

# Bank API's Full stack Web Page

## Bank API's Backend

### MicroServices:

### UserService:

### Model:

User ID: Unique.

Username: Unique.

Email: Email address.

Address: Useraddress.

### EndPoints:

"http://localhost:8081/user"

**GET /user:** Retrieve users information

The screenshot shows a REST client interface with a sidebar on the left listing various API endpoints under 'BankAPI's'. The 'UserService' section is expanded, showing endpoints like 'register', 'login', 'UpdateUser', 'getAllUsers', and 'getUserById'. The 'getAllUsers' endpoint is selected, and a GET request is shown with the URL 'http://localhost:8081/user'. The response is a 200 OK status with a response time of 89 ms and a size of 1.62 KB. The response body is displayed in JSON format, showing an array of three user objects.

```
79  ],
80  {
81    "userId": 14,
82    "username": "Some123",
83    "password": "Some@123",
84    "address": "some thing"
85  },
86  {
87    "userId": 15,
88    "username": "HariHarsha",
89    "password": "Hari@123",
90    "address": "Some where"
91  },
92  {
93    "userId": 16,
94    "username": "Mouna",
95    "password": "Mouna123",
96    "address": "1234 Elm Street"
97  }
98  ]
```

"http://localhost:8081/user/{id}"

## GET /users/{id}: Retrieve user information by username.

BankAPI's / UserService / getUserById

GET http://localhost:8081/user/16

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (8) Test Results

200 OK • 379 ms • 335 B

Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "userId": 16,
3   "username": "Mouna",
4   "password": "Mouna123",
5   "address": "1234 Elm Street"
6 }
```

"http://localhost:8081/user/register" body with user information

## POST /user/register: Register a new user.

space New Import

BankAPI's / UserService / register

POST http://localhost:8081/user/register

Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "username": "Mouna",
3   "password": "Mouna123",
4   "address": "1234 Elm Street"
5 }
6
```

Body Cookies Headers (8) Test Results

201 Created • 659 ms • 340 B

Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "userId": 16,
3   "username": "Mouna",
4   "password": "Mouna123",
5   "address": "1234 Elm Street"
6 }
```

"<http://localhost:8081/user/login>" body with user information

**POST /user/login:** Login user.

The screenshot shows a REST client interface with a sidebar on the left listing API endpoints under 'BankAPI's / UserService'. The 'login' endpoint is selected. The main panel shows a POST request to 'http://localhost:8081/user/login'. The request body is a JSON object: 

```
{  "username": "Mouna",  "password": "Mouna123"}
```

. The response is displayed in the 'Body' tab, showing a 200 OK status with a response time of 61 ms and a body size of 389 B. The response body is a JSON object: 

```
{  "user": {    "userId": 16,    "username": "Mouna",    "password": "Mouna123",    "address": "1234 Elm Street"  },  "token": "User_2024-10-28T10:25:16.191198600"}
```

"<http://localhost:8081/user/{id}>"

**PUT /users/{id}:** Update user profile.

The screenshot shows a REST client interface with a sidebar on the left listing API endpoints under 'BankAPI's / UserService'. The 'UpdateUser' endpoint is selected. The main panel shows a PUT request to 'http://localhost:8081/user/16'. The request body is a JSON object: 

```
{  "username": "MounaReddy",  "password": "Mouna123",  "address": "Bangalore"}
```

. The response is displayed in the 'Body' tab, showing a 200 OK status with a response time of 48 ms and a body size of 334 B. The response body is a JSON object: 

