**Capstone Project Documentation**

**Title:**

**Capstone Project - Bank Management System**

**Introduction:**

The **Bank Management System** is a web-based application designed to provide users with secure and efficient tools for managing their financial activities. This system includes user authentication, account management, transaction processing, and an EMI calculator for loan planning. The application integrates a robust backend powered by **Spring Boot** and a responsive frontend built with **React.js**, ensuring a seamless user experience.

**Objective:**

The primary goal of this project is to provide users with an intuitive platform for managing their finances, with features such as account management, secure transactions, and loan planning. The system ensures data security, user privacy, and accurate financial operations.

**Topics Covered:**

1. **Modules and Features**
   * Authentication Module
   * Account Management Module
   * Transaction Management Module
   * Loan Module (EMI Calculator)
2. **User flow**
3. **Backend Endpoints**
   * Overview of API endpoints and their functionality.
4. **Frontend URLs**
   * Mapping of user-facing URLs to their respective components.
5. **Database Schema**
   * Details of tables and relationships used in the application.
6. ER-Daigram
7. **Tools and Technologies**
   * Spring Boot, React.js, MySQL, JWT, and more.
8. Github (Link to code)

**1. Modules and Features**

**1.1 Authentication Module**

Handles user registration, login, and authentication. Uses **JWT** for secure session management.

* **Features**:
  + **User Registration**:  
    Creates a user account and stores hashed passwords in the database.
  + **User Login**:  
    Authenticates the user and issues a JWT token for subsequent requests.
* **Database Changes**:
  + User table is updated with new records for registration.

**1.2 Account Management Module**

Enables users to create, view, and delete financial accounts.

* **Features**:
  + **Create Account**:  
    Adds a new account to the user’s profile.
  + **View Accounts**:  
    Lists all accounts linked to the user.
  + **Delete Account**:  
    Removes an account from the database.
* **Database Changes**:
  + Account table is updated with new or modified records.

**1.3 Transaction Management Module**

Allows users to perform deposits, withdrawals, and transfers between accounts.

* **Features**:
  + **Deposit**: Adds funds to an account.
  + **Withdraw**: Deducts funds from an account.
  + **Transfer**: Moves funds between accounts.
  + **View Transactions**: Lists all transactions for a user or account.
* **Database Changes**:
  + Updates the Account table for balances.
  + Adds entries to the Transaction table for transaction history.

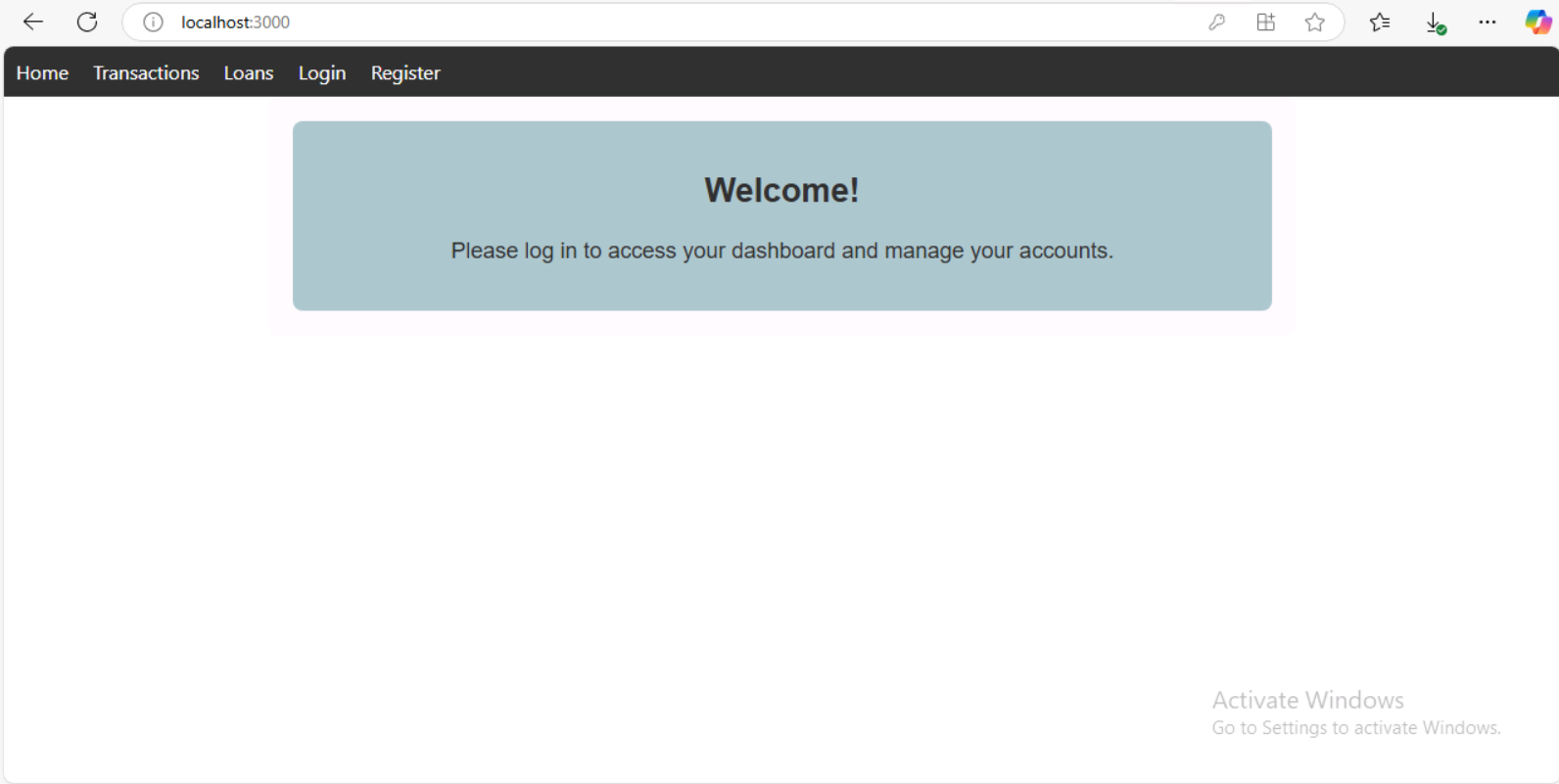
**1.4 Loan Module (EMI Calculator)**

Helps users calculate loan EMIs based on input parameters like loan amount, interest rate, and tenure.

* **Features**:
  + Calculates and displays monthly installment amounts using financial formulas.
  + Provides users with insights for better financial planning.
* **Database Changes**:
  + No direct database impact; functionality is calculation-based.

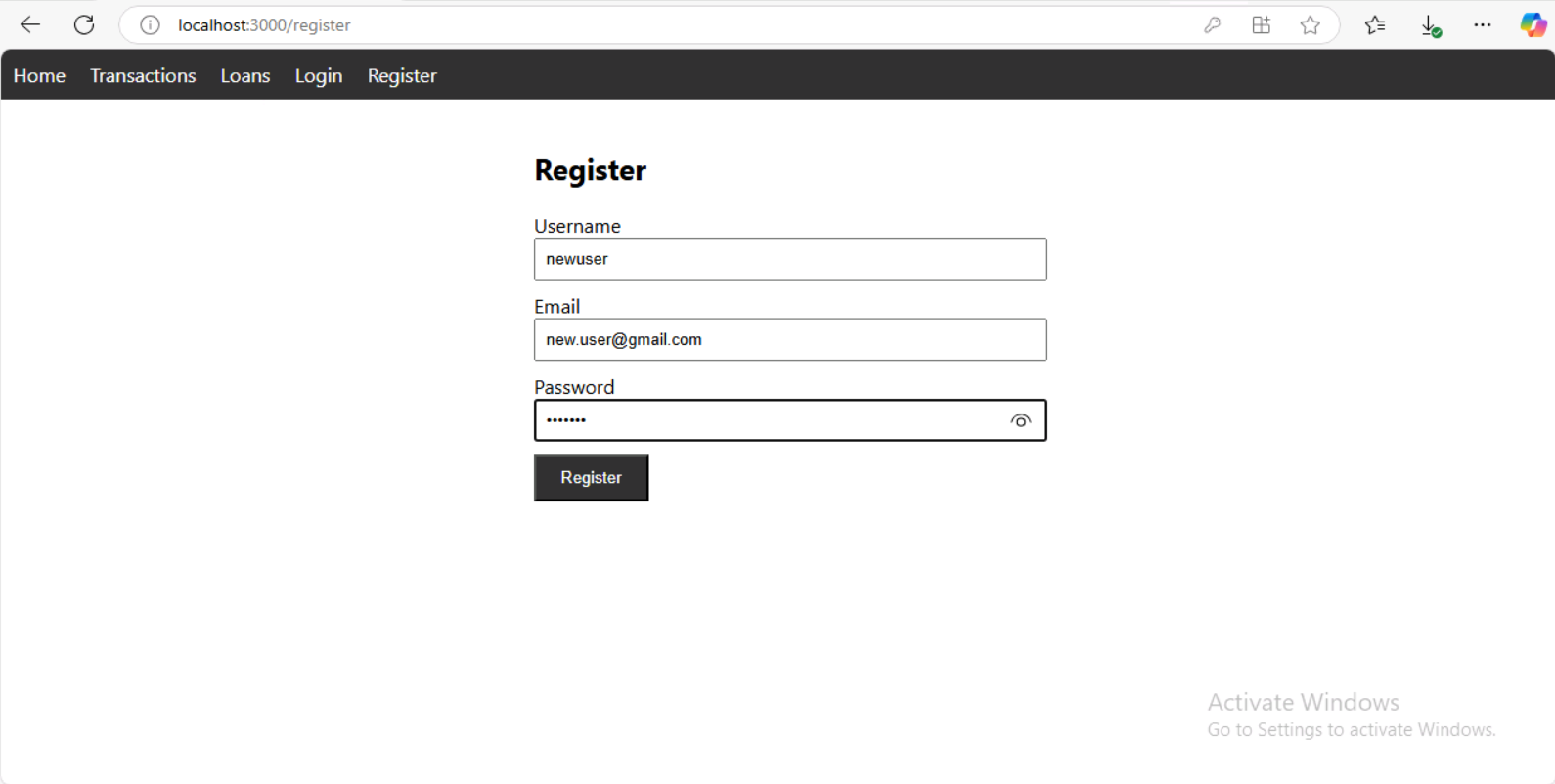
**2. User flow**

**Landing Page:**

****

**2.1 Authentication**

* Register Flow

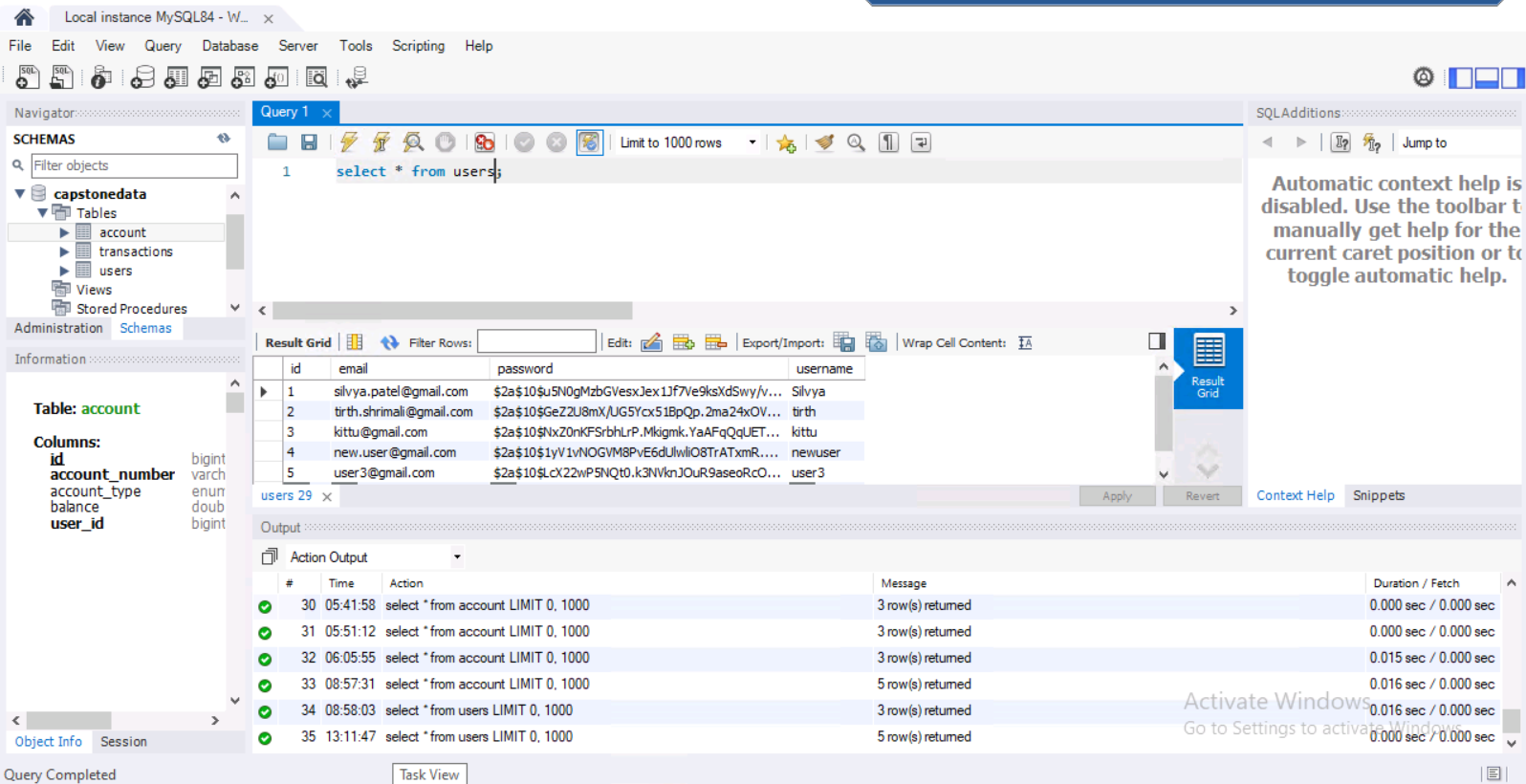


Registration form UI.

Details:

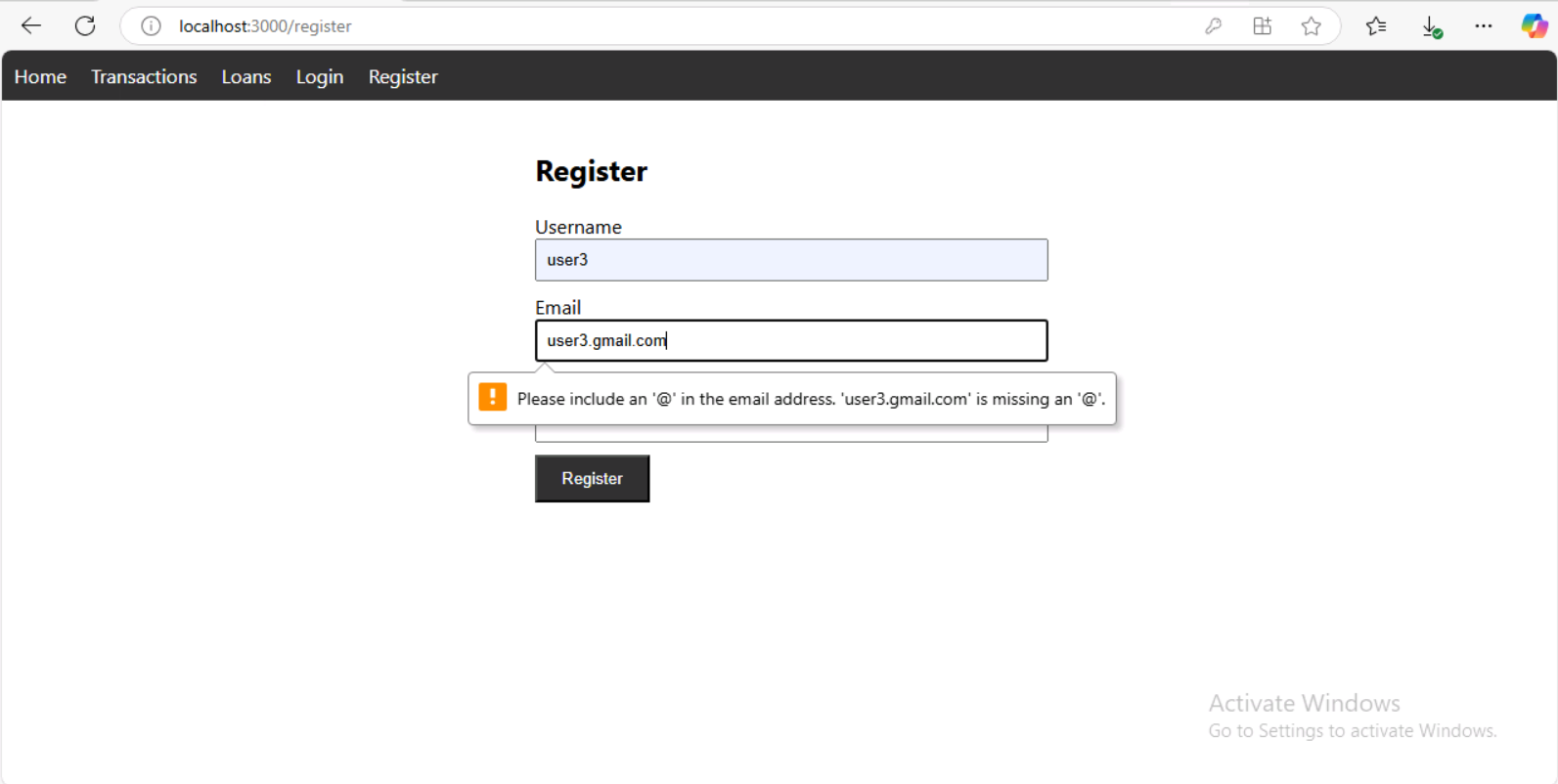
Input: Name, Email, Password.

Process: Validation → Save user details in the database.

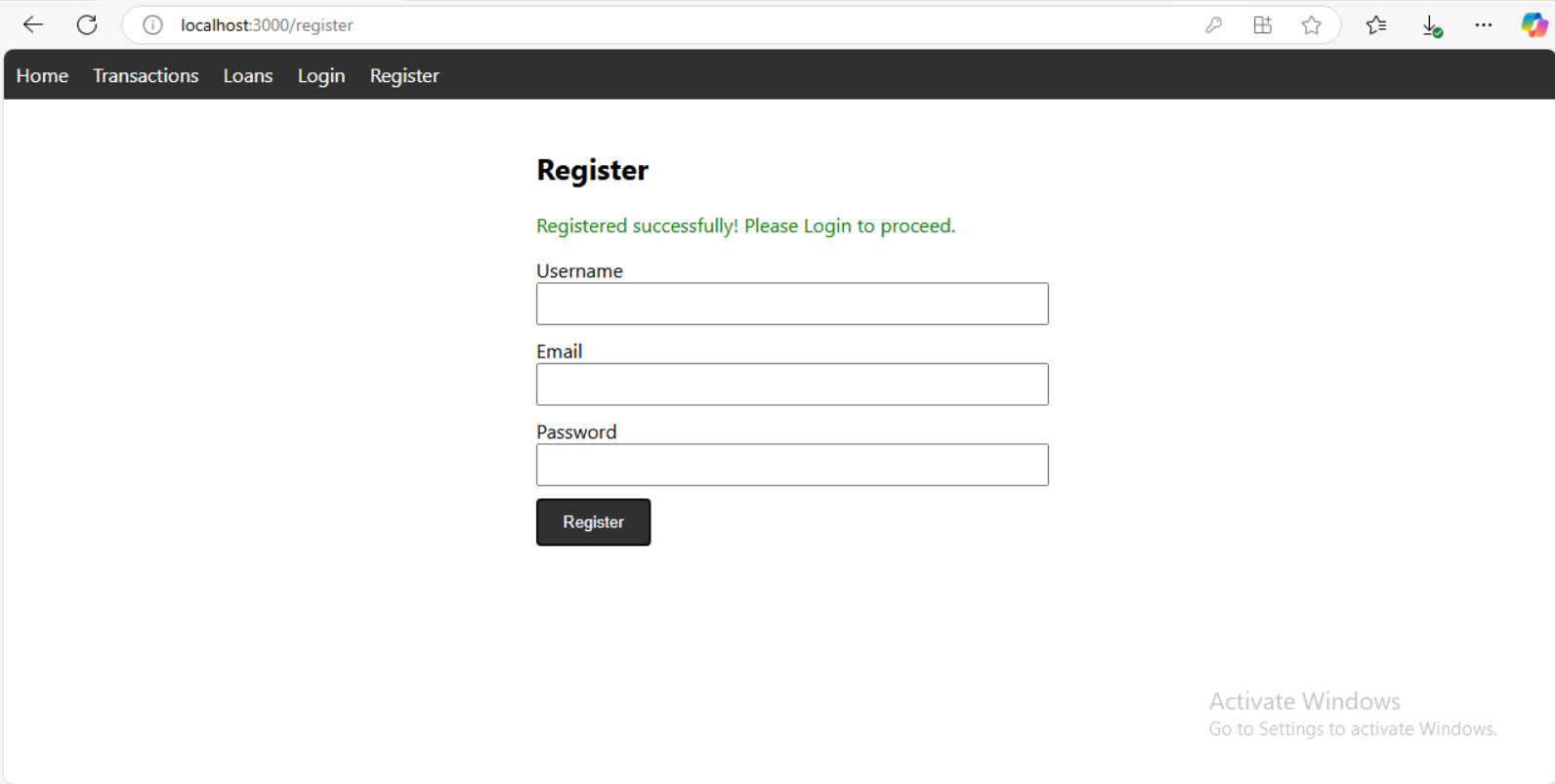


User Table.

Output:

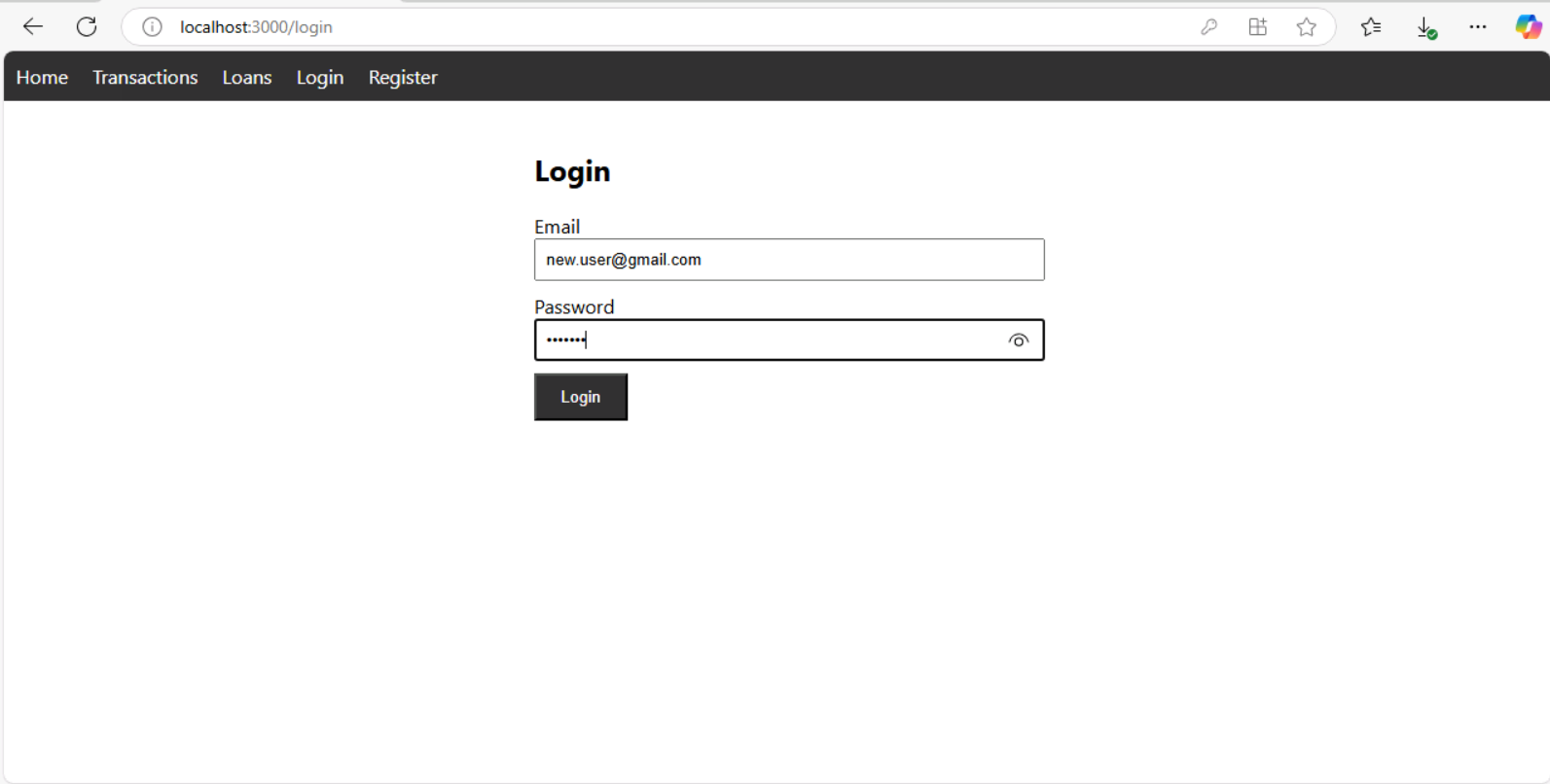


error message.



Success message and **redirect** to login.

* Login Flow



Login form UI.

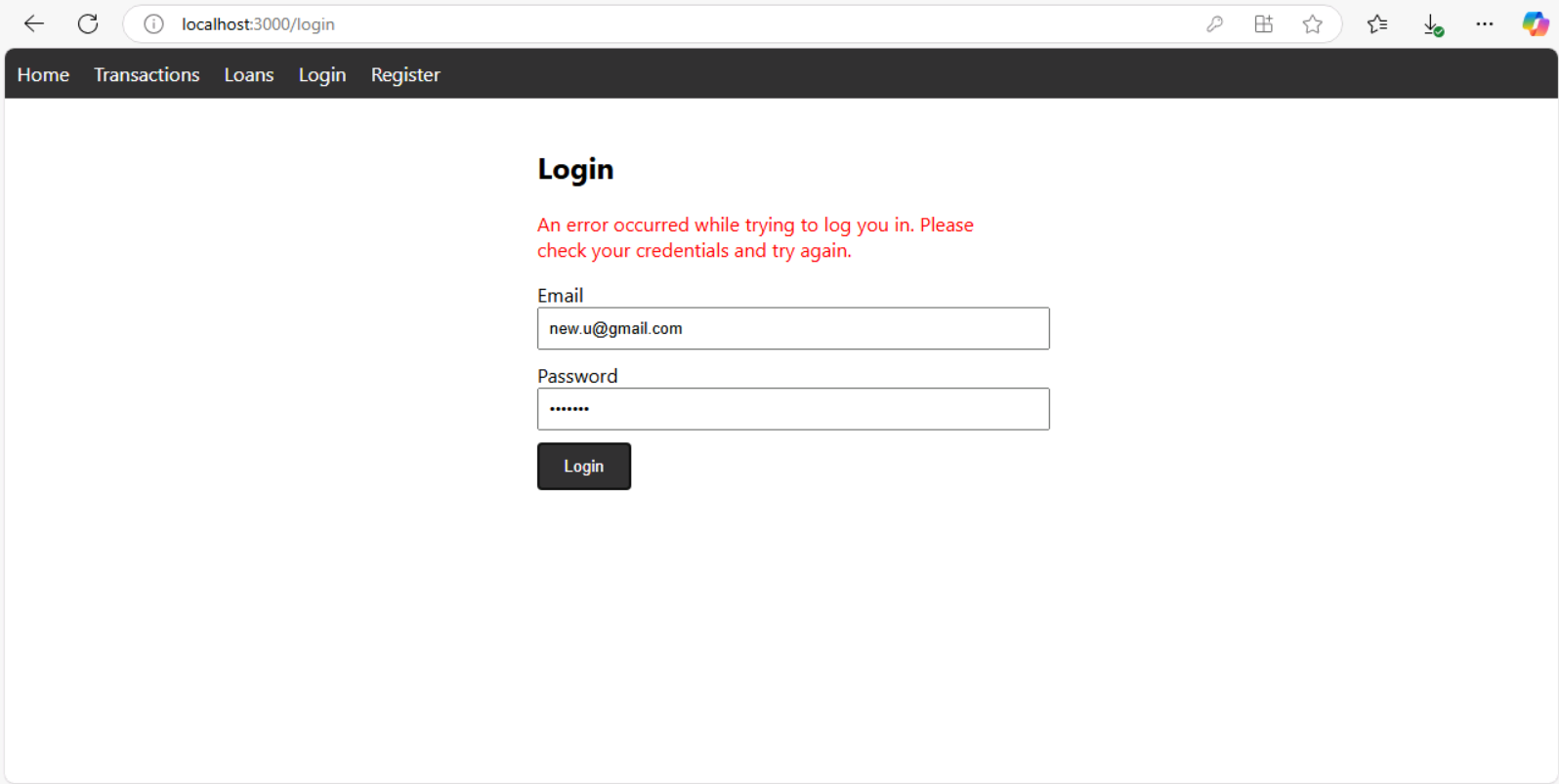
Details:

Input: Email, Password.

Process: Verify credentials → Generate JWT → Store in localStorage.

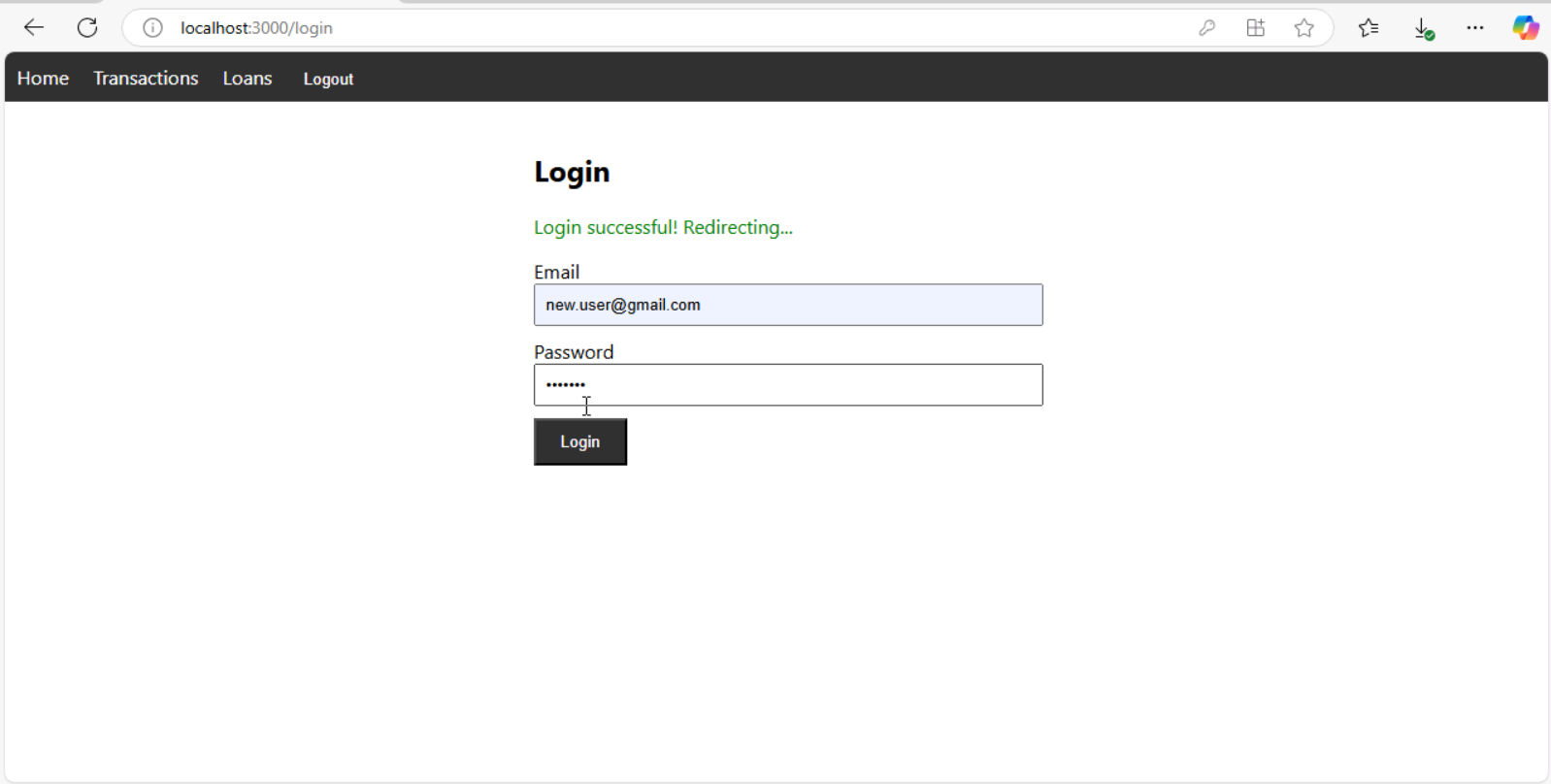
Output:

Upon error:



error message.

Upon success: Login, Register buttons are not displayed anymore. Instead there is Logout now

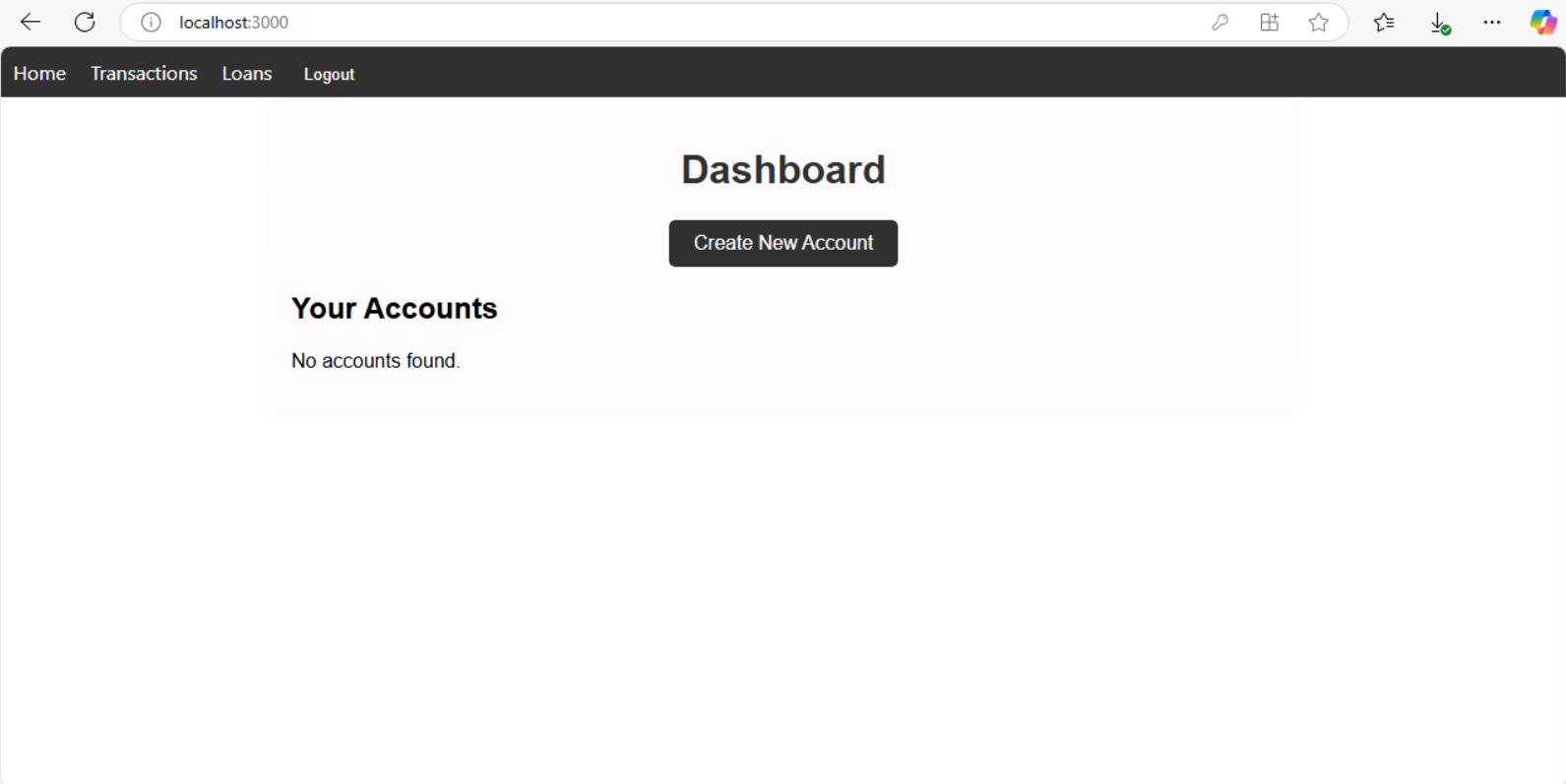


**Redirect** to the dashboard.

**2.2 Account Management**

The landing page changes to Dashboard according to user logged in

* View Accounts

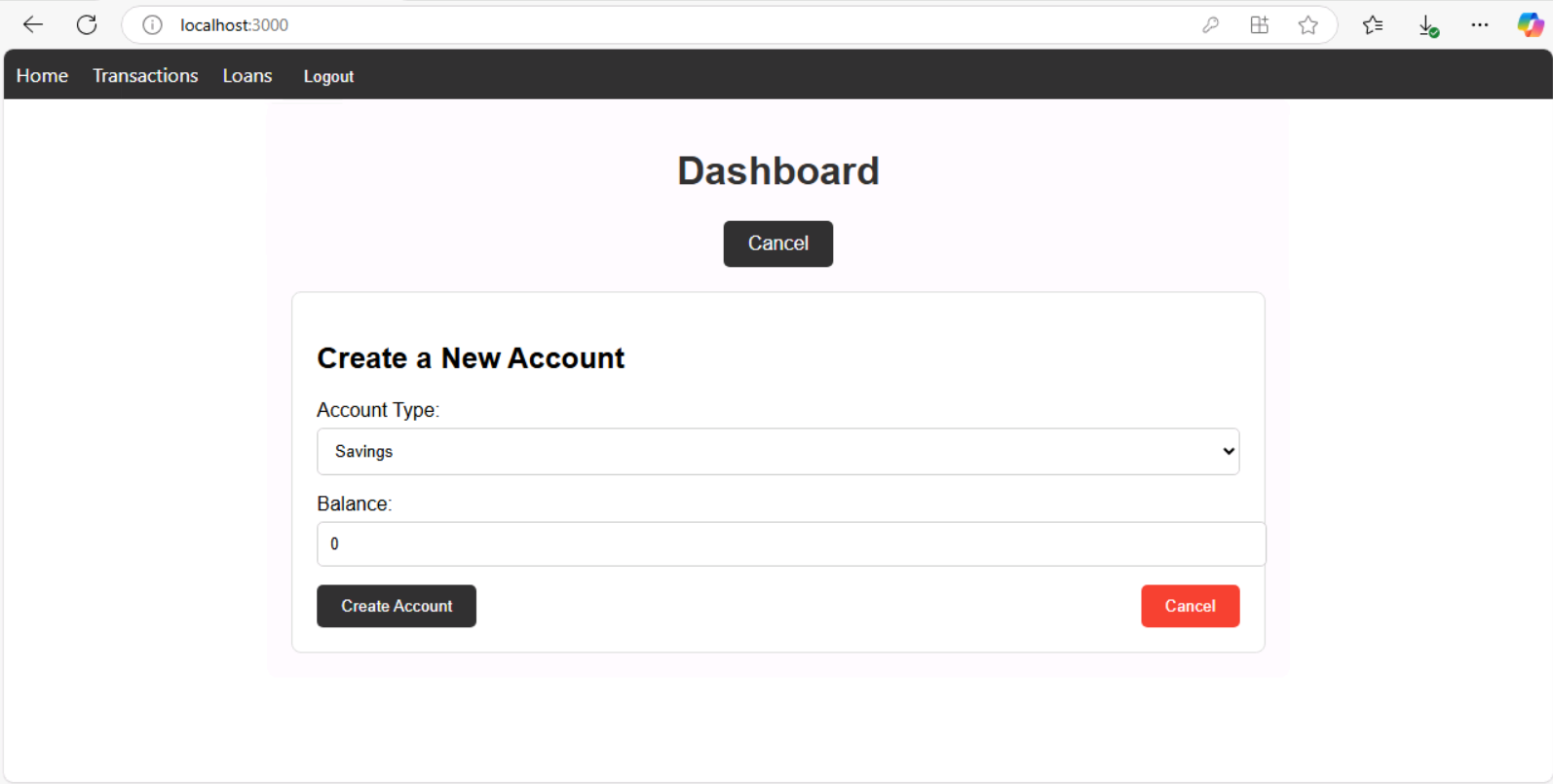


Accounts list table.

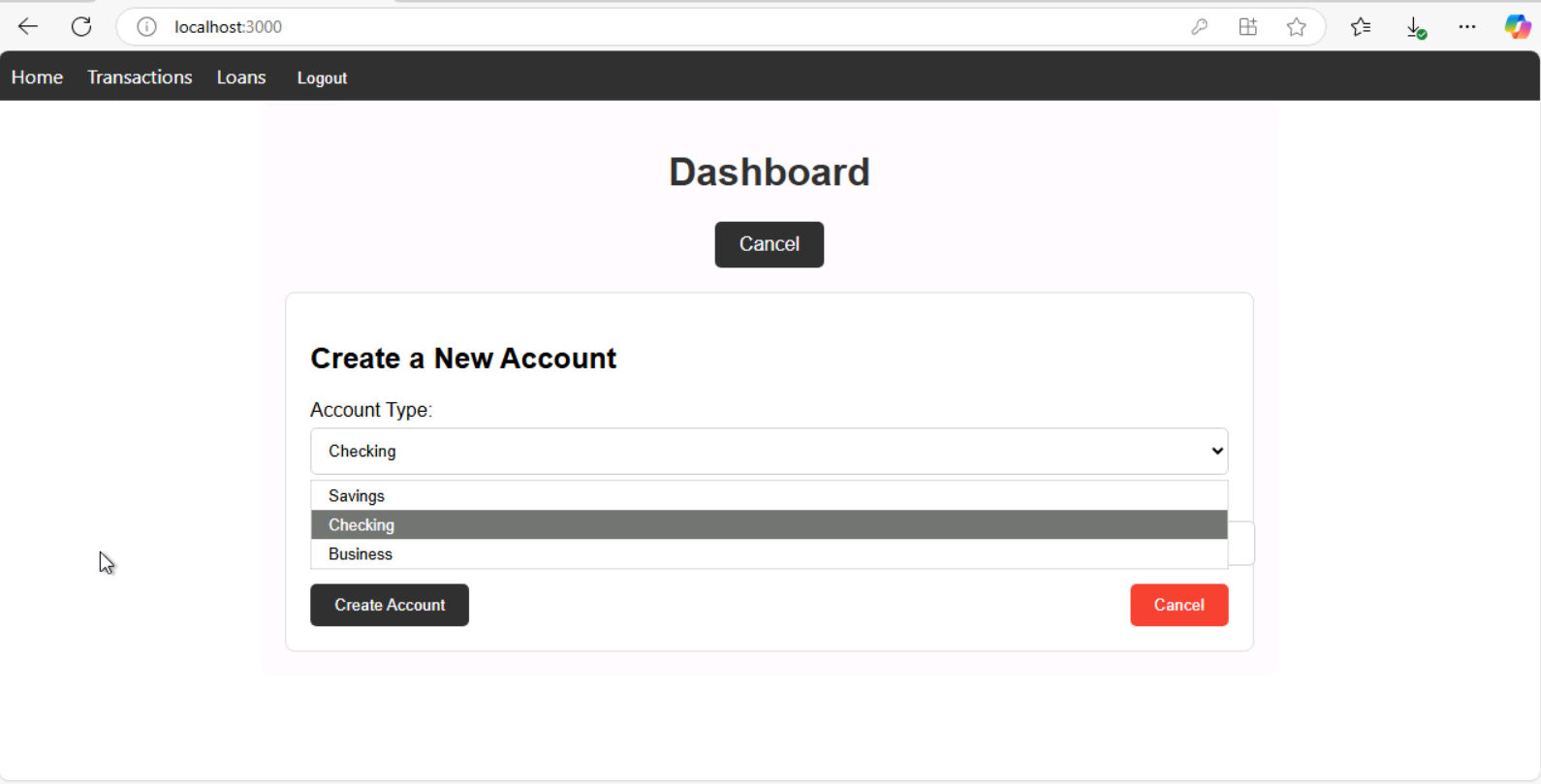
Details:

Backend fetches accounts associated with the authenticated user. Displays account number, type, and balance.

