**REPORT**

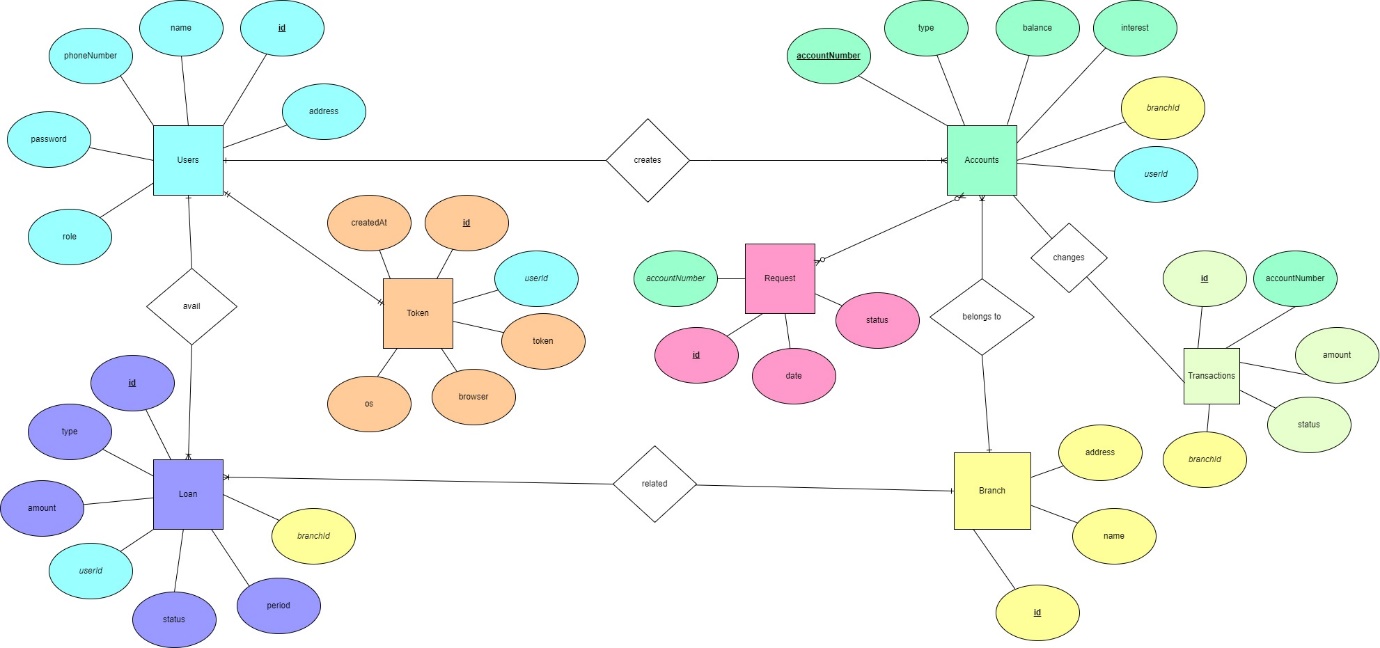
**BANK MANAGEMENT SYSTEM**

Mithesh A

1. **DATABASE**

The database consists of the following tables:

* Users
* Accounts
* Loan
* Transactions
* Branch
* Request
* Token



**Attributes of the tables:**

Users – **id**, name, phoneNumber, password, address, role

Accounts – **accountNumber**, type, balance, interest, branchId, userId, active

Loan – **id**, type, amount, userId, status, period, branchId

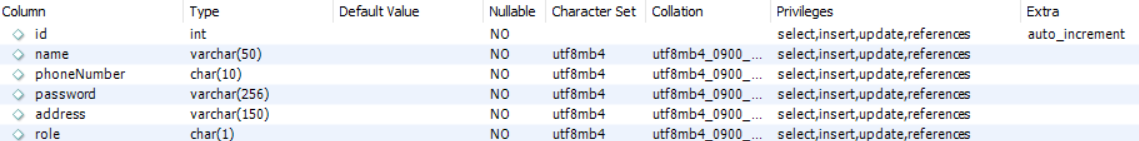
Transactions – **id**, accountNumber, amount, status, type, date

Branch – **id**, name, address

Request – **id**, accountNumber, date, status

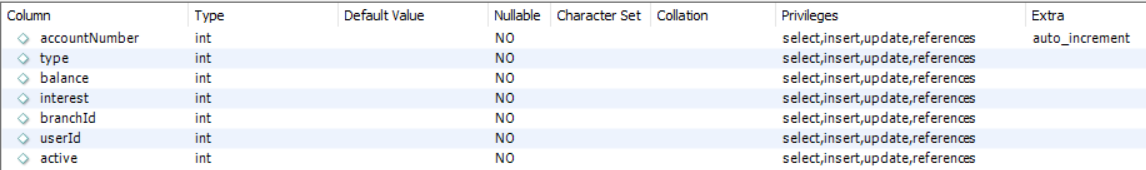
**Databases created in MySql**

**Users**

****

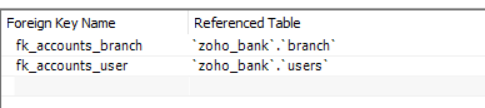
The role 0,1 corresponds to customer and manager respectively.

**Accounts**

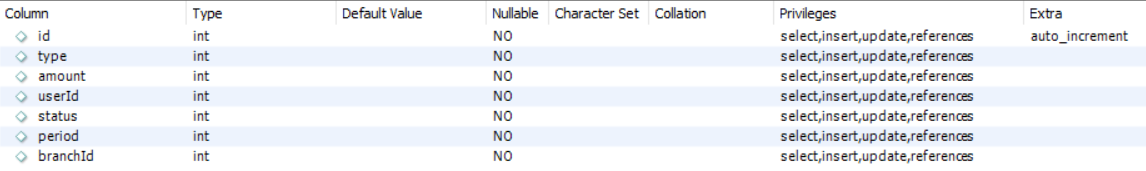


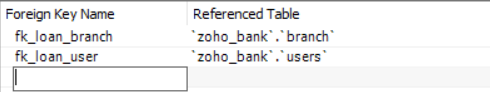
The accounts type can be 1,2,3 which means savings, business and loan accounts respectively.

The active status of the account can be 0,1,2 which means closed, active, requesting to close the account respectively.



**Loans**

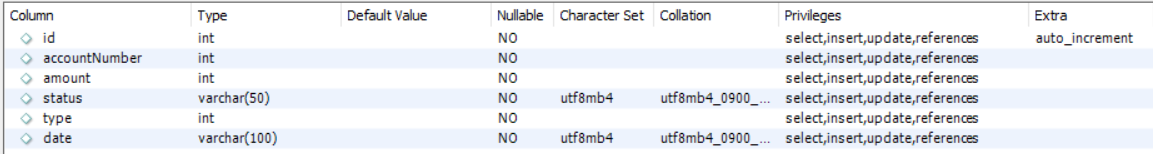


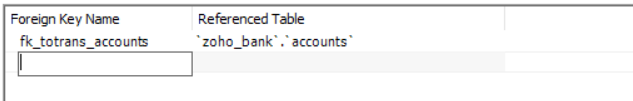


The loan type can be 0,1,2 which corresponds to home, vehicle and personal loan respectively.

The loan status can be 0,1,2 which means the customer requests for loan, the loan approved by the manager and the manager rejected the loan request respectively.

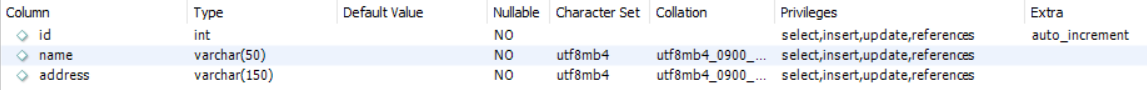
**Transactions**





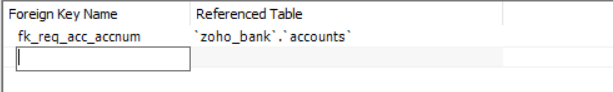
The transaction type can be 1,2,3 which means deposit, withdraw and loan transactions.

**Branch**



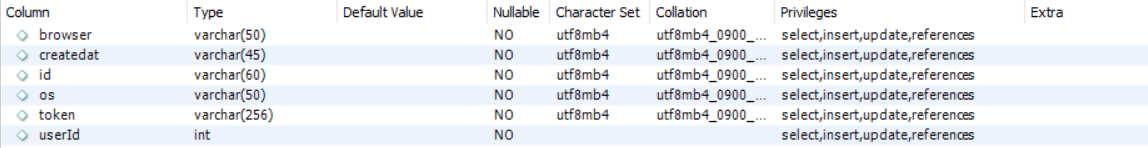
**Requests**

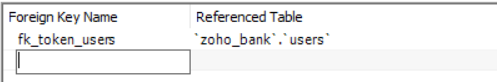




The status of request table can be 1,2,3 which means requesting to close the account, approved, rejected

**Token**

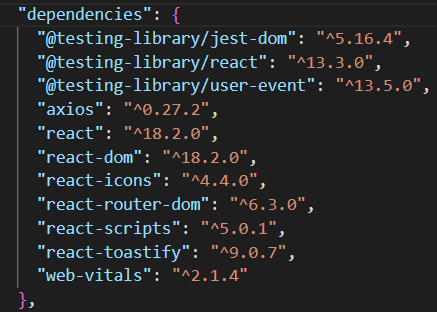
****

****

1. **FRONTEND, BACKEND**

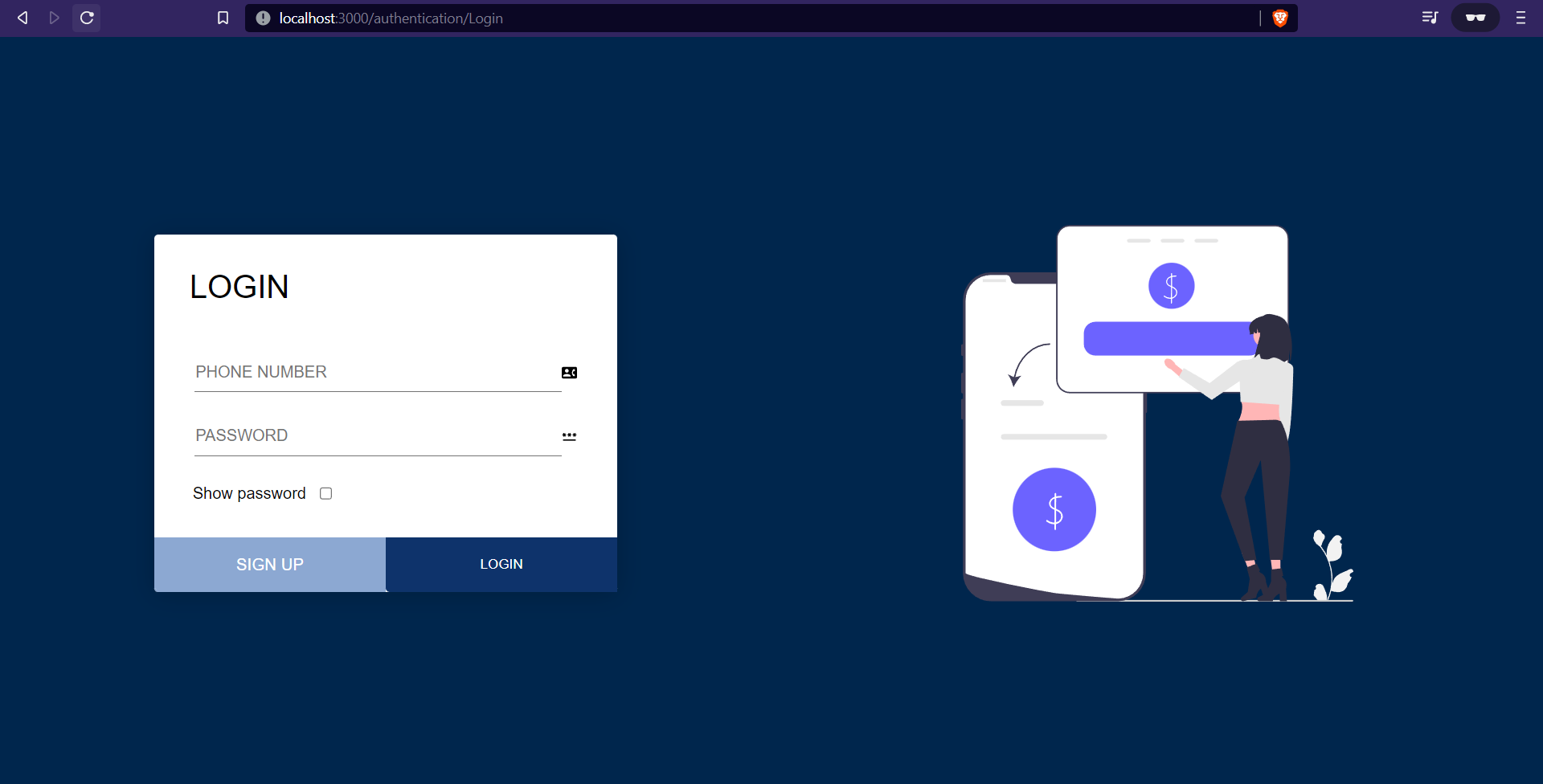
React is used to do the frontend part of this project. It promotes the development of reusable UI components that display dynamic data. React can power native apps using React Native and can render on the server using Node. The main reason I used react is that it uses virtual DOM (Document Object Model) which is a JavaScript object which is faster than regular DOM. Managers and customers uses two separate UI with different functionalities depending on the role. The login page, register page comes under authentication. The login page with phone number and password leads to the manager/customer UI depending on the type of role they play. The role field is added to the users table where 0 means customer, 1 means manager.

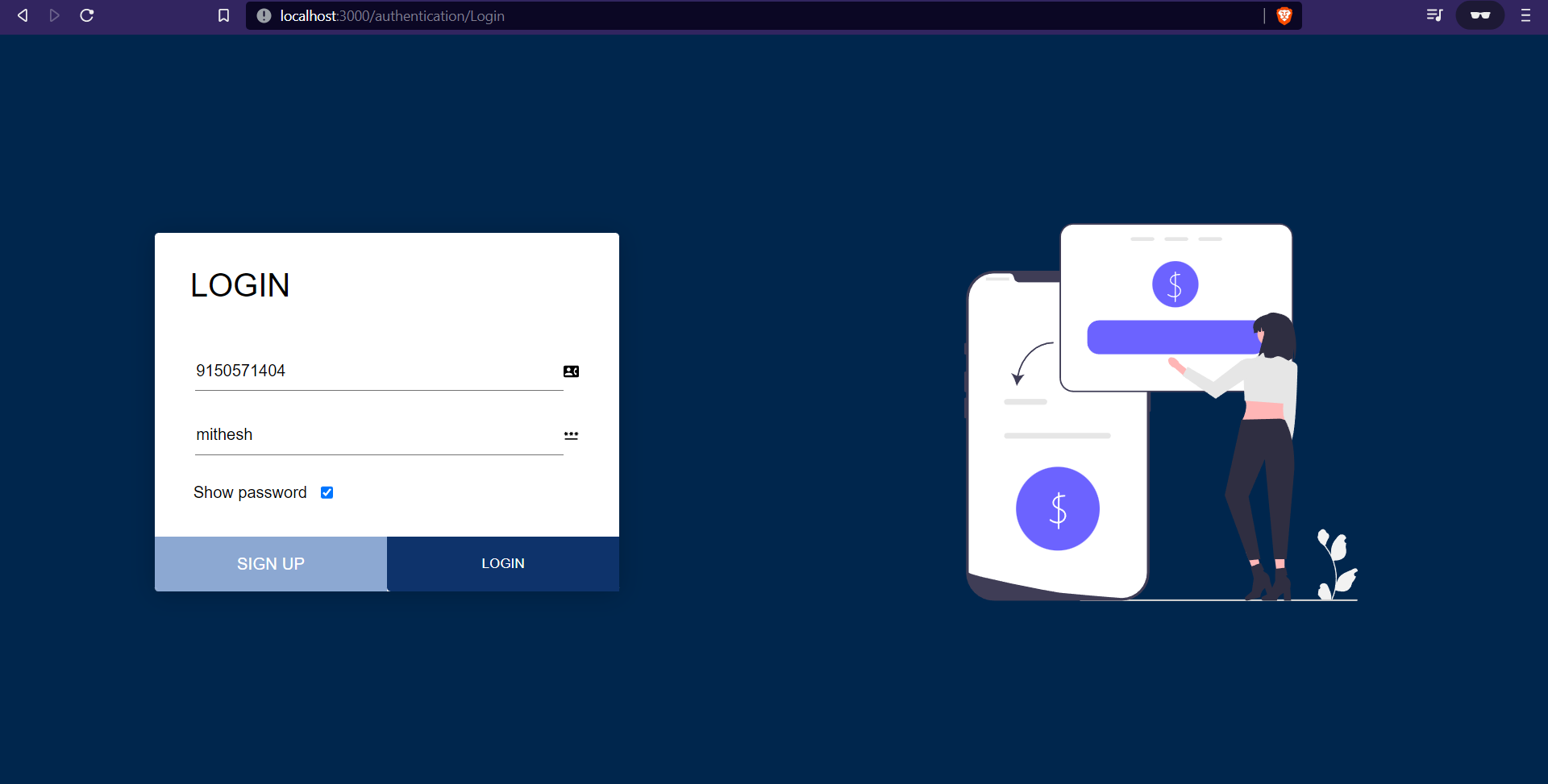
The dependencies used in the frontend are

****

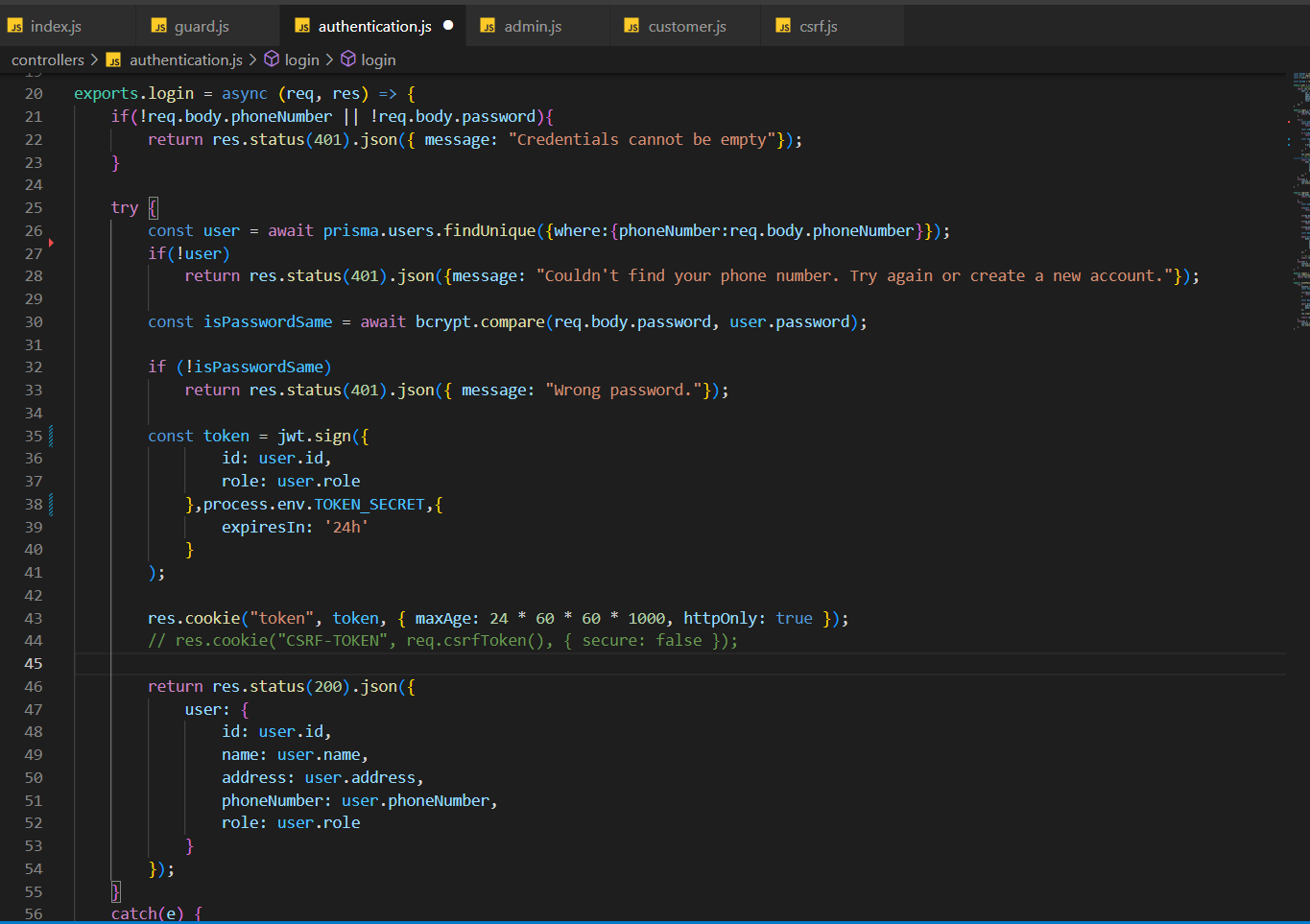
NodeJS is used for doing backend part of this project. Node JS is simpler than Java, easier for code writing and ideal for full-stack developers.