```
{  "userId": 16,  "username": "MounaReddy",  "password": "Mouna123",  "address": "Bangalore"}
```

# AccountService:

## Model:

Account ID: Unique.

Account Name: Name of the bank.

Account Type: Type of the account.

Balance: Balance.

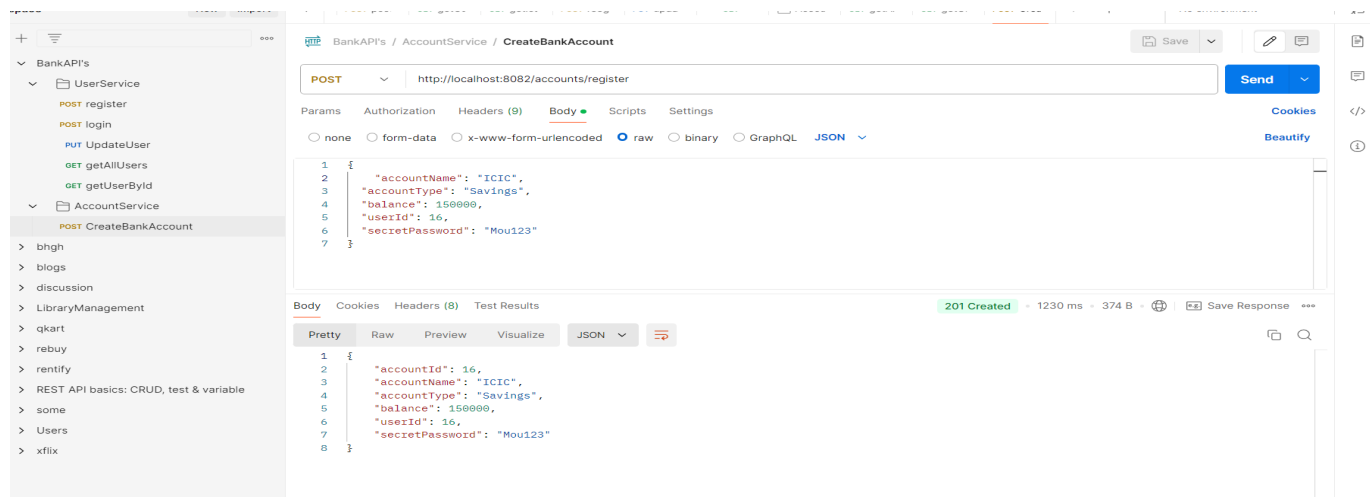
User Id: User Id.

Secret password: Some password.

## EndPoints:

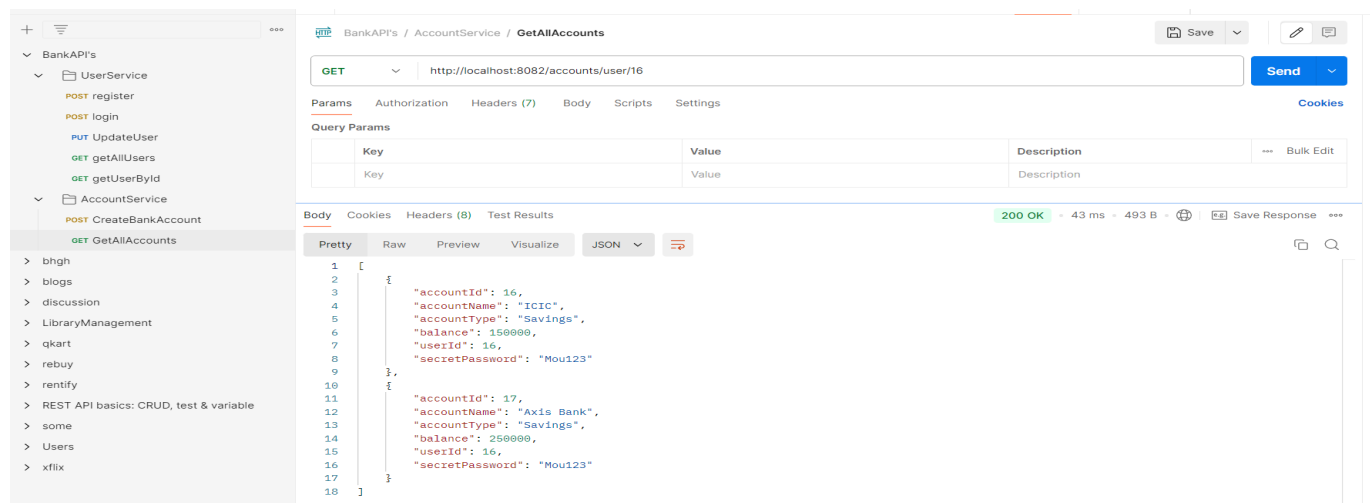
"<http://localhost:8082/accounts/register>" body with bank details

**Post /accounts/register:** Register information of bank



"<http://localhost:8082/accounts/user/{id}>"

**GET /users/{id}:** Retrieve user's bank information by userId.



# AccountService:

## Model:

Transaction ID: Unique.

From Account Id: Id of the From bank.

Transaction Type: Type of the Transaction.

