PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage
Bengaluru 560085

Department of Computer Science and Engineering
B. Tech. CSE - 6th Semester
Jan – May 2023

UE20CS352

OBJECT ORIENTED ANALYSIS AND DESIGN USING JAVA

Project Report

BANKING APPLICATION

Submitted by:

PES1UG20CS441: Sujan M
PES1UG20CS443: Sukruth Keshava Gowda
PES1UG20CS445: Sundeep A
PES1UG20CS460: Tanishq Chugh

Class of Prof. Priya Badrinath

TABLE OF CONTENTS

Objective: The objective of this project is to create a banking application in Java that can perform various functions like creating and managing customer accounts, performing transactions , viewing transactions and providing a secure environment for users.

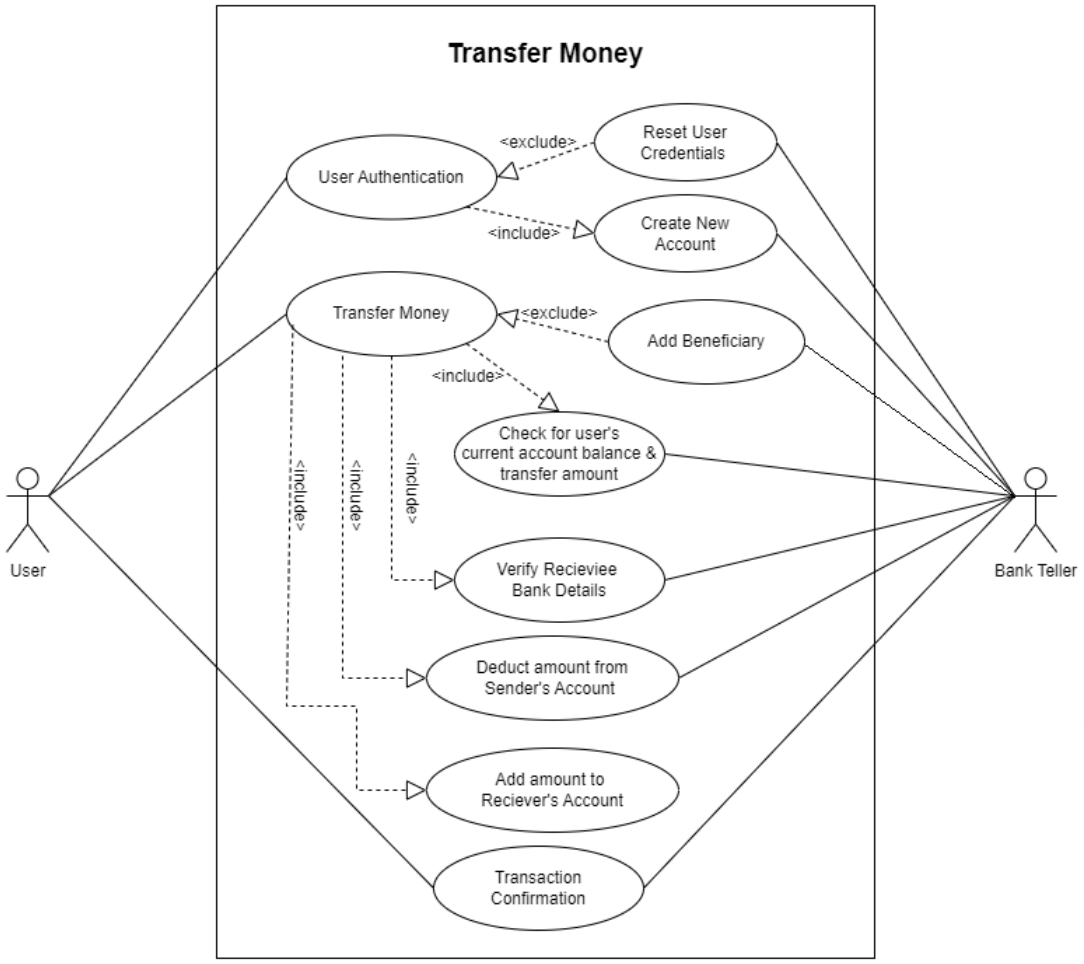
Problem Statement:
Design and develop a banking application using Java and the Spring Boot framework, with the aim of providing users with a secure and convenient way to manage their finances. The application should allow users to create accounts, view their transaction history, transfer money between accounts, apply for loans, invest in mutual funds, and make changes to their account settings. The system should be designed with security in mind, using encryption and other techniques to protect user data and prevent unauthorized access. The application should also be scalable, able to handle a large number of users and transactions without sacrificing performance or reliability.

Features:

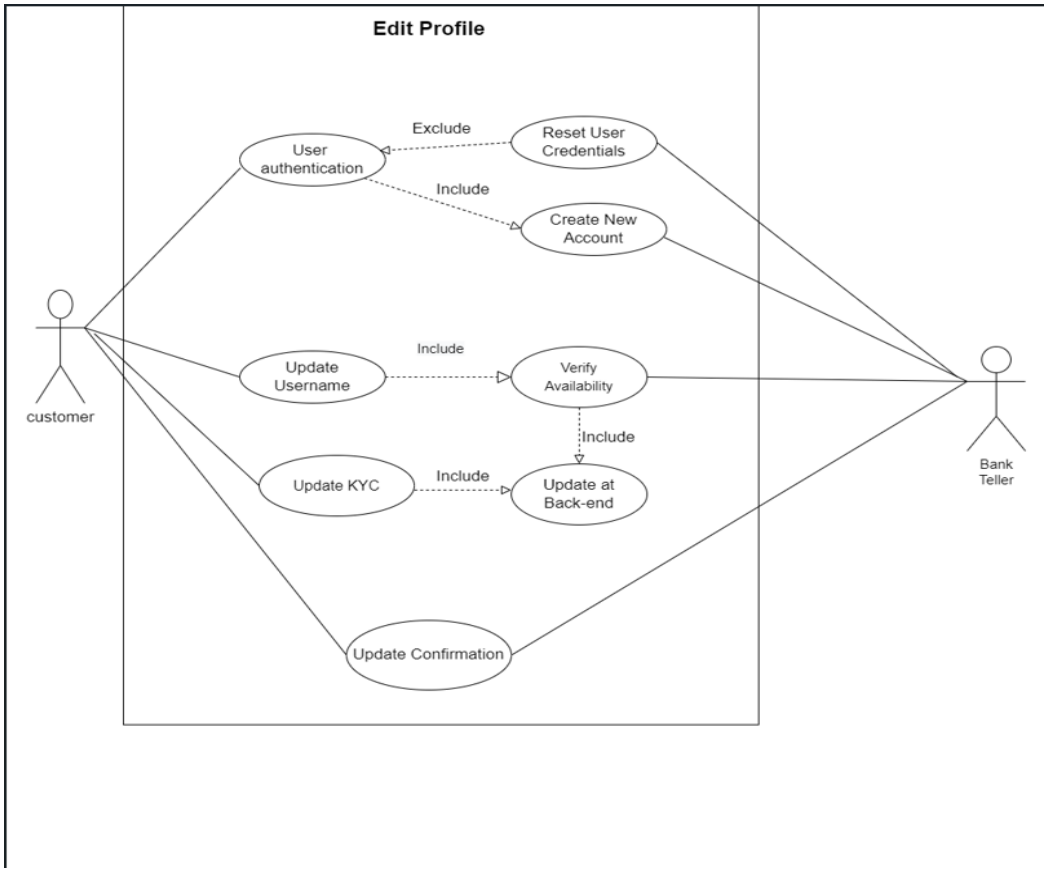
1. Transfer money - This use case involves allowing users to transfer money between their accounts or to other accounts within the bank. The user should be able to enter the recipient's account details and the amount to transfer, and the system should verify that the transaction is valid and update the relevant account balances.
2. Edit profile - This use case allows users to view and update their personal information, such as their name, address, and contact details. The system should provide a secure interface for users to make changes and validate any updates before saving them to the database.
3. Create account - This use case involves allowing users to create new accounts, such as savings, checking, or investment accounts. The system should verify that the user is eligible to open an account and create the account with the appropriate settings.
4. View customer list - This use case allows authorized users to view a list of all customers who have accounts with the bank. This feature can be useful for customer service representatives, managers, or other stakeholders who need to access customer data.
5. View transactions - This use case allows users to view a history of their transactions, including deposits, withdrawals, transfers, and other transactions. The system should provide a secure interface for users to access this information and ensure that only authorized users can view transaction history.
6. Apply for loan - This use case involves allowing users to apply for loans, such as personal loans, auto loans, or mortgages. The system should verify that the user is eligible for the loan and provide a secure interface for the user to enter their application information.
7. Login / Register – Allows user to login / register into our application.
8. Change PIN - This use case allows users to change their PIN (personal identification number), which is used to authenticate the user when accessing their account.

Use Case Diagrams:

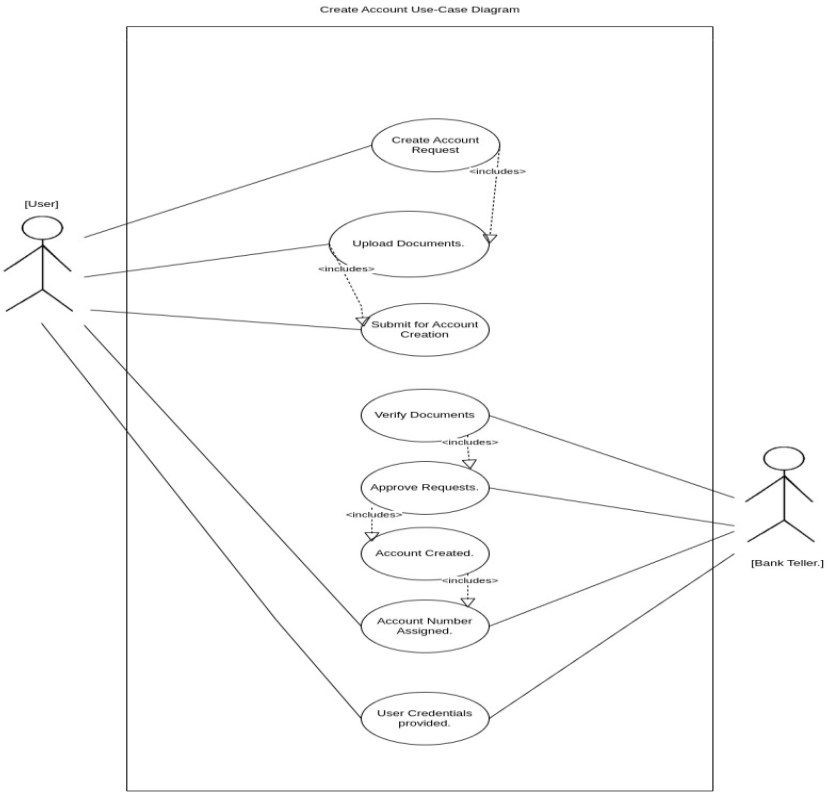
1. Transfer Money (By—Tanishq Chugh)



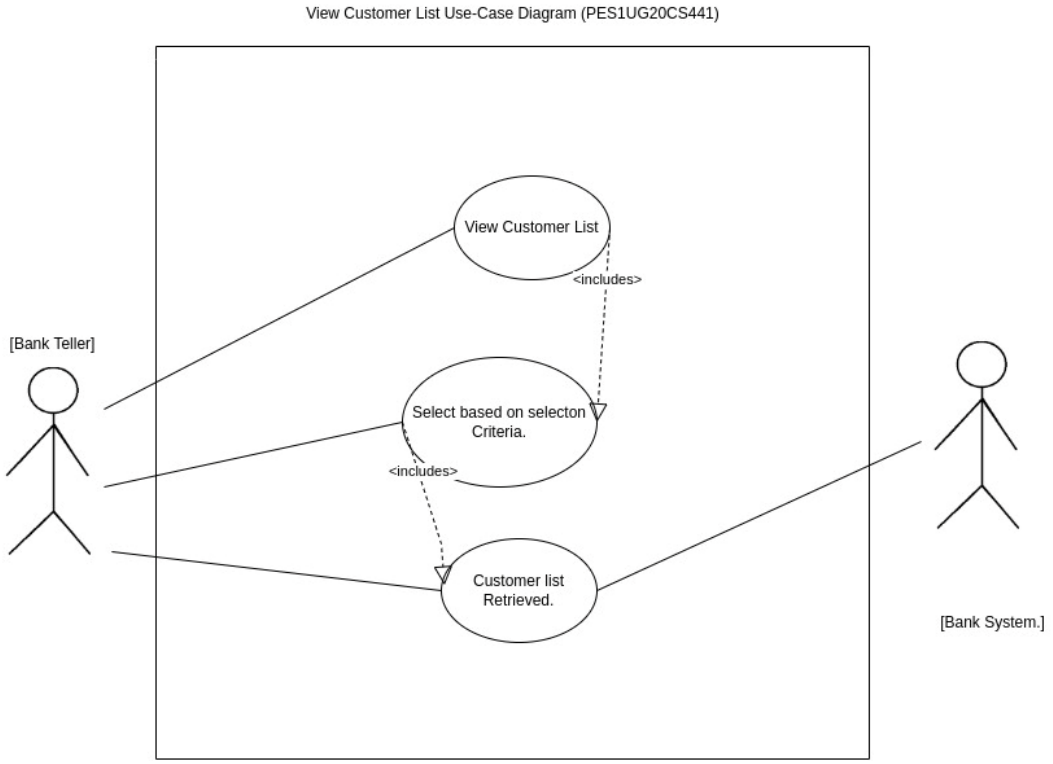
2. Edit Profile (By---Tanishq Chugh)



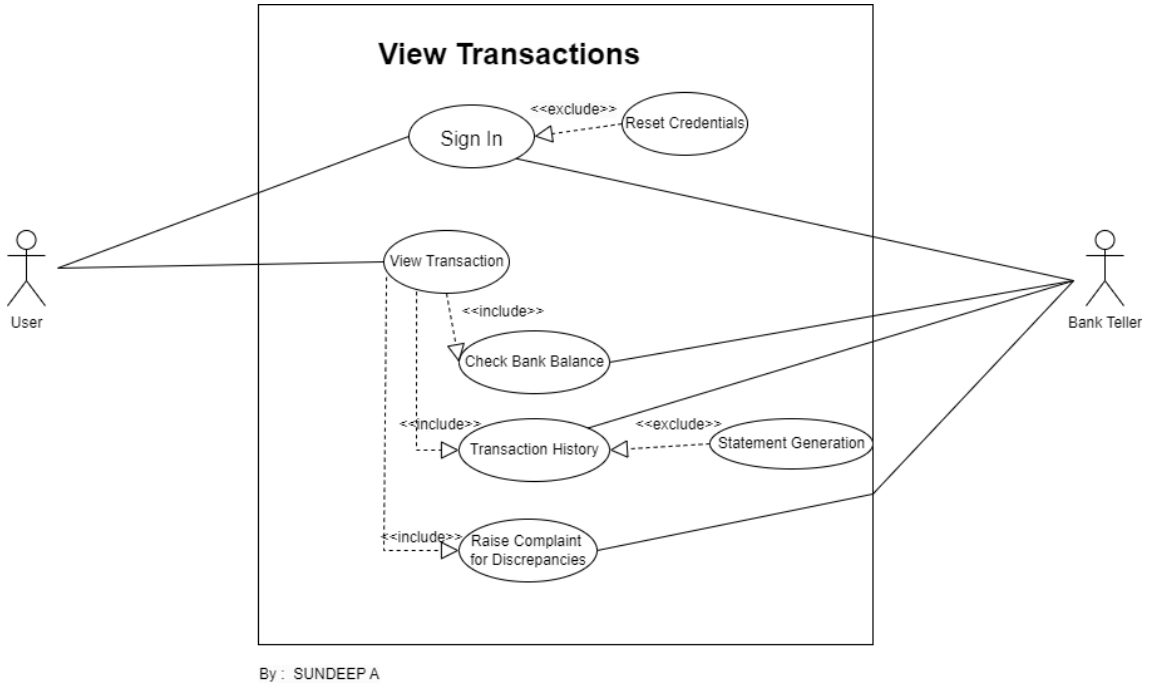
3. Create Account (By---Sujan M)



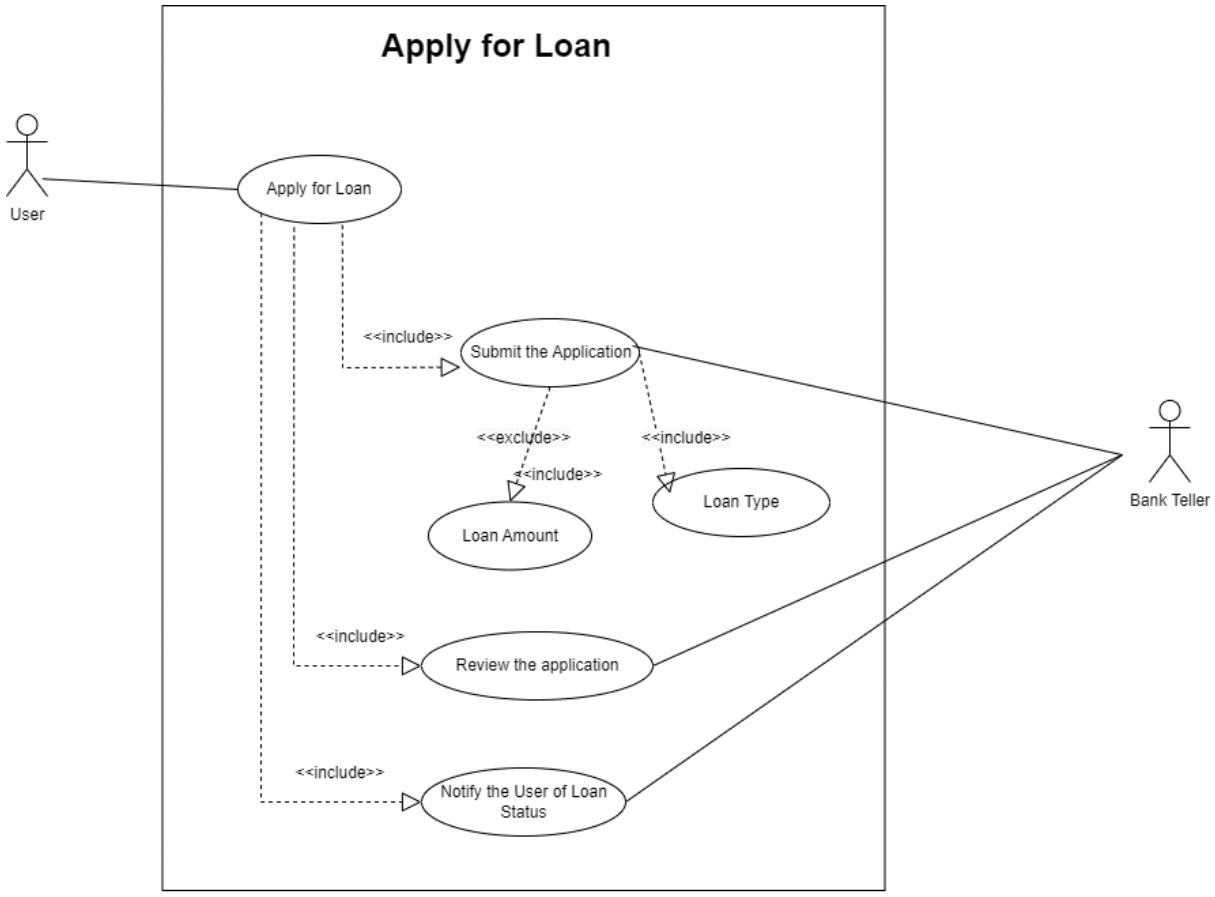
4. View Customer List (By---Sujan M)



5. View Transactions (By---Sundeep A)

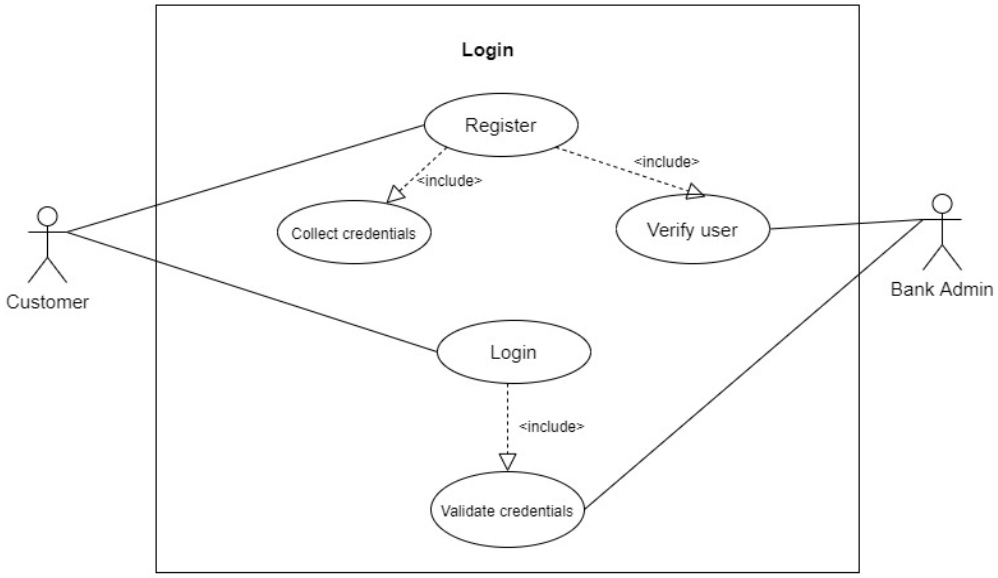


6. Apply for Loan (By---Sundeep A)

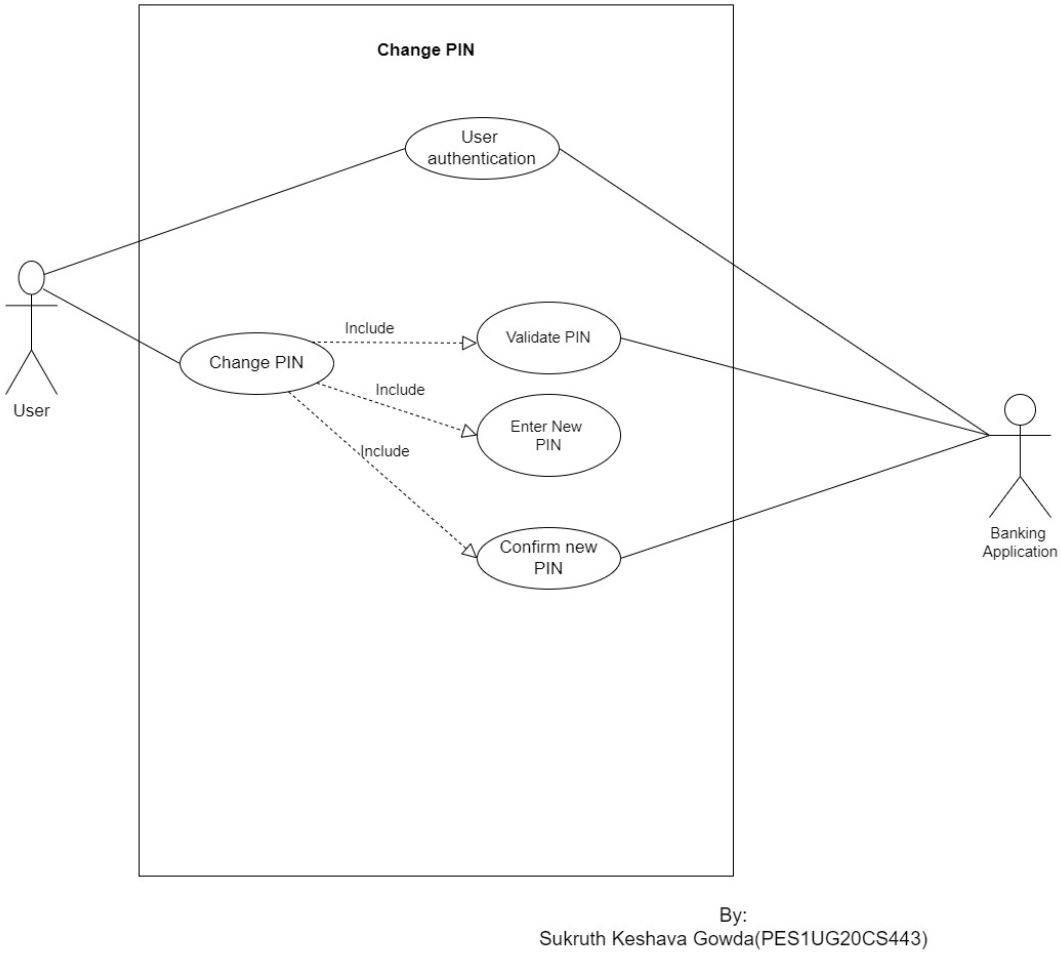


By : SUNDEEP A

7. Login / Register (Done by Sukruth Keshava Gowda)



8.Change Pin (By—Sukruth Keshava Gowda)



Use Case Description:

1. Transfer Money

Name: Transfer Amount
Summary / Overview: Transfer Amount from one Account (Sender's Account) to Another (Recievee's Account)
Actors: User , Bank-Teller
Pre-Conditions: Sender's Account must have sufficient funds
Descriptions:

- Customer identifies the accounts from which and to which funds have to be transferred.
- Enters the amount to be transferred.
- Confirms the transaction.

Exceptions:

- Cancel Transfer, Insufficient Funds, Invalid Recievee's Account Details

- Power Failure, Slow Network

Post-Conditions:

- Funds transferred and account balance updated

2. Edit Profile

Name: Edit Profile
Summary / Overview: Edit the details of user – both KYC & bank details
Actors: User , Bank-Teller
Pre-Conditions: User's Account must exist
Descriptions:

- Customer identifies the account whose details must be updated.
- Updates the required fields.
- Confirms the Updates.

Exceptions:

- Cancel Update, Invalid updates due to Duplication
- Power Failure, Slow Network

Post-Conditions:

- Details updated at the back-End.
- Update Confirmation to User.

3. Create Account

Name: Create Account
Summary / Overview: Create New Account by the User.
Actors: User , Bank-Teller
Pre-Conditions: User Must have all documents ready.
Descriptions:

- User submits the request to open a new account and uploads all the Documents.
- Bank-Teller verifies the information and creates and account.
- The user gets the account credentials.

Exceptions:

- Invaild Documents , Incomplete Details.
- Power Failure, Slow Network

Post-Conditions:

- Account number assigned and Account Credentials sent to user.

4. View Customer List

Name: View Customer List.
Summary / Overview: List all the customers.
Actors: Bank-Teller, Bank System.
Descriptions:

- Bank-Teller submits a request to view customer list.
- The Bank-Teller then selects based on selection-criteria.
- The Bank-Teller then receives the customer-list.

Exceptions:

- Invaild Selection criteria.
- Power Failure, Slow Network.

Post-Conditions:

- The Bank-Teller then receives the customer-list.

5. View Transactions

Name: View Transactions
Summary/Overview: View Transaction details for the specified account
Actor: User, Bank-Teller
Pre-conditions:

- Account must be Active
- Before the "Customer" can view their transaction history, they must be logged into their account on the "Banking Application."

Description:

- The View Transactions use case represents the process of a bank customer viewing the all the transactions done in their account using the banking application
- The process involves several steps, including verifying the customer's identity, check bank balance, view all the transactions done in the account including statement generation and can even raise complaint for any Discrepancies in the transaction details

Exceptions:

- No Transaction history available
- Incorrect login Credentials

Post-Conditions:

- Customer has successfully view all the transaction, can even generate the statement which can be used for Taxing, budgeting, etc.

6. Apply for Loan

Name: Apply for Loan
Summary/Overview: User applies for a loan and the bank can either accept or reject it.
Actor: User, Bank-Teller
Pre-conditions:

• Account must be Active
• Before the "Customer" can Apply for Loan, they must be logged into their account on the "Banking Application."

Description:

• The Apply for Loan use case represents the process of a customer applying for loan and the bank can either accept or reject the application.
• The process includes several steps, including the user submitting the loan application which includes details like the loan amount and the loan type. On submitting the application, the bank reviews the application and can either Accept or Reject the application. The bank can then notify the User about the application status.

Exceptions:
• Invalid application details
• Incorrect Login Credentials

Post-Conditions:
• If the loan application is approved, the customer receives a notification with the loan details, such as loan amount, interest rate, etc.
• If the loan application is rejected, the customer receives a notification

7. Change Pin

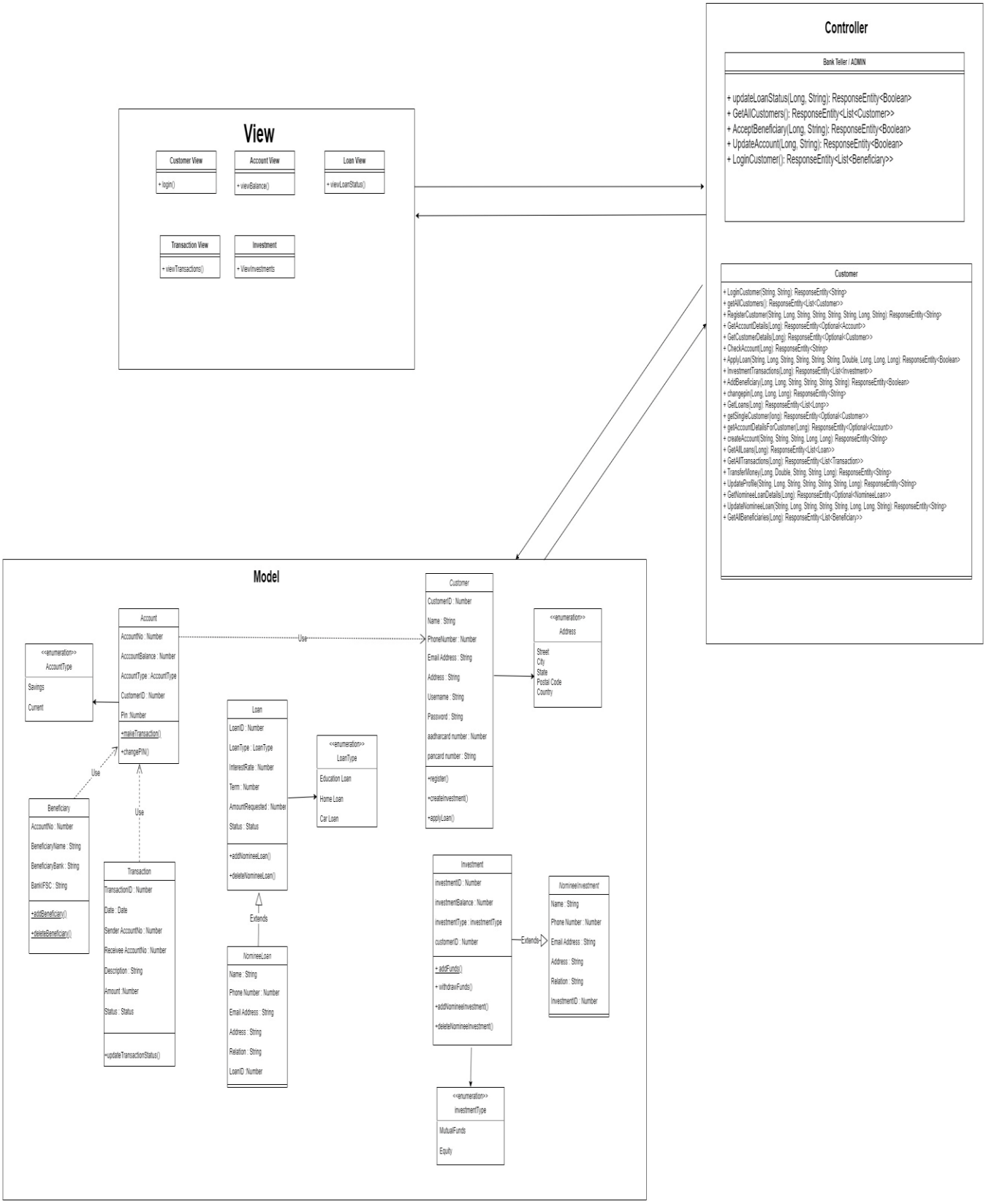
Name: Change PIN
Actors: User, Banking-Teller,Bank-Server
Preconditions: The user is logged into the banking app
Description:

- The user selects the "Change PIN" option in the banking app.
- The banking app prompts the user to enter their current PIN.
- The user enters their current PIN.
- The banking app validates the current PIN and prompts the user to enter a new PIN.
- The user enters a new PIN.
- The banking app prompts the user to confirm the new PIN.
- The user confirms the new PIN.
- The banking app sends a request to the bank server to update the user's PIN.
- The bank server updates the user's PIN.

Postconditions:

- The user's PIN has been changed and can now be used to access their account.
- The user's new PIN must meet the bank's requirements for a valid PIN .
- The user's new PIN must be different from their previous PIN.

Class diagram



ARCHITECTURE PATTERN

In Object-Oriented Programming (OOP), design architecture refers to the overall structure of an application or system. OOP design architecture defines how the various components of an application will interact with each other, and how they will be organized to achieve specific goals.

Model-View-Controller (MVC) Architecture pattern is implemented in our Banking Application.

● **Model**: The Model consists of the classes that represent the data and business logic of the application. In this project, the Model package contains several classes that represent various entities and concepts related to Banking Application, such as Customer , Account , transaction , Loan , Investment . These classes encapsulate the relevant properties and behaviors of their respective entities and provide methods to interact with them.

● **View**: The View is responsible for presenting the data to the user and handling user input. In this project, the View package is not explicitly shown, but it could contain classes that represent the user interface components of the application, such as forms, dialogs, and menus. These classes would use the Model classes to display and manipulate data, and they would also invoke Controller methods to respond to user input.

● **Controller**: The Controller acts as an intermediary between the Model and the View, handling user input, updating the Model, and notifying the View of any changes. In this project, the Controller package contains several classes that represent the controllers of various functionalities of the banking application, such as Bank_AdminController and CustomerController. These classes implement methods that respond to user input, such as updating account , creating account , applying for loans , view transactions etc. The Controller classes interact with the Model classes to perform the necessary operations and update the data, and they also notify the View of any changes through appropriate callbacks or events.

Overall, the MVC pattern provides a clear separation of concerns between the Model, View, and Controller components of the application, enabling better organization, modularity, and maintainability of the code.

DESIGN PRINCIPLES AND PATTERNS

Design principles in OOP are a set of guidelines that help developers create code that is easy to maintain, flexible, and efficient. These principles help to organize and structure code in a way that makes it easier to understand and modify, and they promote code reusability. Following are some of the design principles used in our project.

Dependency Injection (DI): The code uses the @Autowired annotation to inject dependencies, such as BeneficiaryService, LoanService, CustomerService, and AccountService, into the BankController class. This helps in reducing tight coupling between components and allows for better modularization and testability.

Single Responsibility Principle (SRP): The code seems to follow the SRP by having separate classes for each business domain, such as Beneficiary, Loan, Customer, and Account, and their corresponding services. This promotes better separation of concerns and improves the maintainability of the code.

Separation of Concerns: The code separates concerns by having separate classes for handling requests and responses (BankController) and business logic (BeneficiaryService, LoanService, CustomerService, and AccountService). This allows for better modularization and testability.

Open-Closed Principle (OCP): The code seems to follow the OCP by using interfaces to define contracts between components, such as BeneficiaryRepository and BeneficiaryService. This allows for better extensibility and modularity of the code.

Liskov Substitution Principle:
In the above code, we can see an example of LSP in the way the Beneficiary, Loan, Customer, and Account classes are used in the BankController class.
For example, if the BeneficiaryService class is subclassed to create a SpecialBeneficiaryService class, and the SpecialBeneficiaryService class is used instead of the BeneficiaryService class in the BankController class, the program should still work correctly. This is because the SpecialBeneficiaryService class is a subclass of BeneficiaryService and should be able to replace it without affecting the correctness of the program.

Facade Pattern:

By using façade pattern in the bank controller class we have tried make sure that the point of access from the bankside on our application is through only one class controller.

Factory pattern:

By using the Factory Method pattern in our application , we can simplify the process of creating new accounts and ensure that the correct account type is created based on user input. Additionally, we can easily add new account types in the future without having to modify the client code.

Command pattern:

In our project, request from the client is sent to the controller which acts as an invoker. The request data is encapsulated in an object and sent to invoker to be processed. This helps in secure connection between client and backend, enhancing the reliability of the project.

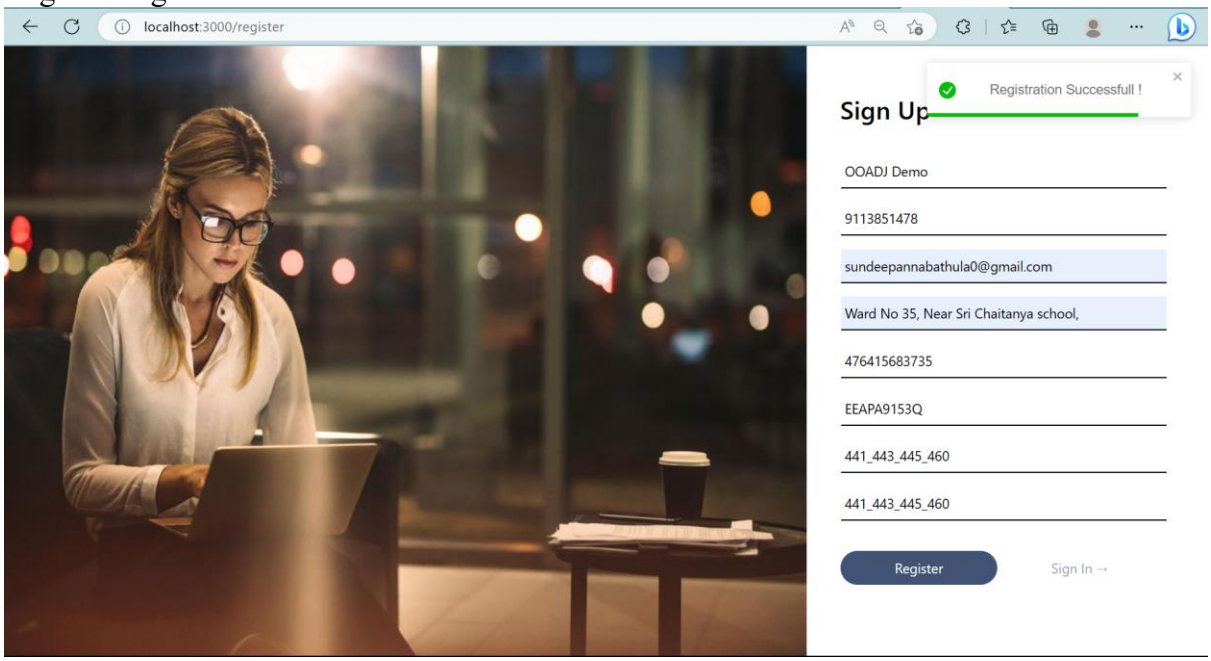
Github repository link
https://github.com/SundeepA28/OOADJ--Banking-Application

Individual contributions:

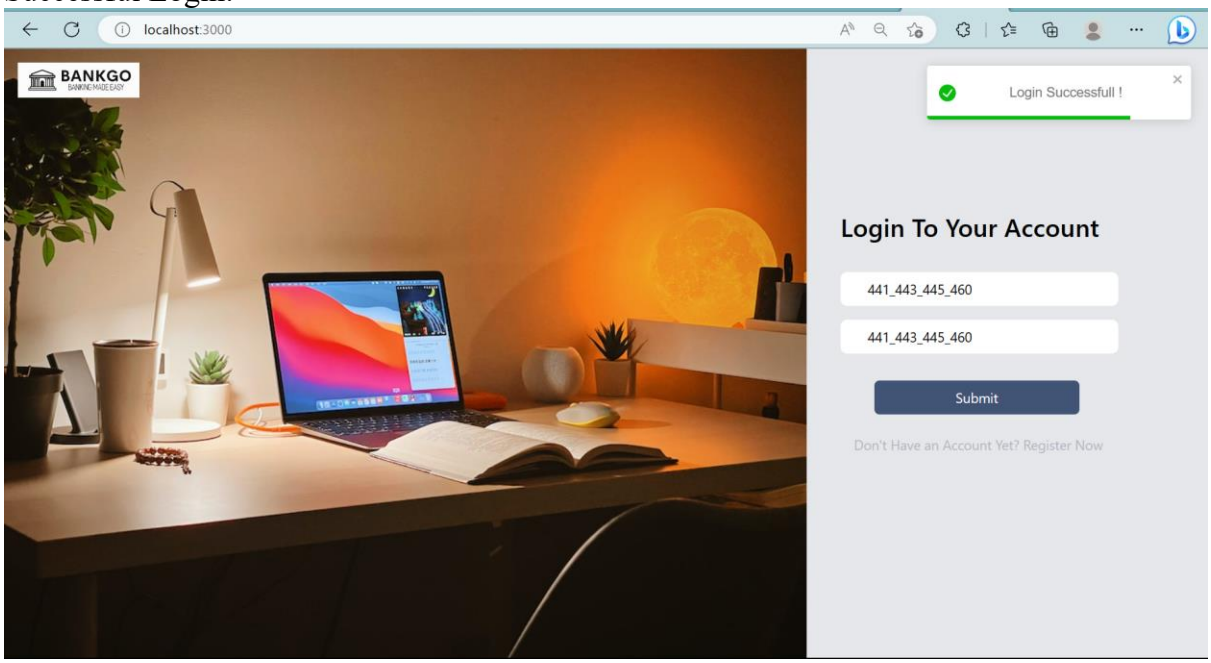
1. Sujan M – Create Account and View Customer List
2. Sukruth K – Login / Register and change pin
3. Sundeep A – View Transactions and Apply for Loan
4. Tanishq Chugh – Transfer Money and Edit profile

Screenshots of Application

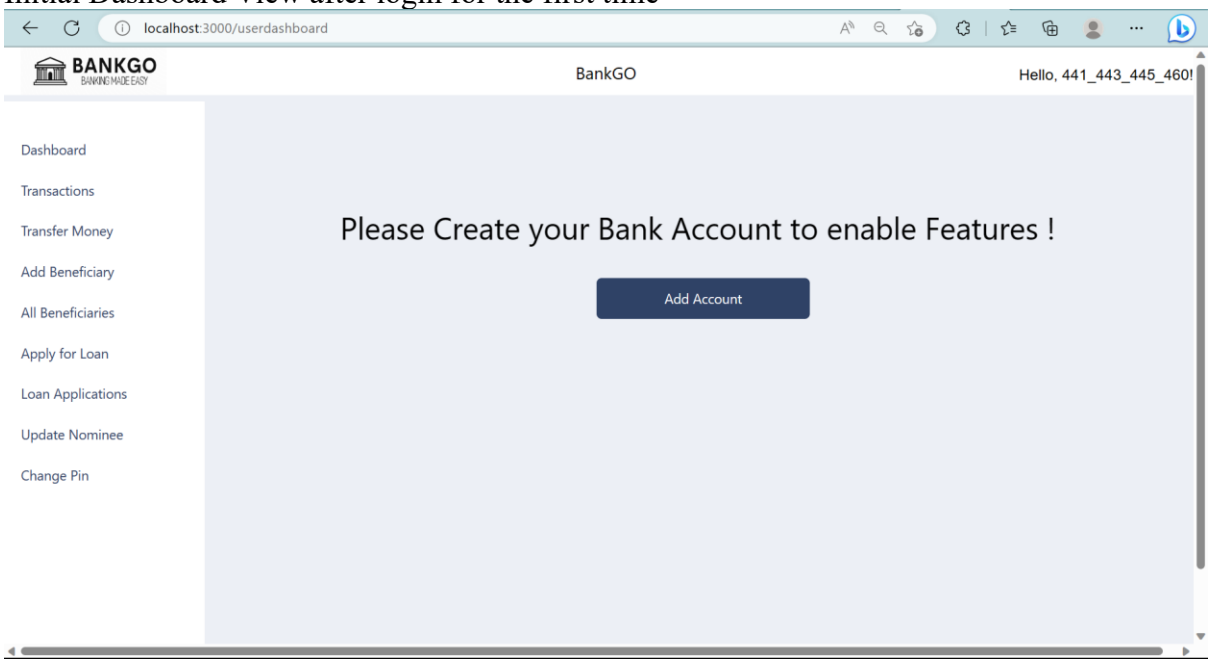
Register Page:



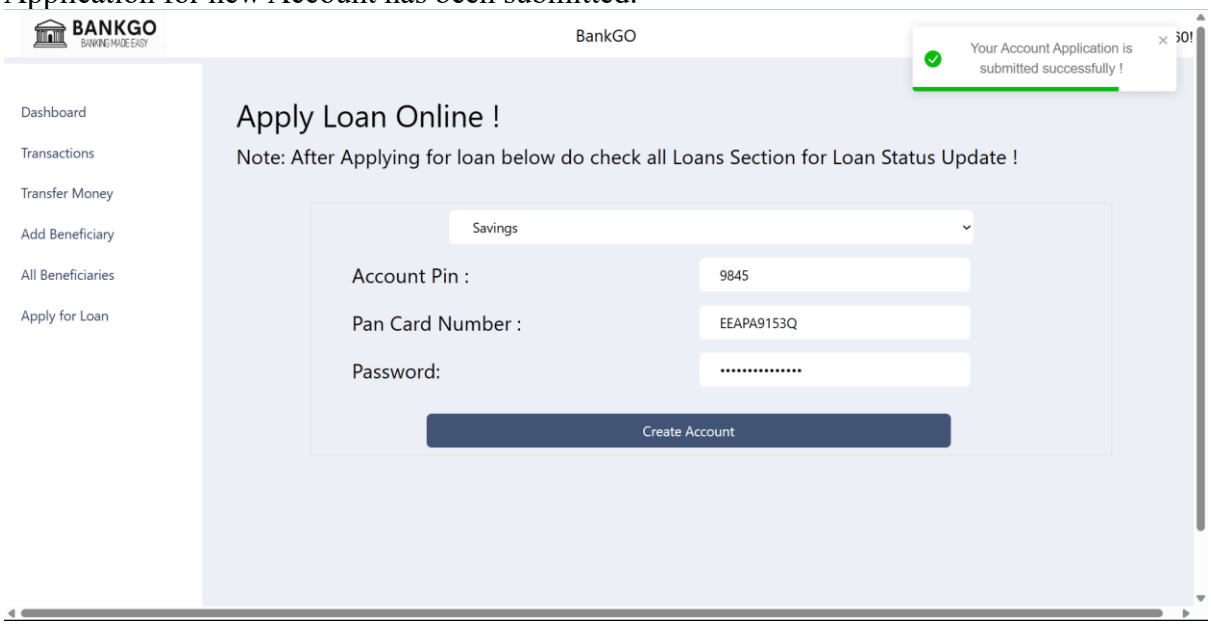
Successful Login:



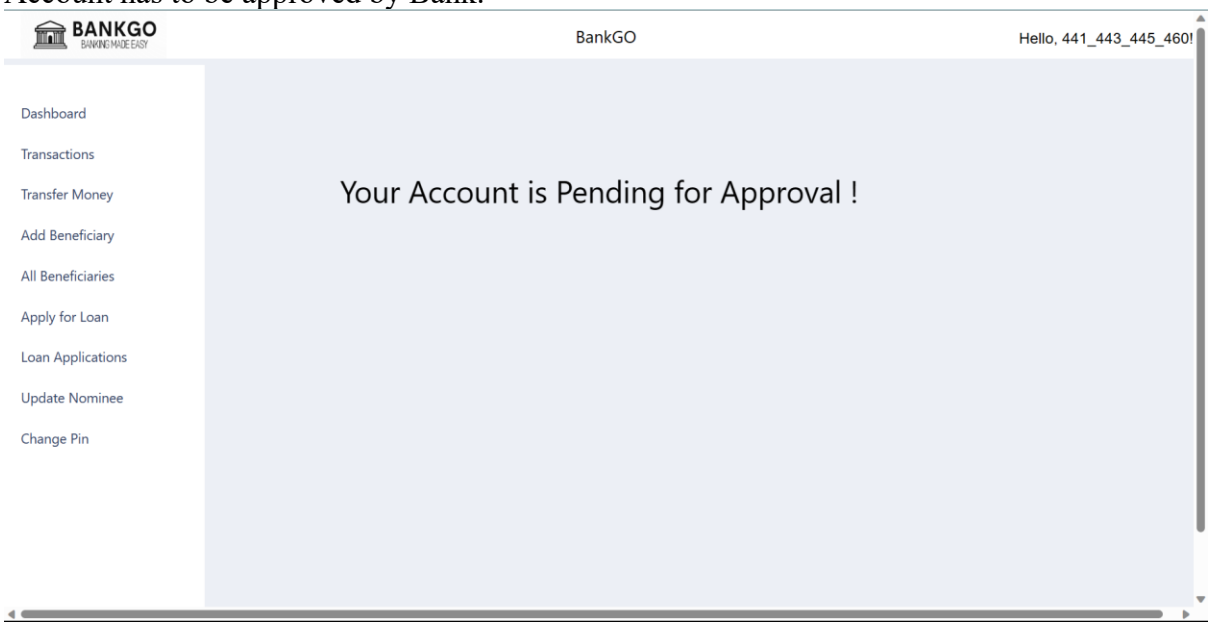
Initial Dashboard View after login for the first time



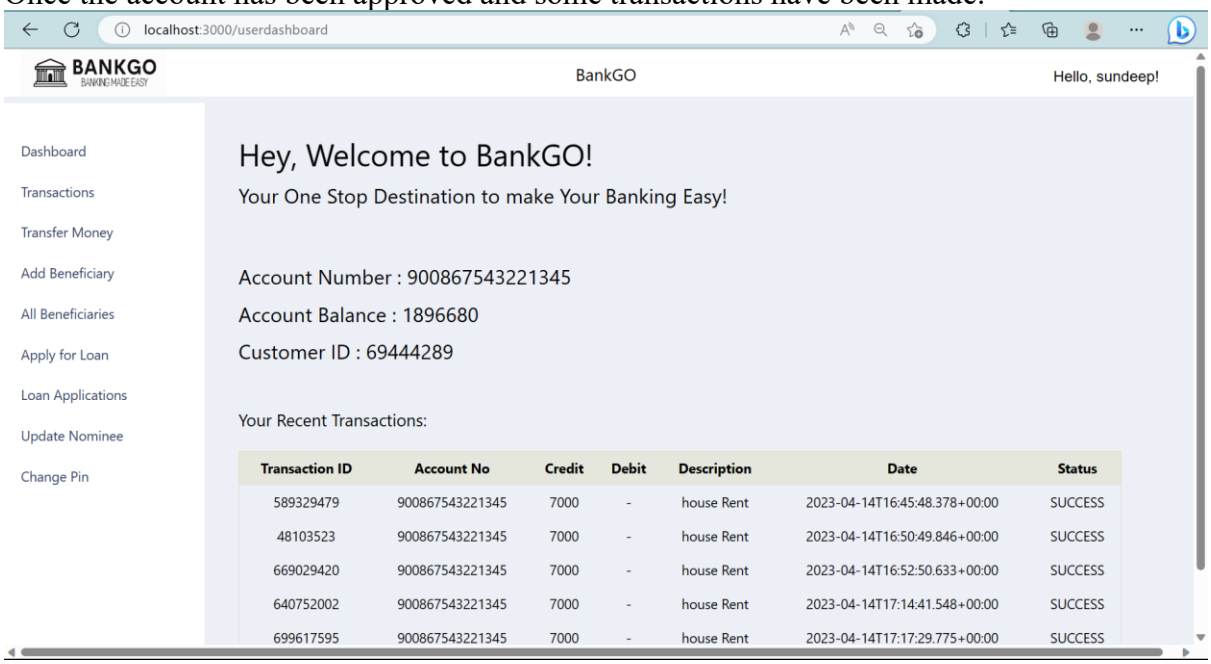
Application for new Account has been submitted.



Account has to be approved by Bank.

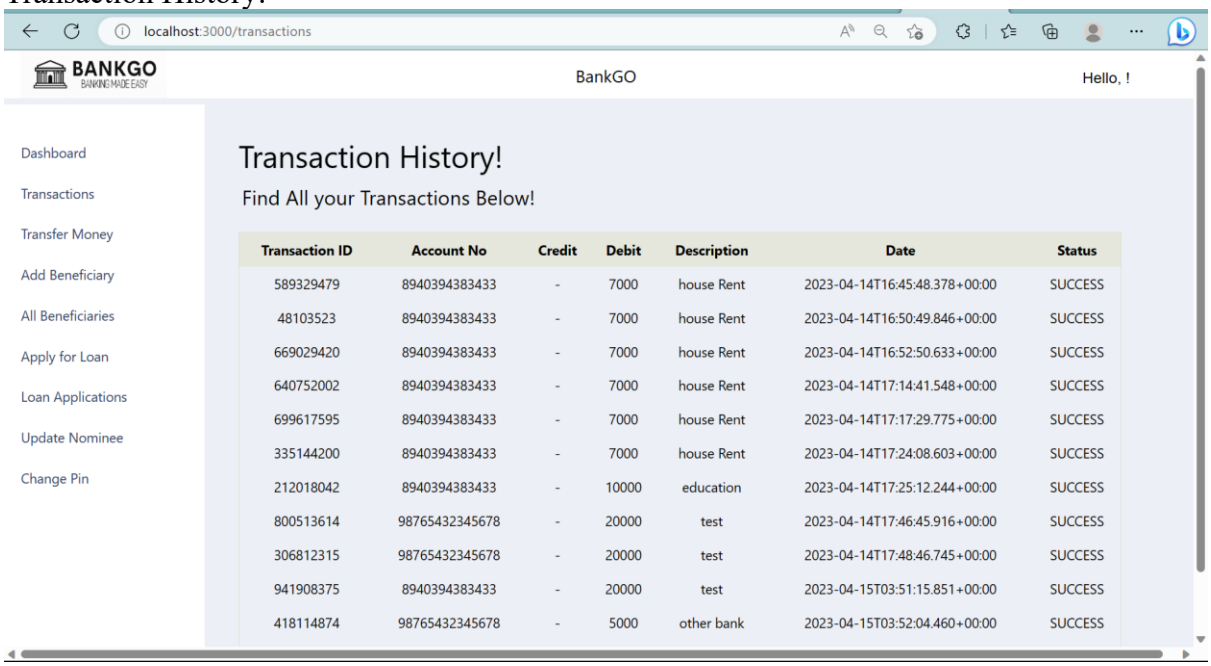


Once the account has been approved and some transactions have been made:



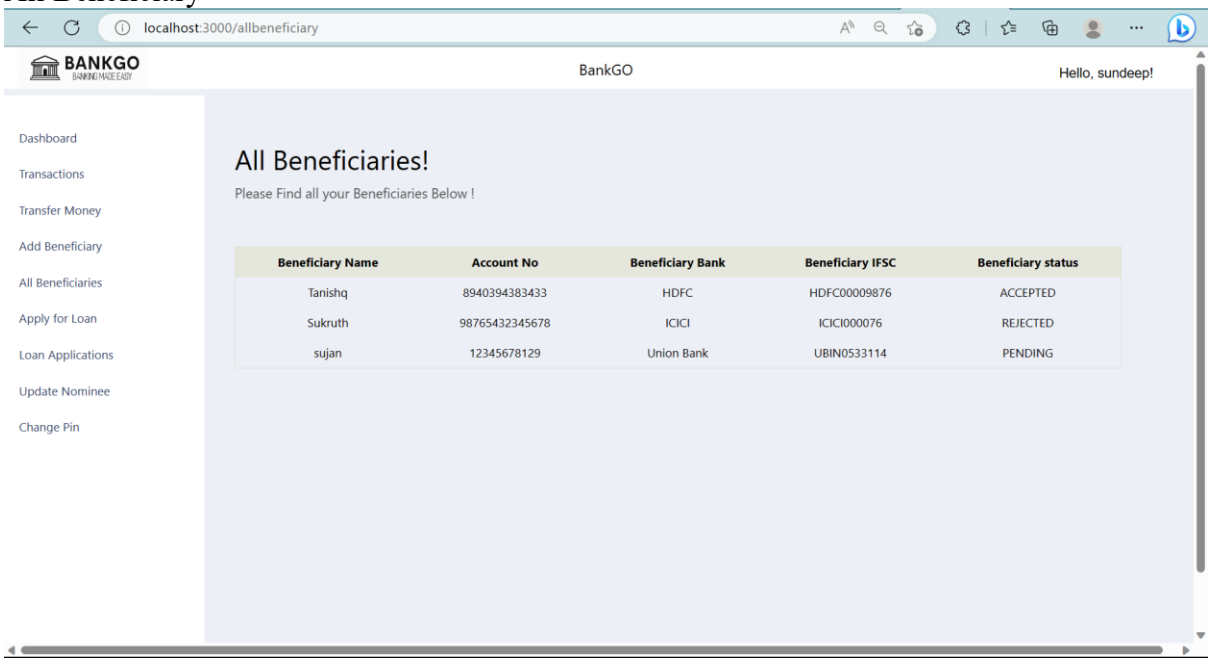
Transaction ID	Account No	Credit	Debit	Description	Date	Status
589329479	900867543221345	7000	-	house Rent	2023-04-14T16:45:48.378+00:00	SUCCESS
48103523	900867543221345	7000	-	house Rent	2023-04-14T16:50:49.846+00:00	SUCCESS
669029420	900867543221345	7000	-	house Rent	2023-04-14T16:52:50.633+00:00	SUCCESS
640752002	900867543221345	7000	-	house Rent	2023-04-14T17:14:41.548+00:00	SUCCESS
699617595	900867543221345	7000	-	house Rent	2023-04-14T17:17:29.775+00:00	SUCCESS

Transaction History:



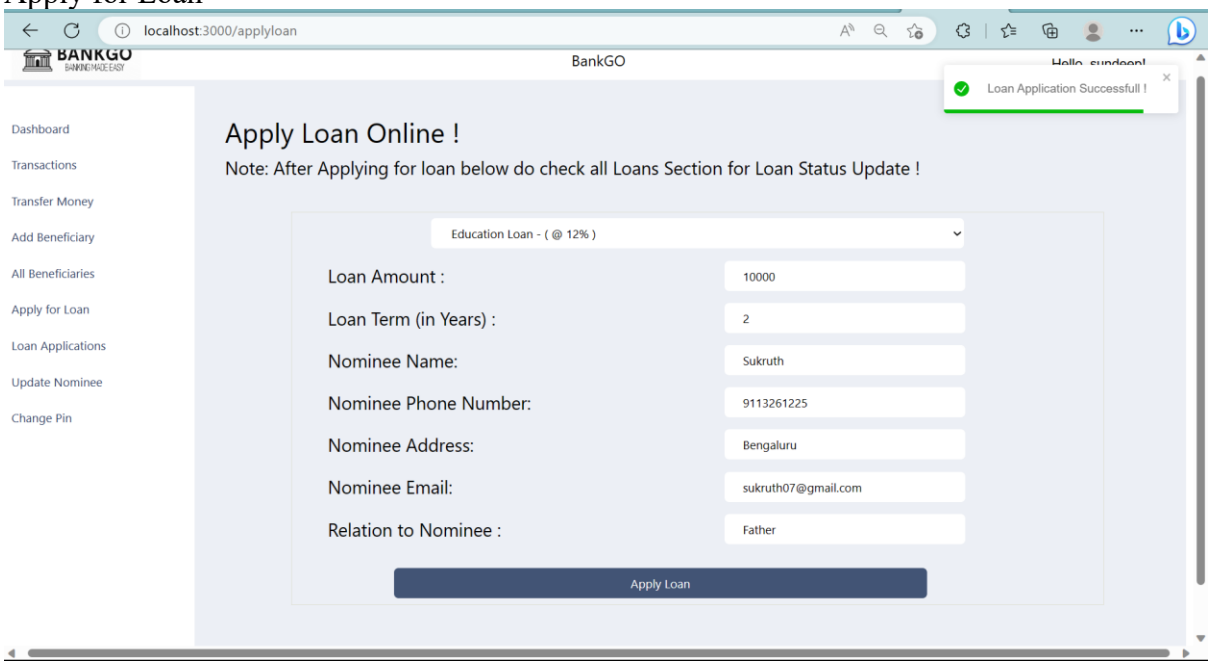
Transaction ID	Account No	Credit	Debit	Description	Date	Status
589329479	8940394383433	-	7000	house Rent	2023-04-14T16:45:48.378+00:00	SUCCESS
48103523	8940394383433	-	7000	house Rent	2023-04-14T16:50:49.846+00:00	SUCCESS
669029420	8940394383433	-	7000	house Rent	2023-04-14T16:52:50.633+00:00	SUCCESS
640752002	8940394383433	-	7000	house Rent	2023-04-14T17:14:41.548+00:00	SUCCESS
699617595	8940394383433	-	7000	house Rent	2023-04-14T17:17:29.775+00:00	SUCCESS
335144200	8940394383433	-	7000	house Rent	2023-04-14T17:24:08.603+00:00	SUCCESS
212018042	8940394383433	-	10000	education	2023-04-14T17:25:12.244+00:00	SUCCESS
800513614	98765432345678	-	20000	test	2023-04-14T17:46:45.916+00:00	SUCCESS
306812315	98765432345678	-	20000	test	2023-04-14T17:48:46.745+00:00	SUCCESS
941908375	8940394383433	-	20000	test	2023-04-15T03:51:15.851+00:00	SUCCESS
418114874	98765432345678	-	5000	other bank	2023-04-15T03:52:04.460+00:00	SUCCESS

All Beneficiary

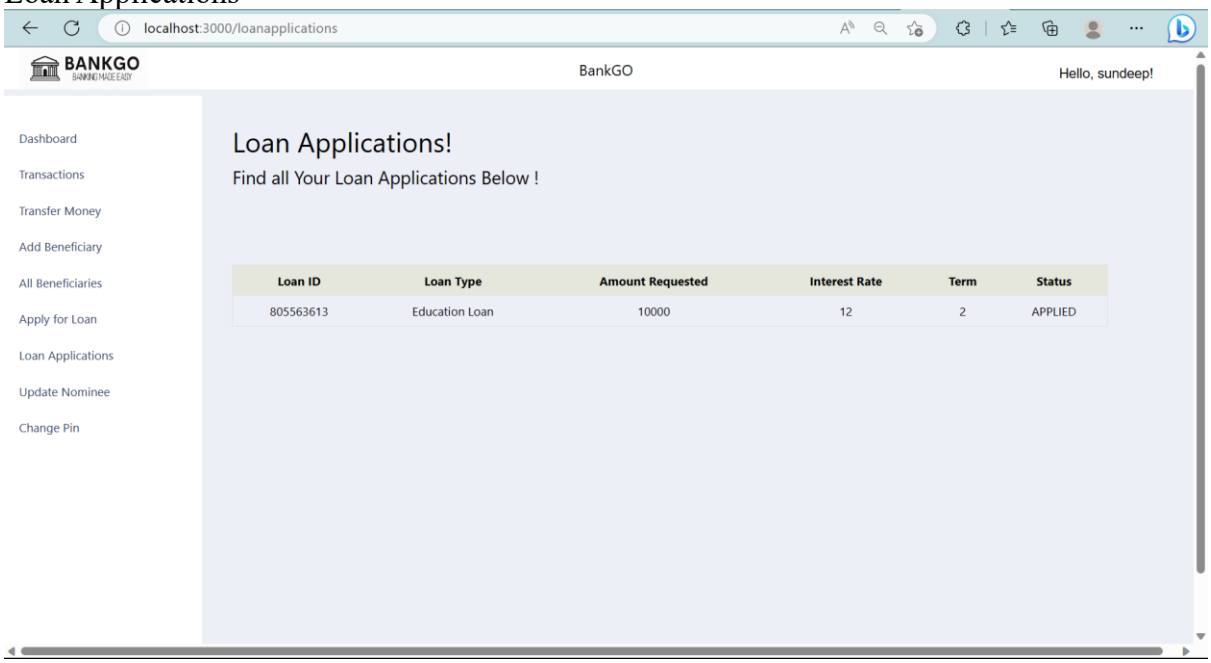


Beneficiary Name	Account No	Beneficiary Bank	Beneficiary IFSC	Beneficiary status
Tanishq	8940394383433	HDFC	HDFC00009876	ACCEPTED
Sukruth	98765432345678	ICICI	ICICI000076	REJECTED
sujan	12345678129	Union Bank	UBIN0533114	PENDING

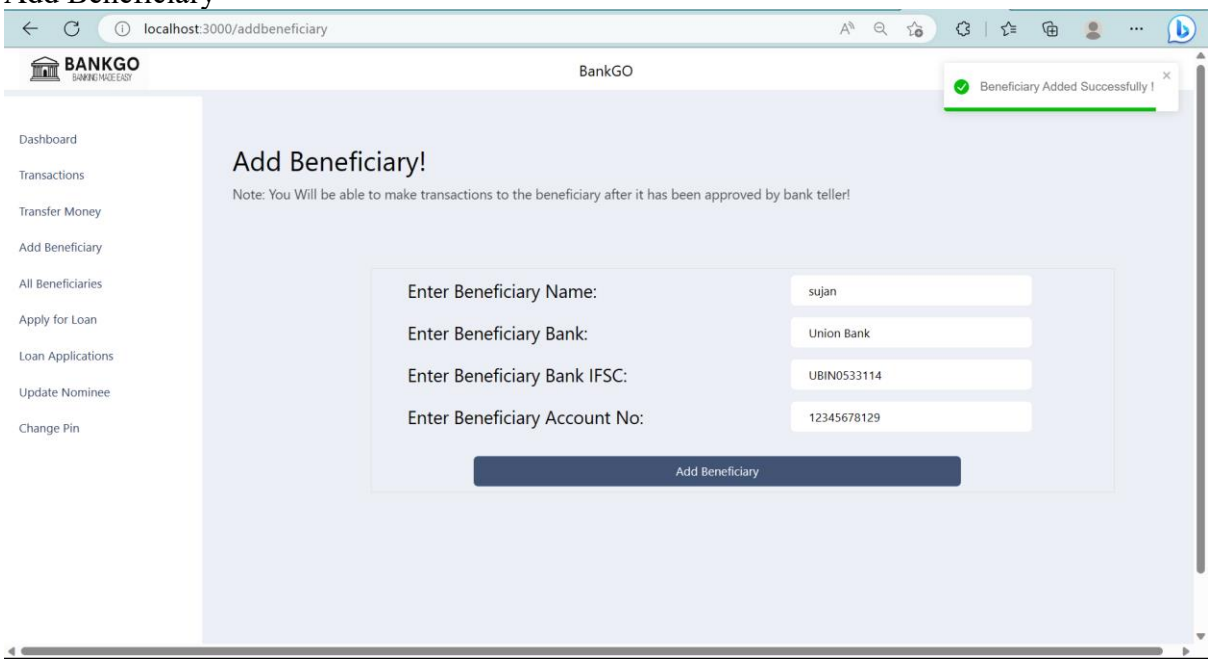
Apply for Loan



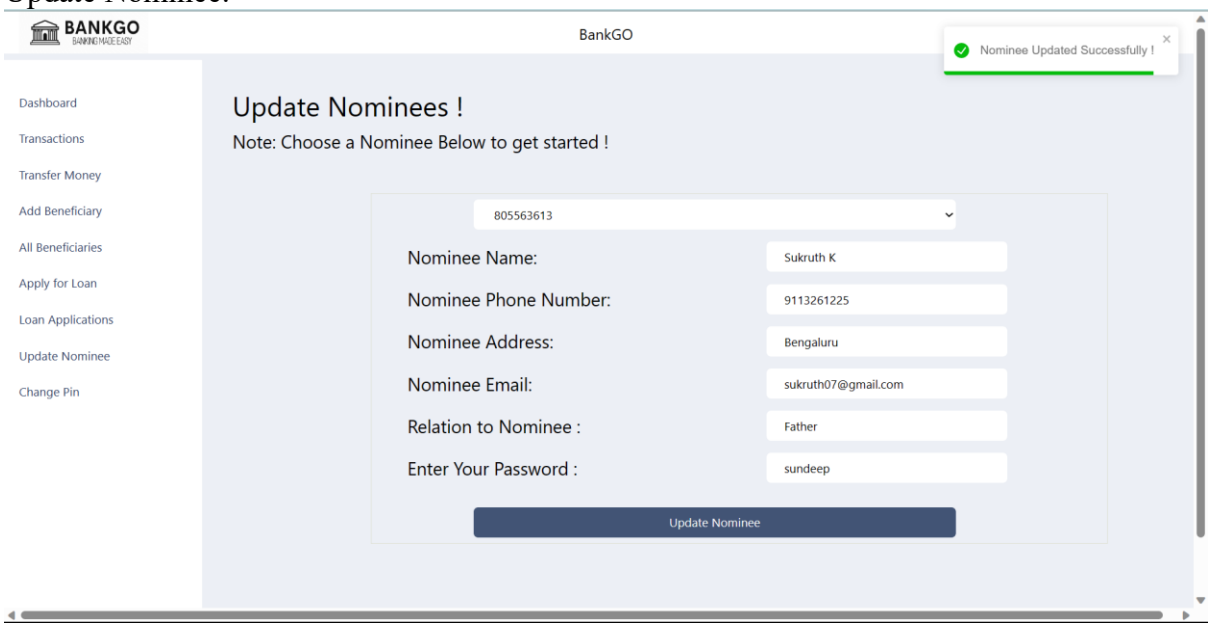
Loan Applications



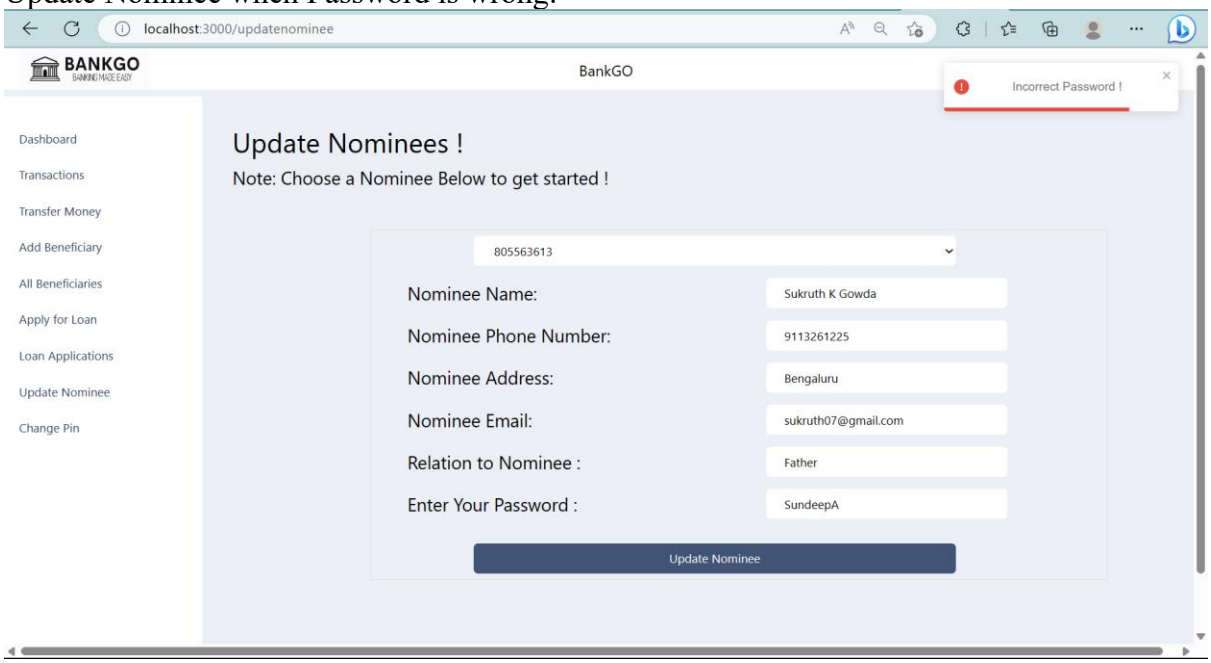
Add Beneficiary



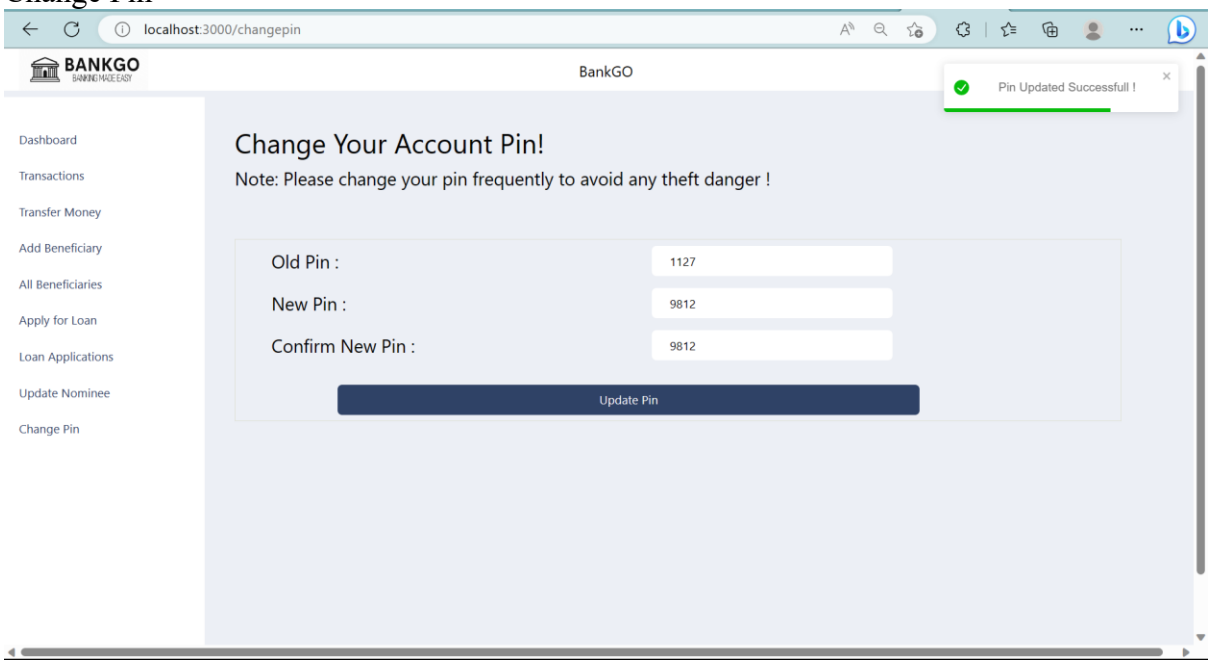
Update Nominee:



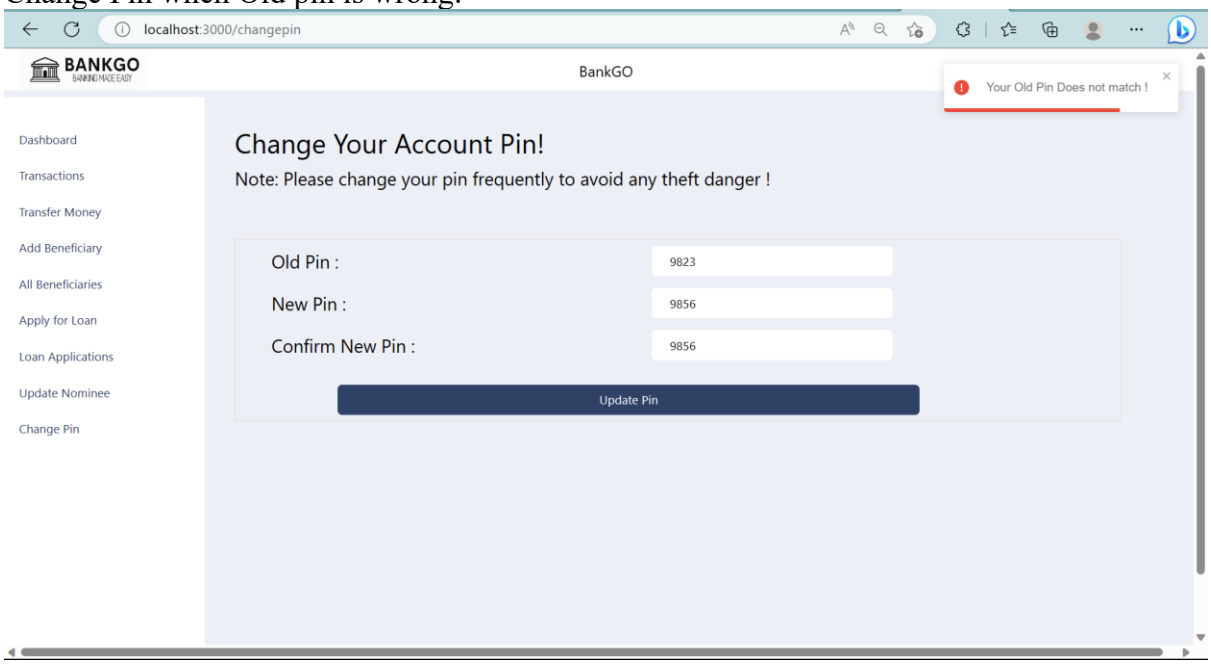
Update Nominee when Password is wrong:



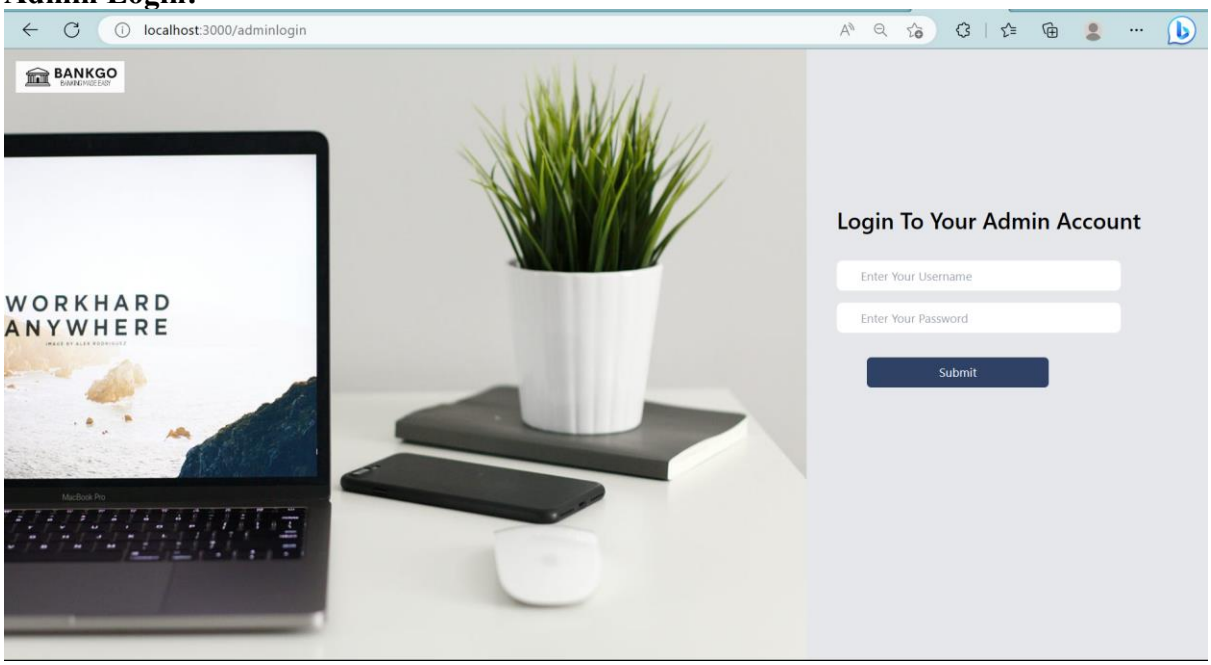
Change Pin



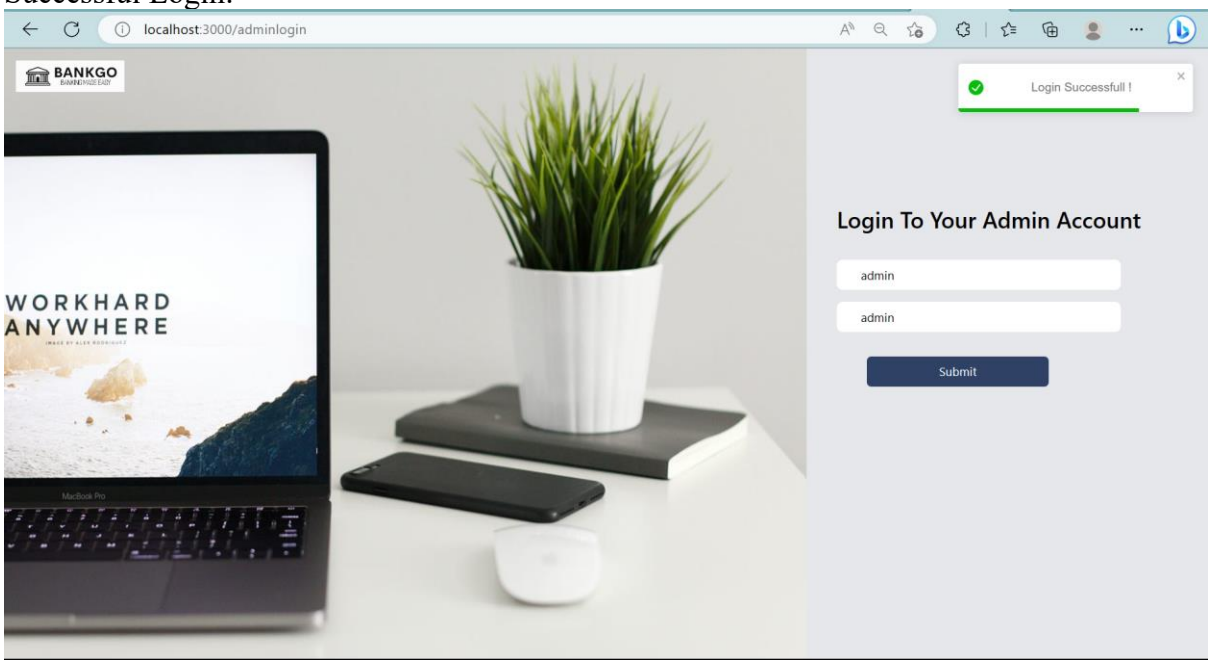
Change Pin when Old pin is wrong:



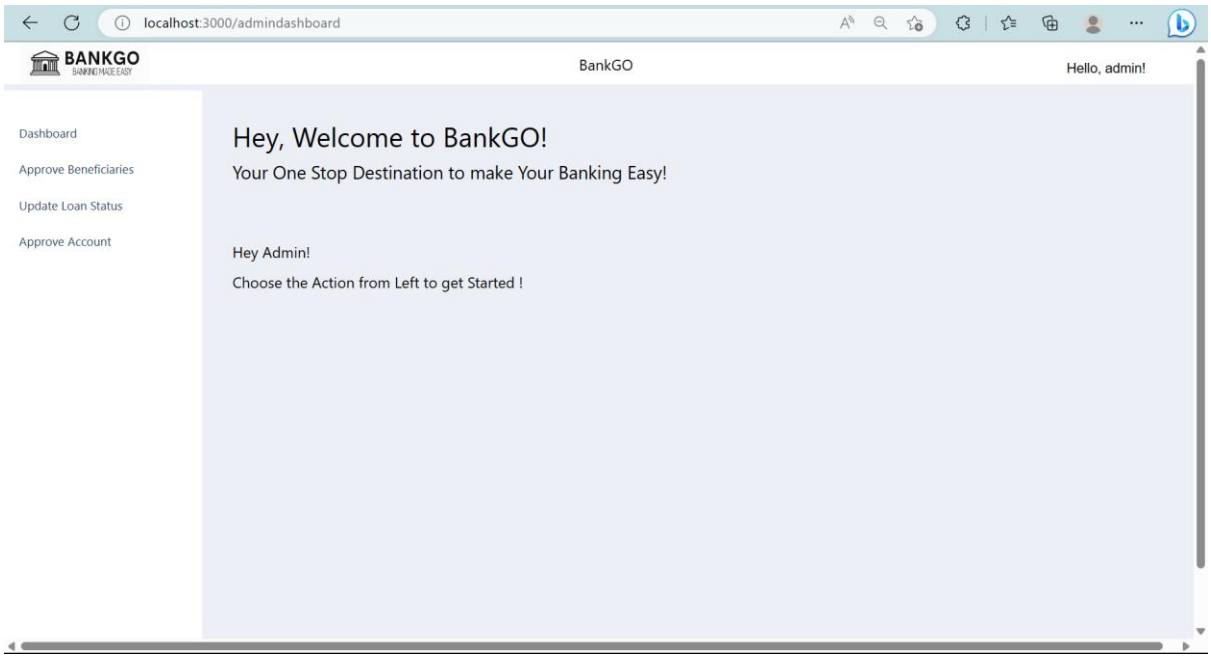
Admin Login:



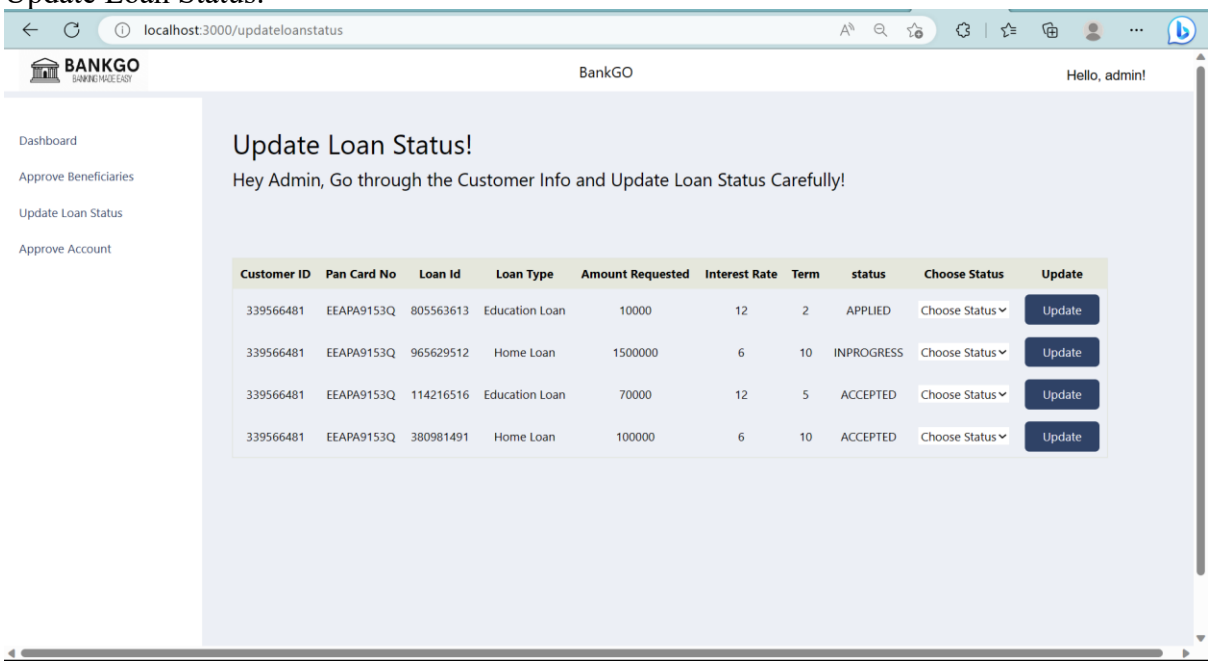
Successful Login:



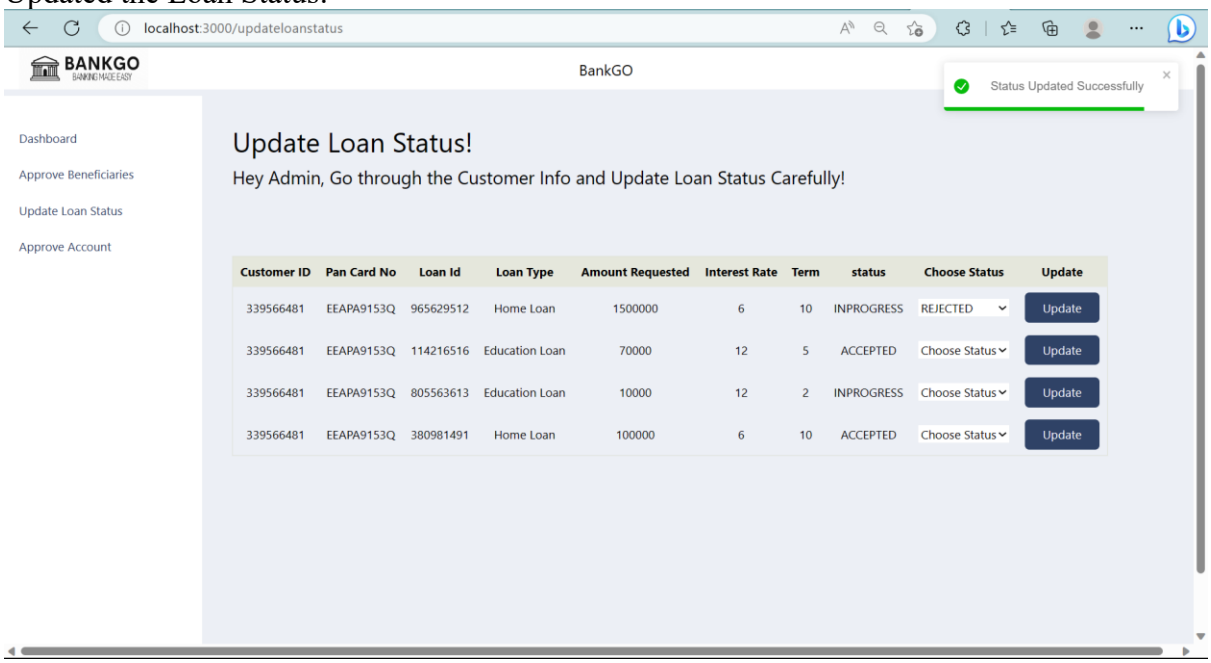
Admin Dashboard:



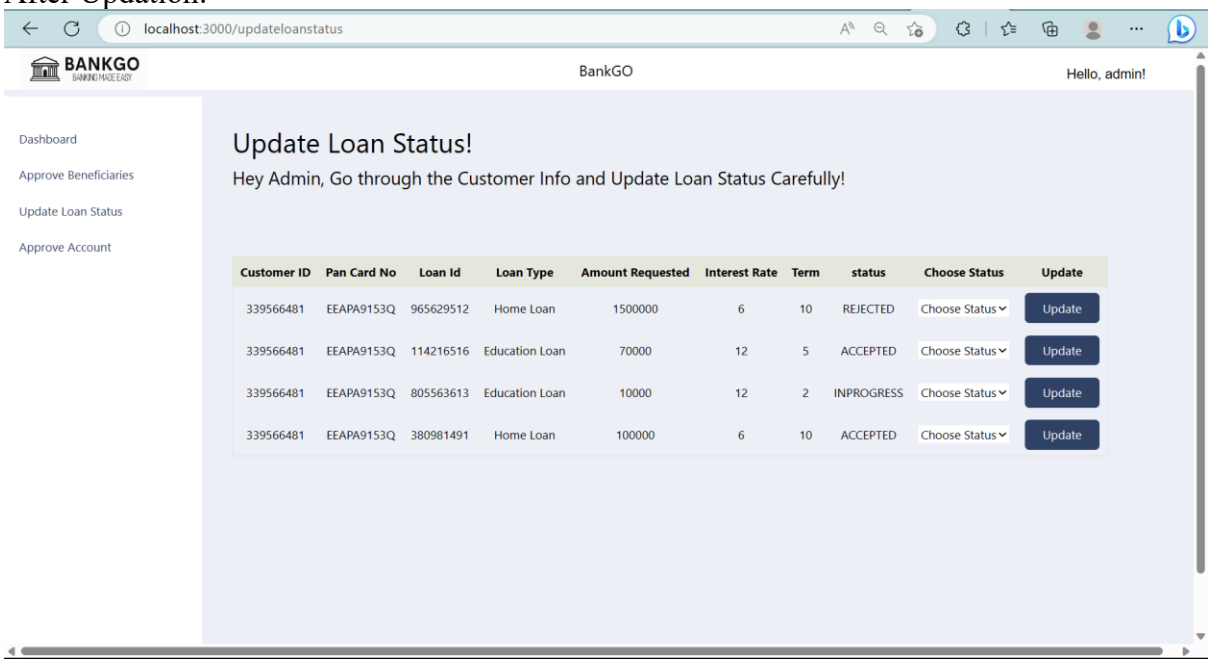
Update Loan Status:



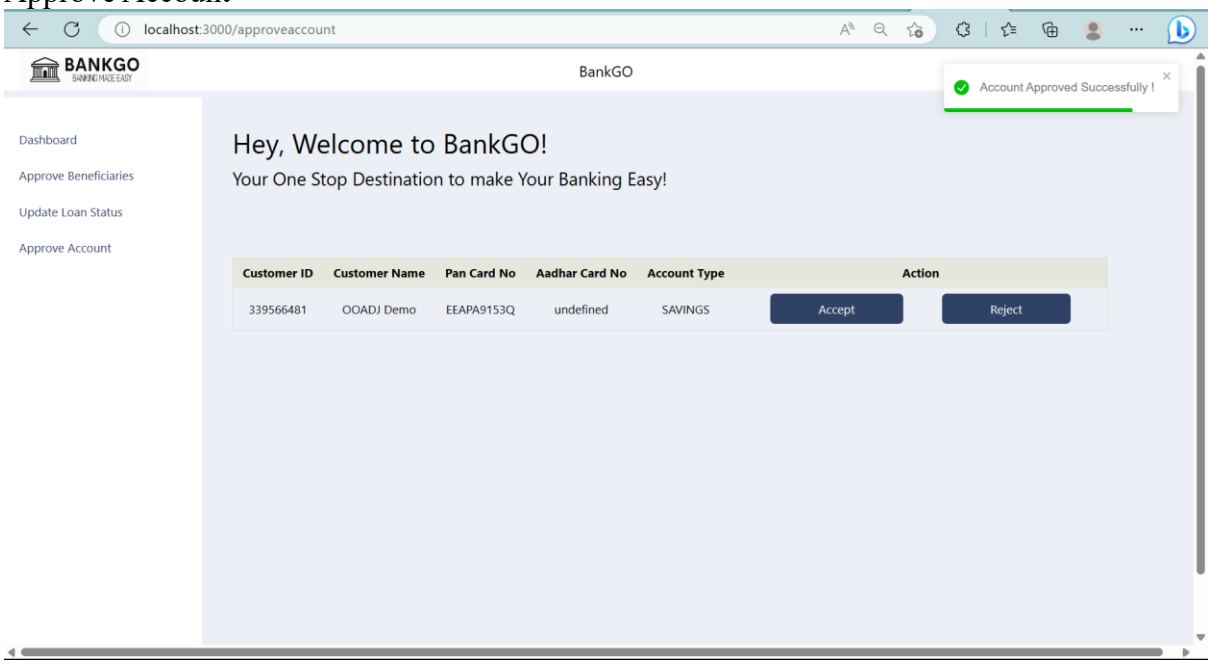
Updated the Loan Status:



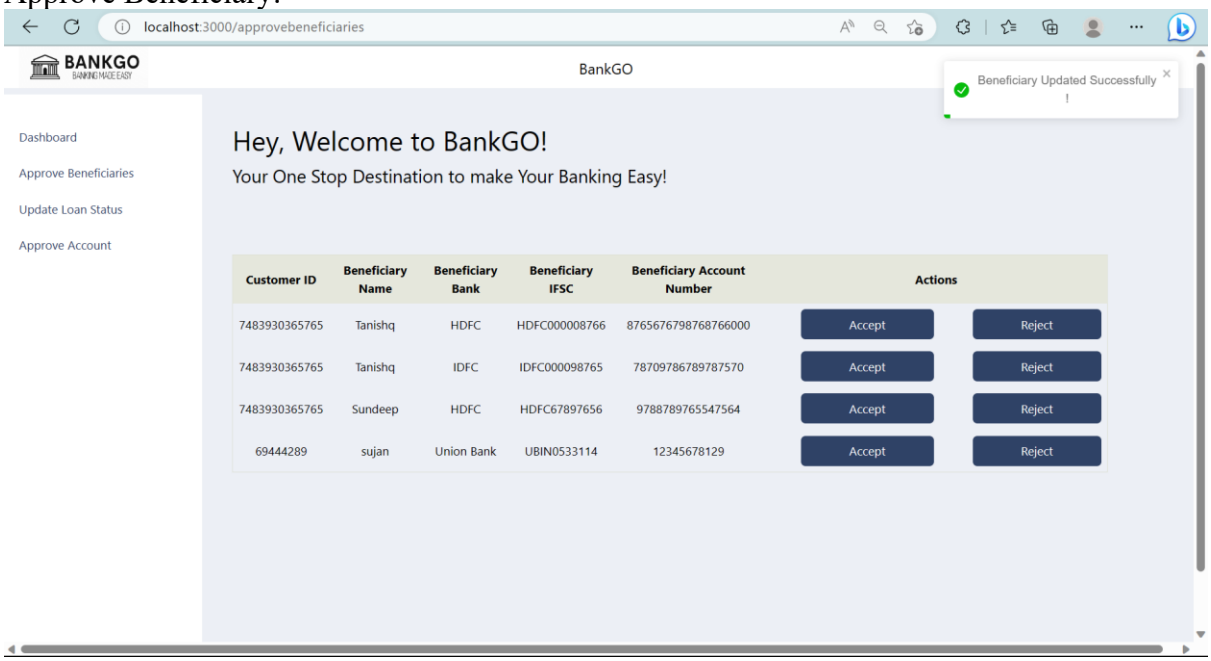
After Updation:



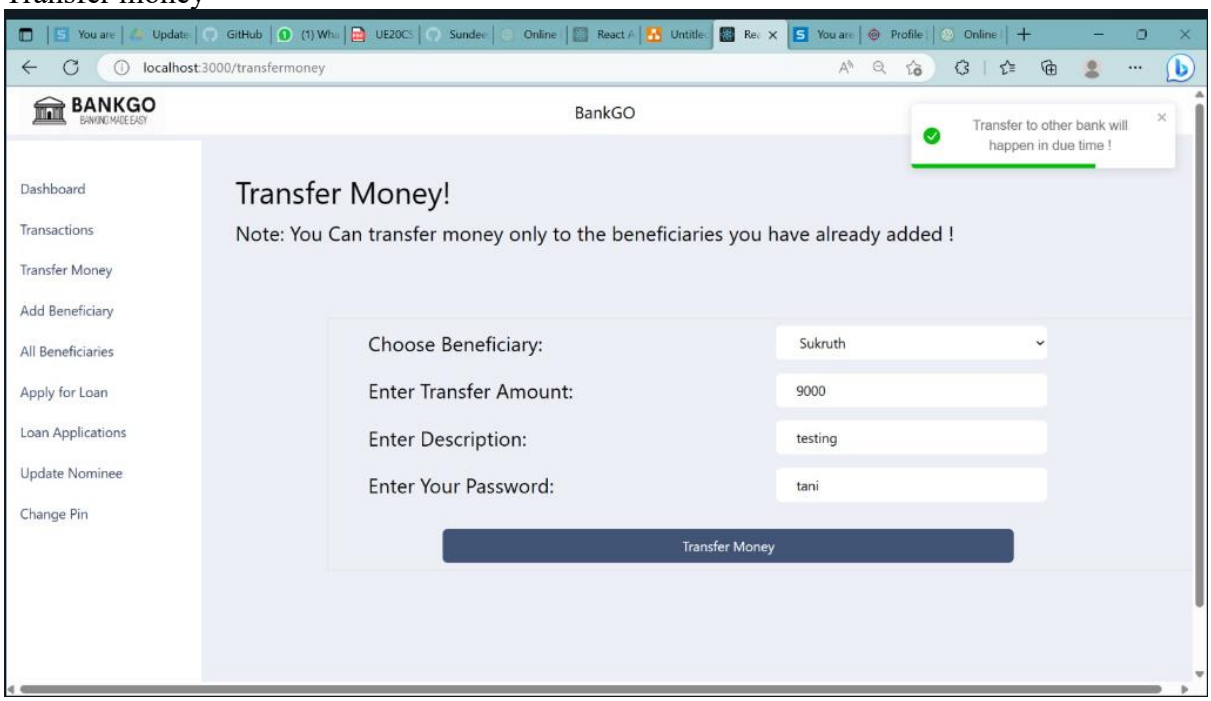
Approve Account



Approve Beneficiary:



Transfer money



Edit profile