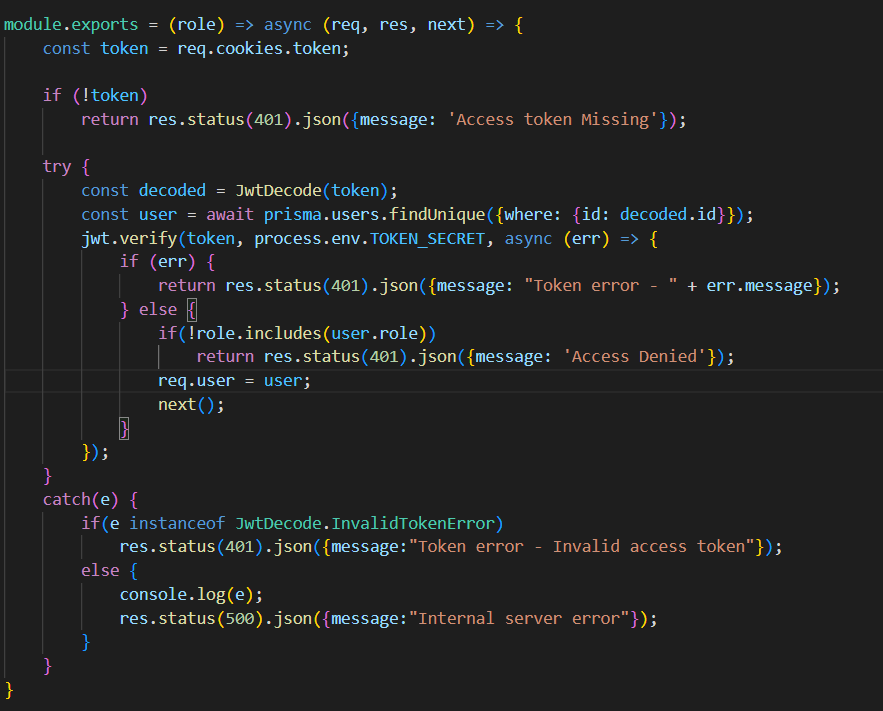
**LOGIN PAGE AND REGISTER PAGE**

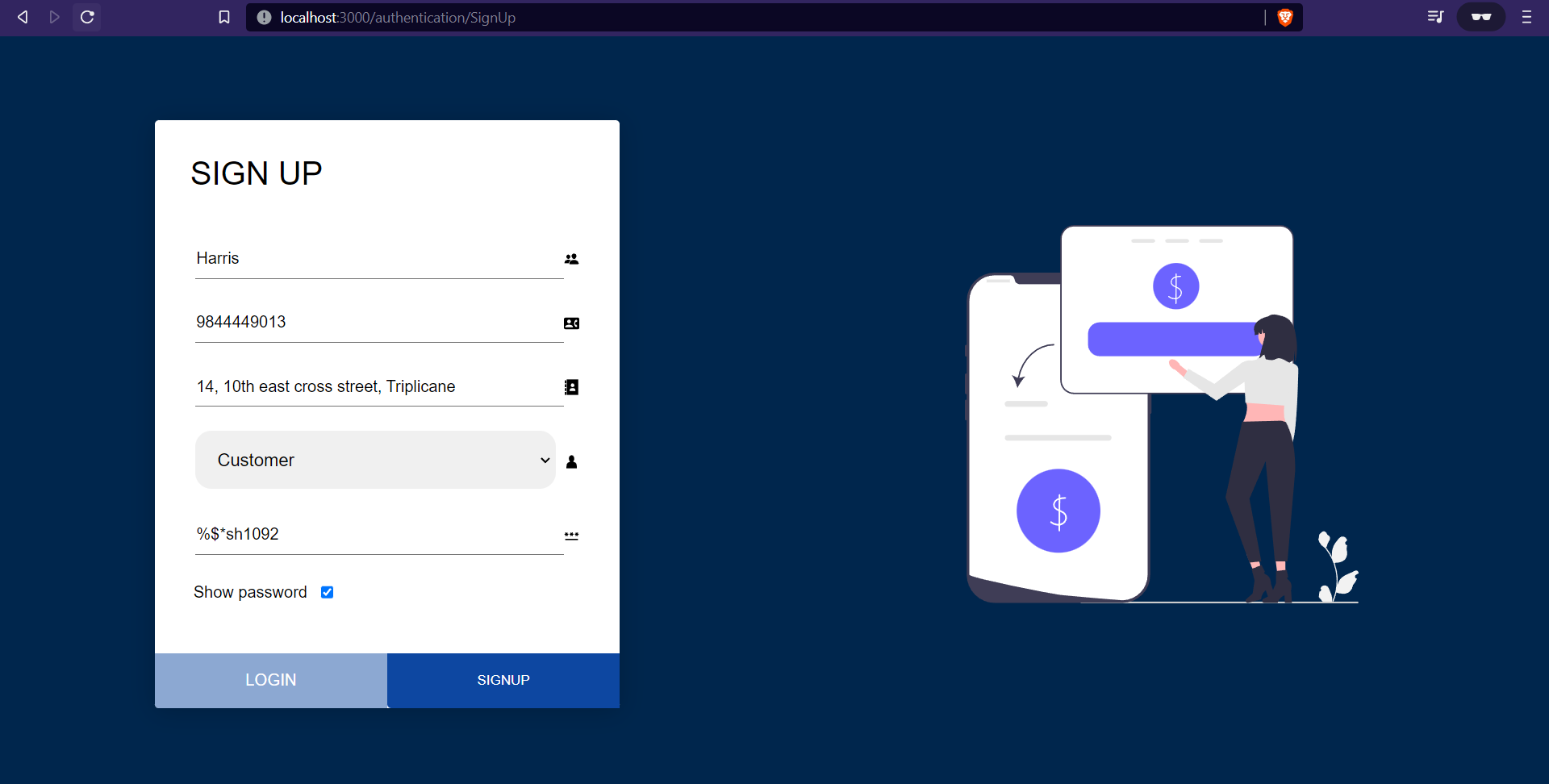
****

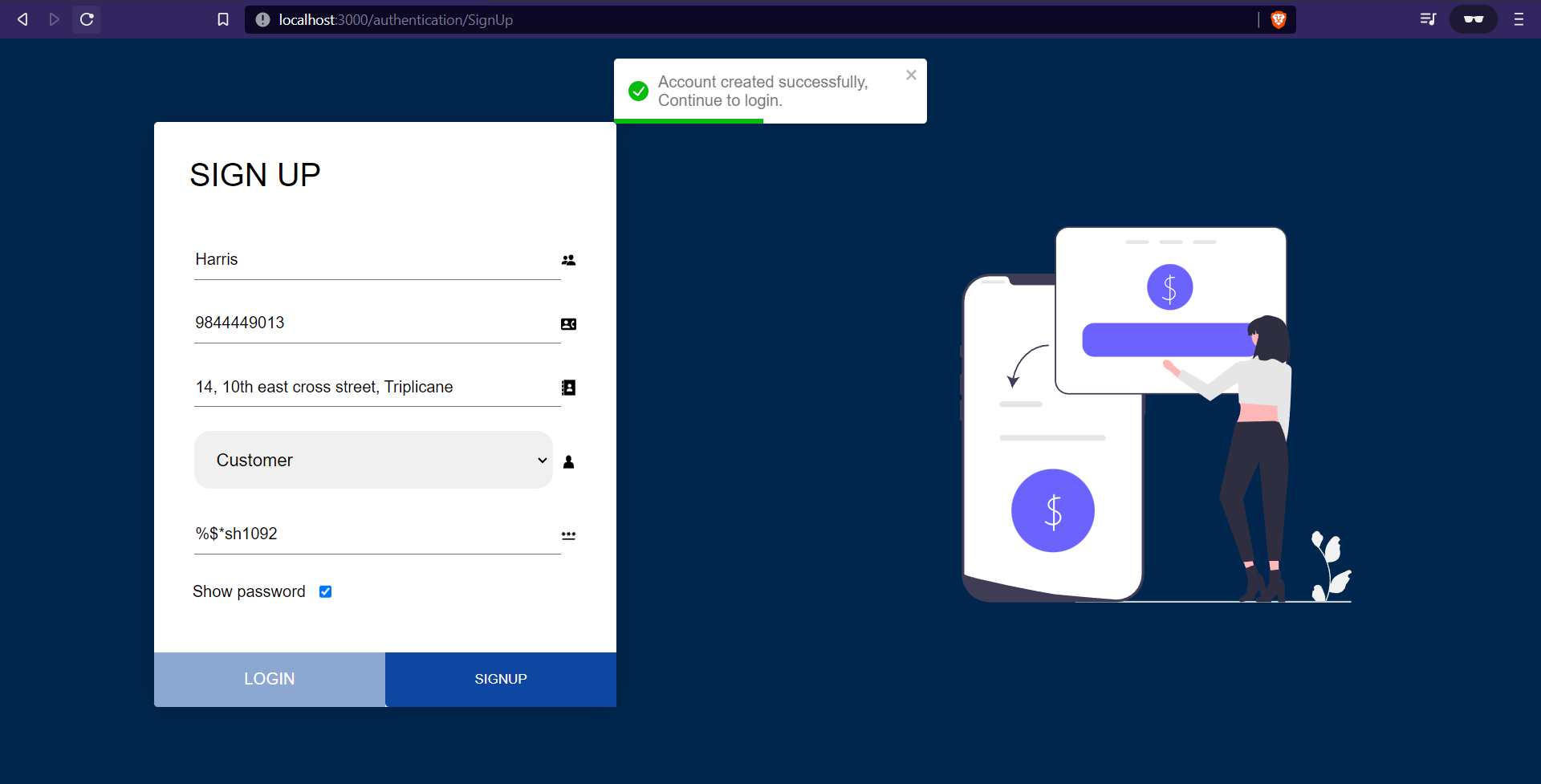
****

The login page backend work is done with the help of token-based authentication (JWT – JSON Web tokens) where the user id and role are mapped together so that the redirections will be done based on their role. The token will get expired in 24hrs. Along with this session management is done in such a way he can see the active sessions, and can terminate other sessions.

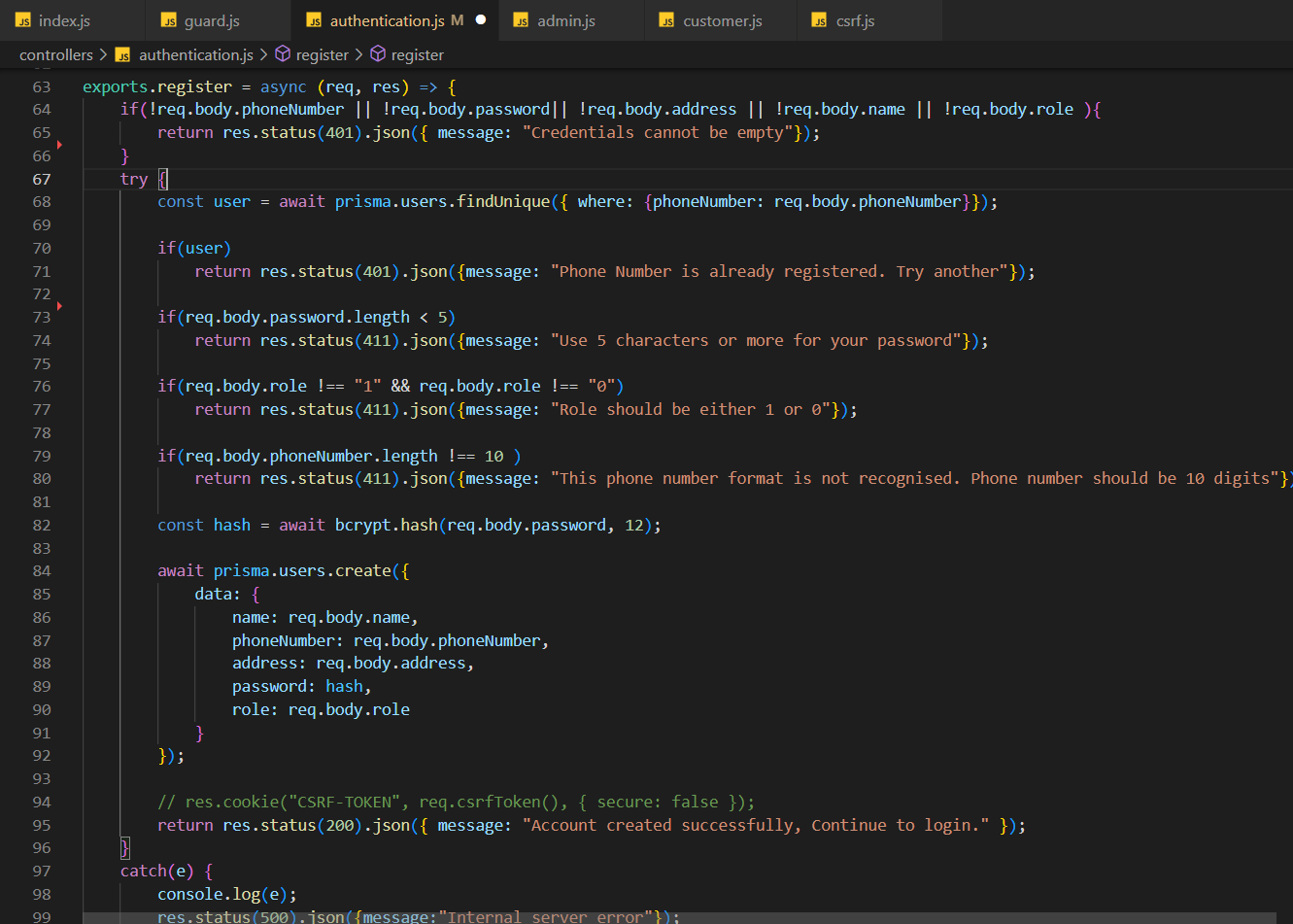
****

****

****

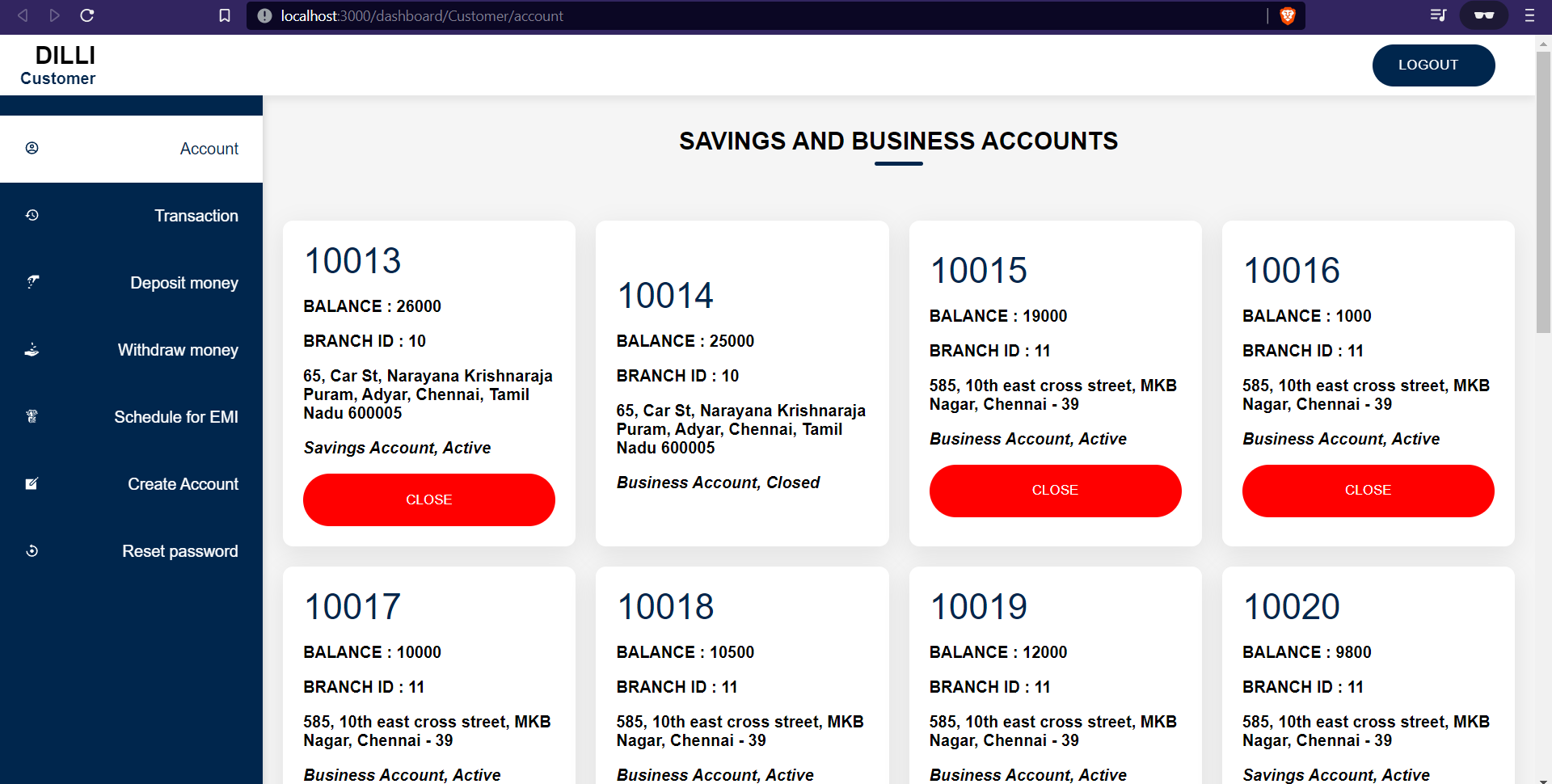
****

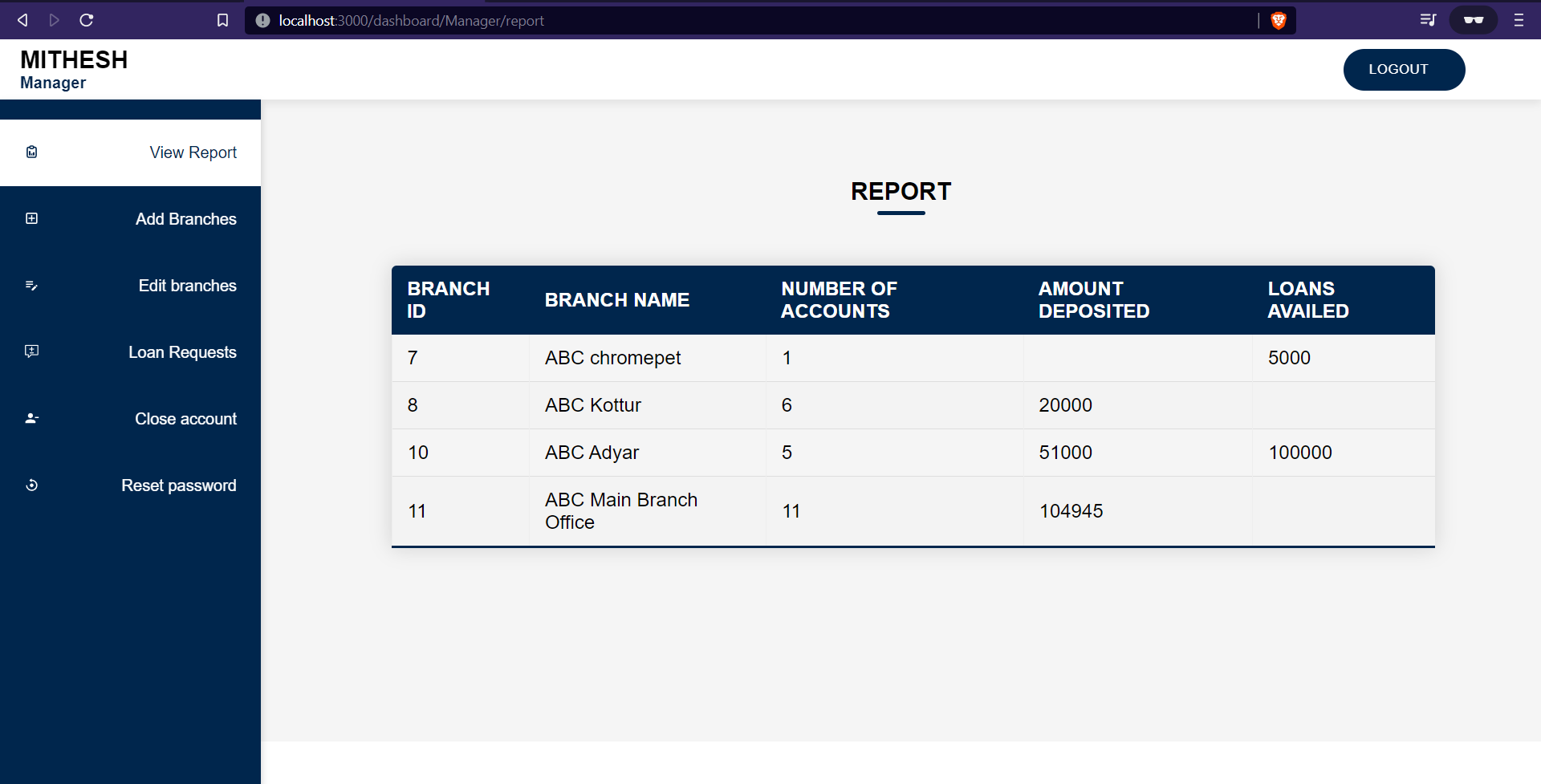
For security purpose, the password is hashed and saved in db. The hashing is done using blowfish encryption algorithm.

****

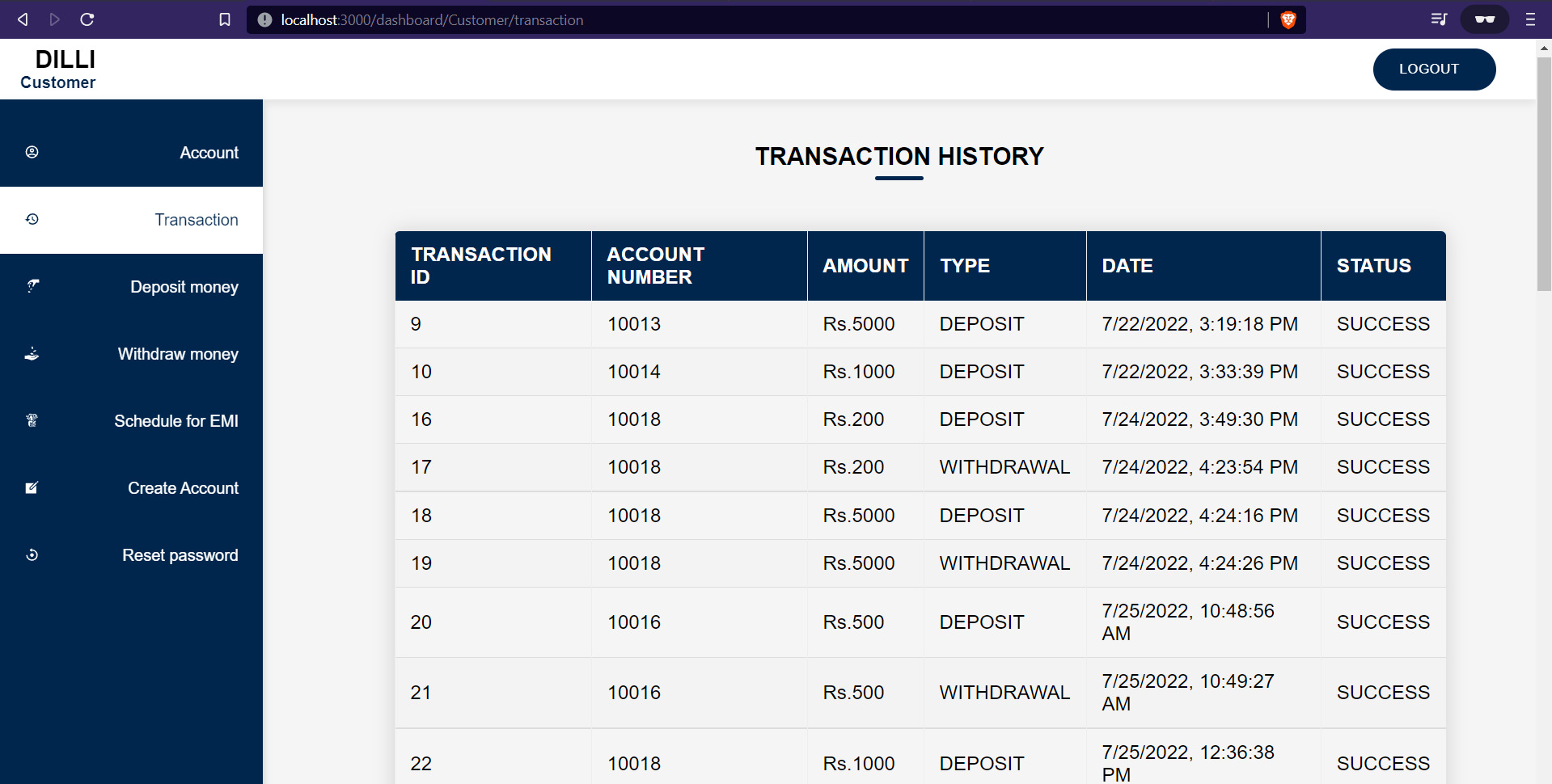
**DASHBOARD**

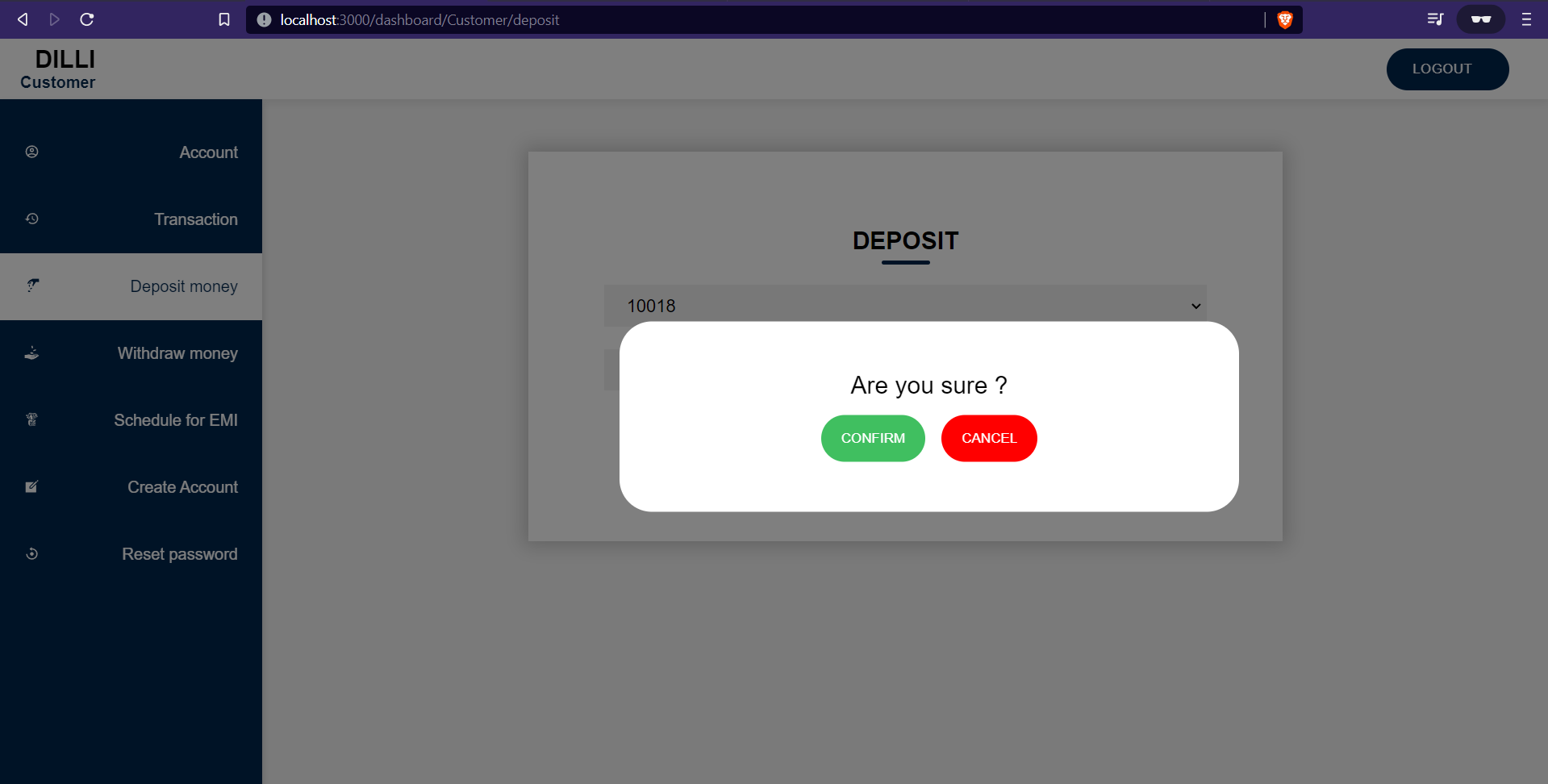
The customer/manager dashboard is designed in such a way that all functionalities are displayed clearly. In the accounts tab, the customer can see his/her savings and business accounts along with the loan accounts.

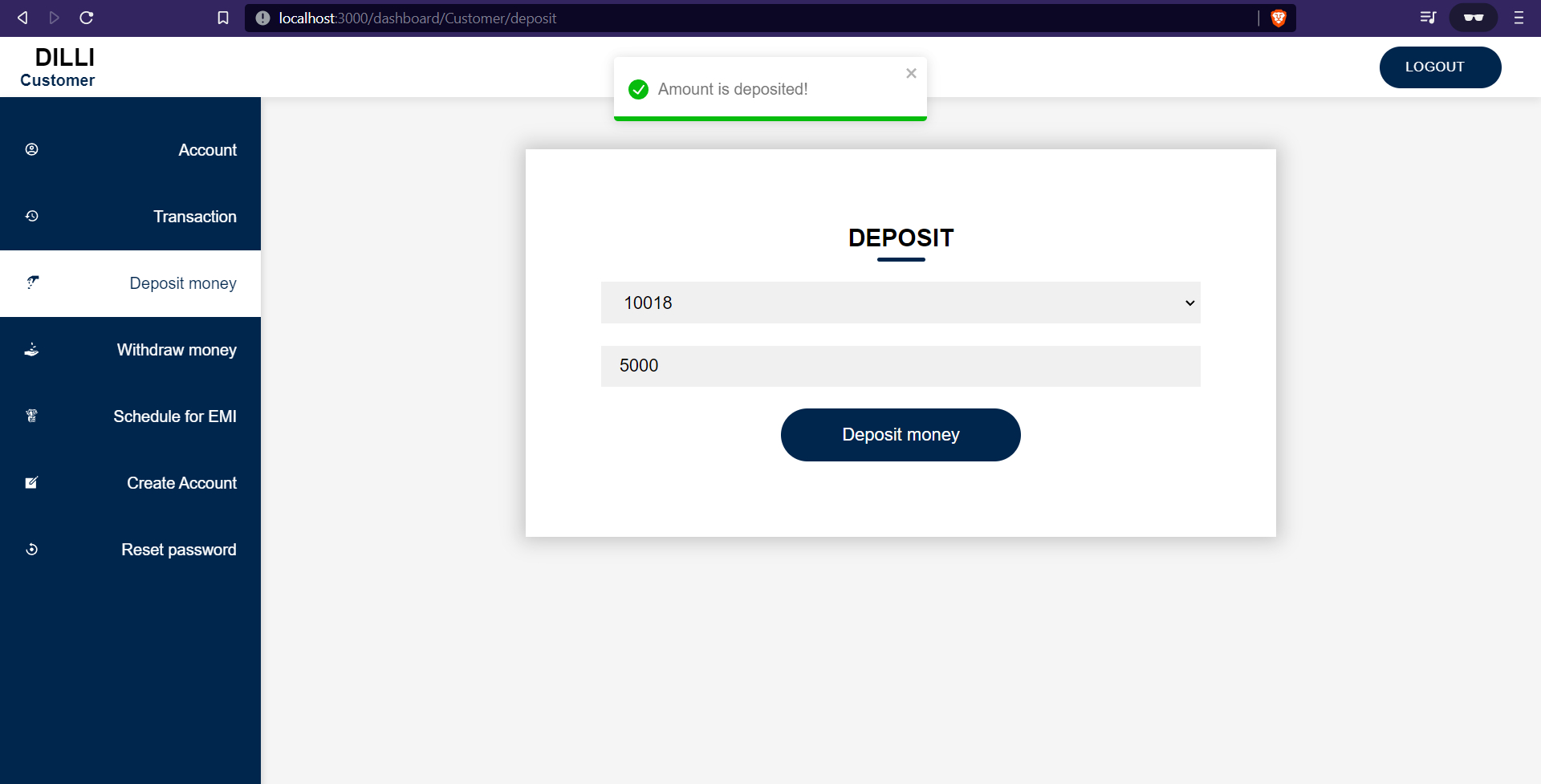
****

****

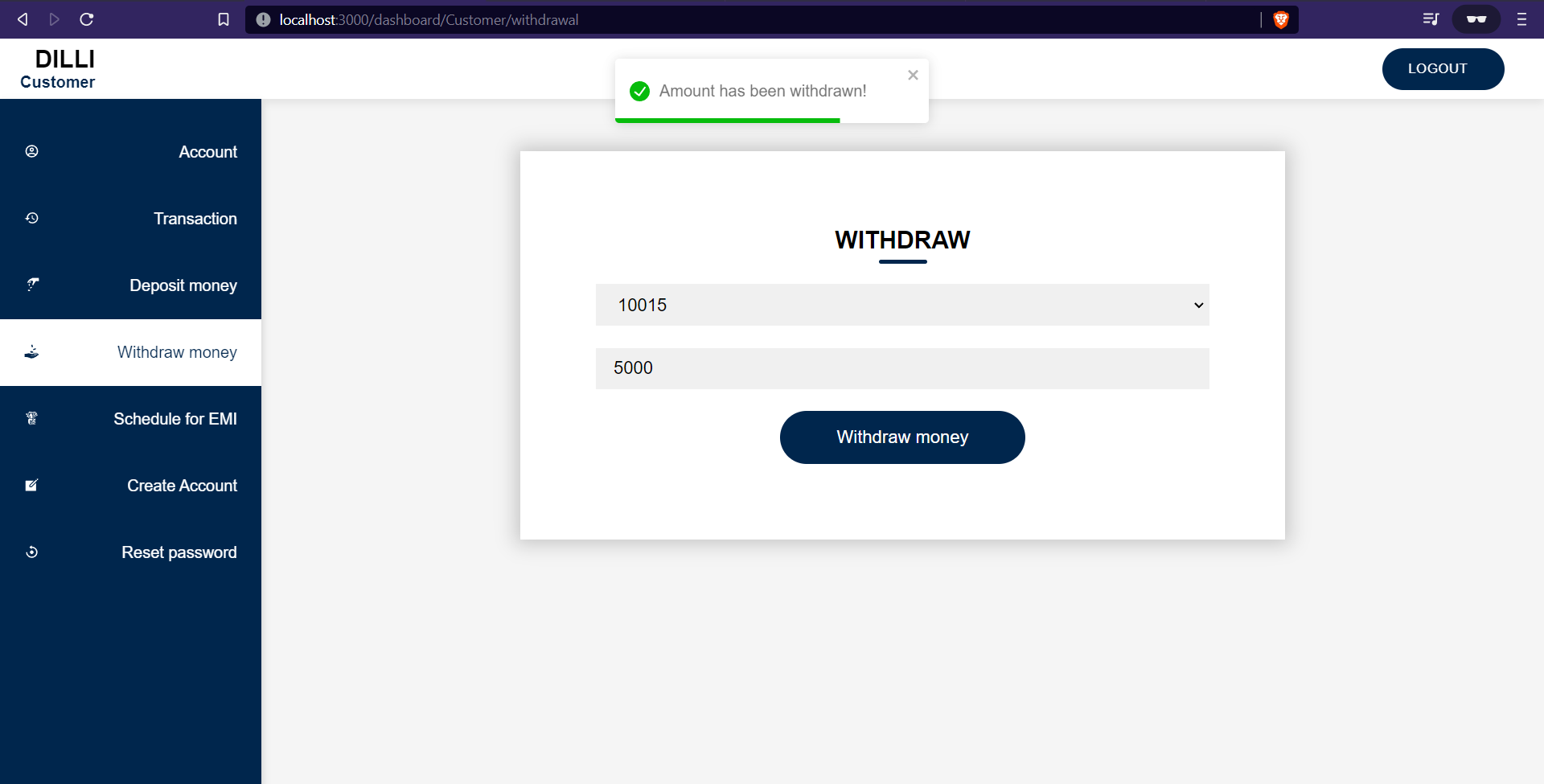
**DEPOSIT, WITHDRAW AND TRANSACTION HISTORY**

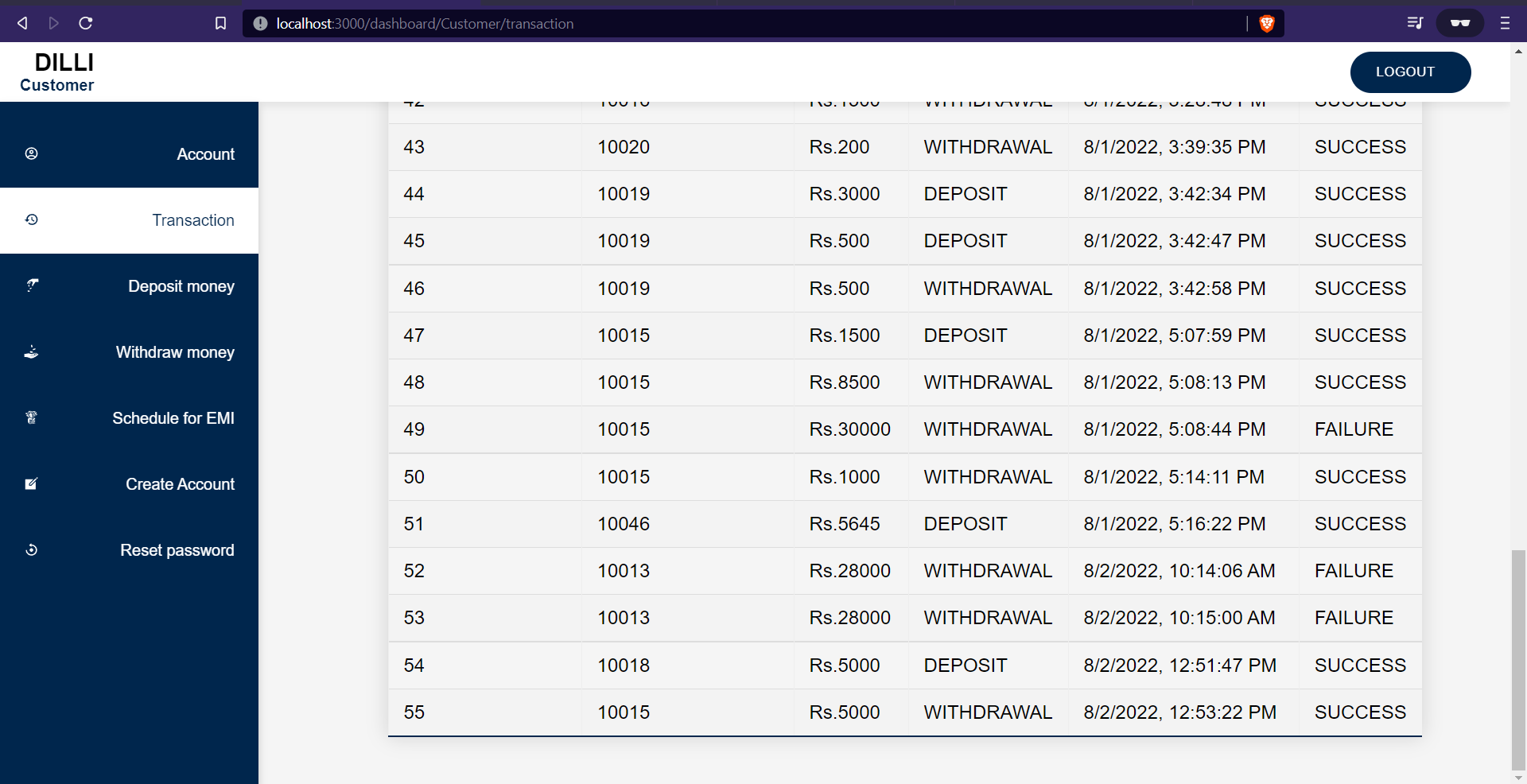
****

****

****

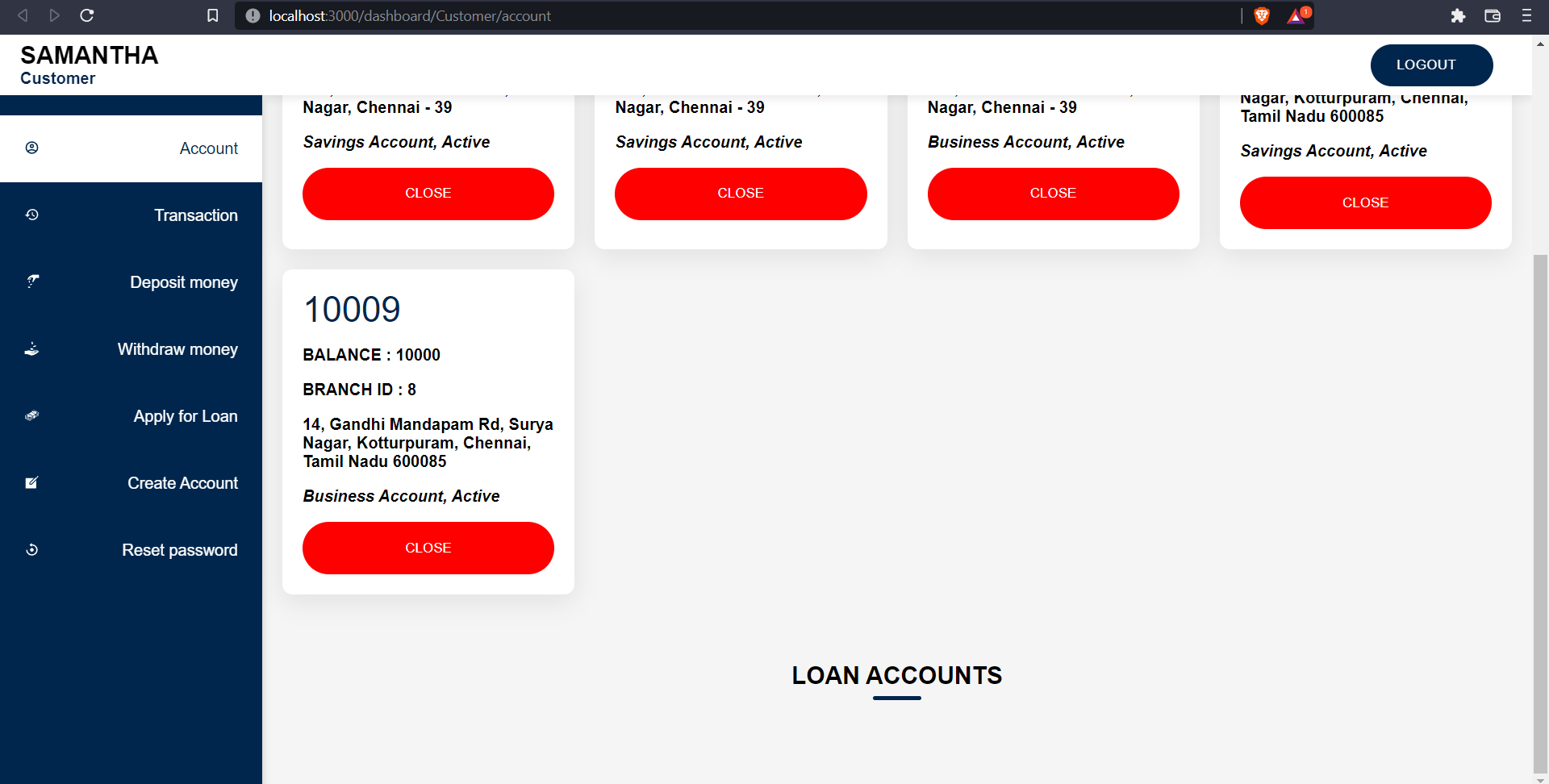
The deposit, withdraw functionality is done by first finding unique accounts in account table, and updating the balance according to the input to the specific selected account. Then the transaction table is updated with the selected account number, the amount deposited/ withdrawn, status showing “SUCCESS”, date is fetched in the UTC format. The status will show “FAILURE” if the requesting amount will cause the balance to be lesser than 1000. This constraint is maintained so that, if the person wants to close the account. The minimum balance 1000 rupees will get decremented from the closing account.

****

****

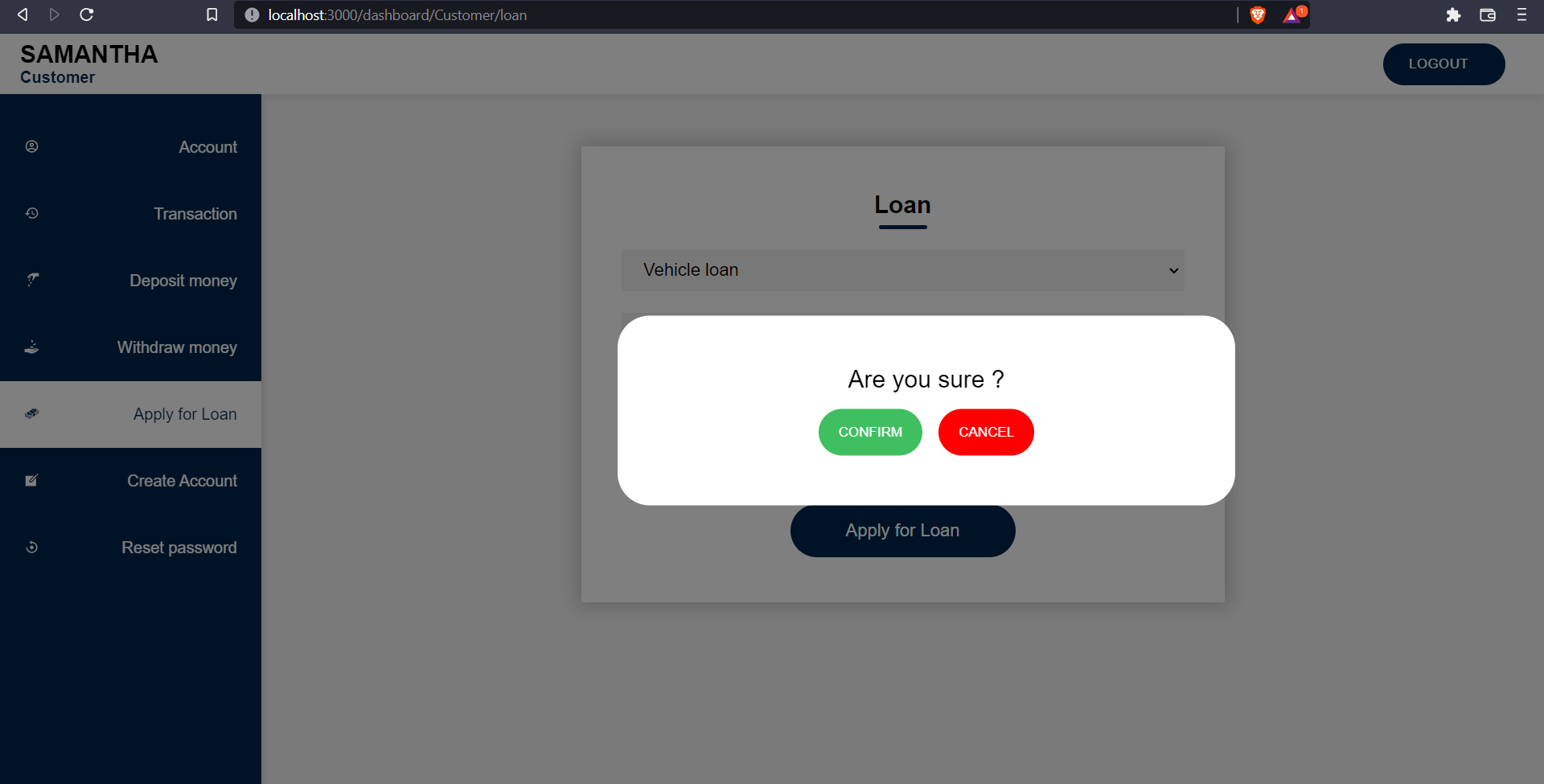
The deposit and withdrawal transactions are reflected in transaction history tab.

**APPLY FOR LOANS**

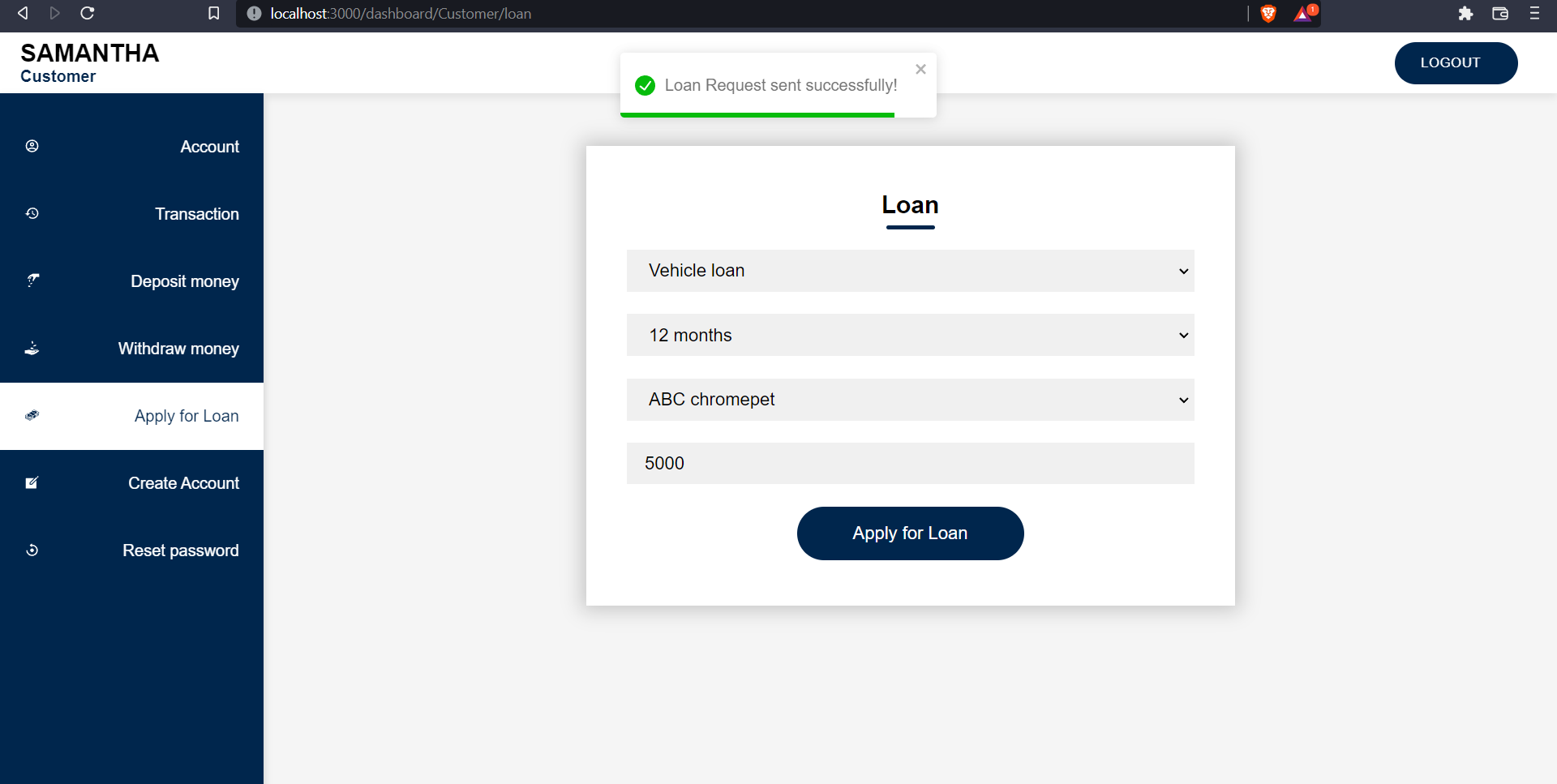


The apply for loan will be visible only if there is no unpaid loan is pending.

**Sending requests**

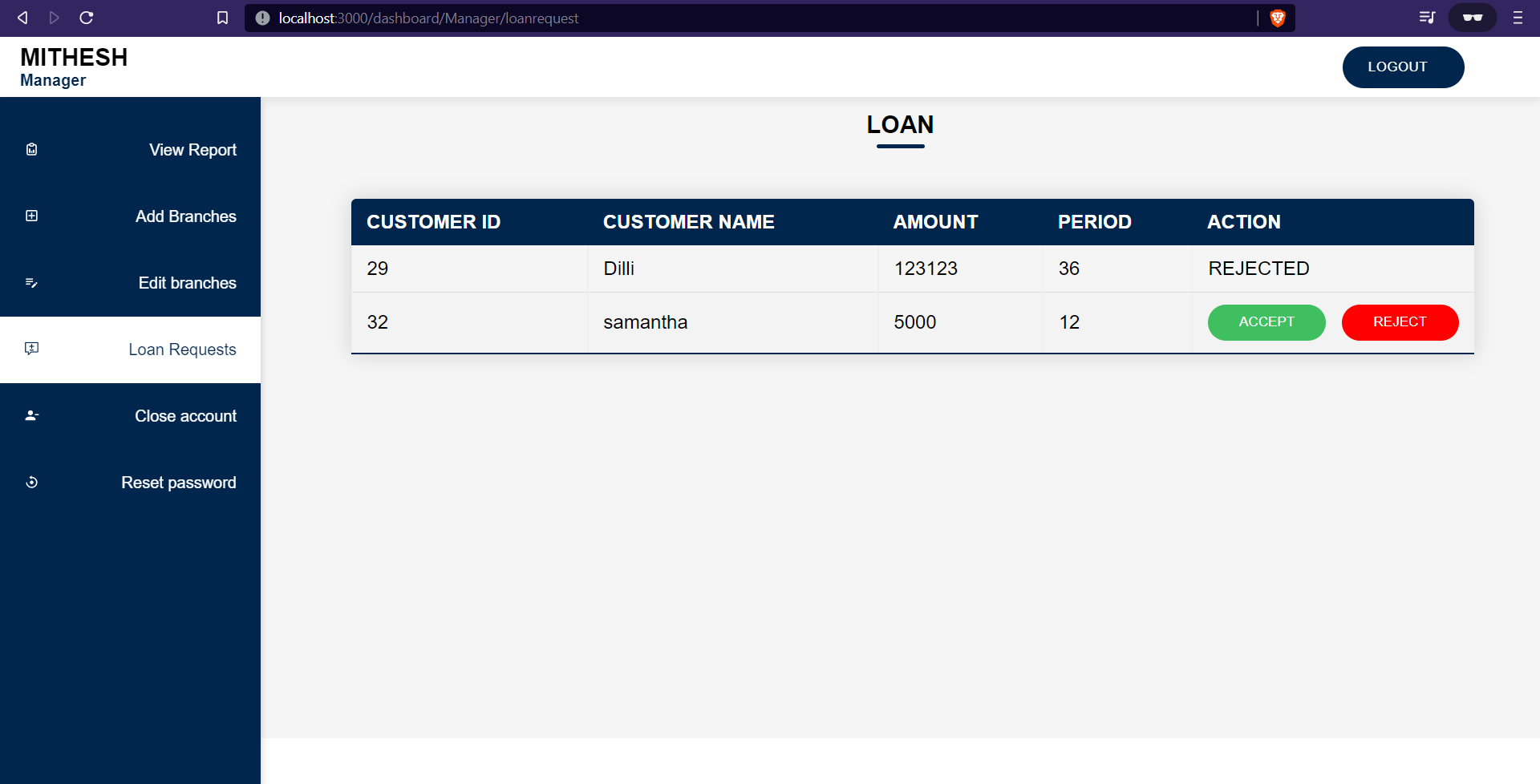


The appyLoan functionality will leads to the creation of request in the loan table with the status being 0, other details fetched from input. This request can be viewed by the manager who will accept/reject the loan requests.

