Software Engineering
Software Requirements Specification (SRS) Document

L.A.S. Banking System

November 2, 2023

Version V.2

By: RongXin(Leo) Liu, Sarvesh Sridher, Saul(Adam) Juarez

We have abided by the UNCG Academic Integrity Policy

Table of Contents

1. Introduction

1.1. Purpose

The L.A.S.(Leo, Adam, and Sarvesh Banking) banking system is a banking management application. The system is going to be focused on easy-to-use formatting of various actions that customers/employees can perform on certain accounts. L.A.S. will eliminate the need for manual, tedious processes in banking. Such actions include checking account balances, transferring funds, applying for loans, approving/denying loans(a tellers-specific function), etc. The main goal of this application is to provide an efficient way for customers and employees to carry out their banking needs, allowing both bankers and customers to save time. The LASB System will be available as a web application, accessible through any internet browser,

1.2. Document Conventions

This Software Requirements Document (SRD) aims to describe the client-view and developer-view requirements for the L.A.S.B. system. The document will include an array of information that is both important to stakeholders and developers. In it, we will describe various requirements of the application, including functional and non-functional requirements, technical requirements, etc. all of which will describe the application from a user and/or developer perspective.

1.3. Definitions, Acronyms, and Abbreviations

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the banking system application.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and create the content of the web application.
CSS	Cascading Style Sheets. This style sheet language will be used to design our web application.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
Intellij	An integrated development environment (IDE) for Java. This is where our system will be created.
GSON	GSON library for converting data to JSON files. This will be used to help save Java

	objects to our database without messy conversions.
API	Application Programming Interface. This will be used to implement a function within the software where the current date and time are displayed on the homepage.
Google Map API	Google Map API is an API created by Google to feed data. This will be used to show our main office location on the web application.
Exchange Rates API	This Exchange Rates API will help us when a customer wants to withdraw money that may be in a currency other than the U.S. dollar.

1.4. Intended Audience

This subsection is dedicated to informing you, our audience, which portions of this document would be of higher interest to you.

Potential Clients:
- Sections 1.1-1.5
- Sections 2.1-2.4

Developers:
- Sections 1.3, 1.4, 1.6
- Sections 2.2, 2.4-2.6
- Sections 3, 5 & 6

Legal:
- Section 1.7
- Sections 5.2 & 5.3

1.5. Project Scope

Our goal for this application is to provide an easy-to-use interface for all customers, employees, and managers of a bank, as well as provide our customers with the flexibility to meet their needs. This will align with the overall banking system goals, as a bank requires a safe and dependable service in order to fulfill the needs of its customers.

The benefits of the project to business include:
- The relief of having a secure place to store your money and having it protected by a banking system.
- Help exchange between different currencies around the globe to enhance their customers' financial satisfaction.
- Our application provides a range of services, such as acquiring a credit card, applying for loans, accessing savings accounts, and accessing checking accounts. This makes it simple for clients to manage their money in one place.

1.6. Technology Challenges

Currently, everything is going smoothly and well.

1.7. References

Currently, we have no references yet.

2. General Description

2.1. Product Perspective

L.A.S. Banking System: its origin was made by three visionaries: Leo, Adam, and Sarvesh. The traditional banking system has always been complex, with multiple steps to achieve a single task. The paperwork, long queues, and manual processes have often deterred customers from with their bank. The primary objective of the L.A.S. is to eliminate the cumbersome processes associated with traditional banking. With a focus on user experience, the system simplifies tasks such as checking balances, funds transfers, credit card applications, and loan applications. The incorporation of features like loan approval/denial and credit card approval/denial for banking employees aims to empower employees and make their jobs easier. Saving precious time for both customers and bank employees.

2.2. Product Features

L.A.S. is a transformative banking management application, designed to revolutionize the conventional banking experience. This product features at its core straightforward account management, efficient transactions, and a streamlined loan application, credit card application, and management process. Customers can easily apply for loans, credit cards, and track their application status, while tellers/bankers are empowered with a specialized dashboard for credit card approvals and customer interaction. Managers are equipped with a specialized dashboard for loan approvals and employee management. Prioritizing user experience, the system boasts an intuitive interface, personalized dashboards, and quick action shortcuts. Security is taken seriously with features like multi-factor authentication, and real-time fraud detection. Providing a comprehensive, user-centric platform for both customers and bank employees.

2.3. User Class and Characteristics

Our application is intended for three specific users: clients of a specific bank, employees of said bank, and managers of those employees. Clients aren't expected to have any computing knowledge in order to use our service, so long as they are able to read and input which option they would like to use. Employees will, however, be expected to use their own discretion when approving/denying credit lines, as well as opening/closing accounts of their clients. Managers will be in a position in which they are expected to know whether or not to approve specific loans as well as access extensive user data and interpret it for the sake of their bank. They will also be responsible for their employee's salaries, having direct access to this information.

2.4. Operating Environment

The application is designed to operate on the web across many different devices. You do need wifi to access the web application.

2.5. Constraints

Efficiency/data storage on scale. Large amounts of the algorithms may be constrained based on the application.

2.6. Assumptions and Dependencies

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within IntelliJ. This application will use Google Map API and Exchange Rates API, allowing us to connect and use data, etc.

3. Functional Requirements

3.1. Primary

Customer:

- Login/Logout
- Check balance
- Deposit/Withdraw
- Apply for a credit card
- Apply for a loan
- Transfer between accounts

Banker:

- Login/Logout
- Check balance
- Transfer between accounts
- Approve/deny credit application
- Open/close account

Manager:

- Login/Logout
- Access database
- Access employee information
- Approve/deny loan application

3.2. Secondary

- Password protection for information that is only accessible to employees and managers.
- Authorization scheme so that customers can only alter and see their banking information.

4. Technical Requirements

4.1. Operating System and Compatibility

The application will be compatible with any operating system that is able to view and interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

The user interface will be as straightforward as it needs to be, with simply named functions such as "Deposit", "Withdraw", etc. An assortment of simple commands will be at each different type of user's disposal in order to produce an easy and efficient banking experience.

4.2.2. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

It must be able to connect to the internet as well as the local database on phpMyAdmin.

The communication protocol, HTTP, must be able to connect to the Google Map API and return the current date and time, as well as the Exchange Rate API to return accurate currency exchange rates.

4.2.4. Software Interfaces

We will use React and Spring Boot ThymeLeaf to help build the front-end, as well as JPA for the back-end database functionality. We will also use Spring Boot with Java to connect the front-end to the back-end.

5. Non-Functional Requirements

5.1. Performance Requirements

- NFR0(R): The novice user will be able to login/logout within 5 minutes.
- NFR1(R): The expert user will be able to create and print a ticket in less than 1 minute.
- **Transactions/Deposit:**
- NFR2(R): The novice user should be able to initiate and complete a fund transfer in less than 7 minutes.
- NFR3(R): An expert user should be able to perform the same fund transfer in less than 2 minutes.
- NFR4(R): All users should be able to view their transaction history within 30 minutes of the request.
- **Loan/Credit Card Application & Management:**
- NFR5(R): A novice user should be able to fill out and submit a loan/credit card application in less than 30 minutes.
- NFR6(R): An expert user should be able to do the same in less than 15 minutes.
- NFR7(R): Teller/Banker should be able to review and process credit card applications in 24 hours
- NFR8(R): Manager should be able to review and process loan applications in 24 hours upon teller/banker's request.

5.2. Safety Requirements

The system will hold a lot of private information including, but not limited to, a user's financial information, credit card information, a database of loan information, employee salaries, etc. The security of our system will be of high importance to ensure the well-being and peace of mind of our investors, as well as their clients.

5.3. Security Requirements

Protecting our customers' privacy and security is L.A.S Banking System's utmost priority. Personal and private information will never be disclosed without a justified reason.

5.4. Software Quality Attributes

5.4.1. Availability

Host by 24/7 on L.A.S Banking System application. Maintenance may be applied and will be announced one week ahead.

5.4.2. Correctness

Our web/application will adhere strictly to the provided use cases unless future updates modify or expand upon them.

5.4.3. Maintainability

There will be consistent updates to help repair any issues that arise from our web/application as time passes.

5.4.4. Reusability

Servers are run by L.A.S Banking System application. As long as we have a business, the service will continue.

5.4.5. Portability

Runs on any web browser and apps.

5.5. Process Requirements

5.5.1. Development Process Used

Currently using the data structures/architecture, then building out the U.I. by using the HTML.

5.5.2. Time Constraints

8-16/11-29, all the time we have for this course.

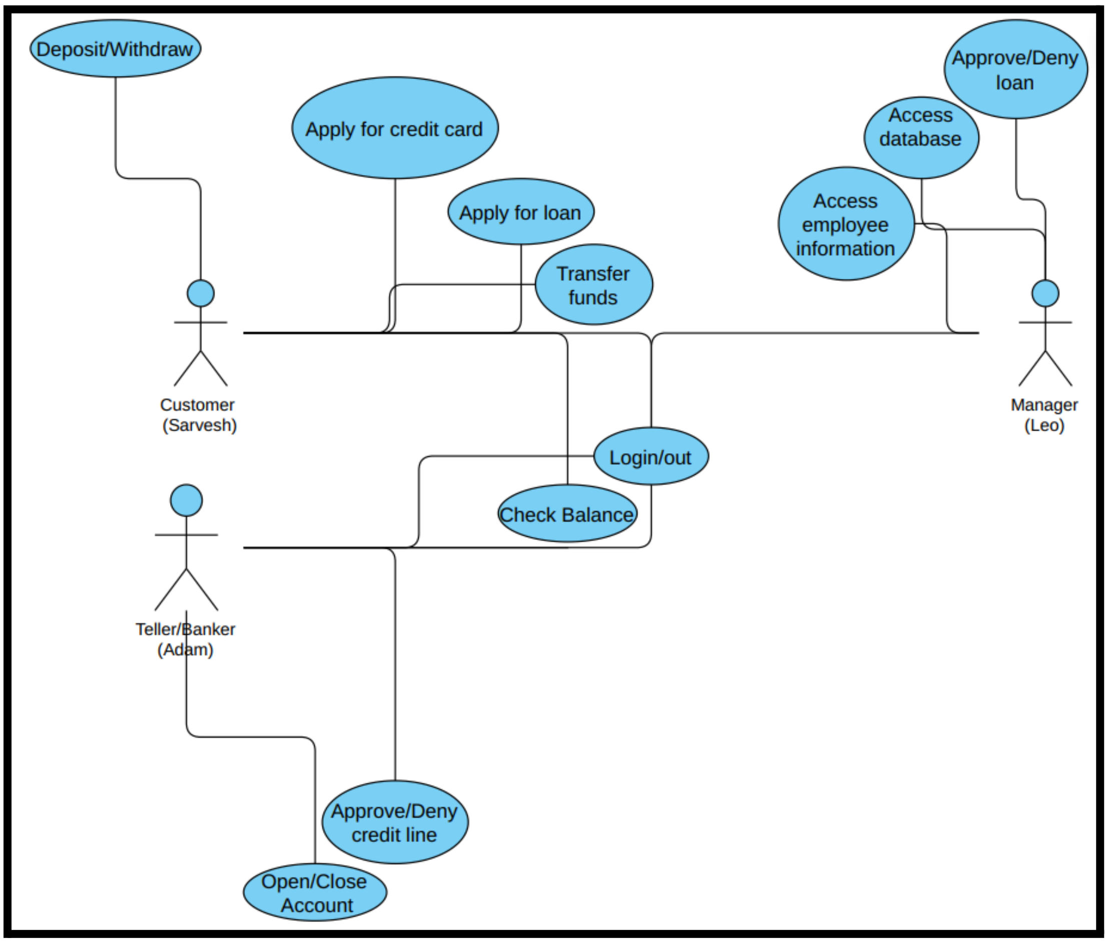
5.5.3. Cost and Delivery Date

Currently there are no costs for development and will be delivered by the end of the course.

5.6. Other Requirements

TBD

5.7. L.A.S. Banking System Use-Case Model Diagram



Use-Case Model Descriptions

5.7.1. Actor: Customer (Sarvesh)

- **Login/Logout**: Customers are required to log in and logout using their authorized accounts.
- **Check Balance**: Customers are able to check their balance using the interface provided by the web/app.
- **Deposit/Withdraw:** The customer is able to withdraw/deposit a certain amount of money. Need to talk with the banker to withdraw/deposit past the threshold.
- **Applying for a Credit Card:** The customer is able to apply for a credit card using the web or app and will be able to be picked up or shipped if approved.
- **Apply for a loan:** The customer is able to apply for a loan that gets checked by the banker later to be approved/declined of the requested loan by the customer.
- **Transfer funds:** The customer is able to transfer funds to other users using the "Transfer Funds" option in the interface.

5.7.2. Actor: Teller/Banker (Adam)

- **Login/Logout**: Bankers are required to login and logout using their authorized accounts.
- **Check Balance**: Bankers will be able to view the balance of the clients of their bank.
- **Approve/Deny Credit Line:** Bankers will be able to approve or deny applications for credit cards.
- **Open/Close Accounts:** Bankers will be able to open new accounts for their customers, as well as close existing accounts.

5.7.3. Actor: Manager (Leo)

- **Login/Logout:** The manager has their own login and logout username and password with their authorized accounts.
- **Access Employee Information**: The manager can view, modify, or update information related to bank employees for management.
- **Access Database:** The manager has the ability to access the main database to retrieve or analyze data, and customer information to ensure the smooth functioning of the bank.
- **Approve/Deny loan:** In special cases or for high-value loans, the manager may need to intervene and make the final decision on loan approval or denial based on the cumulative reason.

5.8. Use-Case Model Scenarios

5.8.1. Actor: Customer (Sarvesh)

- **Use-Case Name**: Login/Logout
    - **Initial Assumption**: Login and logout of the web/app
    - **Normal**: This is to help our customers have security over their account by having a username and password that helps them get in and out of their account.
    - **What Can Go Wrong:** Giving other users your account information can lead them to have access to your account details. They can also type in their incorrect username or password multiple times within a 10-minute period, which may cause your account to be locked for a certain amount of time.
    - **Other Activities**: Having a forgot username and password option to email you reminding you of your password using security questions or resetting your username/password.
    - **System State on Completion**: Customers are able to login/logout. This will lead them to the dashboard where all the options are available.
- **Use-Case Name**: Check Balance
    - **Initial Assumption**: Customers check money in their account
    - **Normal**: The customer is able to look into their account to check their money.
    - **What Can Go Wrong**: There might be some delay in the time you get your money from other users as it might take time to process and submit the payment.
    - **Other Activities**: There is also an option to convert the current balance into different currencies to see what you will have when traveling to other places.
    - **System State on Completion**: Customers can check their balance here and can see other options that will be available in the dashboard down below.
- **Use-Case Name**: Deposit/Withdraw
    - **Initial Assumption**: Customer is able to withdraw and deposit their money
    - **Normal**: Customers are able to withdraw an amount of 5k every day and can deposit an amount of 50k every week using the web/app.
    - **What Can Go Wrong**: Customers accidentally deposit their money to a different account and not being able to access where that money has been transferred to. Withdrawing an amount of money multiple times within the hour or a large amount of money surpassing the time limit threshold.
    - **Other Activities**: There will be other information regarding security info for transferring money, depositing, and withdrawing. There will be a system where you are able to increase the amount you can withdraw/deposit when you reach the deposit limit for 3 consecutive weeks.

    - **System State on Completion**: Customers can withdraw/deposit money in this option. There will be a dashboard separately for deposit/withdrawal options that will be shown.
- **Use-Case Name**: Applying for a Credit Card
    - **Initial Assumption**: Customers can apply for a credit card. There is an option to see your credit buildup.
    - **Normal**: This option allows the user to apply for a credit card and be able to get information on how a credit card is used and how much credit has been built up.
    - **What Can Go Wrong**: Their credit score has a bad impact on their previous purchases. Customers have their credit card application being declined.
    - **Other Activities**: They are able to check their credit score and credit card information and see transactions being made and how they affect their credit score with each purchase.
    - **System State on Completion**: Customers can apply for a credit card. There is a dashboard that shows your credit score and allows you to apply for a credit card.
- **Use-Case Name**: Apply for a Loan
    - **Initial Assumption**: Customers can apply for a loan to get extra money. This will also accumulate interest depending on how much you request. There will be a limit on when people are eligible to apply for a loan. Applying for a high loan will need to be reviewed by a manager.
    - **Normal**: This allows customers to apply for a loan when they are in need of money.
    - **What Can Go Wrong**: Customers might have their request for a loan declined due to not being eligible for a loan.
    - **Other Activities**: Customers can check your history on requesting loans. They can also look when they are able to apply for a loan by seeing the status of loan applications.
    - **System State on Completion**: They are able to see their loan status and previous history of loans.
- **Use-Case Name**: Transfer funds
    - **Initial Assumption**: Able to transfer money to other users and be able to receive money from other users.
    - **Normal**: Allows customers to be able to transfer/receive money from other users. Can check previous transferred/received money.
    - **What Can Go Wrong**: Customers/users transferring money to an unknown person and have to see if it is possible to retrieve their money.
    - **Other Activities**: Can see their previous transfer/receive money. Able to look at which user sent money to you by named users.
    - **System State on Completion**: Customers can transfer/withdraw money. Look at previous transferred/received money. See who transferred/received the money.

5.8.2. Actor: Teller/Banker (Adam)

- **Use-Case Name**: Login/Logout
    - **Initial Assumption**: The banker has a registered account to log into the system that is saved into the database.
    - **Normal**: The banker will enter their username and password into the program.
    - **What Can Go Wrong**: The banker doesn't remember their login credentials.
    - **Other Activities**: The banker can request to reset their password.

- **System State on Completion**: The banker is able to successfully login, allowing them to assist their customer's needs, and is then able to successfully logout.

- **Use-Case Name**: Check Customer Balance
  - **Initial Assumption**: The teller is able to check their customer’s account balance.
  - **Normal**: The teller will gather the customer's information and access a database to see the balance of said customer’s checking and/or savings account.
  - **What Can Go Wrong**: The client that the teller is assisting doesn’t have an account. The teller should have the ability to create an account for their client.
  - **Other Activities**: If requested, the teller can transfer funds from a client’s checking account to their savings account & vice versa.
  - **System State on Completion**: The teller has successfully accessed and can view a client’s balances and transfer funds from account to account.

- **Use-Case Name**: Approve/Deny credit line
  - **Initial Assumption**: The teller is able to access the request for credit lines from various clients.
  - **Normal**: The teller will access a credit card application. The teller will then approve or deny the request.
  - **What Can Go Wrong**: The teller is unable to determine their decision on applications without consulting their administrators first(I.E., they are having a hard time making a good financial decision in regards to the bank's overall profit of such a credit line to exist).
  - **Other Activities**: The teller is able to review the current status of a client's account in the system, allowing them to help the decision process.
  - **System State on Completion**: The teller has the ability to review credit line applications. They can then view information regarding the person who applied for said credit line, and they are able to approve or deny these requests.
- **Use-Case Name**: Open/Close accounts
  - **Initial Assumption**: The banker has permission to open and close their customer's accounts.
  - **Normal**: The banker will either grant new clients an account to better the bank's profits, or they’re able to close accounts that may be negatively affecting the bank.
  - **What Can Go Wrong**: The banker accidentally closes an account. Accounts that are closed won’t be removed from the database completely, and therefore they are able to reenlist previous account holders.
  - **Other Activities**: Upon the creation of new accounts, bankers will collect various information from their clients, information that will then be saved into the database, which is accessible by those who hold managerial accounts.
  - **System State on Completion**: The banker has successfully opened/closed an account. The information that was added/ removed from the database will then be able to be reviewed by management. New account owners are then able to access their accounts, granting them the ability to deposit funds, transfer funds, etc.

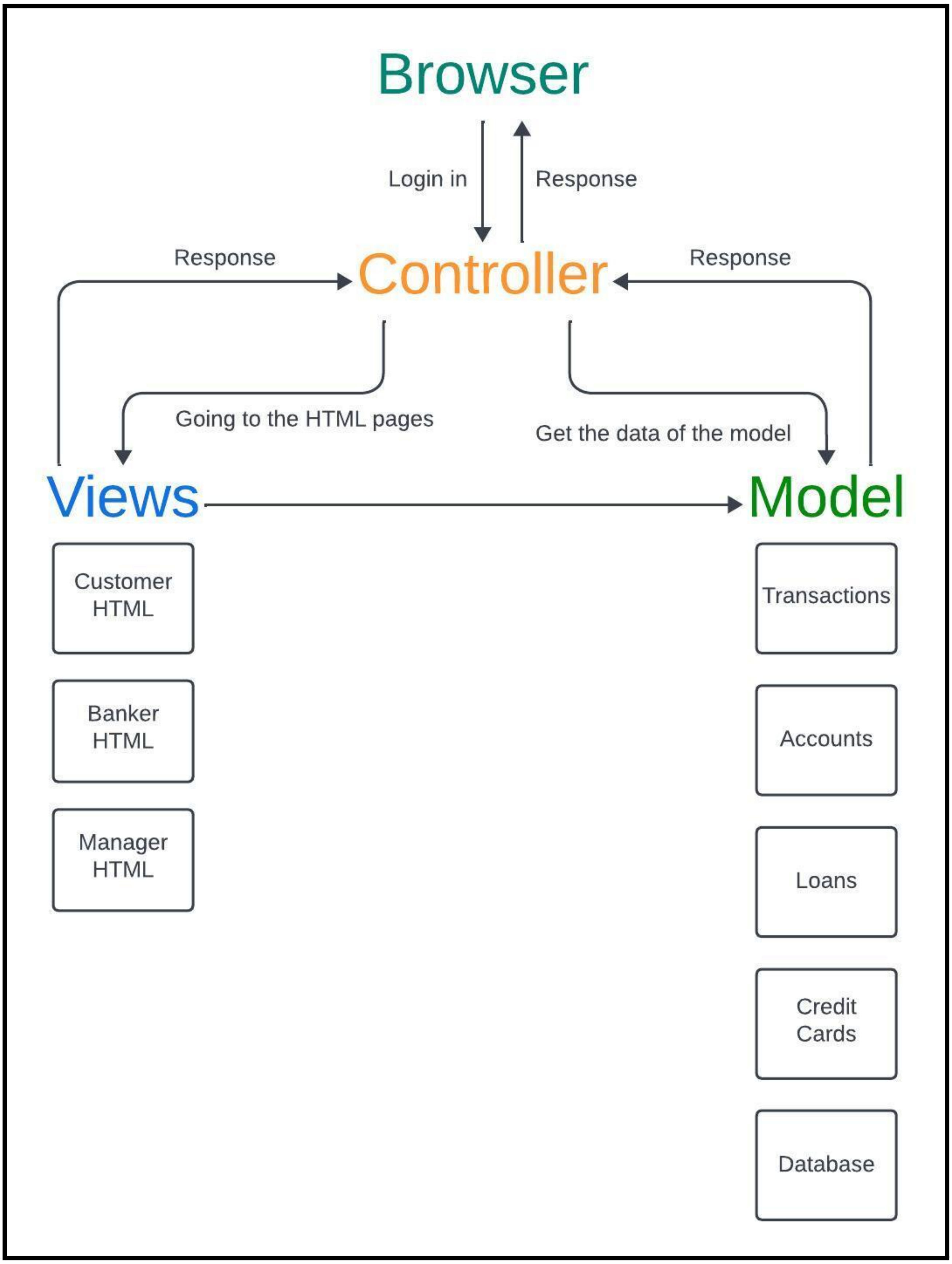
5.8.3. Actor: Manager(Leo)

- **Use-Case Name**:Login/Logout

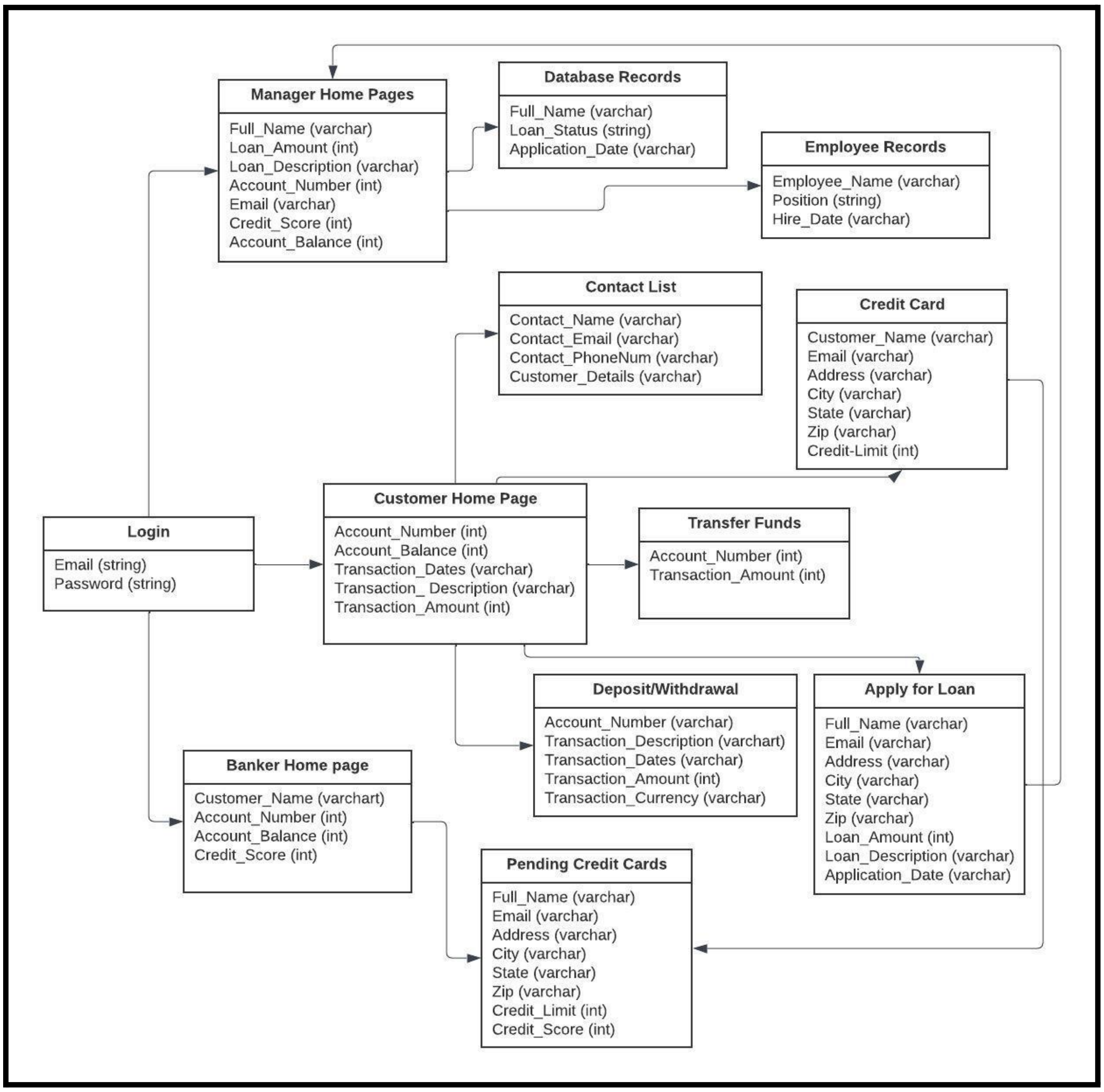
    - **Initial Assumption**: Manager is authorized to have access to all the information of the bank.
    - **Normal**: Manager logs in using a username and password to access a special dashboard with managerial functions.
    - **What Can Go Wrong**: Forgotten password or unauthorized access attempt.
    - **Other Activities**: The manager can opt to reset a forgotten password using security protocols or be notified of unauthorized login attempts.
    - **System State on Completion**: Manager successfully logs in/out, ensuring a secure session and data integrity.
- **Use-Case Name**: Access Employee Information
    - **Initial Assumption**: Only managers can view and edit employee data, administrative changes, and more.
    - **Normal**: Manager reviews, updates, or modifies employee details based on need.
    - **What Can Go Wrong**: Inadvertently altering the data leads to incorrect records. Such as deleting employee information or customer information.
    - **Other Activities**: Manager can generate performance reports or audit logs of employee actions and promotion employees.
    - **System State on Completion**: Employee database remains up-to-date and consistent.
- **Use-Case Name**: Access Database
    - **Initial Assumption**: Manager requires database access for comprehensive data analysis, system checks, and audits.
    - **Normal**: Manager accesses customer data, transaction logs, and other critical data for analytics and decision-making for high-amount of the loan or any other decision that the manager needs to make.
    - **What Can Go Wrong**: Accidental deletion or modification of crucial data; security breach if accessed from insecure networks might make the customer information being disclosure.
    - **Other Activities**: Regular data backups, encryption of sensitive data, and ensuring only authorized personnel have access.
    - **System State on Completion**: Database remains intact, secure, and consistent with all necessary updates.
- **Use-Case Name**: Approve/Deny loan
    - **Initial Assumption**: High-value or risky loans need managerial approval.
    - **Normal**: Manager reviews detailed loan applications, including the applicant's financial background and the bank's risk profile, to make a decision.
    - **What Can Go Wrong**: Inaccurate assessment leading to high-risk loan approval; overlooking crucial customer data.
    - **Other Activities**: Manager can request more detailed financial reports or customer histories to make informed decisions. Such as finance statement or high credit report/score.
    - **System State on Completion**: Loan application status is updated to either approved or denied in the database and the applicant is notified accordingly.

6. Design Documents

6.1. Software Architecture

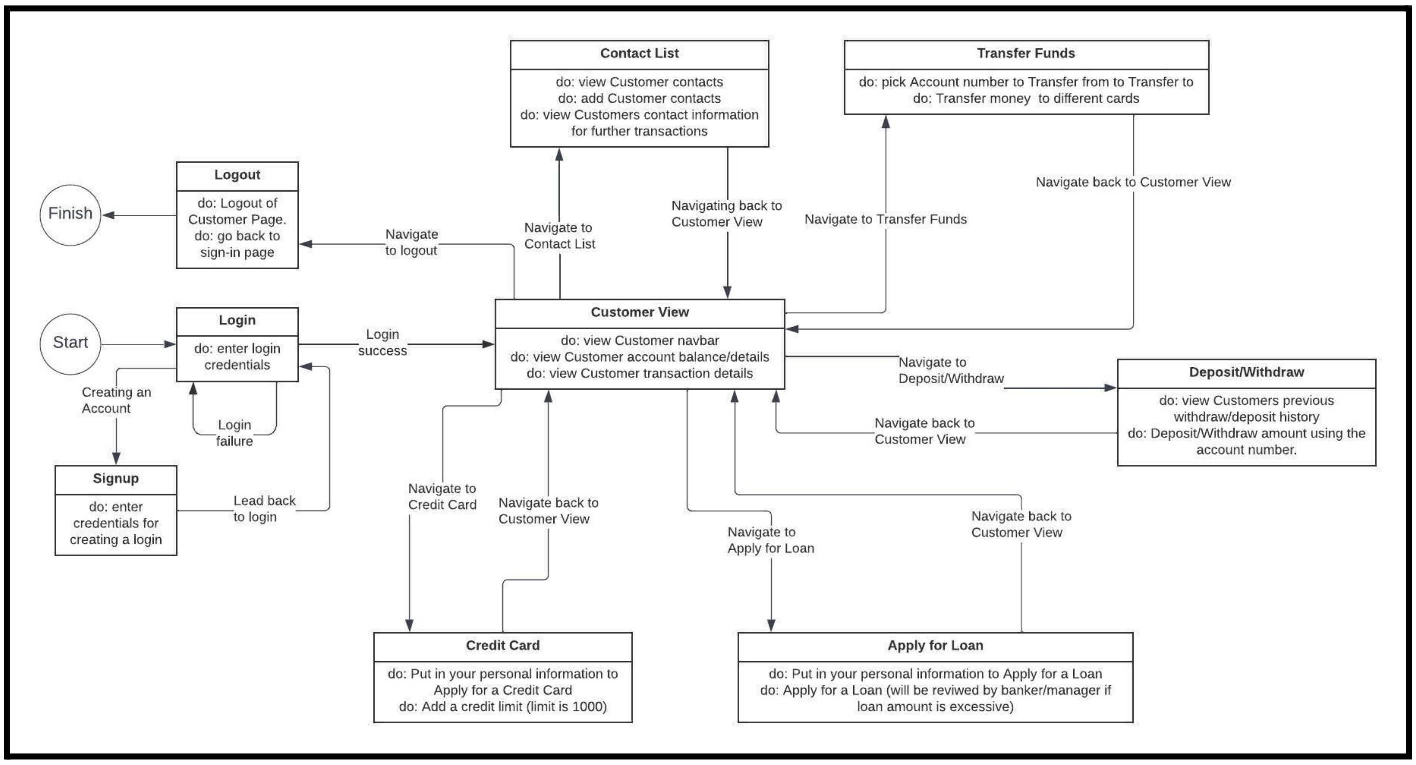


6.2. High-Level Database Schema

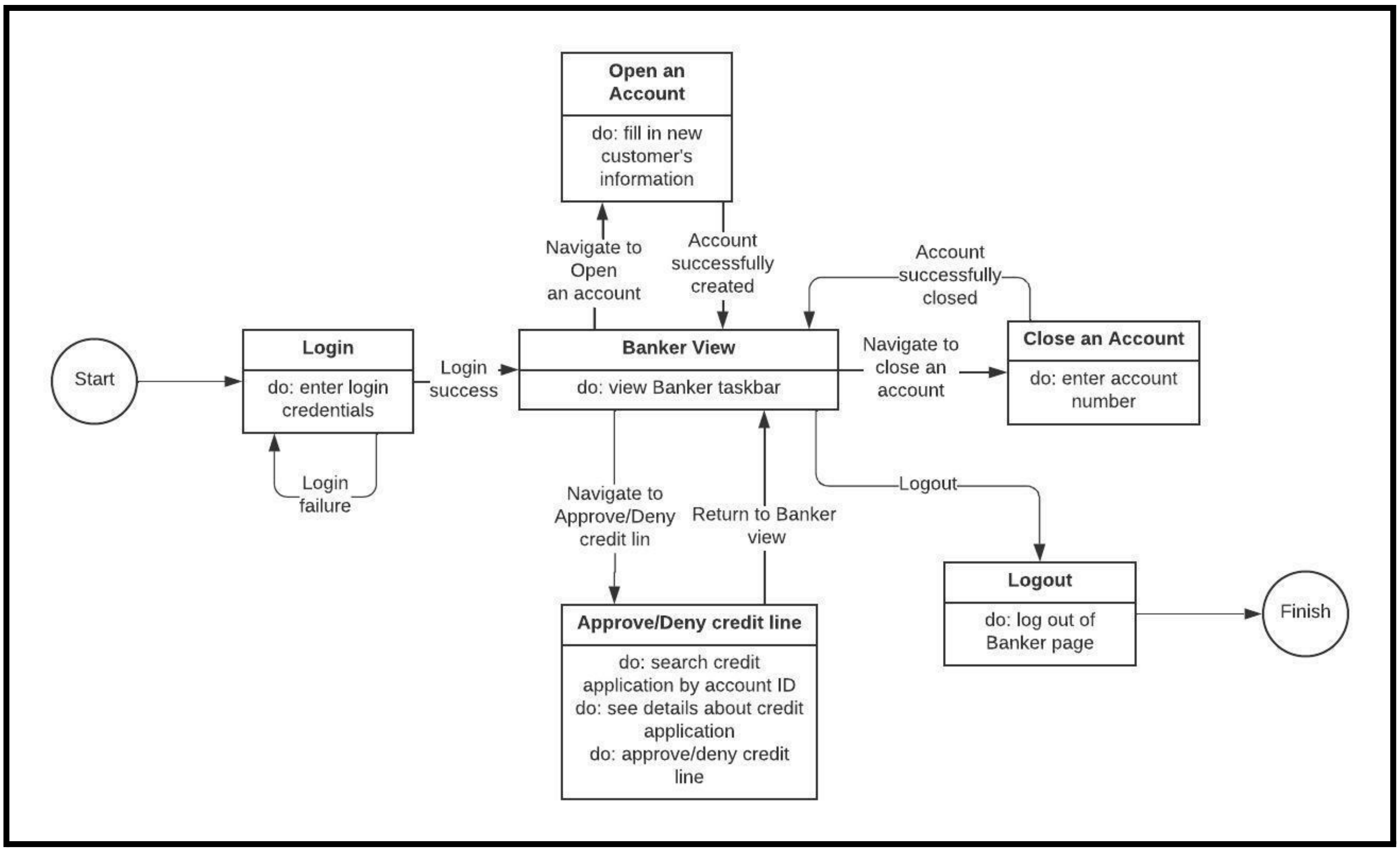


6.3. Software Design

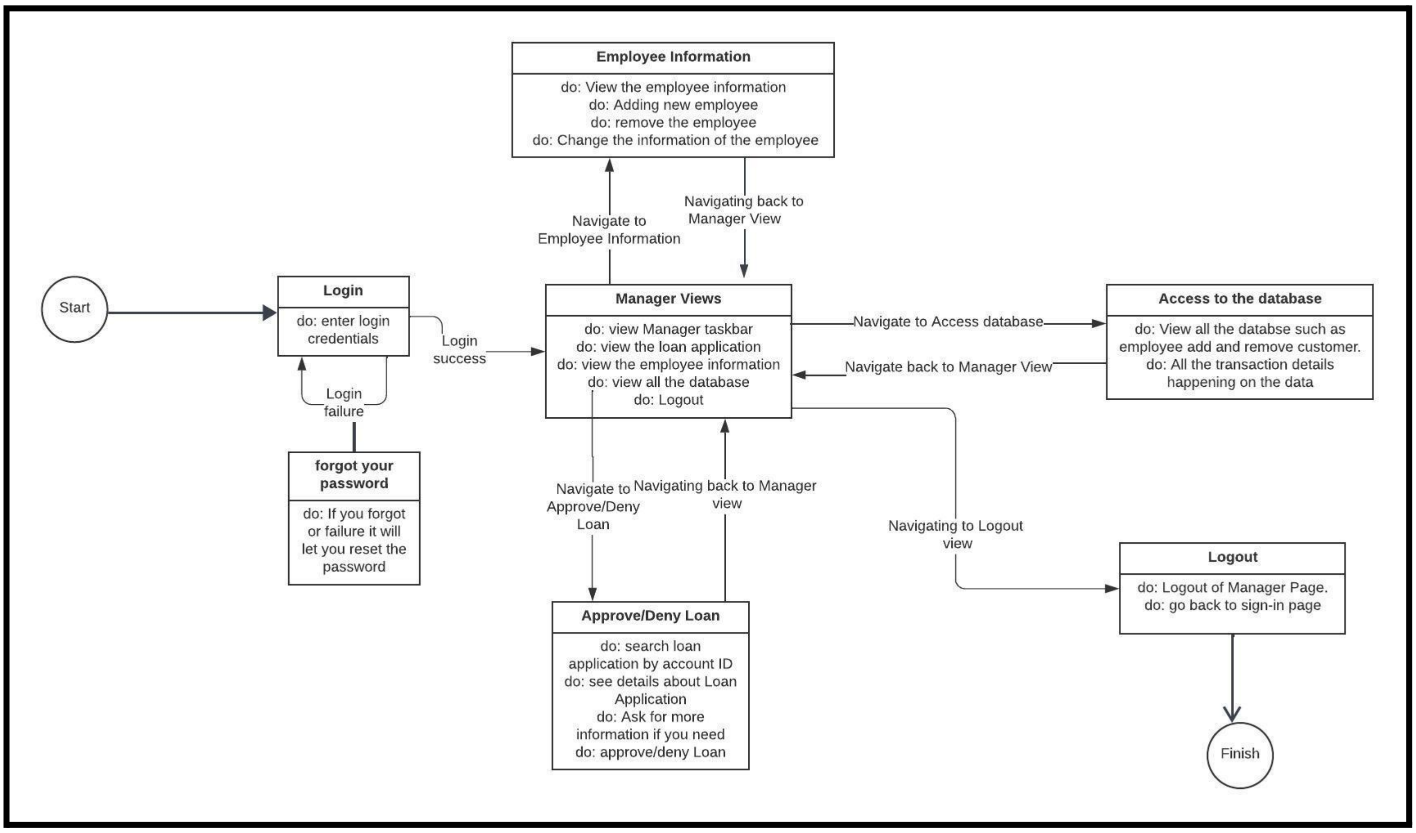
6.3.1. State Machine Diagram: Customer (Sarvesh)


Contact List
do: view Customer contacts
do: add Customer contacts
do: view Customers contact information for further transactions
Transfer Funds
do: pick Account number to Transfer from to Transfer to
do: Transfer money to different cards
Logout
do: Logout of Customer Page.
do: go back to sign-in page
Finish
Navigate to logout
Navigate to Contact List
Navigating back to Customer View
Navigate to Transfer Funds
Navigate back to Customer View
Start
Login
do: enter login credentials
Login success
Customer View
do: view Customer navbar
do: view Customer account balance/details
do: view Customer transaction details
Navigate to Deposit/Withdraw
Deposit/Withdraw
do: view Customers previous withdraw/deposit history
do: Deposit/Withdraw amount using the account number.
Navigate back to Customer View
Creating an Account
Login failure
Signup
do: enter credentials for creating a login
Lead back to login
Navigate to Credit Card
Navigate back to Customer View
Navigate to Apply for Loan
Navigate back to Customer View
Credit Card
do: Put in your personal information to Apply for a Credit Card
do: Add a credit limit (limit is 1000)
Apply for Loan
do: Put in your personal information to Apply for a Loan
do: Apply for a Loan (will be reviwed by banker/manager if loan amount is excessive)

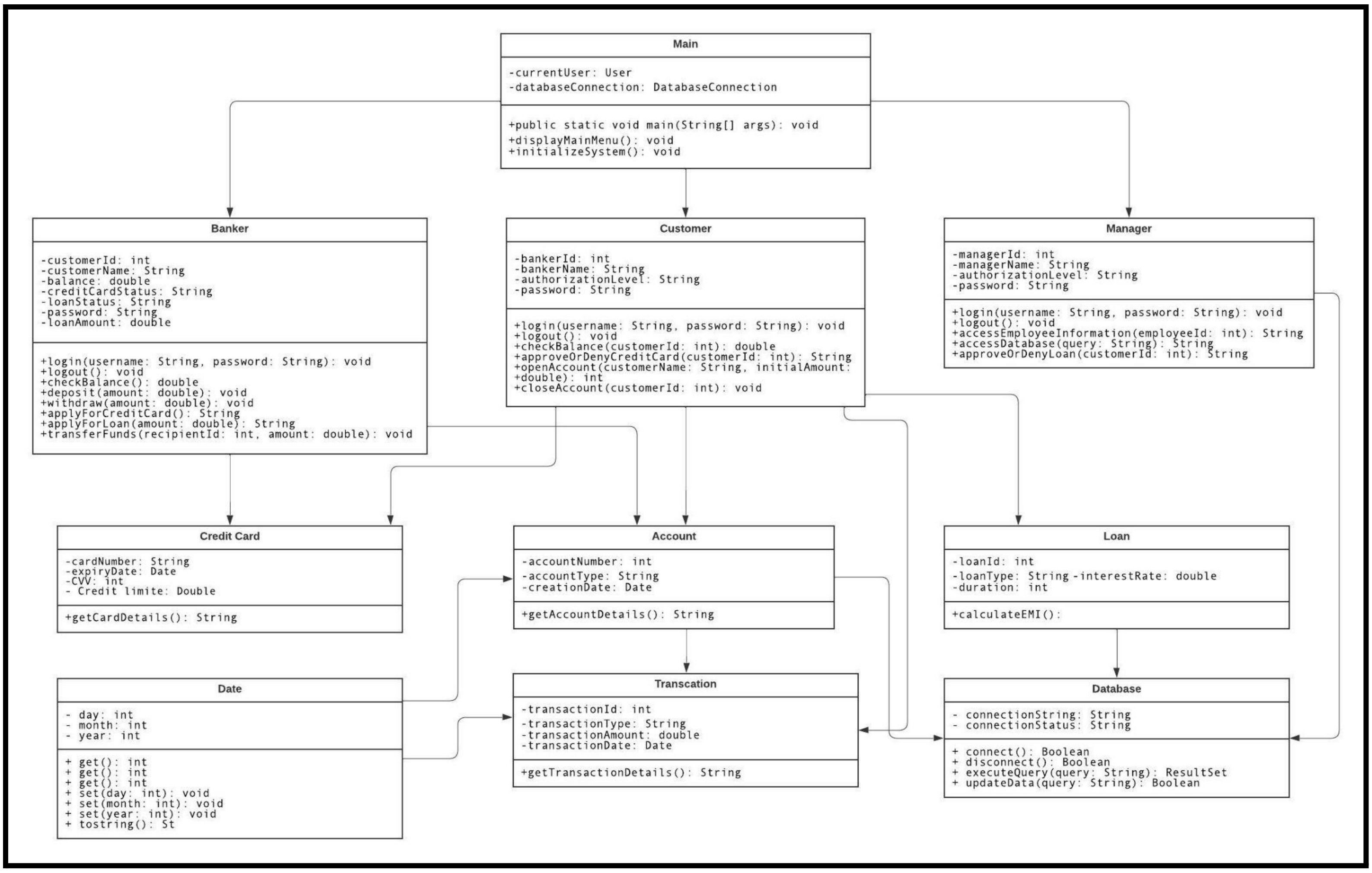
6.3.2. State Machine Diagram: Banker (Adam)


Open an Account
do: fill in new customer's information
Navigate to Open an account
Account successfully created
Account successfully closed
Start
Login
do: enter login credentials
Login success
Banker View
do: view Banker taskbar
Navigate to close an account
Close an Account
do: enter account number
Login failure
Navigate to Approve/Deny credit lin
Return to Banker view
Logout
Logout
do: log out of Banker page
Finish
Approve/Deny credit line
do: search credit application by account ID
do: see details about credit application
do: approve/deny credit line

6.3.3. State Machine Diagram: Manger (Leo)


Employee Information
do: View the employee information
do: Adding new employee
do: remove the employee
do: Change the information of the employee
Navigate to Employee Information
Navigating back to Manager View
Start
Login
do: enter login credentials
Login success
Login failure
forgot your password
do: If you forgot or failure it will let you reset the password
Manager Views
do: view Manager taskbar
do: view the loan application
do: view the employee information
do: view all the database
do: Logout
Navigate to Access database
Navigate back to Manager View
Access to the database
do: View all the databse such as employee add and remove customer.
do: All the transaction details happening on the data
Navigate to Approve/Deny Loan
Navigating back to Manager view
Navigating to Logout view
Approve/Deny Loan
do: search loan application by account ID
do: see details about Loan Application
do: Ask for more information if you need
do: approve/deny Loan
Logout
do: Logout of Manager Page.
do: go back to sign-in page
Finish

6.4. UML Class Diagram


Main
-currentUser: User
-databaseConnection: DatabaseConnection
+public static void main(String[] args): void
+displayMainMenu(): void
+initializeSystem(): void
Banker
-customerId: int
-customerName: String
-balance: double
-creditCardStatus: String
-loanStatus: String
-password: String
-loanAmount: double
+login(username: String, password: String): void
+logout(): void
+checkBalance(): double
+deposit(amount: double): void
+withdraw(amount: double): void
+applyForCreditCard(): String
+applyForLoan(amount: double): String
+transferFunds(recipientId: int, amount: double): void
Customer
-bankerId: int
-bankerName: String
-authorizationLevel: String
-password: String
+login(username: String, password: String): void
+logout(): void
+checkBalance(customerId: int): double
+approveOrDenyCreditCard(customerId: int): String
+openAccount(customerName: String, initialAmount:
+double): int
+closeAccount(customerId: int): void
Manager
-managerId: int
-managerName: String
-authorizationLevel: String
-password: String
+login(username: String, password: String): void
+logout(): void
+accessEmployeeInformation(employeeId: int): String
+accessDatabase(query: String): String
+approveOrDenyLoan(customerId: int): String
Credit Card
-cardNumber: String
-expiryDate: Date
-CVV: int
- Credit limite: Double
+getCardDetails(): String
Account
-accountNumber: int
-accountType: String
-creationDate: Date
+getAccountDetails(): String
Loan
-loanId: int
-loanType: String -interestRate: double
-duration: int
+calculateEMI():
Date
- day: int
- month: int
- year: int
+ get(): int
+ get(): int
+ get(): int
+ set(day: int): void
+ set(month: int): void
+ set(year: int): void
+ tostring(): St
Transcation
-transactionId: int
-transactionType: String
-transactionAmount: double
-transactionDate: Date
+getTransactionDetails(): String
Database
- connectionString: String
- connectionStatus: String
+ connect(): Boolean
+ disconnect(): Boolean
+ executeQuery(query: String): ResultSet
+ updateData(query: String): Boolean

7. Scenario

7.1. Brief Written Scenario with Screenshots