Object-Oriented Concepts and Their Application in UML and Projects

Cleiton Ferreira

Dorset College

Object-Oriented Programming

John Rowley

Due Date

Object-Oriented Concepts and Their Application in UML and Projects

Abstract Classes

In C#, abstract classes are classes that can only serve as a base for another class and cannot be instantiated on their own; they are used as a blueprint. They normally tend to combine both the abstract and concrete (non-abstract) methods. Abstract methods, unlike their concretes counterparts, don't have implementation, but they should be implemented by their derived classes. Abstract classes serve their purpose when you have a varied group of almost similar classes but there specific functionality should be a void responsibility of a derived class. In the furnished code, the `Account` class is an abstract class that sets the design for `current` and `savings` account classes.

Concrete Classes

Concrete classes are those that can be directly subclassed or instantiated. Supplies for all functions that have been inherited from abstract classes or interfaces. Here, the `SavingsAccount` and `CurrentAccount` classes are concrete ones, and they extend the `Account` class which is an abstract one. They operate both `Deposit` and `Withdraw` instruction using their methods implementations for abstract `Deposit` and `Withdraw`.

Inheritance

Inheritance as one of the mechanisms of object-oriented programming appears when a new class (subclass or derived class) is created taking properties and features into consideration from the existing class (superclass or base class). The descending class can add or alter the way the base class performs its function. The `SavingsAccount` and `CurrentAccount` classes extend the functionality of the `Account` class through the inheritance mechanism, making it able to

share some of the previously defined methods and properties while keeping its own particular set of implementation logic.

Polymorphism

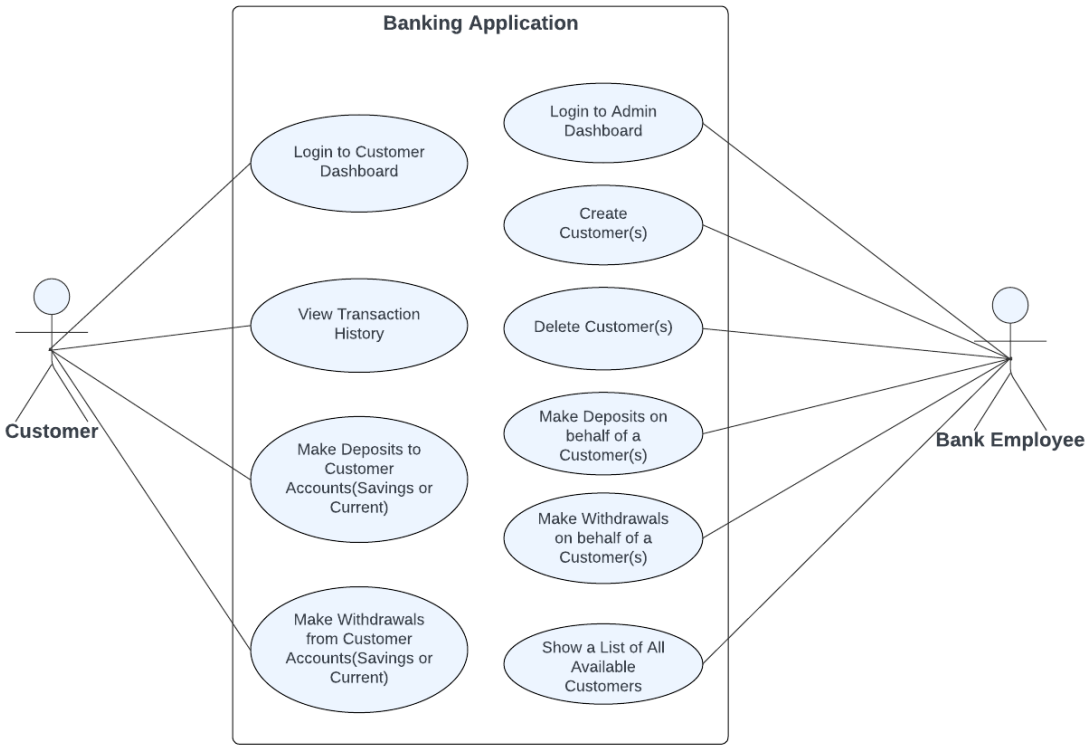
Polymorphism gives the capabilities of placing objects of different types into objects of a common parent class. In this way, we achieve flexibility and extensibility in coding, as methods can be defined to accept superclass objects types, and any objects from a subclass can be passed to these methods. Polymorphism does so by allowing you to have methods that can behave differently based on the type of object they operate on. Morphism is shown by the method `Deposit` and `Withdraw` be invoked on objects of `Account` type, which can point at one of two types, namely, `SavingsAccount` or `CurrentAccount`.

Public, Private, and Protected Access Methods

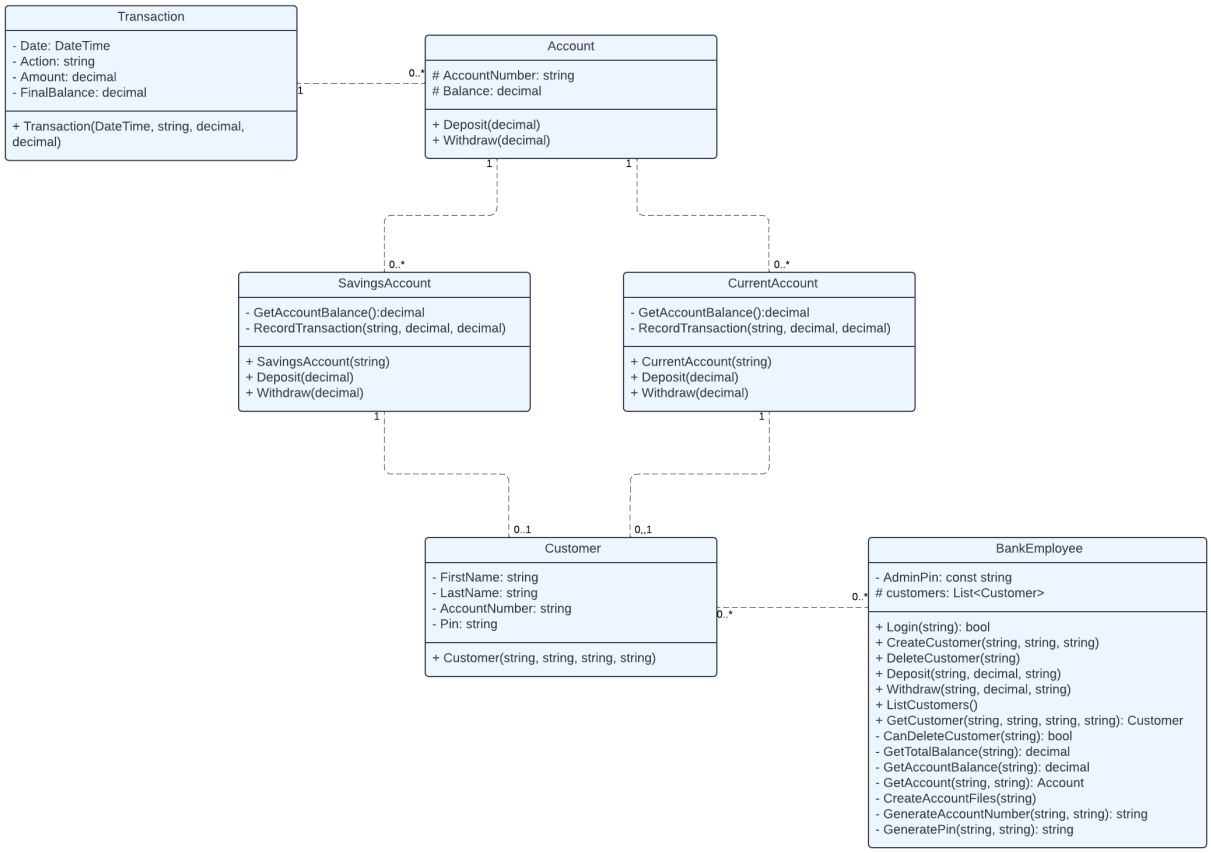
Public methods is an instance of operators, which are located on the outside of the class and can be executed by another classes or piece of code. Private methods are exclusively declared within the class and are only available to other methods within the class. A method of this kind cannot be called from outside of the class in which it is declared. They are only available inside the class and its descendants (these classes are called subclasses) but not from the outside. In the provided code, methods such as `Deposit`, `Withdraw`, `CreateCustomer` and `DeleteCustomer` are publicly available and can be consumed from any outside the classes' classes. Methologies that contain `GetAccountBalance`, `RecordTransaction`, `GenerateAccountNumber` and `GeneratePin` are private methods that are used only within the classes in which they exist. The next, there are special methods like `CanDeleteCustomer`,