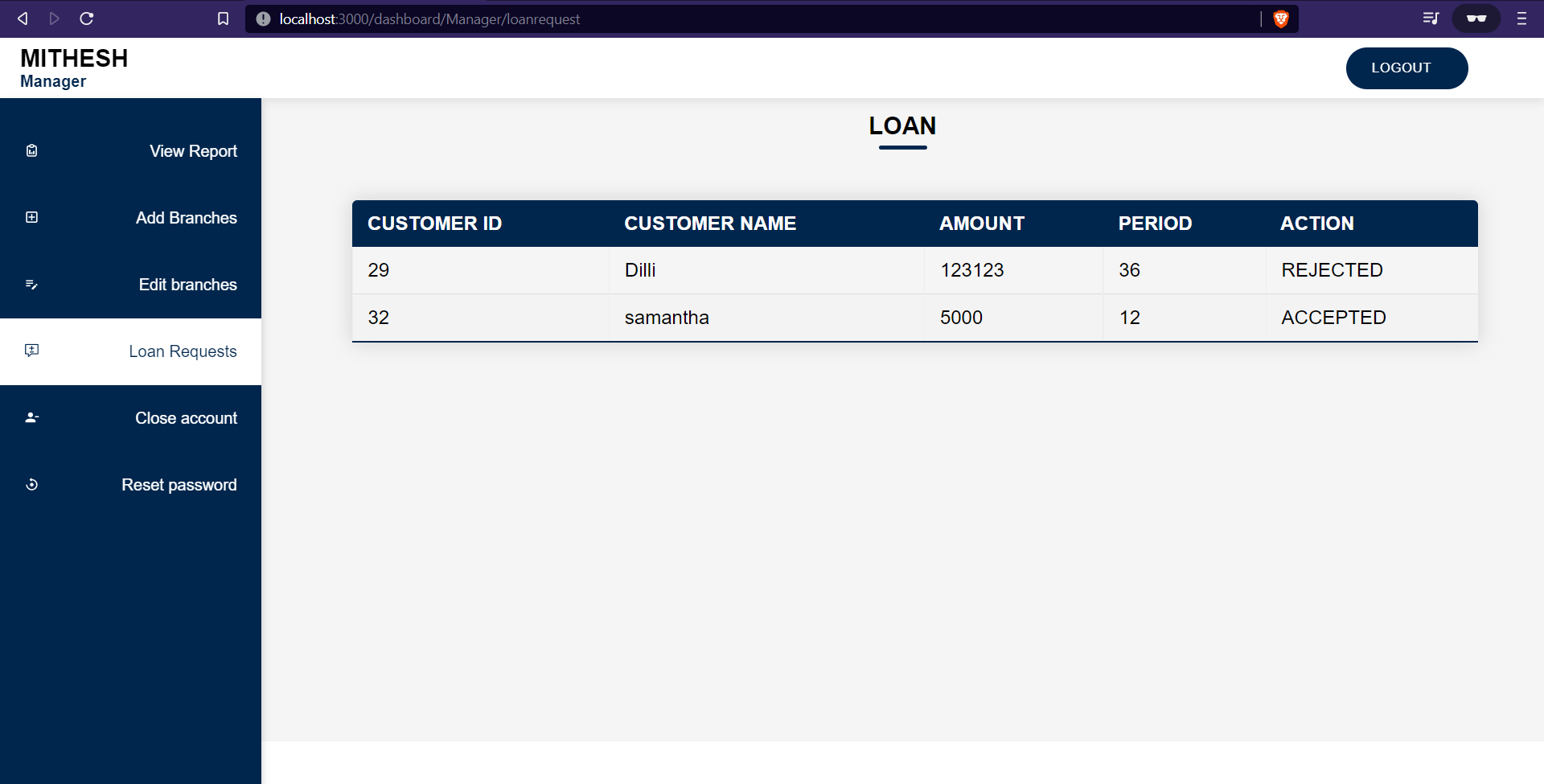


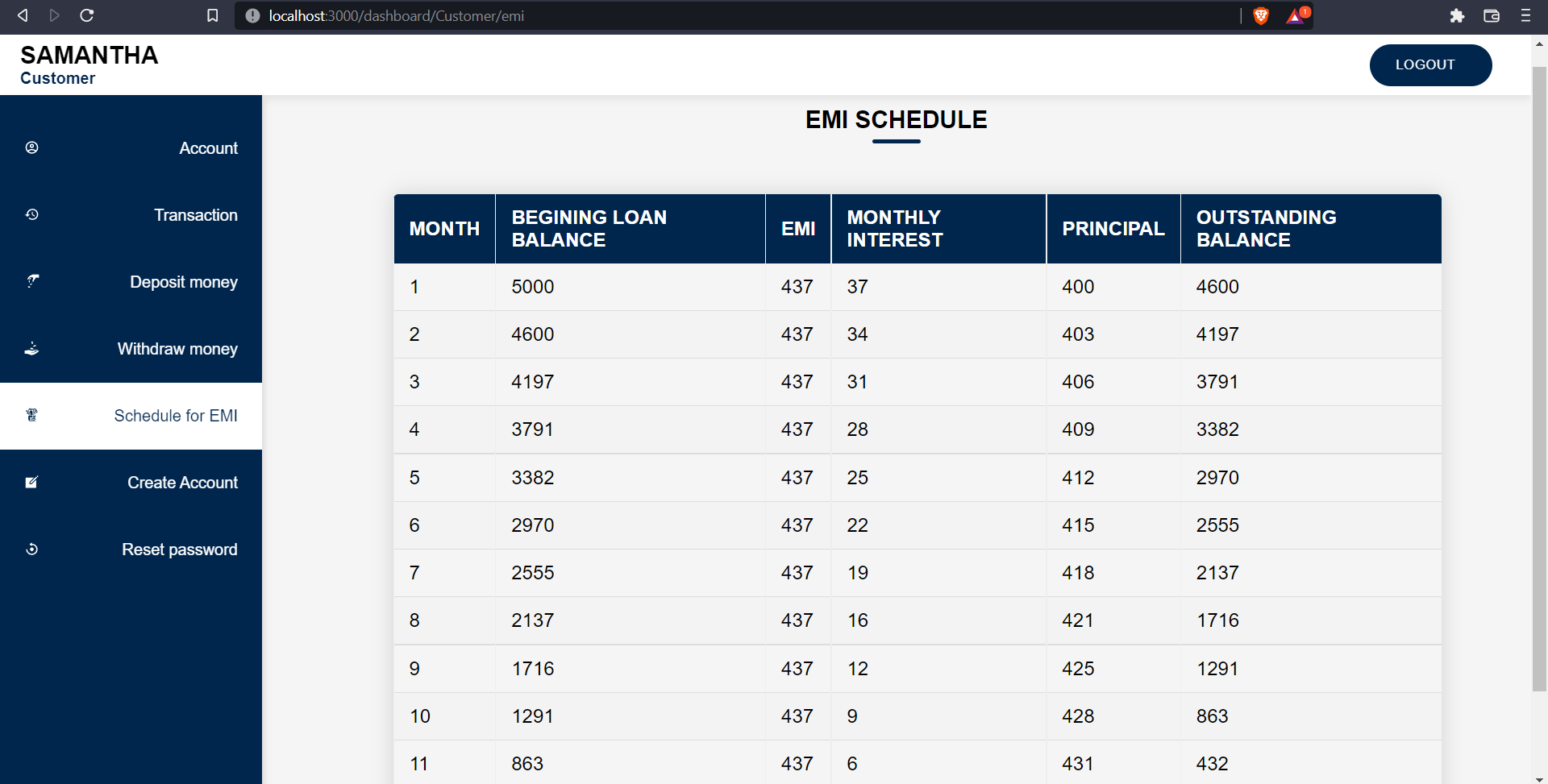
This loan request is sent to the manager. The manager can view the requests in his dashboard using find many relationships where the id, status, amount, type, period and users are selected. The loan requests can be updated using find unique function where the id is selected which includes users, then

* If the manager accepts, then it will lead to the creation of account in accounts table with type 3(loan account) and updating loan account with the status changing from 0 to 1.
* If the manager rejects, the loan table is updated with status changing from 0 to 2.

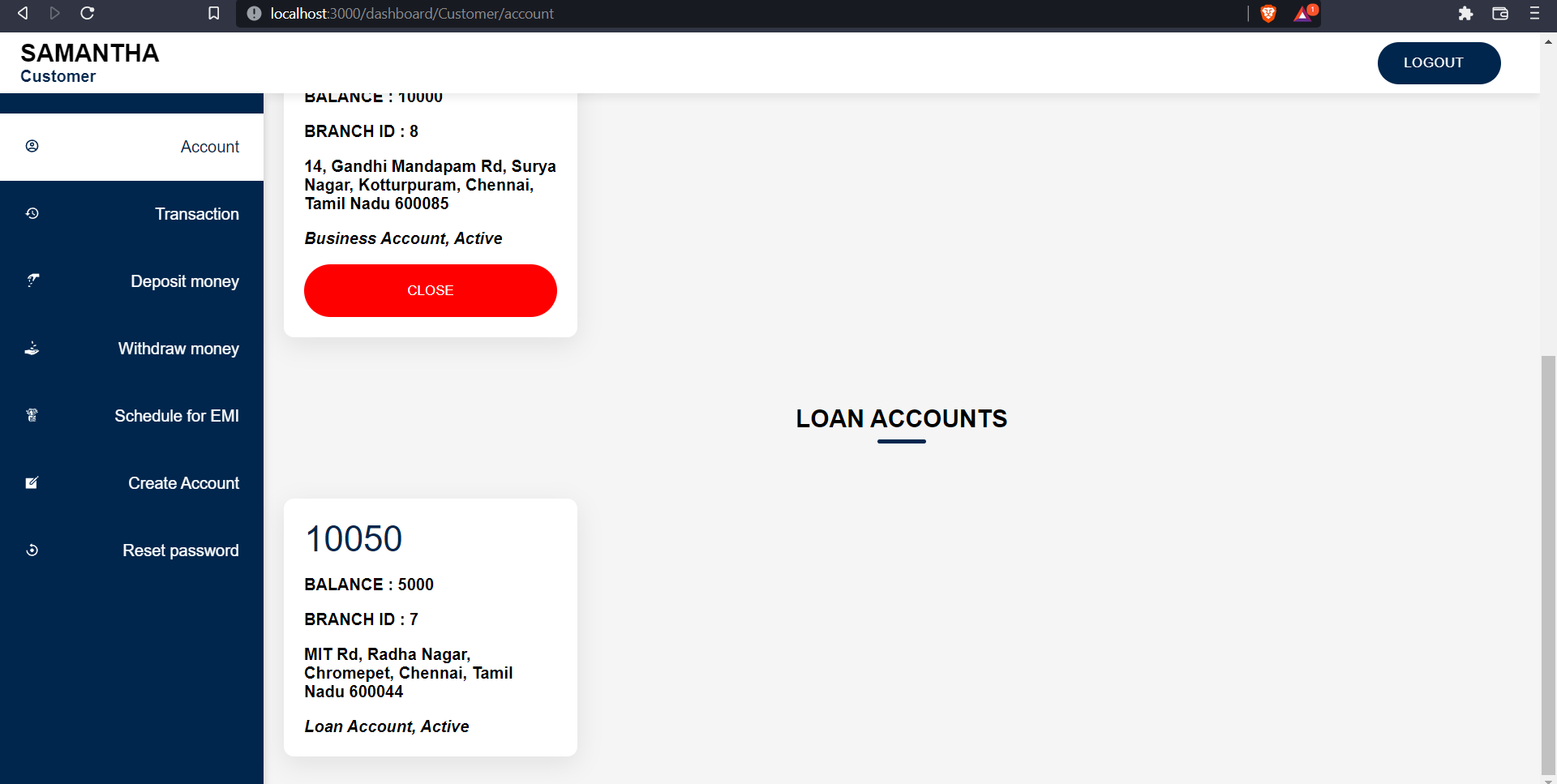


If the manager accepts, the loan account will be created and EMI SCHEDULE tab will replace the APPLY FOR LOAN tab in customer’s dashboard. If the manager rejects, then the customer can apply again if he/she wishes.

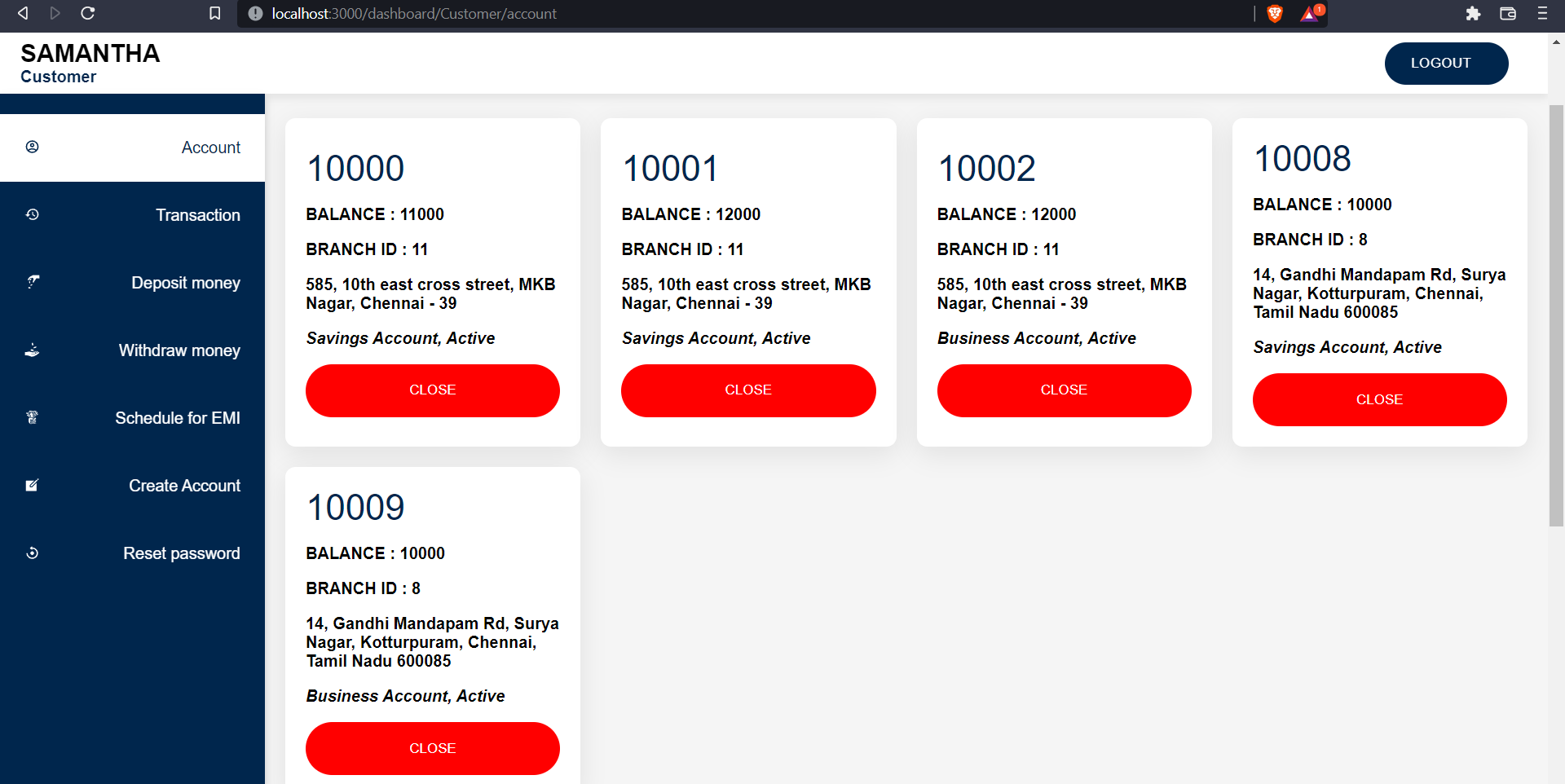




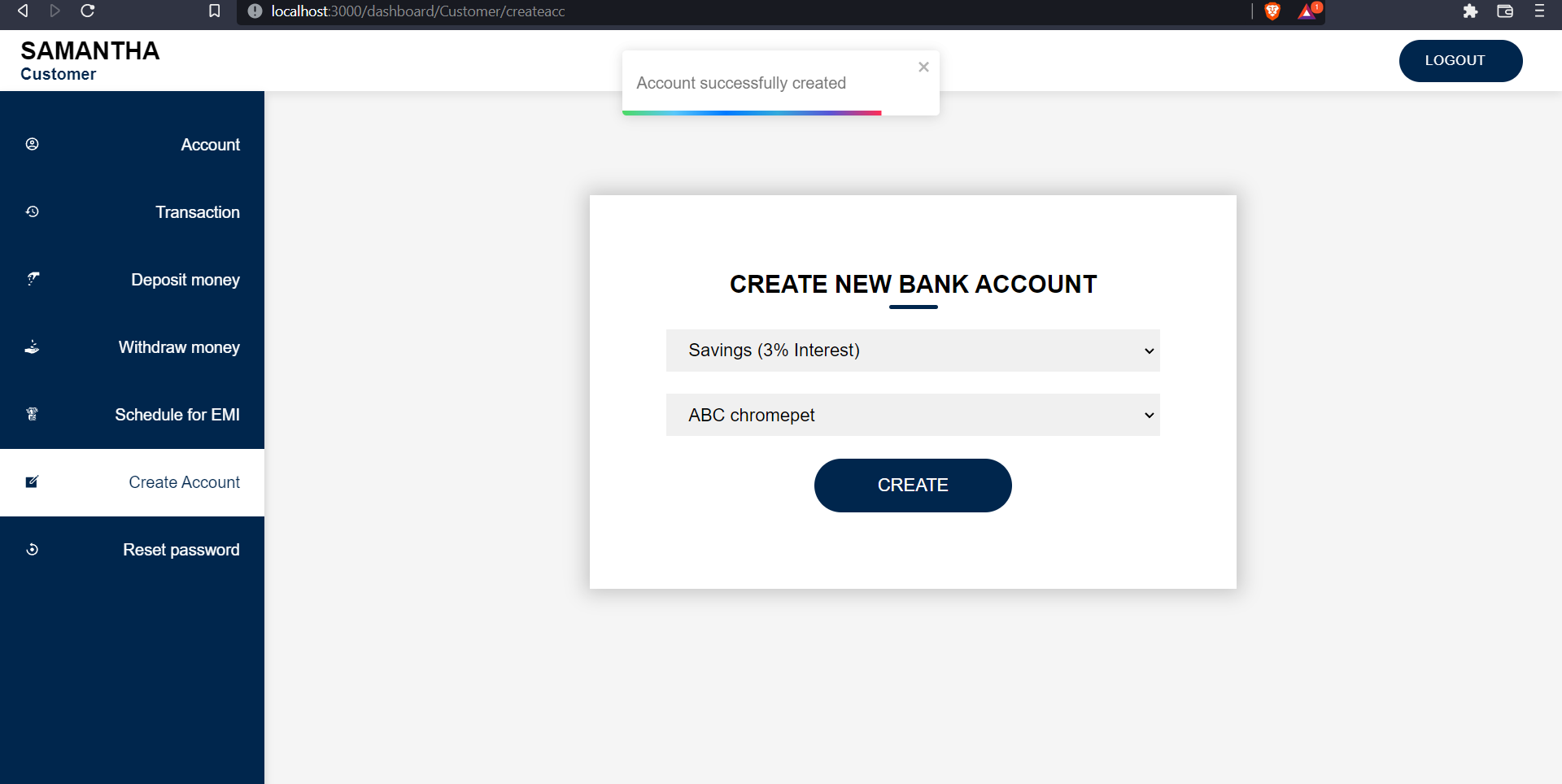
The customer can apply for another loan only if he repays the current loan.

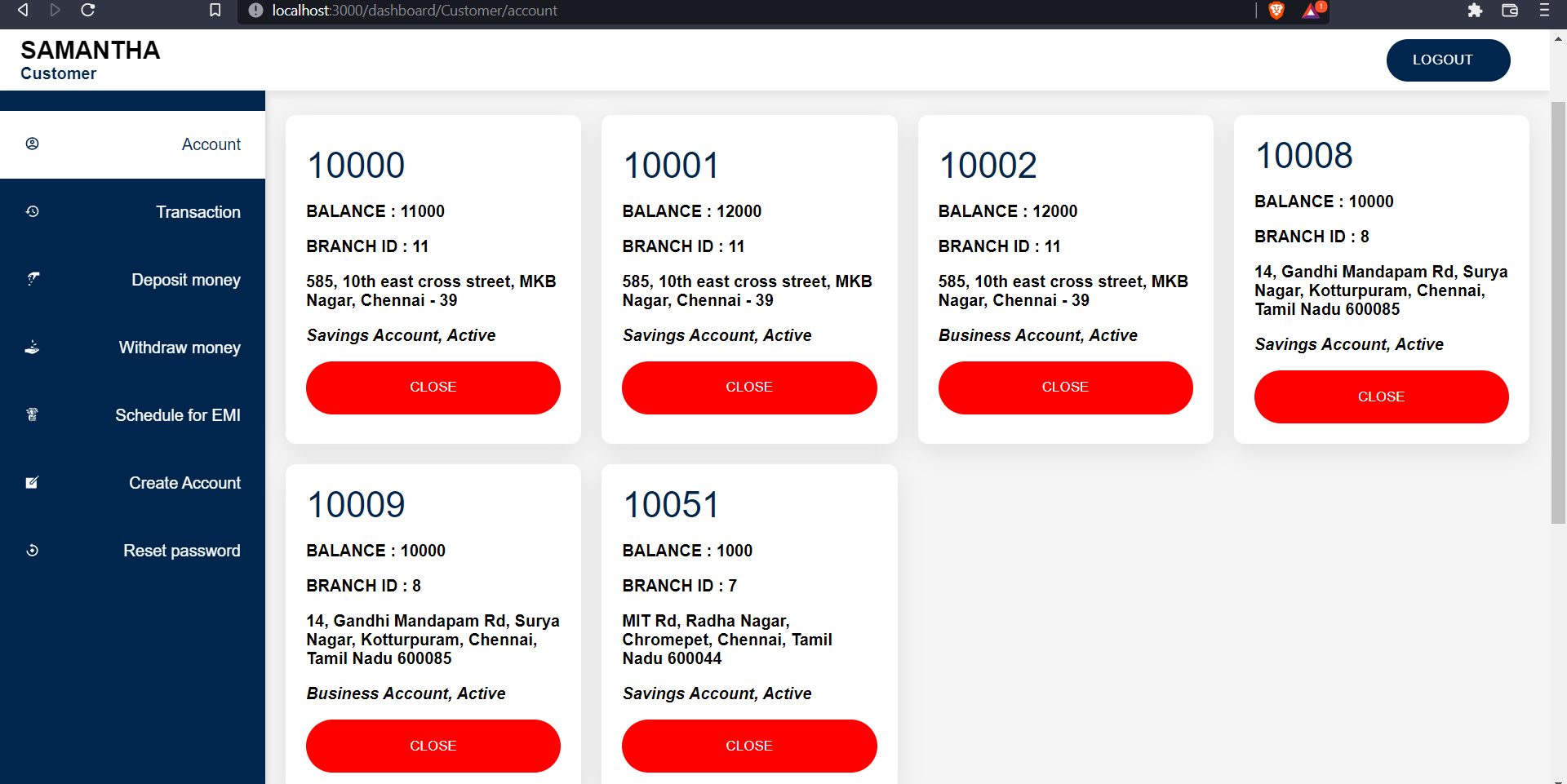


**CREATING NEW ACCOUNTS**



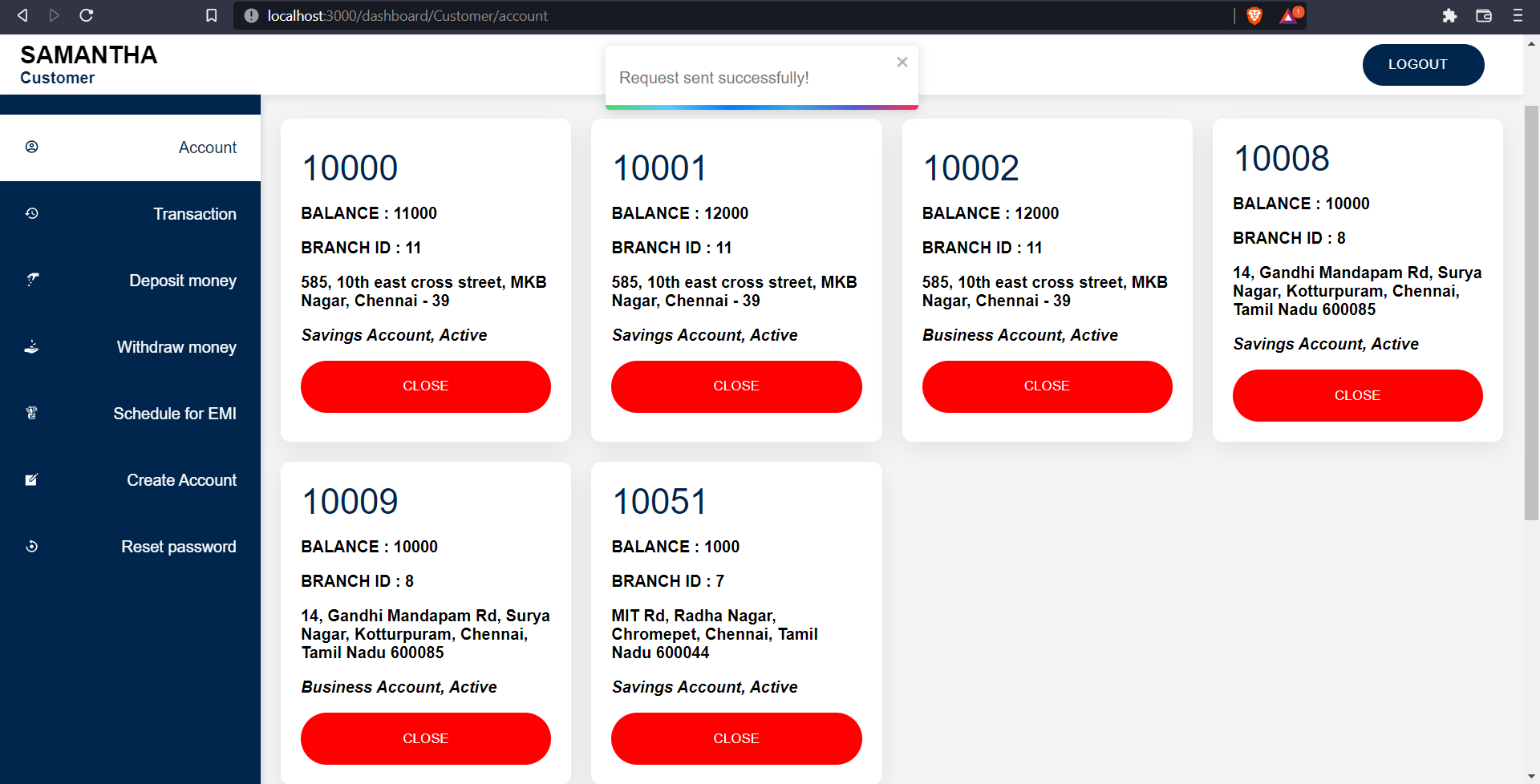
The customer can create a bank account of one of the two types available: savings, business account in any of the available branches. The accounts are created in the accounts table with the type of accounts and branch fetched from the input.



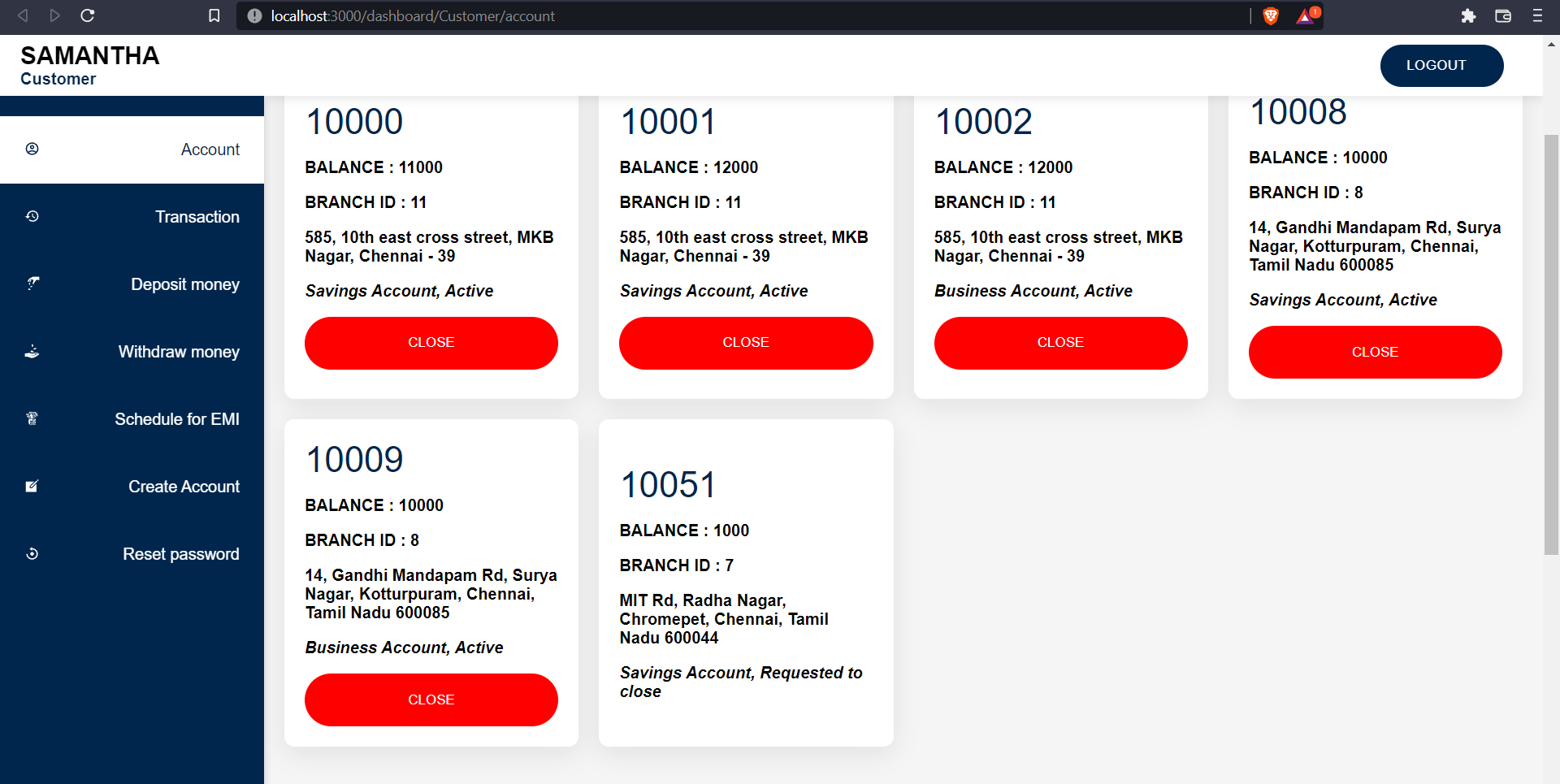


The account number is automatically generated in the database.

**CLOSING THE ACCOUNT**



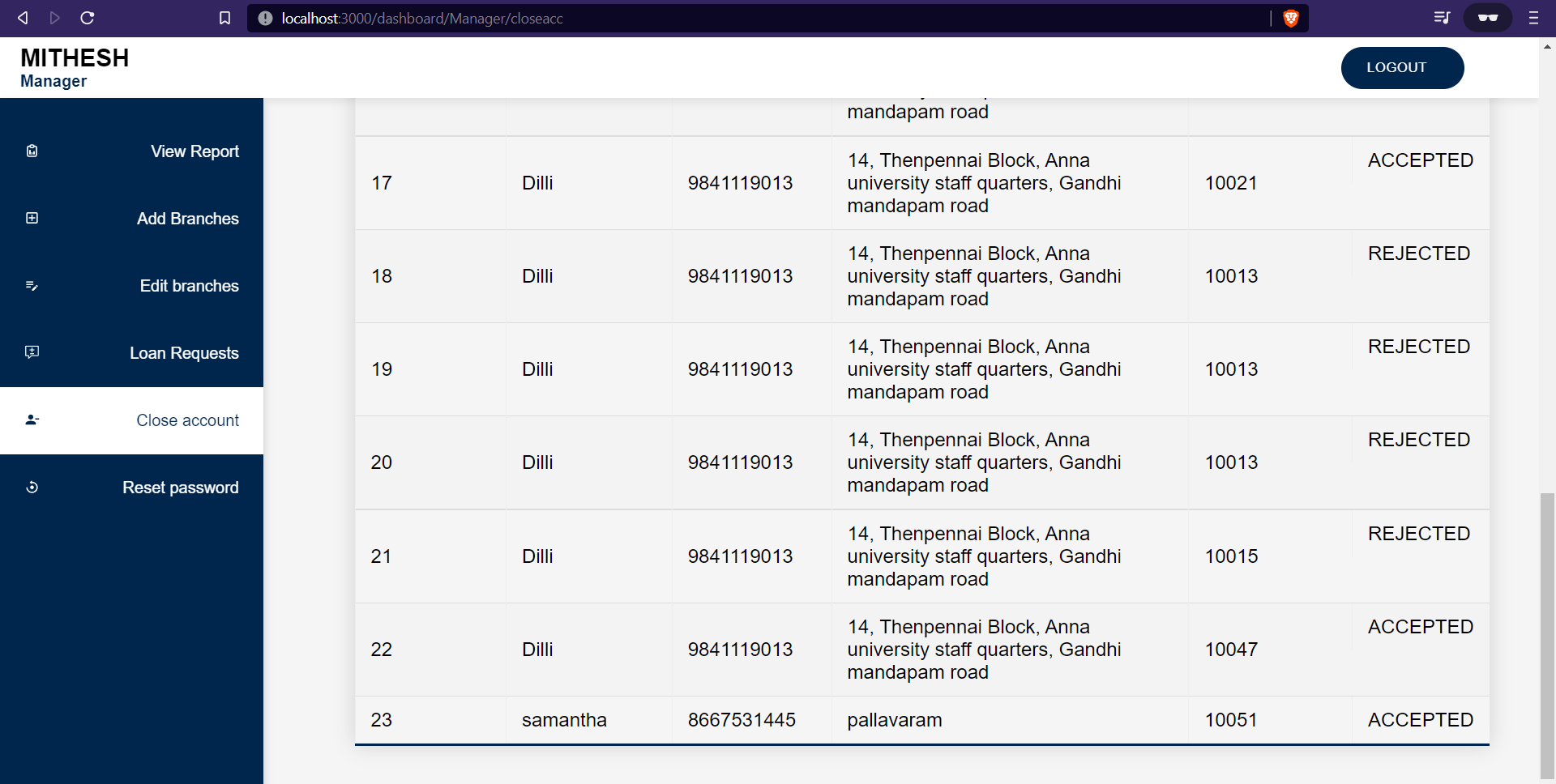
On requesting to close the bank account, the status of the account is changed to REQUESTED TO CLOSE so that no two requests to close the same account occur.

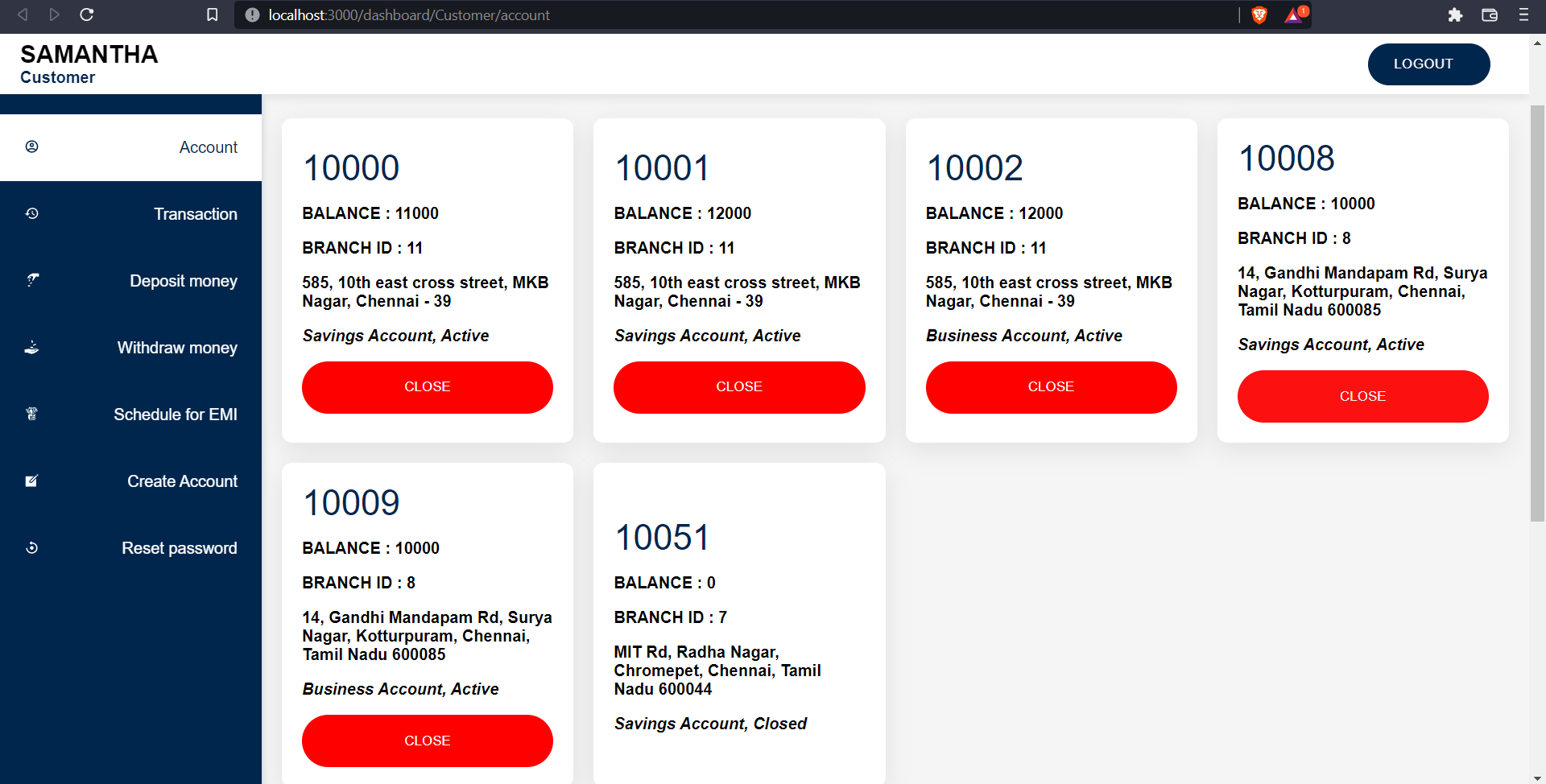


The request can be viewed by the manager with the help of find many relationships where the id, status and a nested select of account number, users from accounts are selected. The updating request with the help of find unique were based on the id, accounts are included. If the manager accepts, then the account table is updated with the amount decrementing by 1000 and changing the active state from 1 to 0. If the manager rejects, then the account table is updated with the active state remains 1 and the request table status is changed.

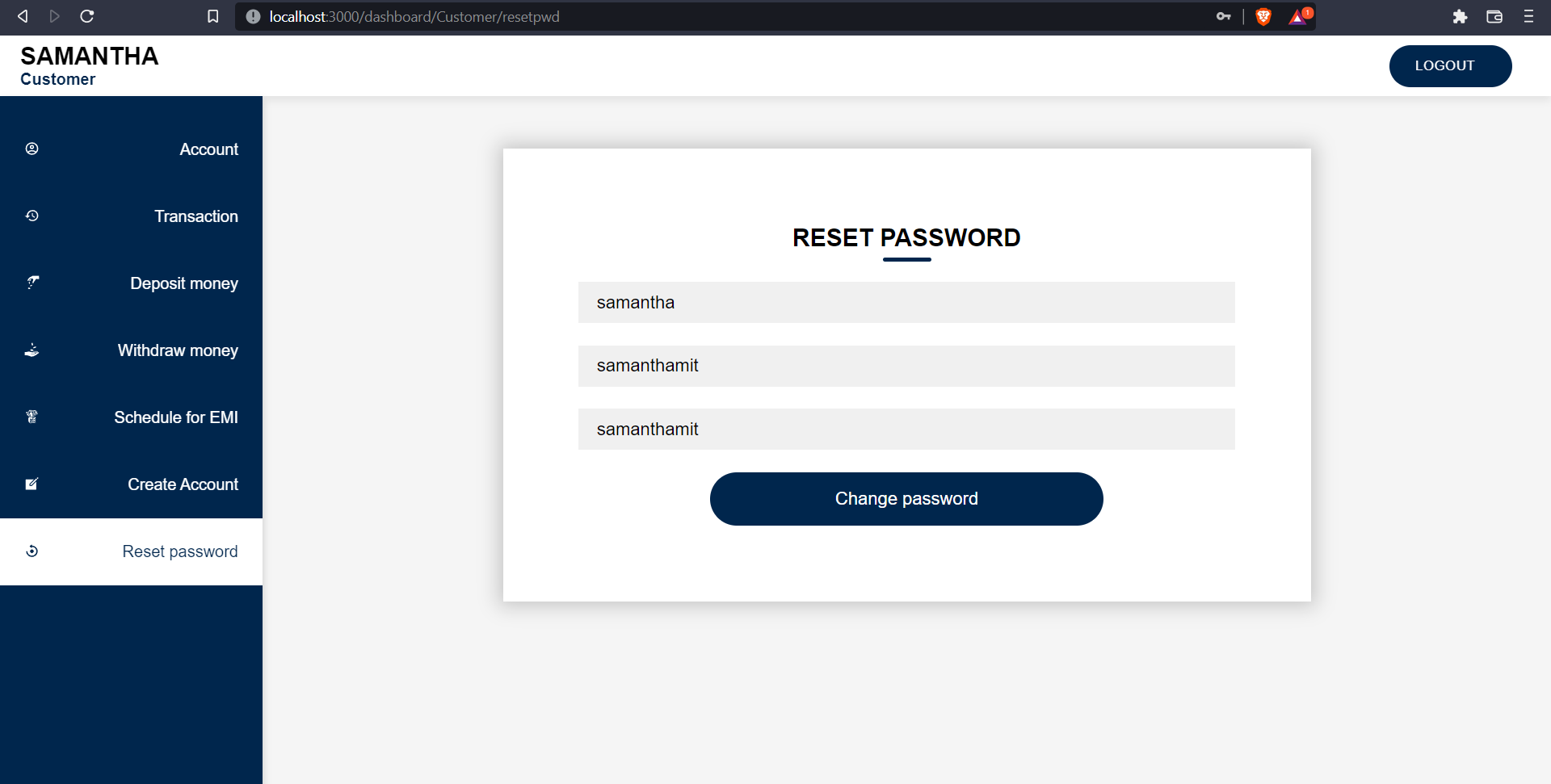


On accepting the request, the customer’s account balance will be decremented by 1000 in charge for closing his account. If the manager rejects the request, then the customer can do his regular deposit and withdraw activities.

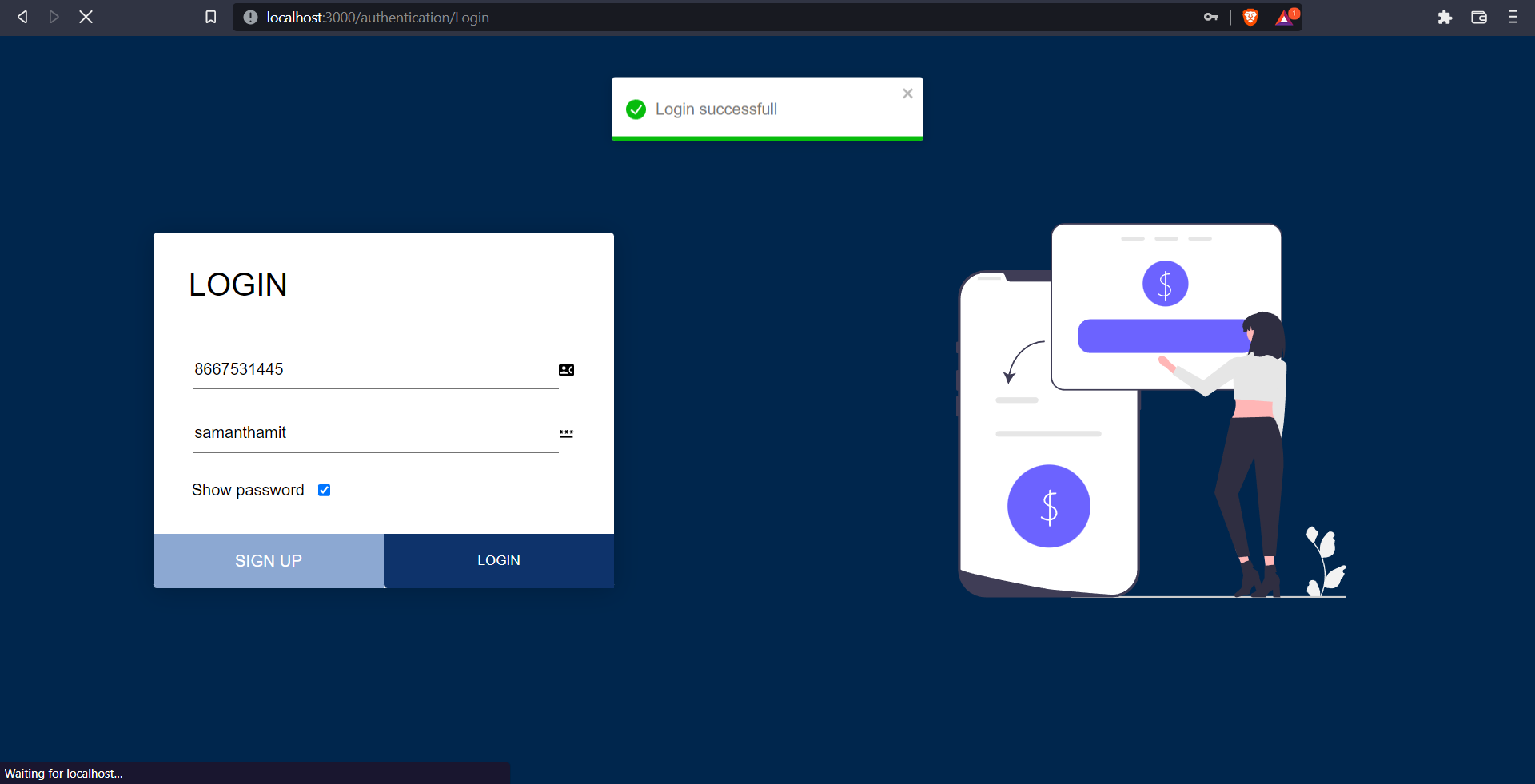




Both the customer and manager can reset their passwords, by mentioning the recent old password. The user’s table is updated with the new password which is hashed using bcrypt (blow fish encryption algorithm) based on the id. The cookie is cleared so that after changing the password, the website get automatically logout.



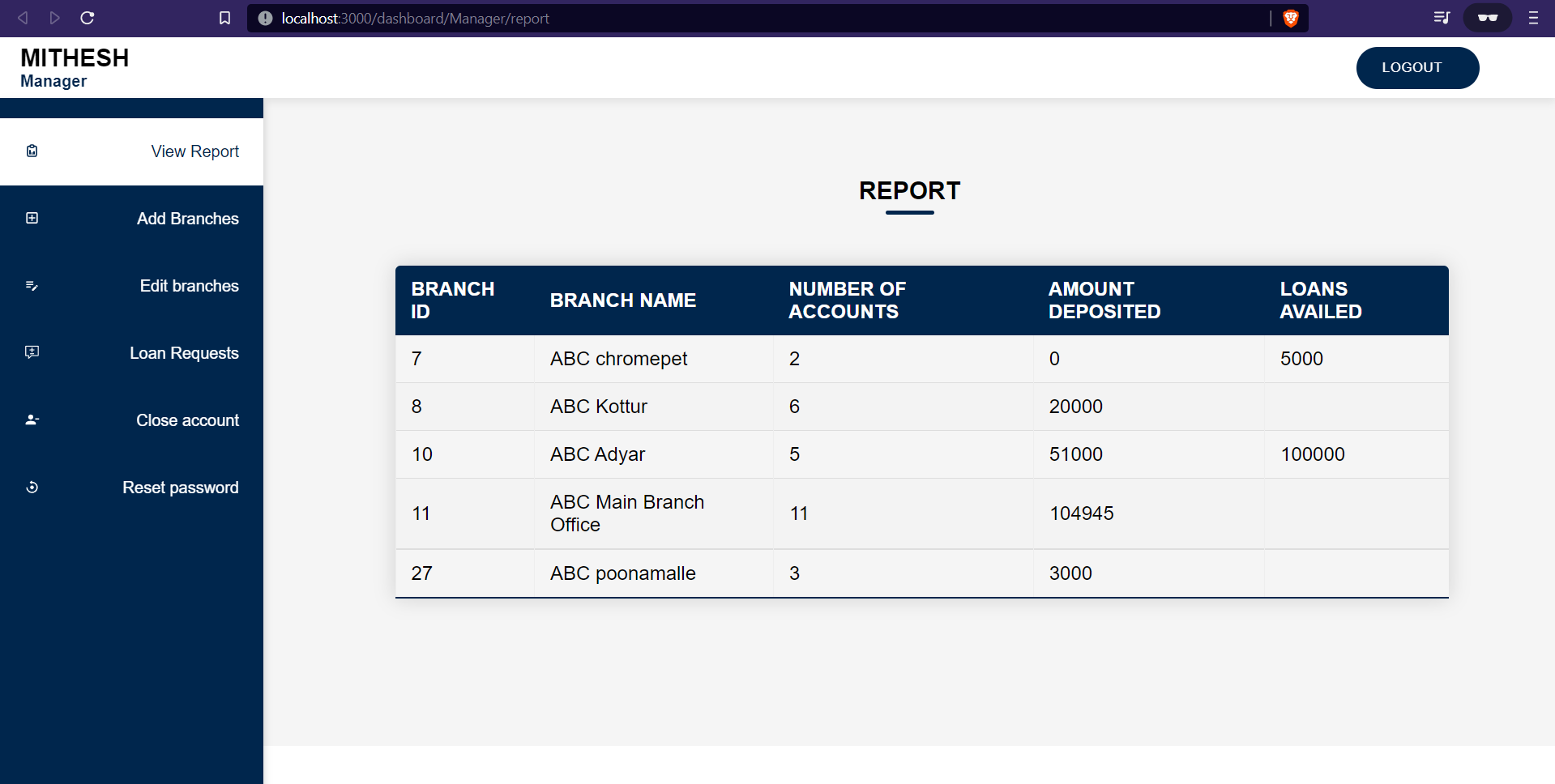


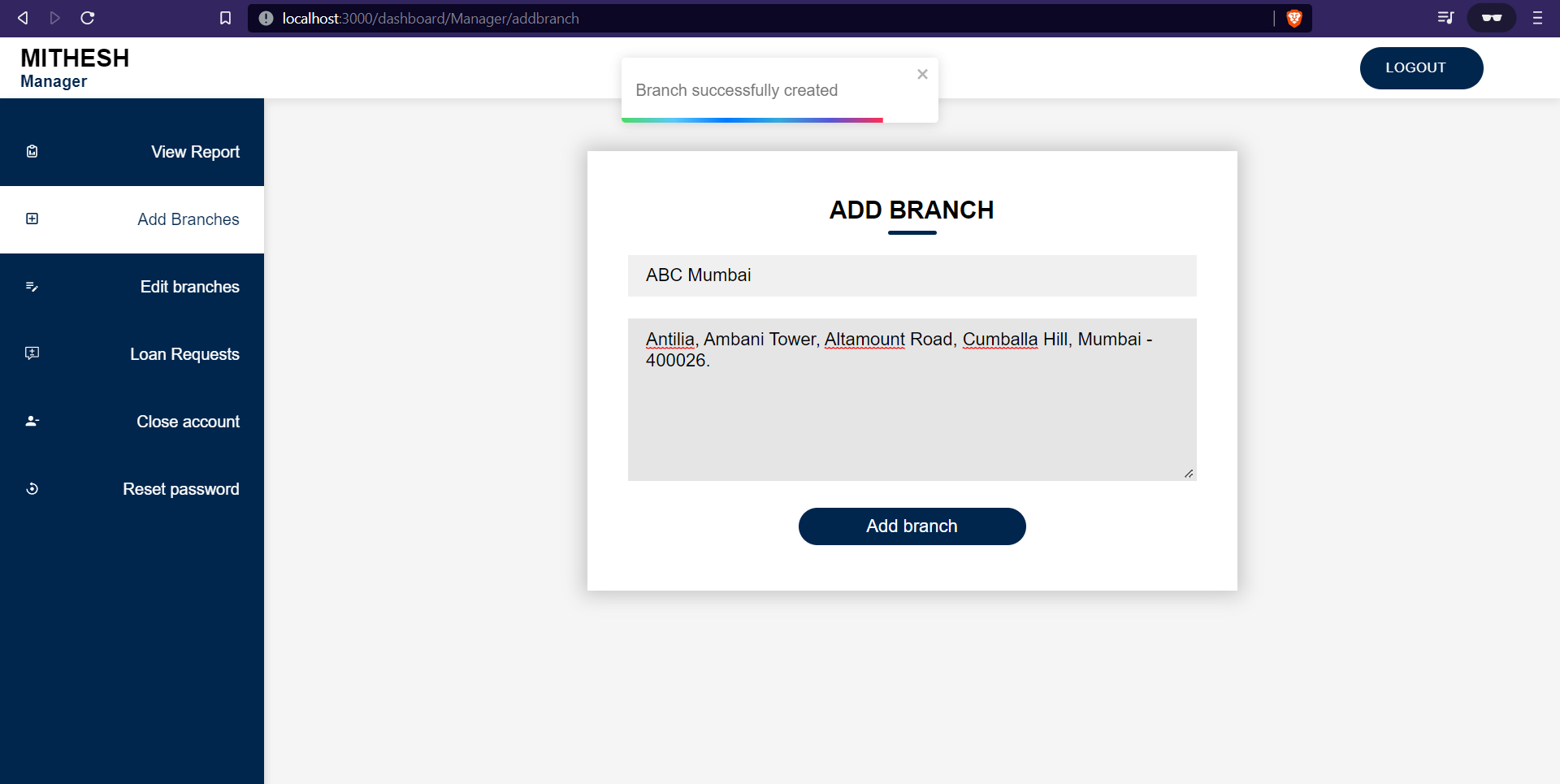


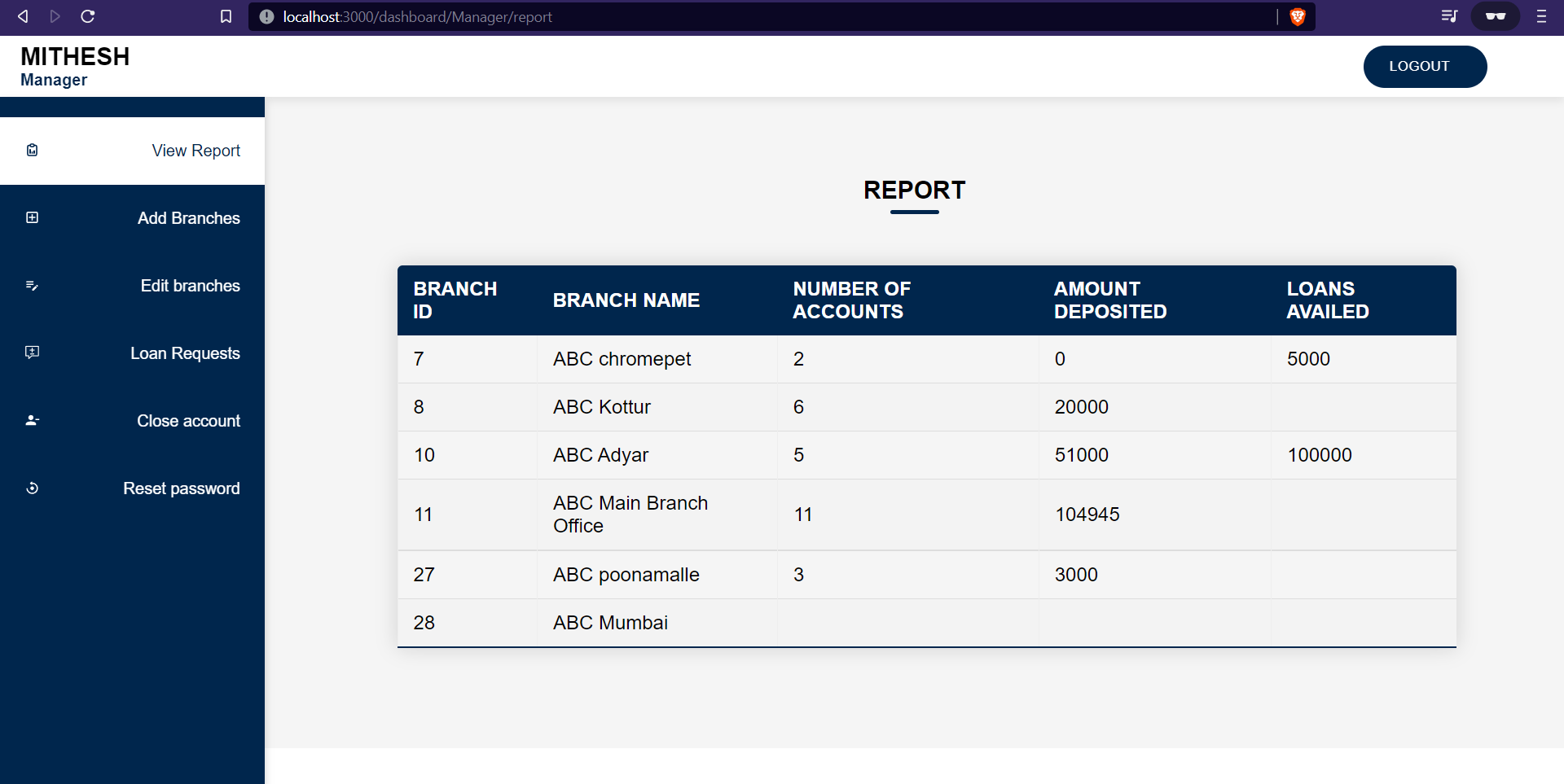
**ADDING, EDITING AND DELETING BRANCHES.**

**ADDING BRANCH**

The branch is created in the branch table with the name, address.

****

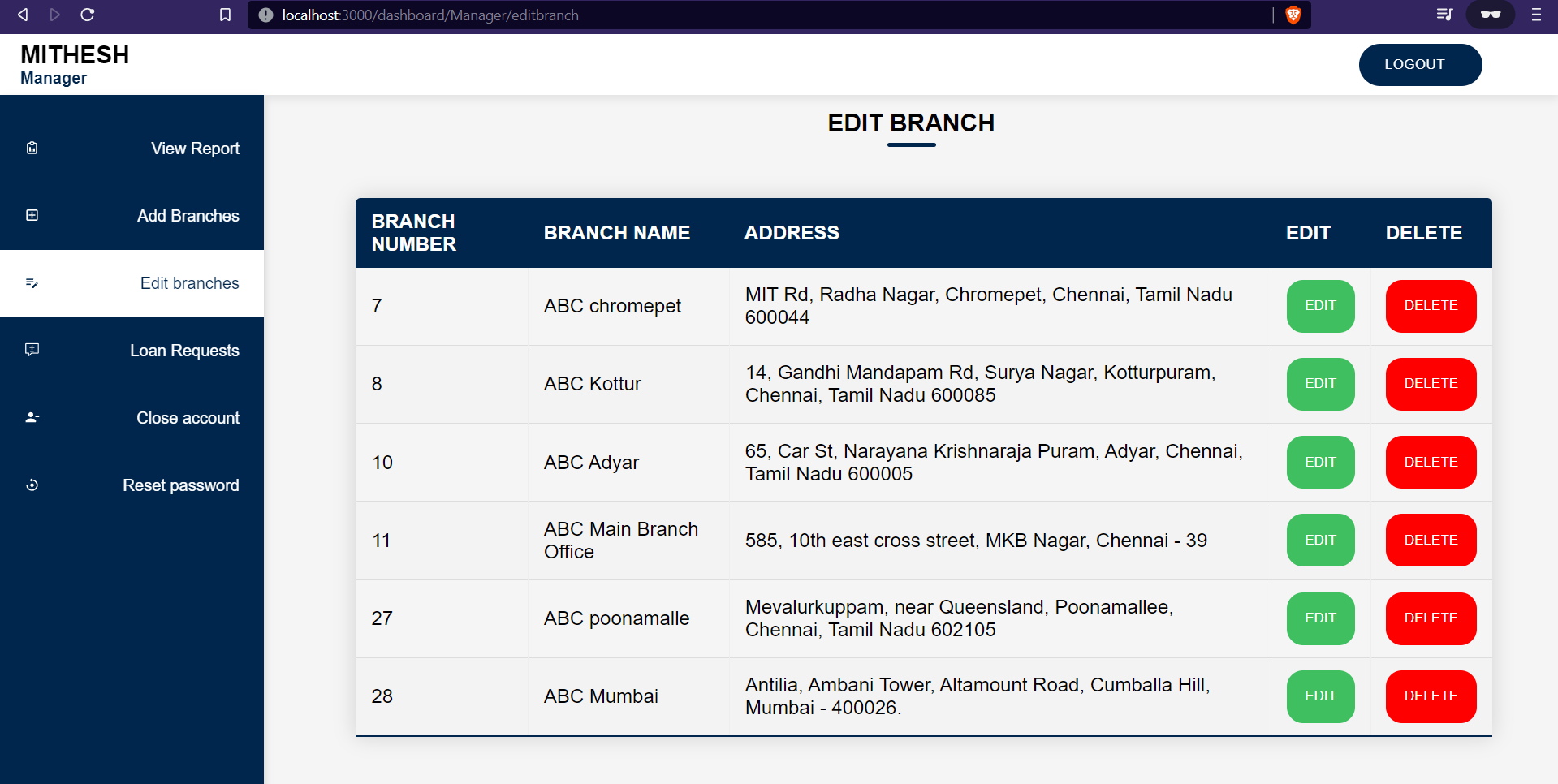
****

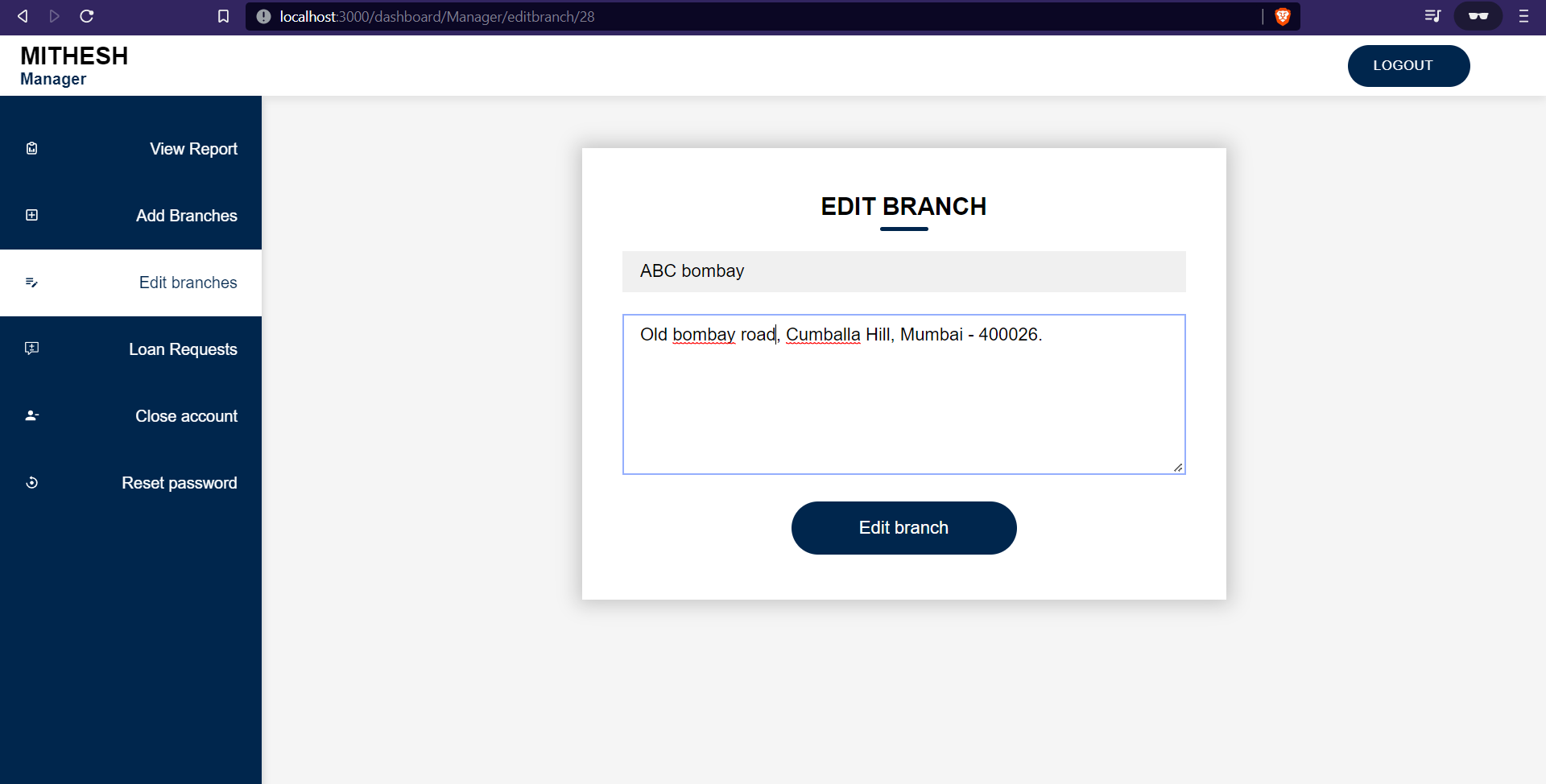
****

The report is generated with the help of group by function on accounts table by branch id, where the account number, and the balance amount in savings, business accounts and loan accounts are fetched and displayed in the report page.

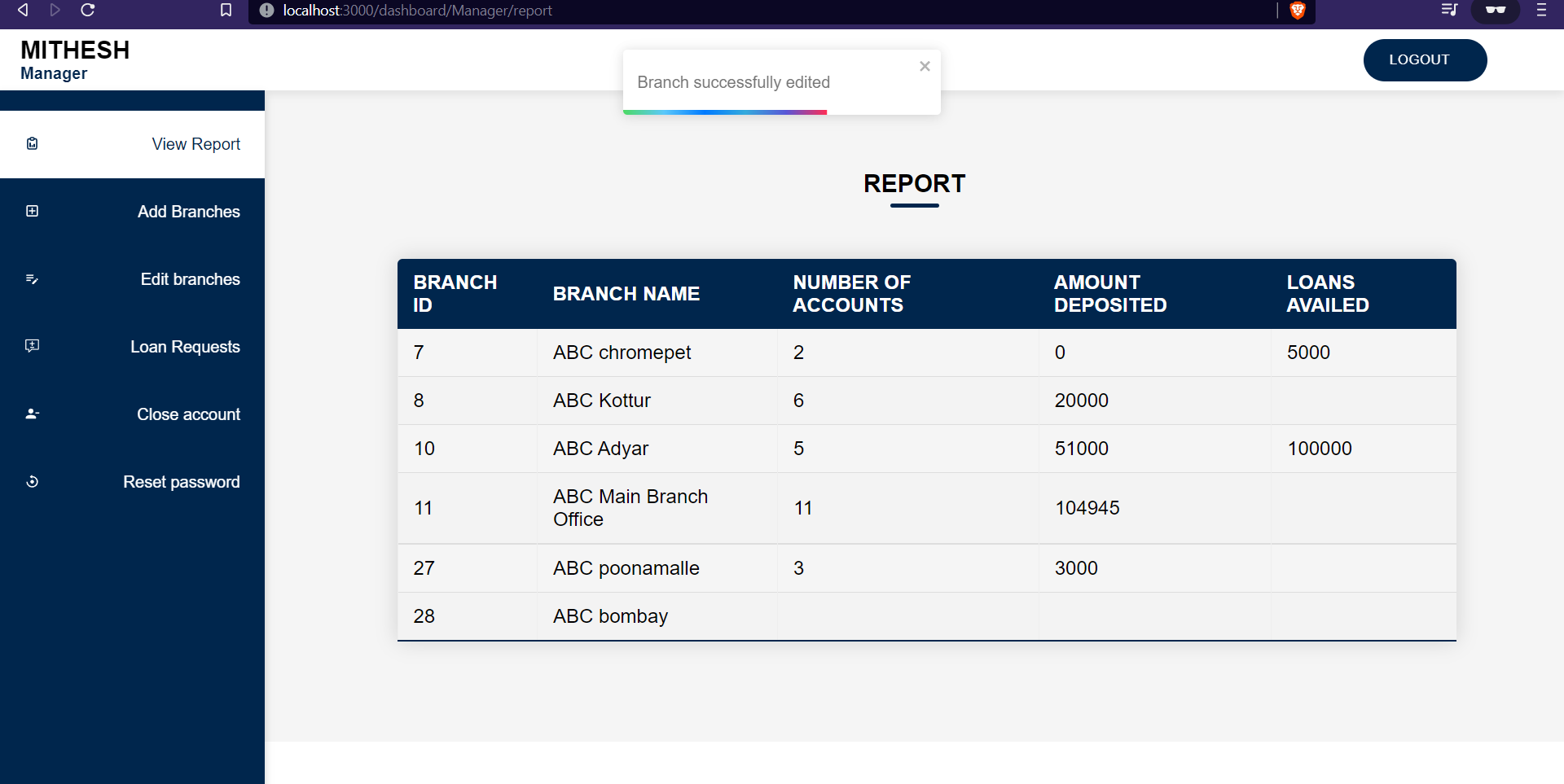
**EDITING BRANCHES**

Changing ABC Mumbai to Bombay and updating the address

****

****

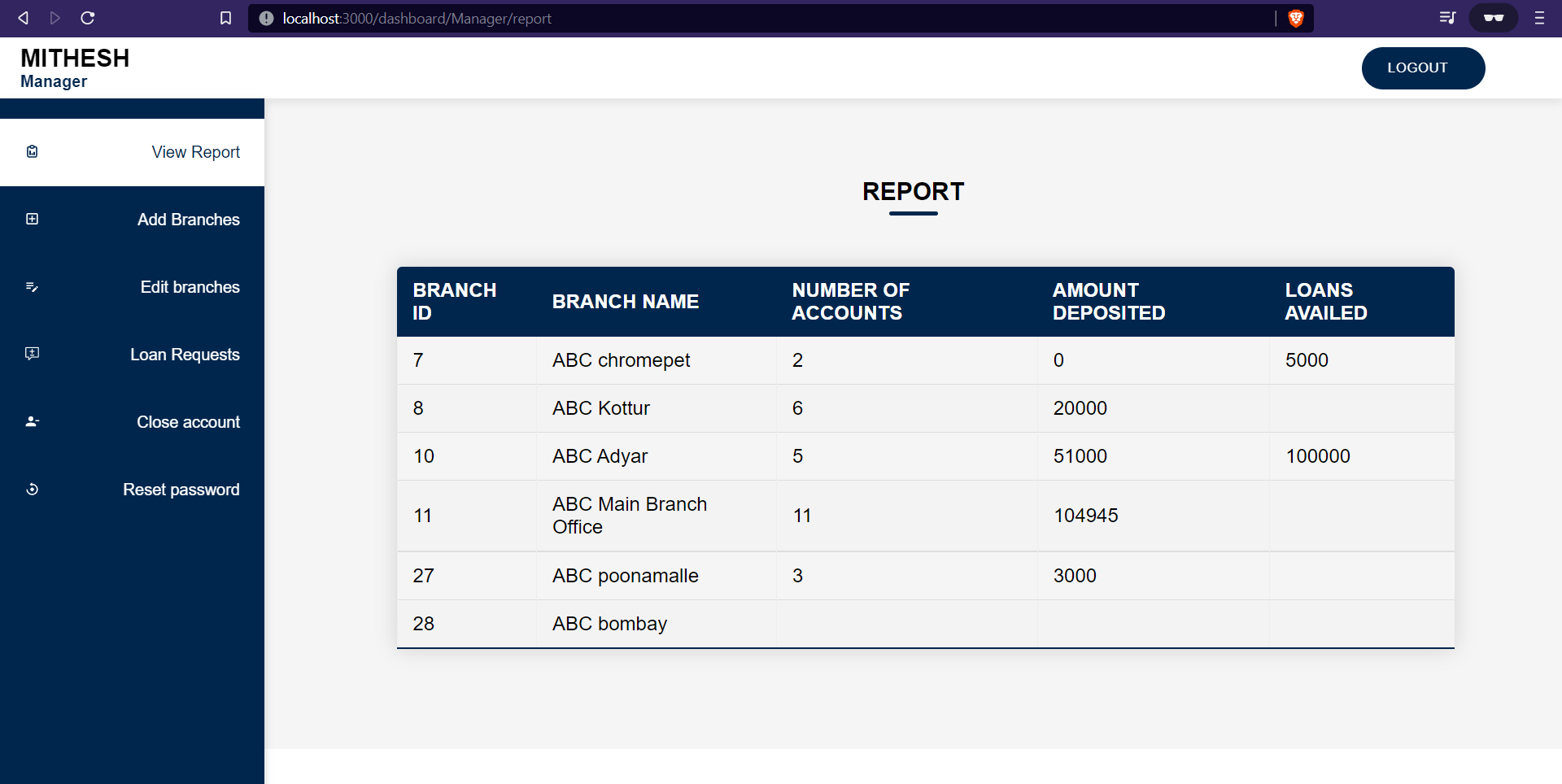
The branch table is updated where id is selected and the data like name and address is changed.

****

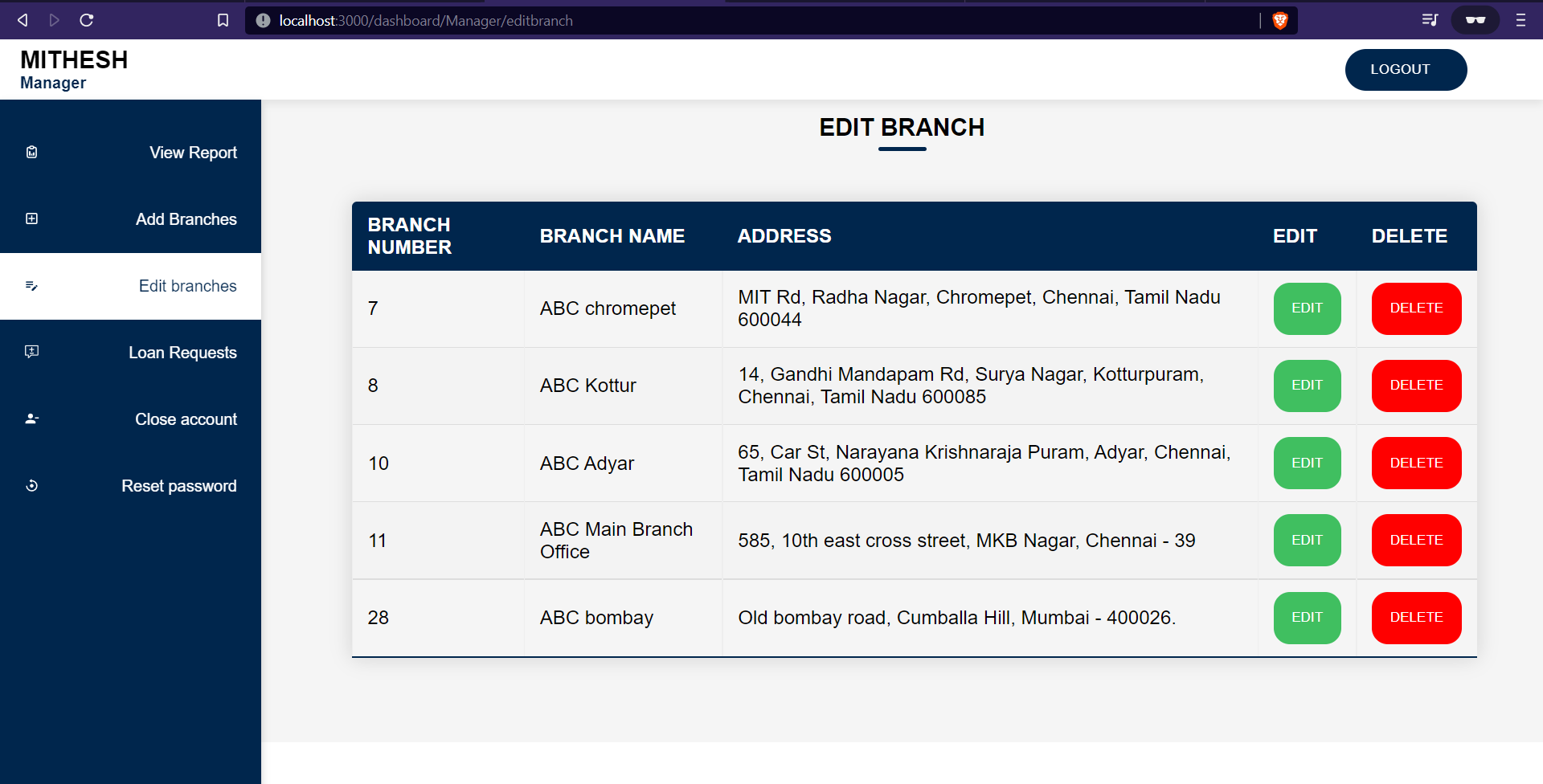
**DELETING BRANCHES**

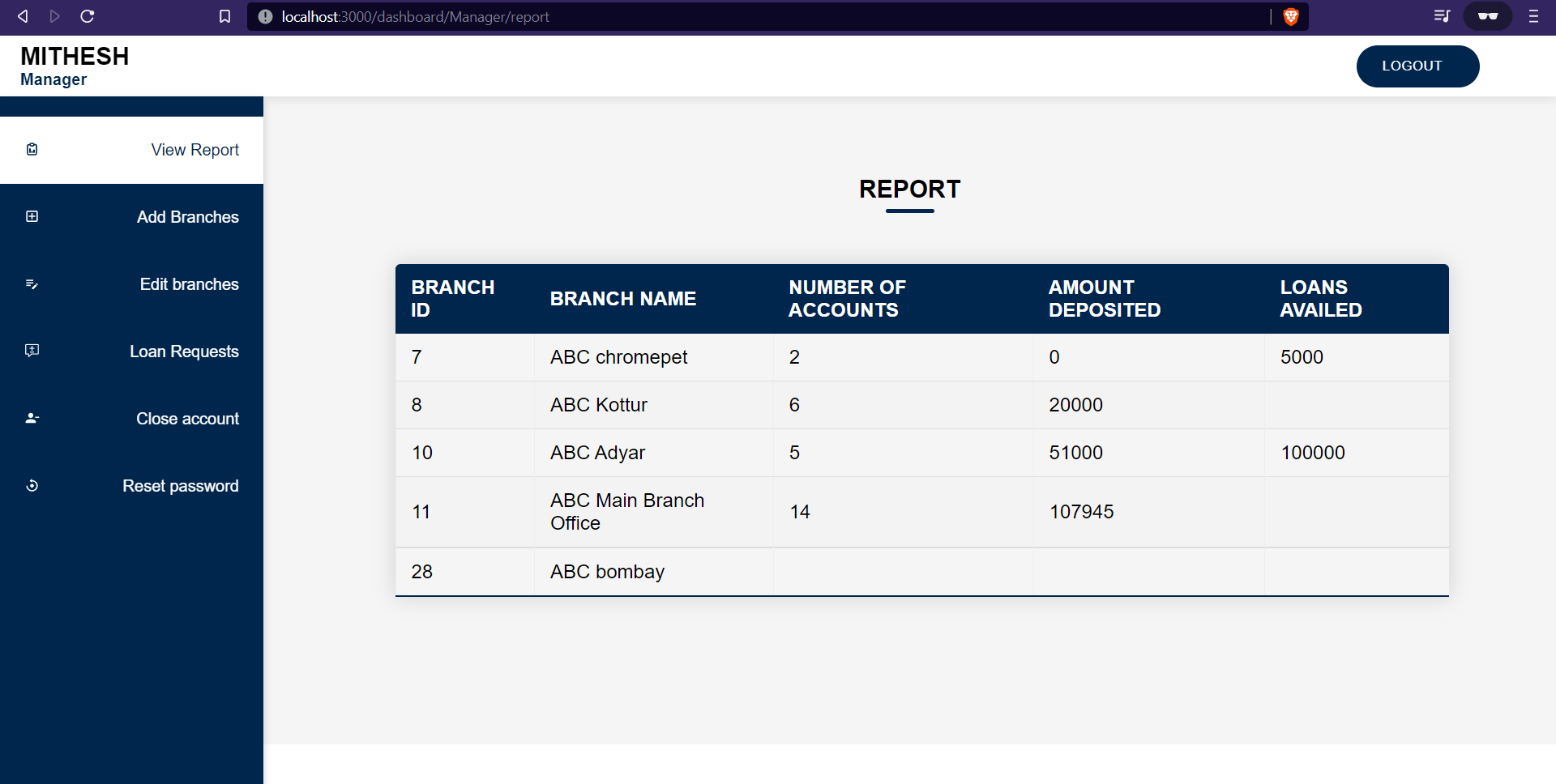
The deletion of branch involves transferring the accounts from the deleted account to the main branch office which is ABC MAIN BRANCH OFFICE. The deletion is done by updating the accounts table, setting branch ID = 11 where the deleted branch ID is selected. Here the branch with branch id – 11 is the main branch office.

**BEFORE**



**AFTER**





The main branch can’t get deleted.

