**CPSC 8720-01\_24SP**

**Internet Programming**

**Spring 2024**

**PROJECT REPORT**

**“Online Banking System Project”**

**Student Name: Kodanda Ramu Yanamalachintala**

Project Report for Online Banking Application

Table of Contents

[Introduction 2](#_Toc165833505)

[Project Implementation 2](#_Toc165833506)

[Roles Implemented: 2](#_Toc165833507)

[Manager Role: 2](#_Toc165833508)

[Customer Role: 2](#_Toc165833509)

[Teller Role: 2](#_Toc165833510)

[Database Design 2](#_Toc165833511)

[Security and User Experience 3](#_Toc165833512)

[Special attention was given to security and user experience: 3](#_Toc165833513)

[ASP.NET Identity: 3](#_Toc165833514)

[Data Protection: 3](#_Toc165833515)

[User-Friendly Interface: 3](#_Toc165833516)

[Skills Applied 3](#_Toc165833517)

[Conclusion 3](#_Toc165833518)

[Screenshots 3](#_Toc165833519)

[Home: 4](#_Toc165833520)

[Registration: 4](#_Toc165833521)

[Login: 5](#_Toc165833522)

[Forget Password: 6](#_Toc165833523)

[Resend Email Confirmation: 6](#_Toc165833524)

[Accounts screen for Teller role Login: 7](#_Toc165833525)

[Withdraw and Deposit screen along with Account Profile: 7](#_Toc165833526)

[Account Summary: 8](#_Toc165833527)

[Bill Pay: 9](#_Toc165833528)

[Payee: 9](#_Toc165833529)

[Add/Update Payee: 10](#_Toc165833530)

[Create Bill Pay: 10](#_Toc165833531)

# Introduction

This project was developed as part of the Internet Programming class to apply the skills learned during the course. The application is an online banking system created using ASP.NET Core MVC and C#. It simulates various banking operations and introduces a new level of interaction between the bank and its customers through an online platform.

# Project Implementation

This project was developed using AspNet tables for identification and used AspNetRoles to differentiate the user roles.

## Roles Implemented:

### Manager Role:

Currently, there are no specific functionalities implemented for this role. Future implementations can include overseeing teller activities, generating reports, and managing customer complaints.

### Customer Role:

Registration and Login: Customers can register on the platform and log in to access their accounts.

Bill Pay: After logging in, customers can navigate to the Bill Pay page to add payees and make payments.

Account Management: Customers can update personal information and review their account balances and transaction history.

### Teller Role:

Account Search: Tellers can search for customer accounts using the account number.

Transaction Management: Tellers can perform deposit and withdrawal transactions on behalf of the customers.

## Database Design

Below are the main tables were created to support the application functionalities:

Customers: Stores information of the registered customers.

Account: Contains bank account details of each customer.

AccountActivities: Logs deposit and withdrawal activities.

Payees: Holds information about payees added by customers.

PaymentHistories: Records details of the payments made by customers.

Persons: Stores the information about Persons.

In addition to the above tables, we added ASP.NET identity tables to handle the user authorization and authentication process and this improve the system’s performance and security.

## Security and User Experience

### Special attention was given to security and user experience:

To secure user data and transactions, data protection and secure login procedures were put in place. All features of the interface are easy to use because to its straightforward and user-friendly design.

ASP.NET Identity: ASP.NET Identity is used by the application to manage security. An API for user interface login features, such as secure password hashing, session management, and user authentication, is provided by this framework. In order to manage users, roles, and claims—all of which are kept in automatically produced identity tables like AspNetUsers, AspNetRoles, AspNetUserRoles, and AspNetUserClaims—ASP.NET Identity was integrated. In order to maintain the security of processes like registration, login, and user management, these tables assist in managing login credentials and access rights.

Data Protection: Sensitive data saved in the database is encrypted, and HTTPS is used to secure data while it is in transit, among other security precautions. By doing this, the client and server may be sure that all user data and transaction details are sent securely.

User-Friendly Interface: The application's user experience was given careful consideration throughout design, and it features a simple, clear interface that makes it simple for users to register, manage their accounts, and complete transactions..

## Skills Applied

This project utilized the following skills learned during the Internet Programming course:

1. ASP.NET Core MVC for building the application framework.
2. C# for backend processing.
3. Entity Framework Code first approach for database interaction.
4. HTML, CSS, and JavaScript for frontend development.

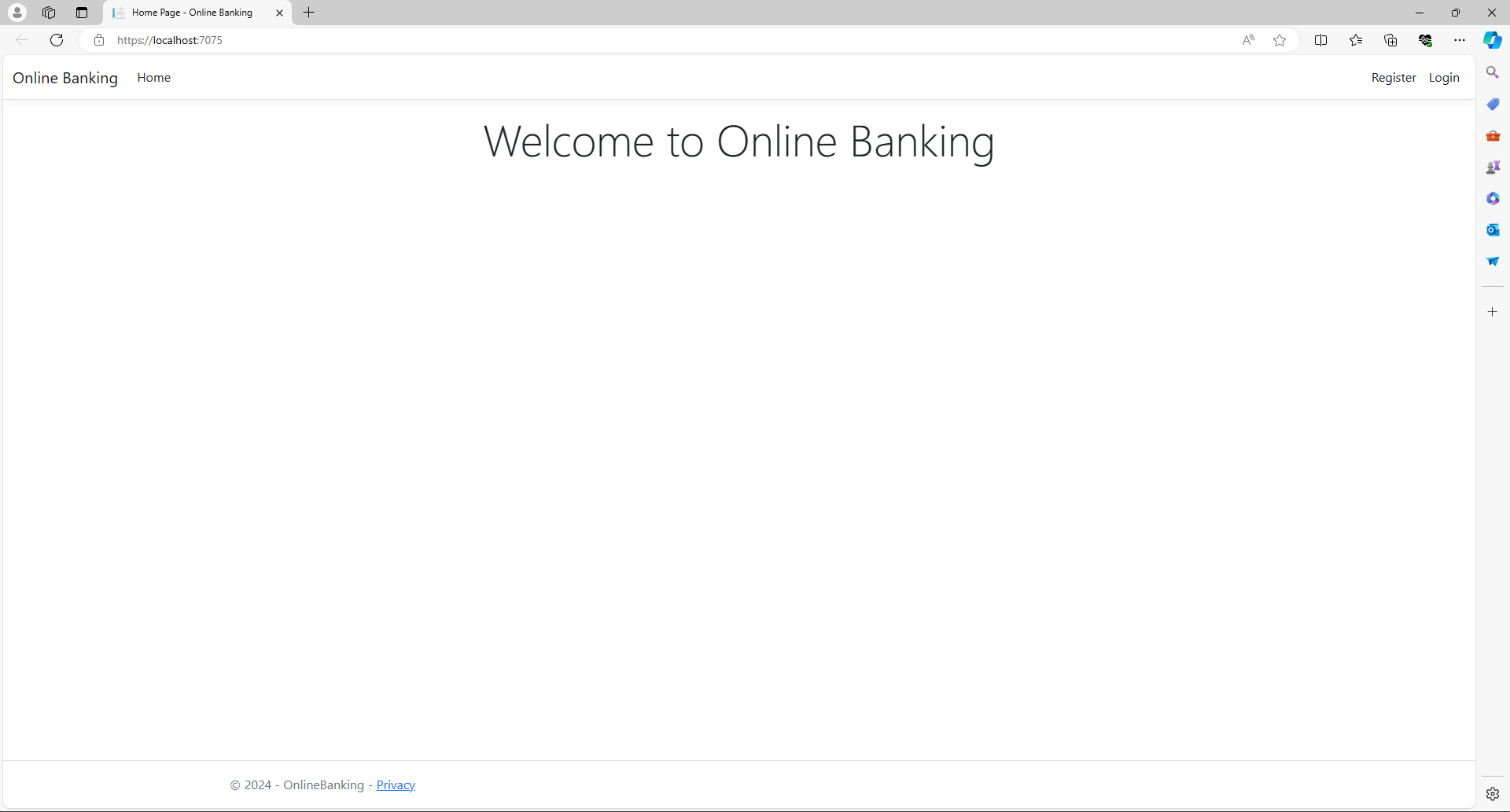
## Conclusion

This online banking application represents a comprehensive approach to applying theoretical knowledge in a practical, real-world project. While the basic functionalities have been implemented, the platform is designed to be scalable with potential future enhancements such as the addition of a manager role with extended capabilities.