`GetTotalBalance`, and `GetAccount` within the `BankEmployee` class and its subclasses that exist as protected methods of the `BankEmployee` class.

The Actor Diagram



The UML Diagram



How to Use the Concepts in the UML/Project

A simple banking system is implemented with functionalities for both bank employees and customers.one is console based and one is an MVC web app sharing the same code. Bank employees can create and delete customer accounts, deposit or withdraw money from customer accounts, and list all existing customers. Customers can view their transaction history, deposit, and withdraw money.

What I have done

Refactored the code to improve readability and maintainability by renaming classes and methods to be more descriptive. Reorganized the code structure to enhance modularity and clarity. Removed redundant comments and streamlined the code. Ensured consistent formatting

and adhered to C# coding conventions for better code consistency. I used the same code used in the console app in the MVC app.

Challenges Faced

Ensuring proper error handling and validation to handle scenarios like invalid PIN, insufficient balance, etc.

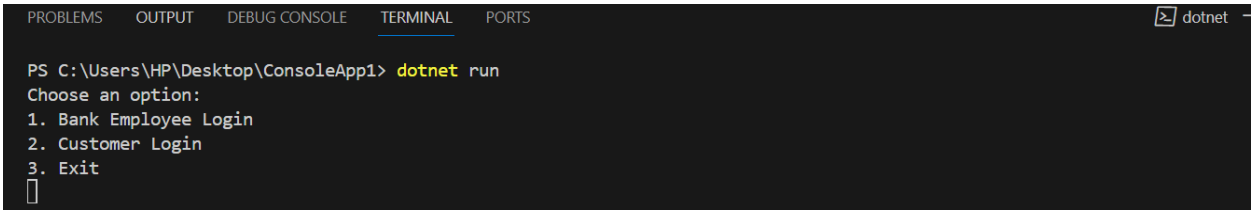
Managing file I/O operations to read from and write to transaction files while ensuring proper file handling and error checking.

Organizing the menu-driven user interfaces for both bank employees and customers to provide a smooth and intuitive experience.

Screenshots

A. Console App

Main Menu



Bank Employee Login

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                dotnet

PS C:\Users\HP\Desktop\ConsoleApp1> dotnet run
Choose an option:
1. Bank Employee Login
2. Customer Login
3. Exit
1
Enter Bank Employee PIN:
A1234
Login successful.
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
```

Bank Employee Menu

```
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
```

Bank Employee: Create Customer

```
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
1
Enter Customer Details:
First Name:
john
Last Name:
grant
Email:
johngrant@gmail.com
Customer created successfully.
```

```
1
Enter Customer Details:
First Name:
joe
Last Name:
smith
Email:
joesmith@gmail.com
Customer created successfully.
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
```

Bank Employee: Delete Customer

```
2
Enter Customer Account Number to Delete:
js-8-10-19
Customer deleted successfully.
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
```

Bank Employee: Deposit Money

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

5. List Customers
6. Back to Main Menu
3
Enter Customer Account Number:
jg-9-10-7
Enter Amount:
4600
Choose Account Type (savings or current):
savings
Deposit of $4,600.00 to savings account successful.
```

```
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
3
Enter Customer Account Number:
jg-9-10-7
Enter Amount:
6300
Choose Account Type (savings or current):
current
Deposit of $6,300.00 to current account successful.
```

Bank Employee: Withdraw Money

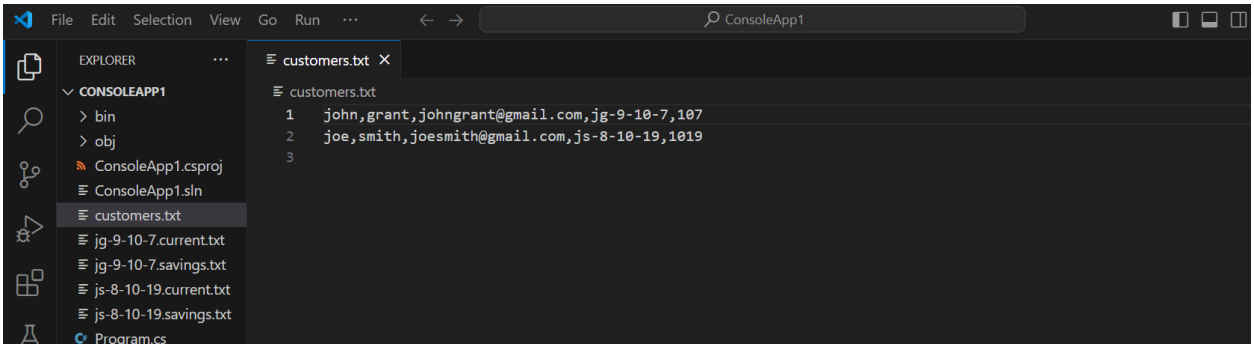
```
4
Enter Customer Account Number:
jg-9-10-7
Enter Amount:
2500
Choose Account Type (savings or current):
savings
Withdrawal of $2,500.00 to savings account successful.
```

```
Bank Employee Menu:
1. Create Customer
2. Delete Customer
3. Deposit Money
4. Withdraw Money
5. List Customers
6. Back to Main Menu
4
Enter Customer Account Number:
jg-9-10-7
Enter Amount:
1800
Choose Account Type (savings or current):
current
Withdrawal of $1,800.00 to current account successful.
```