ToAccount Id: Id of the To bank.

Amount: Amount to send.

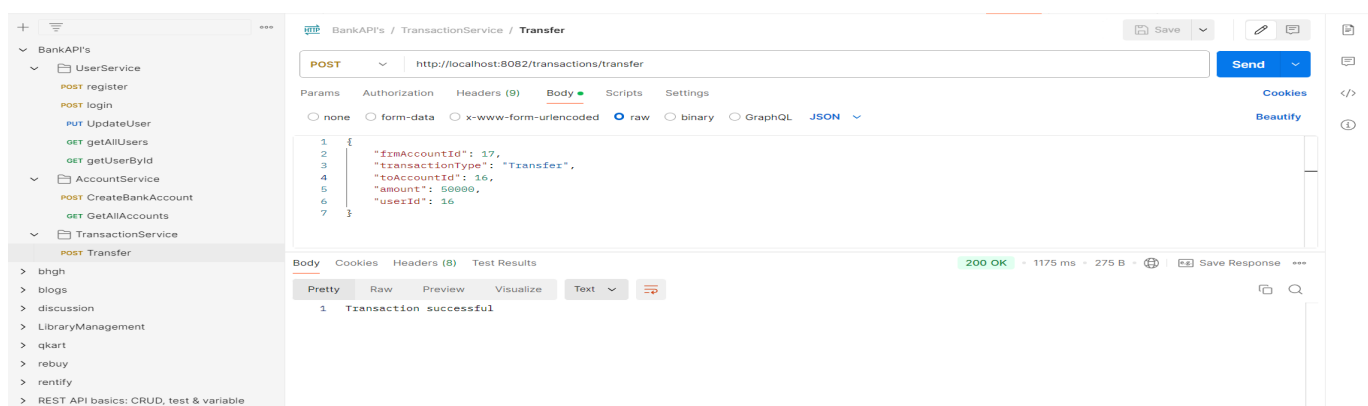
User Id: User Id.

Transaction Date: Date of the transaction done.

## EndPoints:

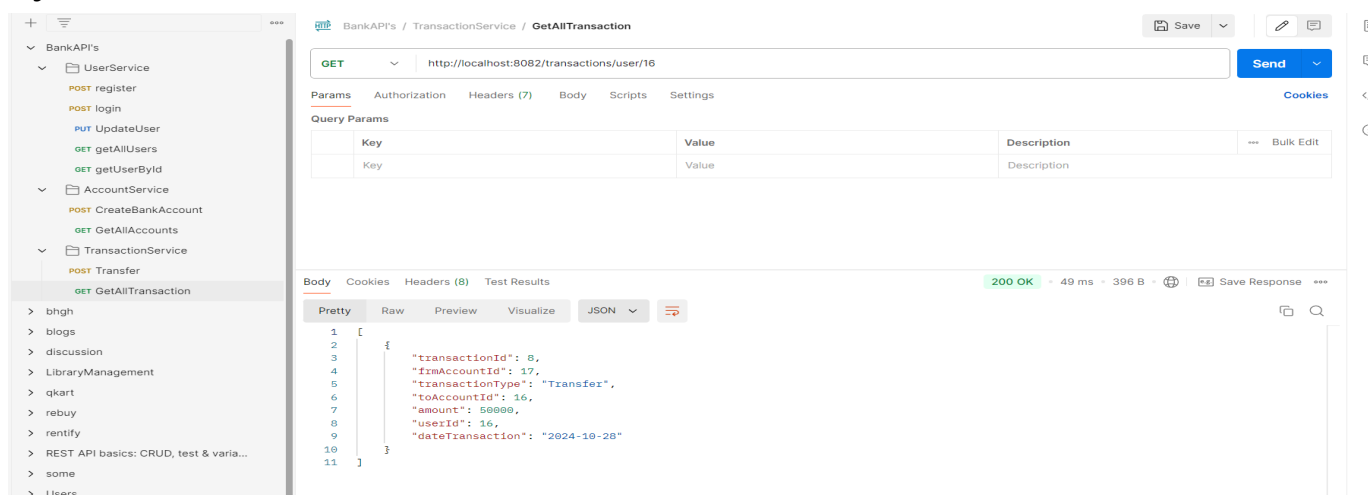
"<http://localhost:8082/transactions/register>" body with transaction details

**Post /accounts/register: Make Transaction.**



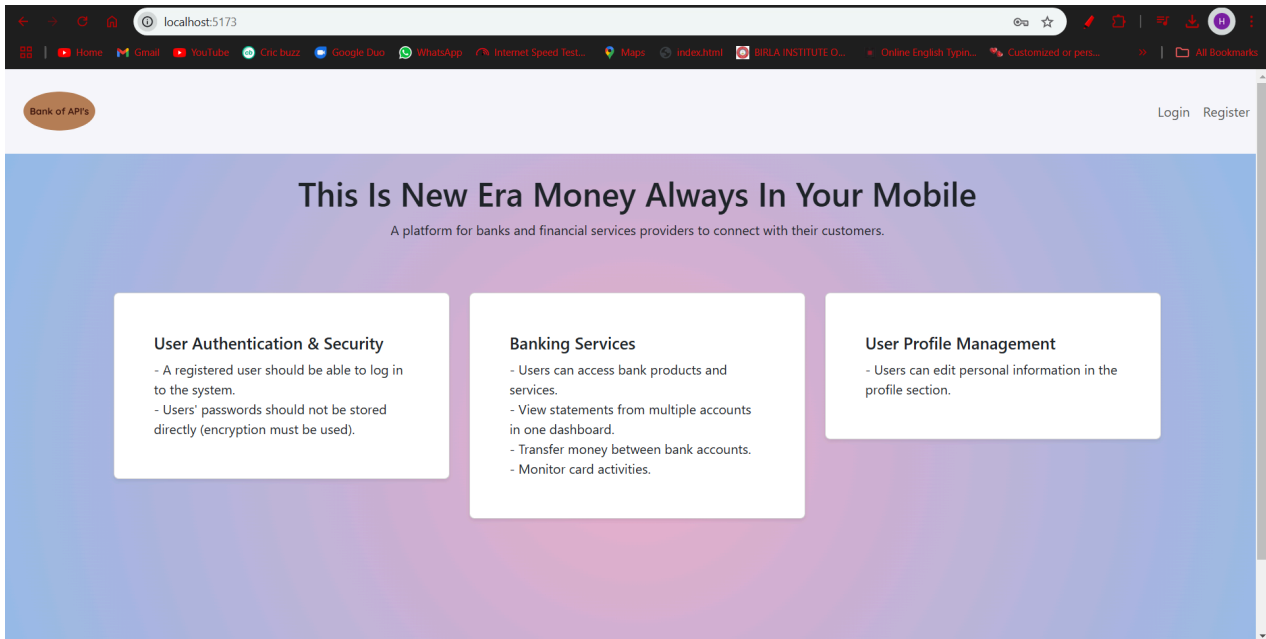
"<http://localhost:8082/transactions/user/{id}>"

**GET /users/{id}: Retrieve user's Transactions information by userId.**



# Bank API's Frontend

## Home page without login:



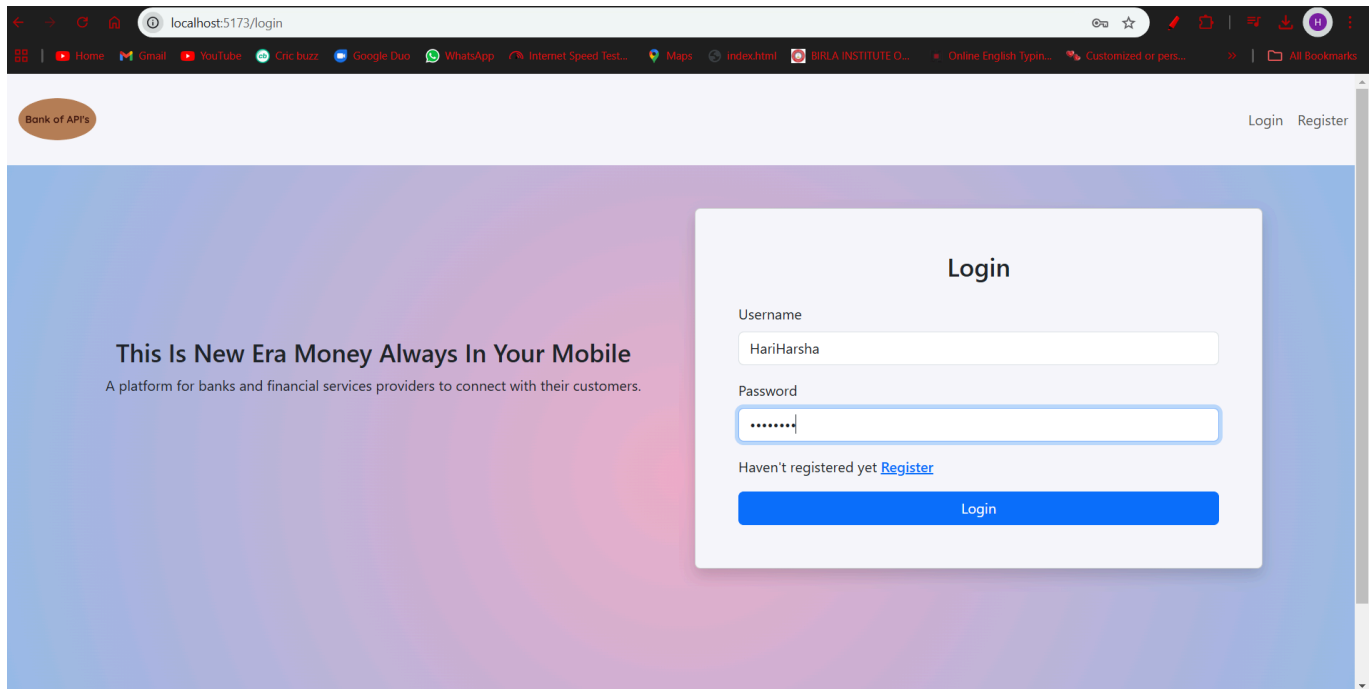
## Register Page:

The screenshot shows the register page of the Bank API's Frontend. The browser address bar displays 'localhost:5173/register'. The page features a header with the 'Bank of API's' logo on the left and 'Login Register' links on the right. The main content area has a blue and purple gradient background. On the left, it says 'This Is New Era Money Always In Your Mobile' followed by the subtitle 'A platform for banks and financial services providers to connect with their customers.' On the right, there is a white box with the title 'Register' and the following form fields:

- Username:
- Password:
- Confirm Password:
- Address:

Below the form fields, there is a link 'Have registered [Login](#)' and a blue 'Register' button.

# Login Page:



The screenshot shows a web browser at localhost:5173/login. The page has a light blue and purple gradient background. On the left, there's a text block: "This Is New Era Money Always In Your Mobile" followed by "A platform for banks and financial services providers to connect with their customers." On the right, there's a white login form titled "Login". The form contains fields for "Username" (filled with "HariHarsha") and "Password" (filled with "\*\*\*\*\*"). Below the password field is a link "Haven't registered yet [Register](#)". At the bottom of the form is a blue "Login" button. The browser's address bar and bookmarks are visible at the top.

Bank of APIs

Login Register

**This Is New Era Money Always In Your Mobile**  
A platform for banks and financial services providers to connect with their customers.

**Login**

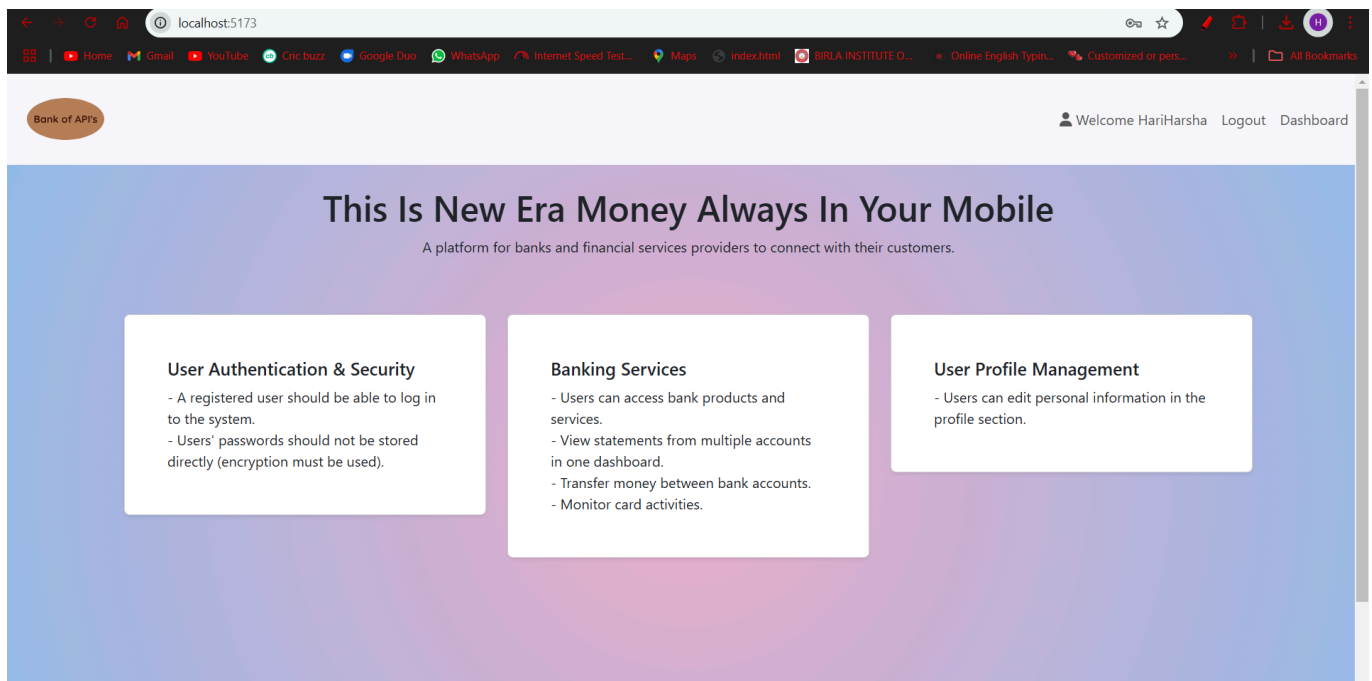
Username  
HariHarsha

Password  
\*\*\*\*\*

Haven't registered yet [Register](#)

Login

