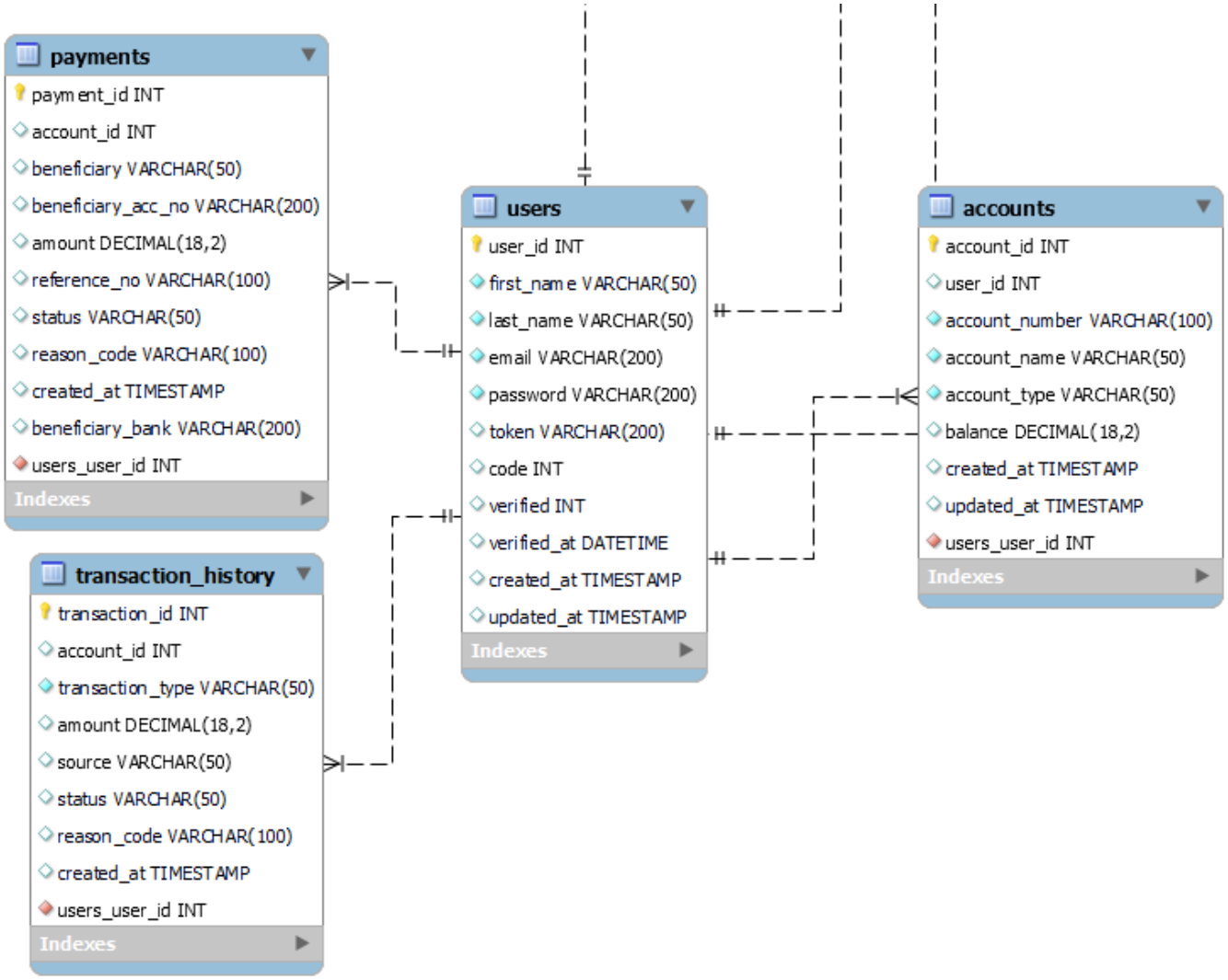

Back-End
Controller
Service / Helpers
Repository
Front-End
JSON
HTTP REST API
Handler Methods
Business Logic
Database Query Methods
Database

4.4 Entity Relationship Diagram

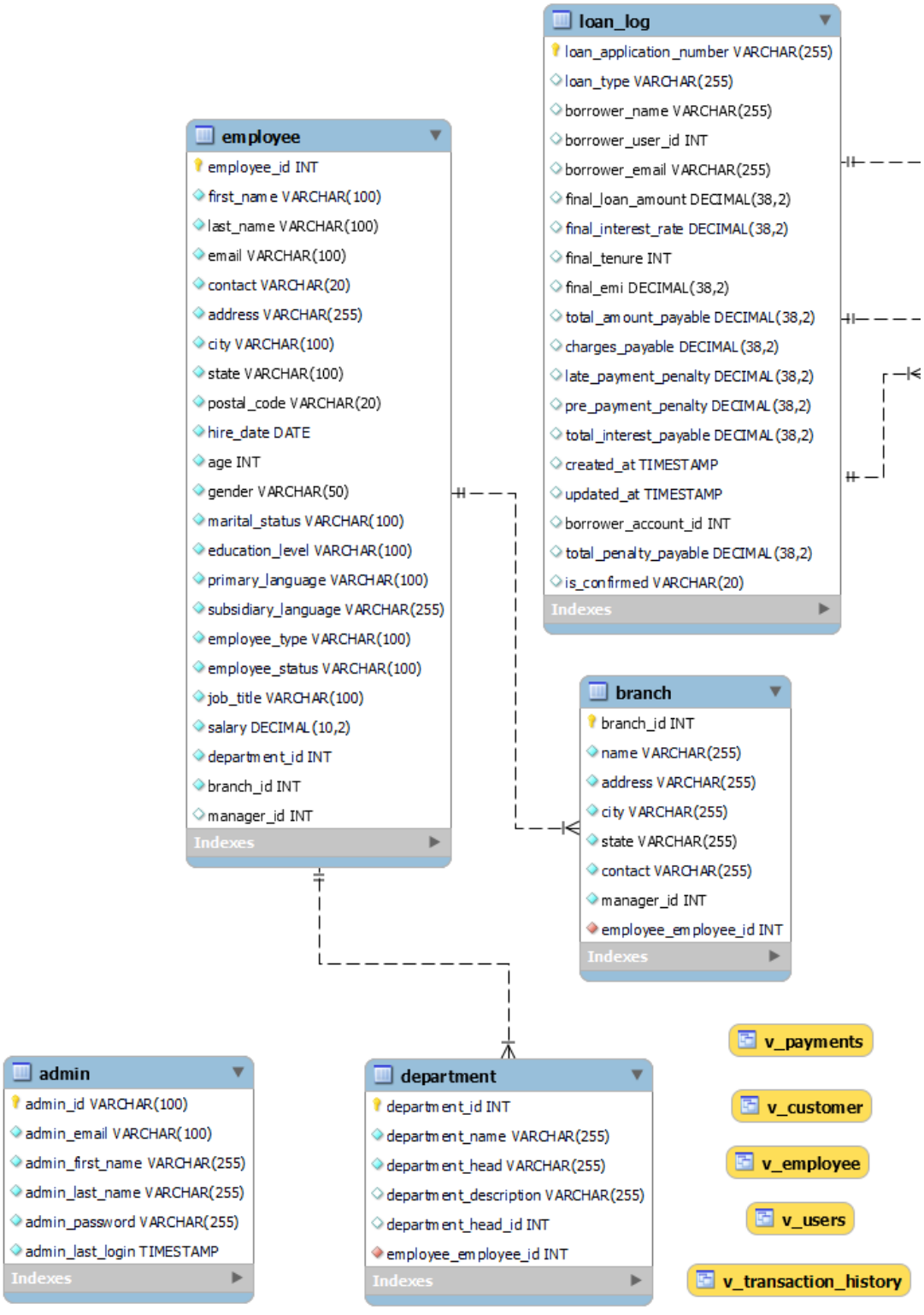
Overview



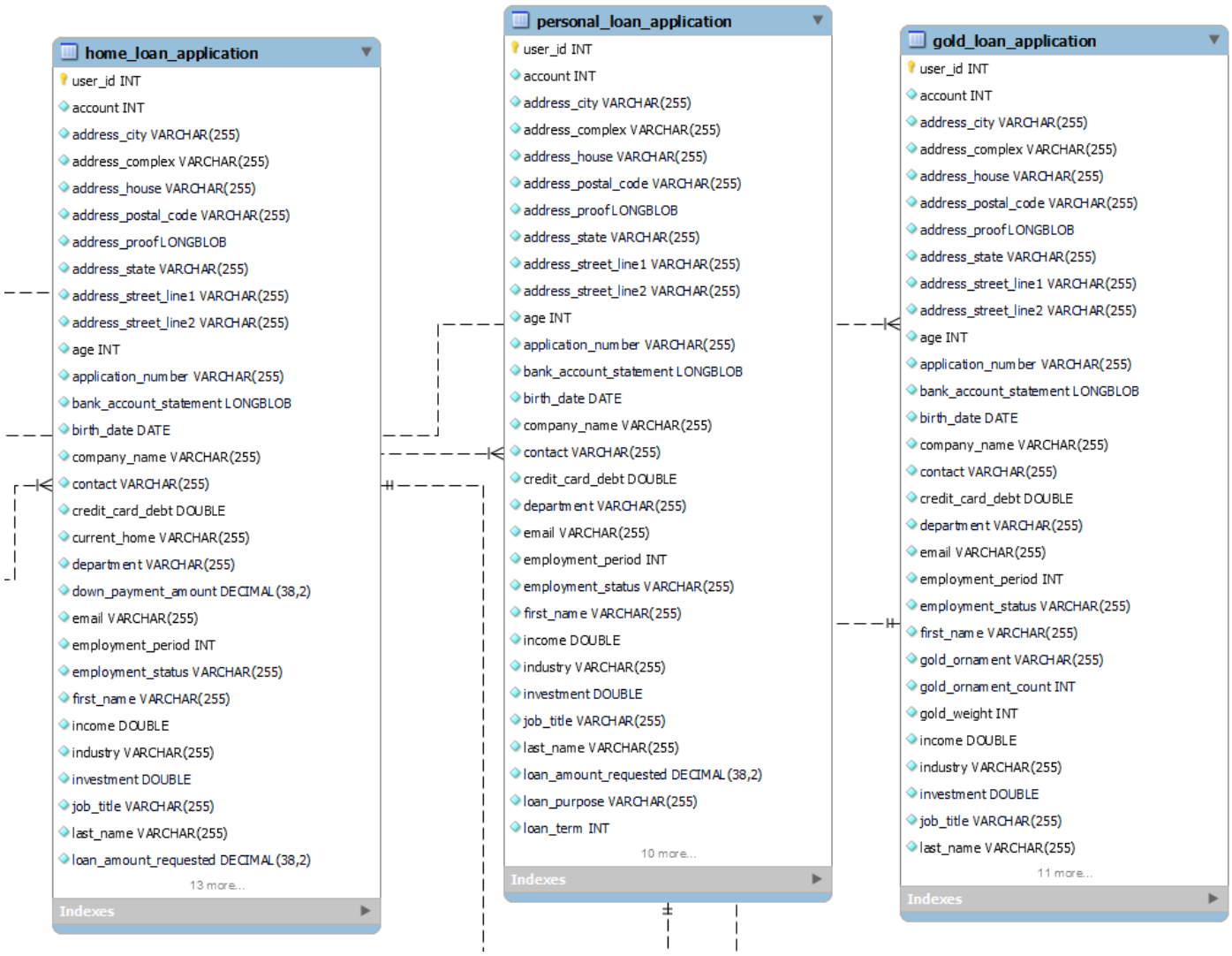
User Section



Employee Section



Loan Section



4.4 Data Dictionary

Accounts Table

Field Name	Data Type	Extra
account_id	INT	Primary Key, Auto Increment
user_id	INT	Foreign Key
account_number	VARCHAR(100)	-
account_name	VARCHAR(50)	-
account_type	VARCHAR(50)	-
balance	DECIMAL(18,2)	Default = 0.00
created_at	TIMESTAMP	-
updated_at	TIMESTAMP	Current Time

Admin Table

Field Name	Data Type	Extra
admin_id	VARCHAR(100)	Primary Key
admin_email	VARCHAR(100)	-
admin_first_name	VARCHAR(255)	-
admin_last_name	VARCHAR(255)	-
admin_password	VARCHAR(255)	-
admin_last_login	TIMESTAMP	-

Branch Table

Field Name	Data Type	Extra
branch_id	INT	Primary Key, Auto Increment
name	VARCHAR(255)	-
address	VARCHAR(255)	-
city	VARCHAR(255)	-
state	VARCHAR(255)	-
contact	VARCHAR(255)	-
manager_id	INT	Foreign Key

Admin Table

Field Name	Data Type	Extra
department_id	INT	Primary Key, Auto Increment
department_name	VARCHAR(255)	-
department_head	VARCHAR(255)	-
department_description	VARCHAR(255)	-
department_head_id	INT	Foreign Key

Employee Table

Field Name	Data Type	Extra
employee_id	INT	Primary Key, Auto Increment
first_name	VARCHAR(100)	-
last_name	VARCHAR(100)	-
email	VARCHAR(100)	-
contact	VARCHAR(20)	-
address	VARCHAR(255)	-
city	VARCHAR(100)	-
state	VARCHAR(100)	-
postal_code	VARCHAR(20)	-
hire_date	DATE	-
age	INT	-
gender	VARCHAR(50)	-
marital_status	VARCHAR(100)	-
education_level	VARCHAR(100)	-
primary_language	VARCHAR(100)	-
subsidiary_language	VARCHAR(255)	-
employee_type	VARCHAR(100)	-
employee_status	VARCHAR(100)	-
job_title	VARCHAR(100)	-
salary	DECIMAL(10,2)	-
department_id	INT	Foreign Key
branch_id	INT	Foreign Key
manager_id	INT	Foreign Key

Gold Loan Application Table

Field Name	Data Type	Extra
user_id	INT	Primary Key
account	INT	-
address_city	VARCHAR(255)	-
address_complex	VARCHAR(255)	-
address_house	VARCHAR(255)	-
address_postal_code	VARCHAR(255)	-
address_proof	LONGBLOB	-
address_state	VARCHAR(255)	-
address_street_line1	VARCHAR(255)	-
address_street_line2	VARCHAR(255)	-
age	INT	-
application_number	VARCHAR(255)	Unique
bank_account_statement	LONGBLOB	-
birth_date	DATE	-
company_name	VARCHAR(255)	-
contact	VARCHAR(255)	-
credit_card_debt	DOUBLE	-
department	VARCHAR(255)	-
email	VARCHAR(255)	-
employment_period	INT	-
employment_status	VARCHAR(255)	-
first_name	VARCHAR(255)	-
gold_ornament	VARCHAR(255)	-
gold_weight	INT	-
income	INT	-
industry	DOUBLE	-
investment	VARCHAR(255)	-
job_title	DOUBLE	-
last_name	VARCHAR(255)	-
loan_purpose	VARCHAR(255)	-
marital_status	VARCHAR(255)	-
proof_of_identity	VARCHAR(255)	-
religion	LONGBLOB	-
salary	VARCHAR(255)	-

salary_slip	DOUBLE	-
work_mode	LONGBLOB	-
approved	VARCHAR(255)	no
confirm	VARCHAR(50)	no

Home Loan Application Table

Field Name	Data Type	Extra
user_id	INT	Primary Key
account	INT	-
address_city	VARCHAR(255)	-
address_complex	VARCHAR(255)	-
address_house	VARCHAR(255)	-
address_postal_code	VARCHAR(255)	-
address_proof	LONGBLOB	-
address_state	VARCHAR(255)	-
address_street_line1	VARCHAR(255)	-
address_street_line2	VARCHAR(255)	-
age	INT	-
application_number	VARCHAR(255)	Unique
bank_account_statement	LONGBLOB	-
birth_date	DATE	-
company_name	VARCHAR(255)	-
contact	VARCHAR(255)	-
credit_card_debt	DOUBLE	-
current_home	VARCHAR(255)	-
department	VARCHAR(255)	-
down_payment_amount	DECIMAL(38,2)	-
email	VARCHAR(255)	-
employment_period	INT	-
employment_status	VARCHAR(255)	-
first_name	VARCHAR(255)	-
income	DOUBLE	-
industry	VARCHAR(255)	-
investment	DOUBLE	-
job_title	VARCHAR(255)	-
last_name	VARCHAR(255)	-

loan_amount_requested	DECIMAL(38,2)	-
marital_status	VARCHAR(255)	-
motto_purchase	VARCHAR(255)	-
proof_of_identity	LONGBLOB	-
purchase_price	DECIMAL(38,2)	-
religion	VARCHAR(255)	-
salary	DOUBLE	-
salary_slip	LONGBLOB	-
sell_current	VARCHAR(255)	-
work_mode	VARCHAR(255)	-
approved	VARCHAR(50)	no
confirm	VARCHAR(50)	no

Loan Log Table

Field Name	Data Type	Extra
loan_application_number	VARCHAR(255)	Primary Key
loan_type	VARCHAR(255)	-
borrower_name	VARCHAR(255)	-
borrower_user_id	INT	-
borrower_email	VARCHAR(255)	-
final_loan_amount	DECIMAL(38,2)	-
final_interest_rate	DECIMAL(38,2)	-
final_tenure	INT	-
final_emi	DECIMAL(38,2)	-
total_amount_payable	DECIMAL(38,2)	-
charges_payable	DECIMAL(38,2)	-
late_payment_penalty	DECIMAL(38,2)	-
pre_payment_penalty	DECIMAL(38,2)	-
total_interest_payable	DECIMAL(38,2)	-
created_at	TIMESTAMP	Current Time
updated_at	TIMESTAMP	Current Time
borrower_account_id	INT	-
total_penalty_payable	DECIMAL(38,2)	0.00
is_confirmed	VARCHAR(20)	no

Payments Table

Field Name	Data Type	Extra
payment_id	INT	Primary Key, Auto Increment
account_id	INT	Foreign Key
beneficiary	VARCHAR(50)	-
beneficiary_acc_no	VARCHAR(200)	-
amount	DECIMAL(18,2)	-
reference_no	VARCHAR(100)	-
status	VARCHAR(50)	-
reason_code	VARCHAR(100)	-
created_at	TIMESTAMP	-
beneficiary_bank	VARCHAR(200)	-

Personal Loan Application Table

Field Name	Data Type	Extra
user_id	INT	Primary Key
account	INT	-
address_city	VARCHAR(255)	-
address_complex	VARCHAR(255)	-
address_house	VARCHAR(255)	-
address_postal_code	VARCHAR(255)	-
address_proof	LONGBLOB	-
address_state	VARCHAR(255)	-
address_street_line1	VARCHAR(255)	-
address_street_line2	VARCHAR(255)	-
age	INT	-
application_number	VARCHAR(255)	Unique
bank_account_statement	LONGBLOB	-
birth_date	DATE	-
company_name	VARCHAR(255)	-
contact	VARCHAR(255)	-
credit_card_debt	DOUBLE	-
department	VARCHAR(255)	-
email	VARCHAR(255)	-
employment_period	INT	-

employment_status	VARCHAR(255)	-
first_name	VARCHAR(255)	-
income	DOUBLE	-
industry	VARCHAR(255)	-
investment	DOUBLE	-
job_title	VARCHAR(255)	-
last_name	VARCHAR(255)	-
loan_amount_requested	DECIMAL(38,2)	-
loan_purpose	VARCHAR(255)	-
loan_term	INT	-
marital_status	VARCHAR(255)	-
proof_of_identity	LONGBLOB	-
religion	VARCHAR(255)	-
salary	DOUBLE	-
salary_slip	LONGBLOB	-
work_mode	VARCHAR(255)	-
approved	VARCHAR(50)	no
confirm	VARCHAR(50)	no

Transaction History Table

Field Name	Data Type	Extra
transaction_id	INT	Primary Key, Auto Increment
account_id	INT	Foreign Key
transaction_type	VARCHAR(50)	-
amount	DECIMAL(18,2)	-
source	VARCHAR(50)	-
status	VARCHAR(50)	-
reason_code	VARCHAR(100)	-
created_at	TIMESTAMP	-

Users Table

Field Name	Data Type	Extra
user_id	INT	Primary Key, Auto Increment
first_name	VARCHAR(50)	Foreign Key
last_name	VARCHAR(50)	-
email	VARCHAR(200)	Unique
password	VARCHAR(200)	-
token	VARCHAR(200)	-
code	INT	-
verified	INT	-
verified_at	DATETIME	-
created_at	TIMESTAMP	-
updated_at	TIMESTAMP	Current Time

Customer View

Field Name	Data Type	Extra
user_id	INT	-
name	VARCHAR(101)	-
email	VARCHAR(200)	-
created_at	TIMESTAMP	-
updated_at	TIMESTAMP	Current Time
verified_at	DATETIME	-
account_count	BIGINT	-

Employee View

Field Name	Data Type	Extra
employee_id	INT	-
Name	VARCHAR(201)	-
age	INT	-
job_title	VARCHAR(100)	-
hire_date	DATE	-
department_name	VARCHAR(255)	-
branch_name	VARCHAR(255)	-

Payment View

Field Name	Data Type	Extra
payment_id	INT	-
account_id	INT	-
user_id	INT	-
beneficiary	VARCHAR(50)	-
beneficiary_acc_no	VARCHAR(200)	-
beneficiary_bank	VARCHAR(200)	-
amount	DECIMAL(18,2)	-
status	VARCHAR(50)	-
reference_no	VARCHAR(100)	-
reason_code	VARCHAR(100)	-
created_at	TIMESTAMP	-

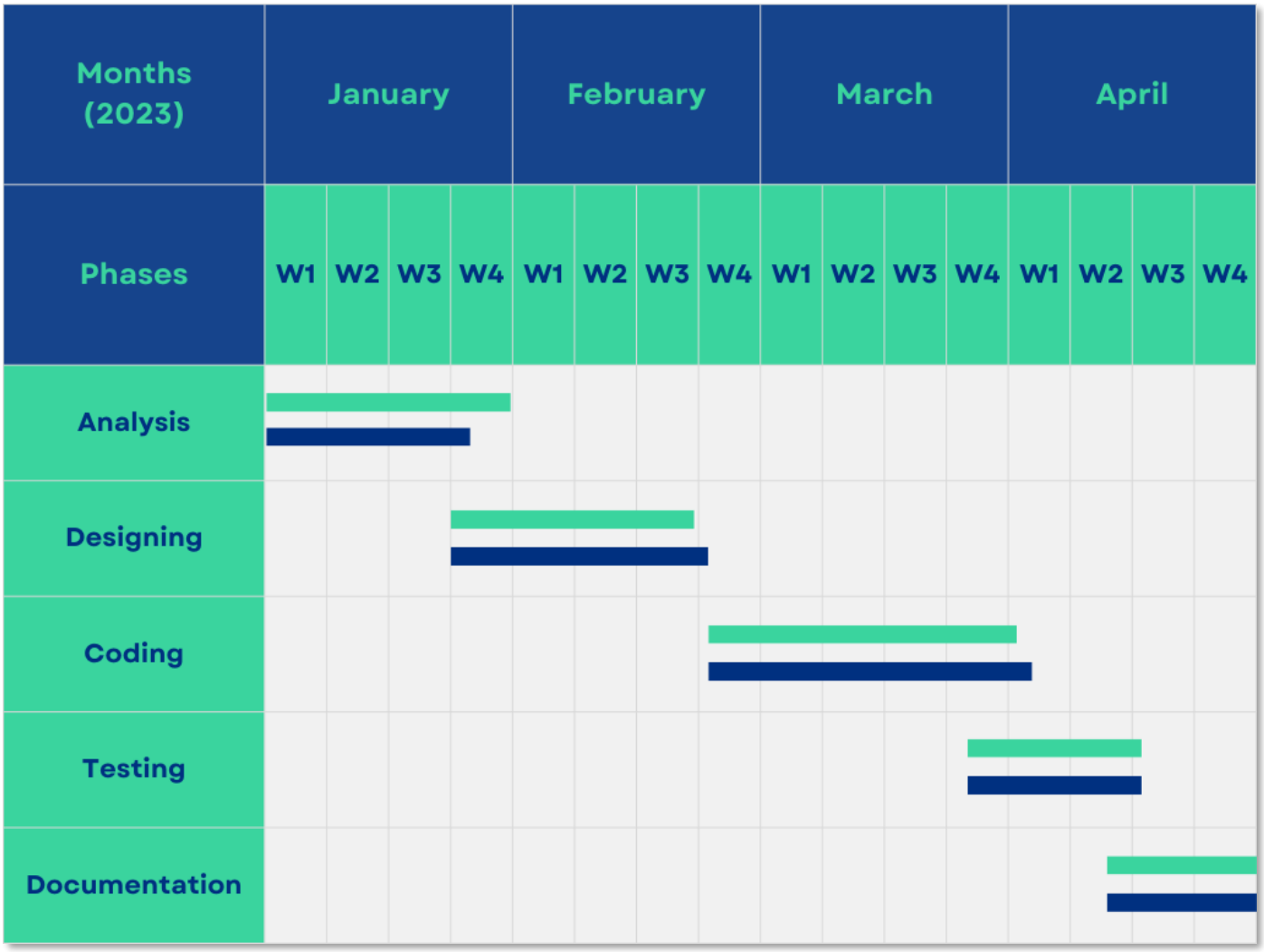
Transaction History View

Field Name	Data Type	Extra
transaction_id	INT	-
account_id	INT	-
user_id	INT	-
transaction_type	VARCHAR(50)	-
amount	DECIMAL(18,2)	-
source	VARCHAR(50)	-
status	VARCHAR(50)	-
reason_code	VARCHAR(100)	-
created_at	TIMESTAMP	-

Users View

Field Name	Data Type	Extra
user_id	int	-
name	varchar(101)	-
email	varchar(200)	-
account_count	bigint	-

4.5 Gantt Chart



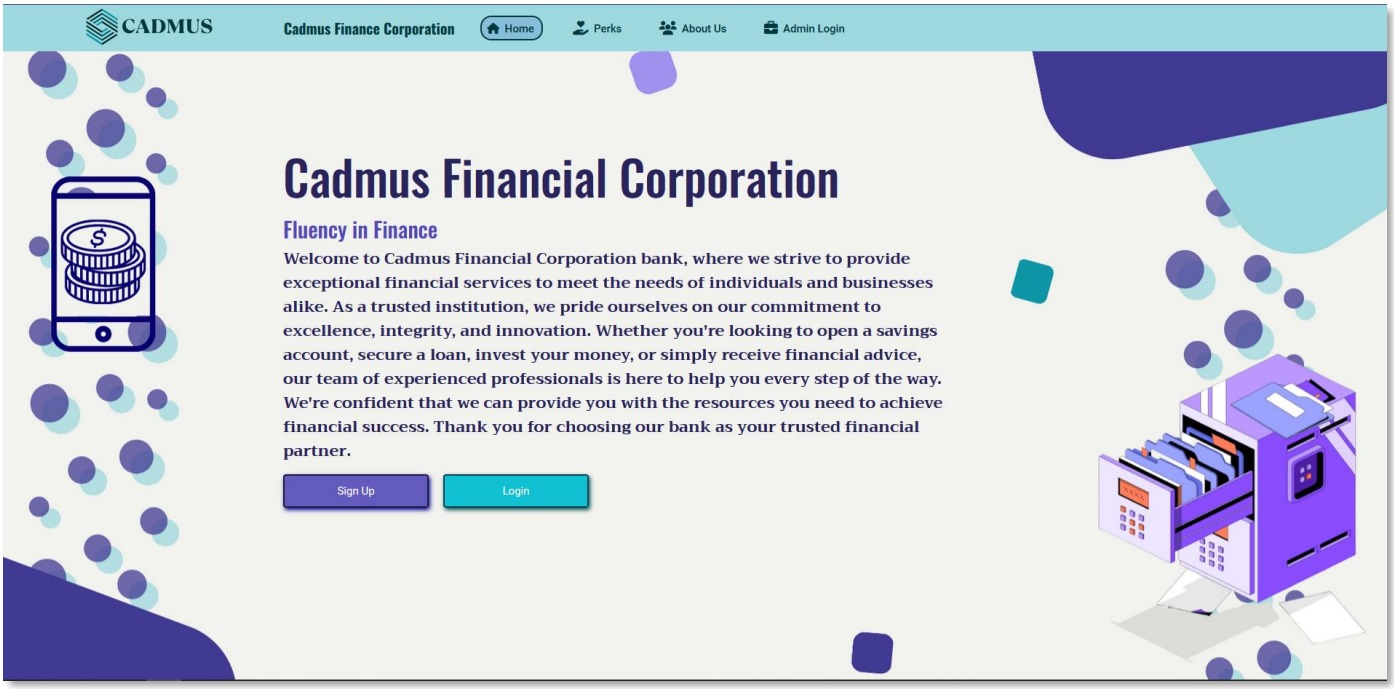
Graphical Symbols

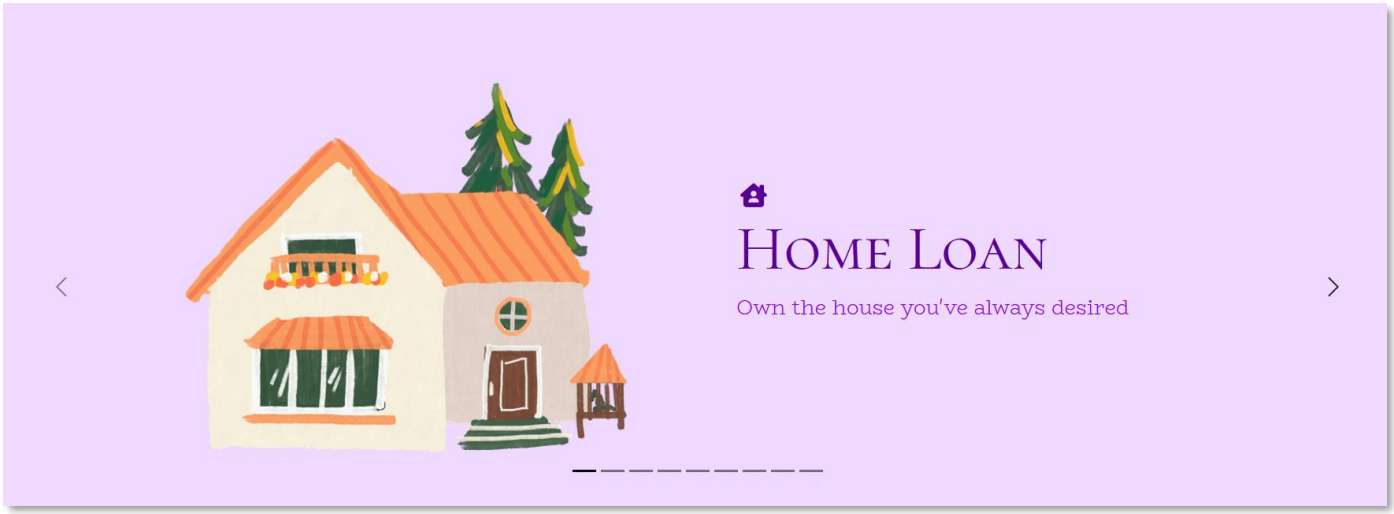
Symbol	Description
	Estimated Date
	Time Taken

5.0 Screen Dumps

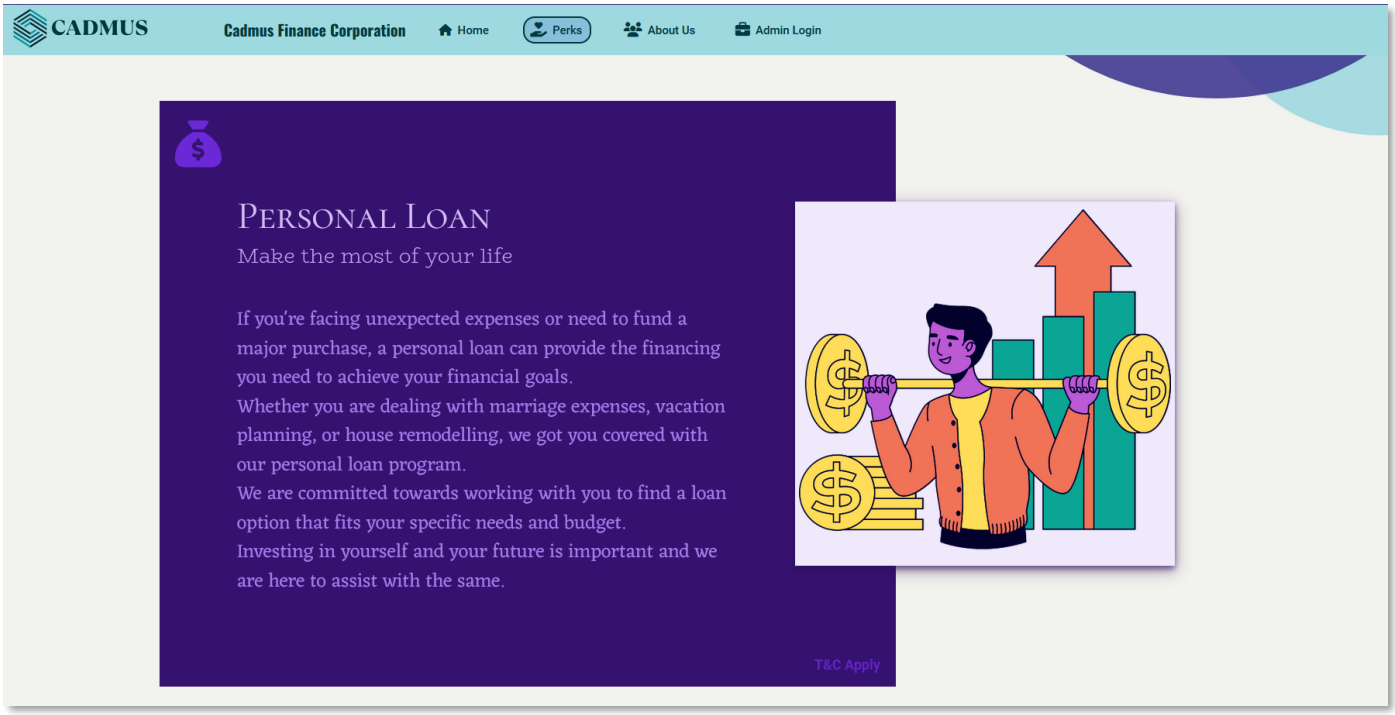
Screen Dumps are a valuable tool for capturing and documenting visual representations of computer screens, applications, etc. The following section displays snapshots of the Banking Management System as various screenshots.

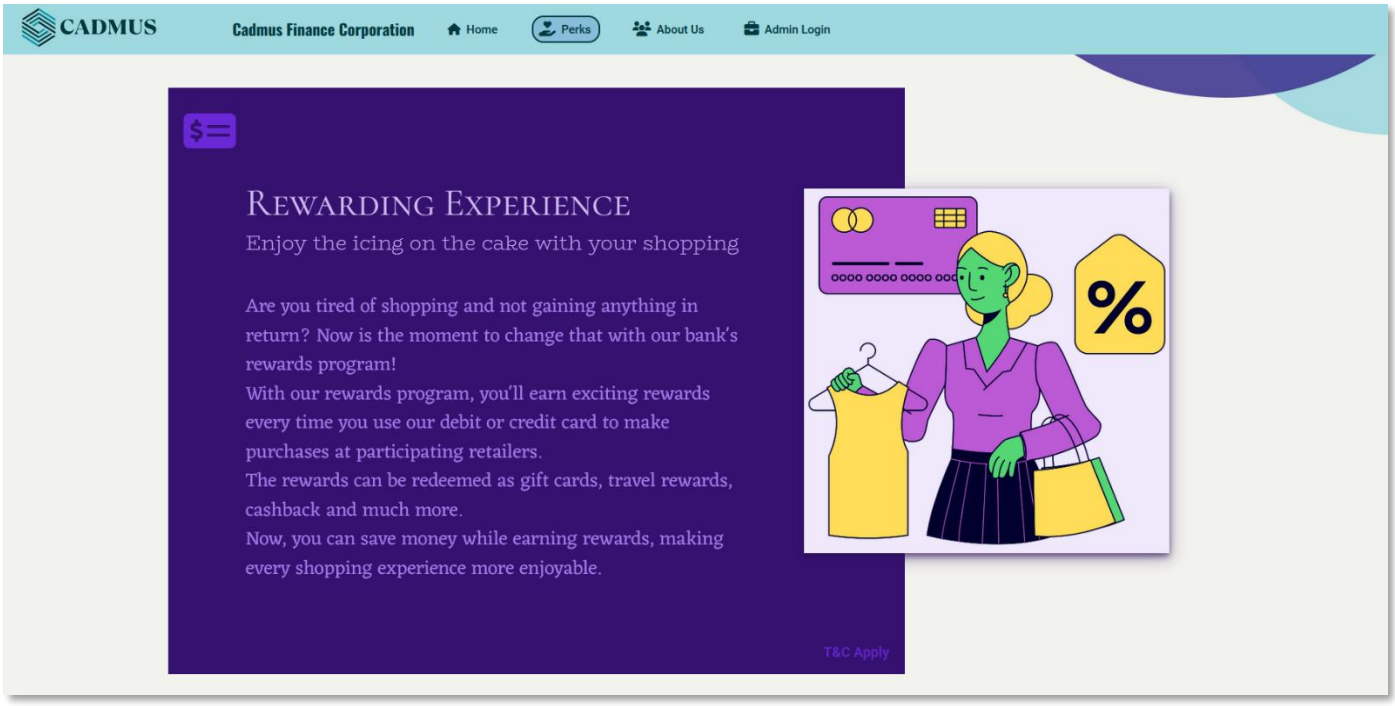
Landing Page



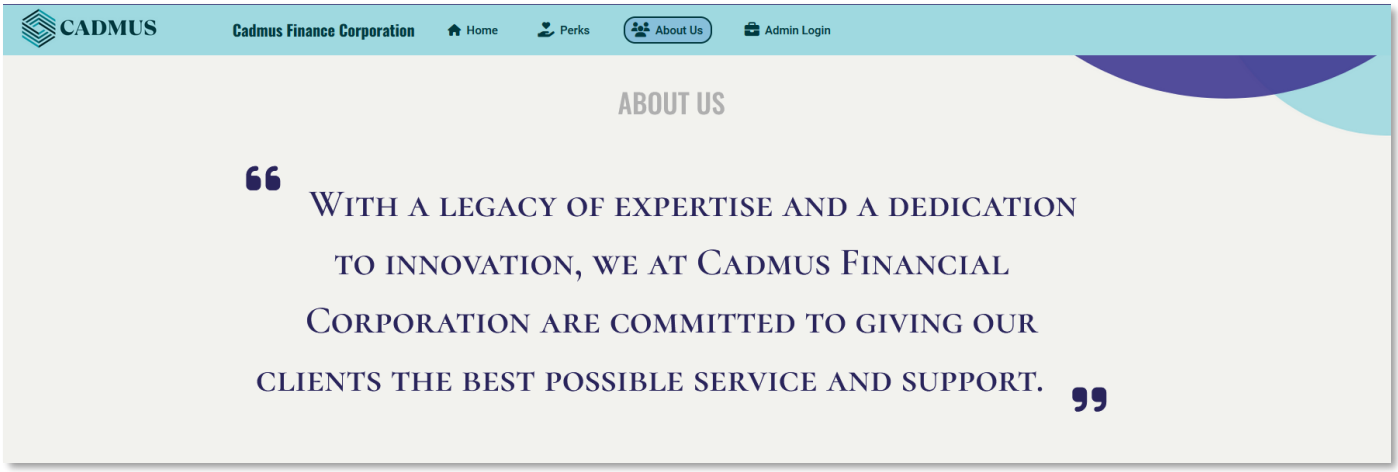


Perks Page


CADMUS
Cadmus Finance Corporation
Home
Perks
About Us
Admin Login
PERSONAL LOAN
Make the most of your life
If you're facing unexpected expenses or need to fund a major purchase, a personal loan can provide the financing you need to achieve your financial goals.
Whether you are dealing with marriage expenses, vacation planning, or house remodelling, we got you covered with our personal loan program.
We are committed towards working with you to find a loan option that fits your specific needs and budget.
Investing in yourself and your future is important and we are here to assist with the same.
T&C Apply


CADMUS
Cadmus Finance Corporation
Home
Perks
About Us
Admin Login
REWARDING EXPERIENCE
Enjoy the icing on the cake with your shopping
Are you tired of shopping and not gaining anything in return? Now is the moment to change that with our bank's rewards program!
With our rewards program, you'll earn exciting rewards every time you use our debit or credit card to make purchases at participating retailers.
The rewards can be redeemed as gift cards, travel rewards, cashback and much more.
Now, you can save money while earning rewards, making every shopping experience more enjoyable.
T&C Apply

About Us Page



History

Cadmus Bank was established in 1955 in a small town in the Suburbs of Mumbai City. The founders of the bank were a group of local businessmen who wanted to provide better financial services to the residents of their community. The bank started as a small operation with just a few employees and one branch location.