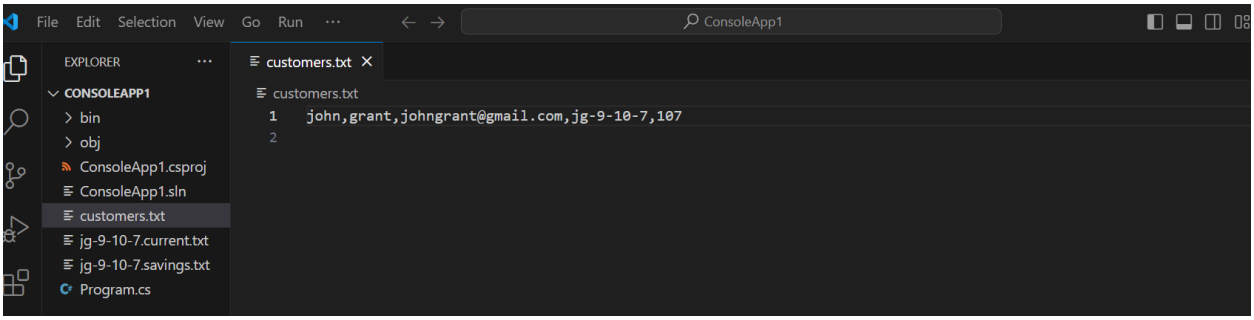
Bank Employee: Back to main

Customers.txt

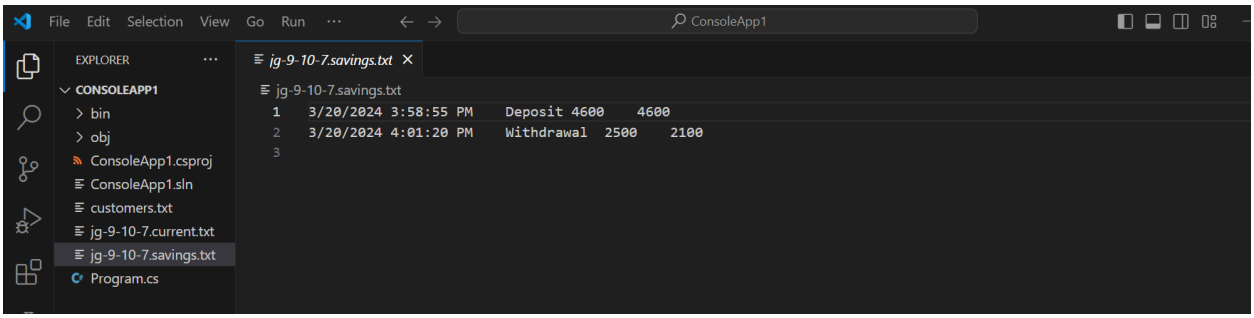
After 2 customers were created



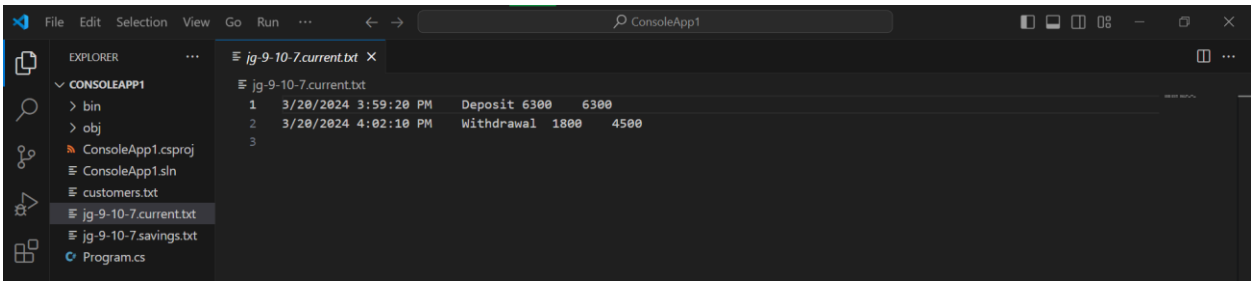
After on customer deletion



jg-9-10-7.savings.txt



jg-9-10-7.current.txt



Bank Employee: List Customers

Bank Worker: Back to Main Menu

```
6
Returning to Main Menu...
Choose an option:
1. Bank Employee Login
2. Customer Login
3. Exit
```

Customer Login

```
Choose an option:
1. Bank Employee Login
2. Customer Login
3. Exit
2
Enter First Name:
john
Enter Last Name:
grant
Enter Account Number:
jg-9-10-7
Enter PIN:
107
Login successful.
```

Customer Menu

```
Customer Menu:
1. View Transaction History
2. Deposit Money
3. Withdraw Money
4. Back to Main Menu
```

Customer: Transaction History

```
Customer Menu:
1. View Transaction History
2. Deposit Money
3. Withdraw Money
4. Back to Main Menu
1
Choose Account Type (savings or current):
savings
Transaction History for savings Account:
3/20/2024 3:58:55 PM    Deposit 4600    4600
3/20/2024 4:01:20 PM    Withdrawal      2500    2100
```

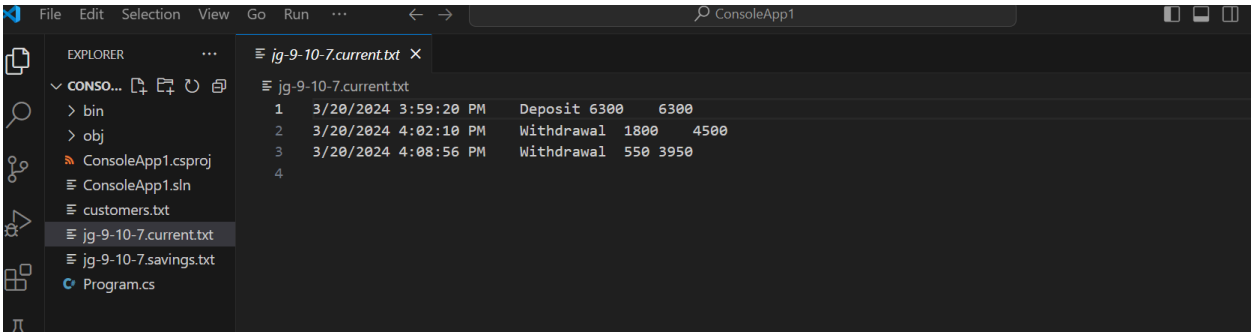
Customer: Deposit Money

```
Customer Menu:
1. View Transaction History
2. Deposit Money
3. Withdraw Money
4. Back to Main Menu
2
Choose Account Type (savings or current):
savings
Enter Amount to Deposit:
300
Deposit of $300.00 to savings account successful.
```

Customer: Withdraw Money

```
Customer Menu:
1. View Transaction History
2. Deposit Money
3. Withdraw Money
4. Back to Main Menu
3
Choose Account Type (savings or current):
current
Enter Amount to Withdraw:
550
Withdrawal of $550.00 to current account successful.
```

jg-9-10-7.current.txt



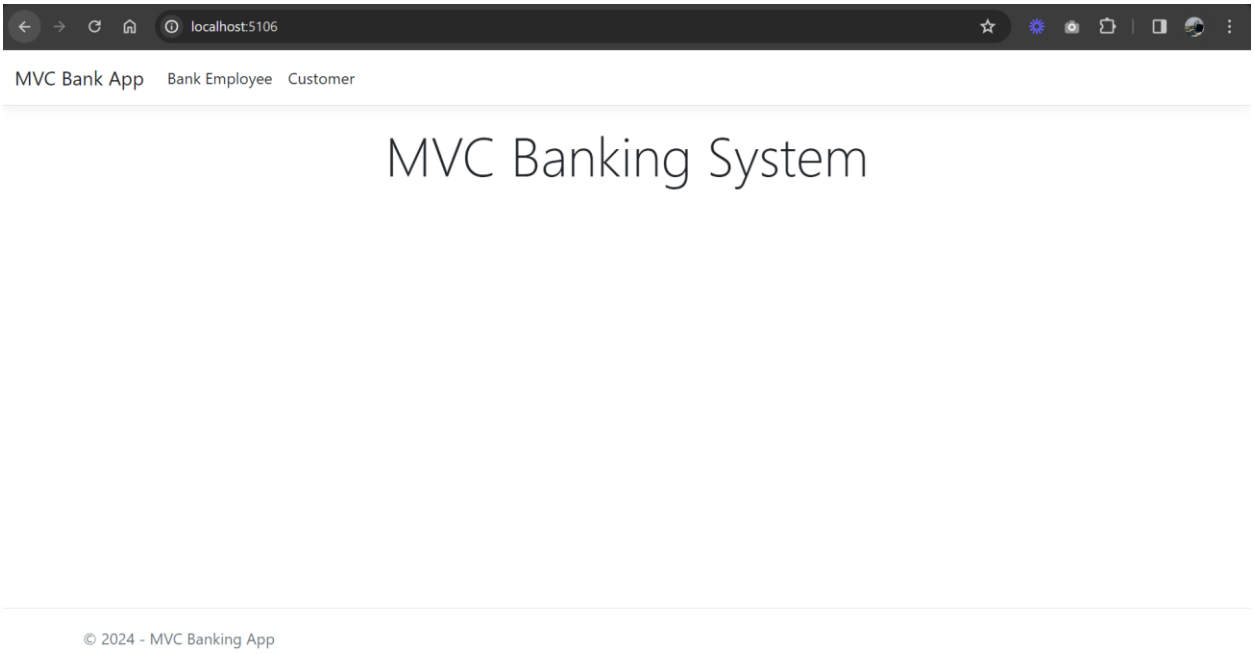
Customer: Back to main Menu

```
Customer Menu:
1. View Transaction History
2. Deposit Money
3. Withdraw Money
4. Back to Main Menu
4
Returning to Main Menu...
Choose an option:
1. Bank Employee Login
2. Customer Login
3. Exit
```

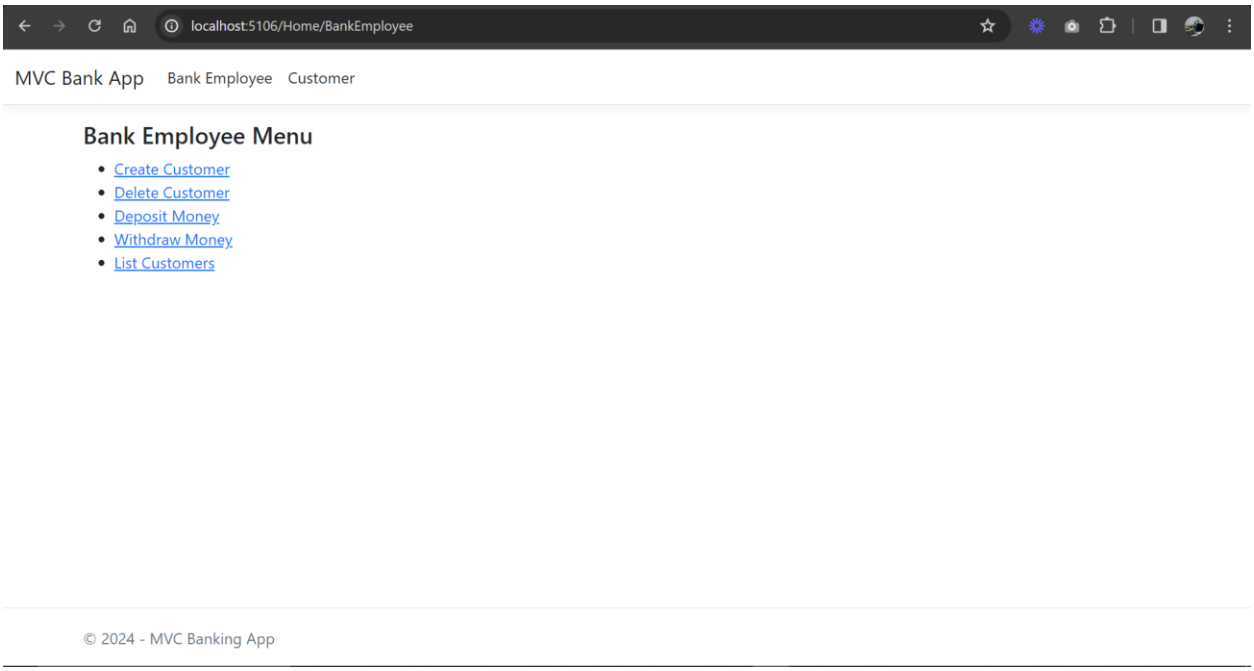
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text Go Live