* Create Account: Each user can create multiple accounts



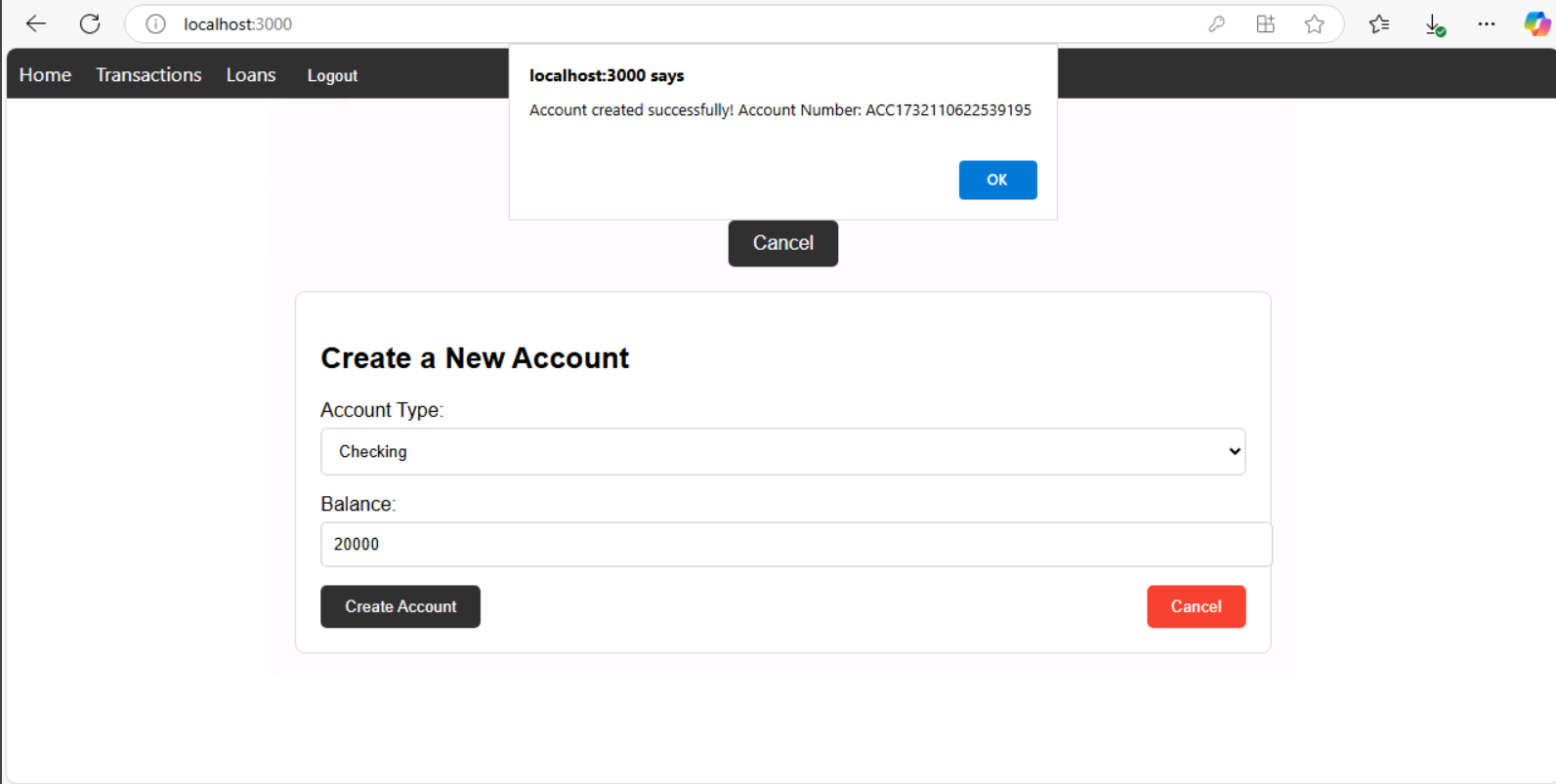
Account creation form.



Account creation form.

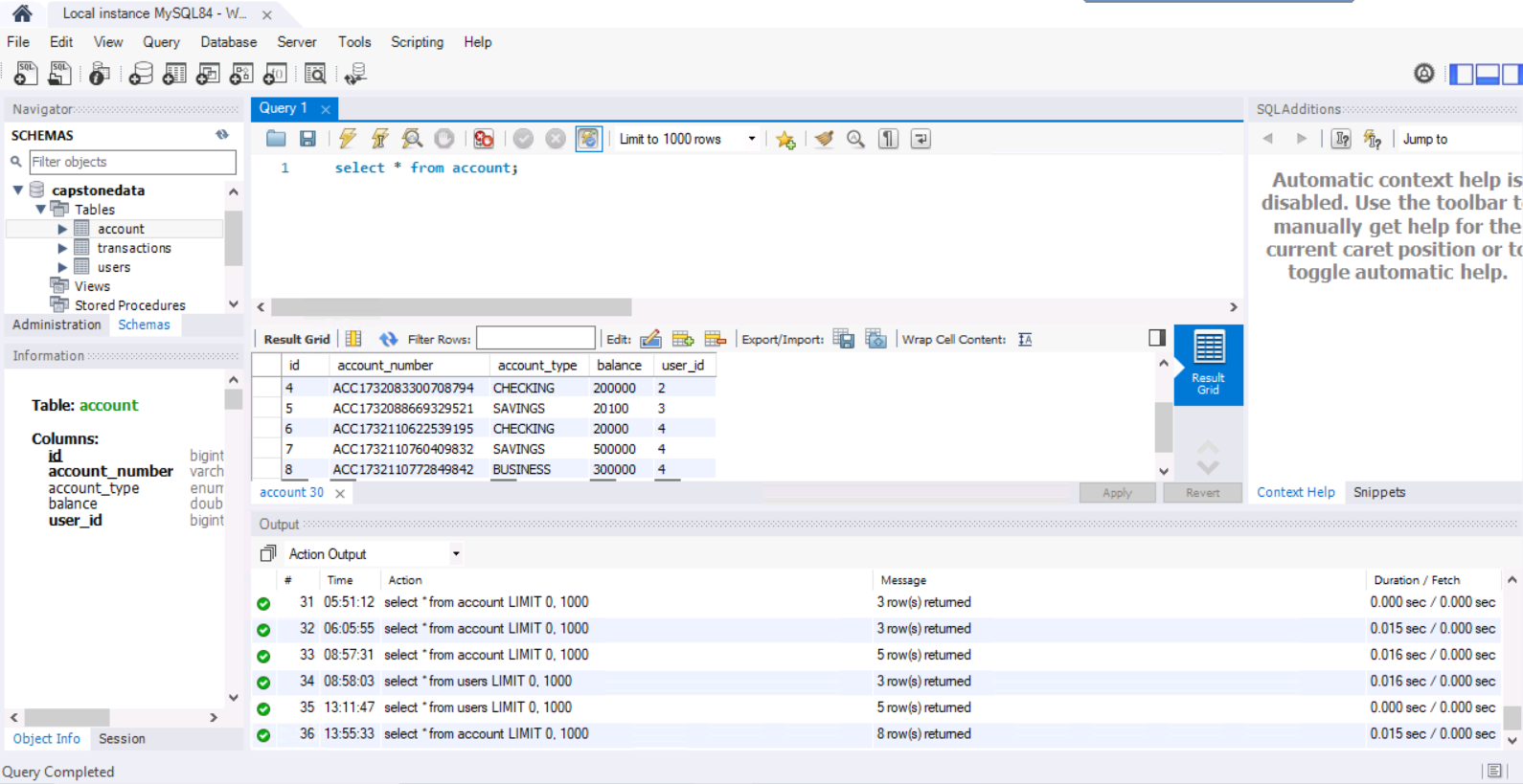
Details:

Input: Account type, Initial deposit



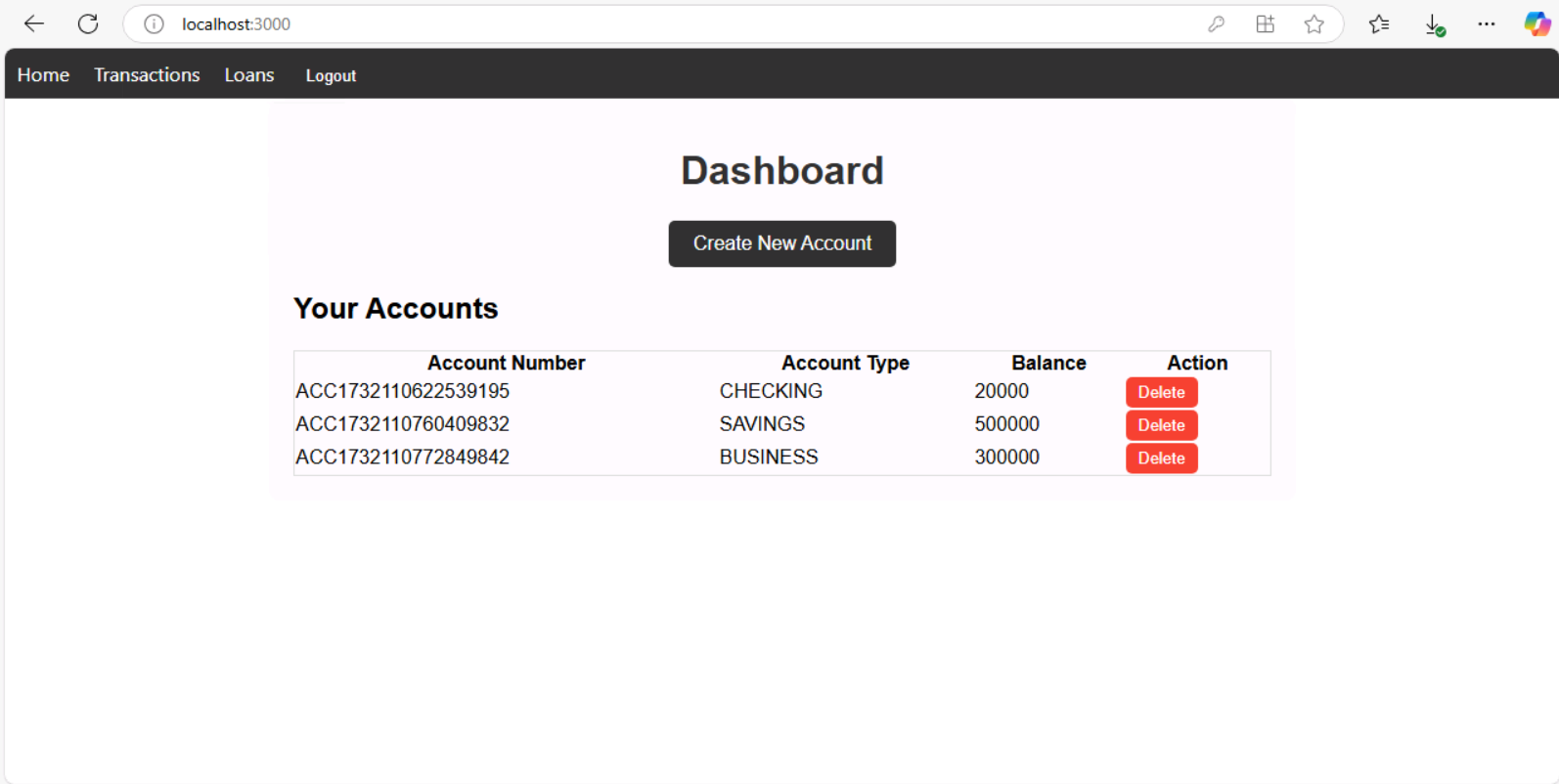
On Button Click.

Process: Save account in the database → Link to user.



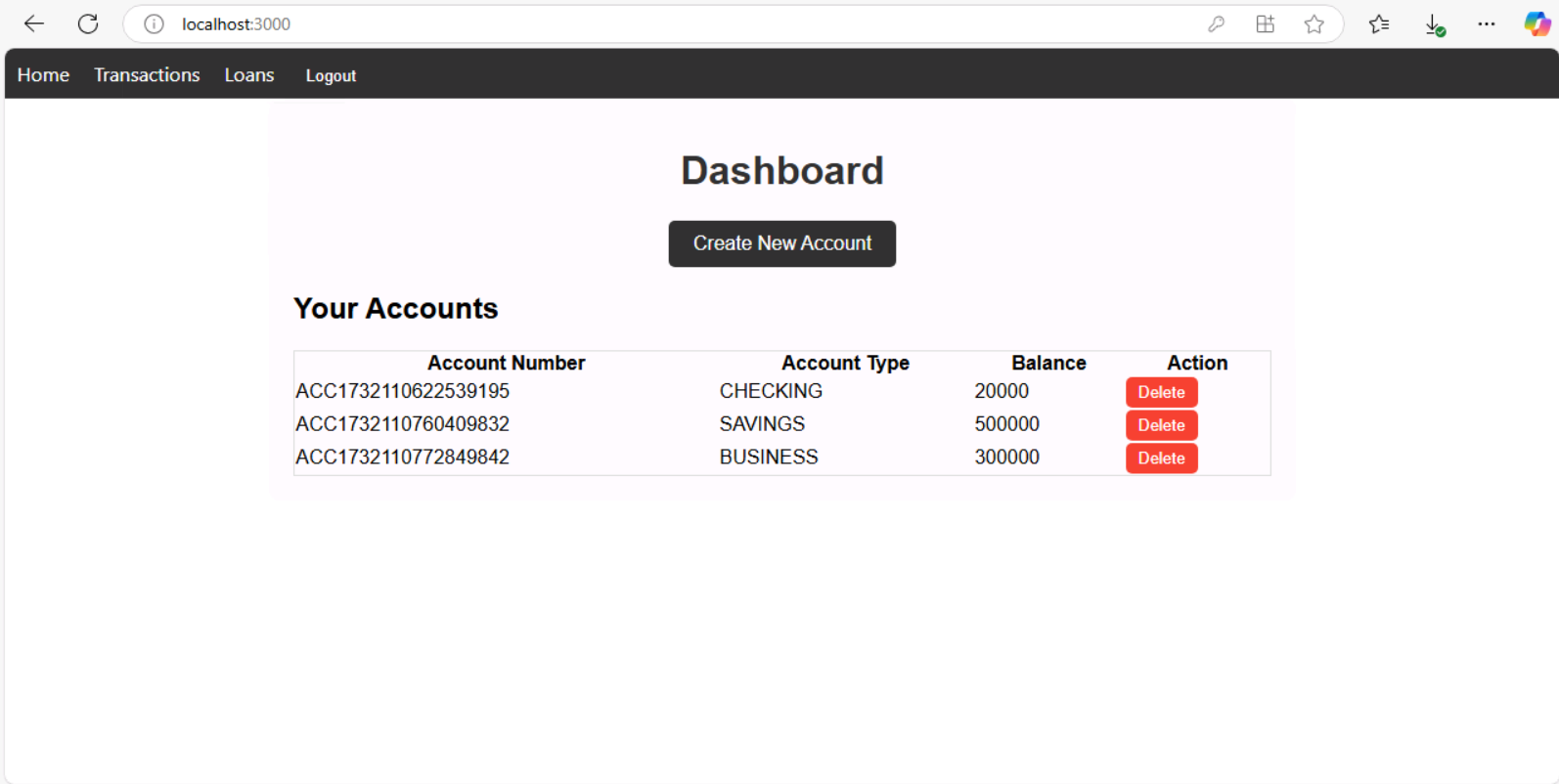
Output:

**Autogenerated** Account Number

Updated accounts table.

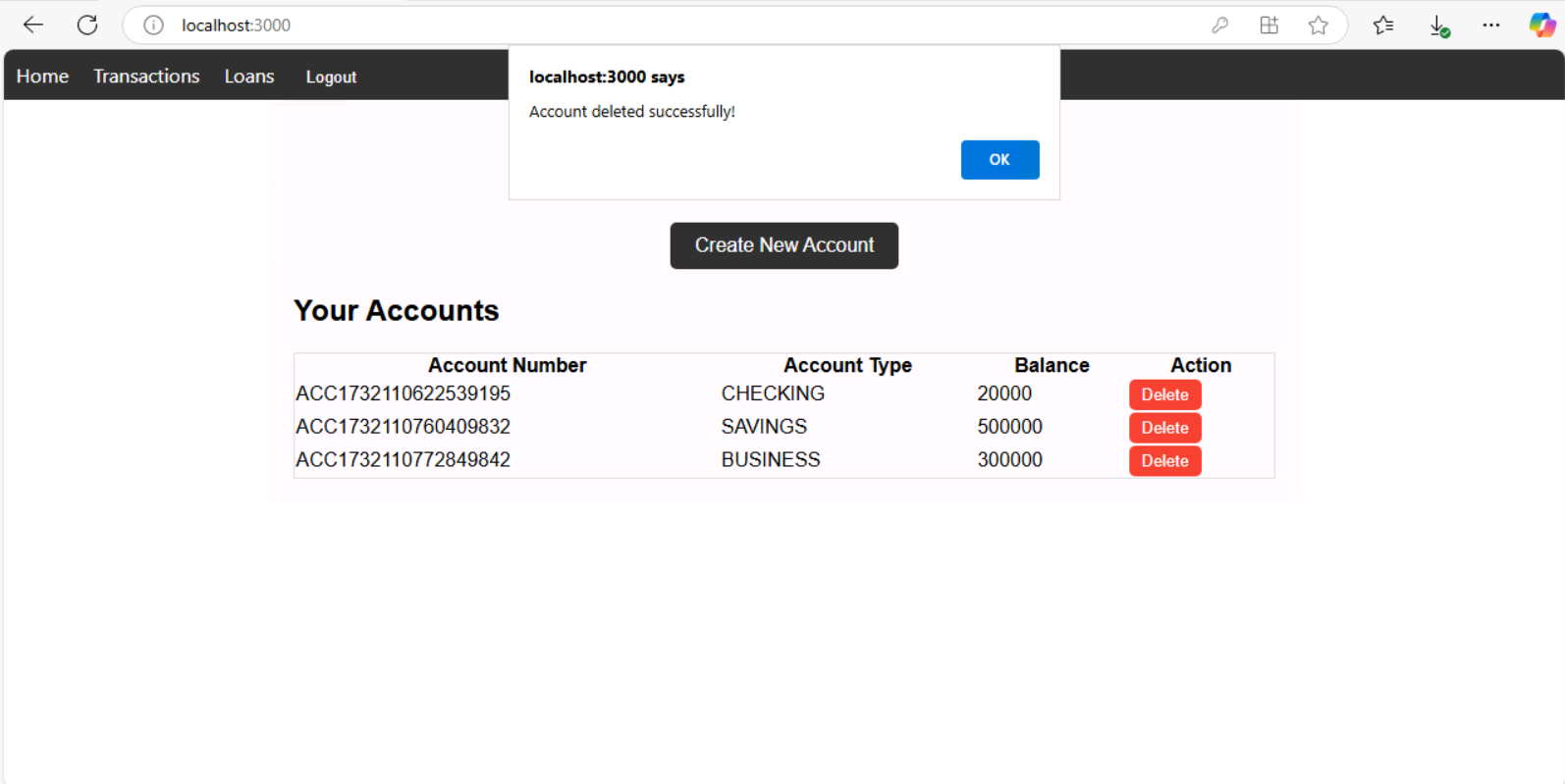
* Delete Account

Input:

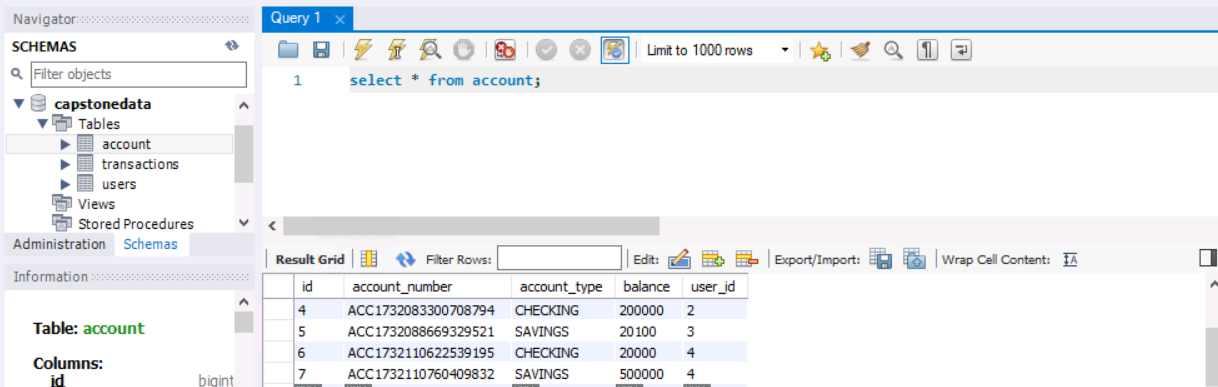


Click Delete.

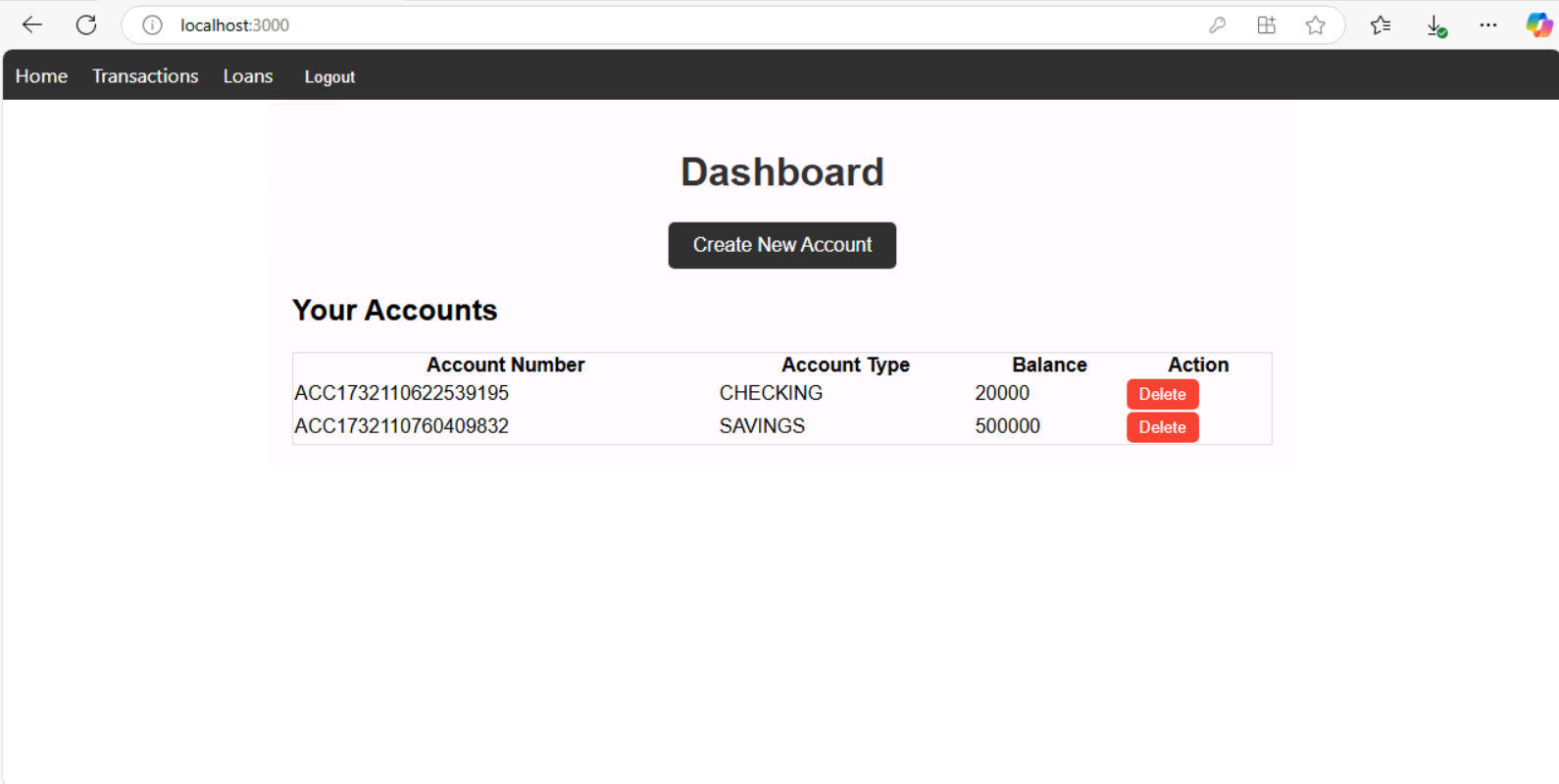
Process:



Alert.



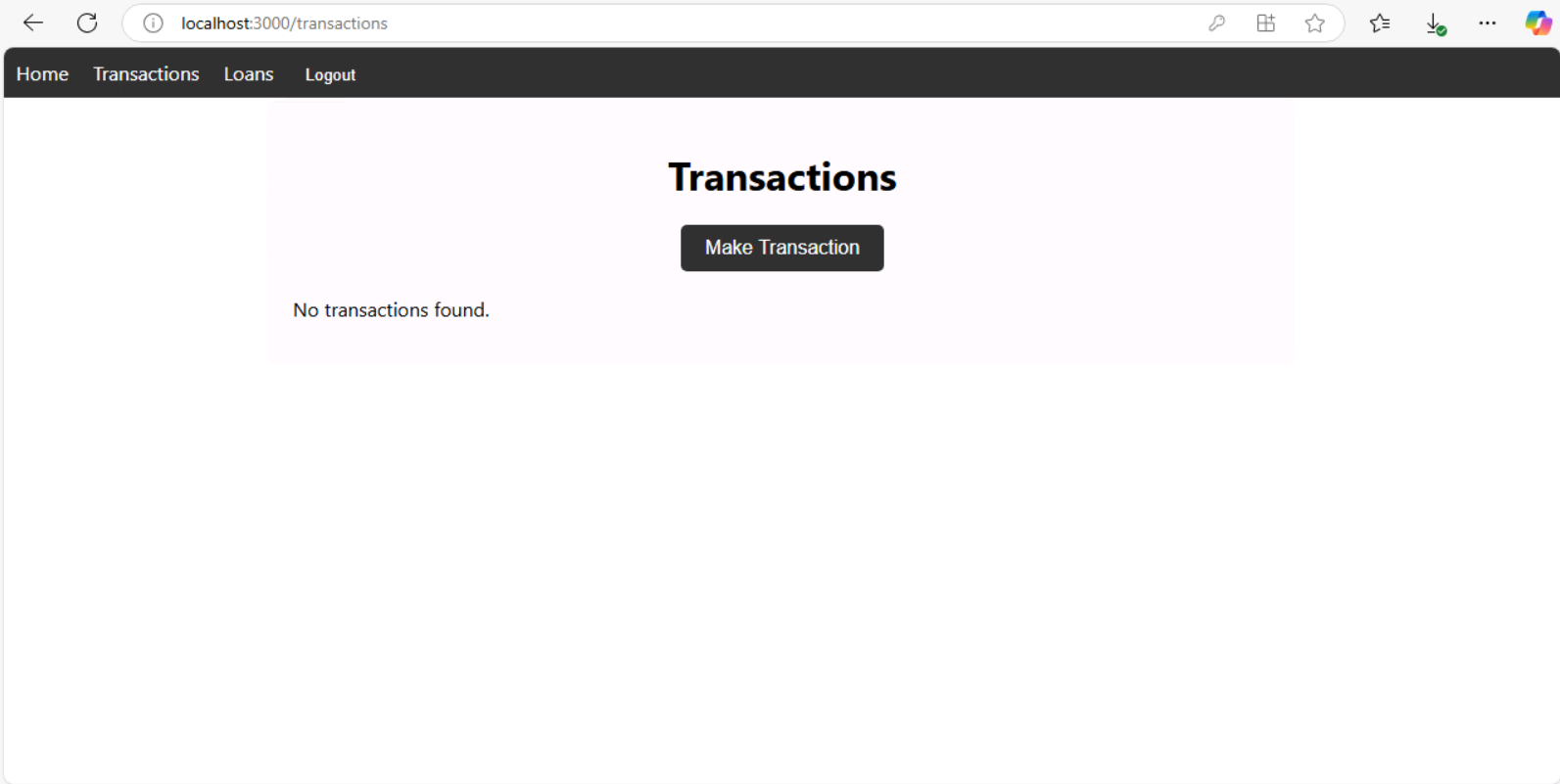
Output:



New list.

**2.3 Transaction Management**

**Landing Page**



* Transfer Funds

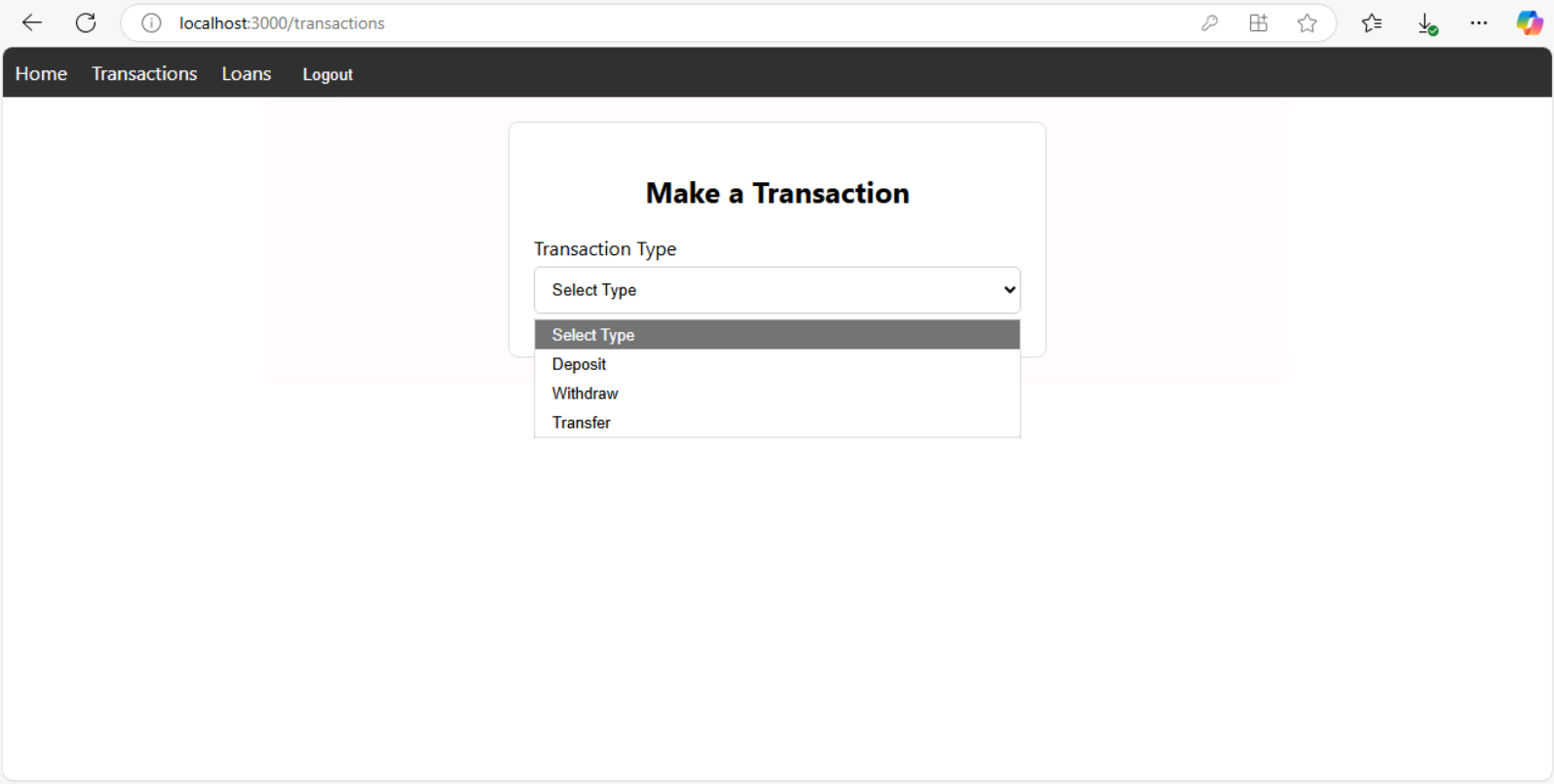
Details:

Input:

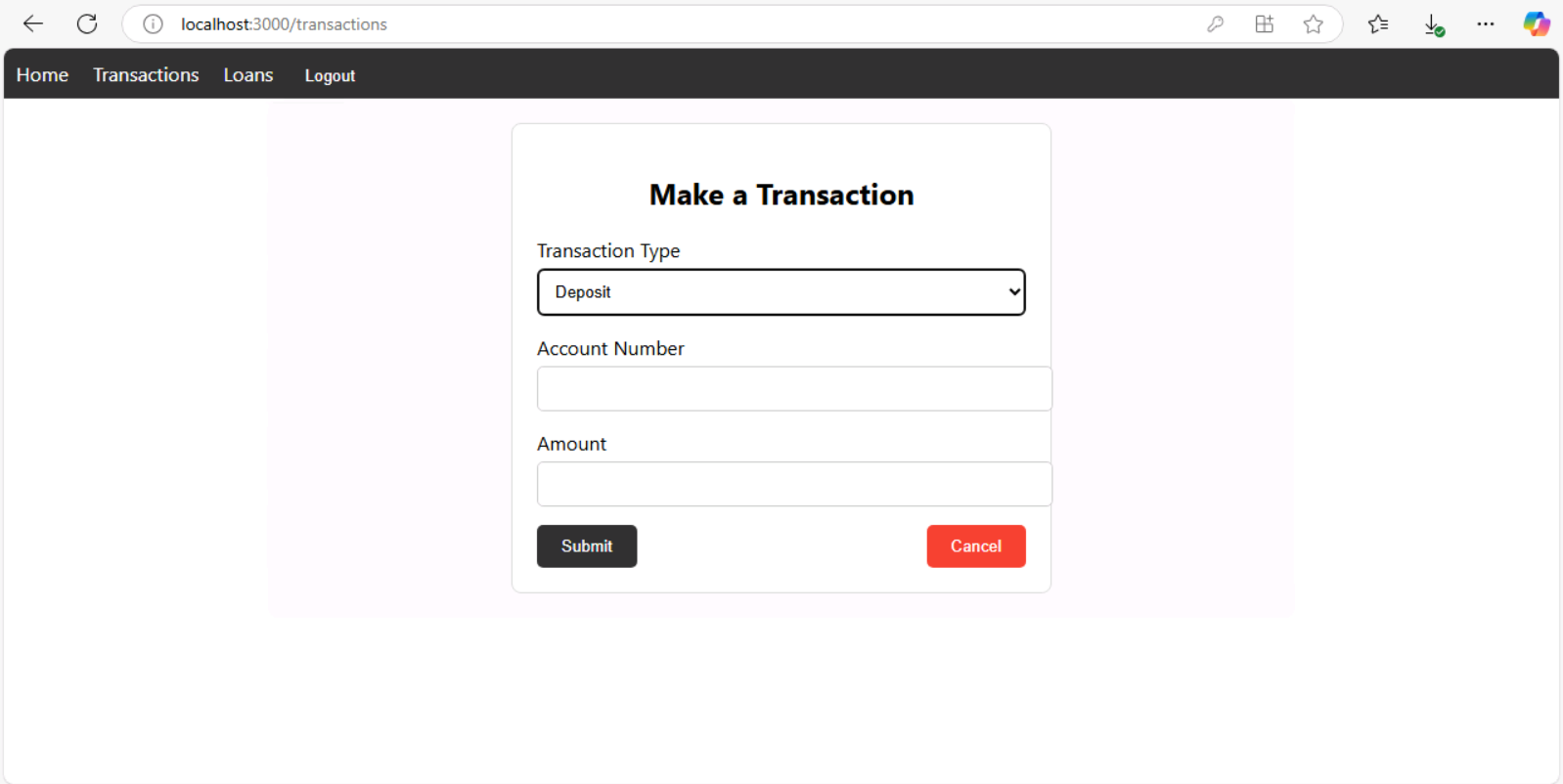
Deposit: Account, amount

Withdraw: Account, Email(for checking if it is someone else’s account), amount

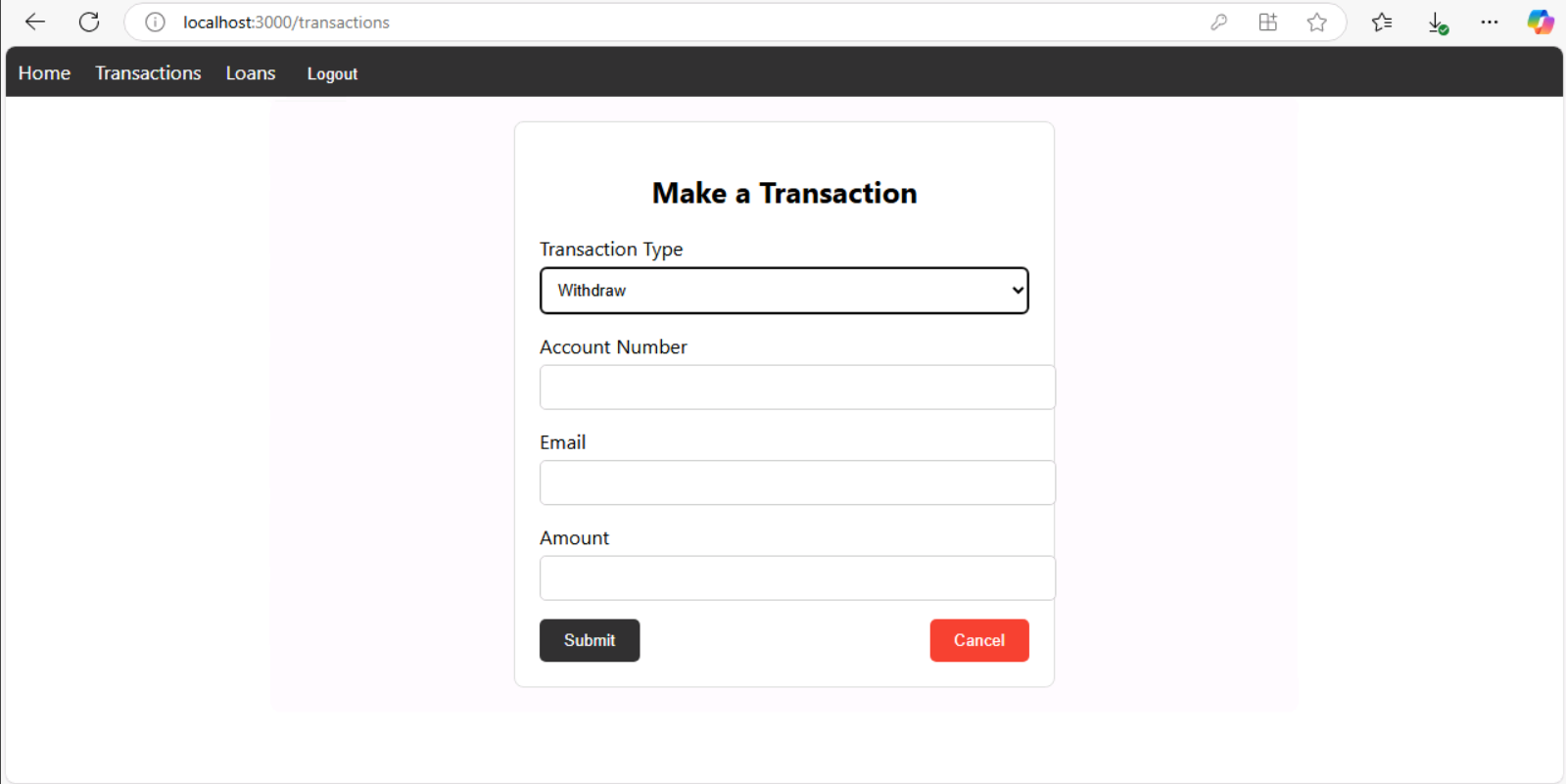
Transfer: Source account, destination account, amount.



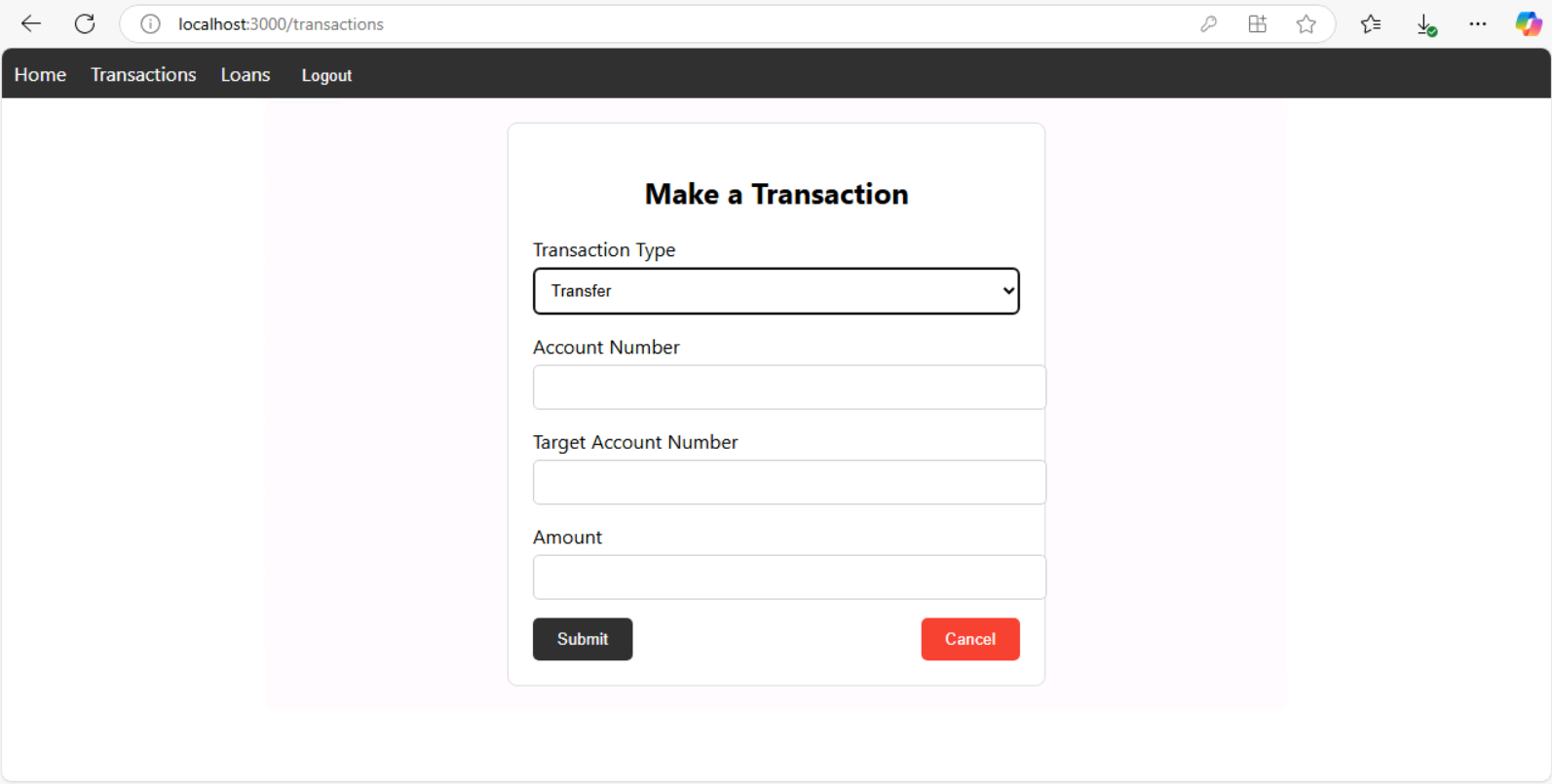
Transaction form.



Deposit funds form.



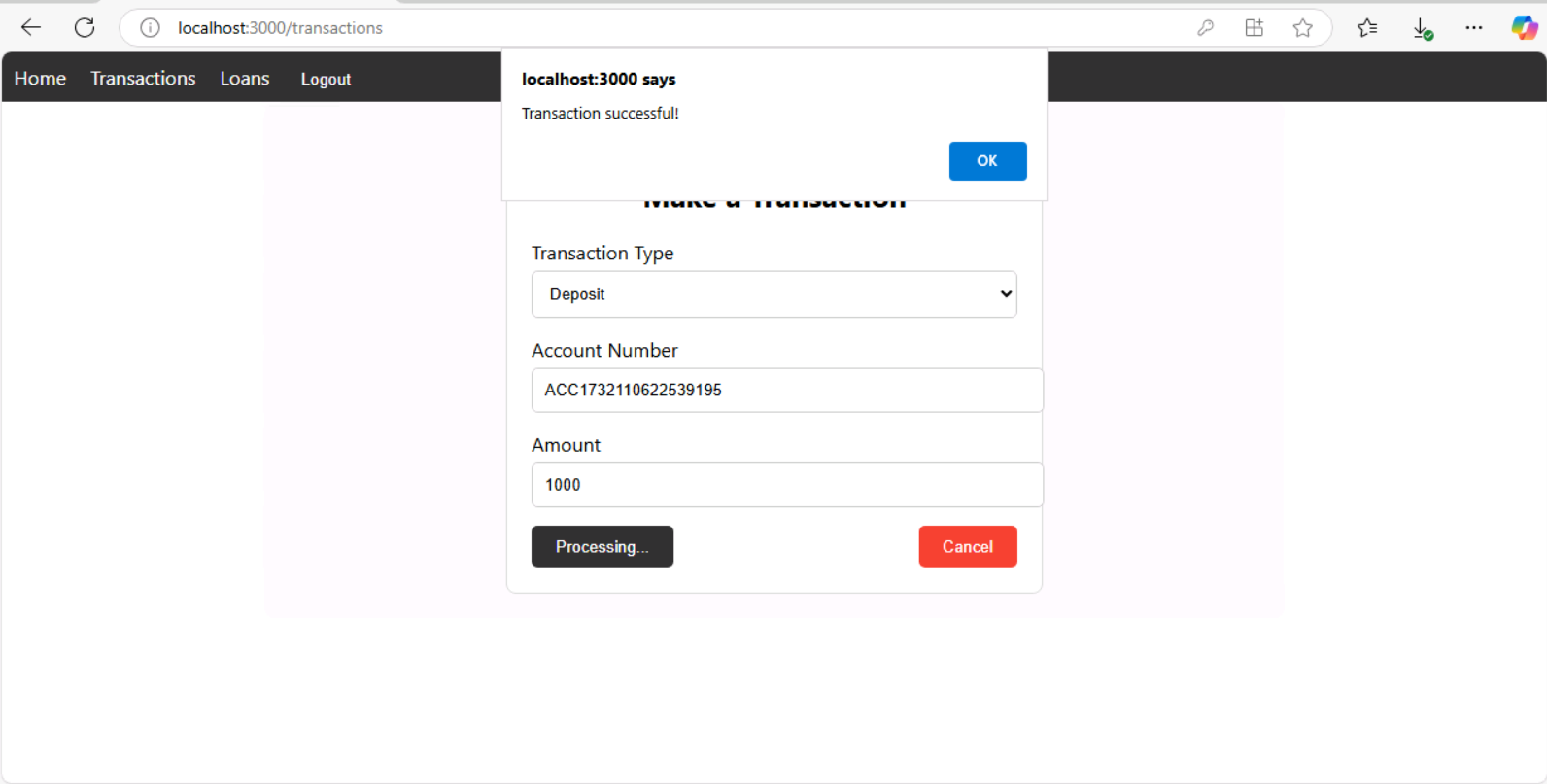
Withdraw funds form.

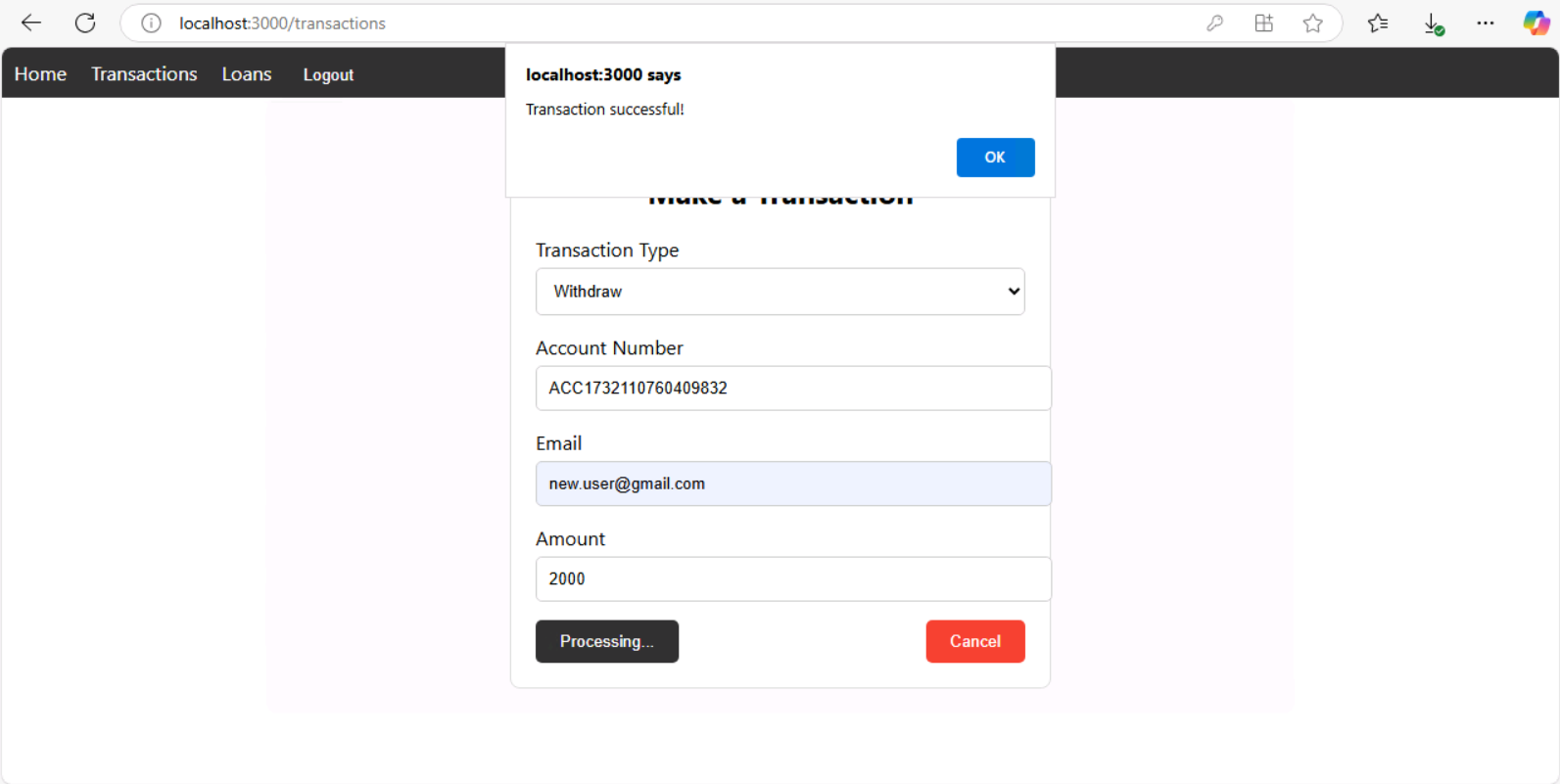


Transfer funds form.

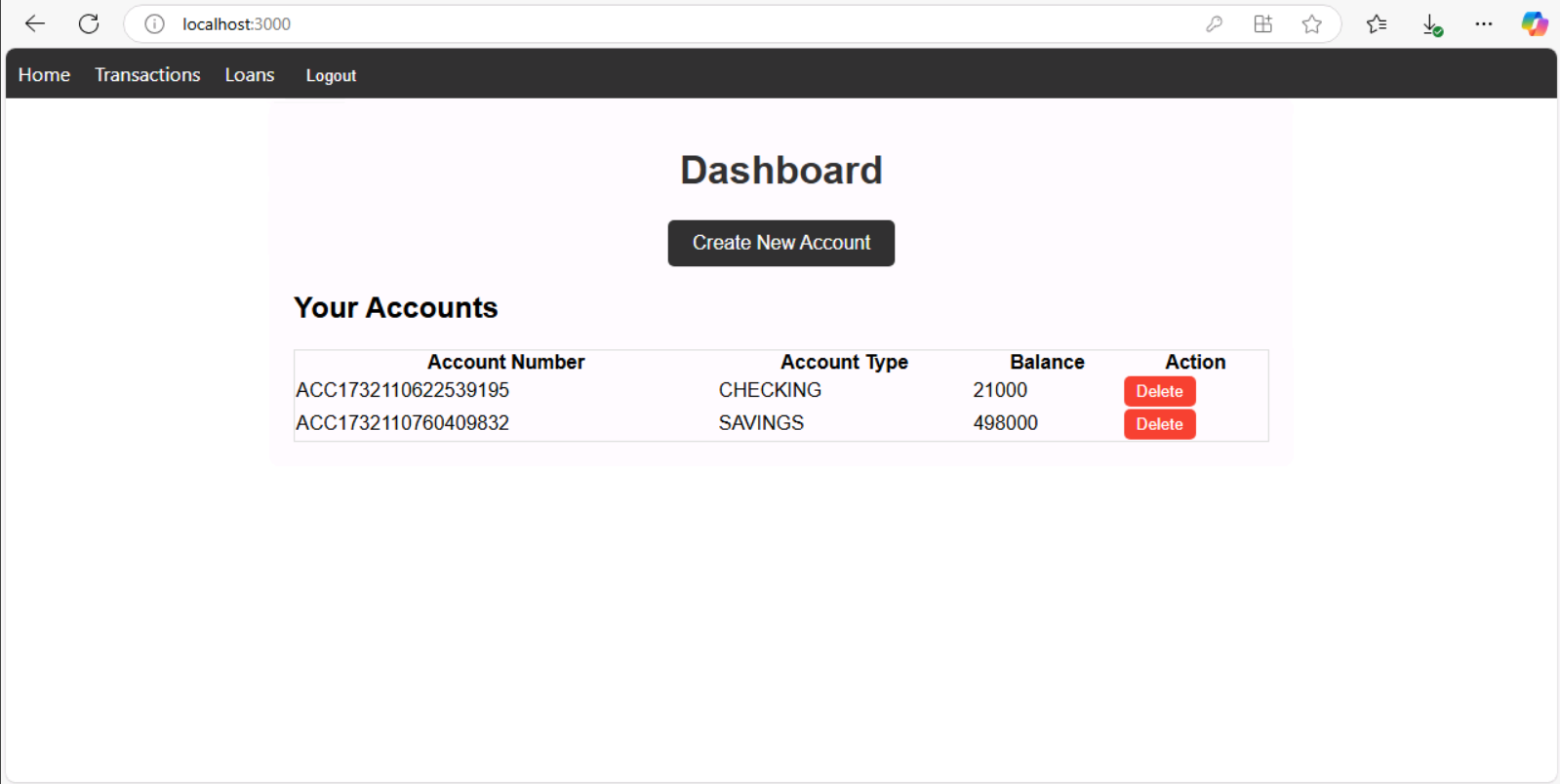
Process: Validate accounts and balance → Save transaction in the database.

Output: Updated balances and transaction list.

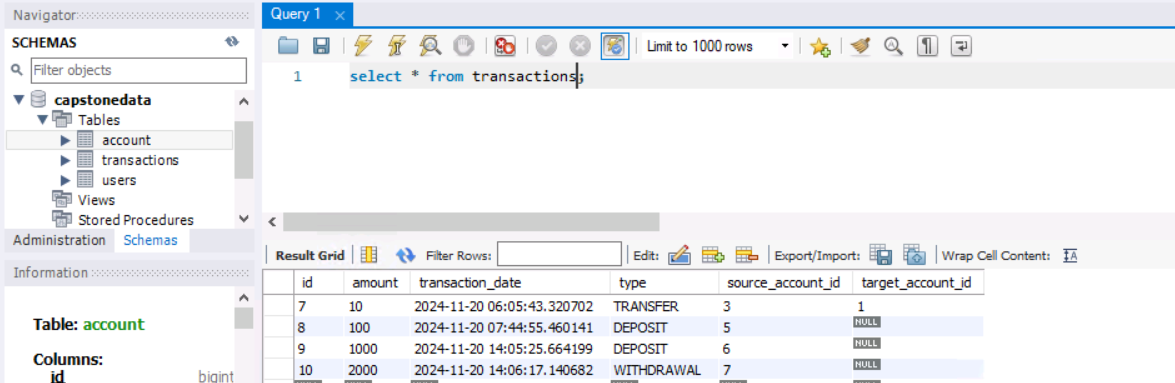




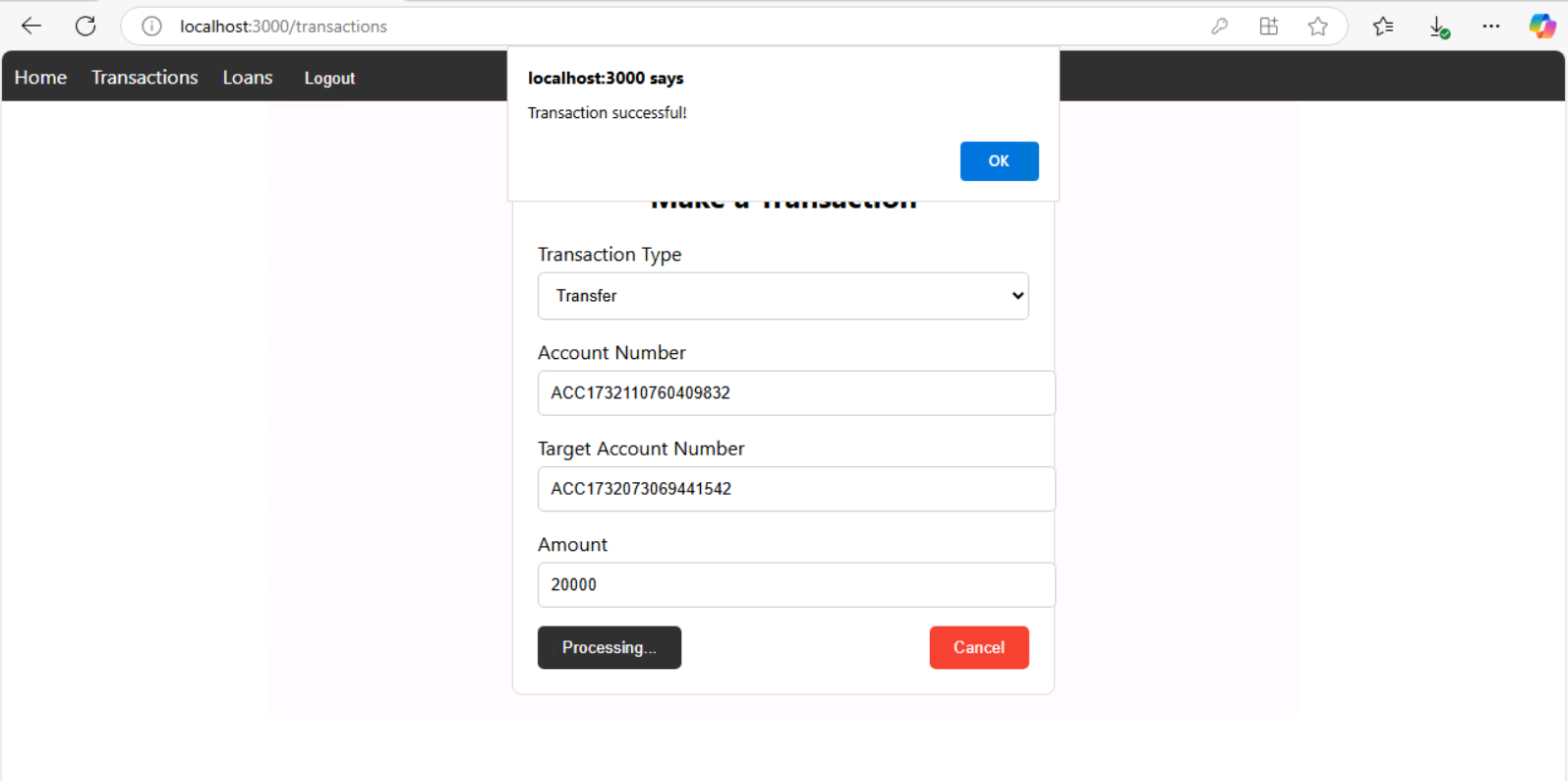
After deposit and Withdrawal.

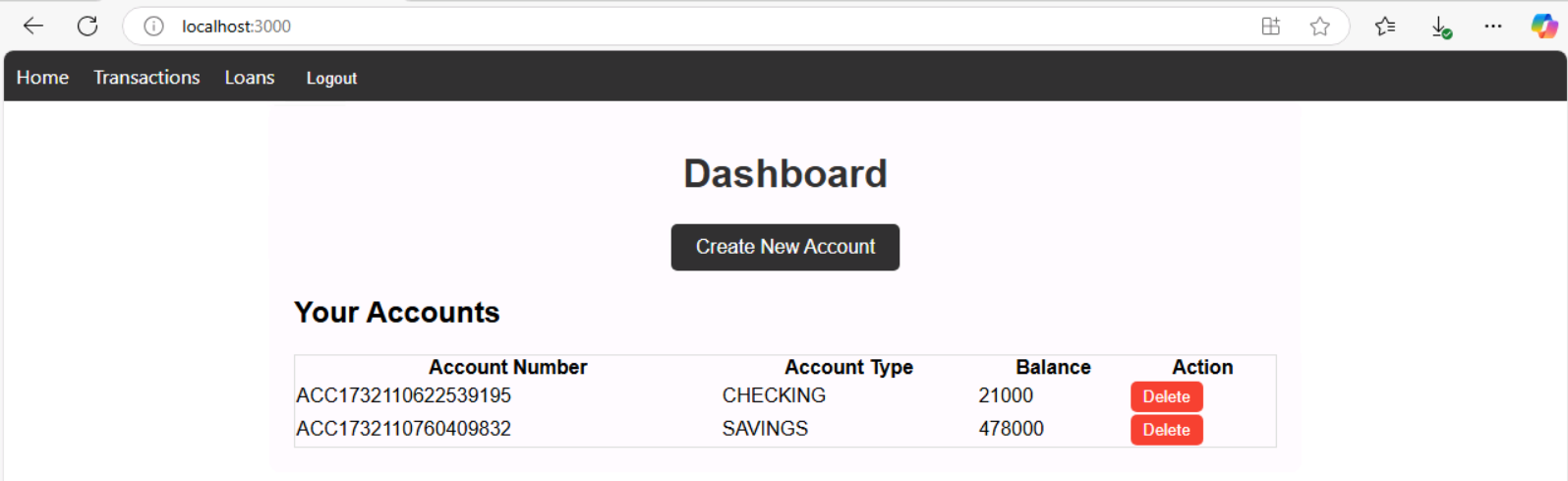


Updated balance reflects on dashboard rereflects

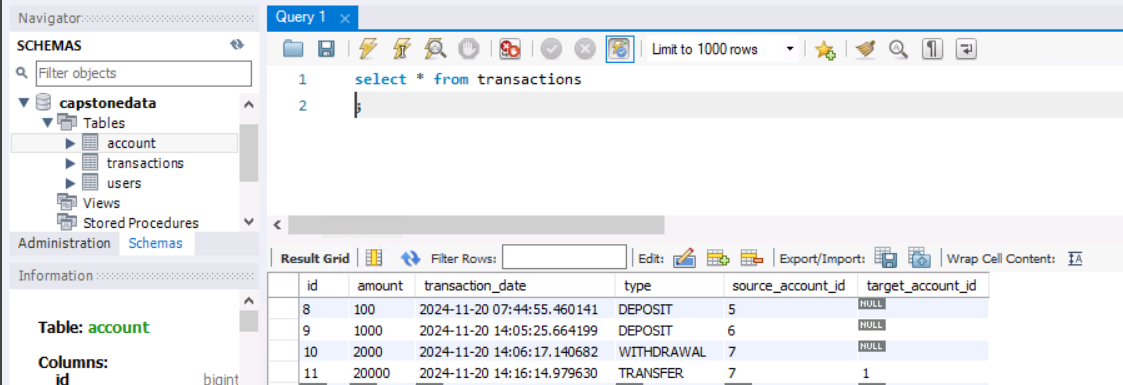


After Transfer.

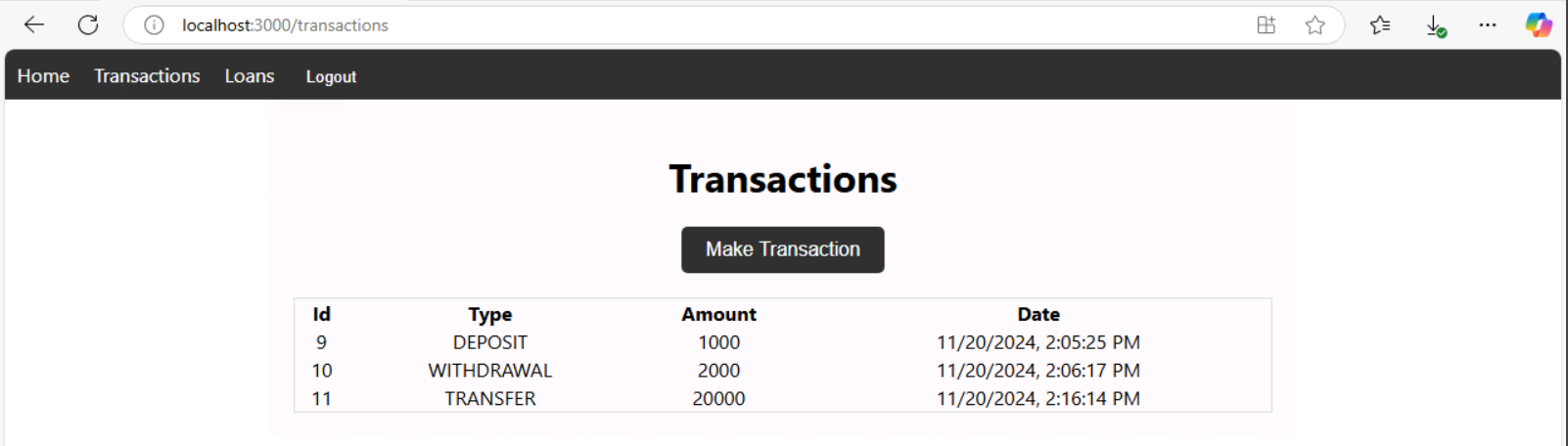




Updated balance after transfer



* View Transactions

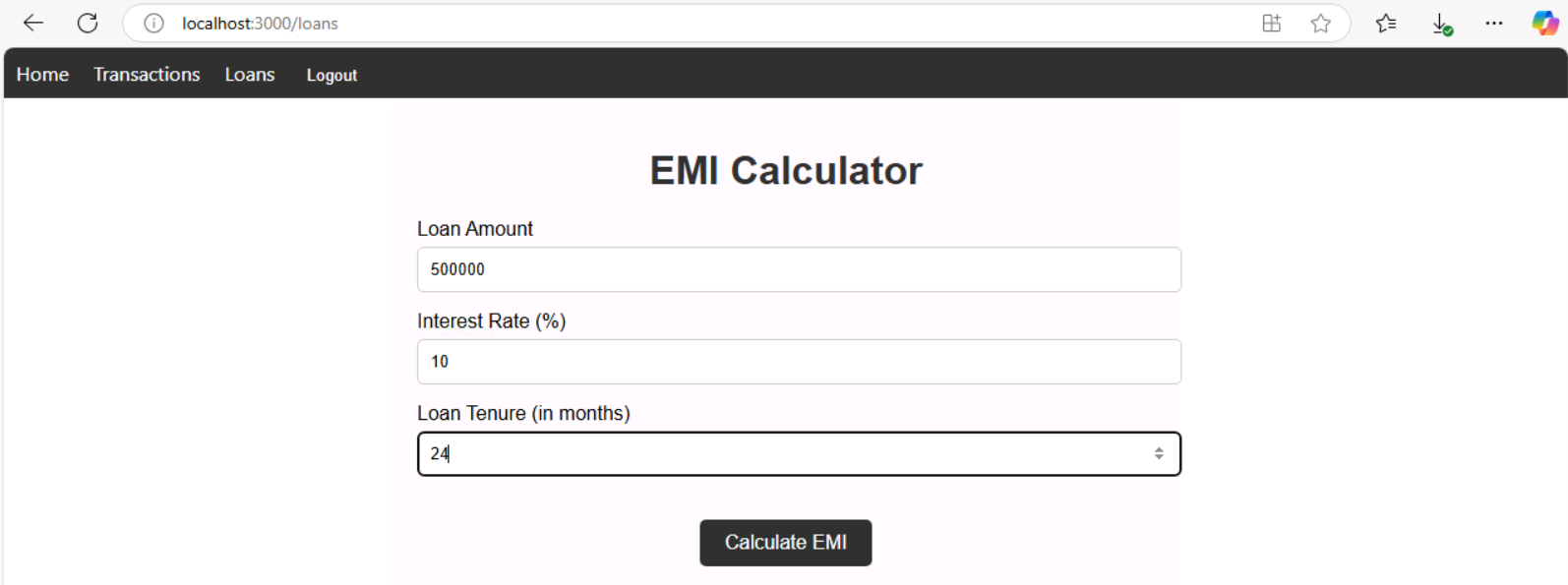


Transactions history table.

Details: Backend fetches transaction records. Displays details such as date, source, destination, and amount.

**2.4 Loan Planning (EMI Calculator)**

* Calculate EMI



EMI calculator UI.

Details:

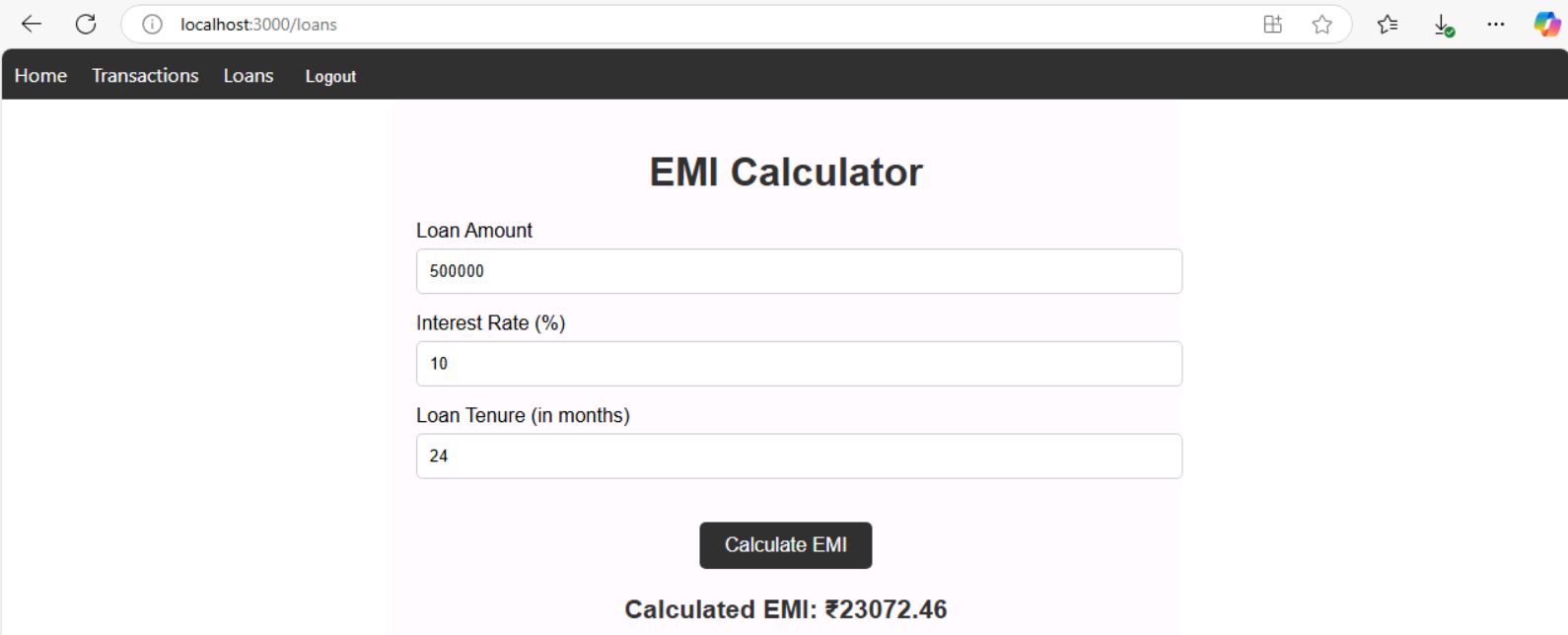
Input: Loan amount, interest rate, tenure.

Process: Calculate EMI using the formula.

EMI= (1+R) N −1 P⋅R⋅(1+R) N

(where 𝑃 P is principal, 𝑅 R is monthly interest, and 𝑁 N is tenure in months).

Output:

Display monthly EMI amount.

**3. Backend Endpoints**

| **Endpoint** | **HTTP Method** | **Description** |
| --- | --- | --- |
| /auth/register | POST | Registers a new user. |
| /auth/login | POST | Authenticates a user and issues a JWT. |
| /accounts | POST | Creates a new account. |
| /accounts/user | GET | Fetches accounts linked to the authenticated user. |
| /accounts/{id} | DELETE | Deletes an account by ID. |
| /transactions/deposit | POST | Processes a deposit transaction. |
| /transactions/withdraw | POST | Processes a withdrawal transaction. |
| /transactions/transfer | POST | Processes a transfer transaction. |
| /transactions/user | GET | Fetches transaction history for a user. |

**4. Frontend URLs**

| **URL** | **Description** |
| --- | --- |
| /login | User login page. |
| /register | User registration page. |
| /dashboard | User dashboard showing account details. |
| /transactions | Displays transaction history and options. |
| /loans | EMI Calculator for loan planning. |

**5. Database Schema**

**Database Schema Documentation**

**5.1. Table: users**

| **Column Name** | **Data Type** | **Attributes** | **Description** |
| --- | --- | --- | --- |
| id | bigint | AUTO\_INCREMENT, PK | Primary key, unique identifier for users. |
| email | varchar(255) | UNIQUE | Email address of the user (used for authentication). |
| password | varchar(255) | NOT NULL | Encrypted password for the user. |
| username | varchar(255) | UNIQUE | Unique username for the user. |

**Primary Key**: Id

**5.2. Table: account**

| **Column Name** | **Data Type** | **Attributes** | **Description** |
| --- | --- | --- | --- |
| id | bigint | AUTO\_INCREMENT, PK | Primary key, unique identifier for accounts. |
| account\_number | varchar(255) | UNIQUE | Unique identifier for each account. |
| account\_type | enum('BUSINESS','CHECKING','SAVINGS') | NOT NULL | Type of account (business, checking, or savings). |
| balance | double | NOT NULL | Current balance in the account. |
| user\_id | bigint | FK | Foreign key referencing users(id). |

**Primary Key**: id

**Foreign Key**: user\_id → users(id)

**Relationships**:

* **One-to-Many**: A user can have multiple accounts.

**5.3. Table: transactions**

| **Column Name** | **Data Type** | **Attributes** | **Description** |
| --- | --- | --- | --- |
| id | bigint | AUTO\_INCREMENT, PK | Primary key, unique identifier for transactions. |
| amount | double | NOT NULL | Amount involved in the transaction. |
| transaction\_date | datetime(6) | NOT NULL | Date and time when the transaction occurred. |
| type | varchar(255) | NOT NULL | Type of transaction (e.g., "deposit," "withdrawal," "transfer"). |
| source\_account\_id | bigint | FK | Foreign key referencing account(id) for the source account. |
| target\_account\_id | bigint | FK | Foreign key referencing account(id) for the target account (if applicable). |

**Primary Key**: id

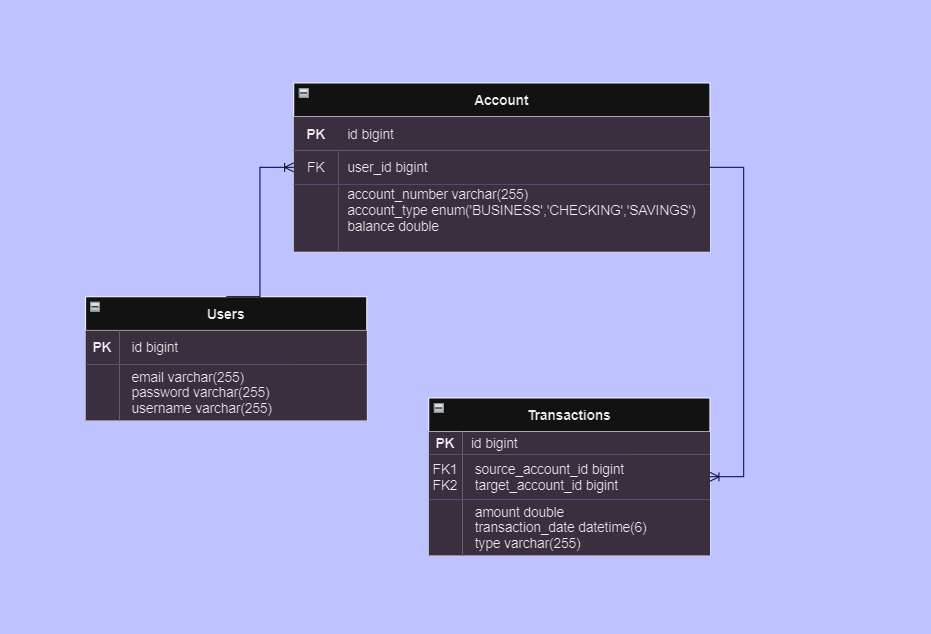
**Foreign Keys**:

* source\_account\_id → account(id)
* target\_account\_id → account(id)

**Relationships**:

* **One-to-Many**: An account can have multiple transactions.
* **Self-Referencing**: Source and target accounts can belong to the same or different users.

**6. ER-diagram**



**7. Tools and Technologies**

* **Backend**: Spring Boot (Java), JWT for authentication.
* **Frontend**: React.js for responsive UI.
* **Database**: MySQL for persistent data storage.
* **Security**: BCrypt for password hashing, JWT for secure token management.

**8. Github (Link to code):** [silvyapatel/Capstone](https://github.com/silvyapatel/Capstone)