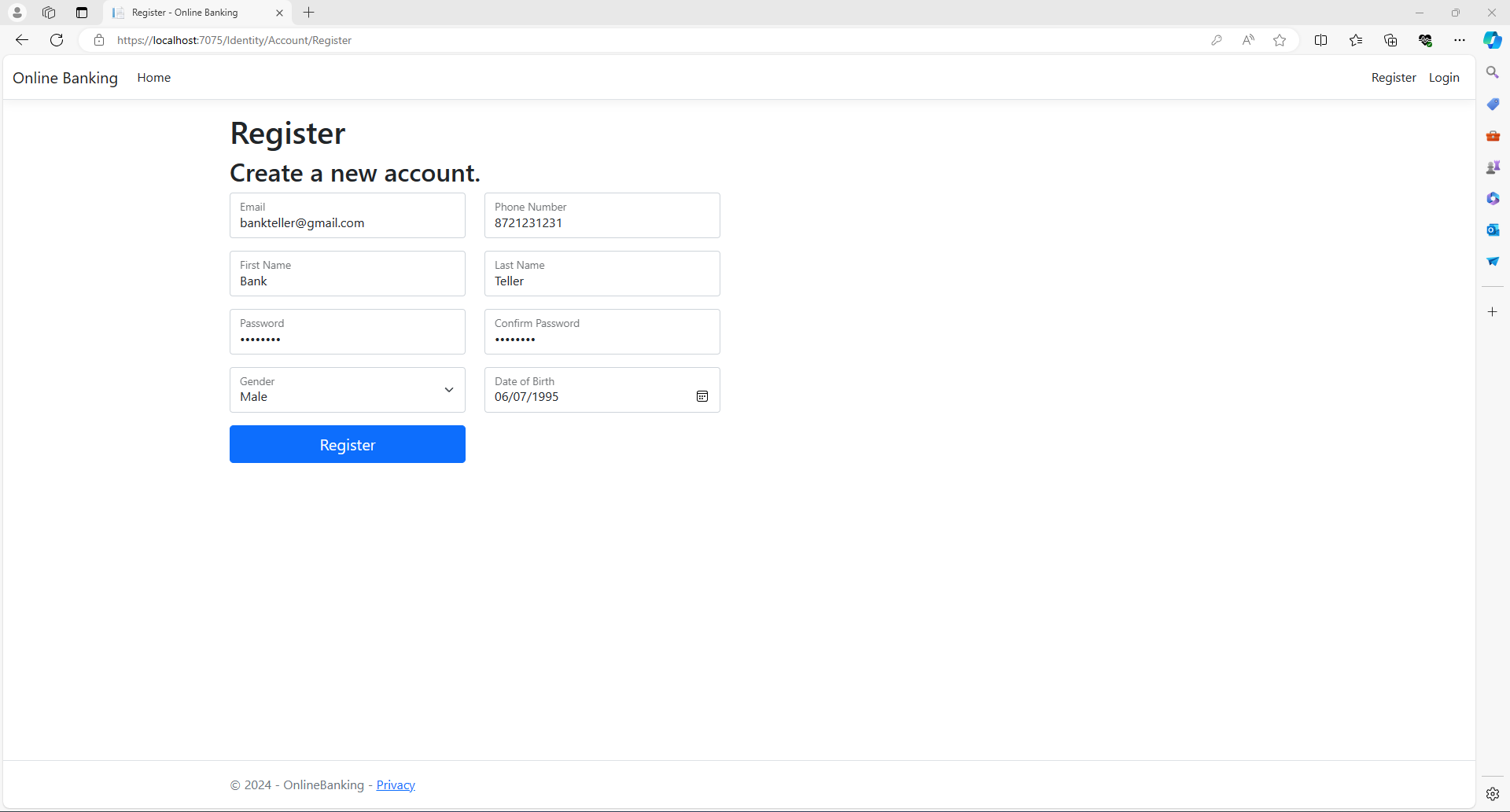
## Screenshots

Please find the below screenshots for each page:

### Home:



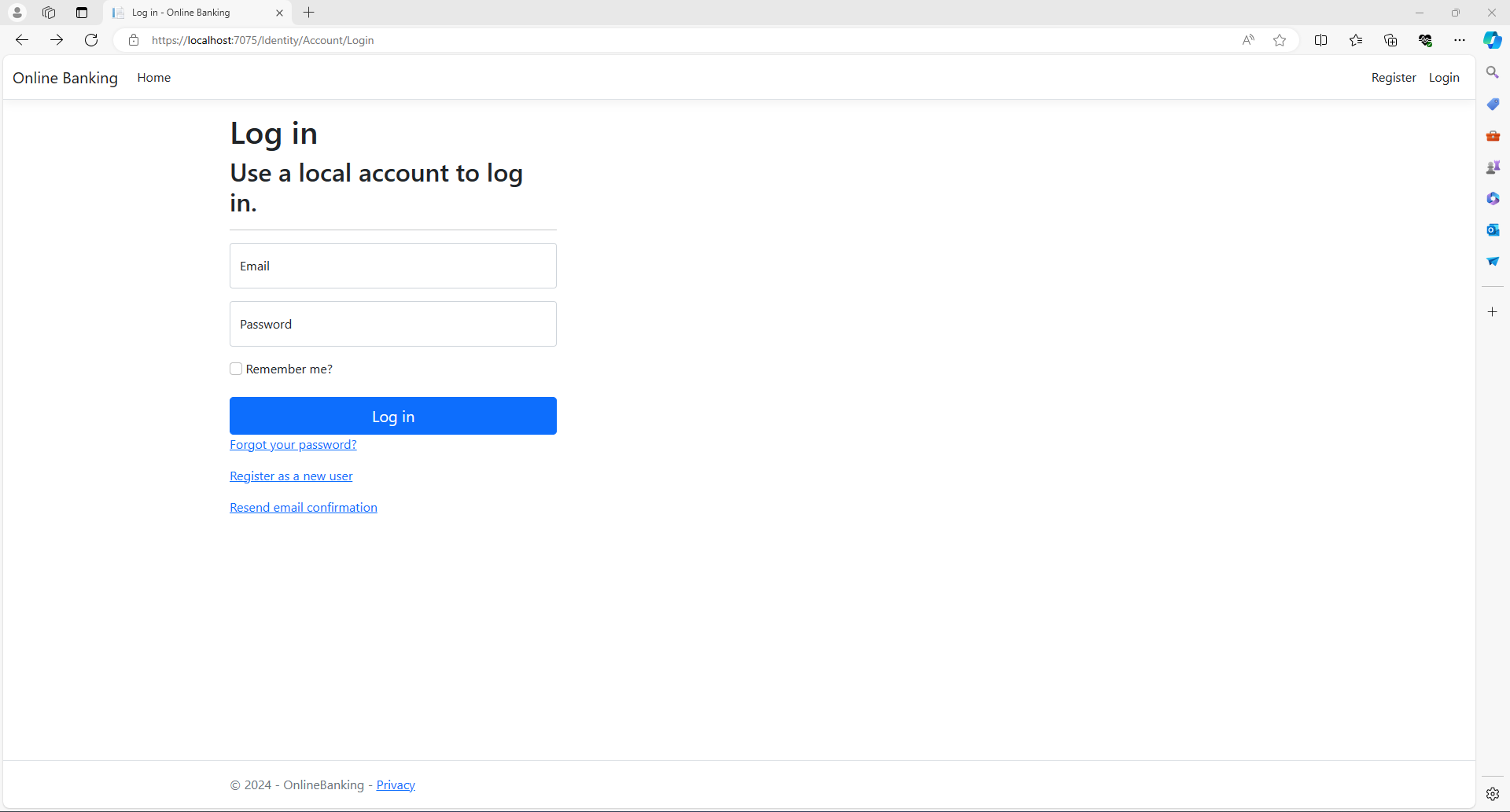
### Registration:



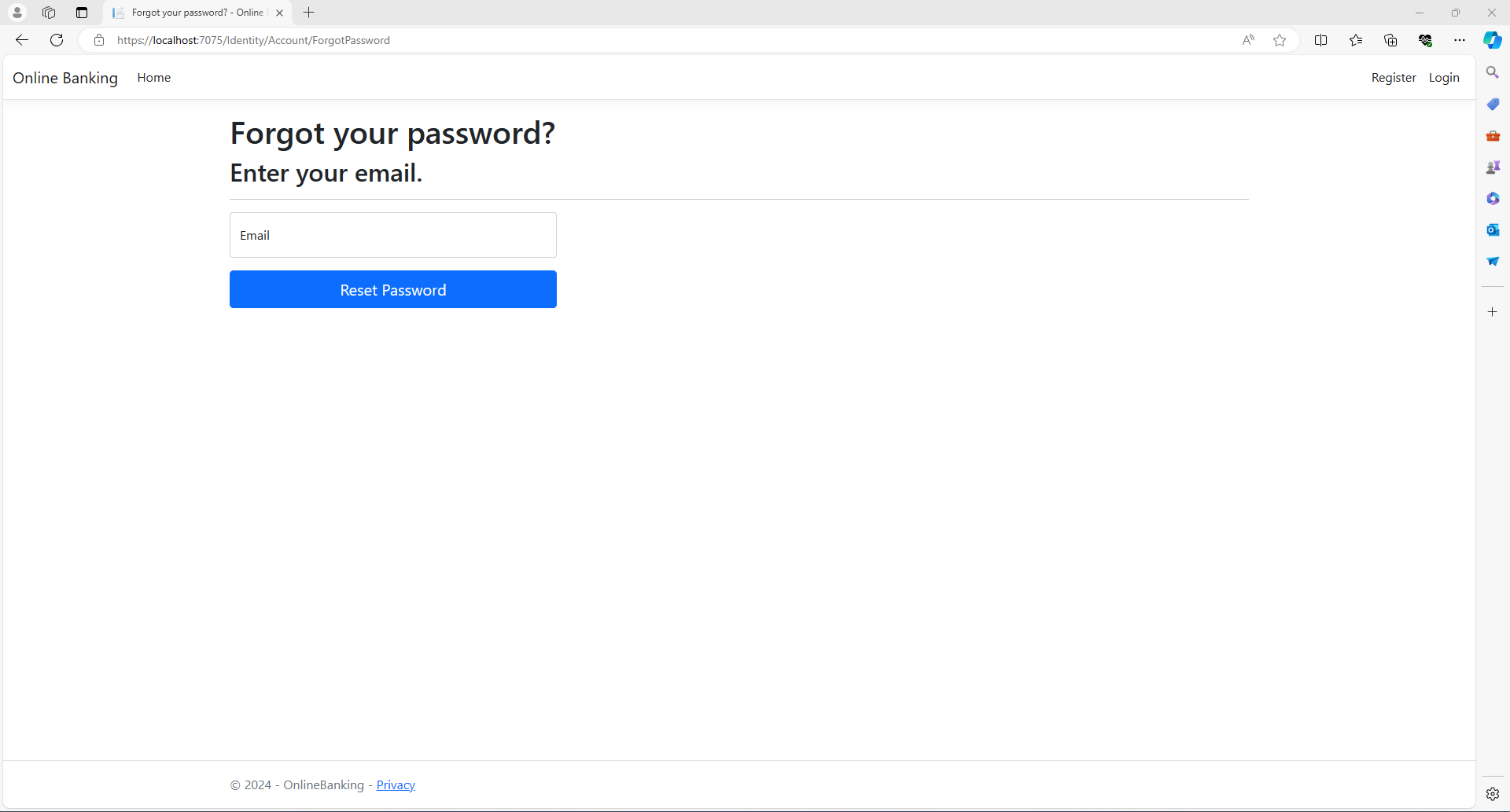
### Login:

As the registration happens as a customer, I changed the role of the Bank Teller user to Teller role using the below command.

UPDATE AspNetUserRoles SET RoleId = '72380C80-A6B5-4168-BFAB-DC7BBC806A46' WHERE USerId = '0782ac5a-4bb0-488b-aa3d-2ceac61978fe'

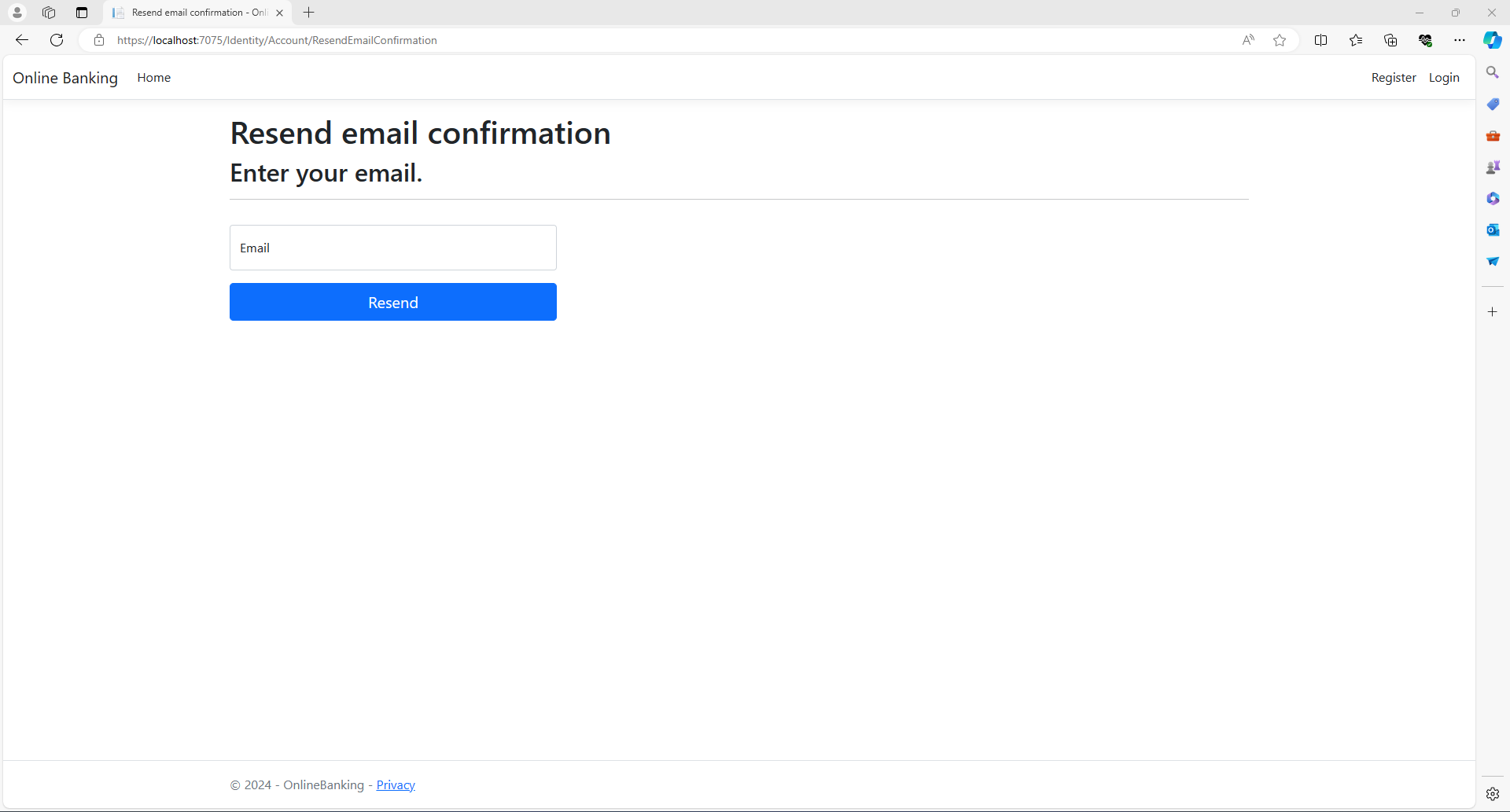


### Forget Password:



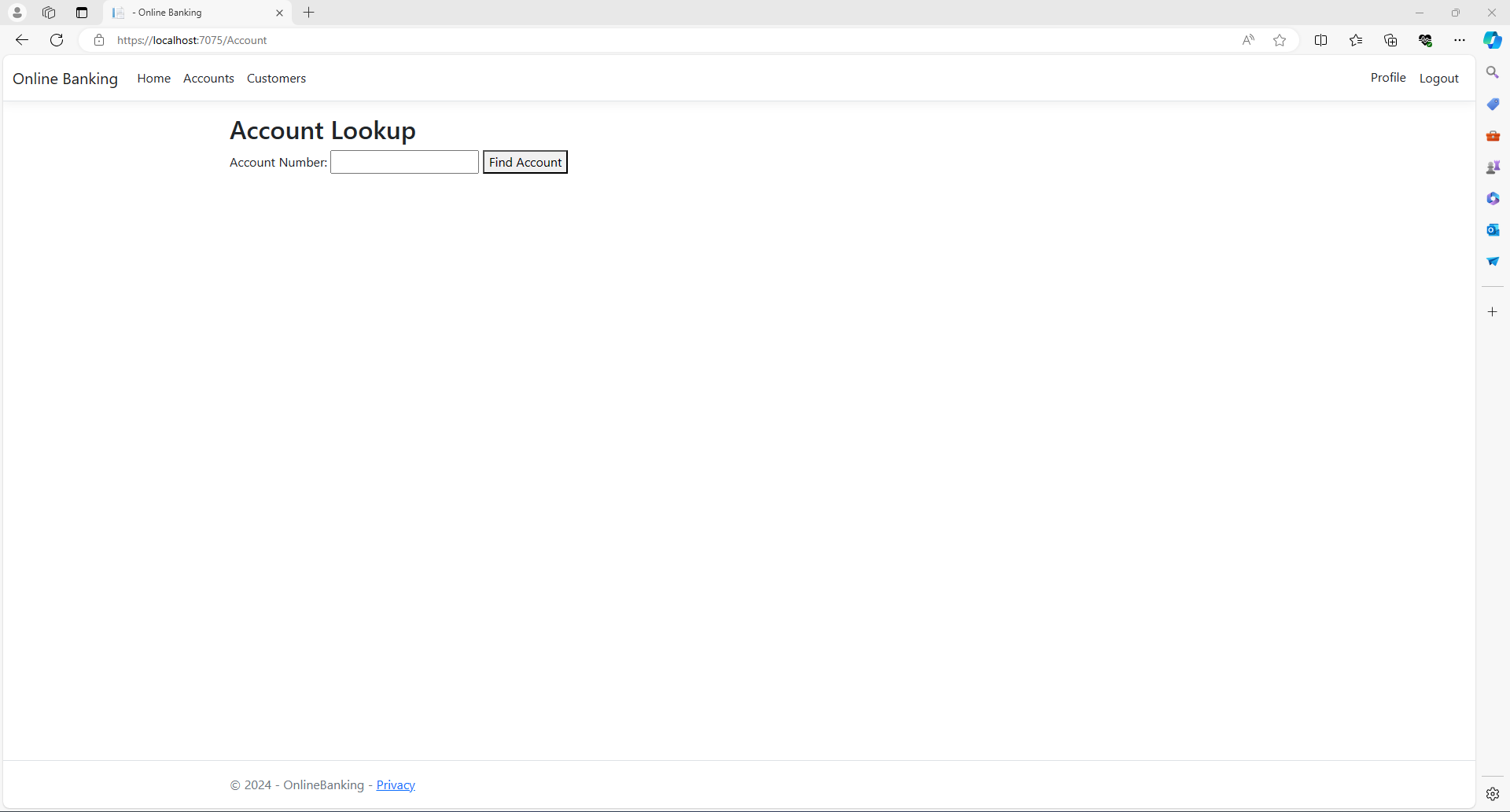
### Resend Email Confirmation:

There is a dummy resend email confirmation screen that we can navigate from Login Screen.



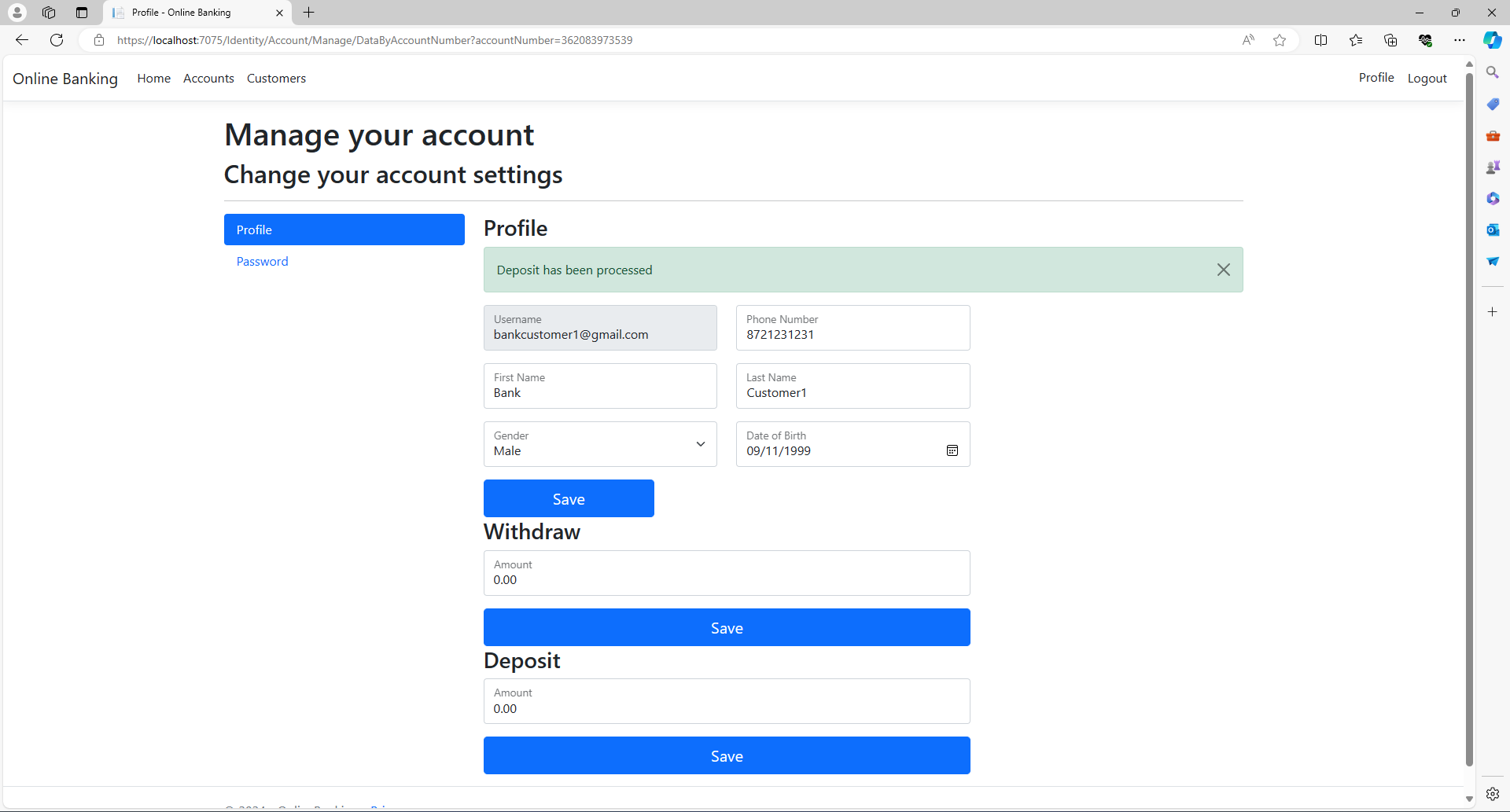
### Accounts screen for Teller role Login:

In this screen, the teller can enter the account number and click on Find Account button which takes the teller to the account holder profile and withdraw deposit screen.

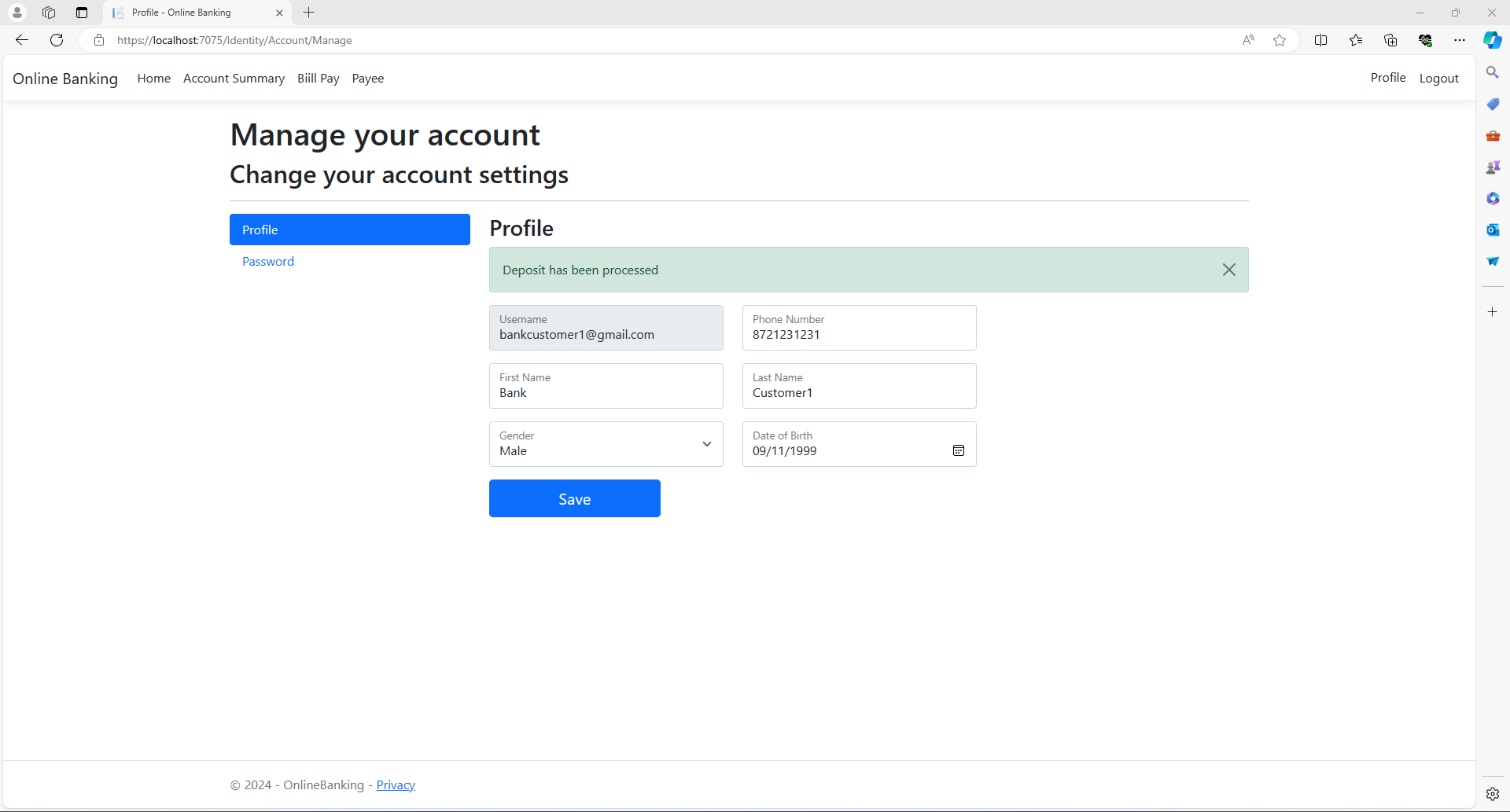


### Withdraw and Deposit screen along with Account Profile:

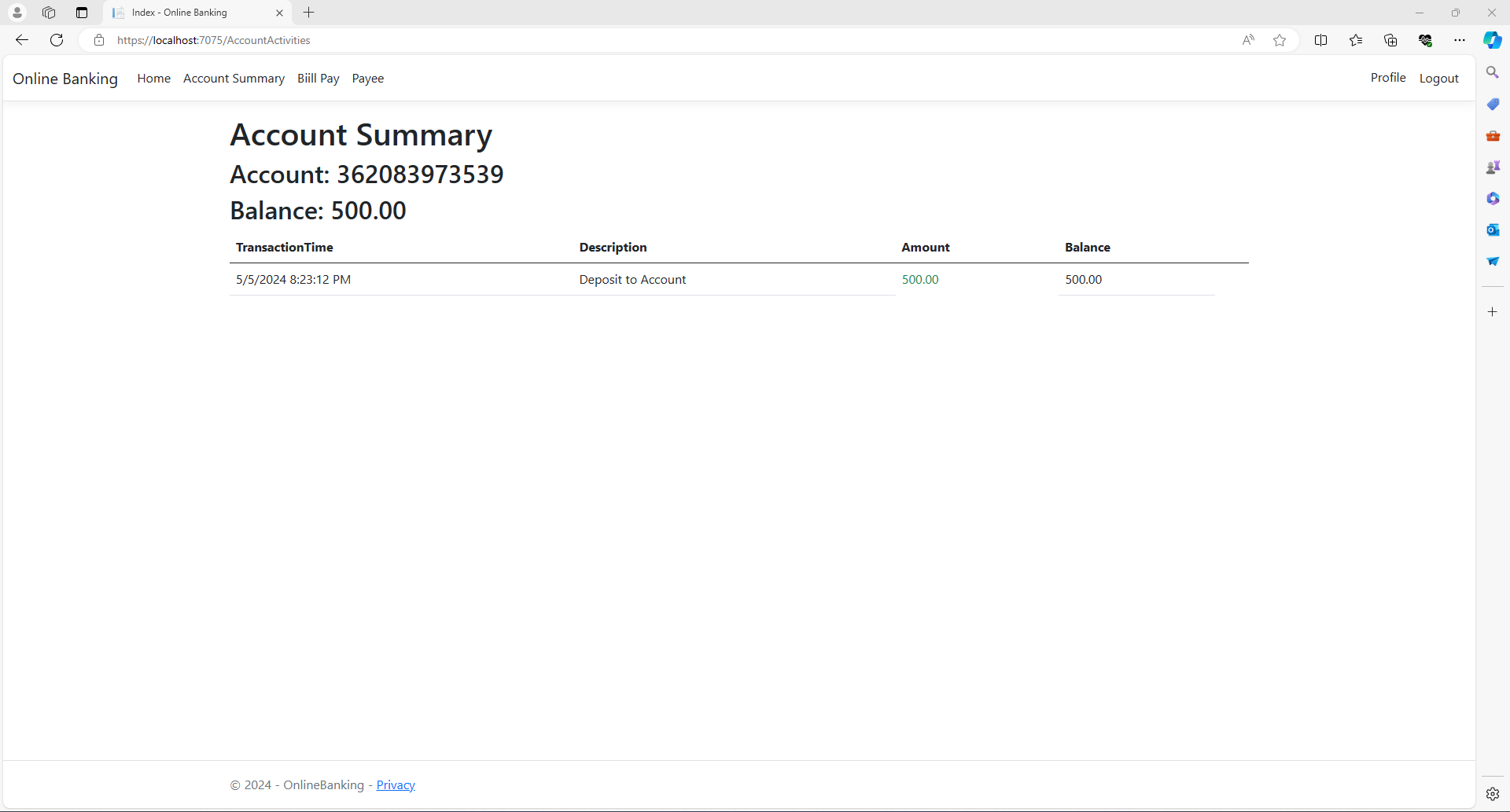
Here the teller can modify the account holder information and withdraw or deposit the money. So, here for the bank customer1, we are depositing 500$ as an example.



Now, I logged in as Bank Customer1:

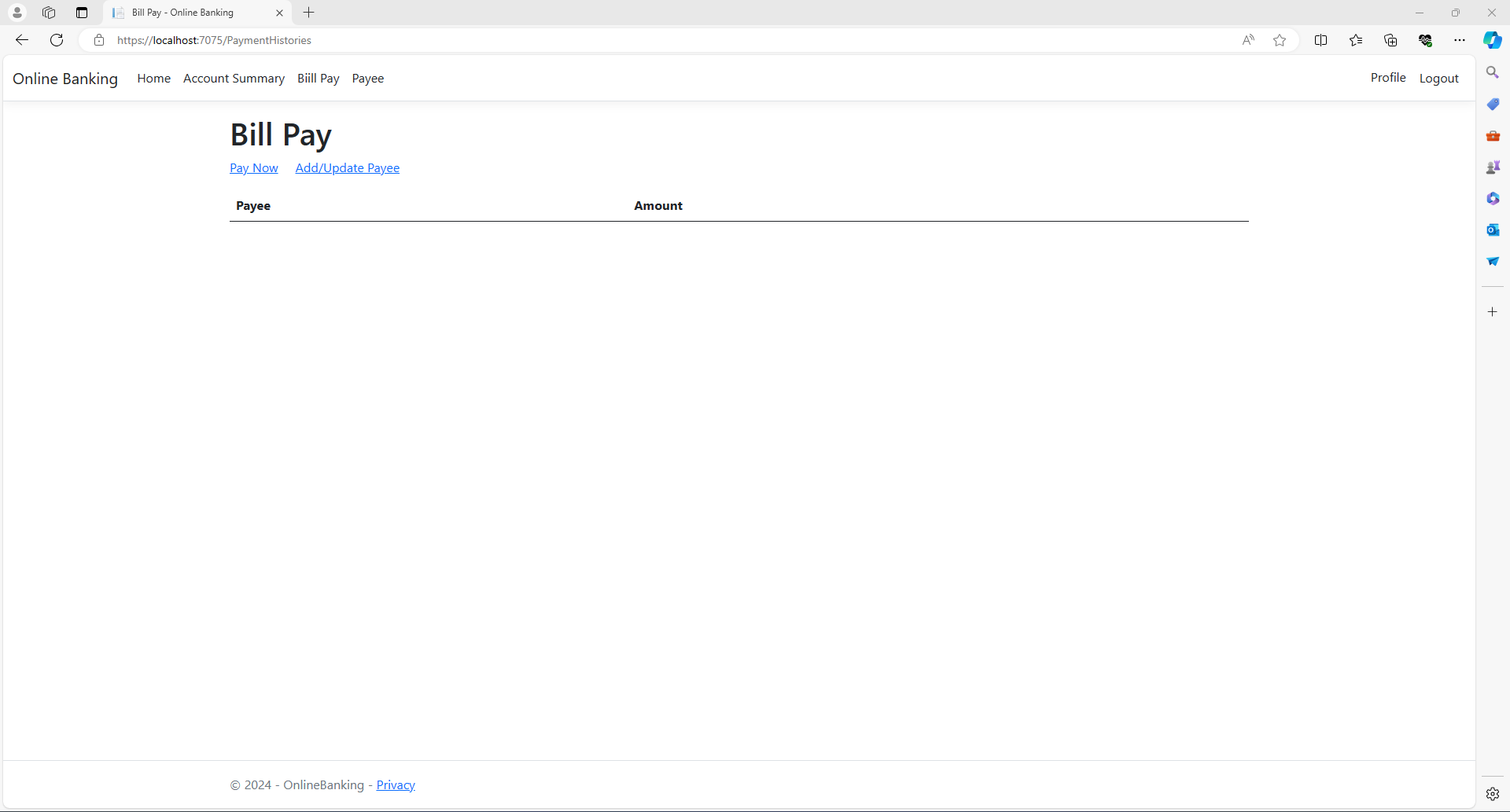


### Account Summary:

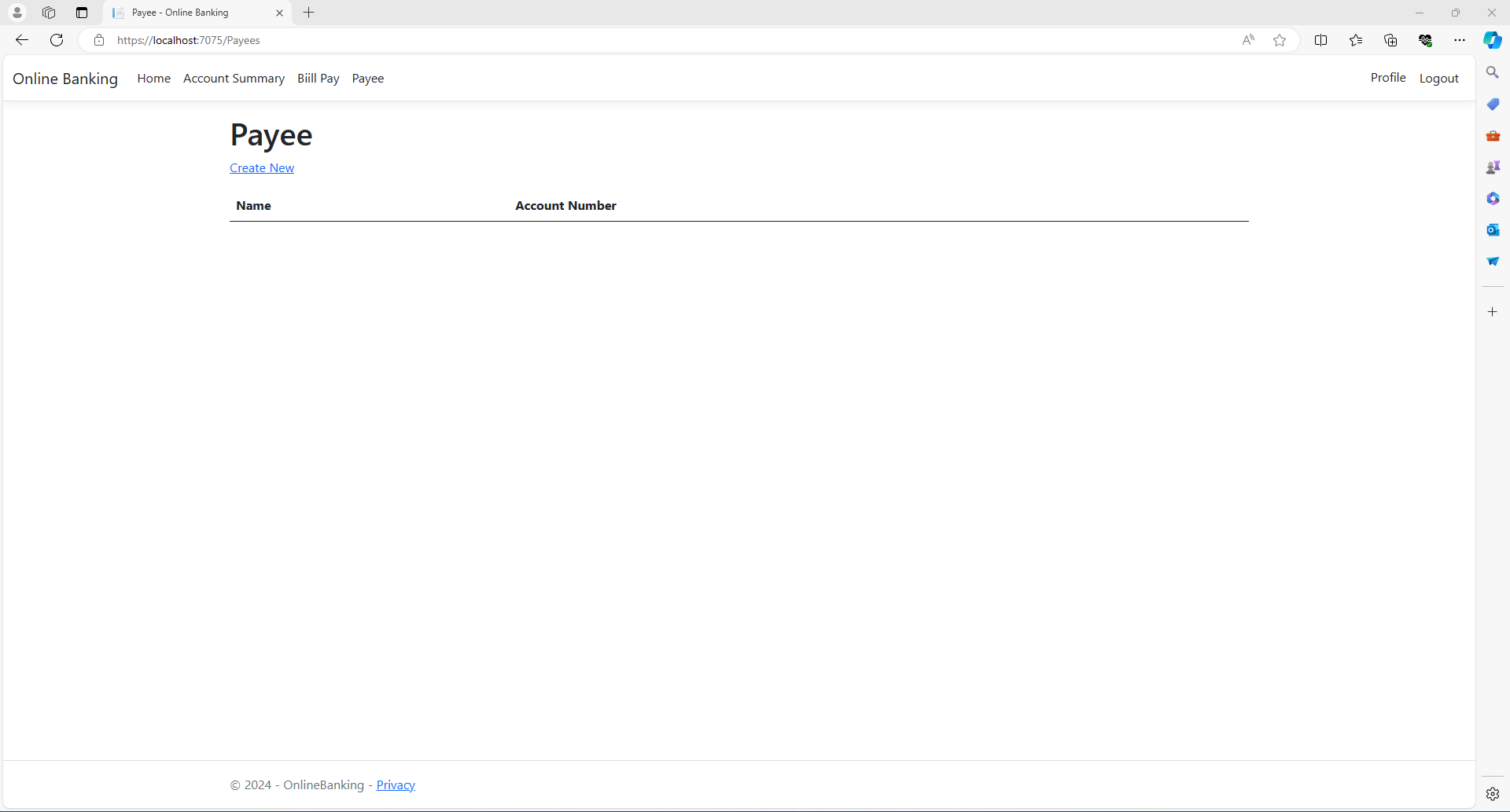