In the early years, Cadmus Bank focused on providing basic banking services such as current and savings accounts, loans, and mortgages. The bank's reputation for excellent customer service and reasonable rates spread fast throughout the neighbourhood, and it began to draw customers from nearby towns.

In the 1970s, the bank began to expand its operations by opening new branch locations in nearby cities. In order to adapt to the evolving requirements of its customers, the bank also provided new products and amenities such as credit cards and online banking.

By the 1980s, the bank had grown into a regional financial institution with dozens of branch locations across several states. The bank maintained its focus on providing personalised service and developing strong customer relationships.

Cadmus Bank weathered the financial crisis in the 1990s by taking a conservative approach to lending and risk management. The bank recovered from the recession as one of the industry's finest and most stable institutions.

Today, Cadmus Bank is a well-known financial institution with a reputation for offering exceptional customer service and a commitment to serving the communities where it operates. In a continuously changing financial landscape, the bank continues to innovate and develop new services and goods to meet the growing needs of its consumers.

Journey

1955	1960	1970	1980	1990	2000
Cadmus Bank Established	Embedded a Robust Reputation	Inaugurated Additional Branch Offices	Formed a Regional Financial Institution	Survived the Financial Crisis	Ambitious towards new innovations

Awards & Recognization

April 25th , 1991

Winner

Best Bank (Private Sector)

Cadmus Bank was named the Best Bank in the Private Sector by NDTV Profit Business.

December 18th , 1993

Winner

Fastest Growing Bank

Cadmus Bank was awarded as the Fastest Growing Bank by Polaris Finance.

July 04th , 1995

Bronze Medal

Digital Marketing Excellence

Cadmus Bank won bronze medal for Digital Marketing at the Digix Virtual Summit

November 23rd , 1996

Runner Up

Best Corporate Bank

Cadmus Bank was named Runner Up for Best Corporate Bank by Global Finance.

May 22nd , 1999

Winner

Consistent Performer

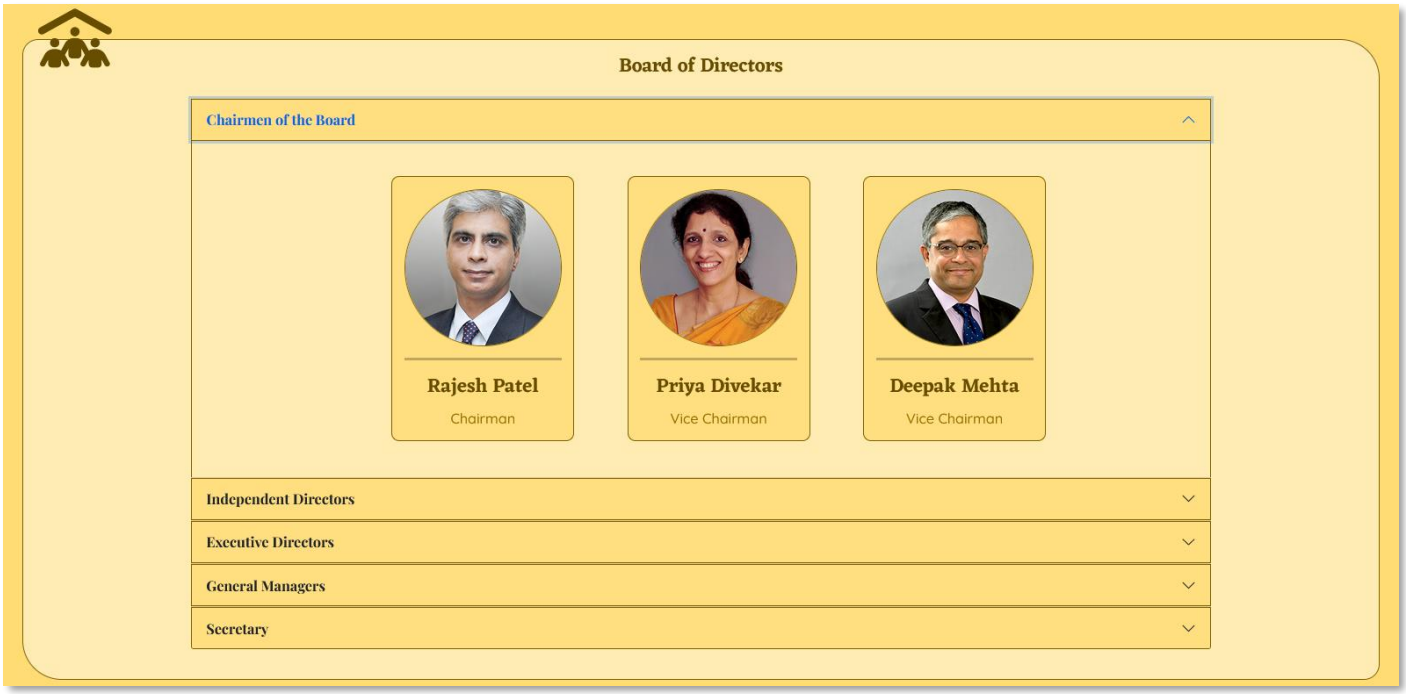
Cadmus Bank was awarded as Consistent Performer by Business Today & KPMG.

August 17th , 2005

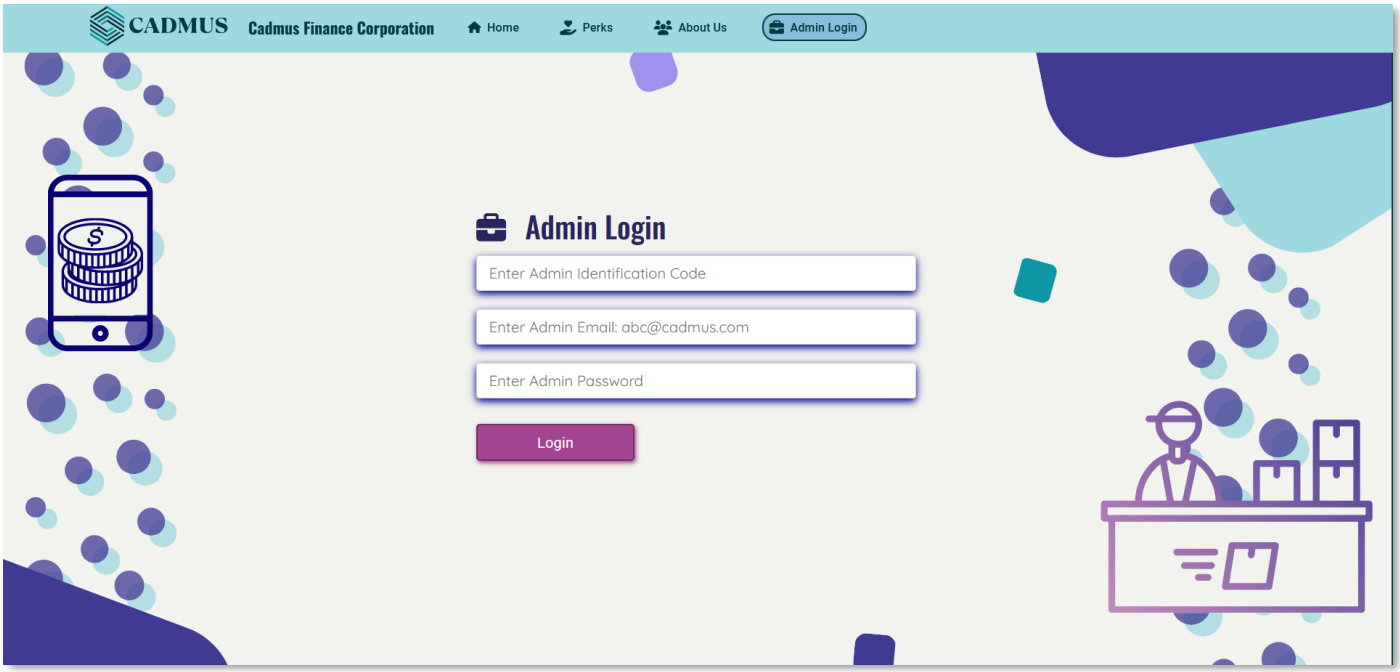
Runner Up

Most Trusted Bank

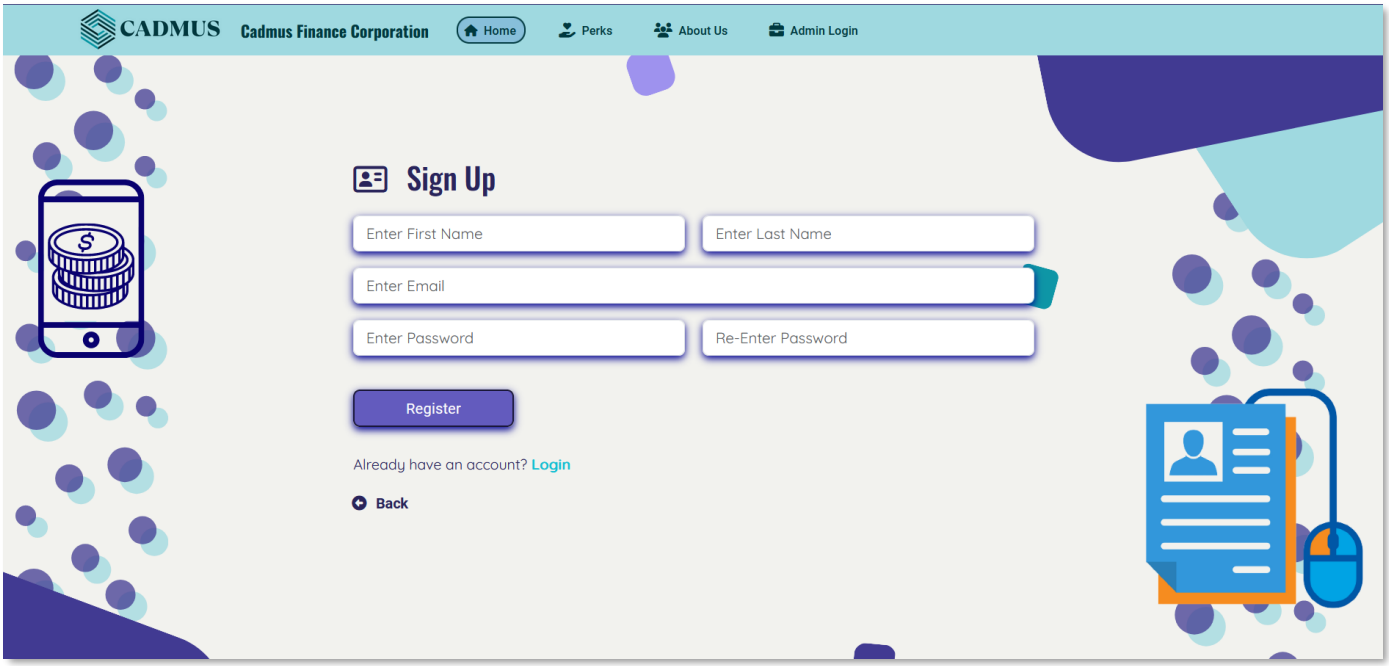
Cadmus Bank was named Runner Up for Most Trusted Bank by Brand Equity Finance.



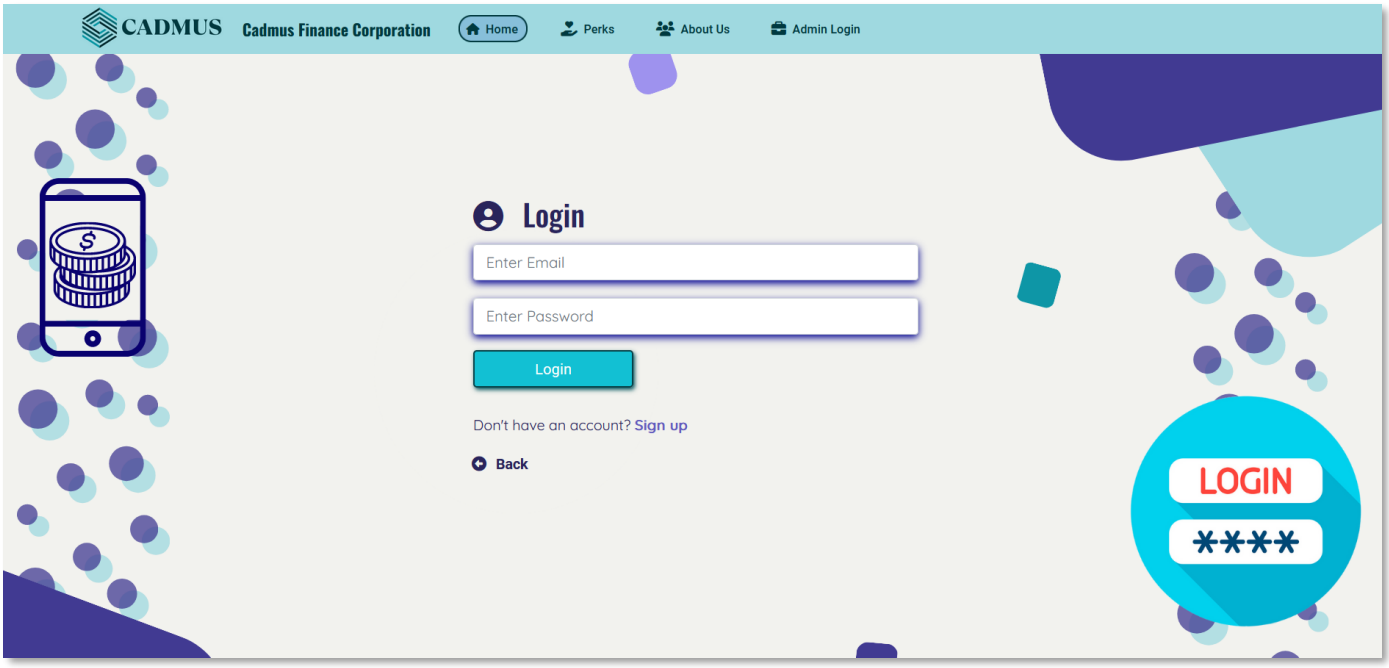
Admin Login Page



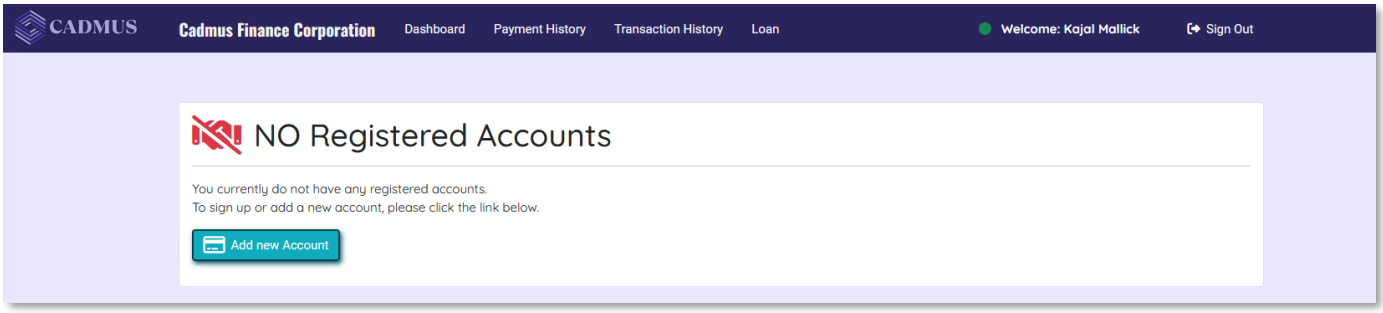
Sign Up Page



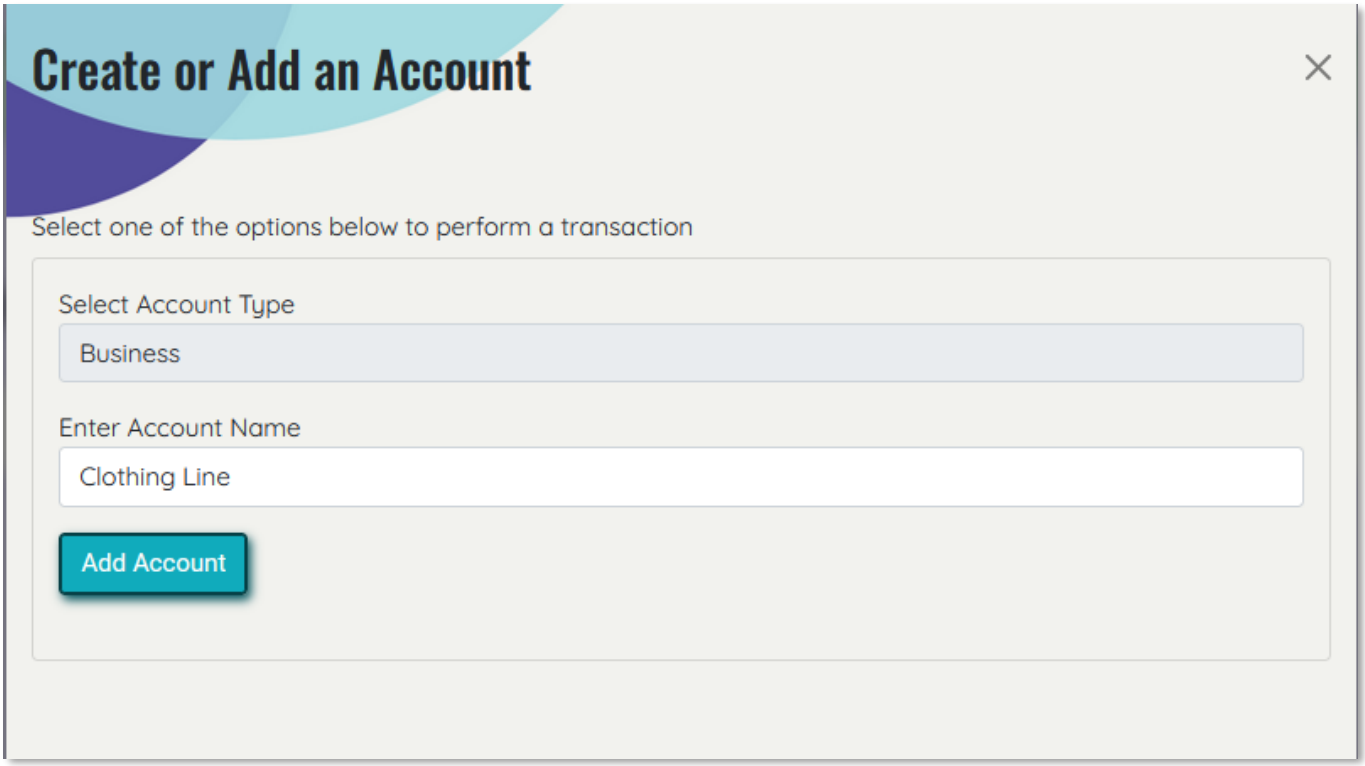
Login In Page



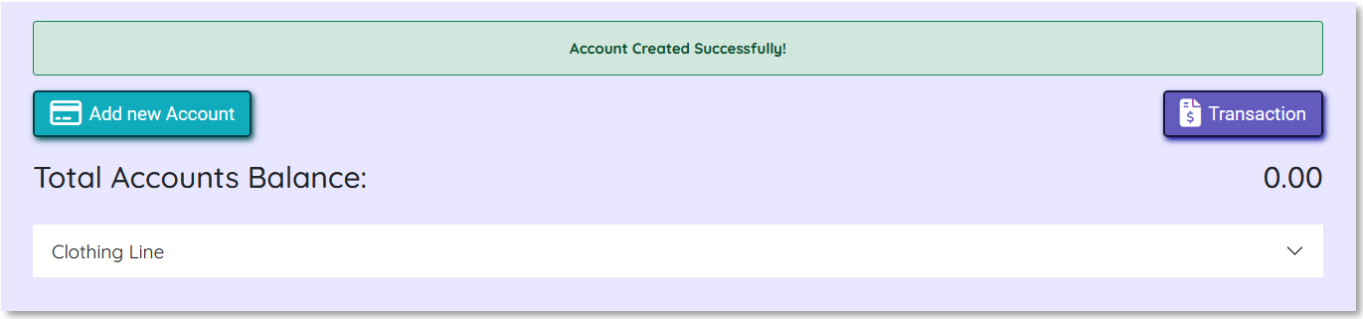
Customer Dashboard (No Account) Page



Create Account Offcanvas

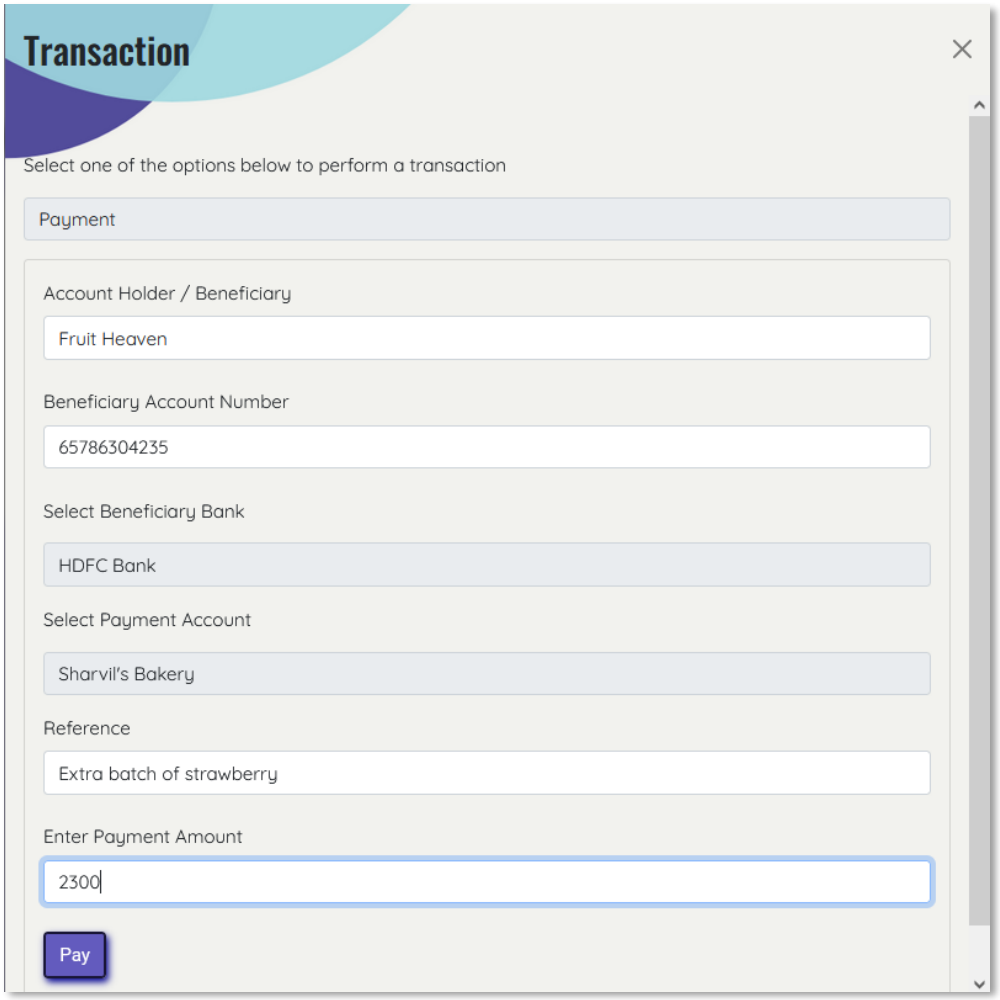


Customer Dashboard (Account Added) Page

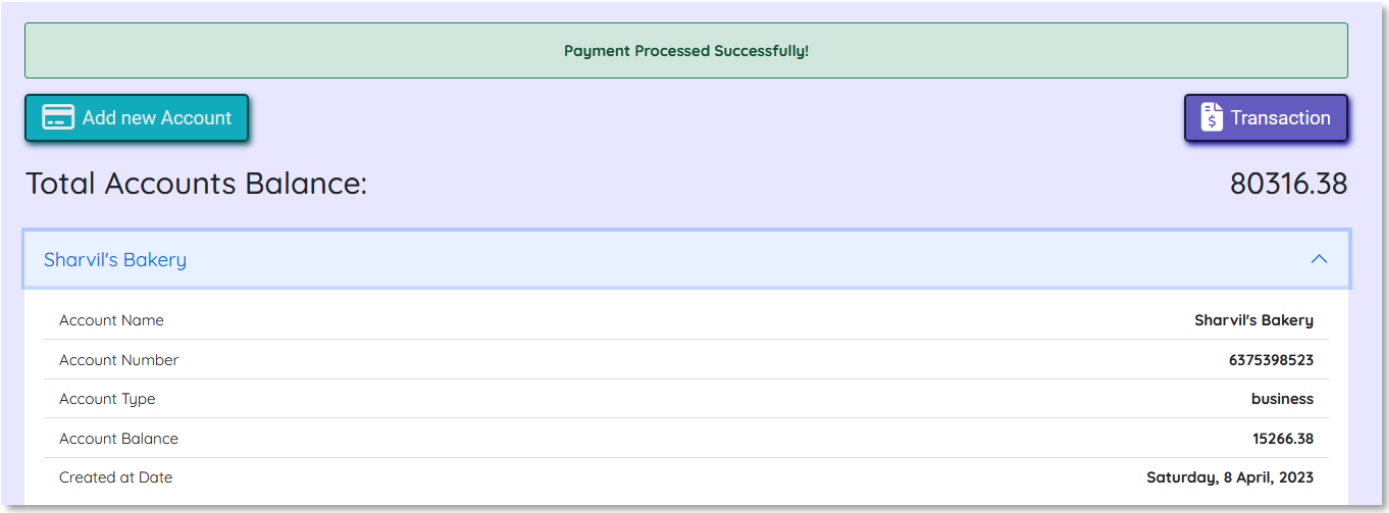


Customer Dashboard (Multiple Accounts) Page

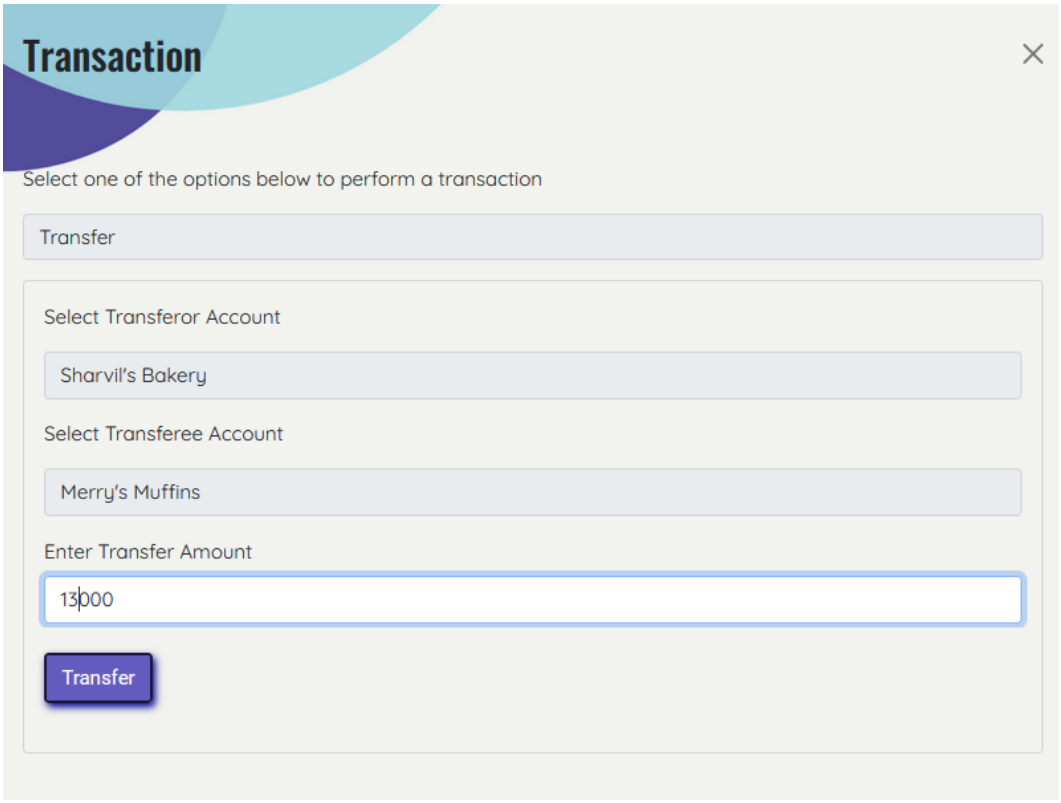
Transaction - Payment Offcanvas



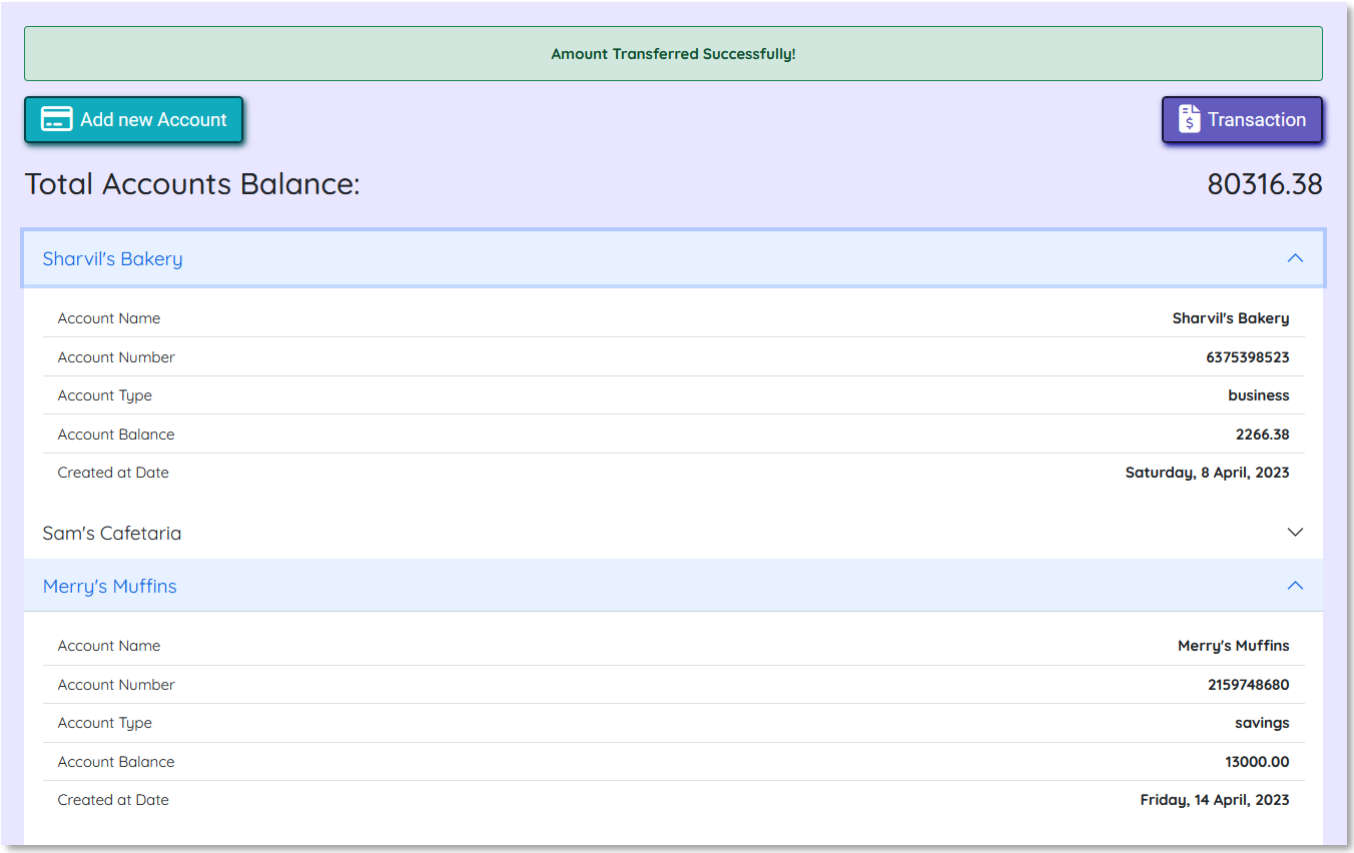
Payment Transaction Successful Page



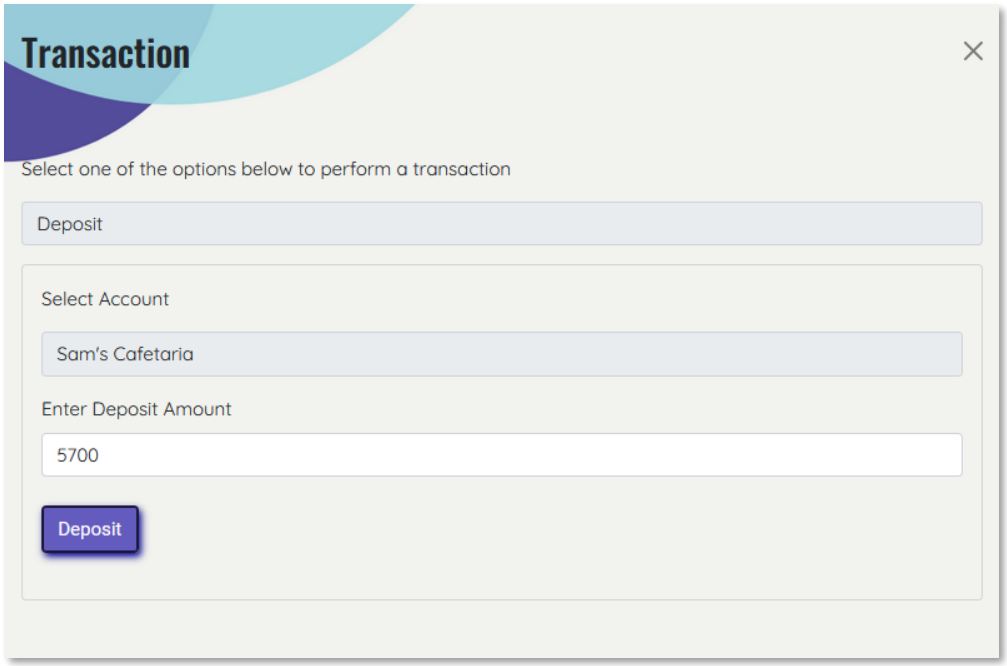
Transaction - Transfer Offcanvas



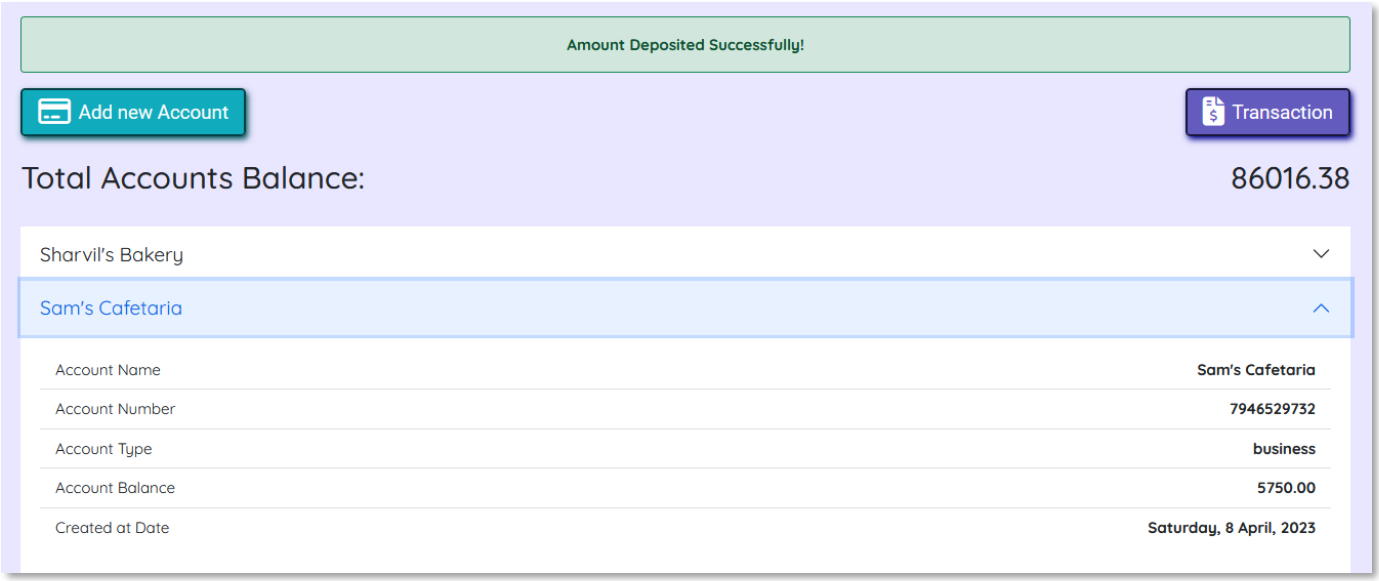
Transfer Transaction Successful Page



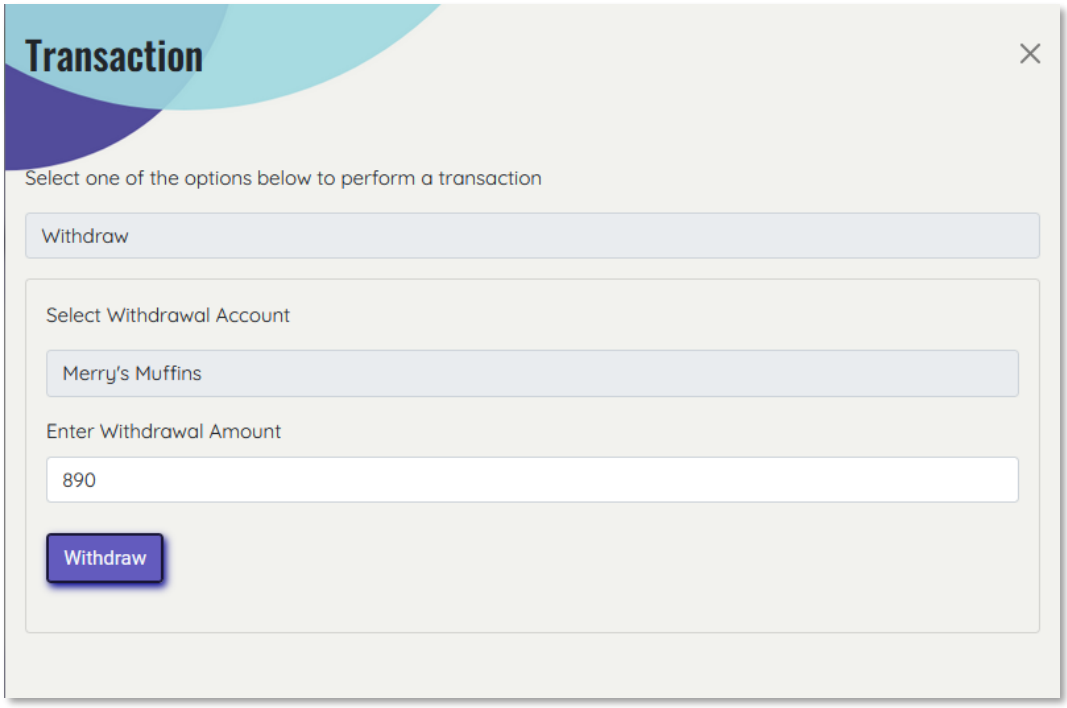
Deposit Transaction Offcanvas



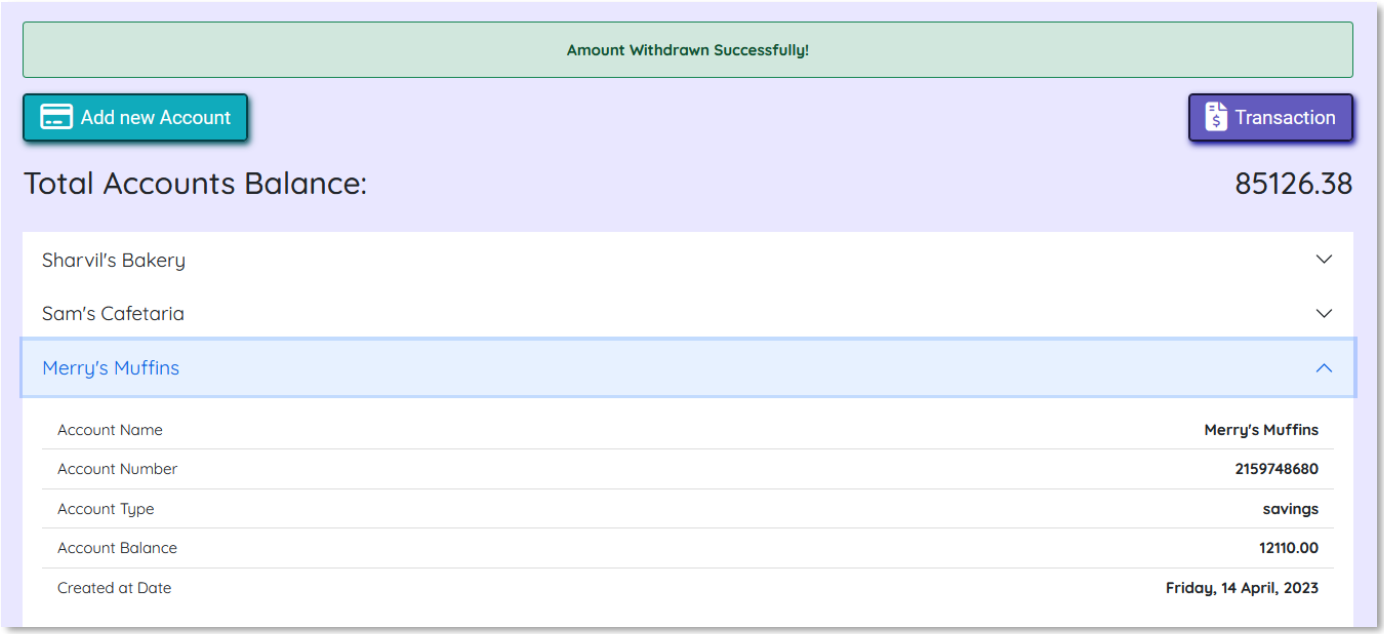
Deposit Transaction Successful Page



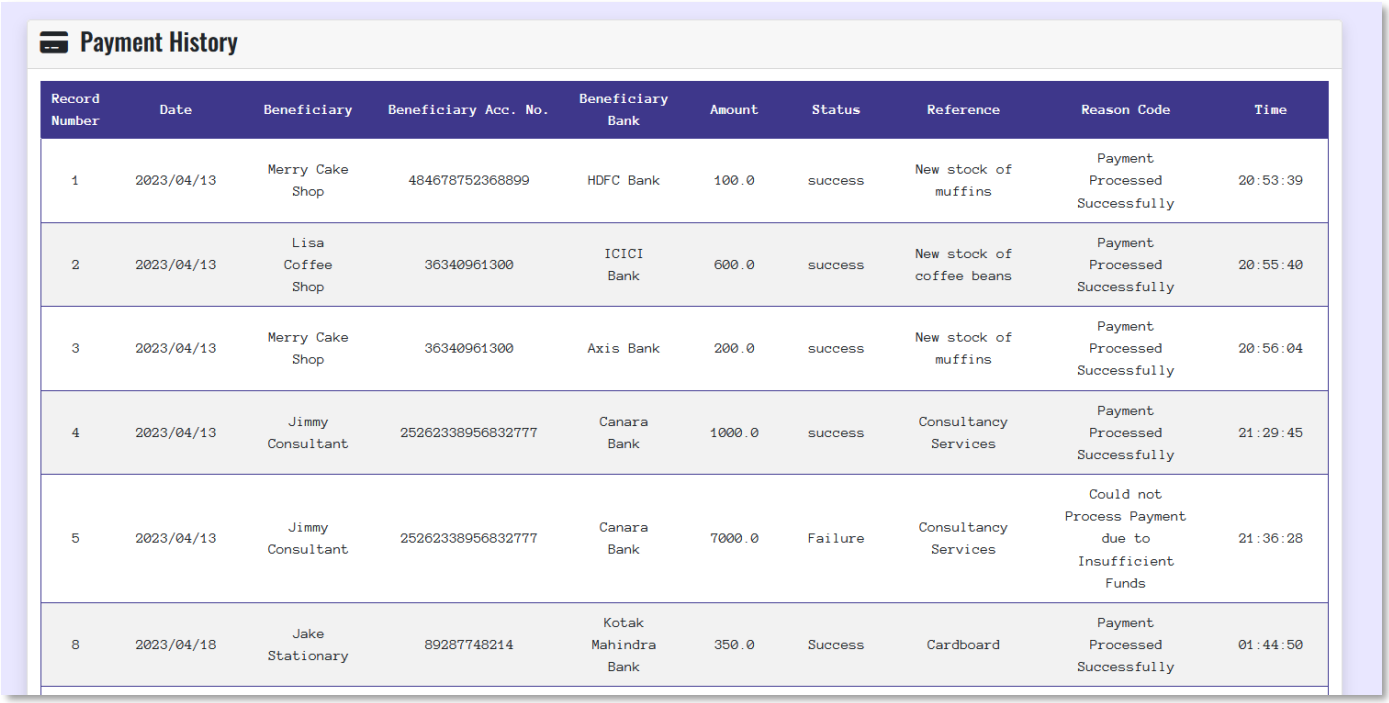
Withdraw Transaction Offcanvas



Withdraw Transaction Successful Page



Payment History Panel Page



Payment History

Record Number	Date	Beneficiary	Beneficiary Acc. No.	Beneficiary Bank	Amount	Status	Reference	Reason Code	Time
1	2023/04/13	Merry Cake Shop	484678752368899	HDFC Bank	100.0	success	New stock of muffins	Payment Processed Successfully	20:53:39
2	2023/04/13	Lisa Coffee Shop	36340961300	ICICI Bank	600.0	success	New stock of coffee beans	Payment Processed Successfully	20:55:40
3	2023/04/13	Merry Cake Shop	36340961300	Axis Bank	200.0	success	New stock of muffins	Payment Processed Successfully	20:56:04
4	2023/04/13	Jimmy Consultant	25262338956832777	Canara Bank	1000.0	success	Consultancy Services	Payment Processed Successfully	21:29:45
5	2023/04/13	Jimmy Consultant	25262338956832777	Canara Bank	7000.0	Failure	Consultancy Services	Could not Process Payment due to Insufficient Funds	21:36:28
8	2023/04/18	Jake Stationary	89287748214	Kotak Mahindra Bank	350.0	Success	Cardboard	Payment Processed Successfully	01:44:50

Payment History Downloaded PDF

Payment History

Record No.	Date	Beneficiary	Beneficiary Acc No.	Beneficiary Bank	Amount	Status	Reference	Reason Code	Time
1	2023/04/13	Merry Cake Shop	4846787523 68899	HDFC Bank	100.0	success	New stock of muffins	Payment Processed Successfully	20:53:39
2	2023/04/13	Lisa Coffee Shop	3634096130 0	ICICI Bank	600.0	success	New stock of coffee beans	Payment Processed Successfully	20:55:40
3	2023/04/13	Merry Cake Shop	3634096130 0	Axis Bank	200.0	success	New stock of muffins	Payment Processed Successfully	20:56:04
4	2023/04/13	Jimmy Consultant	2526233895 6832777	Canara Bank	1000.0	success	Consultanc y Services	Payment Processed Successfully	21:29:45
5	2023/04/13	Jimmy Consultant	2526233895 6832777	Canara Bank	7000.0	Failure	Consultanc y Services	Could not Process Payment due to Insufficient Funds	21:36:28
8	2023/04/18	Jake Stationary	8928774821 4	Kotak Mahindra Bank	350.0	Success	Cardboard	Payment Processed Successfully	01:44:50
9	2023/04/18	Jimmy Consultant	6836256284	Punjab National Bank	200.0	Success	New stock of coffee beans	Payment Processed Successfully	02:01:45
22	2023/05/18	Fruit Heaven	6578630423	HDFC Bank	2300.0	Success	Extra batch of strawberry	Payment Processed Successfully	01:00:11

Transaction History Panel Page

Transaction History

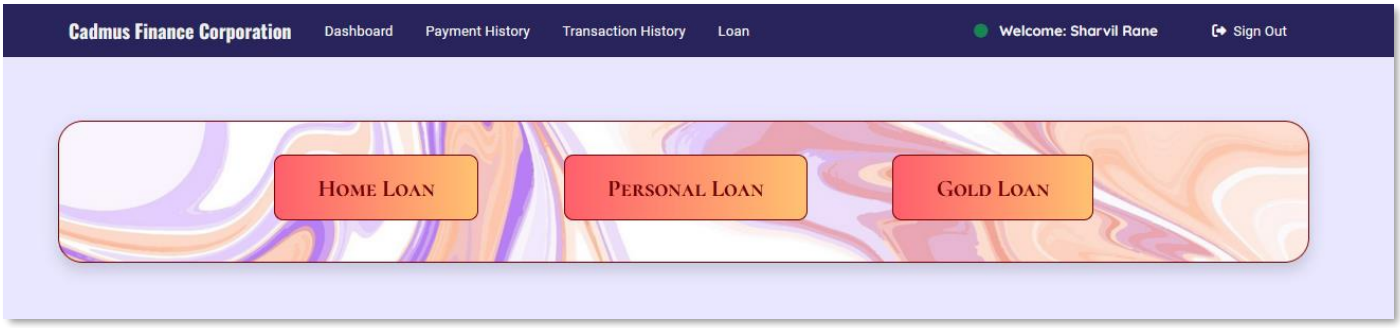
Date	Transaction ID	Transaction Type	Amount	Source	Status	Reason Code	Time
2023/04/13	1	Transfer	650.0	Online	Success	Amount Transfer Successful	21:29:45
2023/04/13	2	Deposit	450.0	Online	Success	Amount Deposit Successful	21:29:45
2023/04/13	5	Withdraw	8000.0	Online	Failure	Insufficient Funds	21:29:45
2023/04/13	3	Withdraw	250.0	Online	Success	Amount Withdraw Successful	21:29:45
2023/04/13	6	Transfer	9000.0	Online	Failure	Insufficient Funds	21:29:45
2023/04/13	4	Payment	1000.0	Online	Success	Amount Process Successful	21:29:45
2023/04/13	7	Payment	7000.0	Online	Failure	Insufficient Funds	21:36:28
2023/04/14	8	Transfer	100.0	Online	Success	Amount Transfer Successful	01:17:52
2023/04/14	9	Deposit	277.0	Online	Success	Amount Deposit Successful	02:07:59
2023/04/14	10	Withdraw	100000.0	Online	Failure	Insufficient Funds	02:32:58
2023/04/16	11	Transfer	200.0	Online	Success	Amount Transfer Successful	20:26:50
2023/04/18	16	Payment	350.0	Online	Success	Amount Process Successful	01:44:50
2023/04/18	17	Payment	200.0	Online	Success	Amount Process Successful	02:01:45
2023/04/18	20	Reception	350.0	Online	Success	Amount Received Successfully	17:22:04
2023/04/18	36	Reception	200.0	Online	Success	Amount Received Successfully	18:07:08
2023/04/18	41	Reception	200.0	Online	Success	Amount Received Successfully	18:26:33

Transaction History Downloaded PDF

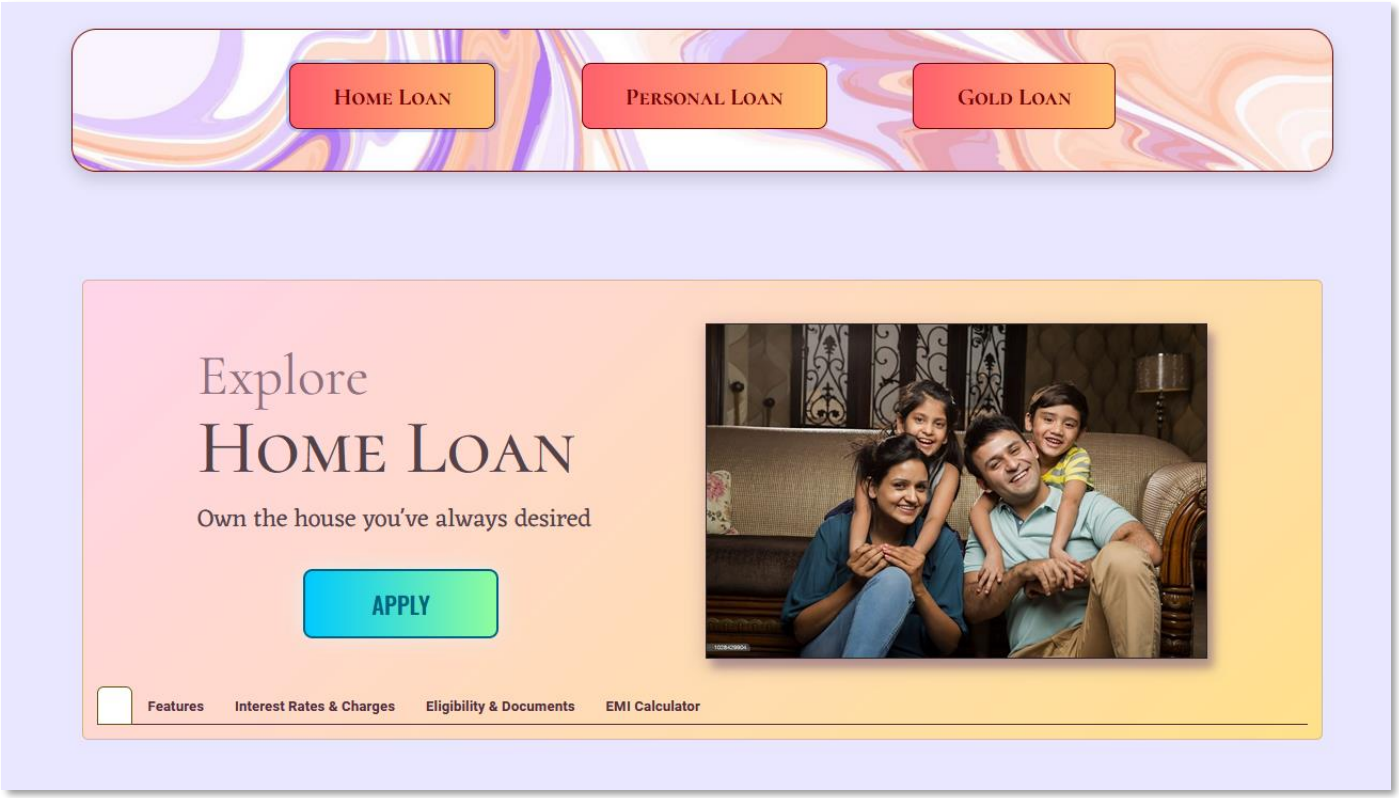
Transaction History

Date	Transaction ID	Transaction Type	Amount	Source	Status	Reason Code	Time
2023/04/13	1	Transfer	650.0	Online	Success	Amount Transfer Successful	21:29:45
2023/04/13	2	Deposit	450.0	Online	Success	Amount Deposit Successful	21:29:45
2023/04/13	5	Withdraw	8000.0	Online	Failure	Insufficient Funds	21:29:45
2023/04/13	3	Withdraw	250.0	Online	Success	Amount Withdraw Successful	21:29:45
2023/04/13	6	Transfer	9000.0	Online	Failure	Insufficient Funds	21:29:45
2023/04/13	4	Payment	1000.0	Online	Success	Amount Process Successful	21:29:45
2023/04/13	7	Payment	7000.0	Online	Failure	Insufficient Funds	21:36:28
2023/04/14	8	Transfer	100.0	Online	Success	Amount Transfer Successful	01:17:52
2023/04/14	9	Deposit	277.0	Online	Success	Amount Deposit Successful	02:07:59
2023/04/14	10	Withdraw	100000.0	Online	Failure	Insufficient Funds	02:32:58
2023/04/16	11	Transfer	200.0	Online	Success	Amount Transfer Successful	20:26:50
2023/04/18	16	Payment	350.0	Online	Success	Amount Process Successful	01:44:50
2023/04/18	17	Payment	200.0	Online	Success	Amount Process Successful	02:01:45
2023/04/18	20	Reception	350.0	Online	Success	Amount Received Successfully	17:22:04

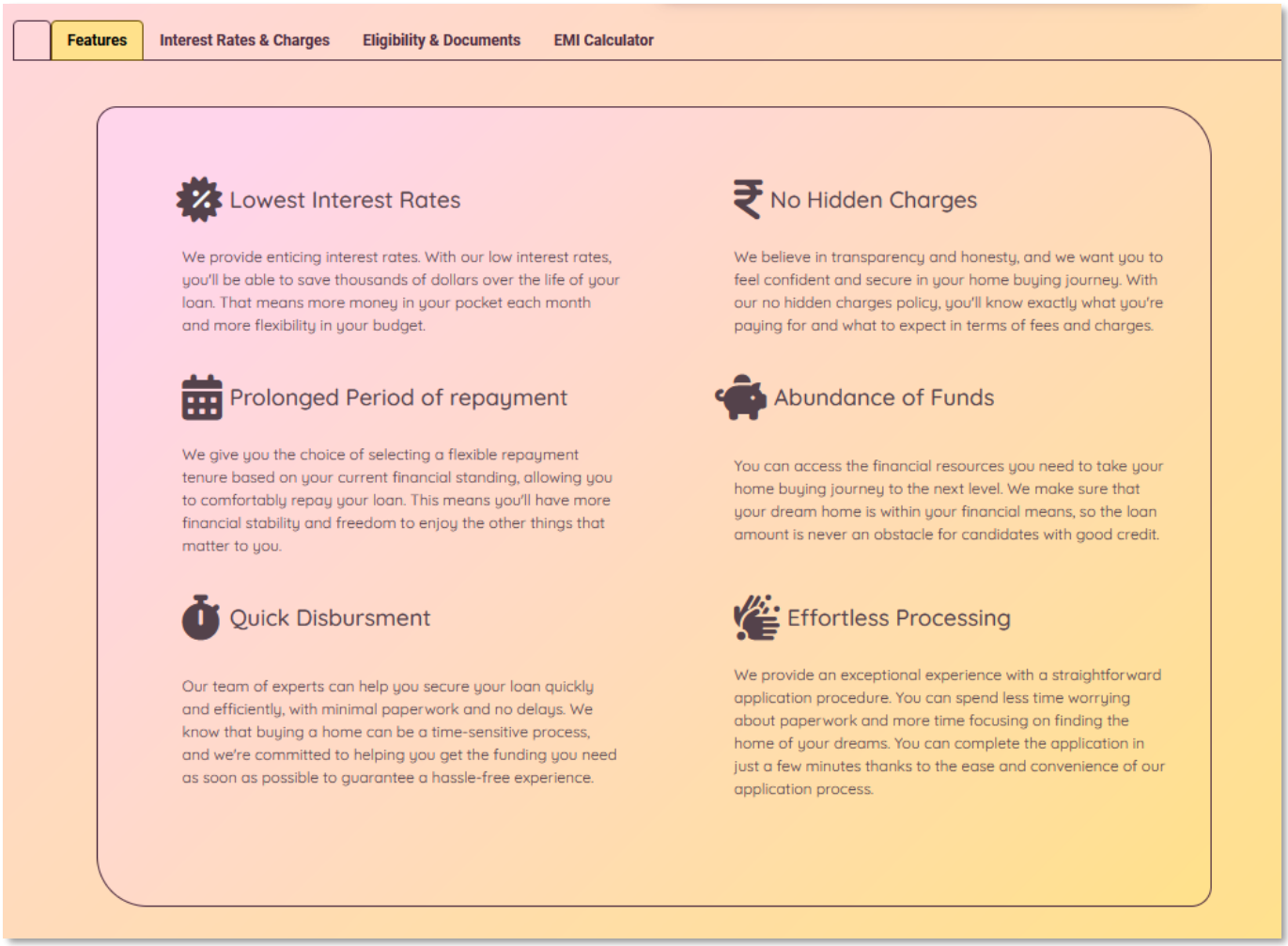
Loan Panel Page



Home Loan Section



Home Loan - Features Section



Home Loan (Interest Rate & Charges) Section

Standard Home Loan Interest Rates

Loan Slab	Salaried	Self-Employed
Upto ₹ 35 lakhs	9.25%	9.60%
₹ 35 lakhs to ₹ 75 lakhs	10.75%	11.00%
Above ₹ 75 lakhs	11.60%	11.85%

Standard Home Loan Charges

Fee	Amount	Minimum	Maximum
★ **Processing Fee:** A one-time fee charged to process the loan application	0.50% of the Loan Amount	₹ 1500	₹ 20000
★ **Administration Fee:** A one-time fee charged to cover the administrative tasks	0.3% of the Loan Amount	₹ 1500	₹ 10000
★ **Appraisal Fee:** A fee charged to have the home appraised to determine its value	₹ 2000	-	-
★ **Title Search Fee:** A fee charged to ensure that the property has a clear title and is free from any license or legal issues	0.15% of Property Value	₹ 1500	₹ 8000
★ **Late Payment Penalty:** A fee charged if the candidate misses a payment or makes a payment after the due date	2% of the Overdue Amount	-	-
★ **Pre Payment Penalty:** A fee charged if the candidate pays off the loan prematurely.	3% of Principal Due	-	-

Disclaimer:
Cadmus Bank reserves the right to revise the rate of interest and processing fee from time to time, at its sole discretion
GST and other government taxes and levies, as applicable, will be levied in addition to these charges at the bank's discretion.

Home Loan (Eligibility & Documents) Section

Home Loan Eligibility Criteria

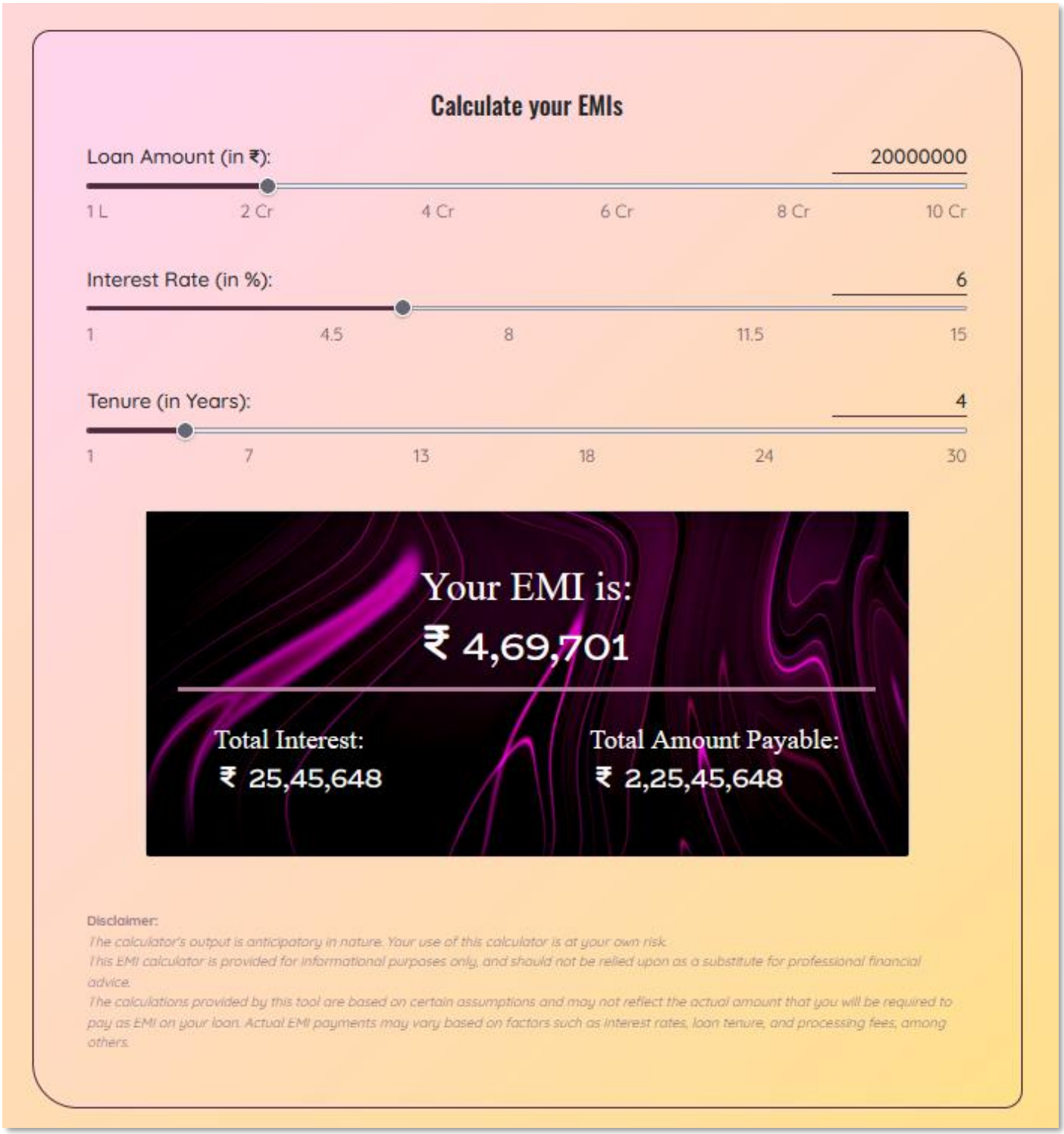
Eligibility	Criteria
Nationality	Indian
Age	23-62 years
Employement Status	Atleast 3 years of Work Experience
Business Status	Atleast 5 years of Business Continuity in Current Business
Minimum Net Salary	₹ 30,000
Minimum Property Value	₹ 15 lakh

Home Loan Document Requirements

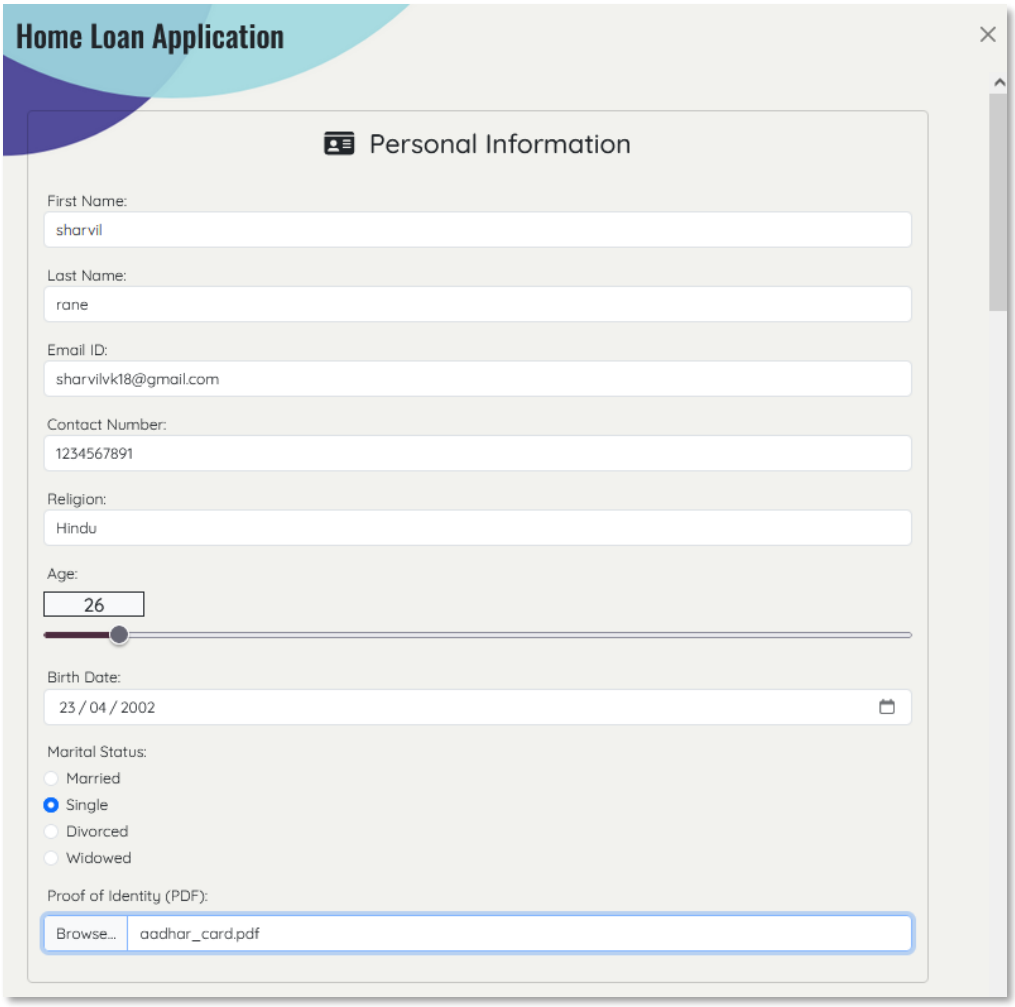
Driving License, Aadhar Card, Voter ID *(Any One)*
Employee ID Card
Salary Slip of last 2 months
Bank Account Statement for the Last 3 Months *(Salaried)* **6 months** *(Self-Employed)*
Document of Proof of Business of minimum 5 years *(Businessmen / Self-Employed)*

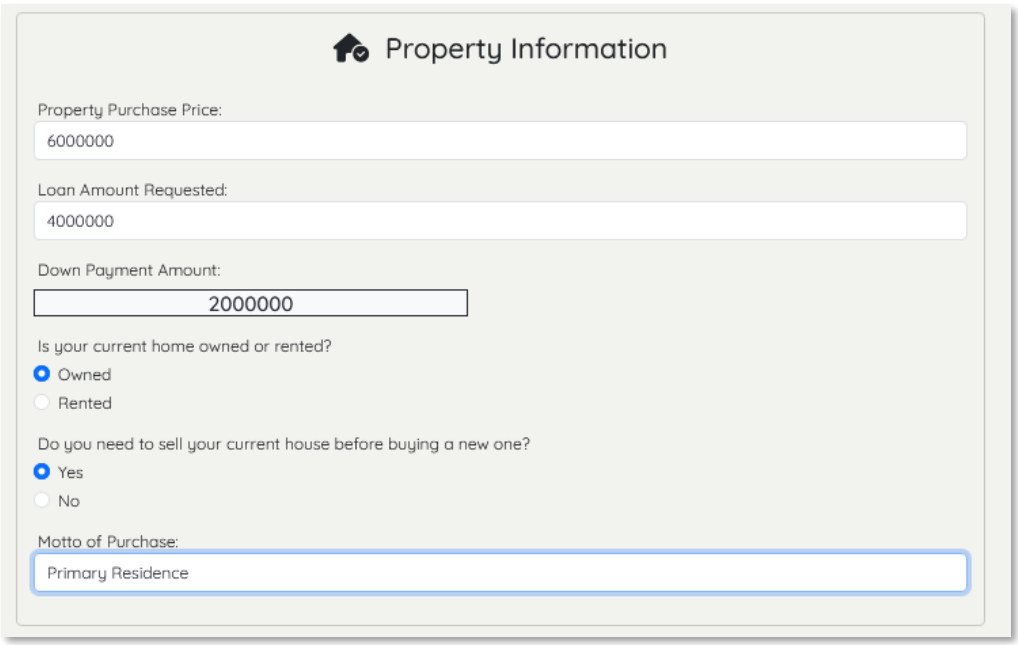
Disclaimer: *Please take note that the above list of qualifying requirements is only indicative. Terms & Conditions apply.*
Age at loan maturity is taken into account while determining the upper age limit.
Please be aware that the list of given documents is only representative. Additional documents might be required during the loan application process.

Home Loan (EMI Calculator) Section



Home Loan Application Offcanvas





Home Loan Application Successful Page

Personal Loan Section

Explore

PERSONAL LOAN

Make the most of your life

APPLY

Features | Interest Rates & Charges | Eligibility & Documents | EMI Calculator

Lowest Interest Rates

We provide enticing interest rates. With our low interest rates, you'll be able to save thousands of dollars over the life of your loan. That means more money in your pocket each month and more flexibility in your budget.

No Hidden Charges

We believe in transparency and honesty, and we want you to feel confident and secure in fulfilling your future endeavours . With our no hidden charges policy, you'll know exactly what you're paying for and what to expect in terms of fees and charges.

No Collateral

With our no collateral personal loans, you can get the funds you need without having to worry about putting any of your assets at risk. You are not required to offer any collateral, such as property documents or gold ornaments, or to have someone else stand in as a guarantor.

Customer Oriented

Candidates are exempt from premature payment penalties, allowing them to pay early without any precussions. Our personal loan offers the option to apply for loan forgiveness or deferment in certain circumstances such as job loss or illness. Hence the candidate can have peace of mind knowing that you have options in the event of financial hardship.

Quick Disbursment

Our team of experts can help you secure your loan quickly and efficiently, with minimal paperwork and no delays. We know that chasing your dream can be a time-sensitive process, and we're committed to helping you get the funding you need as soon as possible to guarantee a hassle-free experience.

Effortless Processing

We provide an exceptional experience with a straightforward application procedure. You can spend less time worrying about paperwork and more time focusing on hustling for your goals. You can complete the application in just a few minutes thanks to the ease and convenience of our application process.

Gold Loan Section

Explore

GOLD LOAN

Get Quick Cash Using Your Gold

APPLY

Features | Interest Rates & Charges | Eligibility & Documents | EMI Calculator

Lowest Interest Rates

We provide enticing interest rates. With our low interest rates, you'll be able to save thousands of dollars over the life of your loan. We have a high loan to value ratio, which indicates that a significant percentage of the appraised value of the pledged gold will be received as loan amount.

No Hidden Charges

We believe in transparency and honesty, and we want you to feel confident and secure in lending your gold. With our no hidden charges policy, you'll know exactly what you're paying for and what to expect in terms of fees and charges.

Part Release Facility

Our gold loan now offers a part release facility that allows you to pledge only a portion of your gold and get a reduced loan amount. With this feature, you can release the amount of gold you don't want to pledge and reduce the loan amount accordingly. This means you can borrow only the amount you need and repay the loan with a lower interest rate.

Security

We ensure the safety of the pledged gold by storing the pledged gold in a safety locker or security vault. Additionally, we provide insurance coverage in case your gold jewellery is stolen or lost while it is in our care. The storage facility has 24/7 surveillance with fire extinguishers, smoke detectors, and security alarms and heavily audited by authorized personnel.

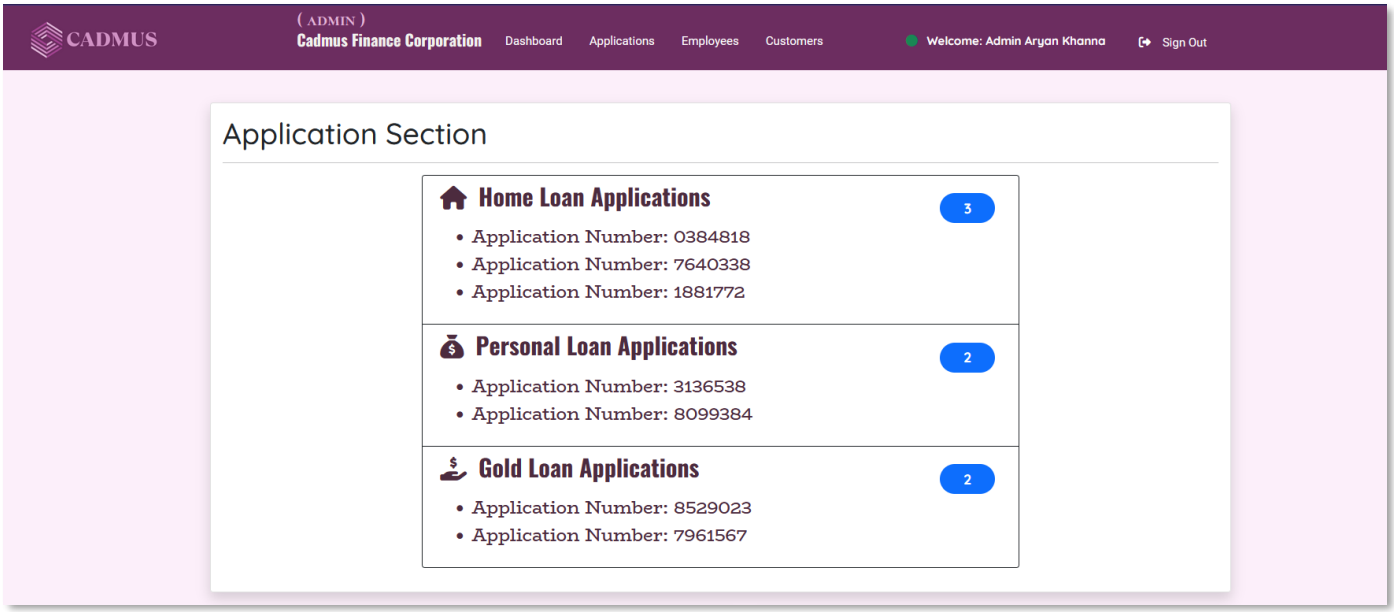
Quick Disbursment

Our team of experts can help you secure your loan quickly and efficiently, with minimal paperwork and no delays. We know that lending gold is a tedious process, and we're committed to helping you get the funding you need as soon as possible to guarantee a hassle-free experience. Your bank account will be credited with your loan amount within 24 hours.

Effortless Processing

We provide an exceptional experience with a straightforward application procedure. You can spend less time worrying about paperwork and more time focusing on planning your tasks. You can complete the application in just a few minutes thanks to the ease and convenience of our application process. We utilise best-in-class gold metres to analyse quality and assure the greatest value of gold assets.

Admin Dashboard Page



Employee DB

ID	Name	Age	Job Title	Hire Date	Department	Branch
64	Kavya Shah	25	Customer Service Trainer	2020-12-10	Customer Service	Ahmedabad West
10	Shruti Patil	25	Accountant	2020-11-27	Accounting	Pune East
26	Amitabh Kumar	27	Financial Auditor	2020-11-11	Audit	New Delhi Main
108	Deepa Patil	27	Sales Consultant	2020-10-15	Sales	Pune East
39	Praveen Shetty	32	Branch Manager	2020-09-10	Branch Operations	Pune East
104	Shweta Gupta	28	Sales Representative	2020-08-22	Sales	Pune East

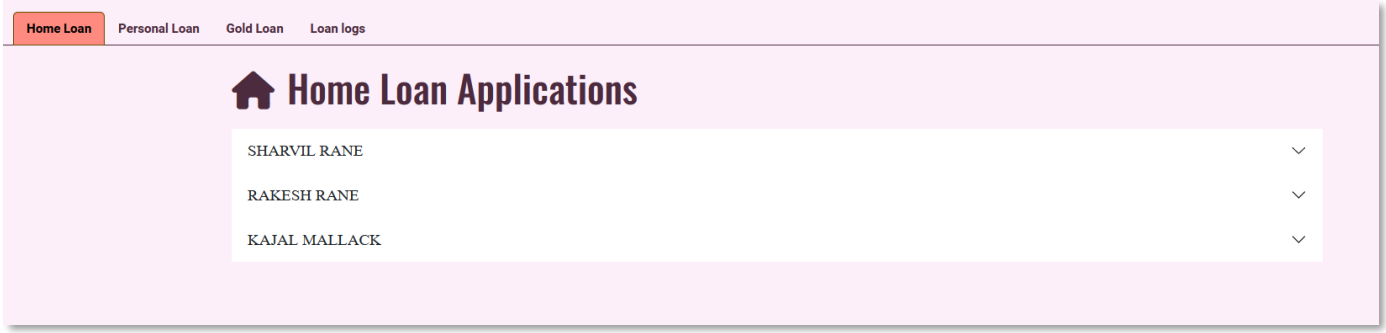
Customer DB

ID	Name	Email	Account Count
12	Sharvil Rane	sharvil.rane@cadmus.com	5
13	Rakesh Rane	rakesh.rane@cadmus.com	2
14	Shilpa Rane	jerejad858@lieboe.com	1
22	Sharat Chandran	sharat.chandran@gmail.com	1
23	Kajal Mallick	kajal.mallick@gmail.com	1
24	Nahar Singh	nahar.singh@gmail.com	0
25	Kanika Kathuria	kanika.kathuria@gmail.com	0
26	Jyoti Mathur	jyoti.mathur@gmail.com	0
27	Maansingh Aswal	maansingh.aswal@gmail.com	0
28	Priya Jain	priya.jain@gmail.com	0

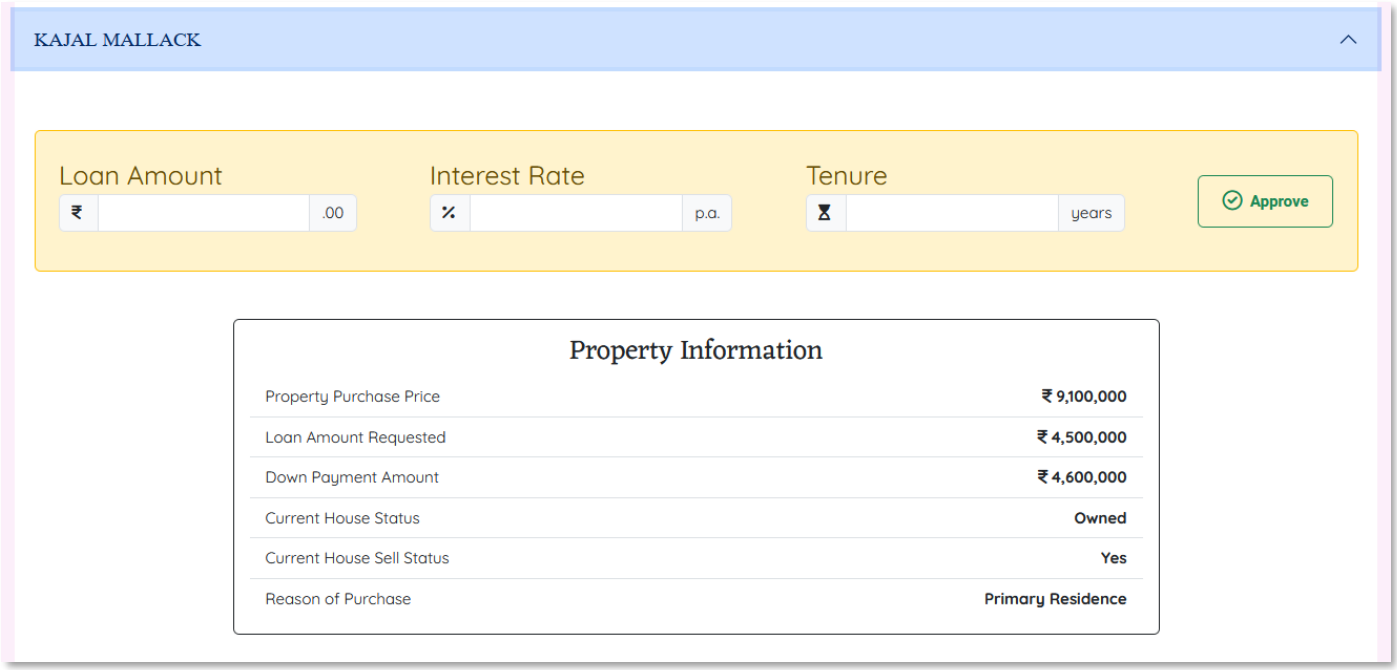
Transaction Section

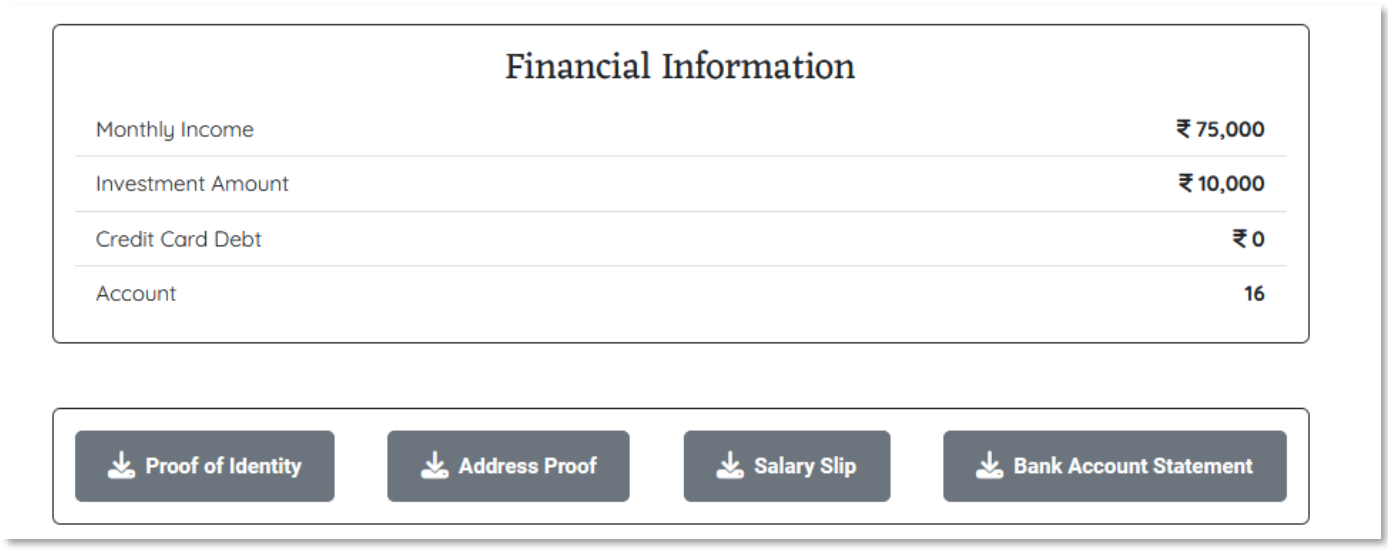
Transaction ID	Account ID	User ID	Type	Amount	Source	Status	Reason Code	Date	Time
1	4	12	Transfer	650.0	Online	Success	Amount Transfer Successful	21:29:45	2023/04/13
2	4	12	Deposit	450.0	Online	Success	Amount Deposit Successful	21:29:45	2023/04/13
5	4	12	Withdraw	8000.0	Online	Failure	Insufficient Funds	21:29:45	2023/04/13
3	5	12	Withdraw	250.0	Online	Success	Amount Withdraw Successful	21:29:45	2023/04/13
4	5	12	Payment	1000.0	Online	Success	Amount Process Successful	21:29:45	2023/04/13
6	5	12	Transfer	9000.0	Online	Failure	Insufficient Funds	21:29:45	2023/04/13
7	4	12	Payment	7000.0	Online	Failure	Insufficient Funds	21:36:28	2023/04/13
8	4	12	Transfer	100.0	Online	Success	Amount Transfer Successful	01:17:52	2023/04/14
9	4	12	Deposit	277.0	Online	Success	Amount Deposit Successful	02:07:59	2023/04/14
10	5	12	Withdraw	100000.0	Online	Failure	Insufficient Funds	02:32:58	2023/04/14
11	4	12	Transfer	200.0	Online	Success	Amount Transfer Successful	20:26:50	2023/04/16
12	8	13	Deposit	1000.0	Online	Success	Amount Deposit Successful	19:51:18	2023/04/17
13	8	13	Payment	1000.0	Online	Success	Amount Process Successful	19:51:18	2023/04/17
14	9	14	Deposit	1000.0	Online	Success	Amount Deposit Successful	00:29:23	2023/04/18
15	9	14	Payment	200.0	Online	Success	Amount Process Successful	00:29:23	2023/04/18
16	5	12	Payment	350.0	Online	Success	Amount Process Successful	01:44:50	2023/04/18

Admin Dashboard Application Panel

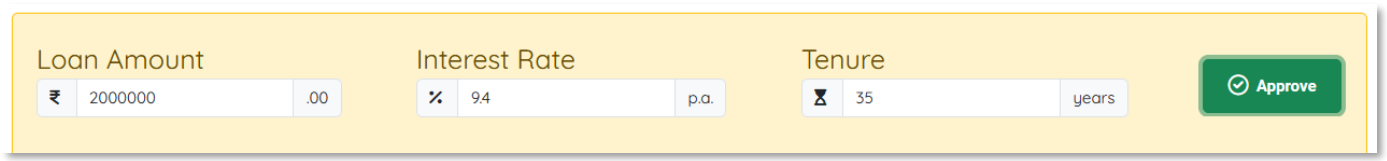


Applied Home Loan Information

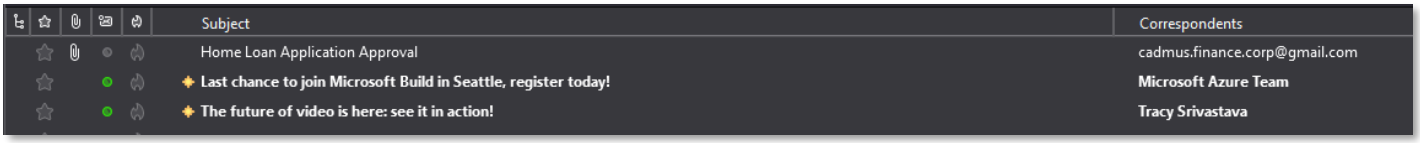




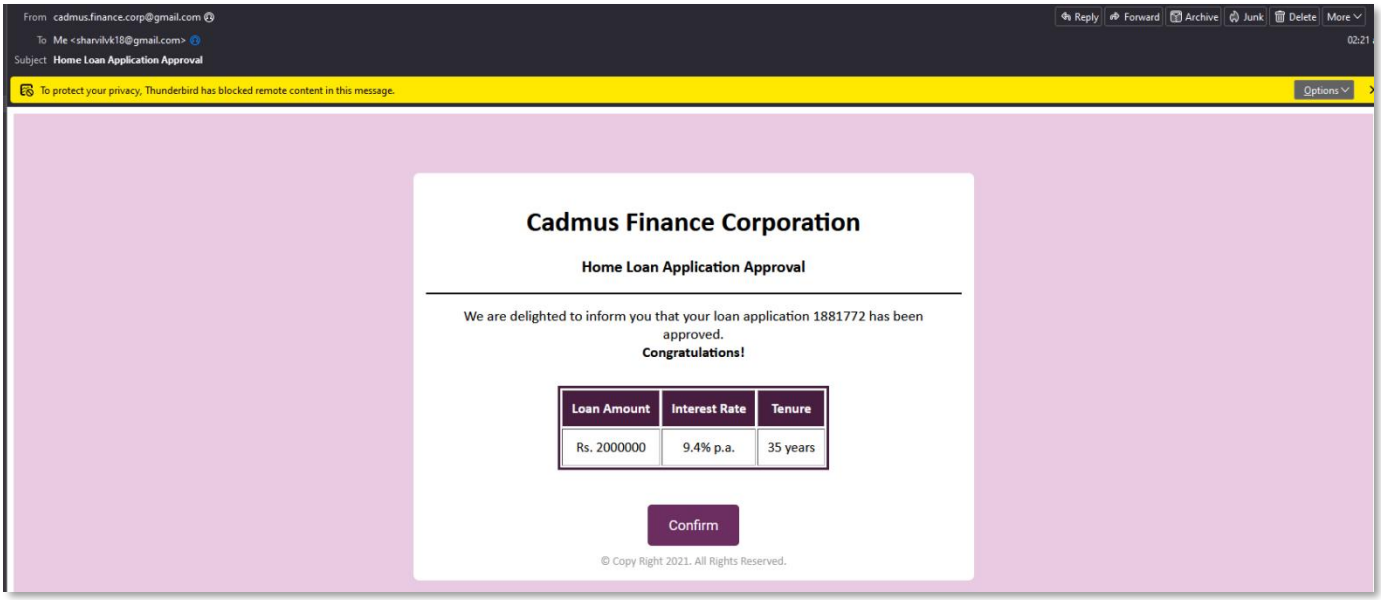
Approve Loan Section



Approval Email in Mail Application



Approval Email Content



Insufficient Amount in Account Email

Dear Sharvil Rane,

We hope this email finds you in good health.

We're writing to notify you of a recent loan payment attempt on your Gold Loan with application number 8529023 . Unfortunately, your account did not have enough funds to cover the scheduled Equated Monthly Installment (EMI). As a result, a penalty has been applied to your outstanding loan amount.

We understand that unforeseen circumstances can occur, leading to temporary financial difficulties. However, it is crucial to ensure that sufficient funds are available at the time of the next EMI to avoid further penalties and potential consequences.

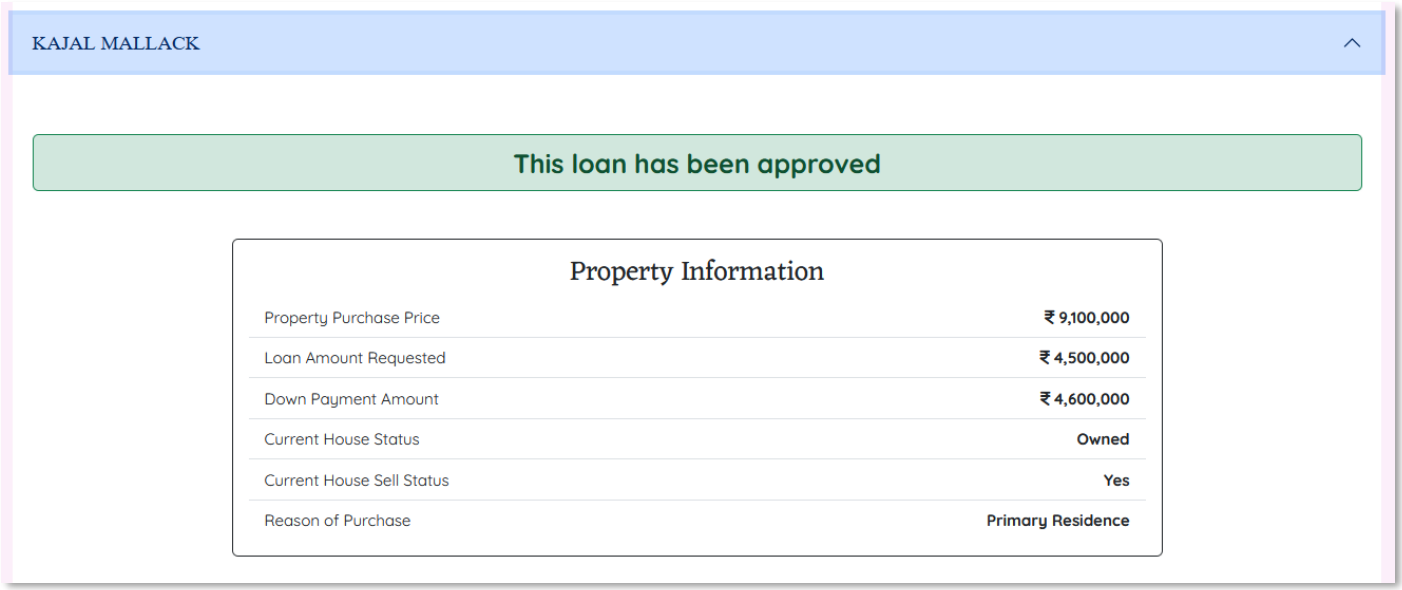
Penalty Details:

- Loan Application Number: 8529023
- Penalty Amount: 691.11
- New Total Amount Payable: 230007.57

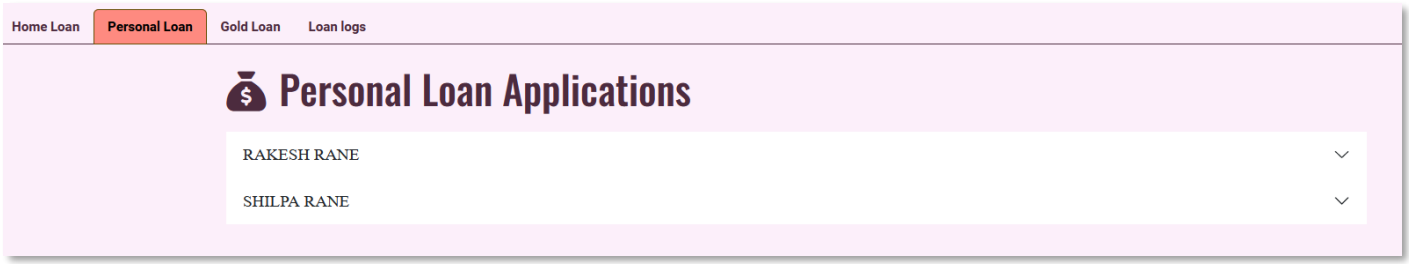
To rectify the situation, we kindly request that you make the necessary arrangements to ensure your account maintains sufficient funds before the next EMI payment date.

Best Regards,
Cadmus Financial Corporation.

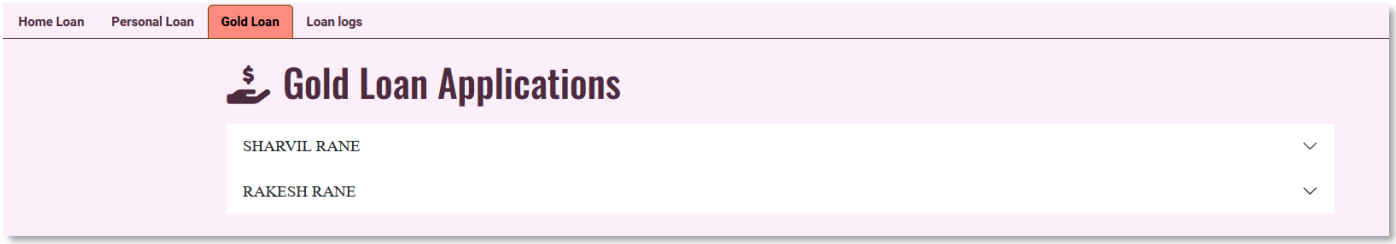
Applied Home Loan after Approval



Personal Loan Section



Gold Loan Section



Loan Log Section

Home Loan | Personal Loan | Gold Loan | Loan logs

Loan Application Number	Loan Type	Borrower Name	Borrower User ID	Borrower Account ID	Borrower Email	Loan Amount	Interest Rate	Tenure	EMI	Amount Payable	Interest Payable	Charges Payable	Late Payment Penalty	Pre Payment Penalty	Confirmation	Creation Time	Update Time
1881772	Home Loan	Kajal Mallack	23	16	sharvilvk18@gmail.com	2000000.0	9.4	420	16281.01	6838025.6	4838025.6	21000.0	325.62	488.43	no	2023-05-18 02:21:31	2023-05-18 02:21:31
3136538	Personal Loan	Rakesh Rane	13	8	sharvilvk18@gmail.com	400000.0	17.2	14	31737.44	0.0	44324.09	0.0	634.75	0.0	yes	2023-05-17 02:34:19	2023-05-17 05:47:59
7640338	Home Loan	Rakesh Rane	13	8	sharvilvk18@gmail.com	2000000.0	6.7	288	13979.1	2970526.63	2025980.03	0.0	279.58	419.37	yes	2023-05-17 02:34:19	2023-05-17 06:48:59
7961567	Gold Loan	Rakesh Rane	13	8	sharvilvk18@gmail.com	1312000.0	8.0	12	114128.93	0.0	57547.13	0.0	2282.58	0.0	yes	2023-05-17 02:34:19	2023-05-17 05:47:59
8529023	Gold Loan	Sharvil Rane	12	4	sharvilvk18@gmail.com	400000.0	6.7	12	34555.68	230007.57	14668.19	0.0	691.11	0.0	yes	2023-05-17 06:39:34	2023-05-17 06:46:59

Employee Panel

All Records | Personal | Work Representation | Departments | Branches

ID	Name	Email	Contact	Address	City	State	Postal Code	Hire Date	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language	Employee Type	Employee Status	Job Title
109	Vikas Khanna	vikas.khanna@gmail.com	1022158000	Karol Bagh	Faridabad	Delhi	110005	2010-01-11	28	Male	Single	Post Graduate	English	Hindi	Full Time	Active	Territory Sales Manager
92	Shruti Rao	shruti.rao@gmail.com	1447749839	Jubilee Hills	Hyderabad	Telangana	500033	2010-01-23	25	Female	Single	Graduate	English	Telugu	Full Time	Active	Product Marketing
43	Tanvi Singh	tanvi.singh@gmail.com	9223239152	Tagore Nagar	Ludhiana	Punjab	141002	2010-03-09	29	Female	Married	Post Graduate	English	Punjabi	Full Time	Active	Corporate Banking Analyst
15	Aarav Kapoor	aarav.kapoor@gmail.com	4310622993	Ravi Road	Pune	Maharashtra	400001	2010-04-09	25	Male	Single	Post Graduate	English	Hindi	Full Time	Active	Bookkeeper
18	Rajat Singh	rajat.singh@gmail.com	9660738259	GT Road	Amritsar	Punjab	143001	2010-04-13	28	Male	Married	Post Graduate	English	Punjabi	Full Time	Active	Tax Counsel

Employee Personal Information Section

-- Select Grouping By Column -- | GROUP | -- Select Sorting Column -- | Ascending | Descending | SORT

ID	Name	Email	Contact	Address	City	State	Postal Code	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language
109	Vikas Khanna	vikas.khanna@gmail.com	1022158000	Karol Bagh	Faridabad	Delhi	110005	28	Male	Single	Post Graduate	English	Hindi
92	Shruti Rao	shruti.rao@gmail.com	1447749839	Jubilee Hills	Hyderabad	Telangana	500033	25	Female	Single	Graduate	English	Telugu
43	Tanvi Singh	tanvi.singh@gmail.com	9223239152	Tagore Nagar	Ludhiana	Punjab	141002	29	Female	Married	Post Graduate	English	Punjabi
15	Aarav Kapoor	aarav.kapoor@gmail.com	4310622993	Ravi Road	Pune	Maharashtra	400001	25	Male	Single	Post Graduate	English	Hindi
18	Rajat Singh	rajat.singh@gmail.com	9660738259	GT Road	Amritsar	Punjab	143001	28	Male	Married	Post Graduate	English	Punjabi

Personal Information Group by City

ID	Name	Email	Contact	Address	City	State	Postal Code	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language
66	Isha Malhotra	isha.malhotra@gmail.com	9887766344	Vijay Nagar Colony	Ghaziabad	Delhi	110001	32	Female	Married	Graduate	English	Punjabi
80	Pallavi Saini	pallavi.saini@gmail.com	9163575721	Raj Nagar		Delhi	110016	26	Female	Single	Graduate	English	Hindi

ID	Name	Email	Contact	Address	City	State	Postal Code	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language
18	Rajat Singh	rajat.singh@gmail.com	9660738259	GT Road	Amritsar	Punjab	143001	28	Male	Married	Post Graduate	English	Punjabi
48	Raman Chawla	raman.chawla@gmail.com	1024502111	Lawrence Road		Punjab	143001	29	Male	Single	Post Graduate	English	Punjabi
58	Aman Mitra	aman.mitra@gmail.com	7071179111	Rani Ka Bagh		Punjab	143001	29	Male	Married	Bachelor's Degree	English	Punjabi

ID	Name	Email	Contact	Address	City	State	Postal Code	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language
27	Sneha Yadav	sneha.yadav@gmail.com	4596303506	Civil Lines	Jalandhar	Punjab	144001	28	Female	Married	Graduate	English	Punjabi
94	Amrita Dube	amrita.dube@gmail.com	4601547786	Durga Nagar		Punjab	144003	29	Female	Married	Graduate	English	Punjabi

ID	Name	Email	Contact	Address	City	State	Postal Code	Age	Gender	Marital Status	Education Level	Primary Language	Subsidiary Language
13	Rahul Maitra	rahul.maitra@gmail.com	1577452739	Connaught Place	Gurugram	Delhi	110001	24	Male	Single	Graduate	English	Bengali
28	Ravi Narang	ravi.narang@gmail.com	7983000649	Ashok Vihar		Delhi	110052	32	Male	Married	Post Graduate	English	Hindi
78	Karan Hariharan	karan.hariharan@gmail.com	7169059869	Jasmine Heights		Delhi	110016	29	Male	Single	Post Graduate	English	Hindi

Employee Work Representation with Searched Employee

All Records | Personal | Work Representation | Departments | Branches

Shruti Rao SEARCH

-- Select Grouping By Column -- GROUP

-- Select Sorting Column -- Ascending Descending SORT

ID	Name	Hire Date	Education Level	Employee Type	Employee Status	Job Title	Salary	Department ID	Branch ID	Manager ID
109	Vikas Khanna	2010-01-11	Post Graduate	Full Time	Active	Territory Sales Manager	60000.0	12	6	114
92	Shruti Rao	2010-01-23	Graduate	Full Time	Active	Product Marketing	50000.0	10	2	101
43	Tanvi Singh	2010-03-09	Post Graduate	Full Time	Active	Corporate Banking Analyst	50000.0	6	5	48

Employee Department Section

ID	Name	Head	Description
1	Accounting	Mira Rao	Responsible for maintaining the financial records and transactions of the bank.
2	Audit	Sneha Yadav	Responsible for monitoring and ensuring compliance with the bank's policies and procedures.
3	Branch Operations	Deepika Singh	Responsible for managing the day-to-day operations of the bank branch.
5	Customer Service	Isha Malhotra	Responsible for providing high-quality customer service and support to clients.
6	Finance	Raman Chawla	Responsible for managing the bank's financial resources and investments.
7	Human Resources	Mihir Shah	Responsible for managing employee relations, recruitment, and compensation.
8	Information Technology	Rajesh Nayar	Responsible for managing the bank's technology infrastructure and systems.
10	Marketing	Vikram Trivedi	Responsible for creating and executing marketing strategies to promote bank services and products.
12	Sales	Kavya Thakur	Responsible for generating revenue and sales for the bank.

Employee Branch Section

ID	Name	Address	City	State	Contact	Branch Manager ID
1	Mumbai Central	Cathedral Road, Dadar	Mumbai	Maharashtra	9876543210	23
2	Hyderabad Central	Sapphire Street, Banjara Hills	Hyderabad	Telangana	4078901234	53
3	Ahmedabad West	Skyline Heights, CG Road	Ahmedabad	Gujarat	7989012345	17
4	Pune East	Coral Avenue, Koregaon Park	Pune	Maharashtra	2090123456	82
5	Chandigarh City	Royal Gardens, Sector 17	Chandigarh	Punjab	1723456789	64
6	New Delhi Main	Diamond Drive, Connaught Place	New Delhi	Delhi	1123456789	77
7	Coimbatore West	Silver Springs, Race Course Road	Coimbatore	Tamil Nadu	4228901234	59

Customer Panel Sort by Name Descending

Name | Ascending | Descending | SORT

ID	Name	Email	Creation Time	Verification Time	Update Time	Number of Accounts
34	Yash Mittal	yash.mittal@gmail.com	2010-12-09 17:14	2010-12-09 17:19	2023-05-18T01:58:27	0
53	Vipin Garg	vipin.garg@gmail.com	2016-04-03 09:38	2016-04-03 09:43	2023-05-18T01:58:27	0
32	Vikash Senger	vikash.senger@gmail.com	2023-02-15 04:27	2023-02-15 04:32	2023-05-18T01:58:27	0
38	Vidhi Kashyap	vidhi.kashyap@gmail.com	2013-02-03 12:32	2013-02-03 12:37	2023-05-18T01:58:27	0
29	Vandana Krishnamurthy	vandana.krishnamurthy@gmail.com	2011-01-13 07:35	2011-01-13 07:40	2023-05-18T01:58:27	0
49	Tina Singal	tina.singal@gmail.com	2019-12-03 16:11	2019-12-03 16:16	2023-05-18T01:58:27	0
31	Tapas Das	tapas.das@gmail.com	2015-12-15 15:20	2015-12-15 15:25	2023-05-18T01:58:27	0
42	Swati Joshi	swati.joshi@gmail.com	2016-01-16 02:46	2016-01-16 02:51	2023-05-18T01:58:27	0
50	Subhash Jha	subhash.jha@gmail.com	2017-03-03 03:57	2017-03-03 04:02	2023-05-18T01:58:27	0
46	Subhadra Katoch	subhadra.katoch@gmail.com	2019-06-01 05:52	2019-06-01 05:57	2023-05-18T01:58:27	0

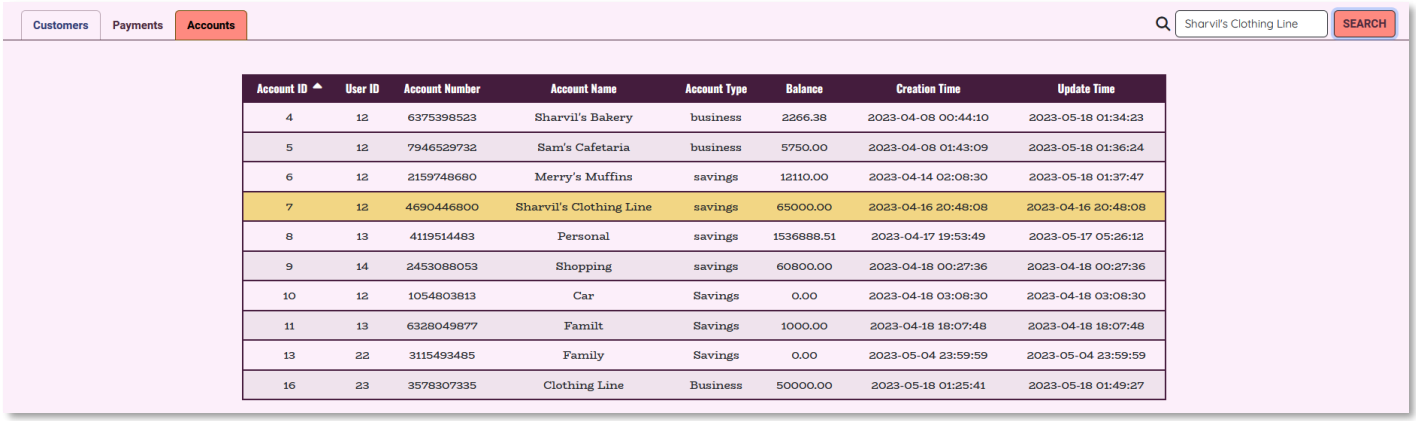
Payments Section Group by Beneficiary Bank

Beneficiary Bank | GROUP | -- Select Sorting Column -- | Ascending | Descending | SORT

Payment ID	Account ID	User ID	Beneficiary	Beneficiary Account No.	Beneficiary Bank	Amount	Status	Reference	Reason Code	Payment Time
3	4	12	Merry Cake Shop	36340961300	Axis Bank	200.0	success	New stock of muffins	Payment Processed Successfully	2023-04-13 20:56:04

Payment ID	Account ID	User ID	Beneficiary	Beneficiary Account No.	Beneficiary Bank	Amount	Status	Reference	Reason Code	Payment Time
2	5	12	Lisa Coffee Shop	36340961300	ICICI Bank	600.0	success	New stock of coffee beans	Payment Processed Successfully	2023-04-13 20:55:40

Accounts Section with Searched Account Name



Account ID	User ID	Account Number	Account Name	Account Type	Balance	Creation Time	Update Time
4	12	6375398523	Sharvil's Bakery	business	2266.38	2023-04-08 00:44:10	2023-05-18 01:34:23
5	12	7946529732	Sam's Cafetaria	business	5750.00	2023-04-08 01:43:09	2023-05-18 01:36:24
6	12	2159748680	Merry's Muffins	savings	12110.00	2023-04-14 02:08:30	2023-05-18 01:37:47
7	12	4690446800	Sharvil's Clothing Line	savings	65000.00	2023-04-16 20:48:08	2023-04-16 20:48:08
8	13	4119514483	Personal	savings	1536888.51	2023-04-17 19:53:49	2023-05-17 05:26:12
9	14	2453088053	Shopping	savings	60800.00	2023-04-18 00:27:36	2023-04-18 00:27:36
10	12	1054803813	Car	Savings	0.00	2023-04-18 03:08:30	2023-04-18 03:08:30
11	13	6328049877	Familt	Savings	1000.00	2023-04-18 18:07:48	2023-04-18 18:07:48
13	22	3115493485	Family	Savings	0.00	2023-05-04 23:59:59	2023-05-04 23:59:59
16	23	3578307335	Clothing Line	Business	50000.00	2023-05-18 01:25:41	2023-05-18 01:49:27

6.0 Coding

The coding section refers to the phase of software development where programmers write instructions or code to create computer programs or applications. Coding involves translating the logic and requirements of a software project into a programming language that a computer can understand and execute. The following section contains the code written to construct the Banking Management System.

NOTE: Please visit the GitHub Repository for a well detailed and structured coding section: https://github.com/Sharvil18/Cadmus_Financial_Corporation.git

6.1 Setup Files

application.properties -

```
spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/bank

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.servlet.multipart.max-file-size=100MB

spring.servlet.multipart.max-request-size=100MB


# Server Attributes / Config:

server.address=127.0.0.1

server.port=8070


# Disable Default Error Page:

server.error.whitelabel.enabled=false
```

pom.xml -

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>3.0.5</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.bank</groupId>
        <artifactId>demoBank</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>demoBank</name>
        <description>Banking Management System</description>
        <properties>
                <java.version>17</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-mail</artifactId>
```

```
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.3.2</version>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
```

```
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
        <version>2.4.1</version>
</dependency>
<dependency>
        <groupId>jakarta.servlet.jsp.jstl</groupId>
        <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
        <version>2.0.0</version>
</dependency>
<dependency>
        <groupId>org.glassfish.web</groupId>
        <artifactId>jakarta.servlet.jsp.jstl</artifactId>
        <version>2.0.0</version>
</dependency>
<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-crypto</artifactId>
        <version>5.4.2</version>
</dependency>
<dependency>
        <groupId>com.github.librepdf</groupId>
        <artifactId>openpdf</artifactId>
        <version>1.3.8</version>
</dependency>
<dependency>
```

```
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <optional>true</optional>
            </dependency>
    </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>
</project>
```

database_script.sql -

```
--GO TO my.ini CONFIGURATION FILE OF DATABASE AND MAKE THE
FOLLOWING CHANGE:
--max_allowed_packet = 128*1024*1024
DROP DATABASE IF EXISTS bank;
CREATE DATABASE bank;
USE bank;
-- user table structure
CREATE TABLE users(
   user_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
```

```
    email VARCHAR(200) NOT NULL UNIQUE,
    password VARCHAR(200) NOT NULL,
    token VARCHAR(200) NULL,
    code INT NULL,
    verified INT DEFAULT 0,
    verified_at DATETIME,
    created_at TIMESTAMP,
    updated_at TIMESTAMP DEFAULT NOW()
);
-- bank accounts table structure
CREATE TABLE accounts(
    account_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    account_number VARCHAR(100) NOT NULL,
    account_name VARCHAR(50) NOT NULL,
    account_type VARCHAR(50) NOT NULL,
    balance DECIMAL(18, 2) DEFAULT 0.00,
    created_at TIMESTAMP,
    updated_at TIMESTAMP DEFAULT NOW(),
    FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE
CASCADE
);
-- transaction history table
CREATE TABLE transaction_history(
    transaction_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(50) NOT NULL,
    amount DECIMAL(18, 2),
    source VARCHAR(50) NULL,
```

```
    status VARCHAR(50) NULL,
    reason_code VARCHAR(100) NULL,
    created_at TIMESTAMP,
    FOREIGN KEY(account_id) REFERENCES accounts(account_id) ON
DELETE CASCADE
);
--  payment table structure
CREATE TABLE payments(
    payment_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    account_id INT,
    beneficiary VARCHAR(50) NULL,
    beneficiary_acc_no VARCHAR(200) NULL,
    amount DECIMAL(18, 2) NULL,
    reference_no VARCHAR(100) NULL,
    status VARCHAR(50) NULL,
    reason_code VARCHAR(100) NULL,
    created_at TIMESTAMP,
    FOREIGN KEY(account_id) REFERENCES accounts(account_id) ON
DELETE CASCADE
);
-- transaction history view
CREATE VIEW v_transaction_history
AS
    SELECT t.transaction_id, a.account_id, u.user_id, t.transaction_type, t.amount,
t.source, t.status, t.reason_code, t.created_at
    FROM transaction_history AS t
    INNER JOIN accounts AS a
    ON t.account_id = a.account_id
    INNER JOIN users as u
```

```
    ON a.user_Id = u.user_id;
-- payment history view
CREATE VIEW v_payments
AS
    SELECT p.payment_id, a.account_id, u.user_id, p.beneficiary,
p.beneficiary_acc_no, p.amount, p.status, p.reference_no, p.reason_code,
p.created_at
    FROM payments AS p
    INNER JOIN accounts as a
    ON p.account_id = a.account_id
    INNER JOIN users as u
    ON u.user_id = a.user_id;

select * from v_transaction_history;

select * from v_payments;

-- ADMIN TABLE CREATION
-- department table
CREATE TABLE department (
    department_id INT PRIMARY KEY,
    department_name INT NOT NULL,
    department_head VARCHAR(255) NOT NULL,
    department_description VARCHAR(255));
--branch table
CREATE TABLE branch (
    branch_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
```

```
    city VARCHAR(255) NOT NULL,
    state VARCHAR(255) NOT NULL,
    contact VARCHAR(255) NOT NULL,
    manager_id VARCHAR(255) NOT NULL);
--employee table
CREATE TABLE employee (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    contact VARCHAR(20) NOT NULL,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(100) NOT NULL,
    state VARCHAR(100) NOT NULL,
    postal_code VARCHAR(20) NOT NULL,
    hire_date DATE NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(50) NOT NULL,
    marital_status VARCHAR(100) NOT NULL,
    education_level VARCHAR(100) NOT NULL,
    primary_language VARCHAR(100) NOT NULL,
    subsidiary_language VARCHAR(255) NOT NULL,
    employee_type VARCHAR(100) NOT NULL,
    employee_status VARCHAR(100) NOT NULL,
    job_title VARCHAR(100) NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT NOT NULL,
    branch_id INT NOT NULL
```

```
    manager_id INT,
    FOREIGN_KEY(manager_id) REFERENCES employee(employee_id),
    FOREIGN_KEY(department_id) REFERENCES department(department_id),
    FOREIGN_KEY(branch_id) REFERENCES branch(branch_id));
```

admin_side_script.sql -

```
CREATE TABLE employee (
    employee_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    contact VARCHAR(20) NOT NULL,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(100) NOT NULL,
    state VARCHAR(100) NOT NULL,
    postal_code VARCHAR(20) NOT NULL,
    hire_date DATE NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(50) NOT NULL,
    marital_status VARCHAR(100) NOT NULL,
    education_level VARCHAR(100) NOT NULL,
    primary_language VARCHAR(100) NOT NULL,
    subsidiary_language VARCHAR(255) NOT NULL,
    employee_type VARCHAR(100) NOT NULL,
    employee_status VARCHAR(100) NOT NULL,
    job_title VARCHAR(100) NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT NOT NULL,
```

```
    branch_id INT NOT NULL,
    manager_id INT,
    FOREIGN KEY(manager_id) REFERENCES department(manager_id),
    FOREIGN KEY(department_id) REFERENCES department(department_id)
ON DELETE CASCADE,
    FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE
CASCADE);
CREATE TABLE department (
    department_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(255) NOT NULL,
    department_head VARCHAR(255) NOT NULL,
    department_description VARCHAR(255));
CREATE TABLE branch (
    branch_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(255) NOT NULL,
    state VARCHAR(255) NOT NULL,
    contact VARCHAR(255) NOT NULL,
    manager_id INT NOT NULL,
    FOREIGN KEY(manager_id) REFERENCES employee(employee_id));
INSERT INTO department (department_name, department_head,
department_description) VALUES
('Accounting', 'Deepak Kumar', 'Responsible for maintaining the financial records
and transactions of the bank.'),
('Audit', 'Rajat Sharma', 'Responsible for monitoring and ensuring compliance
with the bank\'s policies and procedures.'),
('Branch Operations', 'Neeraj Verma', 'Responsible for managing the day-to-day
operations of the bank branch.'),
('Customer Service', 'Priya Sharma', 'Responsible for providing high-quality
customer service and support to clients.'),
```

('Finance', 'Amit Patel', 'Responsible for managing the bank\'s financial resources and investments.'),

('Human Resources', 'Sonia Gupta', 'Responsible for managing employee relations, recruitment, and compensation.'),

('Information Technology', 'Vikram Singh', 'Responsible for managing the bank\'s technology infrastructure and systems.'),

('Marketing', 'Amita Singh', 'Responsible for creating and executing marketing strategies to promote bank services and products.'),

('Sales', 'Sudhir Gupta', 'Responsible for generating revenue and sales for the bank.');

INSERT INTO branch (name, address, city, state, contact, manager_id) VALUES

('Mumbai Central', 'Cathedral Road, Dadar', 'Mumbai', 'Maharashtra', '9876543210', 23),

('Hyderabad Central', 'Sapphire Street, Banjara Hills', 'Hyderabad', 'Telangana', '4078901234', 53),

('Ahmedabad West', 'Skyline Heights, CG Road', 'Ahmedabad', 'Gujarat', '7989012345', 17),

('Pune East', 'Coral Avenue, Koregaon Park', 'Pune', 'Maharashtra', '2090123456', 82),

('Chandigarh City', 'Royal Gardens, Sector 17', 'Chandigarh', 'Punjab', '1723456789', 64),

('New Delhi Main', 'Diamond Drive, Connaught Place', 'New Delhi', 'Delhi', '1123456789', 77),

('Coimbatore West', 'Silver Springs, Race Course Road', 'Coimbatore', 'Tamil Nadu', '4228901234', 59);

-- insert employees for accountant department

INSERT INTO employee (first_name, last_name, email, contact, address, city, state, postal_code, hire_date, age, gender, marital_status, education_level, primary_language, subsidiary_language, employee_type, employee_status, job_title, salary, department_id, branch_id) VALUES

('Akash', 'Gupta', 'akash.gupta@gmail.com', '9876543210', '10, MG Road', 'Bangalore', 'Karnataka', '560001', '2022-05-01', 27, 'Male', 'Married', 'Post Graduate', 'Hindi', 'English', 'Full Time', 'Active', 'Financial Analyst', 75000, 1, 8),

('Amit', 'Shah', 'amit.shah@gmail.com', '9876543211', '11, Nehru Road', 'Mumbai', 'Maharashtra', '400001', '2022-05-01', 29, 'Male', 'Married', 'Post Graduate', 'Hindi', 'English', 'Full Time', 'Active', 'Accountant', 45000, 1, 1),

('Shruti', 'Patil', 'shruti.patil@gmail.com', '9876543212', '12, Karve Road', 'Pune', 'Maharashtra', '411001', '2022-05-01', 25, 'Female', 'Single', 'Graduate', 'Marathi', 'English', 'Full Time', 'Active', 'Accountant', 40000, 1, 1),

-- insert employees for operation department

INSERT INTO employee (first_name, last_name, email, contact, address, city, state, postal_code, hire_date, age, gender, marital_status, education_level, primary_language, subsidiary_language, employee_type, employee_status, job_title, salary, department_id, branch_id) VALUES

('Ravi', 'Narang', 'ravi.narang@gmail.com', '9876543224', '24, Ashok Vihar', 'Delhi', 'Delhi', '110052', '2022-05-03', 32, 'Male', 'Married', 'Post Graduate', 'Hindi', 'English', 'Full Time', 'Active', 'Teller', 40000, 3, 1),

('Sneha', 'Mehta', 'sneha.mehta@gmail.com', '9876543225', '25, Kandivali', 'Mumbai', 'Maharashtra', '400101', '2022-05-03', 29, 'Female', 'Single', 'Post Graduate', 'Marathi', 'English', 'Full Time', 'Active', 'Teller', 40000, 3, 1),

('Aditya', 'Joshi', 'aditya.joshi@gmail.com', '9876543226', '26, Maninagar', 'Ahmedabad', 'Gujarat', '380008', '2022-05-03', 27, 'Male', 'Single', 'Graduate', 'Gujarati', 'English', 'Full Time', 'Active', 'Teller', 40000, 3, 3),

-- insert employees for finance department

INSERT INTO employee (first_name, last_name, email, contact, address, city, state, postal_code, hire_date, age, gender, marital_status, education_level, primary_language, subsidiary_language, employee_type, employee_status, job_title, salary, department_id, branch_id) VALUES

('Pallavi', 'Ahuja', 'pallavi.ahuja@gmail.com', '9876543201', '1, Green Park', 'Delhi', 'Delhi', '110016', '2022-05-02', 26, 'Female', 'Single', 'Graduate', 'Hindi', 'English', 'Part Time', 'Active', 'Investment Banking Analyst', 70000, 6, 11),

('Rajat', 'Verma', 'rajat.verma@gmail.com', '9876543202', '2, Shankar Nagar', 'Nagpur', 'Maharashtra', '440010', '2022-05-02', 30, 'Male', 'Married', 'Post Graduate', 'Marathi', 'English', 'Full Time', 'Active', 'Investment Banking Associate', 85000, 6, 10),

('Arjun', 'Kapoor', 'arjun.kapoor@gmail.com', '9876543203', '3, Ellis Road', 'Chennai', 'Tamil Nadu', '600002', '2022-05-02', 28, 'Male', 'Single', 'Post Graduate', 'Tamil', 'English', 'Full Time', 'Active', 'Asset Management', 75000, 6, 12),

6.2 Configuration Files

AppConfig.java -

```
package com.bank.config;

import com.bank.interceptors.AppInterceptor;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

import
org.springframework.web.servlet.config.annotation.WebMvcConfigurationSuppor
t;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

import org.springframework.web.servlet.view.JstlView;

@Configuration

@ComponentScan(basePackages = {"com.bank"})

public class AppConfig extends WebMvcConfigurationSupport {

   @Override

   protected void addResourceHandlers(ResourceHandlerRegistry registry) {

      registry.addResourceHandler("css/**","images/**", "js/**",

                  "/app/css/**" , "/app/images/**", "/app/js/**",

                  "/app/dashboard/css/**" , "/app/dashboard/images/**",
"/app/dashboard/js/**",

                  "/account/css/**" , "/account/images/**", "/account/js/**",

                  "/login/css/**" , "/login/images/**", "/login/js/**",

                  "/transact/css/**" , "/transact/images/**", "/transact/js/**",

                  "/apply/css/**" , "/apply/images/**", "/apply/js/**",

                  "/app/admin/css/**" , "/app/admin/images/**", "/app/admin/js/**"
```

```
                )
            .addResourceLocations("classpath:/static/css/",
"classpath:/static/images/", "classpath:/static/js/");
    }
    // End Of Resource Handler.
    @Bean
    public InternalResourceViewResolver viewResolver(){
        InternalResourceViewResolver jspViewResolver = new
InternalResourceViewResolver();
        jspViewResolver.setPrefix("/WEB-INF/jsp/");
        jspViewResolver.setSuffix(".jsp");
        jspViewResolver.setViewClass(JstlView.class);

        return jspViewResolver;
    }
    @Override
    protected void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new AppInterceptor()).addPathPatterns("/app/*");
    }
}
```

MailConfig.java -

```
package com.bank.config;
import jakarta.mail.Authenticator;
import jakarta.mail.Session;
import org.springframework.context.annotation.Bean;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import java.util.Properties;

public class MailConfig {
   @Bean
   public static JavaMailSenderImpl getMailConfig() {
      JavaMailSenderImpl emailConfig = new JavaMailSenderImpl();
      Properties props = emailConfig.getJavaMailProperties();
      props.setProperty("mail.smtp.host", "smtp.gmail.com");
      props.setProperty("mail.smtp.port", "465");
      props.setProperty("mail.smtp.auth", "true");
      props.setProperty("mail.smtp.ssl.enable", "true");
      emailConfig.setUsername("cadmus.finance.corp@gmail.com");
      emailConfig.setPassword("cghm ssum irce npjf");
      return emailConfig;
   }
}
```

6.3 Miscellaneous Files

AdvisorController.java -

```
package com.bank.controller_advisor;
import com.bank.models.Admin;
import com.bank.models.User;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ModelAttribute;
@ControllerAdvice
public class AdvisorController {
   @ModelAttribute("registerUser")
   public User getUserDefaults() {
      return new User();
   }
}
```

MailMessenger.java -

```
package com.bank.mailMessenger;
import com.bank.config.MailConfig;
import com.oracle.wls.shaded.org.apache.bcel.generic.NEW;
import jakarta.mail.MessagingException;
import jakarta.mail.internet.MimeMessage;
import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.InputStreamResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.xml.sax.InputSource;
import java.io.ByteArrayInputStream;
```

```
public class MailMessenger {

   public static void htmlEmailMessenger(String from, String toMail, String
subject, String body, byte[] pdfBytes) {

      //Get email config:

      JavaMailSender sender = MailConfig.getMailConfig();

      //Set mime message

      MimeMessage message = sender.createMimeMessage();

      try {

         //Set mime message helper

         MimeMessageHelper htmlMessage = new MimeMessageHelper(message,
true);

         //Set mail attributes/properties

         htmlMessage.setTo(toMail);

         htmlMessage.setFrom(from);

         htmlMessage.setSubject(subject);

         htmlMessage.setText(body, true);

         if(pdfBytes!= null)

            htmlMessage.addAttachment("details.pdf", new
ByteArrayResource(pdfBytes));

      } catch (MessagingException e) {

         throw new RuntimeException(e);

      }

      //Send message

      sender.send(message);

   }

}
```

DemoBankApplication.java -

```
package com.bank;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.security.crypto.bcrypt.BCrypt;
import java.util.ArrayList;
import java.util.List;
@SpringBootApplication
@EnableScheduling
public class DemoBankApplication {
	public static void main(String[] args) {
		SpringApplication.run(DemoBankApplication.class, args);
	}
}
```

6.4 Models

Account.java -

```
@Entity
@Table(name = "accounts")
public class Account {
   @Id
   private int account_id;
   private int user_id;
   private String account_number;
   private String account_name;
   private String account_type;
   private BigDecimal balance;
   private LocalDateTime created_at;
   private LocalDateTime updated_at;
   public int getAccount_id() {
      return account_id;
   }
   public void setAccount_id(int account_id) {
      this.account_id = account_id;
   }
   public int getUser_id() {
      return user_id;
   }
   public void setUser_id(int user_id) {
      this.user_id = user_id;
   }
   public String getAccount_number() {
      return account_number;
```

```
}
public void setAccount_number(String account_number) {
    this.account_number = account_number;
}
public String getAccount_name() {
    return account_name;
}
public void setAccount_name(String account_name) {
    this.account_name = account_name;
}
public String getAccount_type() {
    return account_type;
}
public void setAccount_type(String account_type) {
    this.account_type = account_type;
}
public BigDecimal getBalance() {
    return balance;
}
public void setBalance(BigDecimal balance) {
    this.balance = balance;
}
public LocalDateTime getCreated_at() {
    return created_at;
}
public void setCreated_at(LocalDateTime created_at) {
    this.created_at = created_at;
}
```

```
    public LocalDateTime getUpdated_at() {
        return updated_at;
    }
    public void setUpdated_at(LocalDateTime updated_at) {
        this.updated_at = updated_at;
    }
}
```

Admin.java -

```
@Entity
public class Admin {
    @Id
    private String admin_id;
    private String admin_email;
    private String admin_first_name;
    private String admin_last_name;
    private String admin_password;
    private LocalDateTime admin_last_login;
    public String getAdmin_id() {
        return admin_id;
    }
    public void setAdmin_id(String admin_id) {
        this.admin_id = admin_id;
    }
    public String getAdmin_email() {
        return admin_email;
    }
    public void setAdmin_email(String admin_email) {
```

```
        this.admin_email = admin_email;
    }
    public String getAdmin_first_name() {
        return admin_first_name;
    }
    public void setAdmin_first_name(String admin_first_name) {
        this.admin_first_name = admin_first_name;
    }
    public String getAdmin_last_name() {
        return admin_last_name;
    }
    public void setAdmin_last_name(String admin_last_name) {
        this.admin_last_name = admin_last_name;
    }
    public String getAdmin_password() {
        return admin_password;
    }
    public void setAdmin_password(String admin_password) {
        this.admin_password = admin_password;
    }
    public LocalDateTime getAdmin_last_login() {
        return admin_last_login;
    }
    public void setAdmin_last_login(LocalDateTime admin_last_login) {
        this.admin_last_login = admin_last_login;
    }
}
```

Branch.java -

```
public class Branch {
   @Id
   private int branch_id;
   private String name;
   private String address;
   private String city;
   private String state;
   private String contact;
   private int manager_id;
   public int getBranch_id() {
      return branch_id;
   }
   public void setBranch_id(int branch_id) {
      this.branch_id = branch_id;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getAddress() {
      return address;
   }
   public void setAddress(String address) {
      this.address = address;
   }
```

```
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
    public String getContact() {
        return contact;
    }
    public void setContact(String contact) {
        this.contact = contact;
    }
    public int getManager_id() {
        return manager_id;
    }
    public void setManager_id(int manager_id) {
        this.manager_id = manager_id;
    }
}
```

Customer.java -

```
@Entity
@Table(name = "v_customer")
public class Customer {
   @Id
   private int user_id;
   private String name;
   private String email;
   private LocalDateTime created_at;
   private LocalDateTime updated_at;
   private LocalDateTime verified_at;
   private int account_count;
   public int getUser_id() {
      return user_id;
   }
   public void setUser_id(int user_id) {
      this.user_id = user_id;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getEmail() {
      return email;
   }
   public void setEmail(String email) {
```

```
        this.email = email;
    }
    public LocalDateTime getCreated_at() {
        return created_at;
    }
    public void setCreated_at(LocalDateTime created_at) {
        this.created_at = created_at;
    }

    public LocalDateTime getUpdated_at() {
        return updated_at;
    }
    public void setUpdated_at(LocalDateTime updated_at) {
        this.updated_at = updated_at;
    }
    public LocalDateTime getVerified_at() {
        return verified_at;
    }
    public void setVerified_at(LocalDateTime verified_at) {
        this.verified_at = verified_at;
    }
    public int getAccount_count() {
        return account_count;
    }
    public void setAccount_count(int account_count) {
        this.account_count = account_count;
    }
}
```

Department.java -

```
@Entity
public class Department {
   @Id
   private int department_id;
   private String department_name;
   private String department_head;
   private String department_description;
   private int department_head_id;
   public int getDepartment_id() {
      return department_id;
   }
   public void setDepartment_id(int department_id) {
      this.department_id = department_id;
   }
   public String getDepartment_name() {
      return department_name;
   }
   public void setDepartment_name(String department_name) {
      this.department_name = department_name;
   }
   public String getDepartment_head() {
      return department_head;
   }
   public void setDepartment_head(String department_head) {
      this.department_head = department_head;
   }
   public String getDepartment_description() {
```

```
        return department_description;
    }
    public void setDepartment_description(String department_description) {
        this.department_description = department_description;
    }
    public int getDepartment_head_id() {
        return department_head_id;
    }
    public void setDepartment_head_id(int department_head_id) {
        this.department_head_id = department_head_id;
    }
}
```

Employee.java -

```
@Entity
public class Employee {
    @Id
    private int employee_id;
    private String first_name;
    private String last_name;
    private String email;
    private String contact;
    private String address;
    private String city;
    private String state;
    private String postal_code;
    private LocalDate hire_date;
    private int age;
```

```
private String gender;
private String marital_status;
private String education_level;
private String primary_language;
private String subsidiary_language;
private String employee_type;
private String employee_status;
private String job_title;
private double salary;
private int department_id;
private int branch_id;
private Integer manager_id;
public int getEmployee_id() {
   return employee_id;
}
public void setEmployee_id(int employee_id) {
   this.employee_id = employee_id;
}
public String getFirst_name() {
   return first_name;
}
public void setFirst_name(String first_name) {
   this.first_name = first_name;
}
public String getLast_name() {
   return last_name;
}
public void setLast_name(String last_name) {
```

```
    this.last_name = last_name;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
public String getContact() {
    return contact;
}
public void setContact(String contact) {
    this.contact = contact;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}

public String getCity() {
    return city;
}
public void setCity(String city) {
    this.city = city;
}
public String getState() {
```

```
        return state;
    }
}
```

EmployeeView.java -

```
@Entity
@Table(name = "v_employee")
public class EmployeeView {
    @Id
    private int employee_id;
    private String Name;
    private int age;
    private String job_title;
    private LocalDate hire_date;
    private String department_name;
    private String branch_name;
    public int getEmployee_id() {
        return employee_id;
    }
    public void setEmployee_id(int employee_id) {
        this.employee_id = employee_id;
    }
    public String getName() {
        return Name;
    }
    public void setName(String name) {
        Name = name;
    }
```

```
public int getAge() {
    return age;
}
public void setAge(int age) {
    this.age = age;
}
public String getJob_title() {
    return job_title;
}
public LocalDate getHire_date() {
    return hire_date;
}
public void setHire_date(LocalDate hire_date) {
    this.hire_date = hire_date;
}
public void setJob_title(String job_title) {
    this.job_title = job_title;
}
public String getDepartment_name() {
    return department_name;
}
public void setDepartment_name(String department_name) {
    this.department_name = department_name;
}
public String getBranch_name() {
    return branch_name;
}
public void setBranch_name(String branch_name) {
```

```
        this.branch_name = branch_name;
    }
}
```

LoanLog.java -

```
@Entity
@Table(name = "loan_log")
public class LoanLog {
    @Id
    private String loan_application_number;
    private String loan_type;
    private String borrower_name;
    private int borrower_user_id;
    private String borrower_email;
    private double final_loan_amount;
    private double final_interest_rate;
    private int final_tenure;
    private double final_emi;
    private double total_amount_payable;
    private double total_interest_payable;
    private double charges_payable;
    private double late_payment_penalty;
    private double pre_payment_penalty;
    private LocalDateTime created_at;
    private LocalDateTime updated_at;
    private int borrower_account_id;
    private double total_penalty_payable;
    private String is_confirmed;
```

```
public String getLoan_application_number() {
   return loan_application_number;
}
public void setLoan_application_number(String loan_application_number) {
   this.loan_application_number = loan_application_number;
}
public String getLoan_type() {
   return loan_type;
}
public void setLoan_type(String loan_type) {
   this.loan_type = loan_type;
}
public String getBorrower_name() {
   return borrower_name;
}
public void setBorrower_name(String borrower_name) {
   this.borrower_name = borrower_name;
}
public int getBorrower_user_id() {
   return borrower_user_id;
}
public void setBorrower_user_id(int borrower_user_id) {
   this.borrower_user_id = borrower_user_id;
}
public String getBorrower_email() {
   return borrower_email;
}
public void setBorrower_email(String borrower_email) {
```

```
        this.borrower_email = borrower_email;
    }
    public double getFinal_loan_amount() {
        return final_loan_amount;
    }
    public void setFinal_loan_amount(double final_loan_amount) {
        this.final_loan_amount = final_loan_amount;
    }

}
```

Payment.java -

```
@Entity
@Table(name = "payments")
public class Payment {
    @Id
    private int payment_id;
    private int account_id;
    private String beneficiary;
    private String beneficiary_acc_no;
    private String beneficiary_bank;
    private double amount;
    private String reference_no;
    private String status;
    private String reason_code;
    private LocalDateTime created_at;
    public int getPayment_id() {
        return payment_id;
```

```
    }
    public void setPayment_id(int payment_id) {
        this.payment_id = payment_id;
    }
    public int getAccount_id() {
        return account_id;
    }
    public void setAccount_id(int account_id) {
        this.account_id = account_id;
    }
    public String getBeneficiary() {
        return beneficiary;
    }
    public void setBeneficiary(String beneficiary) {
        this.beneficiary = beneficiary;
    }
    public String getBeneficiary_acc_no() {
        return beneficiary_acc_no;

    public void setBeneficiary_acc_no(String beneficiary_acc_no) {
        this.beneficiary_acc_no = beneficiary_acc_no;
    }

}
```

PaymentHistory.java -

```
@Entity
@Table(name = "v_payments")
public class PaymentHistory {
   @Id
   private int payment_id;
   private int account_id;
   private int user_id;
   private String beneficiary;
   private String beneficiary_acc_no;
   private String beneficiary_bank;
   private double amount;
   private String status;
   private String reference_no;
   private String reason_code;
   private LocalDateTime created_at;
   public int getPayment_id() {
      return payment_id;
   }
   public void setPayment_id(int payment_id) {
      this.payment_id = payment_id;
   }
   public int getAccount_id() {
      return account_id;
   }
   public void setAccount_id(int account_id) {
      this.account_id = account_id;
   }
```

```
    public int getUser_id() {
        return user_id;
    }
    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }
    public String getBeneficiary() {
        return beneficiary;
    }
    public void setBeneficiary(String beneficiary) {
        this.beneficiary = beneficiary;
    }
    public String getBeneficiary_acc_no() {
        return beneficiary_acc_no;
    }
    public void setBeneficiary_acc_no(String beneficiary_acc_no) {
        this.beneficiary_acc_no = beneficiary_acc_no;
    }
}
```

Transact.java -

```
@Entity
@Table(name = "transaction_history")
public class Transact {
    @Id
    private int transaction_id;
    private int account_id;
    private String transaction_type;
```

```
private double amount;
private String source;
private String status;
private String reason_code;
private String created_at;
public int getTransaction_id() {
   return transaction_id;
}
public void setTransaction_id(int transaction_id) {
   this.transaction_id = transaction_id;
}
public int getAccount_id() {
   return account_id;
}
public void setAccount_id(int account_id) {
   this.account_id = account_id;
}
public String getTransaction_type() {
   return transaction_type;
}
public void setTransaction_type(String transaction_type) {
   this.transaction_type = transaction_type;
}
public double getAmount() {
   return amount;
}
public void setAmount(double amount) {
   this.amount = amount;
```

```
    }
    public String getSource() {
        return source;
    }
    public void setSource(String source) {
        this.source = source;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public String getReason_code() {
        return reason_code;
    }
    public void setReason_code(String reason_code) {
        this.reason_code = reason_code;
    }
    public String getCreated_at() {
        return created_at;
    }
    public void setCreated_at(String created_at) {
        this.created_at = created_at;
    }
}
```

TransactionHistory.java -

```
@Entity
@Table(name = "v_transaction_history")
public class TransactionHistory {
   @Id
   private int transaction_id;
   private int account_id;
   private int user_id;
   private String transaction_type;
   private double amount;
   private String  source;
   private String  status;
   private String  reason_code;
   private LocalDateTime created_at;
   public int getTransaction_id() {
      return transaction_id;
   }
   public void setTransaction_id(int transaction_id) {
      this.transaction_id = transaction_id;
   }
   public int getAccount_id() {
      return account_id;
   }
   public void setAccount_id(int account_id) {
      this.account_id = account_id;
   }
   public int getUser_id() {
      return user_id;
```

```
    }
    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }
    public String getTransaction_type() {
        return transaction_type;
    }
    public void setTransaction_type(String transaction_type) {
        this.transaction_type = transaction_type;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
}
```

User.java -

```
@Entity
@Table(name = "users")
public class User {
    @Id
    private int user_id;
    @NotEmpty(message = "First name cannot be empty")
    @Size(min=3, message = "First name must be greater than 3 characters")
    private String first_name;
    @NotEmpty(message = "Last name cannot be empty")
```

```
    @Size(min=3, message = "Last name must be greater than 3 characters")
    private String last_name;
    @Email
    @NotEmpty(message="Email cannot be empty")
    @Pattern(regexp = "([a-zA-Z0-9]+(?:[._+-][a-zA-Z0-9]+)*)@([a-zA-Z0-9]+(?:[.-][a-zA-Z0-9]+)*[.][a-zA-Z]{2,})", message = "Please enter a valid email")
    private String email;
    @NotEmpty(message="Password cannot empty")
    @NotNull
    private String password;
    private String token;
    private String code;
    private int verified;
    private LocalDate verified_at;
    private LocalDateTime created_at;
    private LocalDateTime updated_at;
    public int getUser_id() {
        return user_id;
    }
    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }
    public String getFirst_name() {
        return first_name;
    }
    public void setFirst_name(String first_name) {
        this.first_name = first_name;
    }
```

```
    public String getLast_name() {
        return last_name;
    }
    public void setLast_name(String last_name) {
        this.last_name = last_name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

UserView.java -

```
@Entity
@Table(name = "v_users")
public class UserView {
    @Id
    private int user_id;
    private String name;
    private String email;
    private int account_count;
    public int getUser_id() {
        return user_id;
    }
    public void setUser_id(int user_id) {
        this.user_id = user_id;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public int getAccount_count() {
        return account_count;
    }

    public void setAccount_count(int account_count) {
        this.account_count = account_count;
    }
}
```

6.5 Repositories

AccountRepository.java -

```
package com.bank.repository;

@Repository

public interface AccountRepository extends CrudRepository<Account, Integer> {

   @Query(value = "SELECT * FROM accounts WHERE user_id = :user_id",
nativeQuery = true)

   List<Account> getUserAccountsById(@Param("user_id")int user_id);

   @Query(value = "SELECT * FROM accounts", nativeQuery = true)

   List<Account> getAllAccounts();

   @Query(value = "SELECT sum(balance) FROM accounts WHERE user_id =
:user_id", nativeQuery = true)

   BigDecimal getTotalBalance(@Param("user_id") int user_id);

   @Query(value = "SELECT balance FROM accounts WHERE user_id =
:user_id AND account_id = :account_id", nativeQuery = true)

   Double getAccountBalance(@Param("user_id") int user_id,
@Param("account_id") int account_id);

   @Query(value = "SELECT balance FROM accounts WHERE account_number
= :account_number", nativeQuery = true)

   Double getBalanceByAccountNumber(@Param("account_number") String
account_number);

   @Query(value = "SELECT account_id FROM accounts WHERE
account_number = :account_number", nativeQuery = true)

   int getAccountIdByAccountNumber(@Param("account_number") String
account_number);

   @Query(value = "SELECT account_number FROM accounts", nativeQuery =
true)

   List<String> getAllAccountNumber();

   @Query(value = "SELECT account_number FROM accounts WHERE user_id
= :user_id", nativeQuery = true)

   List<String> getAccountNumberById(@Param("user_id") int user_id);
```

```
    @Modifying
    @Transactional
    @Query(value = "UPDATE accounts set balance = :new_balance WHERE
account_id = :account_id", nativeQuery = true)
    void changeAccountBalanceById(@Param("new_balance") double
new_balance, @Param("account_id") int account_id);
    @Modifying
    @Transactional
    @Query(value = "UPDATE accounts set balance = :new_balance WHERE
account_number = :account_number", nativeQuery = true)
    void changeAccountBalanceByAccountNumber(@Param("new_balance")
double new_balance, @Param("account_number") String account_number);
    @Modifying
    @Transactional
    @Query(value = "UPDATE accounts set updated_at = NOW() WHERE
account_id = :account_id", nativeQuery = true)
    void setUpdatedAt(@Param("account_id") int account_id);
    @Modifying
    @Query(value = "INSERT INTO accounts(user_id, account_number,
account_name, account_type, created_at) VALUES" +
            "(:user_id, :account_number, :account_name, :account_type, NOW())",
nativeQuery = true)
    @Transactional
    void createBankAccount(@Param("user_id") int user_id,
                           @Param("account_number") String account_number,
                           @Param("account_name") String account_name,
                           @Param("account_type") String account_type);
}
```

AdminRepository.java -

```
package com.bank.repository;

@Repository

public interface AdminRepository extends CrudRepository<Admin, String> {

   @Query(value = "SELECT admin_id FROM admin", nativeQuery = true)

   List<String> getAllAdminID();

   @Query(value = "SELECT admin_email FROM admin WHERE admin_id =
:admin_id", nativeQuery = true)

   String getAdminEmail(@Param("admin_id") String admin_email);

   @Query(value = "SELECT admin_password FROM admin WHERE admin_id
= :admin_id", nativeQuery = true)

   String getAdminPassword(@Param("admin_id") String admin_id);

   @Query(value = "SELECT * FROM admin WHERE admin_id = :admin_id",
nativeQuery = true)

   Admin getAdminDetails(@Param("admin_id") String admin_id);

   @Modifying @Transactional

   @Query(value = "UPDATE admin SET admin_last_login = :admin_last_login
WHERE admin_id = :admin_id", nativeQuery = true)

   void updateAdminLastLogin(@Param("admin_last_login") LocalDateTime
admin_last_login, @Param("admin_id") String admin_id);

}
```

BranchRepository.java -

```
package com.bank.repository;

@Repository

public interface BranchRepository extends CrudRepository<Branch, Integer> {

   @Query(value = "SELECT name FROM branch WHERE
branch_id=:branch_id", nativeQuery = true)

   String getBranchNameById(@Param("branch_id") String branch_id);

   @Query(value = "SELECT * FROM branch", nativeQuery = true)
```

```
    List<Branch> getAllBranch();
}
```

CustomerRepository.java -

```
package com.bank.repository;

@Repository
public interface CustomerRepository extends CrudRepository<Customer,
Integer> {
    @Query(value = "SELECT * FROM v_customer", nativeQuery = true)
    List<Customer> getAllCustomers();
    @Query(value = "SELECT * FROM v_customer ORDER BY user_id",
nativeQuery = true)
    List<Customer> getCustomersOrderByID();
    @Query(value = "SELECT * FROM v_customer ORDER BY user_id DESC",
nativeQuery = true)
    List<Customer> getCustomersOrderByIDDesc();
}
```

DepartmentRepository.java -

```
package com.bank.repository;

@Repository
public interface DepartmentRepository extends CrudRepository<Department,
Integer> {
    @Query(value = "SELECT * FROM department", nativeQuery = true)
    List<Department> getAllDepartment();
    @Query(value = "SELECT department_name FROM department WHERE
department_id=:department_id", nativeQuery = true)
    String getDepartmentNameById(@Param("department_id") String
department_id);
}
```

EmployeeRepository.java -

```
package com.bank.repository;

@Repository

public interface EmployeeRepository extends CrudRepository<Employee,
Integer> {

   @Query(value = "SELECT * FROM employee ORDER BY hire_date",
nativeQuery = true)

   List<Employee> getAllEmployee();

   @Query(value = "SELECT * FROM employee ORDER BY employee_id",
nativeQuery = true)

   List<Employee> getAllEmployeeSortByID();

   @Query(value = "SELECT * FROM employee ORDER BY employee_id
DESC", nativeQuery = true)

   List<Employee> getAllEmployeeSortByIDDesc();

   @Query(value = "SELECT * FROM employee ORDER BY email",
nativeQuery = true)

   @Query(value="SELECT DISTINCT(city) FROM employee", nativeQuery =
true)

   List<String> getDistinctCity();

   @Query(value = "SELECT * FROM employee where city=:city", nativeQuery
= true)

   List<Employee> getSpecificCity(@Param("city") String city);

}
```

EmployeeViewRepository.java -

```
package com.bank.repository;

@Repository

public interface EmployeeViewRepository extends
CrudRepository<EmployeeView, Integer> {

   @Query(value = "SELECT * FROM v_employee", nativeQuery = true)

   List<EmployeeView> getEmployeeView();

}
```

HomeLoanApplicationRepository.java -

```
package com.bank.repository;

@Repository

public interface HomeApplicationRepository extends
CrudRepository<HomeLoanApplication, Integer> {

   @Query(value = "SELECT user_id FROM home_loan_application",
nativeQuery = true)

   List<Integer> getAllUsersAppliedHomeLoan();

   @Query(value = "SELECT * FROM home_loan_application", nativeQuery =
true)

   List<HomeLoanApplication> getAllHomeLoanApplications();

   @Query(value = "SELECT email FROM home_loan_application WHERE
application_number=:application_number", nativeQuery = true)

   String
getEmailHomeLoanApplicationByApplicationNumber(@Param("application_nu
mber") String application_number);

   @Query(value = "SELECT address_proof FROM home_loan_application
WHERE application_number=:application_number", nativeQuery = true)

   byte[]
getAddressProofHomeLoanApplicationByApplicationNumber(@Param("applicat
ion_number") String application_number);

   @Query(value = "SELECT proof_of_identity FROM home_loan_application
WHERE application_number=:application_number", nativeQuery = true)
```

```
    byte[]
getIdentityProofHomeLoanApplicationByApplicationNumber(@Param("applicati
on_number") String application_number);

    @Query(value = "SELECT salary_slip FROM home_loan_application
WHERE application_number=:application_number", nativeQuery = true)

    byte[]
getSalarySlipHomeLoanApplicationByApplicationNumber(@Param("application
_number") String application_number);

    @Query(value = "SELECT bank_account_statement FROM
home_loan_application WHERE application_number=:application_number",
nativeQuery = true)

    byte[]
getAccountStatementHomeLoanApplicationByApplicationNumber(@Param("app
lication_number") String application_number);

    @Query(value = "SELECT COUNT(*) FROM home_loan_application",
nativeQuery = true)

    int getHomeLoanApplicationCount();

    @Query(value = "SELECT * FROM home_loan_application WHERE
application_number=:application_number", nativeQuery = true)

    HomeLoanApplication
getHomeLoanApplicationByApplicationNumber(@Param("application_number")
String application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE home_loan_application SET approved='yes'
WHERE application_number=:application_number", nativeQuery = true)

    void
setApprovedToYesHomeLoanApplication(@Param("application_number") String
application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE home_loan_application SET confirm='yes'
WHERE application_number=:application_number", nativeQuery = true)
```

```
    void setConfirmToYesHomeLoanApplication(@Param("application_number")
String application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE home_loan_application SET
final_loan_amount=:final_loan_amount, final_interest_rate=:final_interest_rate,
final_tenure=:final_tenure, final_emi=:final_emi,
total_amount_payable=:total_amount_payable, " +

            "total_interest_payable=:total_interest_payable,
charges_payable=:charges_payable,
late_payment_penalty=:late_payment_penalty,
pre_payment_penalty=:pre_payment_penalty", nativeQuery = true)

    void calculateEMI(@Param("final_loan_amount") double final_loan_amount,

                    @Param("final_interest_rate") double final_interest_rate,

                    @Param("final_tenure") int final_tenure,

                    @Param("final_emi") double final_emi,

                    @Param("total_amount_payable") double total_amount_payable,

                    @Param("total_interest_payable") double total_interest_payable,

                    @Param("charges_payable") double charges_payable,

                    @Param("late_payment_penalty") double late_payment_penalty,

                    @Param("pre_payment_penalty") double pre_payment_penalty);

    @Modifying

    @Transactional

    @Query(value = "INSERT INTO home_loan_application(user_id,
application_number, first_name, last_name, email, contact, religion, age,
birth_date, marital_status, proof_of_identity, address_house, address_complex,
address_street_line1, address_street_line2, " +

            "address_city, address_state, address_postal_code, address_proof,
company_name, job_title, employment_status, employment_period, work_mode,
department, salary, industry, salary_slip, income, investment, credit_card_debt,
account, " +
```

```
        "bank_account_statement, purchase_price, loan_amount_requested,
down_payment_amount, current_home, sell_current, motto_purchase) VALUES "
+

        "(:user_id, :application_number, :first_name, :last_name, :email, :contact,
:religion, :age, :birth_date, :marital_status, :proof_of_identity, :address_house,
:address_complex, :address_street_line1, :address_street_line2, :address_city, " +

        ":address_state, :address_postal_code, :address_proof, :company_name,
:job_title, :employment_status, :employment_period, :work_mode, :department,
:salary, :industry, :salary_slip, :income, :investment, :credit_card_debt, " +

        ":account, :bank_account_statement, :purchase_price,
:loan_amount_requested, :down_payment_amount, :current_home, :sell_current,
:motto_purchase)", nativeQuery = true)

    void applyHomeLoan(@Param("user_id") int user_id,

                @Param("application_number") String application_number,

                @Param("first_name") String first_name,

                @Param("last_name") String last_name,

                @Param("email") String email,

                @Param("contact") String contact,

                @Param("religion") String religion,

                @Param("age") int age,

                @Param("birth_date") LocalDate birth_date,

                @Param("marital_status") String marital_status,

                @Param("proof_of_identity") byte[] proof_of_identity,

                @Param("address_house") String address_house,

                @Param("address_complex") String address_complex,

                @Param("address_street_line1") String address_street_line1,

                @Param("address_street_line2") String address_street_line2,

                @Param("address_city") String address_city,

                @Param("address_state") String address_state,

                @Param("address_postal_code") String address_postal_code,

                @Param("address_proof") byte[] address_proof,
```

```
                @Param("company_name") String company_name,
                @Param("job_title") String job_title,
                @Param("employment_status") String employment_status,
                @Param("employment_period") int employment_period,
                @Param("work_mode") String work_mode,
                @Param("department") String department,
                @Param("salary") double salary,
                @Param("industry") String industry,
                @Param("salary_slip") byte[] salary_slip,
                @Param("income") double income,
                @Param("investment") double investment,
                @Param("credit_card_debt") double credit_card_debt,
                @Param("account") int account,
                @Param("bank_account_statement") byte[]
bank_account_statement,
                @Param("purchase_price") double purchase_price,
                @Param("loan_amount_requested") double
loan_amount_requested,
                @Param("down_payment_amount") double
down_payment_amount,
                @Param("current_home") String current_home,
                @Param("sell_current") String sell_current,
                @Param("motto_purchase") String motto_purchase);
}
```

LoanLogRepository.java -

```
package com.bank.repository;
@Repository
public interface LoanLogRepository extends CrudRepository<LoanLog, String> {
```

```
    @Query(value = "SELECT * FROM loan_log", nativeQuery = true)

    List<LoanLog> getAllLoanLogs();

    @Query(value = "SELECT * FROM loan_log WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    LoanLog
getLoanLogsByApplicationNumber(@Param("loan_application_number") String
loan_application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE loan_log SET
total_amount_payable=:total_amount_payable WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    void
updateAmountPayableByApplicationNumber(@Param("total_amount_payable")
double total_amount_payable, @Param("loan_application_number") String
loan_application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE loan_log SET charges_payable=0 WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    void
updateChargesPayableByApplicationNumber(@Param("loan_application_numbe
r") String loan_application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE loan_log SET
total_penalty_payable=:total_penalty_payable WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    void
setPenaltyPayableByApplicationNumber(@Param("total_penalty_payable")
double total_penalty_payable, @Param("loan_application_number") String
loan_application_number);

    @Modifying

    @Transactional
```

```
    @Query(value = "UPDATE loan_log SET updated_at=:updated_at WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    void logUpdateTimeByApplicationNumber(@Param("updated_at")
LocalDateTime updated_at, @Param("loan_application_number") String
loan_application_number);

    @Modifying

    @Transactional

    @Query(value = "UPDATE loan_log SET is_confirmed='yes' WHERE
loan_application_number=:loan_application_number", nativeQuery = true)

    void
setConfirmedToYesByApplicationNumber(@Param("loan_application_number")
String loan_application_number);

    @Modifying

    @Transactional

    @Query(value = "INSERT INTO loan_log(loan_application_number,
loan_type, borrower_name, borrower_user_id, borrower_email,
final_loan_amount, final_interest_rate, final_tenure, final_emi,
total_amount_payable, total_interest_payable, " +

            "charges_payable, late_payment_penalty, pre_payment_penalty,
created_at, updated_at, borrower_account_id, is_confirmed) " +

            "VALUES(:loan_application_number, :loan_type, :borrower_name,
:borrower_user_id, :borrower_email, :final_loan_amount, :final_interest_rate,
:final_tenure, :final_emi, :total_amount_payable, :total_interest_payable, " +

            ":charges_payable, :late_payment_penalty, :pre_payment_penalty,
NOW(), NOW(), :borrower_account_id, :is_confirmed)", nativeQuery = true)

    void logLoan(@Param("loan_application_number") String
loan_application_number,

                @Param("loan_type") String loan_type,

                @Param("borrower_name") String borrower_name,

                @Param("borrower_user_id") int borrower_user_id,

                @Param("borrower_email") String borrower_email,

                @Param("final_loan_amount") double final_loan_amount,

                @Param("final_interest_rate") double final_interest_rate,
```

```
            @Param("final_tenure") int final_tenure,
            @Param("final_emi") double final_emi,
            @Param("total_amount_payable") double total_amount_payable,
            @Param("total_interest_payable") double total_interest_payable,
            @Param("charges_payable") double charges_payable,
            @Param("late_payment_penalty") double late_payment_penalty,
            @Param("pre_payment_penalty") double pre_payment_penalty,
            @Param("borrower_account_id") int borrower_account_id,
           @Param("is_confirmed") String is_confirmed);
}
```

PaymentHistoryRepository.java -

```
package com.bank.repository;

@Repository

public interface PaymentHistoryRepository extends
CrudRepository<PaymentHistory, Integer> {

   @Query(value = "SELECT * FROM v_payments where user_id = :user_id
ORDER BY created_at", nativeQuery = true)

   List<PaymentHistory> getPaymentRecordsById(@Param("user_id") int
user_id);

   @Query(value = "SELECT * FROM v_payments", nativeQuery = true)

   List<PaymentHistory> getAllPayments();

   @Query(value = "SELECT * FROM v_payments ORDER BY payment_id",
nativeQuery = true)

   List<PaymentHistory> getCustomersOrderByPaymentId();

   @Query(value = "SELECT * FROM v_payments ORDER BY payment_id
DESC", nativeQuery = true)

   List<PaymentHistory> getCustomersOrderByPaymentIdDesc();

   List<PaymentHistory> getCustomersOrderByPaymentTimeDesc();
```

```
    @Query(value="SELECT DISTINCT(account_id) FROM v_payments",
nativeQuery = true)

    List<String> getDistinctAccountId();

    @Query(value = "SELECT * FROM v_payments where
account_id=:account_id", nativeQuery = true)

    List<PaymentHistory> getSpecificAccountId(@Param("account_id") String
account_id);

}
```

PaymentRepository.java -

```
package com.bank.repository;

@Repository

public interface PaymentRepository extends CrudRepository<Payment, Integer>
{

    @Modifying

    @Query(value = "INSERT INTO payments(account_id, beneficiary,
beneficiary_acc_no, beneficiary_bank, amount, reference_no, status,
reason_code, created_at) " +

            "VALUES(:account_id, :beneficiary, :beneficiary_acc_no,
:beneficiary_bank, :amount, :reference_no, :status, :reason_code, :created_at)",
nativeQuery = true)

    @Transactional

    void makePayment(@Param("account_id") int account_id,

                    @Param("beneficiary") String beneficiary,

                    @Param("beneficiary_acc_no") String beneficiary_acc_no,

                    @Param("beneficiary_bank") String beneficiary_bank,

                    @Param("amount") double amount,

                    @Param("reference_no") String reference_no,

                    @Param("status") String status,

                    @Param("reason_code") String reason_code,

                    @Param("created_at") LocalDateTime created_at);
```

```
}
```

TransactHistoryRepository.java -

```
package com.bank.repository;

@Repository

public interface TransactHistoryRepository extends
CrudRepository<TransactionHistory, Integer> {

   @Query(value = "SELECT * FROM v_transaction_history WHERE user_id =
:user_id ORDER BY created_at", nativeQuery = true)

   List<TransactionHistory> getTransactionRecordsById(@Param("user_id")int
user_id);

   @Query(value = "SELECT * FROM v_transaction_history ORDER BY
created_at", nativeQuery = true)

   List<TransactionHistory> getAllTransactionRecords();

}
```

TransactRepository.java -

```
package com.bank.repository;

@Repository

public interface TransactRepository extends CrudRepository<Transact, Integer> {

   @Modifying

   @Transactional

   @Query(value = "INSERT INTO transaction_history(account_id,
transaction_type, amount, source, status, reason_code, created_at) " +

         "VALUES(:account_id, :transaction_type, :amount, :source, :status,
:reason_code, :created_at)", nativeQuery = true)

   void logTransaction(@Param("account_id") int account_id,

                  @Param("transaction_type") String transaction_type,

                  @Param("amount") double amount,

                  @Param("source") String source,
```

```
                @Param("status") String status,
                @Param("reason_code") String reason_code,
                @Param("created_at") LocalDateTime created_at);
}
```

UserRepository.java -

```
package com.bank.repository;

@Repository
public interface UserRepository extends CrudRepository<User, Integer> {
   @Query(value = "SELECT email FROM users WHERE email = :email",
nativeQuery = true)
   String getUserEmail(@Param("email") String email);
   @Query(value = "SELECT password FROM users WHERE email = :email",
nativeQuery = true)
   String getUserPassword(@Param("email") String email);
   @Query(value = "SELECT * FROM users WHERE email = :email",
nativeQuery = true)
   User getUserDetails(@Param("email") String email);
   @Query(value = "SELECT verified FROM users WHERE email = :email",
nativeQuery = true)
   int isVerified(@Param("email") String email);
   @Query(value = "SELECT email FROM users", nativeQuery = true)
   List<String> getAllEmails();
   @Modifying
   @Query(value = "INSERT INTO users (first_name, last_name, email,
password, token, code, created_at) VALUES" +
   "(:first_name, :last_name, :email, :password, :token, :code, :created_at)",
nativeQuery = true)
   @Transactional
   void registerUser(@Param("first_name") String first_name,
```

```
                @Param("last_name") String last_name,
                @Param("email") String email,
                @Param("password") String password,
                @Param("token") String token,
                @Param("code") int code,
                @Param("created_at")LocalDateTime created_at);
    @Modifying
    @Query(value = "UPDATE users set token=null, code=null, verified=1,
verified_at=NOW(), updated_at=NOW() where " +
          "token=:token AND code=:code", nativeQuery = true)
    @Transactional
    void verifyAccount(@Param("token") String token,
                @Param("code") String code);
    @Query(value = "SELECT token FROM users WHERE token = :token",
nativeQuery = true)
    String checkToken(@Param("token") String token);
    @Query(value = "SELECT * FROM users", nativeQuery = true)
    List<User> getAllUsers();
}
```

UserViewRepository.java -

```
package com.bank.repository;
@Repository
public interface UserViewRepository extends CrudRepository<UserView,
Integer> {
    @Query(value = "SELECT * FROM v_users", nativeQuery = true)
    List<UserView> getUserView();
}
```

6.6. Helpers

DateFormatter.java -

```
package com.bank.helpers;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.time.format.FormatStyle;

public class DateFormatter {

   public static String formatLocalDateTime(LocalDateTime localDateTime,
String pattern) {

      DateTimeFormatter dateTimeFormatter =
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);

      String formattedDateTime = localDateTime.format(dateTimeFormatter);

      return formattedDateTime;

   }

   public static String getFormattedDate(LocalDateTime localDateTime) {

      String pattern = "yyyy/MM/dd";

      DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern(pattern);

      String dateFormat = localDateTime.format(dateFormatter);

      return dateFormat;

   }

   public static String getFormattedTime(LocalDateTime localDateTime) {

      String pattern = "HH:mm:ss";

      DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern(pattern);

      String timeFormat = localDateTime.format(timeFormatter);

      return timeFormat;

   }

}
```

LoanPaymentScheduler.java -

```
package com.bank.helpers;
import com.bank.mailMessenger.MailMessenger;
import com.bank.models.LoanLog;
import com.bank.repository.AccountRepository;
import com.bank.repository.LoanLogRepository;
import com.bank.repository.TransactRepository;
import com.bank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.HashMap;
@Service
public class LoanPaymentScheduler {
   @Autowired
   UserRepository userRepository;
   @Autowired
   AccountRepository accountRepository;
   @Autowired
   LoanLogRepository loanLogRepository;
   @Autowired
   TransactRepository transactRepository;
   @Scheduled(cron = "0 0 0 1 * *")
   public void LoanPayment() {
      HashMap<String, Integer> months = new HashMap<>();
      for(LoanLog log : loanLogRepository.getAllLoanLogs()) {
```

```
        System.out.println("Inside scheduler");

        if(log.getTotal_amount_payable() > 0 &&
(log.getIs_confirmed().equals("yes"))) {

            System.out.println("Time: " + new Date().toString() + "\n" +
log.getLoan_type() + " -- " + log.getLoan_application_number());

            LoanLog loanLog =
loanLogRepository.getLoanLogsByApplicationNumber(log.getLoan_application_
number());

            int borrowerAccountID = loanLog.getBorrower_account_id();

            int borrowerUserID = loanLog.getBorrower_user_id();

            double emi = loanLog.getFinal_emi();

            double charges = loanLog.getCharges_payable();

            double penalty = loanLog.getTotal_penalty_payable();

            double final_amount = emi + charges + penalty;

            double accountBalance =
accountRepository.getAccountBalance(borrowerUserID, borrowerAccountID);

            if (accountBalance < final_amount) {

               System.out.println("Insufficient Balance");

               String email = loanLog.getBorrower_email();

               String emailBody =
HTML.insufficientBalance(loanLog.getBorrower_name(),
loanLog.getLoan_type(), loanLog.getLoan_application_number(),
loanLog.getLate_payment_penalty(),
                     loanLog.getTotal_amount_payable());


MailMessenger.htmlEmailMessenger("cadmus.finance.corp@gmail.com", email,
loanLog.getLoan_type() + " Insufficient Balance Notice", emailBody, null);

               //Log unsuccessful transaction

               transactRepository.logTransaction(borrowerAccountID,
loanLog.getLoan_type() + " Payment", final_amount, "Online", "Failure",
"Insufficient Balance", LocalDateTime.now());


loanLogRepository.setPenaltyPayableByApplicationNumber(loanLog.getTotal_pe
```

```
nalty_payable() + loanLog.getLate_payment_penalty(),
loanLog.getLoan_application_number());

            }

            else {

                double newAccountBalance = accountBalance - final_amount;

                accountRepository.changeAccountBalanceById(newAccountBalance,
borrowerAccountID);


loanLogRepository.updateAmountPayableByApplicationNumber(loanLog.getTot
al_amount_payable() - final_amount, loanLog.getLoan_application_number());


loanLogRepository.updateChargesPayableByApplicationNumber(loanLog.getLoa
n_application_number());

                loanLogRepository.setPenaltyPayableByApplicationNumber(0,
loanLog.getLoan_application_number());


loanLogRepository.logUpdateTimeByApplicationNumber(LocalDateTime.now(),
loanLog.getLoan_application_number());

                transactRepository.logTransaction(borrowerAccountID,
loanLog.getLoan_type() + " Payment", final_amount, "Online", "Success", "Loan
Payment Successful", LocalDateTime.now());

                System.out.println("Successful loan payment");

            }

        }

    }

}
```

Token.java -

```
package com.bank.helpers;
import java.util.UUID;
public class Token {
    public static String generateToken() {
        String token = UUID.randomUUID().toString();
        return token;
    }
}
```

6.7 Controllers

AccountController.java -

```
@Controller
@RequestMapping("/account")
public class AccountController {
   @Autowired
   private AccountRepository accountRepository;
   @PostMapping("/create_account")
   public String createAccount(@RequestParam("account_name") String accountName,
                               @RequestParam("account_type") String accountType,
                               RedirectAttributes redirectAttributes,
                               HttpSession session) {
      //Check for empty strings
      if(accountName.isEmpty() || accountType.isEmpty()) {
         redirectAttributes.addFlashAttribute("error", "Account Name or Account Type Cannot be Empty!");
         return "redirect:/app/dashboard";
      }
      //Get logged in user
      User user = (User) session.getAttribute("user");
      //Generate account number
      String bankAccountNumber = GenerateAccountNumber.generateAccountNumber();
      //Creating account
      accountRepository.createBankAccount(user.getUser_id(), bankAccountNumber, accountName, accountType);
      //Set success message
      redirectAttributes.addFlashAttribute("success", "Account Created Successfully!");
```

```
        return "redirect:/app/dashboard";
    }
}
```

AppController.java

```
package com.bank.controllers;
import com.bank.helpers.*;
import com.bank.mailMessenger.MailMessenger;
import com.bank.models.*;
import com.bank.repository.*;
import com.lowagie.*;
import org.springframework.*;
import java.io.*;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.*;
@Controller
@RequestMapping("/app")
public class AppController {
    @Autowired
    private AccountRepository accountRepository;
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PaymentHistoryRepository paymentHistoryRepository;
    @Autowired
    private TransactHistoryRepository transactHistoryRepository;
    @Autowired
```

```
    private EmployeeViewRepository employeeViewRepository;
    @Autowired
    private UserViewRepository userViewRepository;
    @Autowired
    private EmployeeRepository employeeRepository;
    @Autowired
    private DepartmentRepository departmentRepository;
    @Autowired
    private BranchRepository branchRepository;
    @Autowired
    private HomeApplicationRepository homeApplicationRepository;
    @Autowired
    private PersonalLoanApplicationRepository
personalLoanApplicationRepository;
    @Autowired
    private GoldLoanApplicationRepository goldLoanApplicationRepository;
    @Autowired
    private CustomerRepository customerRepository;
    @Autowired
    private LoanLogRepository loanLogRepository;
    User user;
    @GetMapping("/dashboard")
    public ModelAndView getDashboard(HttpSession session, RedirectAttributes
redirectAttributes) {
        //Set view
        ModelAndView getDashboardPage = new ModelAndView("dashboard");
        //Get the details of the logged in user
        user = (User) session.getAttribute("user");
        //Get the accounts of the logged in user
```

```
        List<Account> getUserAccounts =
accountRepository.getUserAccountsById(user.getUser_id());

        //Get balance

        BigDecimal totalAccountBalance =
accountRepository.getTotalBalance(user.getUser_id());

        userRepository.updatedAtNow();

        //Set objects

        getDashboardPage.addObject("userAccounts", getUserAccounts);

        getDashboardPage.addObject("totalBalance", totalAccountBalance);

        return getDashboardPage;

    }

    @GetMapping("/payment_history")

    public ModelAndView getPaymentHistory(HttpSession session,
HttpServletResponse response) throws DocumentException, IOException {

        //Set view

        ModelAndView getPaymentHistoryPage = new
ModelAndView("paymentHistory");

        //Get logged in user

        user = (User) session.getAttribute("user");

        //Get payment record

        List<PaymentHistory> userPaymentHistory =
paymentHistoryRepository.getPaymentRecordsById(user.getUser_id());

        //Get the accounts of the logged in user

        List<Account> getUserAccounts =
accountRepository.getUserAccountsById(user.getUser_id());

        //Set objects

        getPaymentHistoryPage.addObject("payment_history",
userPaymentHistory);

        getPaymentHistoryPage.addObject("userAccounts", getUserAccounts);

        return getPaymentHistoryPage;

    }
```

```
    @GetMapping("/transact_history")
    public ModelAndView getTransactHistory(HttpSession session) {
        // Set View:
        ModelAndView getTransactHistoryPage = new
ModelAndView("transactHistory");
        // Get Logged In User:\
        user = (User) session.getAttribute("user");
        // Get Transaction History / Records:
        List<TransactionHistory> userTransactHistory =
transactHistoryRepository.getTransactionRecordsById(user.getUser_id());
        //Get the accounts of the logged in user
        List<Account> getUserAccounts =
accountRepository.getUserAccountsById(user.getUser_id());
        getTransactHistoryPage.addObject("transact_history", userTransactHistory);
        getTransactHistoryPage.addObject("userAccounts", getUserAccounts);
        return getTransactHistoryPage;
    }
    @GetMapping("/payment_history_export_pdf")
    public void exportPaymentHistoryToPDF(HttpServletResponse response,
HttpSession session) throws DocumentException, IOException {
        //Prepare for exporting PDF
        response.setContentType("application/pdf");
        DateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-dd");
        String currentDateTime = dateFormatter.format(new Date());
        //Get logged in user
        user = (User) session.getAttribute("user");
        //Set header
        String headerKey = "Content-Disposition";
        String headerValue = "attachment; filename=" + user.getFirst_name() + " " +
user.getLast_name() + "_payment_history_" + currentDateTime + ".pdf";
```

```
        response.setHeader(headerKey, headerValue);
        //Get payment record
        List<PaymentHistory> userPaymentHistoryList =
paymentHistoryRepository.getPaymentRecordsById(user.getUser_id());
        //Export Payment history to PDF
        PaymentHistoryPDFExporter paymentHistoryPDFExporter = new
PaymentHistoryPDFExporter(userPaymentHistoryList);
        paymentHistoryPDFExporter.export(response);
    }
    @GetMapping("/loan")
    public ModelAndView getLoan(HttpSession session, RedirectAttributes
redirectAttributes) {
        //Set view
        ModelAndView getLoanPage = new ModelAndView("loan");
        //Get the details of the logged in user
        user = (User) session.getAttribute("user");
        //Get the accounts of the logged in user
        List<Account> getUserAccounts =
accountRepository.getUserAccountsById(user.getUser_id());
        //Get total balance
        BigDecimal totalAccountBalance =
accountRepository.getTotalBalance(user.getUser_id());
        int accBalance;
        if(totalAccountBalance == null)
            accBalance = 0;
        else
           accBalance = totalAccountBalance.intValue();
        getLoanPage.addObject("userAccounts", getUserAccounts);
        getLoanPage.addObject("totalAccountBalance", accBalance);
        return getLoanPage;
```

```
    }

    @GetMapping("/admin_dashboard")

    public ModelAndView getAdminDashboard(HttpSession session,
RedirectAttributes redirectAttributes) {

        //Set view

        ModelAndView getDashboardPage = new
ModelAndView("admindashboard");

        //Get the details of the logged in admin

        Admin admin = (Admin) session.getAttribute("admin");

        getDashboardPage.addObject("admin", admin);

        //Get employee view

        List<EmployeeView> employeeViews =
employeeViewRepository.getEmployeeView();

        getDashboardPage.addObject("employeeViews", employeeViews);

        //Get user view

        List<UserView> userViews = userViewRepository.getUserView();

        getDashboardPage.addObject("userViews", userViews);

        //Get Home loan count

        int homeLoanCount =
homeApplicationRepository.getHomeLoanApplicationCount();

        getDashboardPage.addObject("homeLoanCount", homeLoanCount);

        //Get Personal loan count

        int personalLoanCount =
personalLoanApplicationRepository.getPersonalLoanApplicationCount();

        getDashboardPage.addObject("personalLoanCount", personalLoanCount);

        //Get Gold loan count

        int goldLoanCount =
goldLoanApplicationRepository.getGoldLoanApplicationCount();

        getDashboardPage.addObject("goldLoanCount", goldLoanCount);

        //Get all user applied for home loan
```

```
        List<HomeLoanApplication> allUsersAppliedHomeLoan =
homeApplicationRepository.getAllHomeLoanApplications();

        getDashboardPage.addObject("allUsersAppliedHomeLoan",
allUsersAppliedHomeLoan);

        //Get all user applied for personal loan

        List<PersonalLoanApplication> allUsersAppliedPersonalLoan =
personalLoanApplicationRepository.getAllPersonalLoanApplications();

        getDashboardPage.addObject("allUsersAppliedPersonalLoan",
allUsersAppliedPersonalLoan);

        //Get all user applied for personal loan

        List<GoldLoanApplication> allUsersAppliedGoldLoan =
goldLoanApplicationRepository.getAllGoldLoanApplications();

        getDashboardPage.addObject("allUsersAppliedGoldLoan",
allUsersAppliedGoldLoan);

        //Get all transactions

        List<TransactionHistory> allTransactionRecords =
transactHistoryRepository.getAllTransactionRecords();

        getDashboardPage.addObject("allTransactionRecords",
allTransactionRecords);

        return getDashboardPage;

    }

    @GetMapping("/employee_panel")

    public ModelAndView getEmployeePanel(HttpSession session,
RedirectAttributes redirectAttributes) {

        System.out.println("In employee panel controller");

        //Set view

        ModelAndView getEmployeePage = new
ModelAndView("employeePanel");

        //Get all employees

        List<Employee> allEmployees =  employeeRepository.getAllEmployee();

        getEmployeePage.addObject("allEmployees", allEmployees);

        //Get Departments
```

```
        List<Department> allDepartments =
departmentRepository.getAllDepartment();

        getEmployeePage.addObject("allDepartments", allDepartments);

        //Get Branches

        List<Branch> allBranches = branchRepository.getAllBranch();

        getEmployeePage.addObject("allBranches", allBranches);

        return getEmployeePage;

    }

    @GetMapping("/application_panel")

    public ModelAndView getApplicationPanel(HttpSession session,
RedirectAttributes redirectAttributes) {

        System.out.println("In application panel controller");

        //Set view

        ModelAndView getEmployeePage = new
ModelAndView("applicationPanel");

        //Get all home loan applications

        List<HomeLoanApplication> allHomeLoanApplications =
homeApplicationRepository.getAllHomeLoanApplications();

        getEmployeePage.addObject("allHomeLoanApplications",
allHomeLoanApplications);

        //Get all personal loan applications

        List<PersonalLoanApplication> allPersonalLoanApplications =
personalLoanApplicationRepository.getAllPersonalLoanApplications();

        getEmployeePage.addObject("allPersonalLoanApplications",
allPersonalLoanApplications);

        //Get all gold loan applications

        List<GoldLoanApplication> allGoldLoanApplications =
goldLoanApplicationRepository.getAllGoldLoanApplications();

        getEmployeePage.addObject("allGoldLoanApplications",
allGoldLoanApplications);

        //Get loan logs
```

```
        List<LoanLog> allLoanLogs = loanLogRepository.getAllLoanLogs();
        getEmployeePage.addObject("allLoanLogs", allLoanLogs);
        return getEmployeePage;
    }
    @GetMapping("/application_panel/home_loan/export_address_proof")
    public ResponseEntity<byte[]>
exportAddressProofPDFHomeLoan(@RequestParam("application_number")
String applicationNumber) throws SQLException, IOException {
        System.out.println("Inside exporting home loan address proof pdf
controller");
        byte[] pdfBytes =
homeApplicationRepository.getAddressProofHomeLoanApplicationByApplicatio
nNumber(applicationNumber);
        HttpHeaders httpHeaders = new HttpHeaders();
        httpHeaders.setContentType(MediaType.APPLICATION_PDF);

httpHeaders.setContentDisposition(ContentDisposition.builder("attachment").file
name("Home Loan Address Proof Application Number - " +
applicationNumber).build());
        ResponseEntity<byte[]> response = new ResponseEntity<>(pdfBytes,
httpHeaders, HttpStatus.OK);
        return response;
    }
    @PostMapping("/application_panel/approve/home_loan")
    public String approveHomeLoan(@RequestParam("loan_amount") String
loanAmount,
                                  @RequestParam("interest_rate") String interestRate,
                                  @RequestParam("tenure") String tenure,
                                  @RequestParam("application_number") String
applicationNumber,
                                  HttpSession session,
                                  RedirectAttributes redirectAttributes) throws IOException {
```

```
        System.out.println("In approve home loan panel");
        //Check for empty strings
        if(loanAmount.isEmpty() || interestRate.isEmpty() || tenure.isEmpty()) {
            redirectAttributes.addFlashAttribute("error", "Loan Amount OR Interest
Rate OR Tenure cannot be empty");
            return "redirect:/app/application_panel?application_number=" +
applicationNumber;
        }
        //Set approved to yes
homeApplicationRepository.setApprovedToYesHomeLoanApplication(applicatio
nNumber);
        //Get email
        String email =
homeApplicationRepository.getEmailHomeLoanApplicationByApplicationNumb
er(applicationNumber);
        //Convert Variables
        long principal = Long.parseLong(loanAmount);
        float rate = Float.parseFloat(interestRate);
        int period = Integer.parseInt(tenure);
        //EMI calculation
        rate = rate / (12 * 100);
        period = period * 12;
        double emi = (principal * rate * Math.pow((1 + rate), period)) /
(Math.pow((1 + rate), period) - 1);
        double totalAmountPayable = emi * period;
        double totalInterestPayable = totalAmountPayable - principal;
        //Charges calculation
        double processingFee = principal * 0.5 / 100;
        if(processingFee < 1500)
            processingFee = 1500;
```

```
        else if(processingFee > 20000)
            processingFee = 20000;
        double administrationFee =  principal * 0.3 / 100;
        if(administrationFee < 1500)
            administrationFee = 1500;
        else if(administrationFee > 10000)
            administrationFee = 20000;
        double titleSearchFee = principal * 0.15 / 100;
        if(titleSearchFee < 1500)
            titleSearchFee = 1500;
        else if(titleSearchFee > 8000)
            titleSearchFee = 8000;
        double appraisalFee = 2000;
        double latePaymentPenalty = emi * 2 / 100;
        double prePaymentPenalty = emi * 3 / 100;
        double chargesPayable = processingFee + administrationFee + appraisalFee
+ titleSearchFee;
        //Get home loan application
        HomeLoanApplication homeLoanApplication =
homeApplicationRepository.getHomeLoanApplicationByApplicationNumber(app
licationNumber);
        loanLogRepository.logLoan(applicationNumber, "Home Loan",
homeLoanApplication.getFirst_name() + " " +
homeLoanApplication.getLast_name(), homeLoanApplication.getUser_id(),
email, principal, Float.parseFloat(interestRate),
                period, emi, totalAmountPayable, totalInterestPayable, chargesPayable,
latePaymentPenalty, prePaymentPenalty, homeLoanApplication.getAccount(),
homeLoanApplication.getConfirm());
        //Get email HTML body
        String emailBody = HTML.htmlHomeLoanApprovalTemplate("Home",
applicationNumber, loanAmount, interestRate, tenure);
```

```
        //Get details pdf
        HomeLoanLogPDFExporter homeLoanLogPDFExporter = new
HomeLoanLogPDFExporter(homeLoanApplication,
loanLogRepository.getLoanLogsByApplicationNumber(applicationNumber));
        byte[] pdfBytes = homeLoanLogPDFExporter.export();
        //Send email notification
        MailMessenger.htmlEmailMessenger("cadmus.finance.corp@gmail.com",
email, "Home Loan Application Approval", emailBody, pdfBytes);
        return "redirect:/app/application_panel";
    }
    @GetMapping("/customer_panel")
    public ModelAndView getCustomerPanel(HttpSession session,
RedirectAttributes redirectAttributes) {
        System.out.println("In customer panel controller");
        //Set view
        ModelAndView getCustomerPage = new ModelAndView("customerPanel");
        //Get customers
        List<Customer> allCustomers =  customerRepository.getAllCustomers();
        getCustomerPage.addObject("allCustomers", allCustomers);
        //Get payments
        List<PaymentHistory> allPayments =
paymentHistoryRepository.getAllPayments();
        getCustomerPage.addObject("allPayments", allPayments);
        //Get Accounts
        List<Account> allAccounts = accountRepository.getAllAccounts();
        getCustomerPage.addObject("allAccounts", allAccounts);
        //Set active tab
        getCustomerPage.addObject("activeTab", "customer");
        return getCustomerPage;
    }
```

```
    @PostMapping("/customer_panel")
    public ModelAndView getCustomerPanel(@RequestParam("sort-option") String sortOption,
                                         @RequestParam("sorting-technique") String sortingTechnique,
                                         @RequestParam("sorting-panel") String sortingPanel,
                                         HttpSession session,
                                        RedirectAttributes redirectAttributes) {
        System.out.println("In customer panel POST controller");
        //Set view
        ModelAndView getCustomerPage = new ModelAndView("customerPanel");
        if(sortingPanel.equals("customer")) {
            //Get payments
            List<PaymentHistory> allPayments = paymentHistoryRepository.getAllPayments();
            getCustomerPage.addObject("allPayments", allPayments);
            //Get Accounts
            List<Account> allAccounts = accountRepository.getAllAccounts();
            getCustomerPage.addObject("allAccounts", allAccounts);
            //Get customers
            List<Customer> allCustomers =  customerRepository.getAllCustomers();
            //Set active tab
            getCustomerPage.addObject("activeTab", sortingPanel);
            //Check for empty field
            if(sortOption.equals("")) {
                getCustomerPage.addObject("errorCustomer", "Please select Sorting Column");
                getCustomerPage.addObject("allCustomers", allCustomers);
                return getCustomerPage;
            }
```

```
        List<Customer> customerSorted;
        if(sortOption.equals("ID") && sortingTechnique.equals("ascending")) {
            customerSorted = customerRepository.getCustomersOrderByID();
            getCustomerPage.addObject("customerSorted", customerSorted);
            getCustomerPage.addObject("customerSortingColumn_technique",
"ID_asc");
        }
        if(sortOption.equals("ID") && sortingTechnique.equals("descending")) {
            customerSorted = customerRepository.getCustomersOrderByIDDesc();
            getCustomerPage.addObject("customerSorted", customerSorted);
            getCustomerPage.addObject("customerSortingColumn_technique",
"ID_desc");
        }
        if(sortOption.equals("name") && sortingTechnique.equals("ascending"))
{
            customerSorted = customerRepository.getCustomersOrderByName();
            getCustomerPage.addObject("customerSorted", customerSorted);
            getCustomerPage.addObject("customerSortingColumn_technique",
"name_asc");
        }
        if(sortOption.equals("name") && sortingTechnique.equals("descending"))
{
            customerSorted =
customerRepository.getCustomersOrderByNameDesc();
            getCustomerPage.addObject("customerSorted", customerSorted);
            getCustomerPage.addObject("customerSortingColumn_technique",
"name_desc");
        }
        List<PaymentHistory> paymentSorted;
        String[] groupPaymentByCols = {"account_id", "user_id",
"beneficiary_bank", "status", "reason_code"};
```

```
        List<String> distinctValues = new ArrayList<>();
        Map<String, List<PaymentHistory>> PaymentsGroupedByCol;
        Map<String,Map<String, List<PaymentHistory>>> paymentObj = new
HashMap<>();
        for (String col : groupPaymentByCols) {
           if (sortOption.equals(col)) {
              try {
                 String colFormatted = "";
                 for(String s : col.split("_"))
                    colFormatted += s.substring(0, 1).toUpperCase() +
s.substring(1);
                 Method getDistinctValuesMethod =
PaymentHistoryRepository.class.getDeclaredMethod("getDistinct" +
colFormatted);
                 distinctValues = (List<String>)
getDistinctValuesMethod.invoke(paymentHistoryRepository);
                 Method getSpecificValuesMethod =
PaymentHistoryRepository.class.getDeclaredMethod("getSpecific" +
colFormatted, String.class);
                 PaymentsGroupedByCol = new HashMap<>();
                 for (String value : distinctValues) {
                    List<PaymentHistory> payments = (List<PaymentHistory>)
getSpecificValuesMethod.invoke(paymentHistoryRepository, value);
                    PaymentsGroupedByCol.put(value, payments);
                 }
                 paymentObj.put(col, PaymentsGroupedByCol);
              } catch (NoSuchMethodException | IllegalAccessException |
InvocationTargetException e) {
                 e.printStackTrace();
              }
           }
```

```
            }
            getCustomerPage.addObject("groupedPaymentObj", paymentObj);
        }
        return getCustomerPage;
    }
}
```

AuthController.java -

```
@Controller
public class AuthController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private AdminRepository adminRepository;
    @GetMapping("/login")
    public ModelAndView getLogin() {
        System.out.println("In login controller");
        ModelAndView getLoginPage = new ModelAndView(("login"));
        //Set token string
        String token = Token.generateToken();
        //Set token to view
        getLoginPage.addObject("token", token);
        getLoginPage.addObject("PageTitle", "Login");
        return getLoginPage;
    }
    @PostMapping("/login")
    public String login(@RequestParam("email") String email,
                        @RequestParam("password") String password,
```

```
                @RequestParam("_token") String token,
                Model model,
                HttpSession session) {
        //Validate input fields / form data
        if(email.isEmpty() || email == null || password.isEmpty() || password == null)
{
            model.addAttribute("error", "Username or Password cannot be empty");
            return "login";
        }
        //Check for valid email
        String regex = "([a-zA-Z0-9]+(?:[._+-][a-zA-Z0-9]+)*)@([a-zA-Z0-9]+(?:[.-
][a-zA-Z0-9]+)*[.][a-zA-Z]{2,})";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(email);
        if(!matcher.matches()) {
            model.addAttribute("error", "Please enter a Valid Email Address");
            return "login";
        }
        //Check if email exists
        List<String> allEmails = userRepository.getAllEmails();
        if(!allEmails.contains(email)) {
            model.addAttribute("error", "Your Account does not exist. Please Sign
Up!");
            return "login";
        }
        //Get email from database
        String getEmailInDB = userRepository.getUserEmail(email);
        //Check password
        if(getEmailInDB != null) {
```

```
            //Get password from database
            String getPasswordInDB =
userRepository.getUserPassword(getEmailInDB);
            //Validate password
            if(!BCrypt.checkpw(password, getPasswordInDB)) {
               model.addAttribute("error", "Incorrect Username or Password");
               return "login";
            }
         }
         else {
            model.addAttribute("error", "Something went wrong. Please contact
support");
            return "error";
         }
         //Get verified from database
         int verified = userRepository.isVerified(getEmailInDB);
         //Check if user account is verified
         if(verified != 1) {
            model.addAttribute("error", "This Account is not yet Verified. Please
check email in order to verify account.");
            return "login";
         }
         //Proceed to log the user in
         User user = userRepository.getUserDetails(getEmailInDB);
         //Set session attribute
         session.setAttribute("user", user);
         session.setAttribute("token", token);
         session.setAttribute("authenticated", true);
         return "redirect:/app/dashboard";
```

```
    }
    @GetMapping("/logout")
    public String logout(HttpSession session, RedirectAttributes redirectAttributes)
{
        session.invalidate();
        redirectAttributes.addFlashAttribute("logged_out", "Logged Out
successfully");
        return "redirect:/login";
    }
    @GetMapping("/adminlogin")
    public ModelAndView getAdminLogin() {
        System.out.println("In admin login controller");
        ModelAndView getAdminLoginPage = new ModelAndView("adminlogin");
        return getAdminLoginPage;
    }
    @PostMapping("/adminlogin")
    public String loginAdmin(@RequestParam("admin_email") String
adminEmail,
                             @RequestParam("admin_password") String adminPassword,
                             @RequestParam("admin_id") String adminID,
                             HttpSession session,
                             Model model) {
        if(adminEmail.isEmpty() || adminPassword.isEmpty() || adminID.isEmpty())
{
            model.addAttribute("error", "Email OR Password OR ID cannot be
Empty");
            return "adminlogin";
        }
        //Check if admin ID exists
        List<String> allAdminID = adminRepository.getAllAdminID();
```

```
        if(!allAdminID.contains(adminID)) {
            model.addAttribute("error", "Admin with this ID does not exist");
            return "adminlogin";
        }
        //Check for valid email
        String regex = "([a-zA-Z0-9]+(?:[._+-][a-zA-Z0-9]+)*)@([a-zA-Z0-9]+(?:[.-
][a-zA-Z0-9]+)*[.][a-zA-Z]{2,})";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(adminEmail);
        if(!matcher.matches()) {
            model.addAttribute("error", "Please enter a Valid Email Address");
            return "adminlogin";
        }
        //Check if email matches
        String adminEmailInDB = adminRepository.getAdminEmail(adminID);
        if(!adminEmail.equals(adminEmailInDB)) {
            model.addAttribute("error", "Incorrect Email Address OR Password");
            return "adminlogin";
        }
        //Check if password matches
        String adminPasswordInDB =
adminRepository.getAdminPassword(adminID);
        if(!BCrypt.checkpw(adminPassword, adminPasswordInDB)) {
            model.addAttribute("error", "Incorrect Email Address or Password");
            return "adminlogin";
        }
        //Get admin details
        Admin admin = adminRepository.getAdminDetails(adminID);
        LocalDateTime currentTimeStamp = LocalDateTime.now();
```

```
        adminRepository.updateAdminLastLogin(currentTimeStamp,
admin.getAdmin_id());
        //Set session attribute
        session.setAttribute("admin", admin);
        return "redirect:/app/admin_dashboard";
    }
    @GetMapping("/adminlogout")
    public String AdminLogout(HttpSession session, RedirectAttributes
redirectAttributes) {
        session.invalidate();
        redirectAttributes.addFlashAttribute("logged_out", "Logged Out
successfully");
        return "redirect:/adminlogin";
    }
}
```

IndexController.java -

```
package com.bank.controllers;
@Controller
public class IndexController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private HomeApplicationRepository homeApplicationRepository;
    @Autowired
    private PersonalLoanApplicationRepository
personalLoanApplicationRepository;
    @Autowired
    private GoldLoanApplicationRepository goldLoanApplicationRepository;
    @Autowired
```

```
private LoanLogRepository loanLogRepository;
@GetMapping("/")
public ModelAndView getIndex() {
   ModelAndView getIndexPage = new ModelAndView("index");
   getIndexPage.addObject("PageTitle", "Home");
   System.out.println("In index controller");
   return getIndexPage;
}
@GetMapping("/perk")
public ModelAndView getPerk() {
   ModelAndView getPerkPage = new ModelAndView(("perk"));
   System.out.println("In perk controller");
   getPerkPage.addObject("PageTitle", "Perk");
   return getPerkPage;
}
@GetMapping("/aboutus")
public ModelAndView getAboutus() {
   ModelAndView getAboutusPage = new ModelAndView(("aboutus"));
   System.out.println("In Aboutus controller");
   getAboutusPage.addObject("PageTitle", "Aboutus");
   return getAboutusPage;
}
@GetMapping("/error")
public ModelAndView getError() {
   ModelAndView getErrorPage = new ModelAndView(("error"));
   System.out.println("In Error controller");
   getErrorPage.addObject("PageTitle", "Error");
   return getErrorPage;
```

```
    }
    @GetMapping("/verify")
    public ModelAndView getVerify(@RequestParam("token") String token,
                                  @RequestParam("code") String code) {
        //Set View
        ModelAndView getVerifyPage;
        //Get token in database
        String dbToken = userRepository.checkToken(token);
        //check if token is valid
        if(dbToken == null) {
            getVerifyPage = new ModelAndView("error");
            getVerifyPage.addObject("error", "This Session Has Expired");
            return getVerifyPage;
        }
        //Update verified_at
        userRepository.verifiedAtNow();
        //Update and verify account
        userRepository.verifyAccount(token, code);
        getVerifyPage = new ModelAndView("login");
        System.out.println("In Verify controller");
        getVerifyPage.addObject("success", "Account Verified Successfully, Please
proceed to log in!");
        return getVerifyPage;
    }
    @GetMapping("/confirm")
    public ModelAndView confirmLoan(@RequestParam("loan_type") String
loanType,
                                    @RequestParam("application_number") String
applicationNumber) {
```

```
        System.out.println("In Loan Confirmation Controller");
        ModelAndView getLoginPage = new ModelAndView("login");
        if(loanType.equals("Home")) {

homeApplicationRepository.setConfirmToYesHomeLoanApplication(application
Number);

loanLogRepository.setConfirmedToYesByApplicationNumber(applicationNumbe
r);
        }
        else if(loanType.equals("Personal")) {

personalLoanApplicationRepository.setConfirmToYesPersonalLoanApplication(a
pplicationNumber);

loanLogRepository.setConfirmedToYesByApplicationNumber(applicationNumbe
r);
        }
        else if(loanType.equals("Gold")) {

goldLoanApplicationRepository.setConfirmToYesGoldLoanApplication(applicati
onNumber);

loanLogRepository.setConfirmedToYesByApplicationNumber(applicationNumbe
r);
        }
        return getLoginPage;
    }
}
```

LoanController.java -

```
package com.bank.controllers;

@Controller

@RequestMapping("/apply")

public class LoanController {

   @Autowired

   private HomeApplicationRepository homeApplicationRepository;

   @Autowired

   private PersonalLoanApplicationRepository
personalLoanApplicationRepository;

   @Autowired

   private GoldLoanApplicationRepository goldLoanApplicationRepository;

   User user;

   @PostMapping("/home_loan")

   public String applyHomeLoan(@RequestParam("HL_UserFirstName") String
userFirstName,

                        @RequestParam("HL_UserLastName") String userLastName,

                        @RequestParam("HL_UserEmail") String userEmail,

                        @RequestParam("HL_UserContact") String userContact,

                        @RequestParam("HL_UserReligion") String userReligion,

                        @RequestParam("HL_UserAge") String  userAge,

                        @RequestParam("HL_UserBirthDate") String userBirthDate,

                        @RequestParam("HL_UserMaritalStatus") String
userMaritalStatus,

                        @RequestParam("HL_UserProofIdentity") MultipartFile
userProofIdentity,

                        @RequestParam("HL_UserAddressHouse") String
userAddressHouse,

                        @RequestParam("HL_UserAddressComplex") String
userAddressComplex,
```

```
                    @RequestParam("HL_UserAddressStreetLine1") String
userAddressStreetLine1,
                    @RequestParam("HL_UserAddressStreetLine2") String
userAddressStreetLine2,
                    @RequestParam("HL_UserAddressCity") String
userAddressCity,
                    @RequestParam("HL_userAddressState") String
userAddressState,
                    @RequestParam("HL_UserAddressPostalCode") String
userAddressPostalCode,
                    @RequestParam("HL_UserAddressProof") MultipartFile
userAddressProof,
                    @RequestParam("HL_UserCompanyName") String
userCompanyName,
                    @RequestParam("HL_UserJobTitle") String userJobTitle,
                    @RequestParam("HL_UserEmploymentStatus") String
userEmploymentStatus,
                    @RequestParam("HLEmploymentPeriod") String
userEmploymentPeriod,
                    @RequestParam("HL_UserWorkMode") String
userWorkMode,
                    @RequestParam("HL_UserDepartment") String
userDepartment,
                    @RequestParam("HL_UserSalary") String userSalary,
                    @RequestParam("HL_UserIndustry") String userIndustry,
                    @RequestParam("HL_UserSalarySlip") MultipartFile
userSalarySlip,
                    @RequestParam("HL_UserIncome") String userIncome,
                    @RequestParam("HL_UserInvestment") String
userInvestment,
                    @RequestParam("HL_UserCreditCardDebt") String
userCreditCardDebt,
                    @RequestParam("HL_UserAccount") String userAccount,
```

```
                    @RequestParam("HL_UserBankAccountStatement")
MultipartFile userBankAccountStatement,

                    @RequestParam("HL_UserPurchasePrice") String
userPurchasePrice,

                    @RequestParam("HL_UserLoanAmountRequested") String
userLoanAmountRequested,

                    @RequestParam("HL_UserDownPaymentAmount_hidden")
String userDownPaymentAmount,

                    @RequestParam("HL_UserCurrentHome") String
userCurrentHome,

                    @RequestParam("HL_UserSellCurrent") String
userSellCurrent,

                    @RequestParam("HL_UserMottoPurchase") String
userMottoPurchase,

                    HttpSession session,

                    RedirectAttributes redirectAttributes) throws IOException {

    //Get logged in user

    user = (User) session.getAttribute("user");

    int userId = user.getUser_id();

    List<Integer> usersApplied =
homeApplicationRepository.getAllUsersAppliedHomeLoan();

    if(usersApplied.contains(userId)) {

       redirectAttributes.addFlashAttribute("error", "You have already applied
for Home Loan");

       return "redirect:/app/loan";

    }

    //Converting variables

    int userAgeInt = Integer.parseInt(userAge);

    LocalDate userBirthDateLocalDate = LocalDate.parse(userBirthDate);

    int userEmploymentPeriodInt = Integer.parseInt(userEmploymentPeriod);

    double userSalaryDouble = Double.parseDouble(userSalary);
```

```
        double userIncomeDouble = Double.parseDouble(userIncome);

        double userInvestmentDouble = Double.parseDouble(userInvestment);

        double userCreditCardDebtDouble =
Double.parseDouble(userCreditCardDebt);

        int userAccountInt = Integer.parseInt(userAccount);

        double userPurchasePriceBigDecimal =
Double.parseDouble(userPurchasePrice);

        double userLoanAmountRequestedBigDecimal =
Double.parseDouble(userLoanAmountRequested);

        double userDownPaymentAmountBigDecimal =
Double.parseDouble(userDownPaymentAmount);

        //Generate application number

        String applicationNumber =
GenerateAccountNumber.generateApplicationNumber();

        //Converting PDF files into byte arrays

        byte[] userProofIdentityByteArray = userProofIdentity.getBytes();

        byte[] userAddressProofByteArray = userAddressProof.getBytes();

        byte[] userSalarySlipByteArray = userSalarySlip.getBytes();

        byte[] userBankAccountStatementByteArray =
userBankAccountStatement.getBytes();

        homeApplicationRepository.applyHomeLoan(userId, applicationNumber,
userFirstName, userLastName, userEmail, userContact, userReligion, userAgeInt,
userBirthDateLocalDate, userMaritalStatus, userProofIdentityByteArray,

                userAddressHouse, userAddressComplex, userAddressStreetLine1,
userAddressStreetLine2, userAddressCity, userAddressState,
userAddressPostalCode, userAddressProofByteArray,

                userCompanyName, userJobTitle, userEmploymentStatus,
userEmploymentPeriodInt, userWorkMode, userDepartment, userSalaryDouble,
userIndustry, userSalarySlipByteArray,

                userIncomeDouble, userInvestmentDouble, userCreditCardDebtDouble,
userAccountInt, userBankAccountStatementByteArray,
```

```
            userPurchasePriceBigDecimal, userLoanAmountRequestedBigDecimal,
userDownPaymentAmountBigDecimal, userCurrentHome, userSellCurrent,
userMottoPurchase);

      redirectAttributes.addFlashAttribute("success", "Successfully Applied for
Home Loan!");

      return "redirect:/app/loan";

   }

}
```

RegisterController.java -

```
package com.bank.controllers;

@Controller

public class RegisterController {

   @Autowired

   private UserRepository userRepository;

   @GetMapping("/register")

   public ModelAndView getRegister() {

      ModelAndView getRegisterPage = new ModelAndView(("register"));

      System.out.println("In register controller");

      getRegisterPage.addObject("PageTitle", "Register");

      return getRegisterPage;

   }

   @PostMapping("/register")

   public ModelAndView register(@Valid @ModelAttribute("registerUser") User
user,

                                BindingResult result,

                                @RequestParam("first_name") String first_name,

                                @RequestParam("last_name") String last_name,

                                @RequestParam("email") String email,

                                @RequestParam("password") String password,
```

```
                        @RequestParam("confirm_password") String
confirm_password,
                        @RequestParam("password-strength") String
passwordStrength) throws MessagingException {
    ModelAndView registrationPage = new ModelAndView("register");
    //Check for Errors
    if(result.hasErrors() && confirm_password.isEmpty()) {
      registrationPage.addObject("confirm_pass", "Please confirm your
password");
      return registrationPage;
    }
    if(!passwordStrength.equals("Strong")) {
      registrationPage.addObject("pass", "Password Strength must be Strong");
      return registrationPage;
    }
    //Check for password match
    if(!password.equals(confirm_password)) {
      registrationPage.addObject("passwordMisMatch", "Passwords do not
match");
      return registrationPage;
    }
    List<String> allEmails = userRepository.getAllEmails();
    if(allEmails.contains(email)) {
      registrationPage.addObject("EmailExists", "Email Address is already
registered. Please head to the log in section.");
      return registrationPage;
    }
    //Get token string
    String token = Token.generateToken();
    //Generate random code
```

```
        Random rand = new Random();
        int bound = 123;
        int code = bound * rand.nextInt(bound);
        //Get email HTML body
        String emailBody = HTML.htmlEmailTemplate(token, code);
        //Hash password
        String hashed_password = BCrypt.hashpw(password, BCrypt.gensalt());
        //Get current date
        LocalDateTime localDateTime = LocalDateTime.now();
        //Register user
        userRepository.registerUser(first_name, last_name, email, hashed_password,
token, code, localDateTime);
        //Send email notification
        MailMessenger.htmlEmailMessenger("cadmus.finance.corp@gmail.com",
email, "Account Verification for Cadmus Financial Corporation", emailBody,
null);
        //Return to register page
        String successMessage = "Account Registered Successfully. Please check
your email and verify your account!";
        registrationPage.addObject("success", successMessage);
        return registrationPage;
    }
}
```

TransactController.java -

```
package com.bank.controllers;
@Controller
@RequestMapping("/transact")
public class TransactController {
    @Autowired
```

```
    private AccountRepository accountRepository;
    @Autowired
    private PaymentRepository paymentRepository;
    @Autowired
    private TransactRepository transactRepository;
    User user;
    double accountBalance;
    double newBalance;
    LocalDateTime currentDateTime = LocalDateTime.now();
    //Deposit Method
    @PostMapping("/deposit")
    public String deposit(@RequestParam("deposit_amount") String
depositAmount,
                          @RequestParam("account_id") String accountID,
                          HttpSession session,
                          RedirectAttributes redirectAttributes) {
        //Check for empty strings
        if(depositAmount.isEmpty() || accountID.isEmpty()) {
            redirectAttributes.addFlashAttribute("error", "Deposit Amount OR
Depositing Account cannot be Empty");
            return "redirect:/app/dashboard";
        }
        //Get logged in user
        user = (User) session.getAttribute("user");
        //Get current account balance
        int account_id = Integer.parseInt(accountID);
        accountBalance = accountRepository.getAccountBalance(user.getUser_id(),
account_id);
        //Check if deposit amount is 0
```

```
        double depositAmountValue = Double.parseDouble(depositAmount);
        if (depositAmountValue == 0) {
            redirectAttributes.addFlashAttribute("error", "Deposit Amount cannot be
Zero (0)");
            return "redirect:/app/dashboard";
        }
        //Update balance
        newBalance = accountBalance + depositAmountValue;
        accountRepository.changeAccountBalanceById(newBalance, account_id);
        accountRepository.setUpdatedAt(account_id);
        //Log successful transaction
        transactRepository.logTransaction(account_id, "Deposit",
depositAmountValue, "Online", "Success", "Amount Deposit Successful",
currentDateTime);
        redirectAttributes.addFlashAttribute("success", "Amount Deposited
Successfully!");
        return "redirect:/app/dashboard";
    }
    //Transfer Method
    @PostMapping("/transfer")
    public String transfer(@RequestParam("transfer_from") String transfer_from,
                           @RequestParam("transfer_to") String transfer_to,
                           @RequestParam("transfer_amount") String transfer_amount,
                           HttpSession session,
                           RedirectAttributes redirectAttributes) {
        //Check for empty strings
        if(transfer_from.isEmpty() || transfer_to.isEmpty() ||
transfer_amount.isEmpty()) {
            redirectAttributes.addFlashAttribute("error", "Transferring Account OR
Transferred Account OR Transfer Amount cannot be Empty");
```

```
            return "redirect:/app/dashboard";
        }
        //Get logged in user
        user = (User) session.getAttribute("user");
        //Converting variables
        int transferFromId = Integer.parseInt(transfer_from);
        int transferToId = Integer.parseInt(transfer_to);
        double transferAmount = Double.parseDouble(transfer_amount);
        //Check if user transferring into the same account
        if(transferFromId == transferToId) {
            redirectAttributes.addFlashAttribute("error", "Cannot Transfer into the
Same Account. Please Select an Appropriate Account for Transferring");
            return "redirect:/app/dashboard";
        }
        //Check if transfer amount is zero
        if (transferAmount == 0) {
            redirectAttributes.addFlashAttribute("error", "Transfer Amount cannot be
Zero (0)");
            return "redirect:/app/dashboard";
        }
        //Get current account balance
        double accountBalanceOfAccountTransferringFrom =
accountRepository.getAccountBalance(user.getUser_id(), transferFromId);
        //Check if transfer amount is more than current balance
        if(accountBalanceOfAccountTransferringFrom < transferAmount) {
            redirectAttributes.addFlashAttribute("error", "Insufficient Funds to
Perform this Transfer");
            //Log failed transaction
            transactRepository.logTransaction(transferFromId, "Transfer",
transferAmount, "Online", "Failure", "Insufficient Funds", currentDateTime);
```

```
            return "redirect:/app/dashboard";
        }
        double accountBalanceOfAccountTransferringTo =
accountRepository.getAccountBalance(user.getUser_id(), transferToId);
        //Set new balance
        double newBalanceOfAccountTransferringFrom =
accountBalanceOfAccountTransferringFrom - transferAmount;
        double newBalanceOfAccountTransferringTo =
accountBalanceOfAccountTransferringTo + transferAmount;
        //Change balance of account transferring from

accountRepository.changeAccountBalanceById(newBalanceOfAccountTransferri
ngFrom, transferFromId);
        //Change balance of account transferring to

accountRepository.changeAccountBalanceById(newBalanceOfAccountTransferri
ngTo, transferToId);
        //Update time
        accountRepository.setUpdatedAt(transferFromId);
        //Log successful transaction
        transactRepository.logTransaction(transferFromId, "Transfer",
transferAmount, "Online", "Success", "Amount Transfer Successful",
currentDateTime);
        redirectAttributes.addFlashAttribute("success", "Amount Transferred
Successfully!");
        return "redirect:/app/dashboard";
    }
    //Withdraw Method
    @PostMapping("/withdraw")
    public String withdraw(@RequestParam("account_id") String accountID,
                           @RequestParam("withdrawal_amount") String
withdrawalAmount,
```

```
                    HttpSession session,
                    RedirectAttributes redirectAttributes) {
        //Check for empty strings
        if(withdrawalAmount.isEmpty() || accountID.isEmpty()) {
            redirectAttributes.addFlashAttribute("error", "Withdrawal Account OR
Withdrawal Amount cannot be Empty");
            return "redirect:/app/dashboard";
        }
        //Convert variables
        double withdrawalAmountValue =
Double.parseDouble(withdrawalAmount);
        int account_id = Integer.parseInt(accountID);
        //Check for 0 values
        if(withdrawalAmountValue == 0) {
            redirectAttributes.addFlashAttribute("error", "Withdrawal Amount cannot
be Zero (0)");
            return "redirect:/app/dashboard";
        }
        //Get logged in user
        user = (User) session.getAttribute("user");
        //Get current account balance
        accountBalance = accountRepository.getAccountBalance(user.getUser_id(),
account_id);
        accountRepository.setUpdatedAt(account_id);
        //Check if withdraw amount is more than current balance
        if(accountBalance < withdrawalAmountValue) {
            redirectAttributes.addFlashAttribute("error", "Insufficient Funds to
Perform this Withdrawal");
            //Log failed transaction
```

```
            transactRepository.logTransaction(account_id, "Withdraw",
withdrawalAmountValue, "Online", "Failure", "Insufficient Funds",
currentDateTime);

            return "redirect:/app/dashboard";

        }

        //Set new balance

        newBalance = accountBalance - withdrawalAmountValue;

        //Update account balance

        accountRepository.changeAccountBalanceById(newBalance, account_id);

        //Log successful transaction

        transactRepository.logTransaction(account_id, "Withdraw",
withdrawalAmountValue, "Online", "Success", "Amount Withdraw Successful",
currentDateTime);

        redirectAttributes.addFlashAttribute("success", "Amount Withdrawn
Successfully!");

        return "redirect:/app/dashboard";

    }

}
```

6.8 Static Resources

admin_main.css -

```
/* Font styles */

@import
url('https://fonts.googleapis.com/css2?family=Anonymous+Pro&family=BioRhy
me:wght@400;700&family=Cormorant+SC:wght@300;400;500;600;700&family
=Cormorant:ital,wght@0,300;0,400;0,600;0,700;1,400&family=Eczar:wght@400
;600&family=Oswald:wght@200;300;500;700&family=Playfair+Display&family
=Quicksand:wght@300;400;500;700&family=Roboto:wght@100;300;400;500;7
00&display=swap');

* {

box-sizing: border-box;

font-family: 'Quicksand', sans-serif;

}

/* Main body */

body {

background-color: #fceffa;

}

/* Main page header */

.main-page-header {

padding: 10px 0px;

background-color: #6c2d60;

}

/* Company Logo */

.company-logo {

position: fixed;

top: 8px;

left: 60px;

z-index: 1;

}
```

```
/* Company name */
.company-name {
font-weight: bold;
color: #F2F2EE;
font-family: 'Oswald', sans-serif;
font-size: 20px;
}
.company-name span {
font-family: 'Cormorant SC', serif;
font-size: 23px;
position: relative;
bottom: 27px;
left: 92px;
color: #e1b7d9;
}
/* Navigation */
.navigation li {
display: inline-block;
list-style-type: none;
margin: 5px 15px;
}
.navigation li a {
color: white;
text-decoration: none;
font-family: 'Roboto', sans-serif;
}
```

default.css -

```
.navbar-selected {
    border: solid 2px #00393F;
    border-radius: 15px;
    background-color: #87BFDA;
}
/* Display Name */
.display-name {
    font-weight: bold;
}
/* Buttons */
.btn {
    font-family: 'Roboto', sans-serif;
}
/* Main body */
body.home {
    background-image: url("../images/home_pg_bg.png");
    height: 90vh;
    background-repeat: none;
    background-size: cover;
}
/* Card Main */
#main-card {
    margin: 180px 370px 0px 370px;
    font-family: 'Cormorant SC', serif;
    font-weight: 700;
    font-size: 20px;
}
```

```
/* Card heading */
#main-card h1 {
    font-family: 'Oswald', sans-serif;
    font-size: 66px;
    color: #29245b;
}
/* Card title */
#main-card h2 {
    font-family: 'Oswald', sans-serif;
    color: #5249b6;
}
/* main card paragraph */
#main-card p {
    font-family: 'Trirong', serif;
    font-size: 22px;
    font-weight: 600;
}
/* ADMINLOGIN.HTML STYLING */
#admin-login {
    background-image: url("../images/admin_login_pg_bg.png");
    height: 100vh;
    background-repeat: none;
    background-size: cover;
}
#admin-login-card {
    margin: 188px auto;
}
.admin-login-btn {
```

```
    background-color: #a24491;
    border: 2px solid #69305c;
    box-shadow: 2px 3px 6px #9d4889;
    font-family: 'Roboto', sans-serif;
    color: white;
    margin: 10px 0px;
    width: 200px;
}
```

main.css -

```
.offcanvas .offcanvas-title {
    font-family: 'Oswald', sans-serif;
}
/* Payment Card */
.payment-card {
    display: none
}
/* Transfer Card */
.transfer-card {
    display: none;
}
/* Deposit Card */
.deposit-card {
    display: none;
}
/* Withdraw Card */
.withdraw-card {
    display: none;
```

```
}
.payment-history .card {
    width: 1600px;
    margin: 0px auto;
}
.payment-history .card .card-header h2 {
    font-family: 'Oswald', sans-serif;
}
.payment-history table {
    border-left: 2px solid #413A92;
    border-right: 2px solid #413A92;
}
.payment-history table th {
    position: sticky;
    bottom: 70px;
    background-color: #413A92;
    color: white;
}
.payment-history table td, th {
    font-family: 'Anonymous Pro', monospace;
    font-size: 18px;
    padding: 10px 30px;
    vertical-align: middle;
}
.payment-history a {
    color: #666666;
    text-decoration: none;
}
```

main.js -

```
//Get Transaction Type
const transactType = document.getElementById('transact-type');
//Get Transaction Forms
const paymentCard = document.getElementsByClassName('payment-card')[0];
const transferCard = document.getElementsByClassName('transfer-card')[0];
const depositCard = document.getElementsByClassName('deposit-card')[0];
const withdrawCard = document.getElementsByClassName('withdraw-card')[0];
//Check for transaction type event listener
transactType.addEventListener('change', () => {
   //Check for transaction type and display form
   switch(transactType.value) {
      case 'payment':
         paymentCard.style.display = 'block';
         transferCard.style.display = 'none';
         depositCard.style.display = 'none';
         withdrawCard.style.display = 'none';
      break;
      case 'transfer':
         transferCard.style.display = 'block';
         paymentCard.style.display = 'none';
         depositCard.style.display = 'none';
         withdrawCard.style.display = 'none';
      break;
      case 'deposit':
         depositCard.style.display = 'block';
         paymentCard.style.display = 'none';
         transferCard.style.display = 'none';
```

```
            withdrawCard.style.display = 'none';
        break;
        case 'withdraw':
            withdrawCard.style.display = 'block';
            paymentCard.style.display = 'none';
            depositCard.style.display = 'none';
            transferCard.style.display = 'none';
        break;
        default:
            withdrawCard.style.display = 'none';
            paymentCard.style.display = 'none';
            depositCard.style.display = 'none';
            transferCard.style.display = 'none';
        break;
    }
    //End of Check for transaction type and display form
});
//End of Check for transaction type event listener
//Script for buttons returning from payment history / transaction history to
dashboard
const UrlString = window.location.search;
const UrlParam = new URLSearchParams(UrlString);
const isPayment = UrlParam.get('payment');
const isTransaction = UrlParam.get('transaction');
const isAddAccount = UrlParam.get('addAccount');
if(isPayment == 'true') {
    document.getElementById('transact-btn').click();
    const transactType2 = document.getElementById('transact-type');
    transactType2.selectedIndex = 1;
```

```
    paymentCard.style.display = 'block';
    transferCard.style.display = 'none';
    depositCard.style.display = 'none';
    withdrawCard.style.display = 'none';
}
if(isTransaction == 'true') {
    document.getElementById('transact-btn').click();
    console.log('Inside isTransaction');
}
if(isAddAccount == 'true') {
    document.getElementById('add-account-btn').click();
    console.log('Inside isAddAccount');
}
```

6.9 Views (JSP)

admin_header.jsp -

```
<!--Main Page Header-->
   <div class="main-page-header mb-3" id="header" style="position: fixed; z-index: 1; left: 0px; right: 0px; top: 0px;">
      <!--Container-->
      <div class="container-fluid d-flex align-items-center justify-content-center mb-2 mt-3">
         <!--Company name-->
         <div class="company-name">
            <span>( admin )</span> Cadmus Finance Corporation
         </div>
         <!--End of Company name-->
         <!--Navigation-->
         <div class="navigation ms-3 me-5">
            <li><a href="admin_dashboard">Dashboard</a></li>
            <li><a href="application_panel">Applications</a></li>
            <li><a href="employee_panel">Employees</a></li>
            <li><a href="customer_panel">Customers</a></li>
         </div>
         <!--End of Navigation-->
         <!-- Company Logo -->
            <a href="/app/admin_dashboard"><img src="images/logos/5.png" height="72px" style="position: fixed;" class="company-logo"></a>
         <!--End of Company Logo -->
         <!--Display Name-->
         <div class="display-name ms-5 text-white">
            <i class="fa fa-circle text-success me-2"></i> Welcome: Admin <span>${admin.admin_first_name} ${admin.admin_last_name}</span>
```

```
            </div>
            <!--End of Display Name-->
            <!--Signout button-->
            <a href="/adminlogout" class="btn btn-md text-white mt-1 ms-3">
                <i class="fa-solid fa-right-from-bracket ms-3 me-3"></i>Sign Out
            </a>
            <!--End of Signout button-->
        </div>
        <!--End of Container-->
    </div>
    <!--End of Main Page Header-->
    <div style="height:100px"></div>
```

footer.jsp -

```
<script src="../js/main.js"></script>
</body>
</html>
```

header.jsp -

```
<!--Main Page Header-->
    <div class="main-page-header mb-5" style="position: fixed">
        <!--Container-->
        <div class="container d-flex align-items-center">
            <!--Company name-->
            <div class="company-name">
                Cadmus Finance Corporation
            </div>
            <!--End of Company name-->
```

```
        <!--Navigation-->
        <nav class="navigation">
          <li><a href="dashboard">Dashboard</a></li>
                    <li><a href="payment_history">Payment History</a></li>
                    <li><a href="transact_history">Transaction
History</a></li>
                    <li><a href="loan">Loan</a></li>
        </nav>
        <!--End of Navigation-->
        <!--Display Name-->
                <div class="display-name ms-auto text-white">
                   <i class="fa fa-circle text-success me-2"></i> Welcome:
<span>${user.first_name} ${user.last_name}</span>
                </div>
                <!--End of Display Name-->
                <!--Signout button-->
                <a href="/logout" class="btn btn-md text-white ms-5">
                   <i class="fa-solid fa-right-from-bracket me-2"></i>Sign Out
                </a>
                <!--End of Signout button-->
      </div>
      <!--End of Container-->
   </div>
   <!--End of Main Page Header-->
   <div style="height: 100px">
   </div>
   <!-- Company Logo -->
   <a href="/app/dashboard"><img src="images/logos/4.png" height="52px"
style="position: fixed;" class="company-logo"></a>
```

home_loan.jsp -

```
<div class="offcanvas offcanvas-top col-5 mx-auto" onload="HLDefaultJob()" tabindex="-1" id="offcanvasHL" aria-labelledby="offcanvasHL" style="height: 87vh;">
    <div class="offcanvas-header">
      <h2 class="offcanvas-title" id="offcanvasHL">Home Loan Application</h2>
      <button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria-label="Close"></button>
    </div>
    <div class="offcanvas-body">
      <br>
      <div class="container pe-5">
        <div style="position: fixed;right: 590px;bottom: 140px;">
          <a href="#HL-submit-btn"><i class="fa-solid fa-angles-down fa-3x text-dark"></i></a>
        </div>
          <form action="/apply/home_loan" method="POST" id="HL-form" enctype="multipart/form-data">
            <div class="card bg-transparent">
              <div class="card-title mt-3 text-center">
                <h3><i class="fa-solid fa-id-card me-3"></i>Personal Information</h3>
              </div>
              <div class="card-body">
                <div class="form-group mb-3">
                  <label class="ms-1" for="">First Name:</label>
                  <input type="text" name="HL_UserFirstName" class="form-control" placeholder="Enter First Name">
                </div>
                <div class="form-group mb-3">
```

```
                        <label class="ms-1" for="">Last Name:</label>
                        <input type="text" name="HL_UserLastName"
class="form-control" placeholder="Enter Last Name">
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Email ID:</label>
                        <input type="text" name="HL_UserEmail" class="form-
control" placeholder="Enter Email Address">
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Contact Number:</label>
                        <input type="text" name="HL_UserContact" class="form-
control" placeholder="Enter Contact Number">
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Religion:</label>
                        <input type="text" name="HL_UserReligion" class="form-
control" placeholder="Enter Religion">
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Age:</label>
                        <div style="width: 100px;text-align: center;"><h5
class="bg-light border border-dark mt-2" id="HLAge"></h5> </div>
                        <input type="range" min="23" max="62" value="23"
class="slider" name="HL_UserAge" id="HL_UserAge">
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Birth Date:</label>
                        <input type="date" name="HL_UserBirthDate"
class="form-control" placeholder="Enter Birth Date">
                    </div>
```

```
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Marital Status:</label>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserMaritalStatus"
value="Married" class="form-check-input" id="MaritalStatus_radio1">
                            <label class="form-check-label"
for="MaritalStatus_radio1">Married</label>
                        </div>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserMaritalStatus"
value="Single" class="form-check-input" id="MaritalStatus_radio2">
                            <label class="form-check-label"
for="MaritalStatus_radio2">Single</label>
                        </div>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserMaritalStatus"
value="Divorced" class="form-check-input" id="MaritalStatus_radio3">
                            <label class="form-check-label"
for="MaritalStatus_radio3">Divorced</label>
                        </div>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserMaritalStatus"
value="Widowed" class="form-check-input" id="MaritalStatus_radio4">
                            <label class="form-check-label"
for="MaritalStatus_radio4">Widowed</label>
                        </div>
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1 mb-2" for="">Proof of Identity
(PDF):</label>
                        <input type="file" id=HL_UserProofIdentity
name="HL_UserProofIdentity" class="form-control" accept="application/pdf">
```

```
                    </div>
                    <div class="toast mx-auto mb-5" id="HL_error-toast-personal-info" data-bs-delay="10000" role="alert" aria-live="assertive" aria-atomic="true">
                        <div class="toast-header">
                        <h5 class="me-auto text-danger"><i class="fa-solid fa-triangle-exclamation me-3"></i>Error</h5>
                        <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
                        </div>
                        <div class="toast-body text-white bg-danger text-center" style="font-size: 17px" id="HL_error-toast-msg-personal-info">
                            Toast message
                        </div>
                    </div>
                </div>
                <div class="card bg-transparent mt-4">
                    <div class="card-title mt-3 text-center">
                        <h3><i class="fa-solid fa-house-circle-check me-3"></i>Property Information</h3>
                    </div>
                    <div class="card-body">
                        <div class="form-group mb-3">
                            <label class="ms-1" for="">Property Purchase Price:</label>
                            <input type="text" id="HL_UserPurchasePrice" name="HL_UserPurchasePrice" class="form-control" placeholder="Enter Purchase Price">
                        </div>
                        <div class="form-group mb-3">
```

```
                        <label class="ms-1" for="">Loan Amount
Requested:</label>

                        <input type="text" id="HL_UserLoanAmountRequested"
name="HL_UserLoanAmountRequested" class="form-control"
placeholder="Enter Loan Amount Requested">

                    </div>

                    <div class="form-group mb-3">

                        <label class="ms-1" for="">Down Payment
Amount:</label>

                        <div style="width: 400px;text-align: center;"><h5
class="bg-light border border-dark mt-2"
id="HL_UserDownPaymentAmount"></h5> </div>

                        <input type="hidden"
name="HL_UserDownPaymentAmount_hidden"
id="HL_UserDownPaymentAmount_hidden"/>

                    </div>

                    <div class="form-group mb-3">

                        <label class="ms-1" for="">Is your current home owned or
rented?</label>

                        <div class="form-check my-1">

                            <input type="radio" name="HL_UserCurrentHome"
value="Owned" class="form-check-input" id="CurrentHome_radio1">

                            <label class="form-check-label"
for="CurrentHome_radio1">Owned</label>

                        </div>

                        <div class="form-check my-1">

                            <input type="radio" name="HL_UserCurrentHome"
value="Rented" class="form-check-input" id="CurrentHome_radio2">

                            <label class="form-check-label"
for="CurrentHome_radio2">Rented</label>

                        </div>

                    </div>

                    <div class="form-group mb-3">
```

```
                        <label class="ms-1" for="">Do you need to sell your
current house before buying a new one?</label>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserSellCurrent"
value="Yes" class="form-check-input" id="SellCurrent_radio1">
                            <label class="form-check-label"
for="SellCurrent_radio1">Yes</label>
                        </div>
                        <div class="form-check my-1">
                            <input type="radio" name="HL_UserSellCurrent"
value="No" class="form-check-input" id="SellCurrent_radio2">
                            <label class="form-check-label"
for="SellCurrent_radio2">No</label>
                        </div>
                    </div>
                    <div class="form-group mb-3">
                        <label class="ms-1" for="">Motto of Purchase:</label>
                        <select id="HL_UserMottoPurchase"
name="HL_UserMottoPurchase" class="form-control" id="">
                            <option value="">-- Select Motto of Purchase --
</option>
                            <option value="Primary Residence">Primary
Residence</option>
                            <option value="Secondary Residence">Secondary
Residence</option>
                            <option value="Investment">Investment</option>
                        </select>
                    </div>
                    <div class="toast mx-auto mb-5" id="HL_error-toast-
property-info" data-bs-delay="10000" role="alert" aria-live="assertive" aria-
atomic="true">
                        <div class="toast-header">
```

```
                        <h5 class="me-auto text-danger"><i class="fa-solid fa-
triangle-exclamation me-3"></i>Error</h5>
                        <button type="button" class="btn-close" data-bs-
dismiss="toast" aria-label="Close"></button>
                        </div>
                        <div class="toast-body text-white bg-danger text-center"
style="font-size: 17px" id="HL_error-toast-msg-property-info">
                            Toast message
                        </div>
                    </div>
                </div>
                <div class="form-group my-4">
                    <button class="btn btn-lg loan-submit-btn" id="HL-submit-btn"
type="submit">Submit</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

PaymentHistoryDisplay.jsp -

```
<!-- Container -->
<div class="payment-history">
    <!-- Payment history card -->
    <div class="card shadow">
        <!-- Card header -->
        <div class="card-header">
            <h2><i class="fas fa-credit-card me-2" aria-hidden="true"></i>
Payment History</h2>
```

```
                </div>
                <!-- End of Card header -->
                <!-- Card body -->
                <div class="card-body">
                    <c:if test="${requestScope.payment_history != null}">
                    <!-- Payment History Table -->
                    <table class="table text-center table-striped">
                        <tr>
                            <th style="position: sticky; top:51px">Record Number</th>
                            <th style="position: sticky; top:51px">Date</th>
                            <th style="position: sticky; top:51px">Beneficiary</th>
                            <th style="position: sticky; top:51px">Beneficiary Acc.
No.</th>
                            <th style="position: sticky; top:51px">Beneficiary Bank</th>
                            <th style="position: sticky; top:51px">Amount</th>
                            <th style="position: sticky; top:51px">Status</th>
                            <th style="position: sticky; top:51px">Reference</th>
                            <th style="position: sticky; top:51px">Reason Code</th>
                            <th style="position: sticky; top:51px">Time</th>
                        </tr>
                        <!-- Loop Through Payment History Records -->
                        <c:forEach items="${requestScope.payment_history}"
var="payments">
                        <tr style="border-bottom: 2px solid #413A92;">
                            <td>${payments.payment_id}</td>
                            <td>${f:getFormattedDate(payments.created_at)}</td>
                            <td>${payments.beneficiary}</td>
                            <td>${payments.beneficiary_acc_no}</td>
                            <td>${payments.beneficiary_bank}</td>
```

```
                        <td>${payments.amount}</td>
                        <td>${payments.status}</td>
                        <td>${payments.reference_no}</td>
                        <td>${payments.reason_code}</td>
                        <td>${f:getFormattedTime(payments.created_at)}</td>
                    </tr>
                    </c:forEach>
                    <!-- End Of Loop Through Payment History Records -->
                </table>
                <!-- End Of Payment History Table -->
            </c:if>
            </div>
            <!-- End of Card body -->
            <!-- Card footer -->
            <div class="card-footer">
                <h4 class="my-3"><a href="payment_history_export_pdf"><i
class="fa-solid fa-download me-3"></i>Download Payment History as
PDF</a></h4>
            </div>
            <!-- End of Card footer -->
        </div>
        <!-- End of Payment history card -->
    </div>
    <!-- End of Container -->
```

PaymentHistoryNoDisplay.jsp -

```
<!-- Container -->
    <div class="container">
        <div class="no-payment-history">
            <!--No payment history card -->
            <div class="card shadow">
                <!-- Card header -->
                <div class="card-header">
                    <div style="position: relative;right: 70px;">
                        <i class="fa-solid fa-slash fa-4x my-2" style="position: relative;
left: 75px; top: 7px;color: red;"></i>
                        <i class="fa-solid fa-sack-dollar fa-4x"></i>
                        <h1 style="display: inline; position: relative; bottom: 7px; left:
15px;">NO Payments carried out</h1>
                    </div>
                </div>
                <!--End of Card header -->
                <!-- Card body -->
                <div class="card-body">
                    <!-- Card text -->
                    <div class="card-text">
                        You have not carried out any payments at the moment. <br>
                        To initiate a payment, please head to the transaction section or
click the link below.
                    </div>
                    <!--End of Card text -->
                    <!-- Payment transaction button -->
                    <a href="dashboard?payment=true"><button class="btn btn-md
my-3 px-4" type="button" style="background-color: #635bbe;border: 2px solid
#191637;box-shadow: 2px 3px 6px rgb(0, 0, 145);color: white; font-
```

```
size:20px"><i class="fa-regular fa-money-bill-1 me-3 fa-
xl"></i>Pay</button></a>
                    <!--End of Payment transaction button -->
                </div>
                <!--End of Card body -->
            </div>
            <!--End of No payment history card -->
        </div>
    <div class="container">
  <!--End of Container -->
```

deposit_form.jsp -

```
<!--Deposit card-->
        <div class="card deposit-card bg-transparent">
            <!--Card body-->
            <div class="card-body">
                <!-- Deposit form -->
                <form action="/transact/deposit" method="POST" class="deposit-
form">
                    <!--Select account option-->
                    <!-- <button class="btn dropdown-toggle my-3 mx-0"
type="button" id="accountDropdown" data-bs-toggle="dropdown" aria-
expanded="false">
                        Select Account
                    </button>
                    <ul class="dropdown-menu" aria-
labelledby="accountDropdown">
                        <li><a class="dropdown-item" value="">-- select account --
</a></li>
                    </ul> -->
```

```
                    <!--Form group-->
                    <div class="form-group">
                        <label for="">Select Account</label>
                        <select name="account_id" class="form-control my-3"
id="">
                            <option value="">-- Select Account --</option>
                            <c:if test="${userAccounts != null}">
                                <c:forEach items="${userAccounts}"
var="selectAccount">
                                    <option
value="${selectAccount.account_id}">${selectAccount.account_name}</option>
                                </c:forEach>
                            </c:if>
                        </select>
                    </div>
                    <!-- End of Form group-->
                <!--End of Select account option-->
                <!--Form group-->
                <div class="form-group mb-4">
                    <label for="">Enter Deposit Amount</label>
                    <input type="text" name="deposit_amount" class="form-control
mt-2" placeholder="Enter Deposit Amount"/>
                </div>
                <!--End of Form group-->
                <!--Form group-->
                <div class="form-group mb-4">
                    <button class="btn btn-md transaction-btn">Deposit</button>
                </div>
                <!--End of Form group-->
            </form>
```

```
            <!-- End of Deposit form -->
          </div>
          <!--End of Card body-->
        </div>
<!--End of Deposit card-->
```

account_display.jsp -

```
<div class="container d-flex">
<button data-bs-toggle="offcanvas" data-bs-target="#offcanvasaccount" class="btn btn-lg account-btn" id="add-account-btn" type="button"><i class="fa-regular fa-credit-card fa-xl me-2"></i>Add new Account</button>
<button data-bs-toggle="offcanvas" data-bs-target="#offcanvastransaction" class="btn btn-lg ms-auto transaction-btn" id="transact-btn" type="button"><i class="fa-solid fa-file-invoice-dollar me-2 fa-xl"></i>Transaction</button>
</div>
<div class="container d-flex my-4">
<h2 class="me-auto">Total Accounts Balance:</h2>
<h2 class="ms-auto">
<c:if test="${requestScope.totalBalance != null}">
<c:out value="${totalBalance}" />
</c:if>
</h2>
</div>
<div class="container my-4">
<c:if test="${requestScope.userAccounts != null }">
<c:forEach items="${requestScope.userAccounts}" var="account">
<div class="accordion accordion-flush" id="accordionFlushExample">
<div class="accordion-item">
<h2 class="accordion-header" id="flush-headingOne">
```

```
<button class="accordion-button collapsed" type="button" data-bs-
toggle="collapse" data-bs-target="#flush-${account.account_id}" aria-
expanded="false" aria-controls="flush-collapseOne" style="font-size: 19px;">

${account.account_name}

</button>

</h2>

<div id="flush-${account.account_id}" class="accordion-collapse collapse" aria-
labelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">

<div class="accordion-body">

<ul class="list-group list-group-flush">

<li class="list-group-item d-flex">Account Name <span class="ms-
auto"><b>${account.account_name}</b></span></li>

<li class="list-group-item d-flex">Account Number <span class="ms-
auto"><b>${account.account_number}</b></span></li>

<li class="list-group-item d-flex">Account Type <span class="ms-
auto"><b>${account.account_type}</b></span></li>

<li class="list-group-item d-flex">Account Balance <span class="ms-
auto"><b>${account.balance}</b></span></li>

<li class="list-group-item d-flex">Created at Date <span class="ms-
auto"><b>${f:formatLocalDateTime(account.created_at,
'yyyy/MM/dd')}</b></span></li>

</ul>

</div>

</div>

</div>

</div>

</c:forEach>

</c:if>

</div>
```

add_account_offcanvas.jsp -

```
<!--Accounts Offcanvas-->
   <div class="offcanvas offcanvas-top col-5 mx-auto" tabindex="-1" id="offcanvasaccount" aria-labelledby="offcanvasaccount">
      <!--Offcanvas Header-->
      <div class="offcanvas-header">
         <h2 class="offcanvas-title" id="offcanvasaccount">Create or Add an Account</h2>
         <button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria-label="Close"></button>
      </div>
      <!--End of Offcanvas Header-->
      <!--Offcanvas Body-->
      <div class="offcanvas-body">
         <br>
         <p class="card-text my-2">
            Select one of the options below to perform a transaction
         </p>
         <!--Account form card-->
         <div class="card bg-transparent">
            <!--Card body-->
            <div class="card-body">
               <form action="/account/create_account" method="POST" class="add-account-form">
                  <!--Form group-->
                  <div class="form-group mb-3">
                     <label for="">Select Account Type</label>
                     <select name="account_type" class="form-control" id="">
                        <option value="">-- Select Account Type --</option>
                        <option value="Savings">Savings</option>
```

```
                        <option value="Business">Business</option>
                    </select>
                </div>
                <!--End of Form group-->
                <!--Form group-->
                <div class="form-group mb-3">
                    <label for="">Enter Account Name</label>
                    <input type="text" name="account_name" class="form-control"
placeholder="Enter Account Name">
                </div>
                <!--End of Form group-->
                <!--Form group-->
                <div class="form-group mb-4">
                    <button class="btn btn-md account-btn">Add Account</button>
                </div>
                <!--End of Form group-->
            </form>
        </div>
        <!--End of Card body-->
    </div>
    <!--End of Account form card-->
</div>
<!--End of Offcanvas Body-->
<!--Script for blurring effect-->
<script>
    const offcanvasAcc =
        document.getElementById('offcanvasaccount')
    offcanvasAcc.addEventListener('show.bs.offcanvas', () => {
        document.getElementById('blur-effect').style.filter = "blur(6px)";
```

```
        })
        offcanvasAcc.addEventListener('hide.bs.offcanvas', () => {
            document.getElementById('blur-effect').style.filter = "blur(0px)";
        })
    </script>
    <!--End of Script for blurring effect-->
  </div>
<!--End of Accounts Offcanvas-->
```

loan_display.jsp -

```
<div class="bg-image jumbotron container mt-4 shadow">
<div class="justify-content-center d-flex">
<button class="btn btn-grad mx-5" type="button" data-bs-toggle="collapse" data-bs-target="#home-loan" aria-expanded="false"
aria-controls="home-loan">
Home Loan
</button>
<button class="btn btn-grad mx-5" type="button" data-bs-toggle="collapse" data-bs-target="#personal-loan" aria-expanded="false"
aria-controls="personal-loan">
Personal Loan
</button>
<button class="btn btn-grad mx-5" type="button" data-bs-toggle="collapse" data-bs-target="#gold-loan" aria-expanded="false"
aria-controls="gold-loan">
Gold Loan
</button>
</div>
</div>
```

```
<div class="container py-5">
<c:if test="${success != null}">
<div class="alert alert-success text-center border border-success">
<b>${success}</b>
</div>
</c:if>
<c:if test="${error != null}">
<div class="alert alert-danger text-center border border-danger">
<b>${error}</b>
</div>
</c:if>
</div>
<div id="myGroup">
<c:import url="components/loan_panels/homeLoan.jsp"/>
<c:import url="components/loan_panels/personalLoan.jsp"/>
<c:import url="components/loan_panels/goldLoan.jsp"/>
</div>
<c:import url="components/loan_panels/offcanvas/home_loan.jsp"/>
<c:import url="components/loan_panels/offcanvas/personal_loan.jsp"/>
<c:import url="components/loan_panels/offcanvas/gold_loan.jsp"/>
<script src="js/main.js"></script>
```

transaction_offcanvas.jsp -

```
<!--Transaction Offcanvas-->
   <div class="offcanvas offcanvas-top col-5 mx-auto" tabindex="-1"
id="offcanvastransaction" aria-labelledby="offcanvastransaction">
      <!--Offcanvas Header-->
      <div class="offcanvas-header">
         <h2 class="offcanvas-title" id="offcanvastransaction">Transaction</h2>
```

```
        <button type="button" class="btn-close text-reset" data-bs-
dismiss="offcanvas" aria-label="Close"></button>
      </div>
      <!--End of Offcanvas Header-->
      <!--Offcanvas Body-->
      <div class="offcanvas-body">
        <br>
        <p class="card-text my-2">
          Select one of the options below to perform a transaction
        </p>
        <!--Transaction type drop down-->
        <!-- <button class="btn dropdown-toggle my-3 mx-0" type="button"
id="transactDropdown" data-bs-toggle="dropdown" aria-expanded="false">
          Select Transaction Type
        </button>
        <ul class="dropdown-menu    " aria-labelledby="transactDropdown"
id="transact-type">
          <li><a class="dropdown-item" value="payment">Payment</a></li>
          <li><a class="dropdown-item" value="transfer">Transfer</a></li>
          <li><a class="dropdown-item" value="deposit">Deposit</a></li>
          <li><a class="dropdown-item" value="withdraw">Withdraw</a></li>
        </ul> -->
        <select name="transact-type" class="form-control my-3" id="transact-
type">
          <option value="">-- Select Transaction Type --</option>
          <option value="payment">Payment</option>
          <option value="transfer">Transfer</option>
          <option value="deposit">Deposit</option>
          <option value="withdraw">Withdraw</option>
        </select>
```

```
            <!--End of Transaction type drop down-->
            <!-- Payment form card -->
            <c:import url="components/transact_forms/payment_form.jsp"/>
            <!-- Transfer form card -->
            <c:import url="components/transact_forms/transfer_form.jsp"/>
            <!-- Deposit form card -->
            <c:import url="components/transact_forms/deposit_form.jsp"/>
            <!-- Withdraw form card -->
            <c:import url="components/transact_forms/withdraw_form.jsp"/>
        </div>
        <!--End of Offcanvas Body-->
        <!--Script for blurring effect-->
        <script>
            const offcanvasTransac =
                document.getElementById('offcanvastransaction')
            offcanvasTransac.addEventListener('show.bs.offcanvas', () => {
                document.getElementById('blur-effect').style.filter = "blur(6px)";
            })
            offcanvasTransac.addEventListener('hide.bs.offcanvas', () => {
                document.getElementById('blur-effect').style.filter = "blur(0px)";
            })
        </script>
        <!--End of Script for blurring effect-->
    </div>
<!--End of Transaction Offcanvas-->
```

admindashboard.jsp -

```
<html lang="en">
<body>
<c:import url="components/include/admin_header.jsp"/>
<div class="container mt-4">
<div class="card shadow">
<div class="card-body">
<h1 class="card-title">
Application Section
</h1>
<hr>
<div class="card-text">
<ol class="list-group  mx-auto my-3" style="width: 60%;">
<li class="list-group-item border-bottom border-dark d-flex justify-content-
between align-items-start">
<div class="ms-2 me-auto">
<h2 class="fw-bold" style="font-family: 'Oswald', sans-serif; color:
#4C2B3E"><i class="fa-solid fa-house fa-1x me-3 "></i>Home Loan
Applications</h2>
<ul type="disc" style="color: #4C2B3E; margin: 20px 10px">
<c:if test="${requestScope.allUsersAppliedHomeLoan != null}">
<c:forEach items="${allUsersAppliedHomeLoan}" var="application">
<h4><li style="font-family: 'BioRhyme', serif;">Application Number:
${application.application_number}</li></h4>
</c:forEach>
</c:if>
</ul>
</div>
<span class="badge bg-primary rounded-pill mt-3 me-3" style="font-size: 18px;
padding: 10px 30px">${requestScope.homeLoanCount}</span>
```

```
</li>

<li class="list-group-item border-bottom border-dark d-flex justify-content-
between align-items-start">

<div class="ms-2 me-auto">

<h2 class="fw-bold" style="font-family: 'Oswald', sans-serif; color:
#4C2B3E"><i class="fa-solid fa-sack-dollar fa-1x me-3 "></i>Personal Loan
Applications</h2>

<ul type="disc" style="color: #4C2B3E; margin: 20px 10px">

<c:if test="${requestScope.allUsersAppliedPersonalLoan != null}">

<c:forEach items="${allUsersAppliedPersonalLoan}" var="application">

<h4><li style="font-family: 'BioRhyme', serif;">Application Number:
${application.application_number}</li></h4>

</c:forEach>

</c:if>

</ul>

</div>

<span class="badge bg-primary rounded-pill mt-3 me-3" style="font-size: 18px;
padding: 10px 30px">${requestScope.personalLoanCount}</span>

</li>

<li class="list-group-item border-bottom border-dark d-flex justify-content-
between align-items-start">

<div class="ms-2 me-auto">

<h2 class="fw-bold" style="font-family: 'Oswald', sans-serif; color:
#4C2B3E"><i class="fa-solid fa-hand-holding-dollar fa-1x me-3"></i>Gold Loan
Applications</h2>

<ul type="disc" style="color: #4C2B3E; margin: 20px 10px">

<c:if test="${requestScope.allUsersAppliedGoldLoan != null}">

<c:forEach items="${allUsersAppliedGoldLoan}" var="application">

<h4><li style="font-family: 'BioRhyme', serif;">Application Number:
${application.application_number}</li></h4>

</c:forEach>
```

```
</c:if>
</ul>
</div>
<span class="badge bg-primary rounded-pill mt-3 me-3" style="font-size: 18px;
padding: 10px 30px">${requestScope.goldLoanCount}</span>
</li>
</ol>
</div>
</div>
</div>
</div>
<div class="container mt-4 d-flex">
<div class="container">
<div class="card shadow">
<div class="card-body">
<h1 class="card-title">
Employee DB
</h1>
<hr>
<table class="table">
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Job Title</th>
<th>Hire Date</th>
<th>Department</th>
<th>Branch</th>
```

```
</tr>
</thead>
<c:forEach items="${requestScope.employeeViews}" var="employee">
<tr style="border-bottom: 2px solid #481e40;">
<td>${employee.employee_id}</td>
<td>${employee.name}</td>
<td>${employee.age}</td>
<td>${employee.job_title}</td>
<td>${employee.hire_date}</td>
<td>${employee.department_name}</td>
<td>${employee.branch_name}</td>
</tr>
</c:forEach>
</table>
</div>
</div>
</div>
<div class="container">
<div class="card shadow">
<div class="card-body">
<h1 class="card-title">
Customer DB
</h1>
<hr>
<table class="table">
<thead>
<tr>
<th>ID</th>
```

```
<th>Name</th>
<th>Email</th>
<th>Account Count</th>
</tr>
</thead>
<c:forEach items="${requestScope.userViews}" var="user">
<tr style="border-bottom: 2px solid #481e40;">
<td>${user.user_id}</td>
<td>${user.name}</td>
<td>${user.email}</td>
<td>${user.account_count}</td>
</tr>
</c:forEach>
</table>
</div>
</div>
</div>
</div>
```

applicationPanel.jsp -

```
<html lang="en">
<body>
<c:import url="components/include/admin_header.jsp"/>
<ul class="nav nav-tabs ms-3"
style="--bs-nav-tabs-border-color: hsl(325, 28%, 23%); --bs-nav-tabs-link-hover-border-color: #4C2B3E; --bs-nav-tabs-link-active-bg: #fe8a80; --bs-nav-tabs-link-active-border-color: #664b00 "
id="myTab" role="tablist">
<li class="nav-item" role="presentation">
```

```
<button class="nav-link" id="disabled-tab"
style="--bs-nav-tabs-link-active-bg: white;" data-bs-toggle="tab"
data-bs-target="#disabled-tab-pane" type="button" role="tab"
aria-controls="disabled-tab-pane" aria-selected="false"> </button>
</li>
<li class="nav-item" role="presentation">
<button class="nav-link active"
style="--bs-nav-link-color: #4C2B3E; --bs-nav-link-font-weight: 700; font-
family: 'Roboto', sans-serif;"
id="home-loan-tab" data-bs-toggle="tab" data-bs-target="#home-loan-tab-pane"
type="button"
role="tab" aria-controls="home-loan-tab-pane" aria-selected="true">Home
Loan</button>
</li>
<li class="nav-item" role="presentation">
<button class="nav-link"
style="--bs-nav-link-color: #4C2B3E; --bs-nav-link-font-weight: 700; font-
family: 'Roboto', sans-serif;"
id="loan-log-tab" data-bs-toggle="tab" data-bs-target="#loan-log-tab-pane"
type="button"
role="tab" aria-controls="loan-log-tab-pane" aria-selected="false">Loan
logs</button>
</li>
</ul>
<fmt:setLocale value="en_IN" />
<div class="tab-content" id="myTabContent">
<div class="tab-pane fade show active" id="home-loan-tab-pane" role="tabpanel"
aria-labelledby="home-loan-tab"
tabindex="0">
<div class="container my-4">
```

```
<h2 class="me-auto" style="font-family: 'Oswald', sans-serif;font-size: 45px;
color: #4C2B3E"><i class="fa-solid fa-house fa-1x me-3 "></i>Home Loan
Applications</h2>

</div>

<div class="container my-4">

<c:if test="${requestScope.allHomeLoanApplications != null }">

<c:forEach items="${requestScope.allHomeLoanApplications}"
var="application">

<div class="accordion accordion-flush" id="accordionFlushExample">

<div class="accordion-item">

<h2 class="accordion-header d-flex" id="flush-headingOne">

<button class="accordion-button collapsed" type="button" data-bs-
toggle="collapse" data-bs-target="#flush-${application.application_number}"
aria-expanded="true" aria-controls="flush-collapseOne" style="font-size:
19px;font-family: 'Trirong', serif;">

${fn:toUpperCase(application.first_name)}
${fn:toUpperCase(application.last_name)}

</button>

</h2>

<div id="flush-${application.application_number}" class="accordion-collapse
collapse <c:if test='${application.application_number ==
param.application_number}'>show</c:if>" aria-labelledby="flush-headingOne"
data-bs-parent="#accordionFlushExample">

<div class="accordion-body">

<c:if test="${requestScope.error != null}">

<div class="alert alert-danger text-center border border-danger">

<b>${requestScope.error}</b>

</div>

</c:if>

<c:choose>

<c:when test="${application.approved == 'no'}">
```

```
<div class="container my-5 p-4 alert-warning border-warning border rounded" style="font-size:25px;">

<form action="/app/application_panel/approve/home_loan" method="POST" class="d-flex">

<div class="form-group mb-3">

<label for="">Loan Amount</label>

<div class="input-group">

<span class="input-group-text"><i class="fa-solid fa-indian-rupee-sign me-1"></i></span>

<input type="text" class="form-control" name="loan_amount" />

<span class="input-group-text">.00</span>

</div>

</div>

<div class="form-group ms-auto mb-3">

<label for="">Interest Rate</label>

<div class="input-group">

<span class="input-group-text"><i class="fa-solid fa-percent me-1"></i></span>

<input type="text" class="form-control" name="interest_rate" />

<span class="input-group-text">p.a.</span>

</div>

</div>

<div class="form-group ms-auto mb-3">

<label for="">Tenure</label>

<div class="input-group">

<span class="input-group-text"><i class="fa-solid fa-hourglass-start me-1"></i></span>

<input type="text" class="form-control" name="tenure" />

<span class="input-group-text">years</span>

</div>

</div>
```

```
<input type="hidden" value="${application.application_number}"
name="application_number" />

<button class="btn btn-outline-success ms-auto px-4 my-auto" style="height:
50px;font-weight: bold;" type="submit"><i class="fa-regular fa-circle-check fa-lg
me-2"></i>Approve</button>

</form>

</div>

</c:when>

<c:otherwise>

<div class="container text-center my-5 p-2 alert-success border-success border
rounded" style="font-size:25px;font-weight: bold">

This loan has been approved

</div>

</c:otherwise>

</c:choose>

<div class="container mt-5 p-3 border border-dark rounded" style="width: 70%">

<div class="tab-content" id="myTabContent">

<div class="tab-pane fade" id="loan-log-tab-pane" role="tabpanel" aria-
labelledby="loan-log-tab" tabindex="0">

<div class="mx-auto px-5 my-5" >

<c:if test="${requestScope.allLoanLogs != null}">

<table id='loan-log-table' class="table text-center table-striped">

<tr style="position: sticky; top:80px; background-color: #481e40;color:
white;font-family: 'Oswald', sans-serif; text-align: center;">

<th style="font-family: 'Oswald', sans-serif;text-align:center; color: white">Loan
Application Number</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color: white">Loan
Type</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Borrower Name</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Borrower User ID</th>
```

```
<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Borrower Account ID</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Borrower Email</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color: white">Loan
Amount</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Interest Rate</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Tenure</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">EMI</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Amount Payable</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Interest Payable</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Charges Payable</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color: white">Late
Payment Penalty</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color: white">Pre
Payment Penalty</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Confirmation</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Creation Time</th>

<th style="font-family: 'Oswald', sans-serif;text-align:center; color:
white">Update Time</th>

</tr>

<c:forEach items="${requestScope.allLoanLogs}" var="loan">

<tr style="border-bottom: 2px solid #481e40">

<td>${loan.loan_application_number}</td>

<td>${loan.loan_type}</td>
```

```
<td>${loan.borrower_name}</td>
<td>${loan.borrower_user_id}</td>
<td>${loan.borrower_account_id}</td>
<td>${loan.borrower_email}</td>
<td>${loan.final_loan_amount}</td>
<td>${loan.final_interest_rate}</td>
<td>${loan.final_tenure}</td>
<td>${loan.final_emi}</td>
<td>${loan.total_amount_payable}</td>
<td>${loan.total_interest_payable}</td>
<td>${loan.charges_payable}</td>
<td>${loan.late_payment_penalty}</td>
<td>${loan.pre_payment_penalty}</td>
<td>${loan.is_confirmed}</td>
<td>${fn:replace(loan.created_at, 'T', ' ')}</td>
<td>${fn:replace(loan.updated_at, 'T', ' ')}</td>
</tr>
</c:forEach>
</table>
</c:if>
</div>
</div>
</div>
</body>
</html>
```

dashboard.jsp -

```
<html lang="en">
<body>
    <div id="blur-effect">
        <!-- Header -->
        <c:import url="components/include/header.jsp"/>
        <!-- Container -->
        <div class="container">
            <!-- Display Message -->
            <c:if test="${success != null}">
                <div class="alert alert-success text-center border border-success">
                    <b>${success}</b>
                </div>
            </c:if>
            <!-- End of Display Message -->
            <!-- Display Message -->
            <c:if test="${error != null}">
                <div class="alert alert-danger text-center border border-danger">
                    <b>${error}</b>
                </div>
            </c:if>
            <!-- End of Display Message -->
        </div>
        <!-- End of Container -->
        <c:choose>
            <c:when test="${fn:length(userAccounts) > 0}">
                <!-- Display Accounts -->
                <c:import url="components/account_display.jsp"/>
```

```
            </c:when>
            <c:otherwise>
               <!-- Dont Display Accounts -->
               <c:import url="components/no_account_display.jsp"/>
            </c:otherwise>
         </c:choose>
      </div>
      <!-- Transaction Offcanvas -->
      <c:import url="components/transaction_offcanvas.jsp"/>
      <!-- Add Accounts Offcanvas -->
      <c:import url="components/add_account_offcanvas.jsp"/>
<!-- Footer -->
<c:import url="components/include/footer.jsp"/>
```

error.jsp -

```
<html lang="en">
<body class="d-flex align-items-center justify-content-center">
   <!--Error card-->
   <div class="card col-4 alert alert-danger border-danger text-danger">
      <!--Card title-->
      <h3 class="card-title">
         <i class="fa fa-window-close me-3"></i>Errors:
      </h3>
      <hr>
      <!--End of Card title-->
      <!--Card body-->
      <div class="card-body">
         <!--Card text-->
```

```
            <p class="card-text">
                <!-- Display Message -->
                    <c:if test="${requestScope.error != null}">
                        <div class="alert alert-danger text-center border border-danger">
                            <b>${requestScope.error}</b>
                        </div>
                    </c:if>
                    <!-- End of Display Message -->
            </p>
            <!--End of Card text-->
        </div>
        <!--End of Card body-->
        <hr>
        <!--Card text-->
        <div class="card-text">
            <!--Back to login page-->
            <a href="/login" class="btn btn-md btn-danger">
                <i class="fa fa-arrow-alt-circle-left me-3"></i>Back
            </a>
            <!--End of Back to login page-->
        </div>
        <!--End of Card text-->
    </div>
    <!--End of Error card-->
</body>
</html>
```

loan.jsp -

```
<html lang="en">
<body>
    <c:import url="components/include/header.jsp"/>
        <c:choose>
            <c:when test="${fn:length(userAccounts) == 0}">
                <c:import url="components/no_account_display_view.jsp"/>
            </c:when>
            <c:otherwise>
                <c:choose>
                    <c:when test="${totalAccountBalance >= 50000}">
                        <c:import url="components/loan_display.jsp"/>
                    </c:when>
                    <c:otherwise>
                        <c:import url="components/no_loan_display.jsp"/>
                    </c:otherwise>
                </c:choose>
            </c:otherwise>
        </c:choose>
</body>
</html>
```

7.0 Testing

Software testing is a critical element of software quality assurance and represents ultimate review of specification, design and coding. Testing is a well-planned series of steps that result in the successful construction of the software. The following section describes the tests carried out on the Banking Management System.

7.1 Unit Testing

Unit testing is a type of software testing that focuses on individual units or components of a software system. The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements. The primary goal of unit testing is to identify and fix defects in the code early in the development process, which helps improve the overall quality of the software. By testing each unit separately, developers can identify issues quickly and pinpoint the exact location of the problem, making it easier to debug and maintain the codebase.

7.2 Integration Testing

Integration testing is used to verify the correct functioning of multiple components or modules when they are integrated together. The purpose of integration testing is to detect any defects or issues that may arise due to the interaction between different units or subsystems within a larger system.

Integration testing focuses on the interfaces and interactions between the integrated components. It ensures that the components work together as intended, exchange data correctly, and maintain the expected behaviour in a unified environment.

7.3 System Testing

System testing is a software testing phase that focuses on evaluating the complete and integrated system as a whole. It is performed after integration testing and involves testing the entire system, including all components, subsystems, and their interactions, to ensure that it meets the specified requirements and functions as expected in a real-world environment.

System testing validates the stability and reliability of the system by subjecting it to various scenarios, inputs, and environmental conditions. It helps uncover any issues related to system crashes, data corruption, memory leaks, or other system failures.

7.4 Acceptance Testing

Acceptance testing, also known as User Acceptance Testing (UAT) or End-User Testing, is a type of software testing that verifies whether a system meets the requirements and expectations of its intended users or stakeholders. It is conducted to determine if the system is ready for deployment and use in the real-world environment.

Acceptance testing ensures that the system fulfils the specified business requirements and objectives. Acceptance testing assesses the usability and user-friendliness of the system. It evaluates the system's ease of use, user interface, navigation, and overall user experience to ensure that it meets the needs of the end users.

7.5 Regression Testing

Regression testing is a software testing technique that is performed to validate that changes or modifications in a software application have not introduced new defects or caused any unintended side effects in previously tested functionalities. It aims to ensure that the existing system functionalities continue to work as expected after changes have been made.

Regression testing assesses the impact of changes made in one part of the software on other interconnected components. It ensures that modifications do not cause unexpected behaviour or errors in other parts of the system. The testing process involves selecting the relevant test cases from the existing test suite that cover the functionalities affected by the changes and then verified that the modified software works correctly.

8.0 Future Enhancements

As technology continues to advance at a rapid pace, the world is constantly exploring new ways to enhance its products or services to stay ahead in the competitive landscape. Future enhancements play a crucial role in driving innovation, improving efficiency, and delivering enhanced experiences to customers. With that in mind, the following section lists the future enhancements of the Banking Management System.

8.1 Automated KYC

Automated KYC (Know Your Customer) is a future enhancement that can streamline and expedite the customer onboarding process while ensuring compliance with regulatory requirements. By leveraging advanced technologies and data analysis techniques, automated KYC can enhance the efficiency, accuracy, and security of the KYC process.

Automated KYC can employ optical character recognition (OCR) and a recognition machine learning model to extract relevant information from customer documents such as Aadhar Cards, Pan Cards, etc. This automated data capture eliminates manual data entry errors and speeds up the process.

8.2 Fraud Detection Model

The rise of online transactions and digital financial systems has unfortunately been accompanied by an increase in fraudulent activities. Detecting and preventing fraud has become a critical concern for businesses and financial institutions.

Machine learning models for fraud detection leverage advanced algorithms and data analytics techniques to automatically identify patterns, anomalies, and suspicious activities within vast amounts of transactional data. These models can learn from historical data and continuously adapt to new fraud patterns, enabling organizations to stay one step ahead of fraudsters.

8.3 Integrating Real World Monetary Transactions

At the moment, the Banking Management System deals with fictional currency but incorporating real world finance in the near future holds significant potential to enhance the user experience and expand the scope of financial interactions.

9.0 Conclusion

In conclusion, the development and implementation of the banking management system have proven to be a significant milestone in enhancing the efficiency, accuracy, and security of banking operations. The project has successfully addressed the diverse needs of customers, administrators, and the implementation of automated tasks within the banking industry.

The banking management system provides a user-friendly interface that allows customers to conveniently access their accounts, perform transactions, and apply for a range of loans. It offers a seamless and efficient banking experience, reducing manual processes and streamlining operations. Customers can easily monitor their account balances, view transaction history, transfer funds, and initiate various financial activities at their convenience.

The system also equips bank administrators with powerful tools to efficiently manage customer accounts, track transactions, monitor employees, look after branches and carry out other banking operations. The banking management system thus offers a range of features and functionalities that cater to both customers and bank administrators.

The production of the Banking Management System has been a remarkable undertaking that has expanded my understanding and expertise in this rapidly evolving domain. Throughout this project, I have explored various facets of IT, from cutting-edge technologies to emerging trends and their impact on businesses and society as a whole. My research and analysis have provided me with valuable insights into the challenges and opportunities that arise in the IT landscape. The project has allowed me to apply theoretical knowledge to practical scenarios, honing our problem-solving skills and critical thinking abilities.

10.0 Bibliography

During the course of making this project, an exhaustive list of resource material was utilized. Following is the list of websites referred and studied at the time of creating the Banking Management System:

https://bankofindia.co.in/home

https://sbi.co.in/web/personal-banking/home

https://www.pnbindia.in/downloadprocess.aspx?fid=zvdktQSSd4WoieAHB2eM+Q==

https://www.unionbankofindia.co.in/english/home.aspx

https://www.hdfcbank.com/

https://www.icicibank.com/

https://www.kotak.com/en/home.html

https://www.axisbank.com/

https://canarabank.com/

https://exadel.com/news/how-machine-learning-is-used-in-finance-and-banking/

https://fayrix.com/blog/machine-learning-in-finance

https://www.tutorialspoint.com/bank_management/bank_management_introduction.htm

https://www.bankbazaar.com/

https://wordpress.com/about/

https://www.aboutamazon.com/about-us

https://studioalto.com/about/

https://mycolor.space/

https://www.w3schools.com/colors/colors_picker.asp

Essential Links:

GitHub Repository:
https://github.com/Sharvil18/Cadmus_Financial_Corporation.git

Project Tracker:
https://docs.google.com/spreadsheets/d/1bQ9kKfoAaT9zfbgYB8HdwXSMJ2aoHJFV0s_JJJ41g5U/edit?usp=sharing