B. MVC App

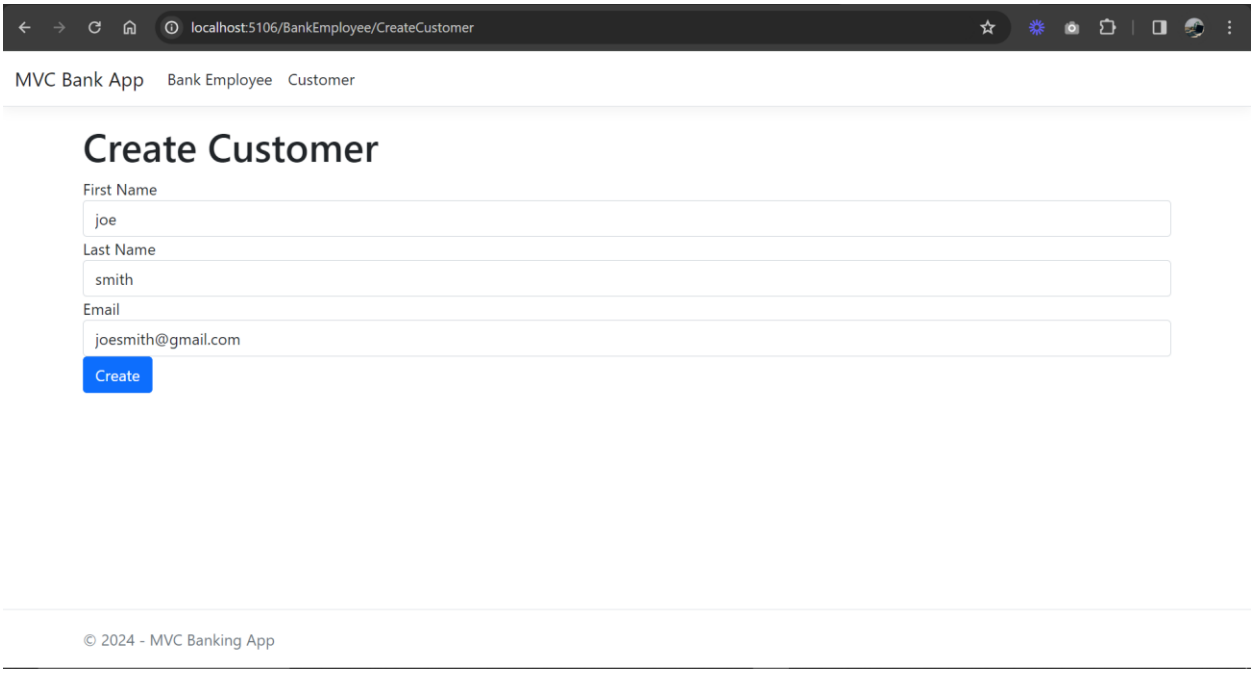
Main menu



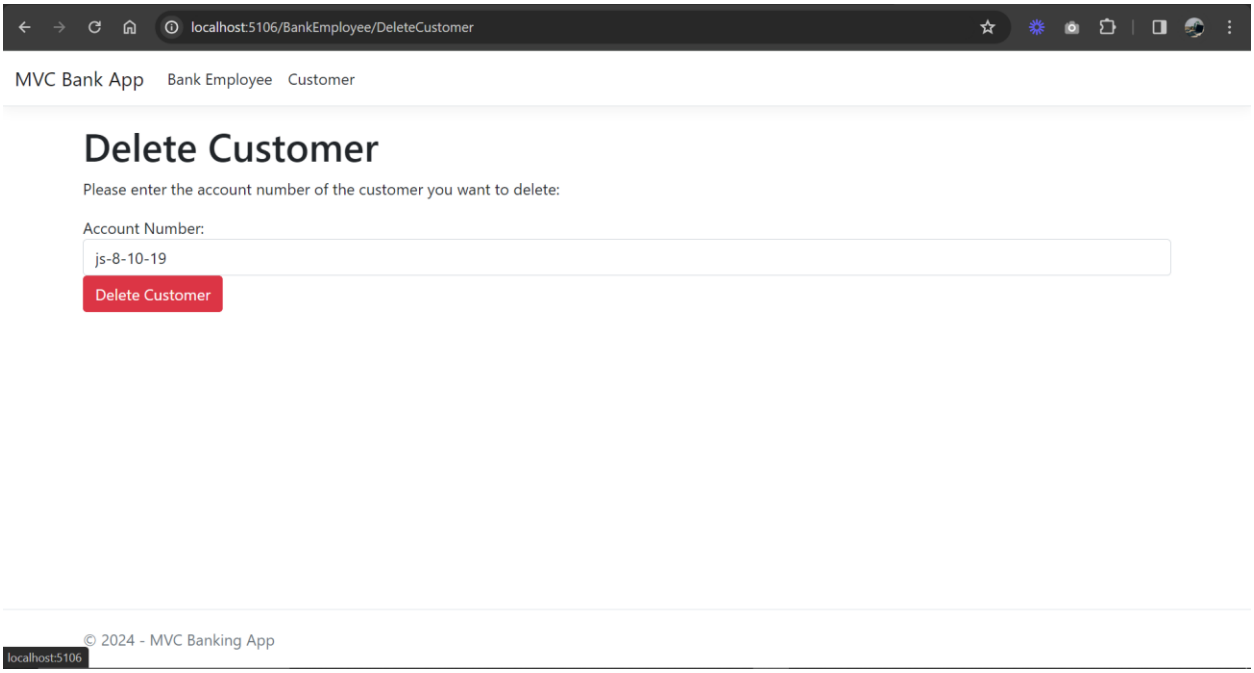
Bank Employee menu



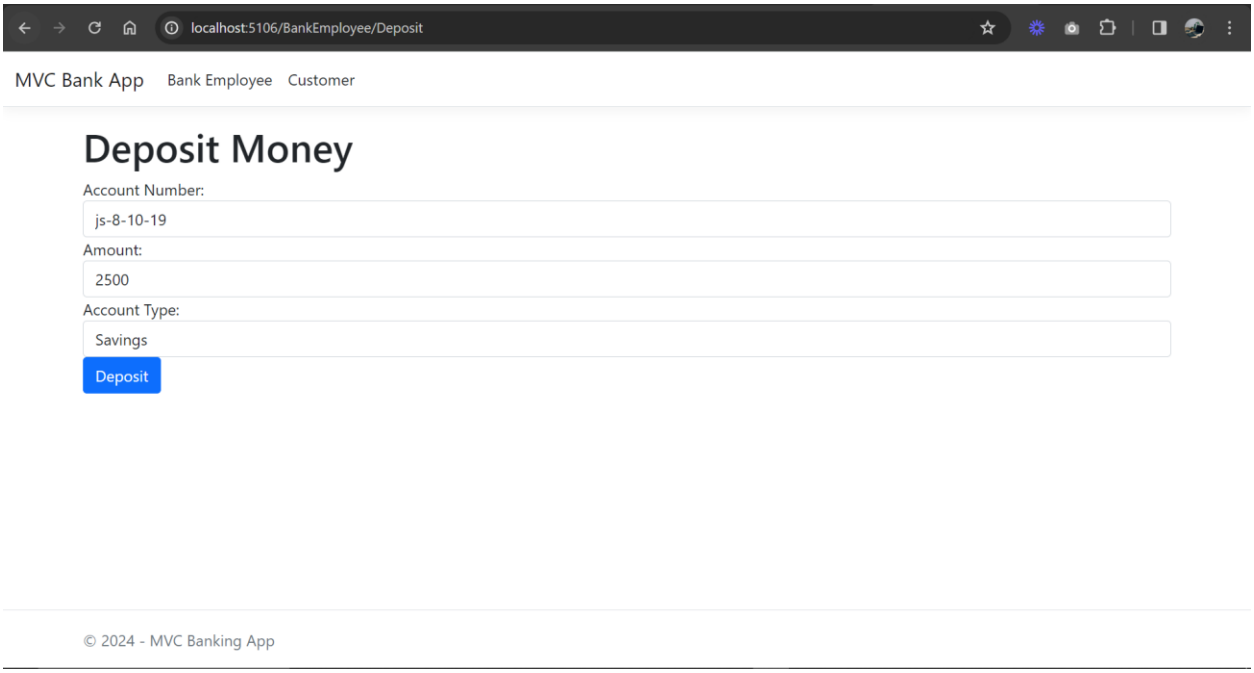
Create Customer



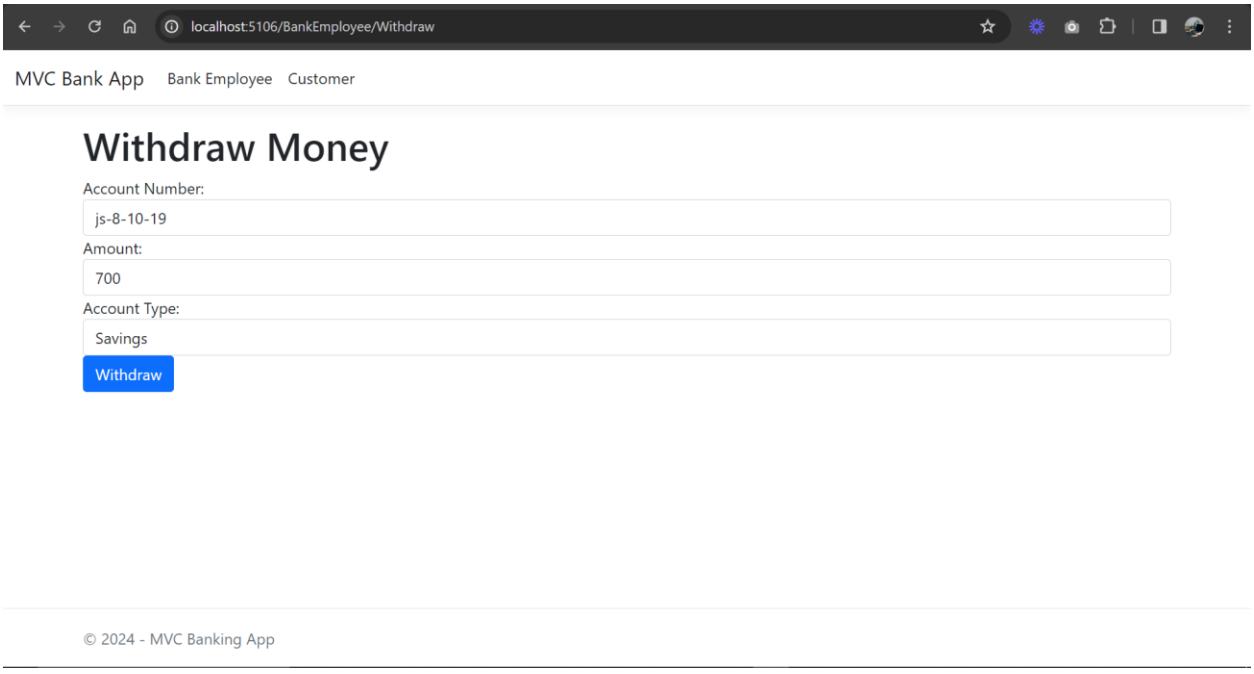
Delete Customer



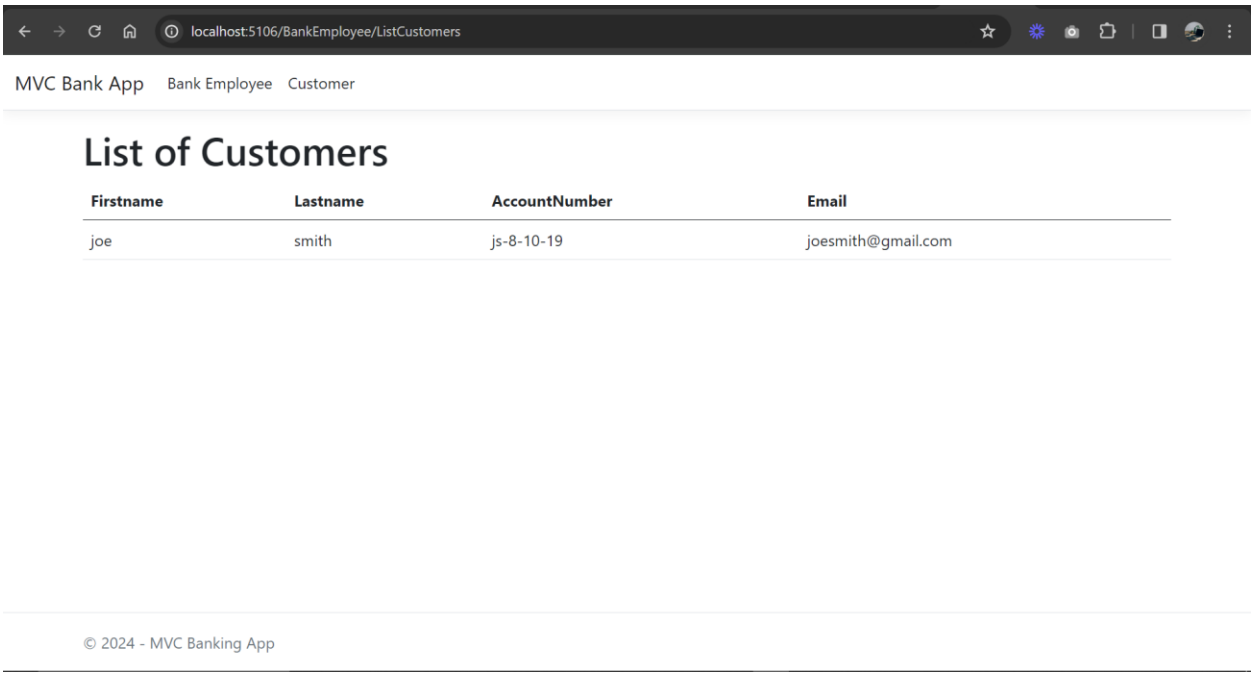
Deposit



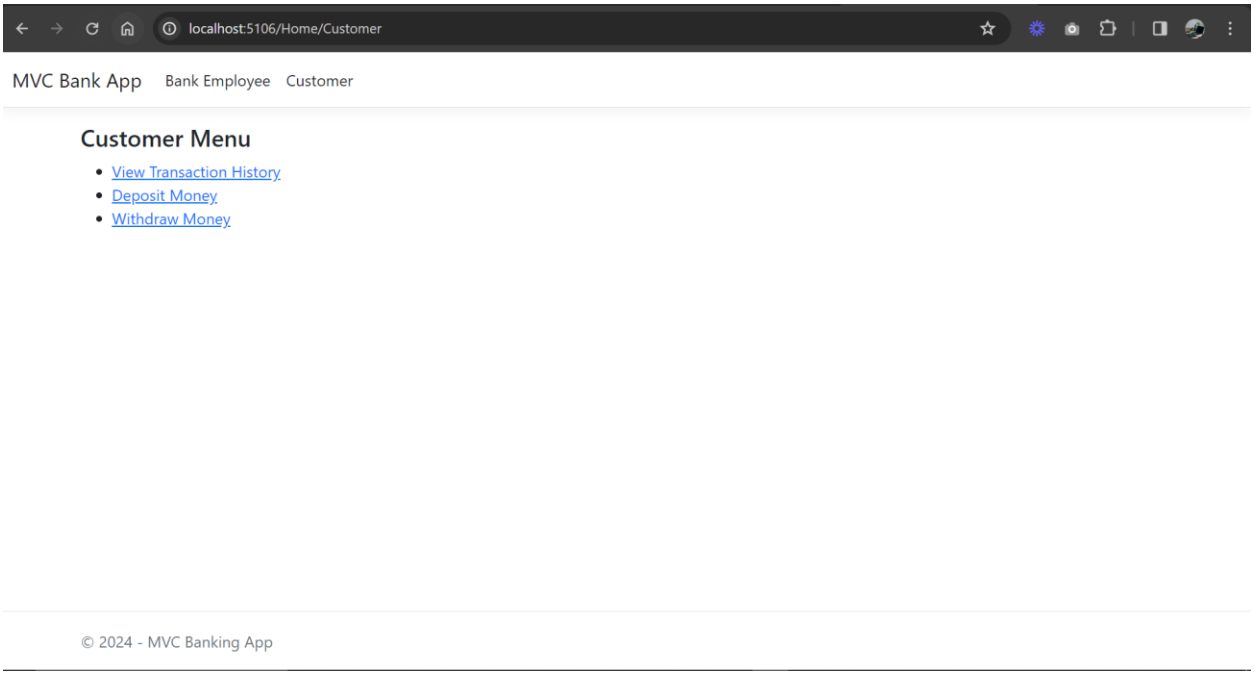
Withdraw



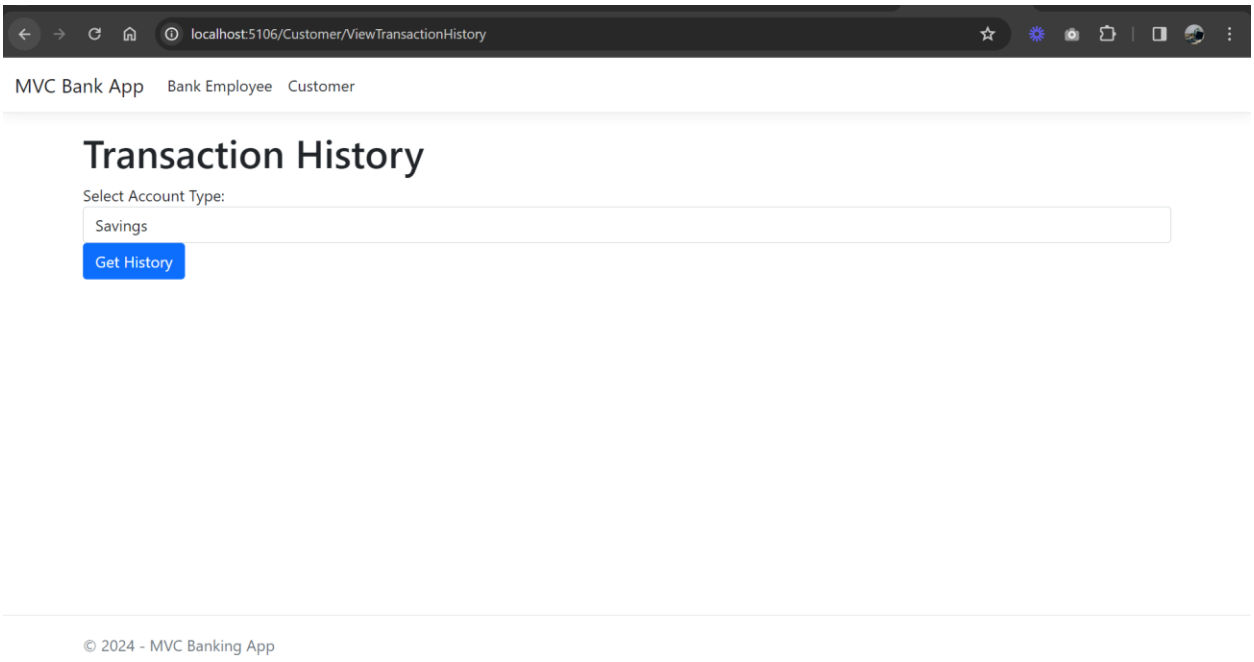
List Customers

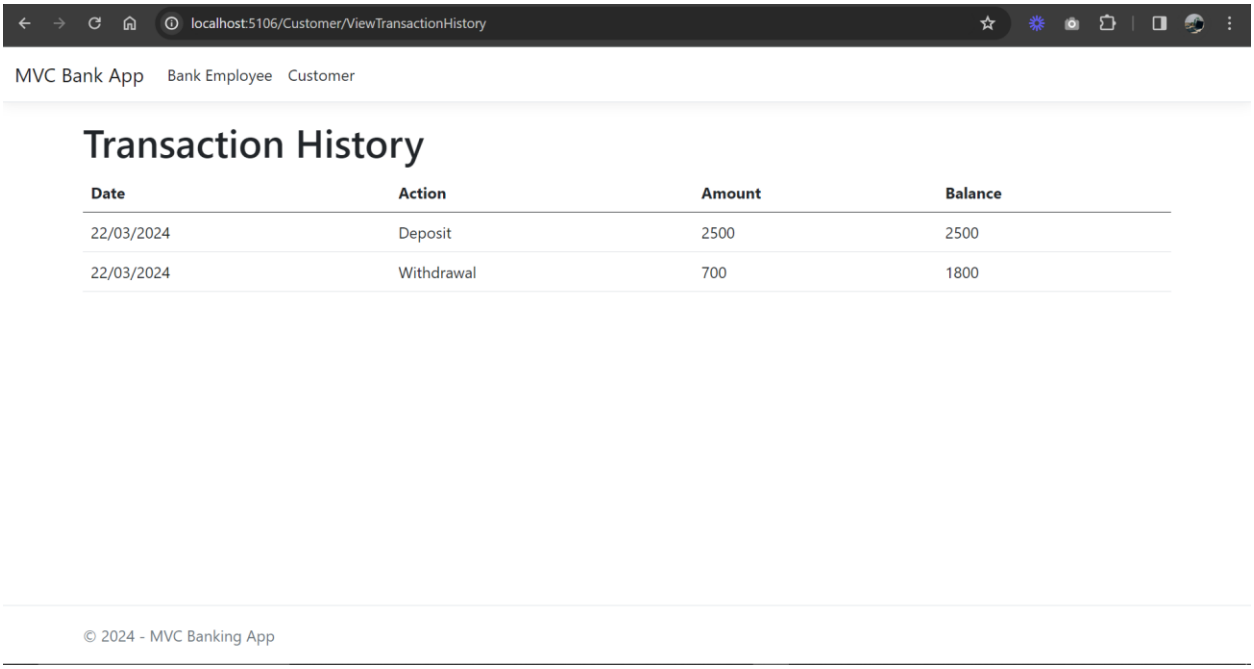


Customer Menu

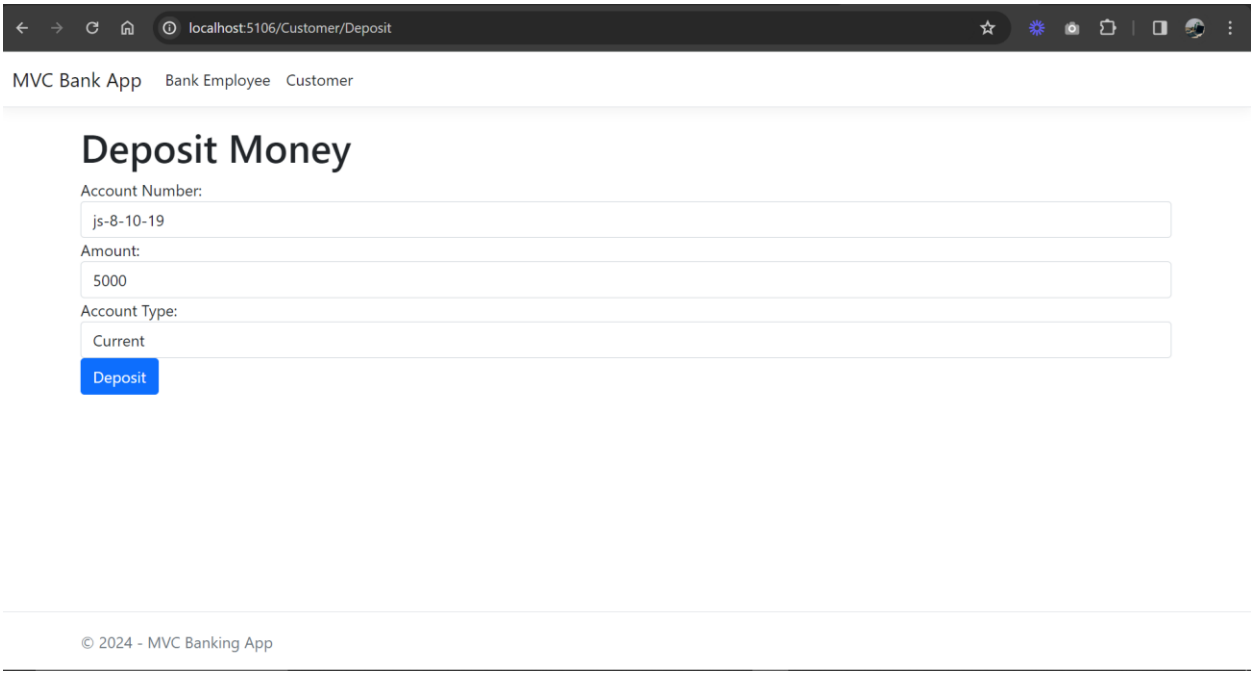


View Transaction History

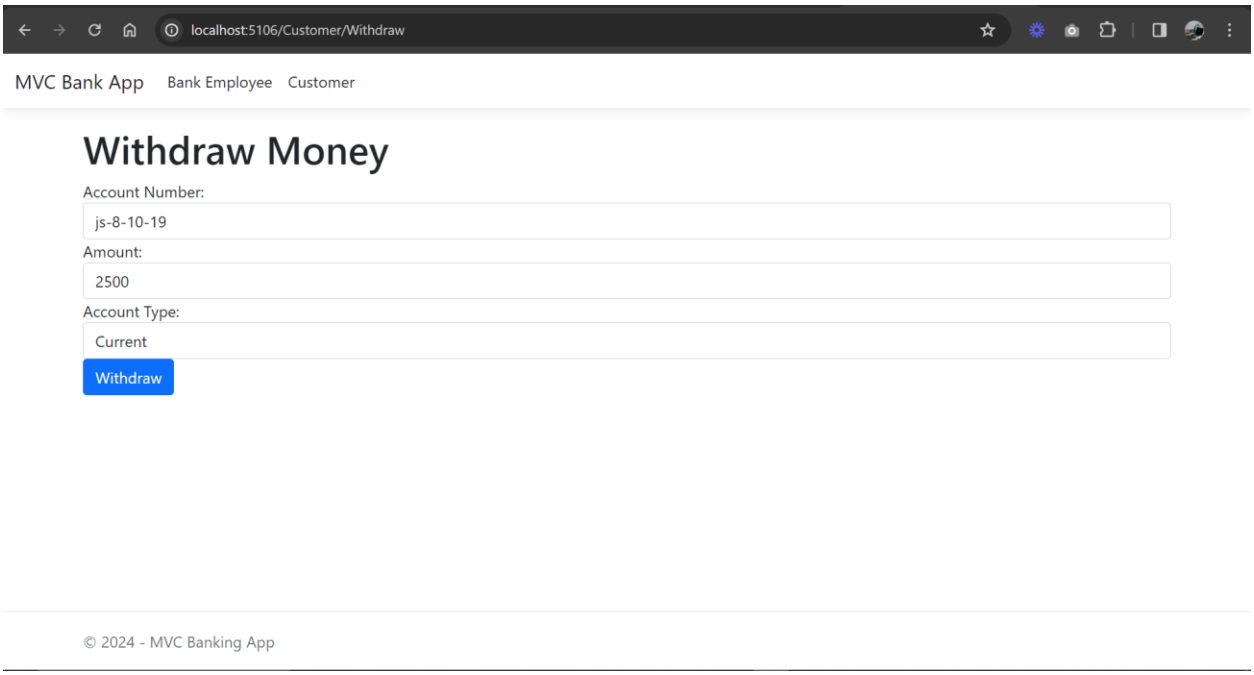

localhost:5106/Customer/ViewTransactionHistory
MVC Bank App
Bank Employee
Customer
Transaction History
Select Account Type:
Savings
Get History
© 2024 - MVC Banking App


localhost:5106/Customer/ViewTransactionHistory
MVC Bank App
Bank Employee
Customer
Transaction History
Date
Action
Amount
Balance
22/03/2024
Deposit
2500
2500
22/03/2024
Withdrawal
700
1800
© 2024 - MVC Banking App

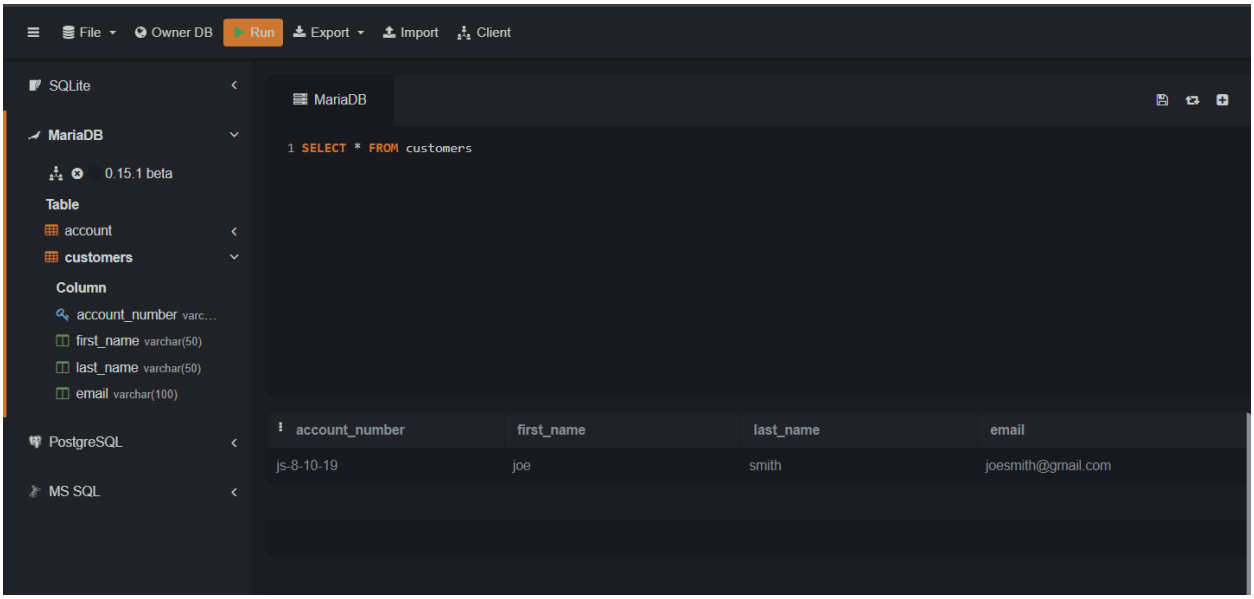
Deposit

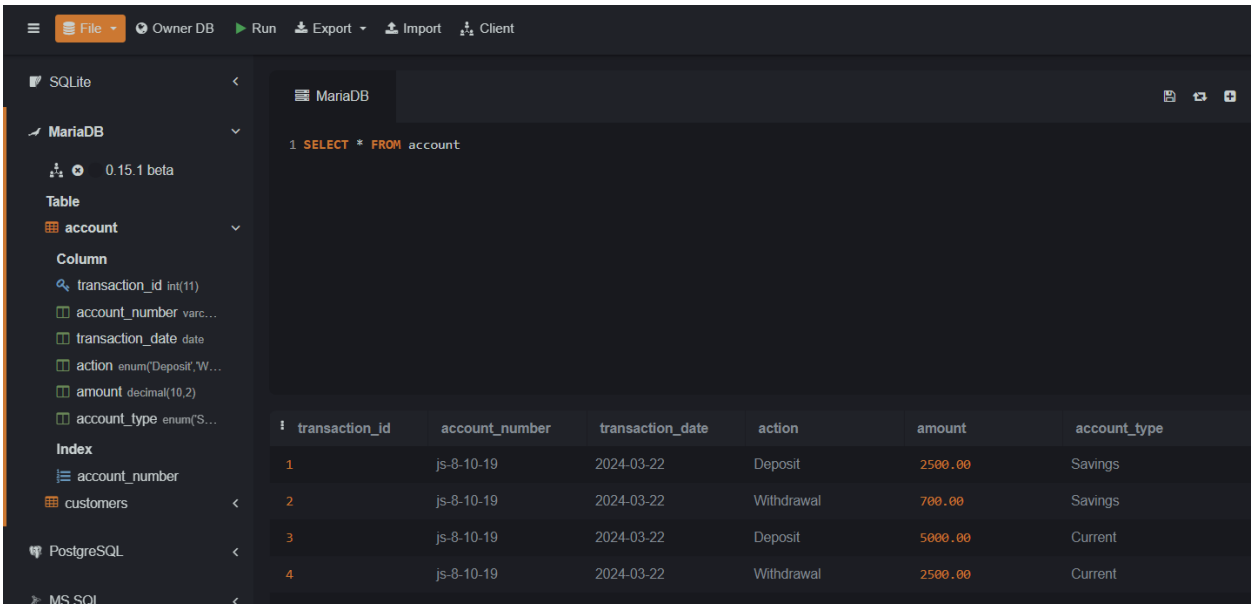


Withdraw



Database tables


File
Owner DB
Run
Export
Import
Client
SQLite
MariaDB
0.15.1 beta
Table
account
customers
Column
account_number varc...
first_name varchar(50)
last_name varchar(50)
email varchar(100)
PostgreSQL
MS SQL
MariaDB
1 SELECT * FROM customers
account_number
first_name
last_name
email
js-8-10-19
joe
smith
joesmith@gmail.com


File
Owner DB
Run
Export
Import
Client
SQLite
MariaDB
0.15.1 beta
Table
account
Column
transaction_id int(11)
account_number varc...
transaction_date date
action enum('Deposit','W...
amount decimal(10,2)
account_type enum('S...
Index
account_number
customers
PostgreSQL
MariaDB
1 SELECT * FROM account
transaction_id
account_number
transaction_date
action
amount
account_type
1
js-8-10-19
2024-03-22
Deposit
2500.00
Savings
2
js-8-10-19
2024-03-22
Withdrawal
700.00
Savings
3
js-8-10-19
2024-03-22
Deposit
5000.00
Current
4
js-8-10-19
2024-03-22
Withdrawal
2500.00
Current