### Bill Pay:

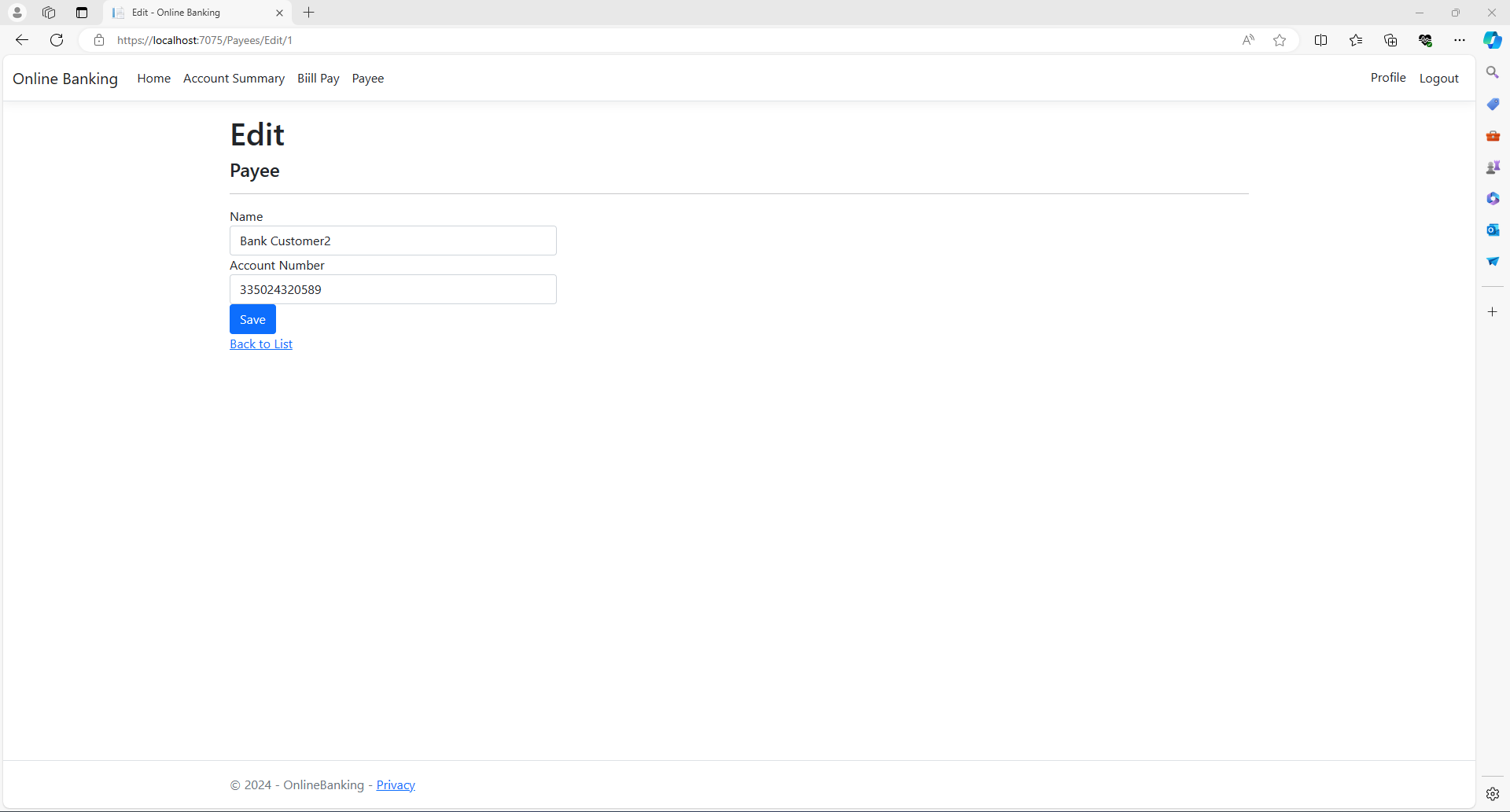
From the Bill Pay screen, the customer can add / update payees or pay the payee.



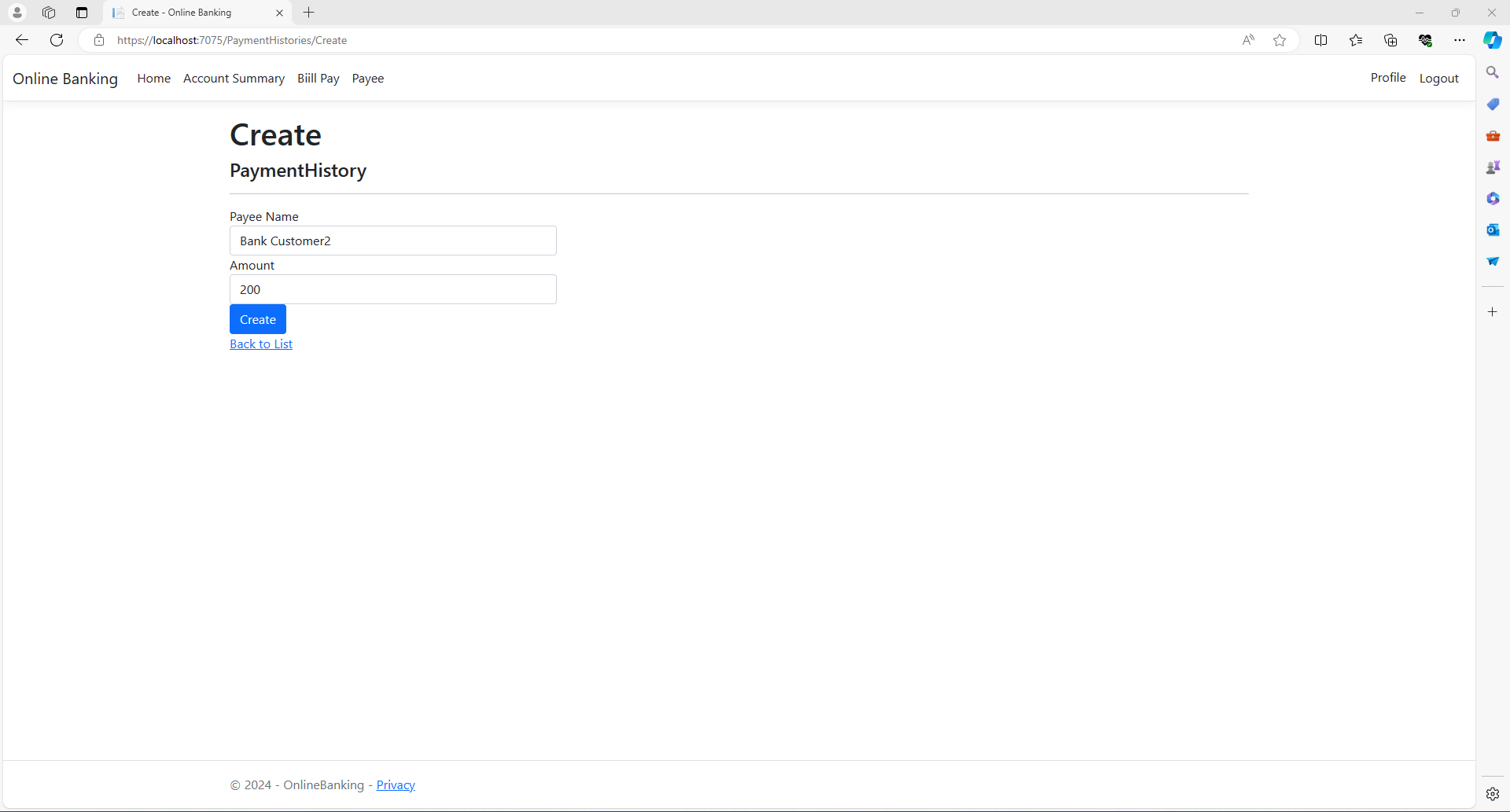
### Payee:

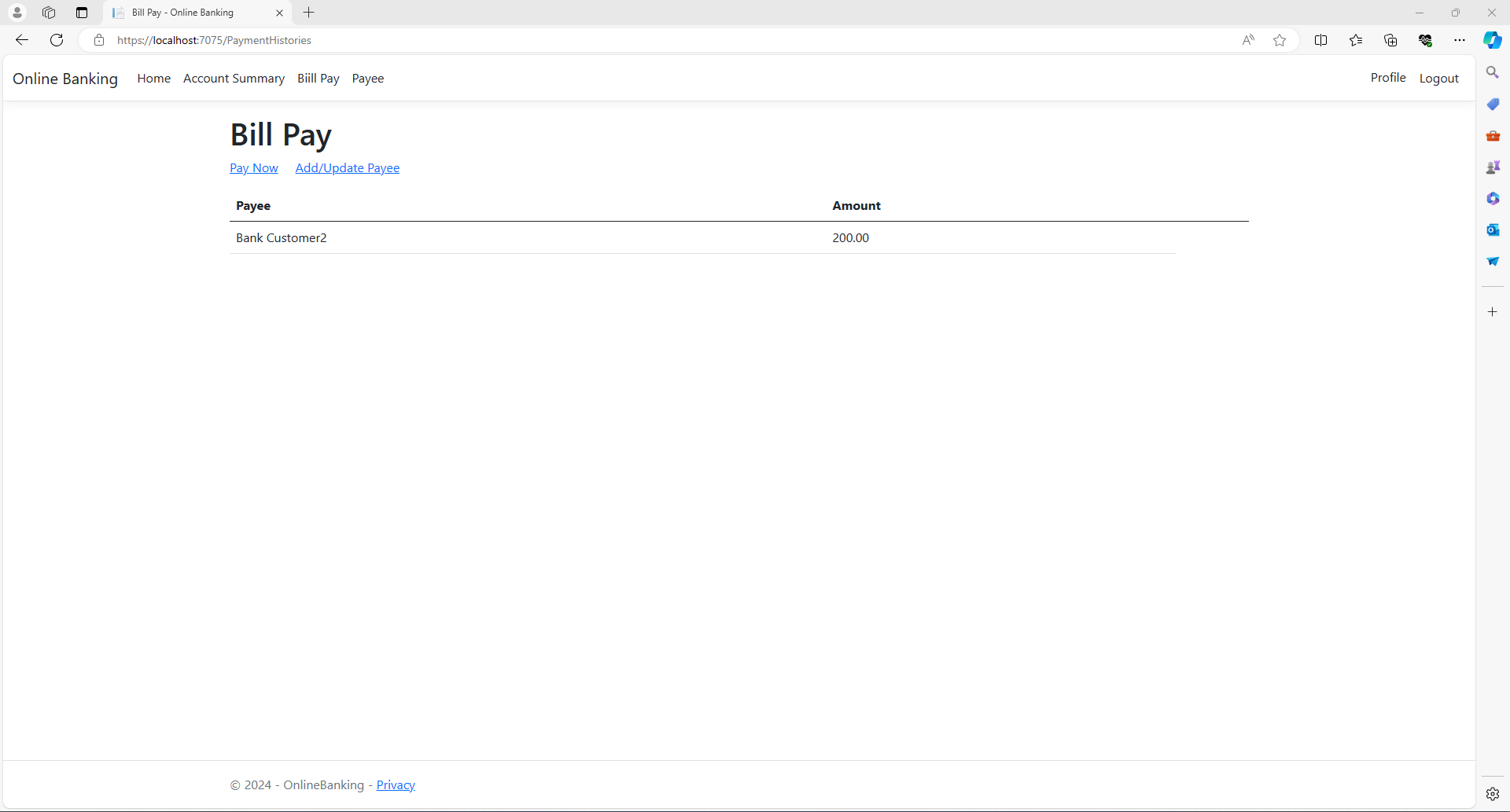


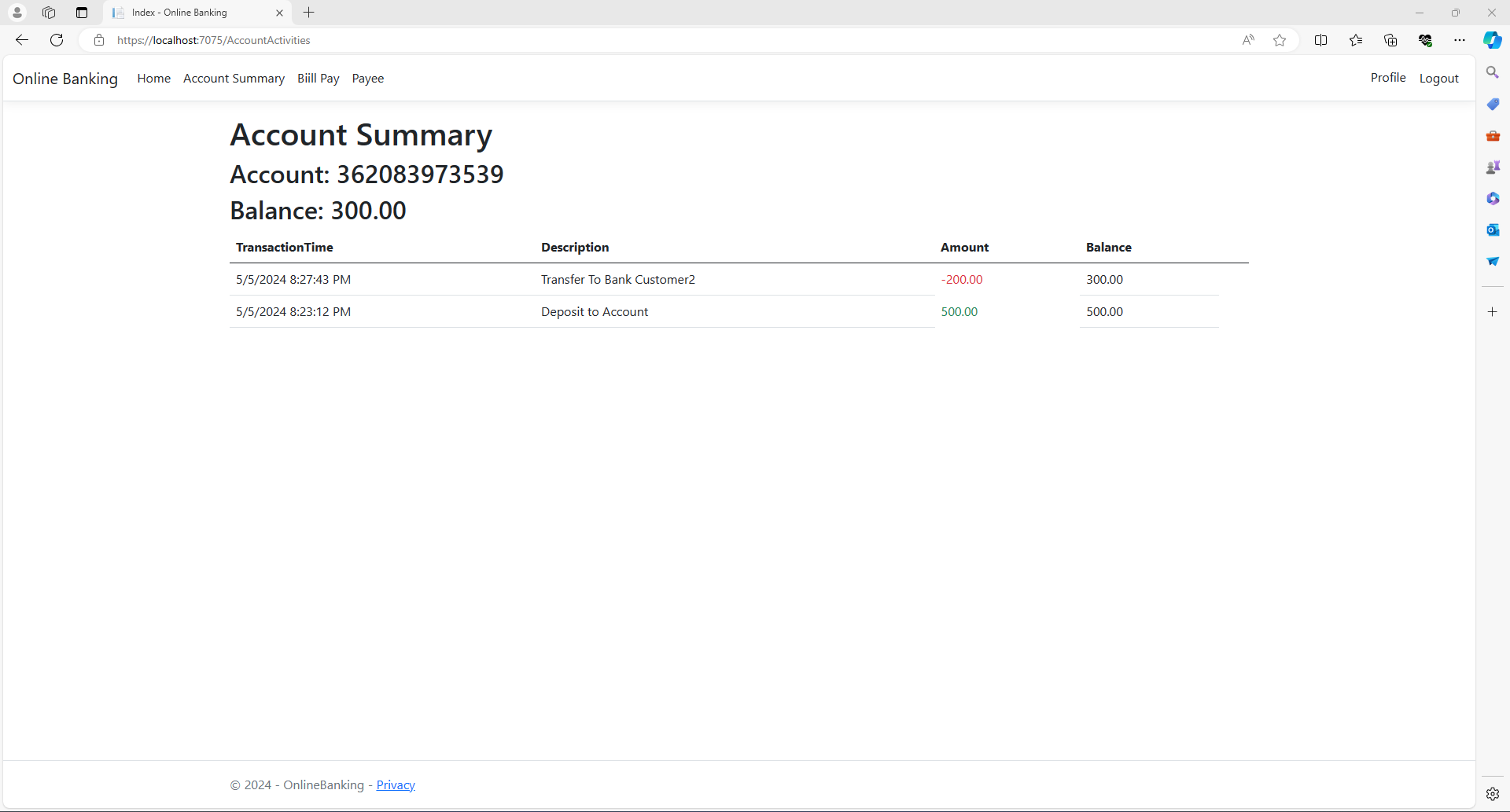
### Add/Update Payee:



### Create Bill Pay:





Here is the update Account Summary after Bill Pay:  


Now, for the Bank Customer2, here is the updated Account Summary with after Bank Customer1 transferred the amount:

