CAPSTONE PROJECT DOCUMENTATION

NAME: SARTHAK JOSHI

OBJECTIVE: Development of a Banking Portal using React and Spring Boot

Introduction:

A banking portal is a web-based platform designed to provide users with secure and seamless access to banking services. In today's fast-paced world, customers demand convenience, speed, and security while managing their finances. Banking portals play a crucial role in meeting these demands by allowing users to perform essential tasks from the comfort of their homes or workplaces, without visiting a physical branch. This reduces operational overhead for banks while providing an enhanced customer experience.

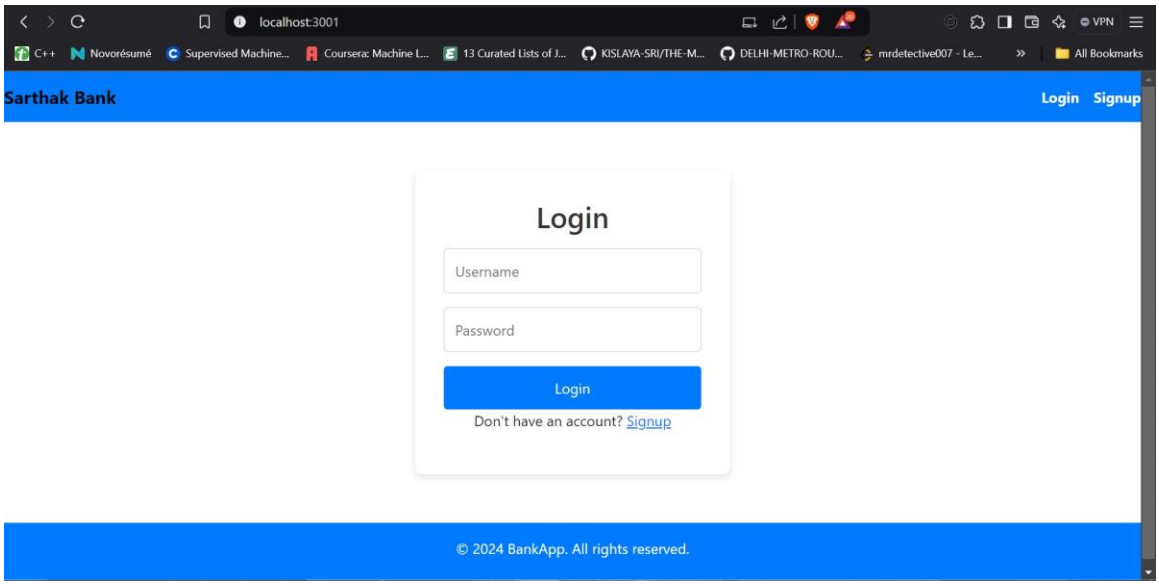
Our banking portal is designed with simplicity and security as its core principles. The frontend is built using React.js to provide a responsive and interactive user experience, while the backend is powered by Spring Boot to ensure reliability and robust functionality. The portal caters to both users and administrators, offering tailored dashboards with specific features.

Features and Functionalities

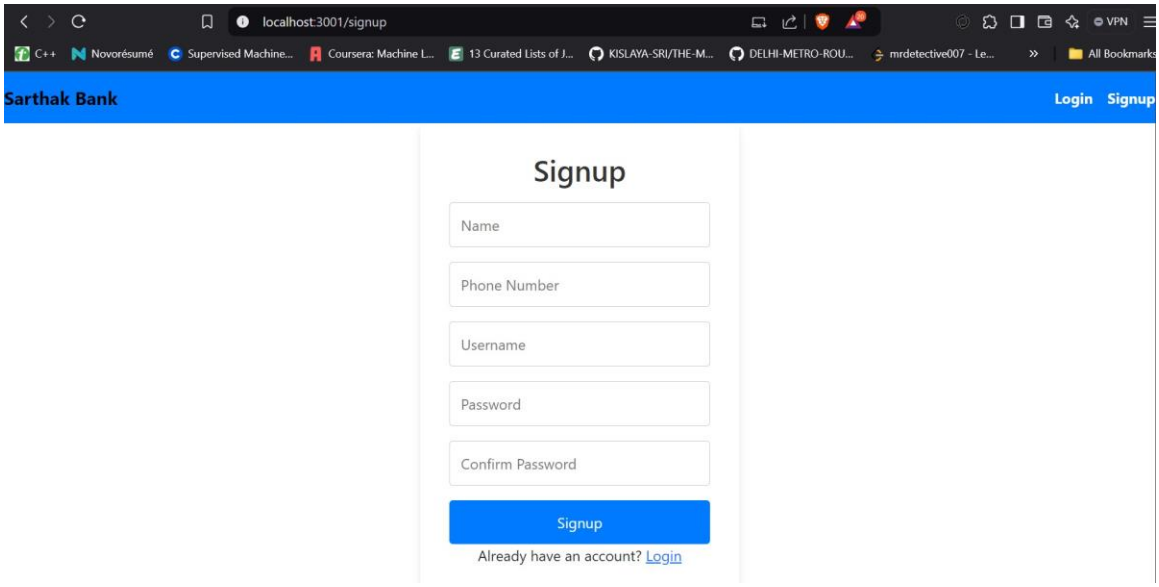
1. Login and Signup

Authentication is a fundamental feature of the banking portal. The application offers both login and signup functionalities to ensure that only registered users can access their accounts.

- **Login**: Users can securely log in using their email and password. Upon successful login, a JWT token is generated and stored in the frontend. This token is used to validate all subsequent requests, ensuring secure communication between the client and server.

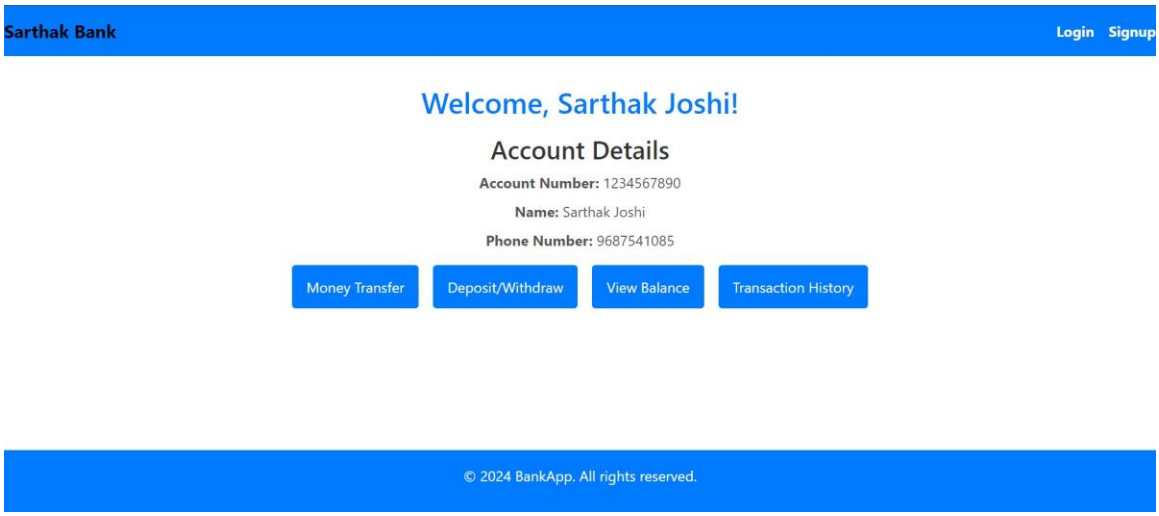


- **Signup**: New users can register by providing essential details such as name, email, password, and an initial deposit amount. The backend validates this data and stores it securely in the database. Each user is assigned a unique account number.



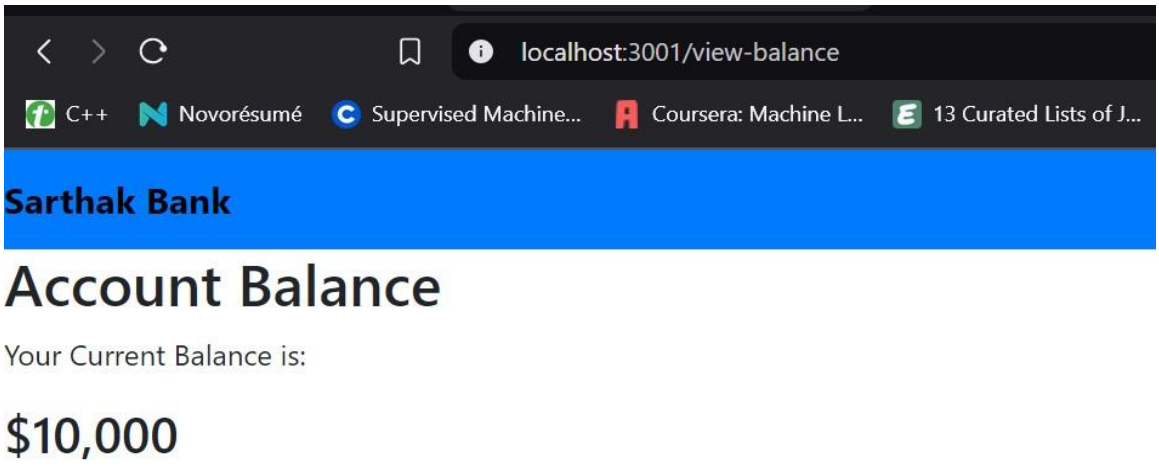
2. User Dashboard

Once logged in, users are directed to their personalized dashboard, which provides an overview of their account details. The dashboard includes:

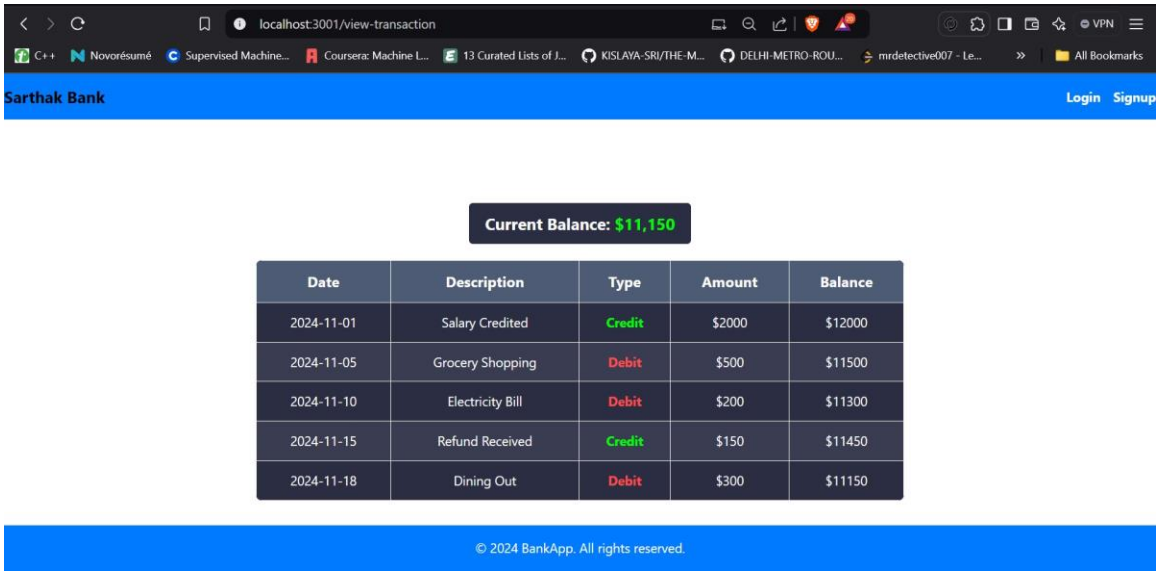


- **Account Information**: Displays the user's name, account number, and balance.
- **Functional Buttons**: Provides access to features such as:

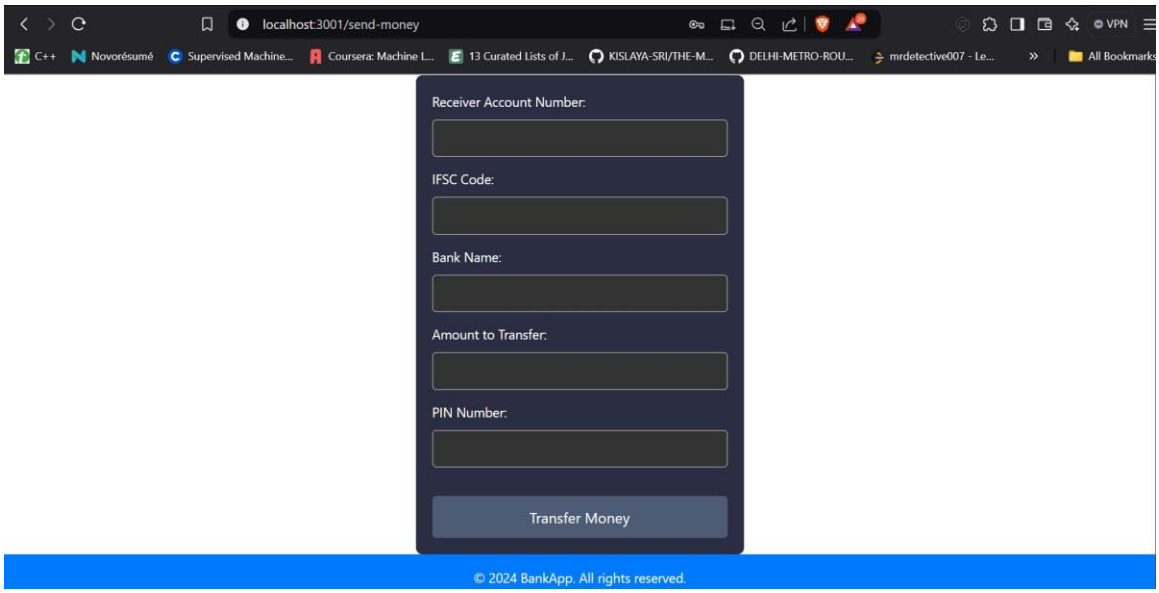
- View Balance:



- View Transaction History:



- Transfer Money:

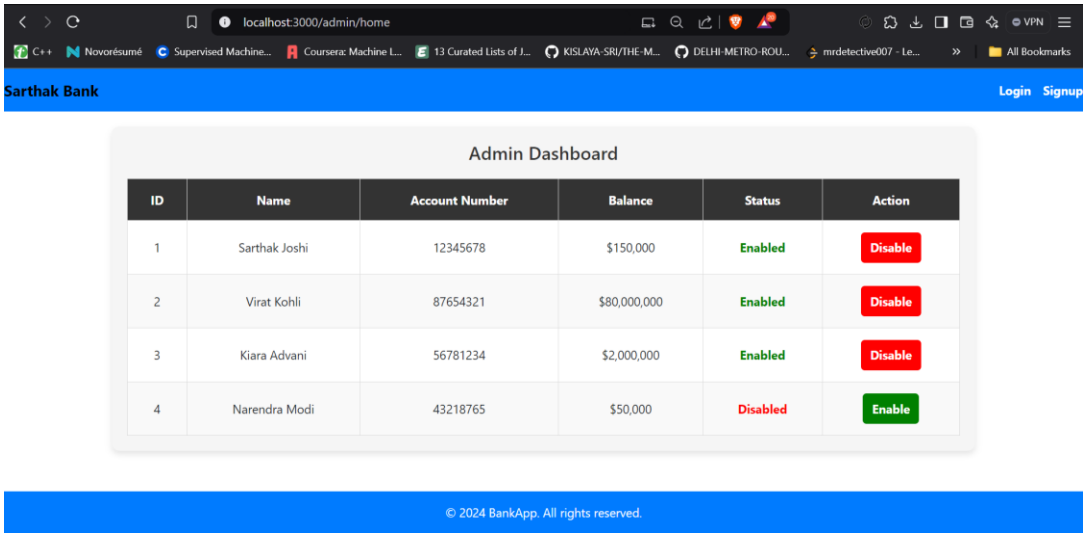


The user dashboard is designed for simplicity and usability, ensuring that users can navigate seamlessly and perform banking operations with ease.

3. Admin Dashboard

The Admin Dashboard is designed for bank administrators to monitor and manage user accounts. It provides a comprehensive view of all registered users, displaying their account numbers, names, and balances.

This dashboard ensures transparency and allows administrators to oversee the system's overall functionality.



Backend Implementation:

The backend is implemented using Spring Boot, a popular Java-based framework for building robust and scalable applications.

1. JWT Authorization

JWT (JSON Web Token) is used to secure the application and ensure that only authenticated users can access their accounts and perform transactions.
- **Login Process**: Upon successful login, the server generates a JWT token that is sent to the client. This token is stored on the client-side and included in the header of every subsequent API request.
- **Token Validation**: The backend validates the token before processing any request. If the token is invalid or expired, the request is denied.

2. Money Transfer Logic

The money transfer functionality enables users to securely transfer funds between accounts. The backend handles the following steps:
- Validates the sender's account balance to ensure sufficient funds.
- Validates the recipient's account details.
- Deducts the transfer amount from the sender's account and adds it to the recipient's account.
- Logs the transaction in the database, recording details such as the sender, recipient, amount, and updated balances.

3. Deposit Logic

Users can deposit funds into their accounts using the deposit feature. The backend processes the deposit request as follows:
- Validates the deposit amount to ensure it is a positive number.
- Adds the deposit amount to the user's balance.
- Logs the transaction in the database.

4. Withdrawal Logic

The withdrawal feature allows users to withdraw funds from their accounts. The backend performs the following checks:
- Ensures that the withdrawal amount does not exceed the user's current balance.
- Deducts the withdrawal amount from the user's balance.
- Logs the transaction in the database.

5. User Registration Logic

New users can register for the portal through the signup page. The backend:
- Validates the user's input data (e.g., email format, password strength).
- Generates a unique account number for the user.
- Stores the user's details in the database.

6. Transaction History

Users can view their transaction history on the dashboard. Each transaction includes:
- Date and time
- Transaction type (credit or debit)
- Amount
- Updated account balance

The transaction history is displayed in a tabular format, with credits highlighted in green and debits in red for clarity.

Database Design:

1. **User Table**:
   - Fields: `id`, `name`, `email`, `password`, `accountNumber`, `balance`.

User Table

ID	Name	Email	Password	Account Number	Balance
1	Sarthak Joshi	sarthak.joshi@example.com	**********	12345678	$25,000
2	Virat Kohli	virat.kohli@example.com	**********	87654321	$50,000

2. **Transaction Table**:
   - Fields: `id`, `senderAccount`, `receiverAccount`, `amount`, `type` (credit/debit), `timestamp`, `updatedBalance`.

Transaction Table

ID	Sender Account	Receiver Account	Amount	Type	Timestamp	Updated Balance
1	12345678	87654321	$5,000	Debit	2024-11-20 10:30:00 AM	$20,000
2	87654321	12345678	$3,000	Credit	2024-11-20 11:00:00 AM	$53,000

API ENDPOINT TABLE:

Method	Endpoint	Description
POST	/auth/login	Authenticates the user and generates a JWT token for secure access.
POST	/auth/signup	Registers a new user by storing user details and creating an account number.
GET	/user/dashboard	Fetches the user's account details, including balance and basic information.
GET	/user/transactions	Retrieves the transaction history of the logged-in user.
POST	/user/transfer	Transfers funds to another account after validating the sender's balance.
POST	/user/deposit	Adds a specified amount to the user's account balance.
POST	/user/withdraw	Deducts a specified amount from the user's account after validating balance.
GET	/admin/users	Fetches a list of all users, including account numbers and balances.
PUT	/admin/user/{id}/enable	Enables a user account by changing their status to active.
PUT	/admin/user/{id}/disable	Disables a user account by changing their status to inactive.

GITHUB LINK: https://github.com/KingfisheR2001/capstone-submit

Future Enhancements:

- **Multi-Currency Support**: Enable transactions and balances in multiple currencies.
- **Loan Management**: Add features to apply for and manage loans.
- **Enhanced Admin Panel**: Include account blocking, analytics, and detailed monitoring capabilities.
- **Two-Factor Authentication**: Add an additional layer of security to user accounts.