# Home Page After Login:



The screenshot shows the home page after login. The browser address bar shows localhost:5173. The page has the same gradient background. The top right now says "Welcome HariHarsha Logout Dashboard". The main heading is "This Is New Era Money Always In Your Mobile" with the same subtext. Below this, there are three white boxes with titles and bullet points:

- User Authentication & Security**
  - A registered user should be able to log in to the system.
  - Users' passwords should not be stored directly (encryption must be used).
- Banking Services**
  - Users can access bank products and services.
  - View statements from multiple accounts in one dashboard.
  - Transfer money between bank accounts.
  - Monitor card activities.
- User Profile Management**
  - Users can edit personal information in the profile section.

Bank of APIs

Welcome HariHarsha Logout Dashboard

**This Is New Era Money Always In Your Mobile**  
A platform for banks and financial services providers to connect with their customers.

**User Authentication & Security**

- A registered user should be able to log in to the system.
- Users' passwords should not be stored directly (encryption must be used).

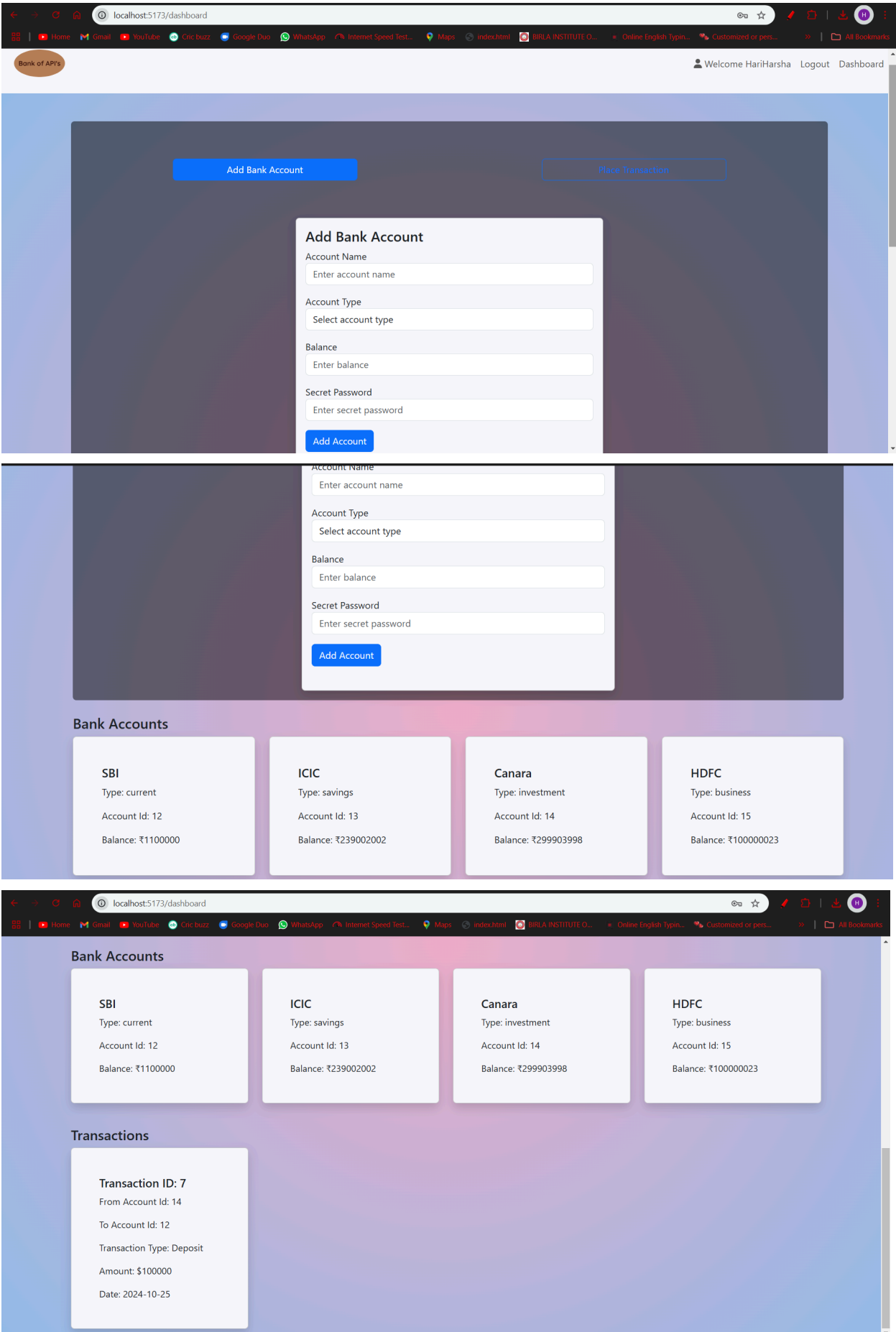
**Banking Services**

- Users can access bank products and services.
- View statements from multiple accounts in one dashboard.
- Transfer money between bank accounts.
- Monitor card activities.

**User Profile Management**

- Users can edit personal information in the profile section.

# DashBoard:





# Profile Page:

