**CAPSTONE PROJECT: BANKING SYSTEMS**

1. **SYSTEM REQUIREMENT SPECIFICATION:**
2. **Software Requirements:**

* Operating System: windows 10
* Coding language: Java
* Data Base: MySQL
* Front end: browser (recent version)

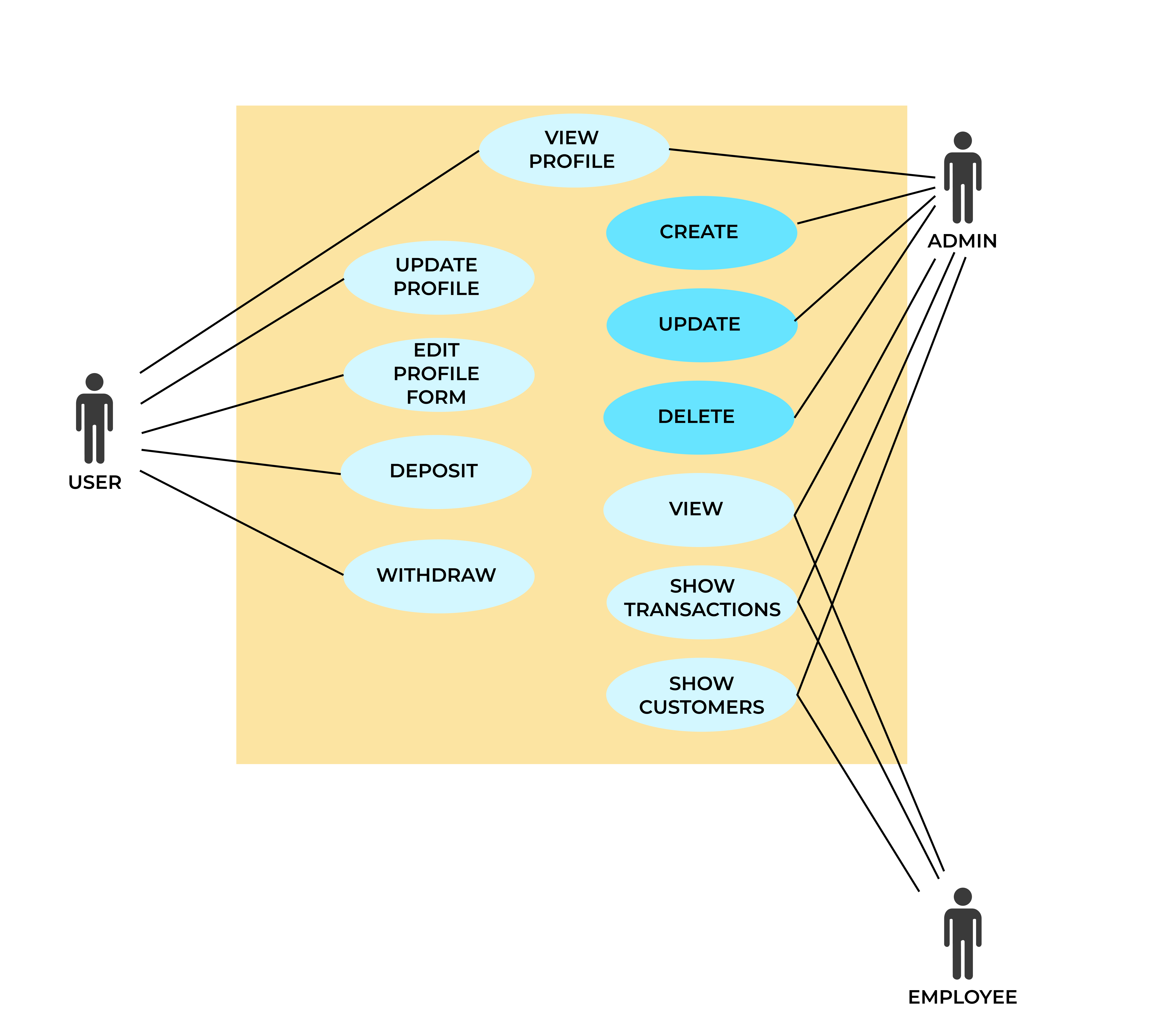
1. **Hardware Requirements:**

* Personal computer with keyboard and mouse maintained with uninterrupted power supply.
* Processor: Intel coreTM i5
* Installed Memory (RAM): 2.00 GB

1. **PROJECT SCOPE:**

* Front-end for the bank employees who can view the various transactional details and perform the required operations.
* Front-end for the customers to be able to view their specific transactions and details. They should also be able to interact with the system and perform required transactions.
* Front-end for the Admin to view all the details of all customers and employees.

1. **USE CASE DIAGRAM:**



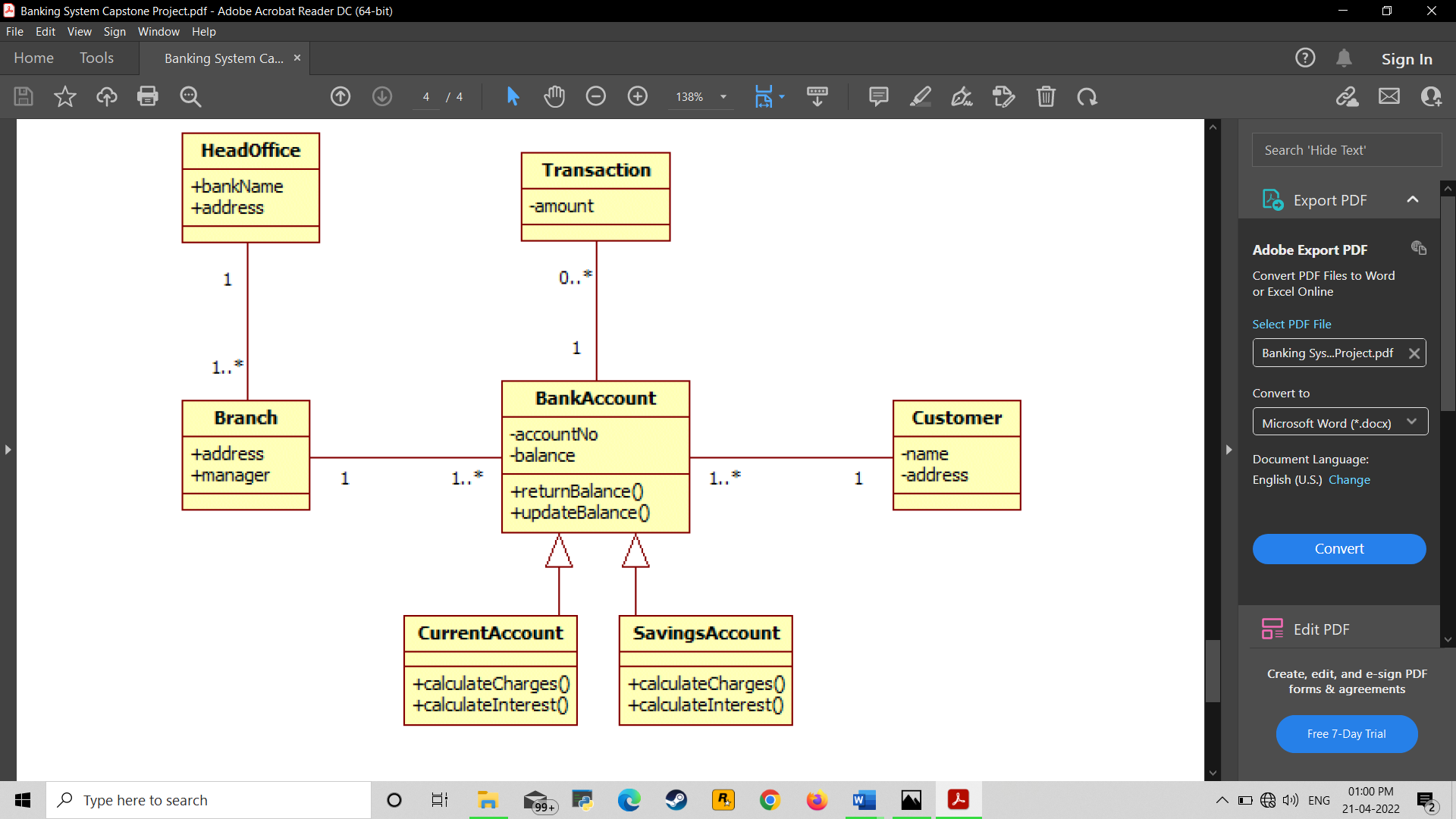
Use Case diagram is used for representing the problem statement, the actors in it, and their functionality in a behavioral manner. In the above figure and the system, there are 3 different actors:

* 1. User
  2. Admin
  3. Employee

1. **FLOW OF EVENTS:**

* Basic flow
  1. **Withdraw:**
     1. User enters username and password.
     2. Bank Database validates the user.
     3. On success user can withdraw money.
  2. **Deposit:**
     1. User enters username and password.
     2. Bank Database validates the user.
     3. On success user can deposit money.
  3. **View Profile**
     1. User/Admin/Employee enters username and password.
     2. Bank Database validates the user.
     3. On success User/Admin/Employee can view profile. (They can also change their own profile details such as password, etc.)
  4. **View Transactions:**
     1. Admin/Employee logs in
     2. Bank Database validates the Admin/Employee.
     3. After successful validation, Admin/Employee can view all transaction details.
  5. **Show Customers:**
     1. Admin/Employee logs in
     2. Bank Database validates the Admin/Employee.
     3. After successful validation, Admin/Employee can view all customers and their details.
* Alternate Flow:
  1. If in the basic flow, the details entered by the user/admin/employee are invalid, then the user/admin/employee is informed that their login attempt has failed. The user may re-enter correct details or create a new account.
* Pre-conditions:
  1. The user must have a valid account in the bank.
* Post-conditions:
  1. The database is modified after transaction.

1. **CLASS DIAGRAM:**



1. **SEQUENCE DIAGRAM:**
   1. LOGIN & LOGOUT COLLABORATION DIAGRAM:



* 1. ONLINE TRANSACTION SEQUENCE 

1. **IDENTIFICATION OF RESPONSIBILITIES OF CLASSES:**
   1. Controllers:

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AdminController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| - View admin profile  - Create update delete employees and customers | Service layer classes. | Admin, employee, customer models and AdminRepository, EmployeeRepository, CustomerRepository. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankAccountController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Contains the creation, updation, and deletion code for employees and customers done by the administrator | Service layer classes. | BankAccount, CurretnAccount, SavingsAccount models and BankAccountRepository. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Create, update, delete, and view details of the bank. | Service layer classes. | Bank model and BankService service and AddressHelper mapper class. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BranchController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| - C.R.U.D. for Branch  - View transactions  - View customers | Service layer classes. | AddressHelper mapper class, Branch and Customer models, and BranchService service layer. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CustomerController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides functionalities for users to view, update, and edit their profiles. | Service layer classes. | BankAccount, Customer, and Transaction models.  BankAccountRepository and CustomerRepository. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: EmployeeController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| - CRUD for customer  - View employee profile details. | Service layer classes. | Bank, BankAccount, Customer, Employee, Transaction models.  BankAccountRepository, CustomerRepository, and EmployeeRepository. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: HomeController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| -Shows welcome screen. | Service layer classes. |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionController** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Contains the deposit and withdraw functionalities for the user and the show all transactions for administrator and employees. | Service layer classes. | Transaction model and Transaction services. |

* 1. DAO:

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AdminDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on AdminRepository |  | Admin, Customer and Transaction models. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankAccountDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on BankAccountRepository |  | BankAccount model |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on BankRepository |  | Bank model |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BranchDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on BranchRepository |  | Branch, Customer, and Transaction models |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CustomerDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on CustomerRepository |  | Customer and Transaction models. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: EmployeeDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on EmployeeRepository |  | Bank, Employee, and Transaction models. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionDAO** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Provides abstract interface for implementation on TransactionRepository |  | Transaction model. |

* 1. Mappers:

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AddressHelper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Address to a result object. | Database. | Address model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AdminMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Admin to a result object. | Database. | Admin model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Bank to a result object. | Database. | Bank model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BranchMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Branch to a result object. | Database. | Branch, Customer, and Transaction model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CurrentAccountMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of CurrentAccount to a result object. | Database. | BankAccount and CurrentAccount models. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CustomerMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Customer to a result object. | Database. | Customer model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: EmployeeMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Employee to a result object. | Database. | Employee model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: ReversalRequestsMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of ReversalRequests to a result object. | Database. | ReversalRequests model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: SavingsAccountMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of SavingsAccount to a result object. | Database. | BankAccount and SavingsAccount models. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionMapper** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Mapping of each row of Transaction to a result object. | Database. | Transaction and TransactionType model. |

* 1. Models:

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AccountType** | | |
| Enumerator for account type. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Address** | | |
| For setting the address and displaying it. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Admin** | | |
| For setting and displaying Admin details. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Bank** | | |
| For setting and displaying bank details such as code, name, address, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankAccount** | | |
| For setting and displaying bank account details of a customer like account number, customer id, branch, available balance, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Branch** | | |
| For setting and displaying branch details like branch code, bank code, manager, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CurrentAccount** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Calculates charge and interest and returns account balance of current account. |  | BankAccount model. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Customer** | | |
| For setting and displaying customer details like name, address, customer id, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Employee** | | |
| For setting and displaying employee details like id, branch code, name, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: ReversalRequests** | | |
| For setting and displaying date, time, request id of reversal requests. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: SavingsAccount** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| Calculates charge and interest and returns account balance of savings account. |  | Extends BankAccount model |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: Transaction** | | |
| For setting and displaying transaction details like timestamp, amount, transaction number, etc. |  |  |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionType** | | |
| Enumerator for account type. |  |  |

* 1. Repository:

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: AdminRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For getting Admin by ID, showing Transactions and showing customers. |  | Admin, Customer and Transaction Mapper.  Admin, Customer and Transaction models.  Admin DAO. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankAccountRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating and showing accounts, getting all accounts by Branch code, Account number and ID and deleting a bank account. |  | SavingAccount Mapper.  BankAccount models.  BankAccount DAO. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating Bank and viewing details by IFSC. |  | Bank Mapper.  Bank models.  Bank DAO. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BranchRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating branch, viewing details by IFSC, showing Transactions, customers and branches, deleting by IFSC and updating Branch manager. |  | Branch, Customer and Transaction Mappers.  Branch, Customer and Transaction Models.  Branch DAO. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: CustomerRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating Customer, deleting customer by ID, getting customer by Account Number, by Branch code or by ID, updating Customer by ID, showing Transactions TypeWise and getting Customer by branches. |  | Customer and Transaction Mappers.  Customer and Transaction models.  Customer DAO |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: EmployeeRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating and deleting employees, getting employee bank, getting employee by ID, showing all employees, updating Employee and getting employee transactions. |  | Bank, Employee and Transaction Mappers.  Bank, Employee and Transaction models.  Employee DAO. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionRepository** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For depositing, showing transactions, withdrawing, depositing in transactions and crediting in transactions |  | Transaction Mapper.  Transaction Model.  Transaction DAO. |

* 1. Services:

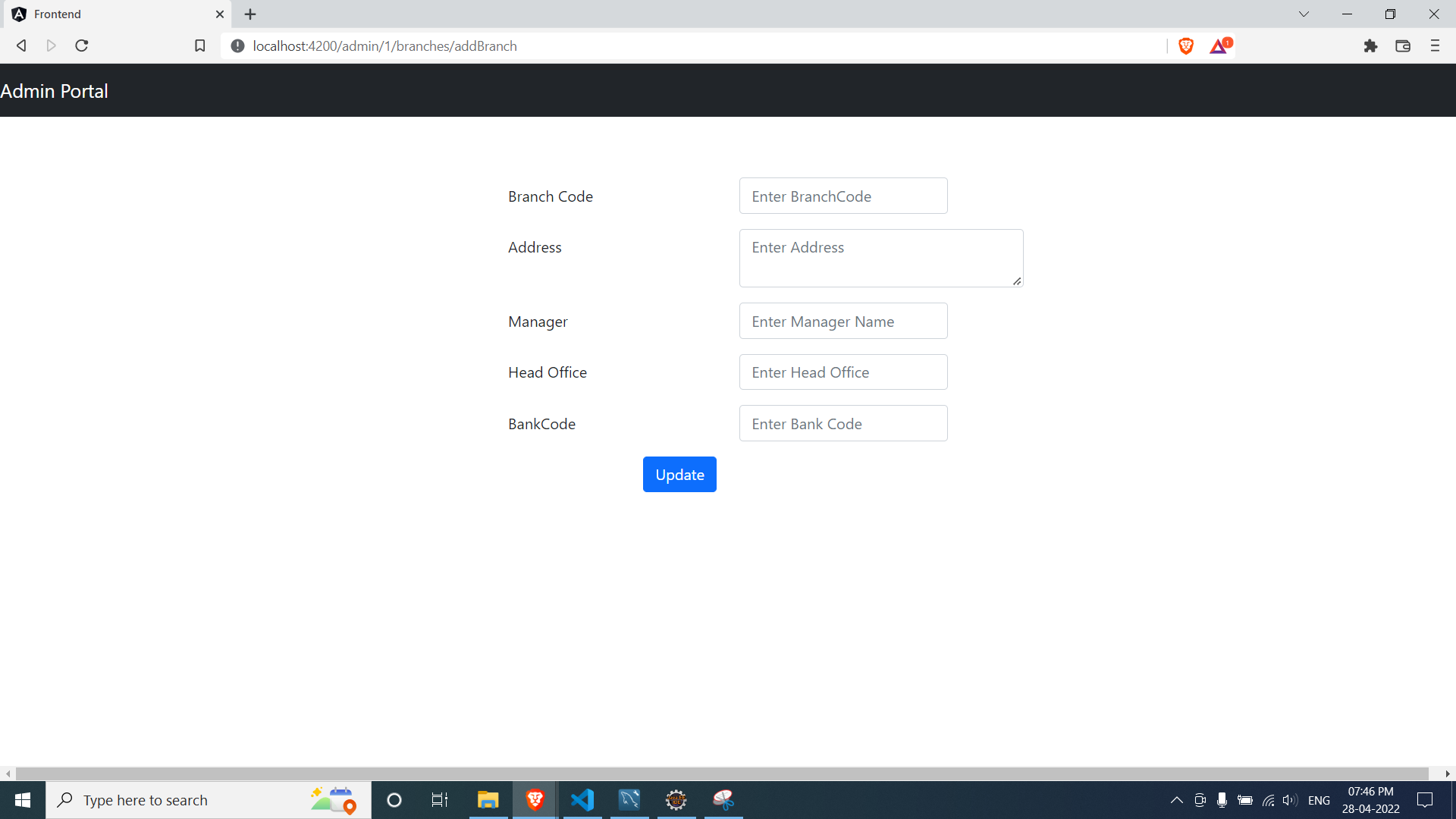
|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BankService** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For Creating Bank and Viewing The details. |  | Bank model.  Bank Repository. |

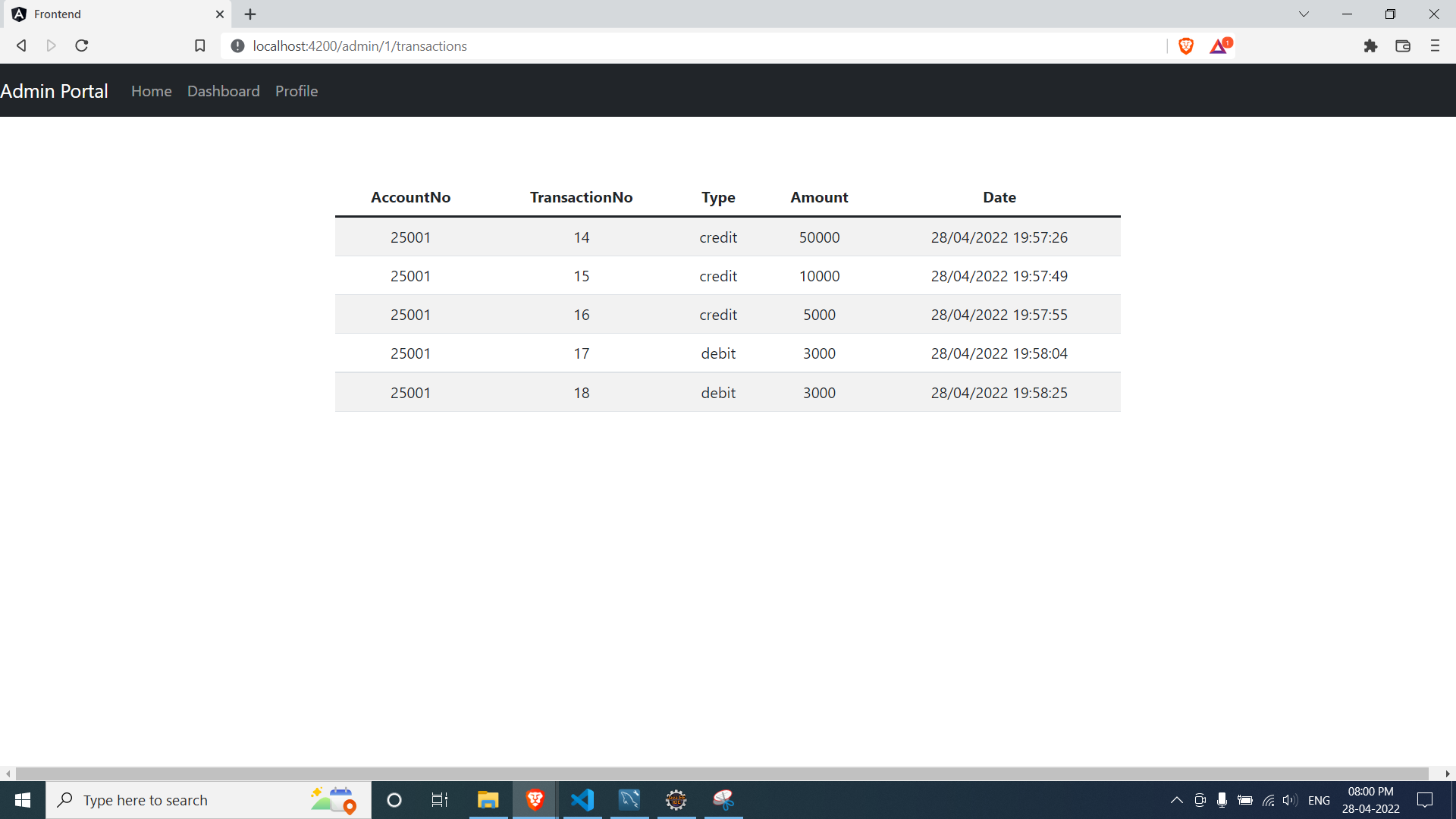
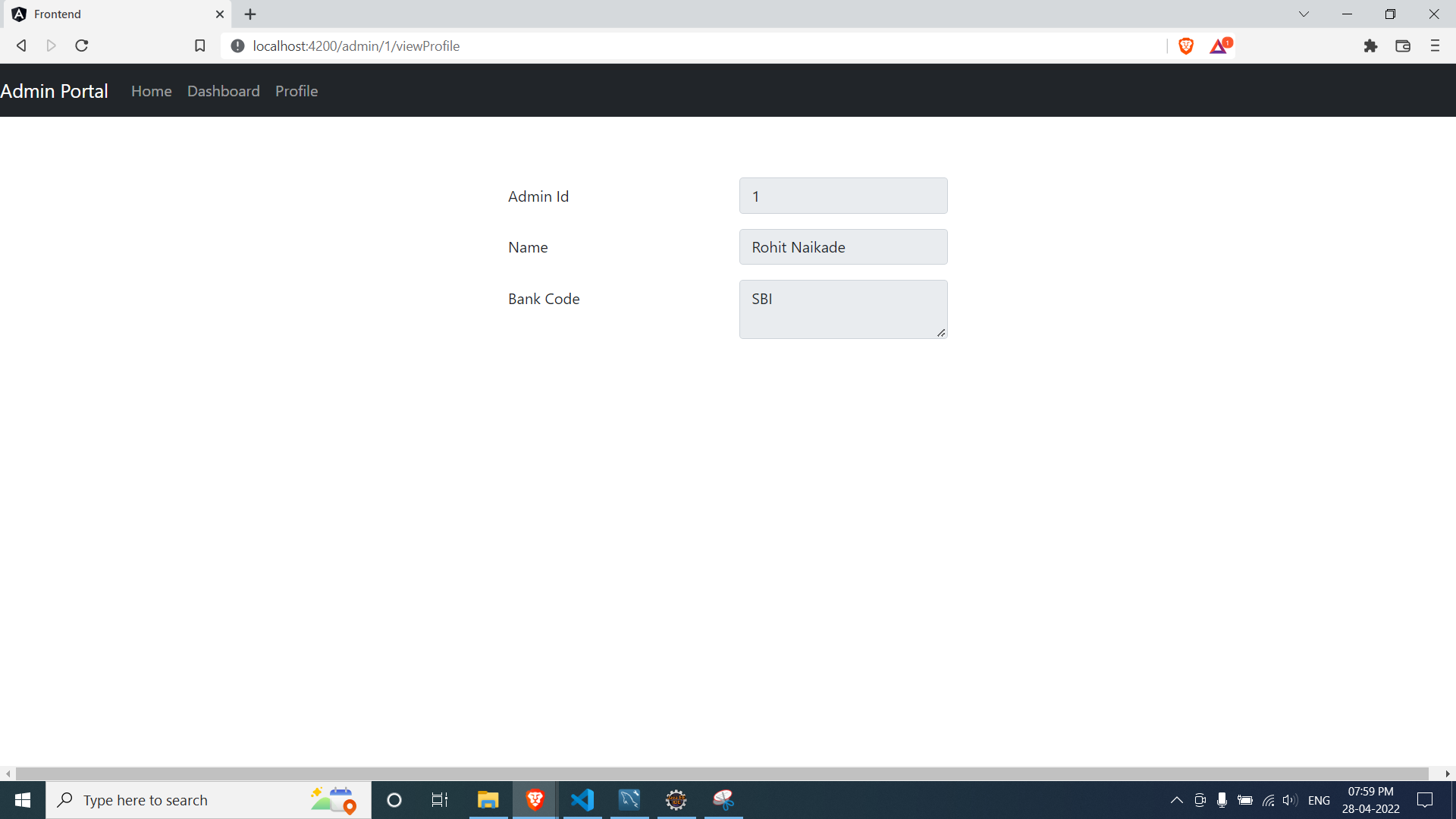
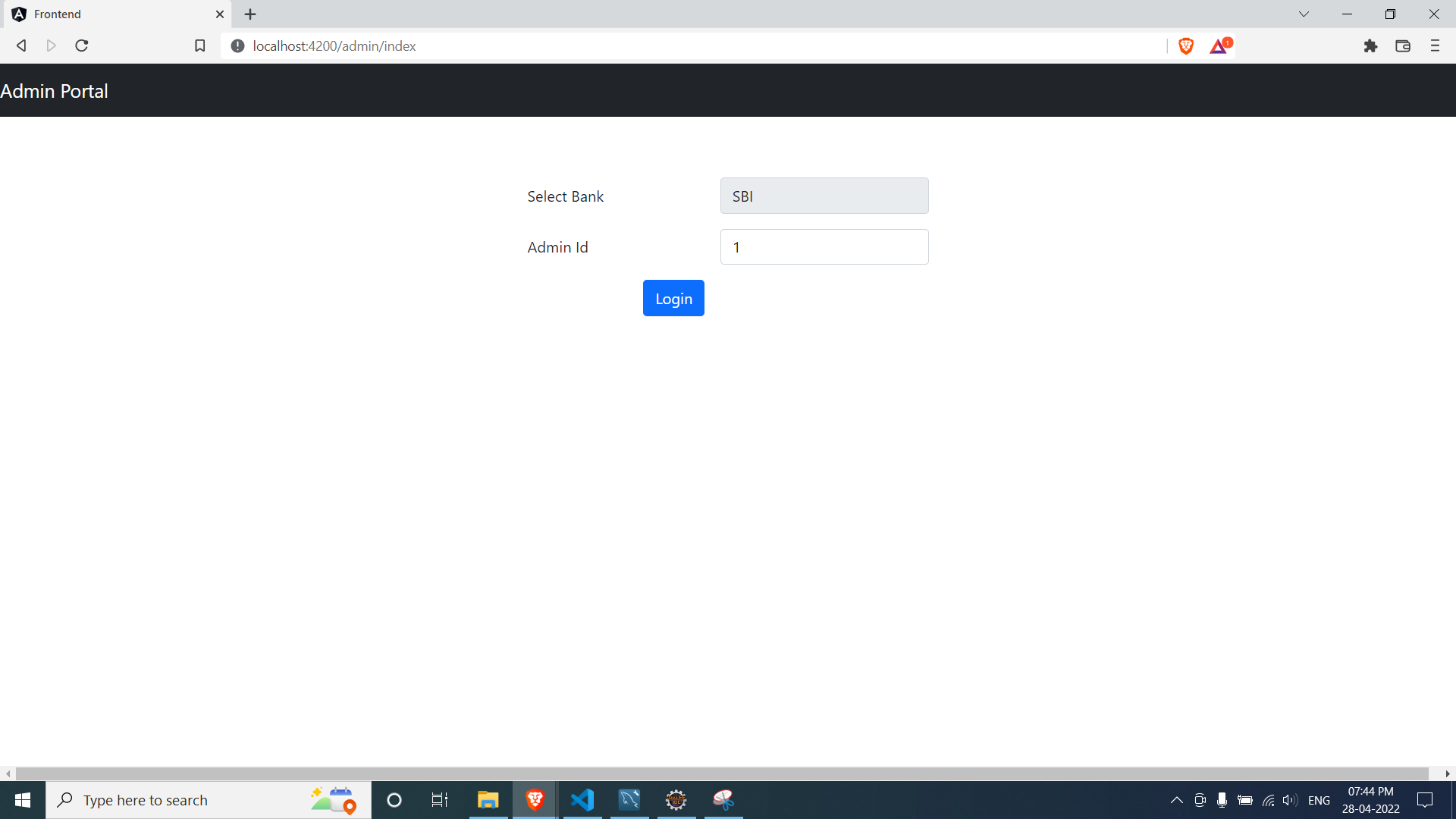
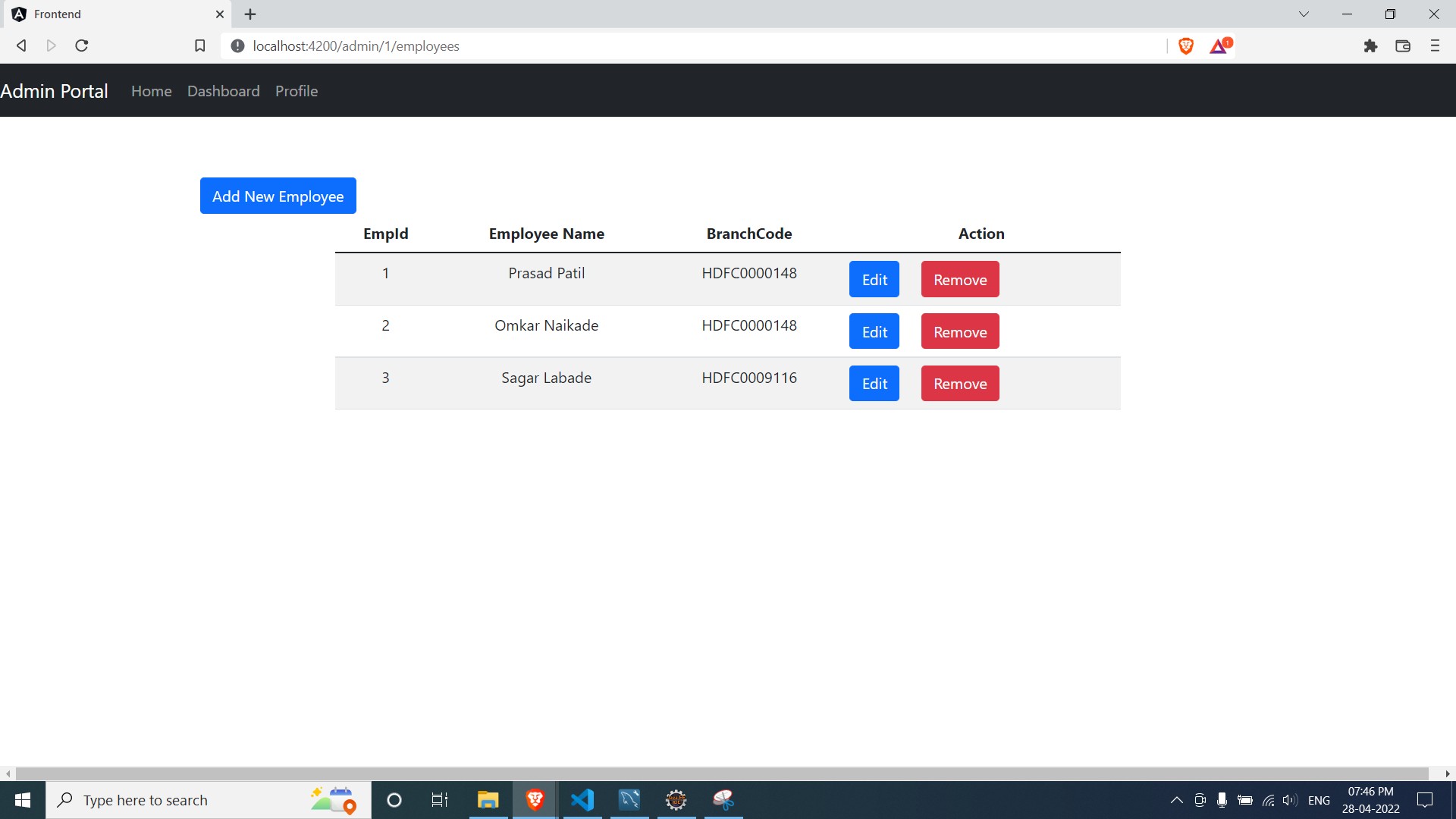
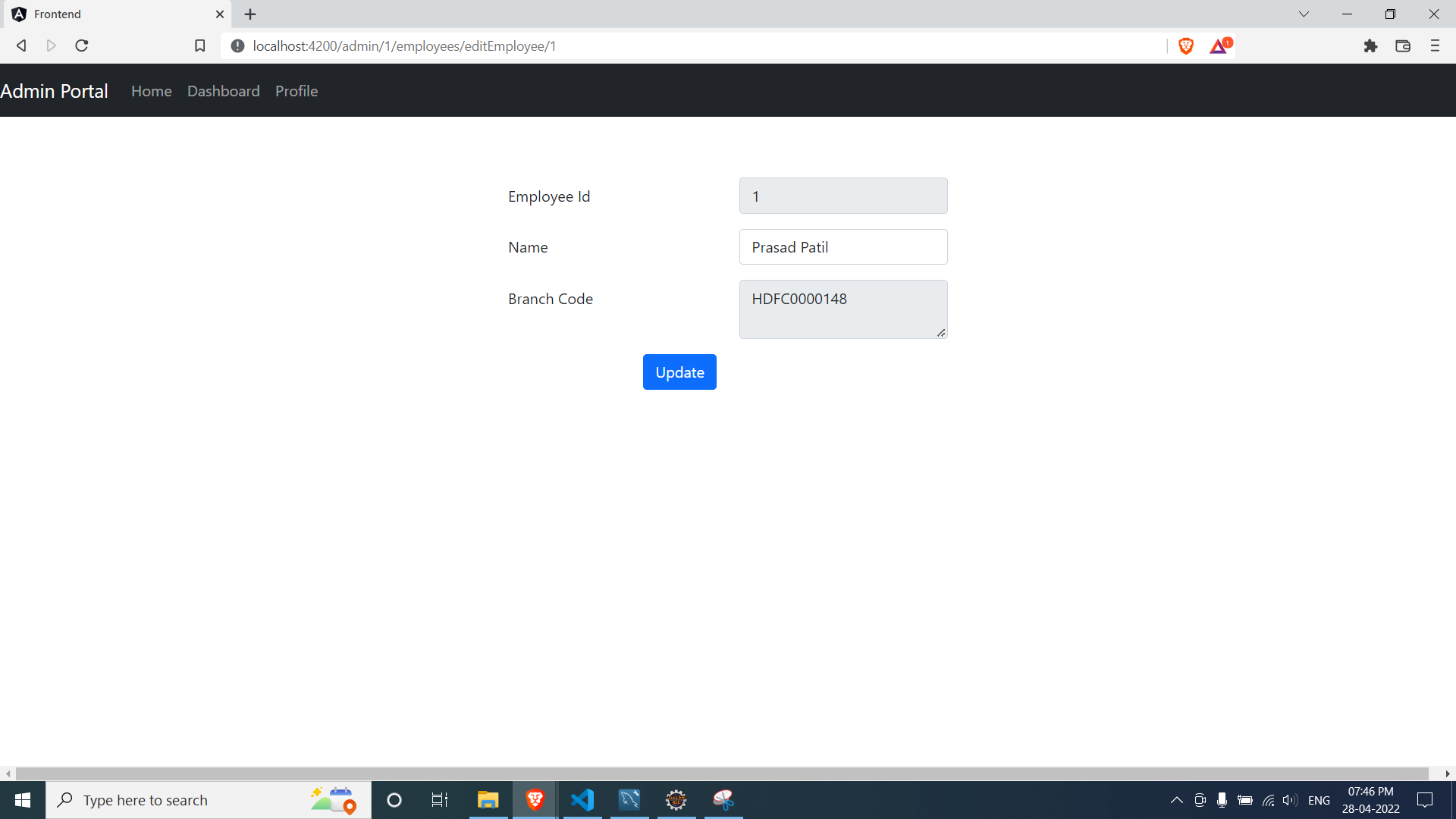
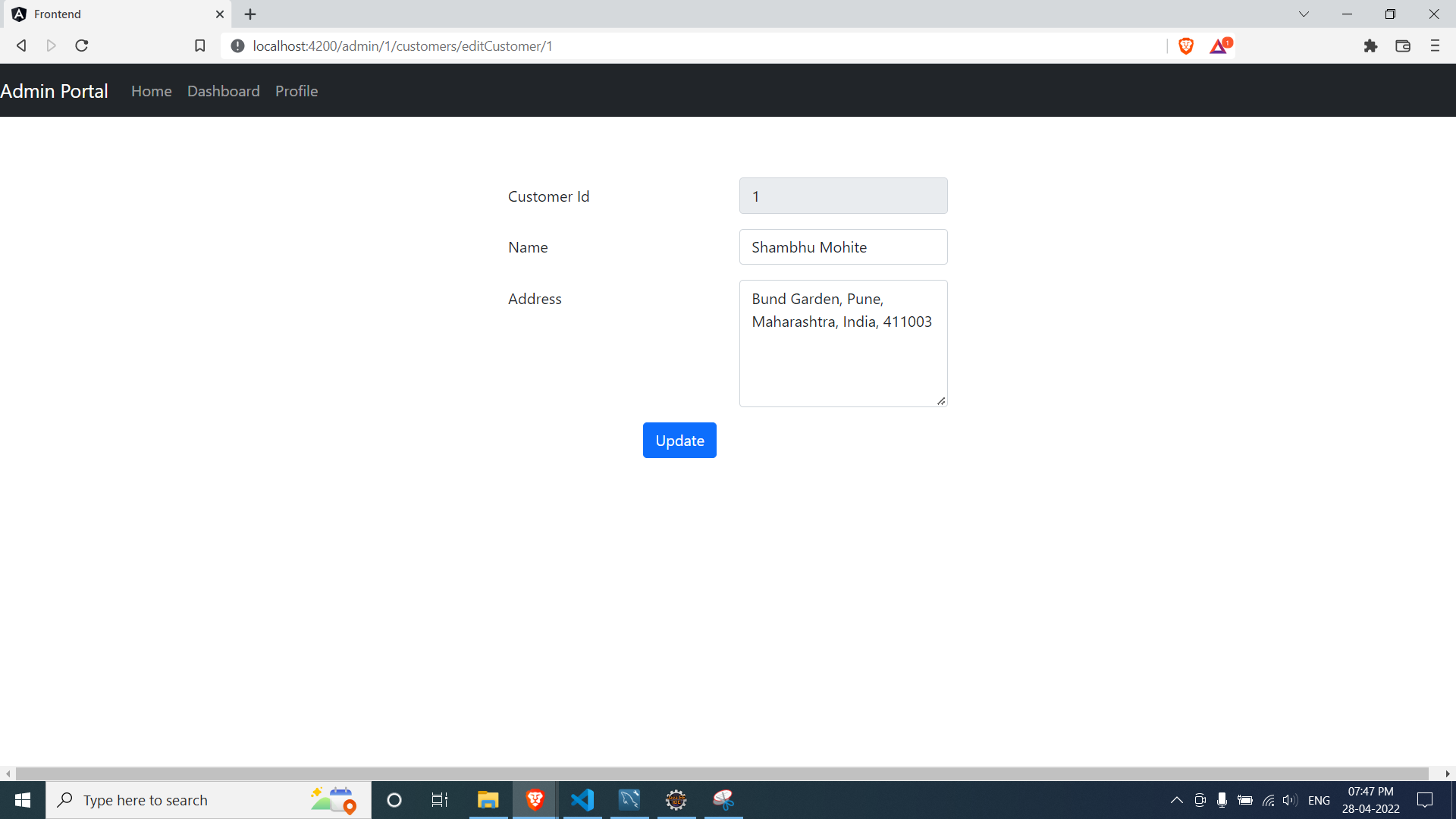
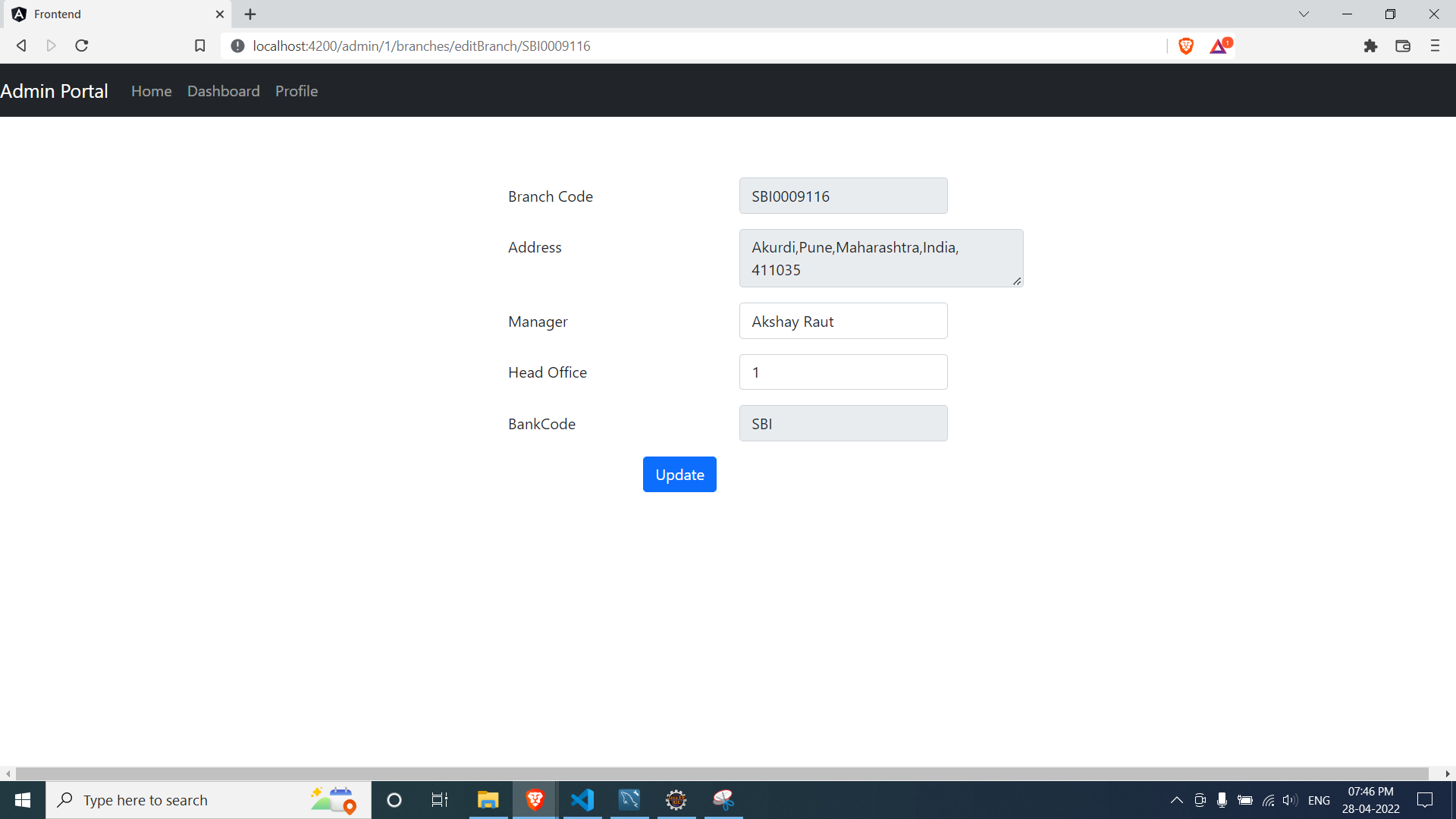
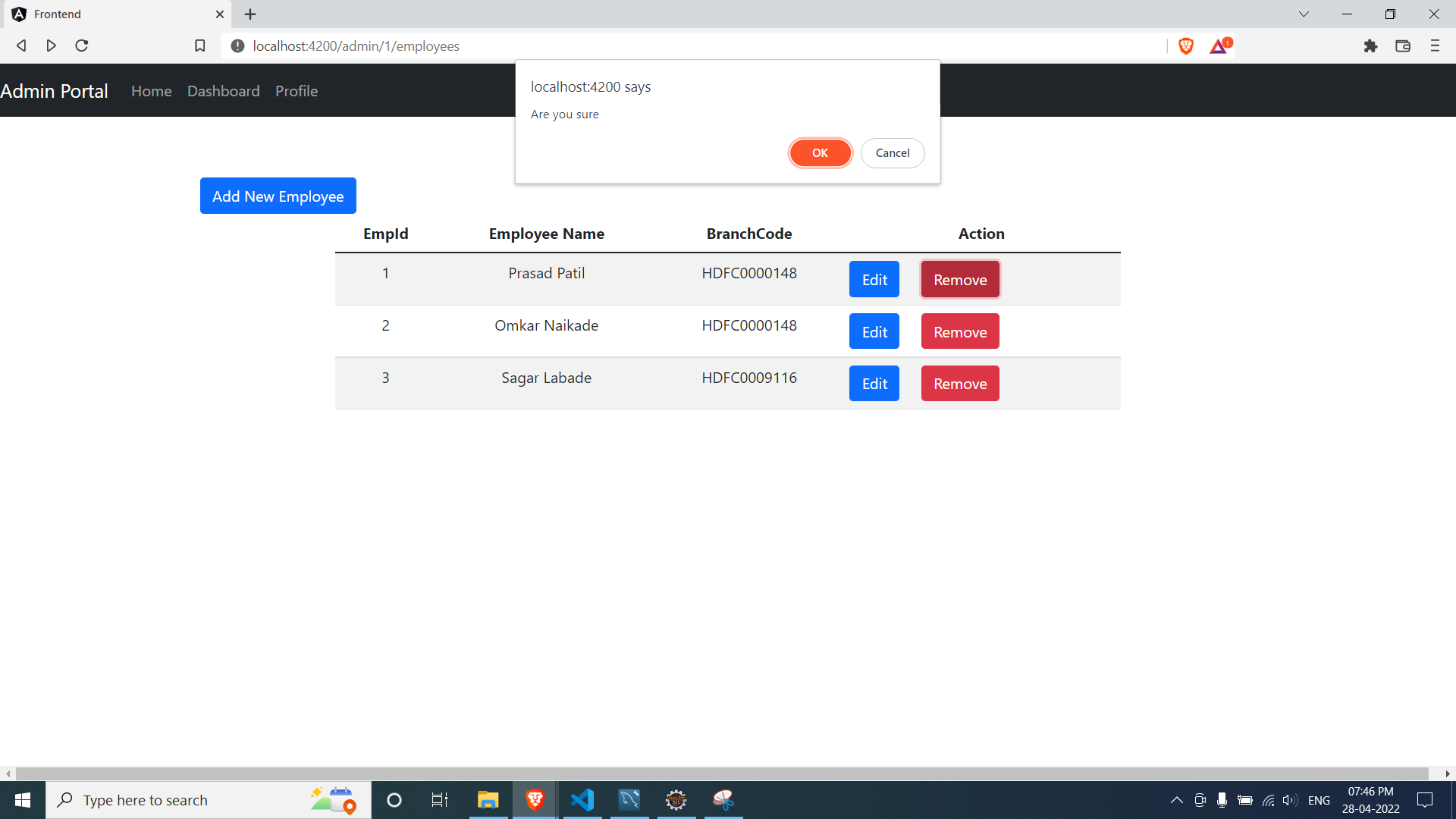
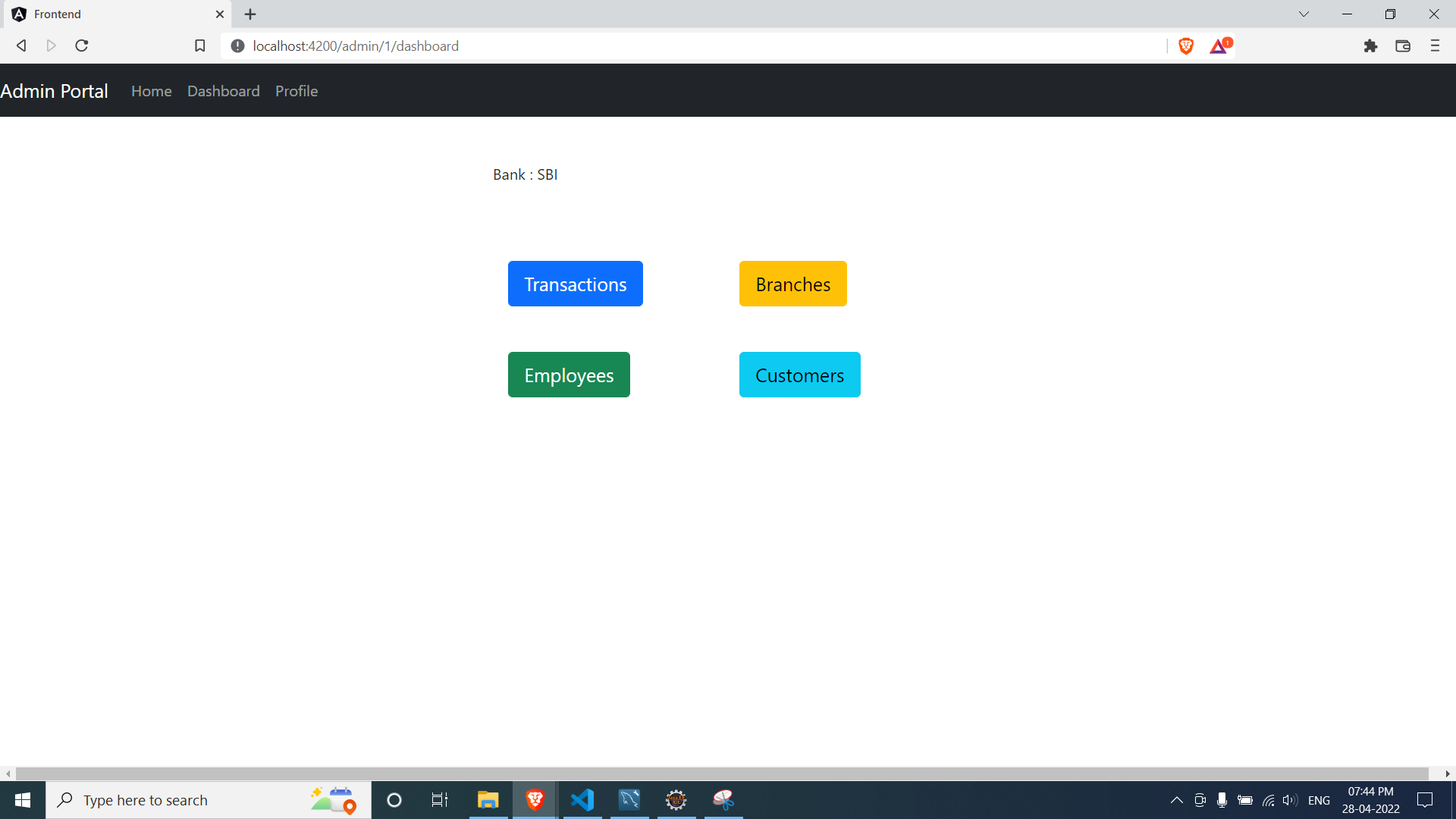
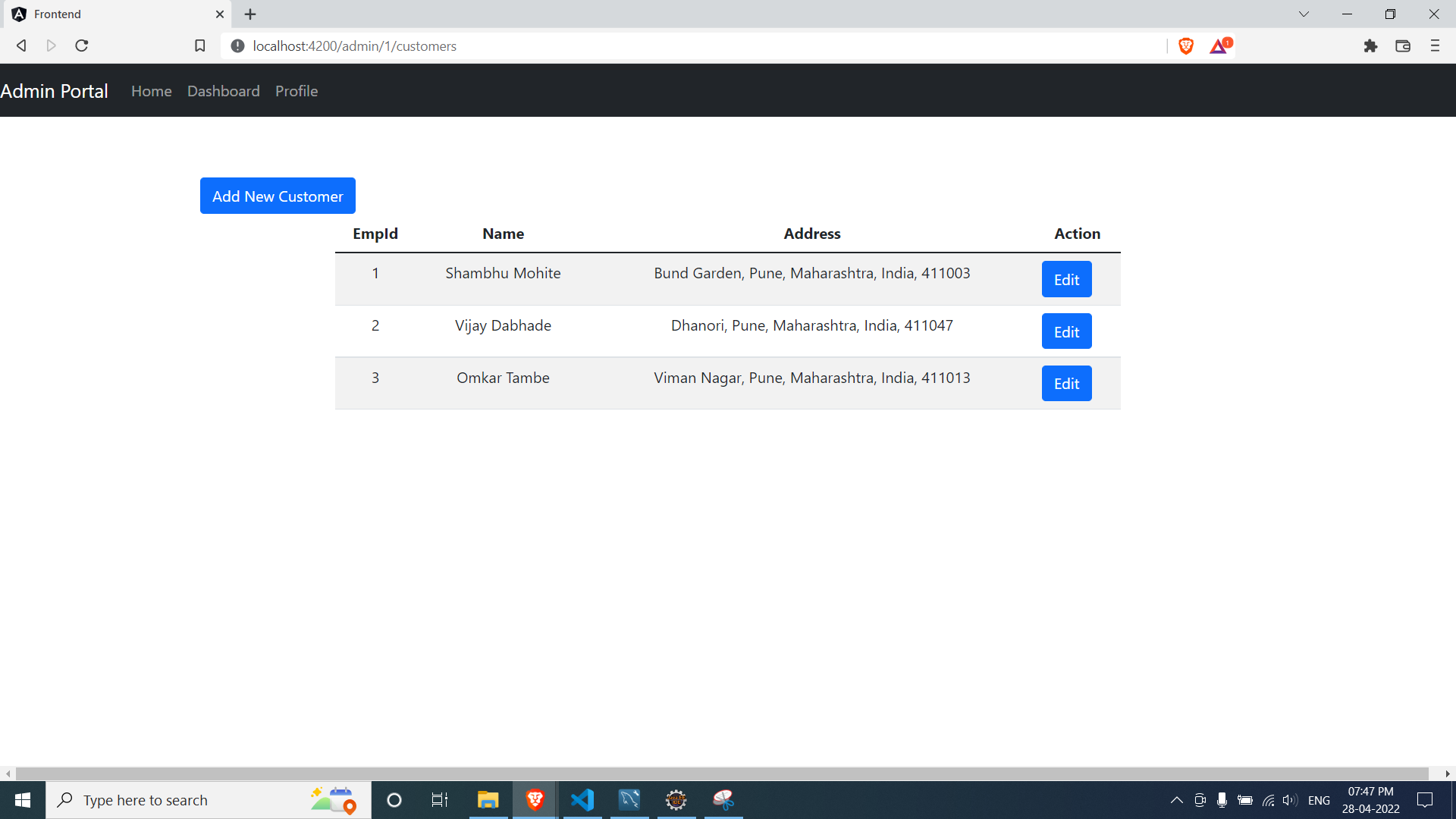
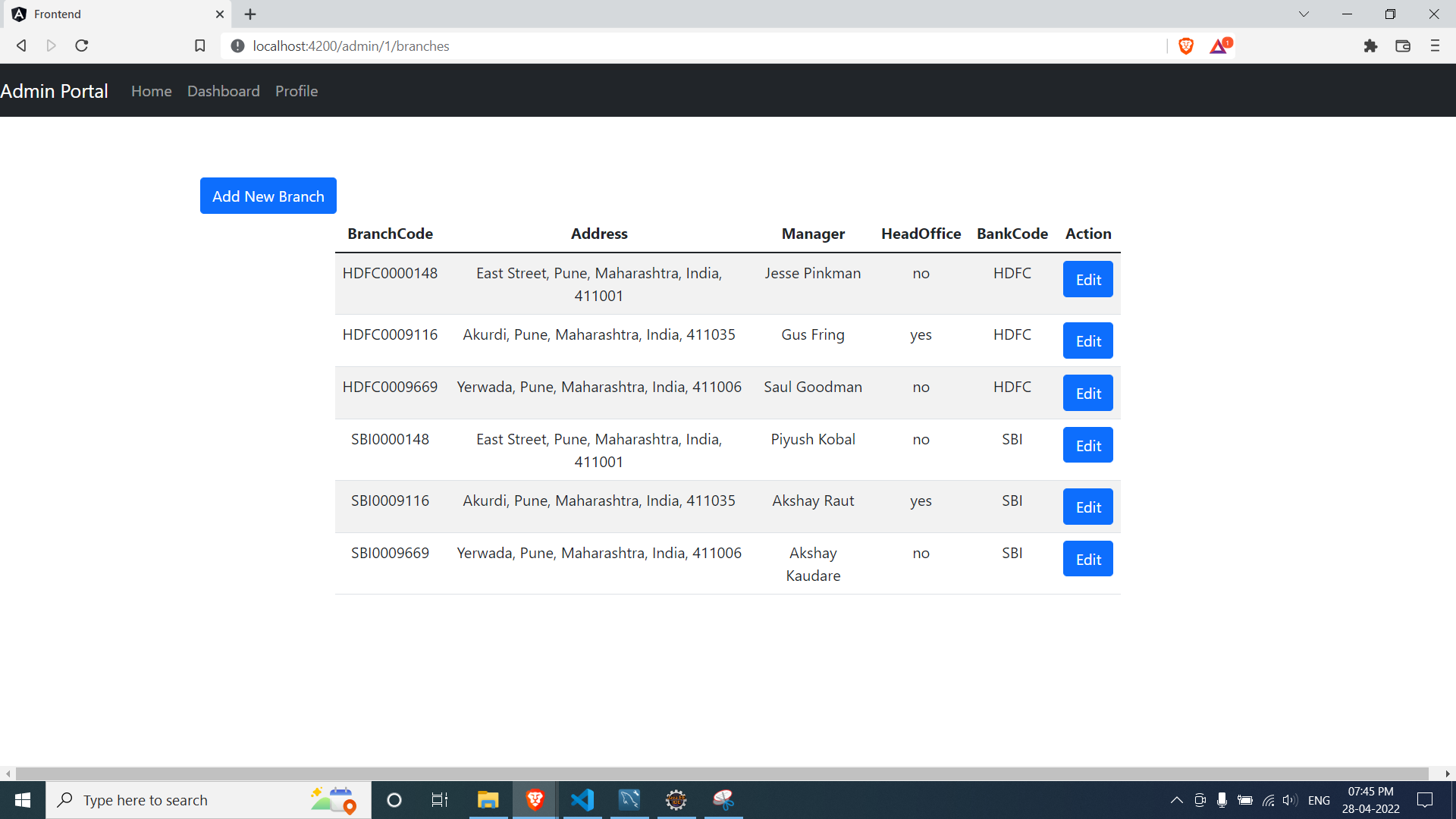
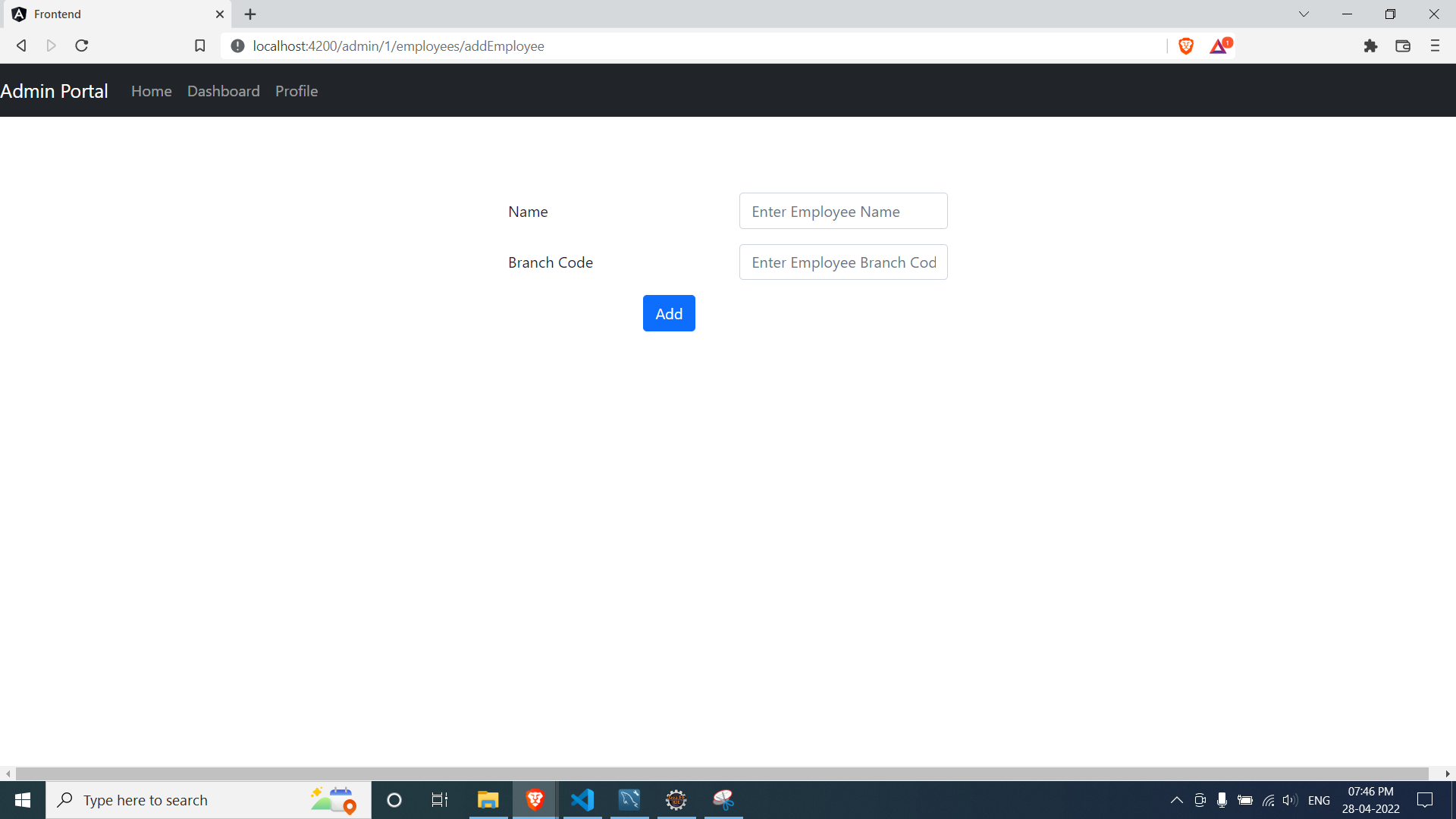
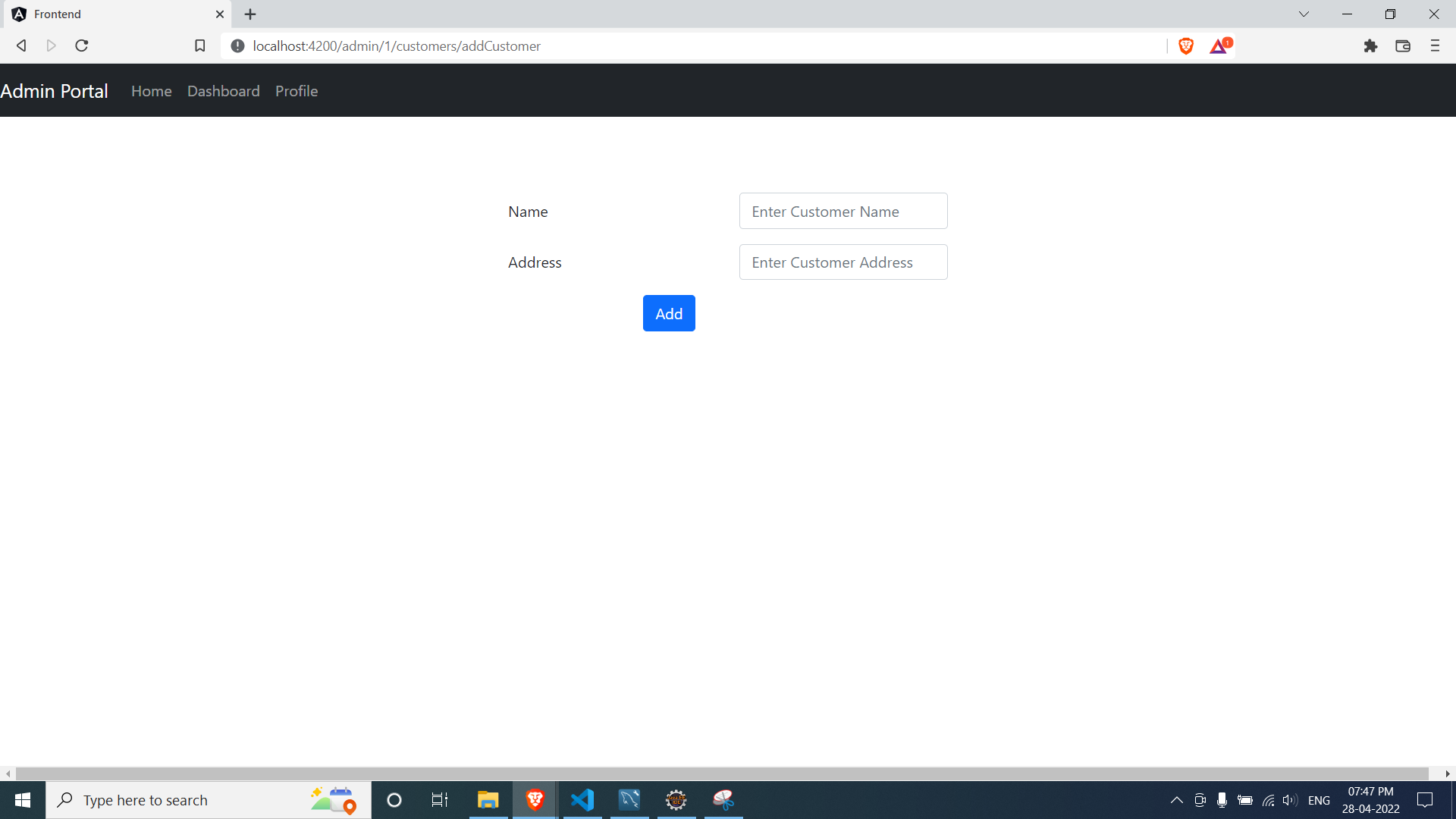
|  |  |  |
| --- | --- | --- |
| **CLASS NAME: BranchService** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For creating Branch, get the branches, get the branches by IFSC, deleting Branch by IFSC, updating Branch manager name and getting Customer by Branches. |  | Branch and Customer models.  Branch and Customer Repository. |

|  |  |  |
| --- | --- | --- |
| **CLASS NAME: TransactionService** | | |
| RESPONSIBILITIES | DEPENDABLES | DEPENDENCIES |
| For depositing, withdrawing and getting all the transactions. |  | Transaction Model.  Transaction Repository. |

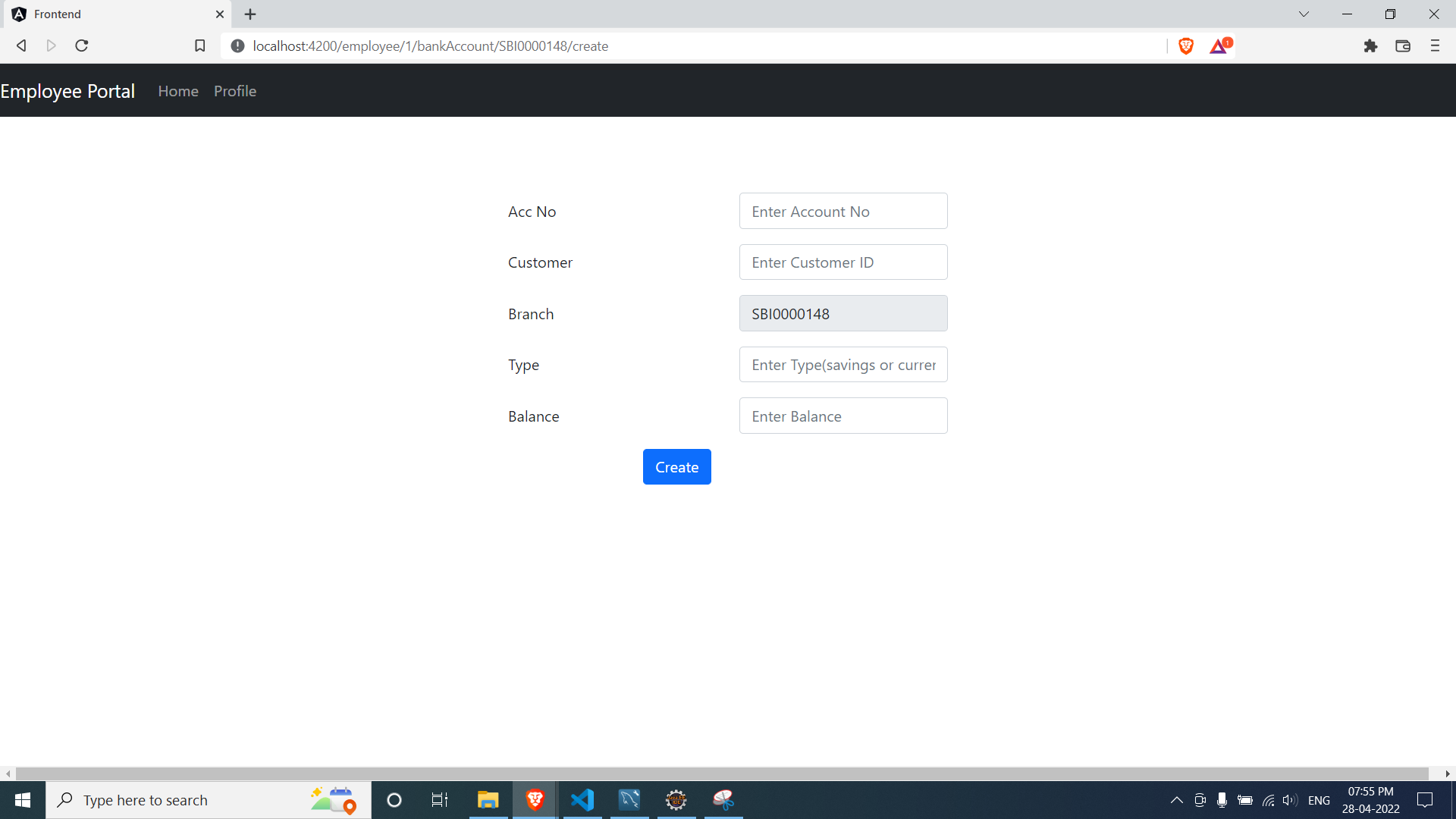
1. **IMPLEMENTATION:**

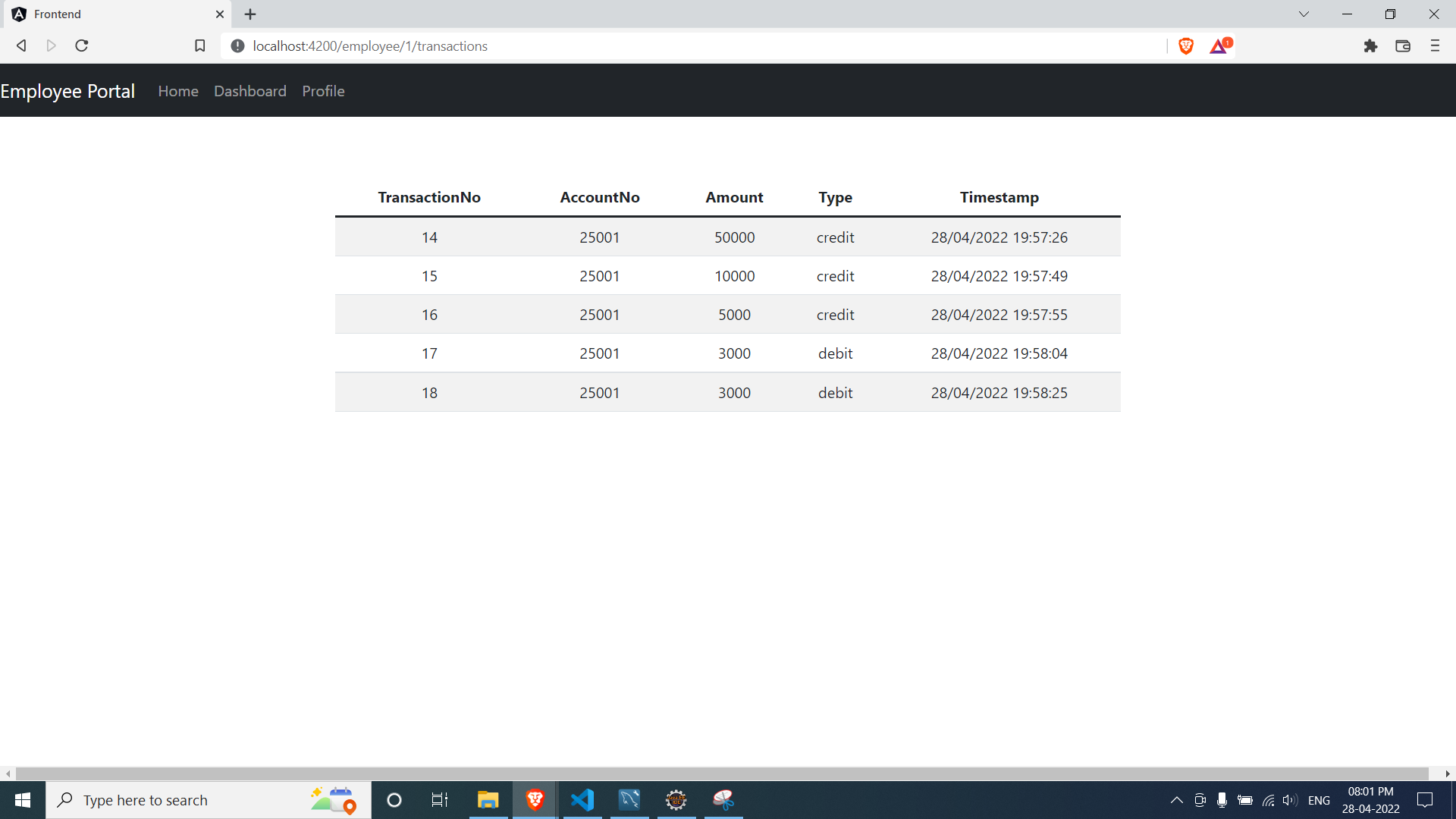
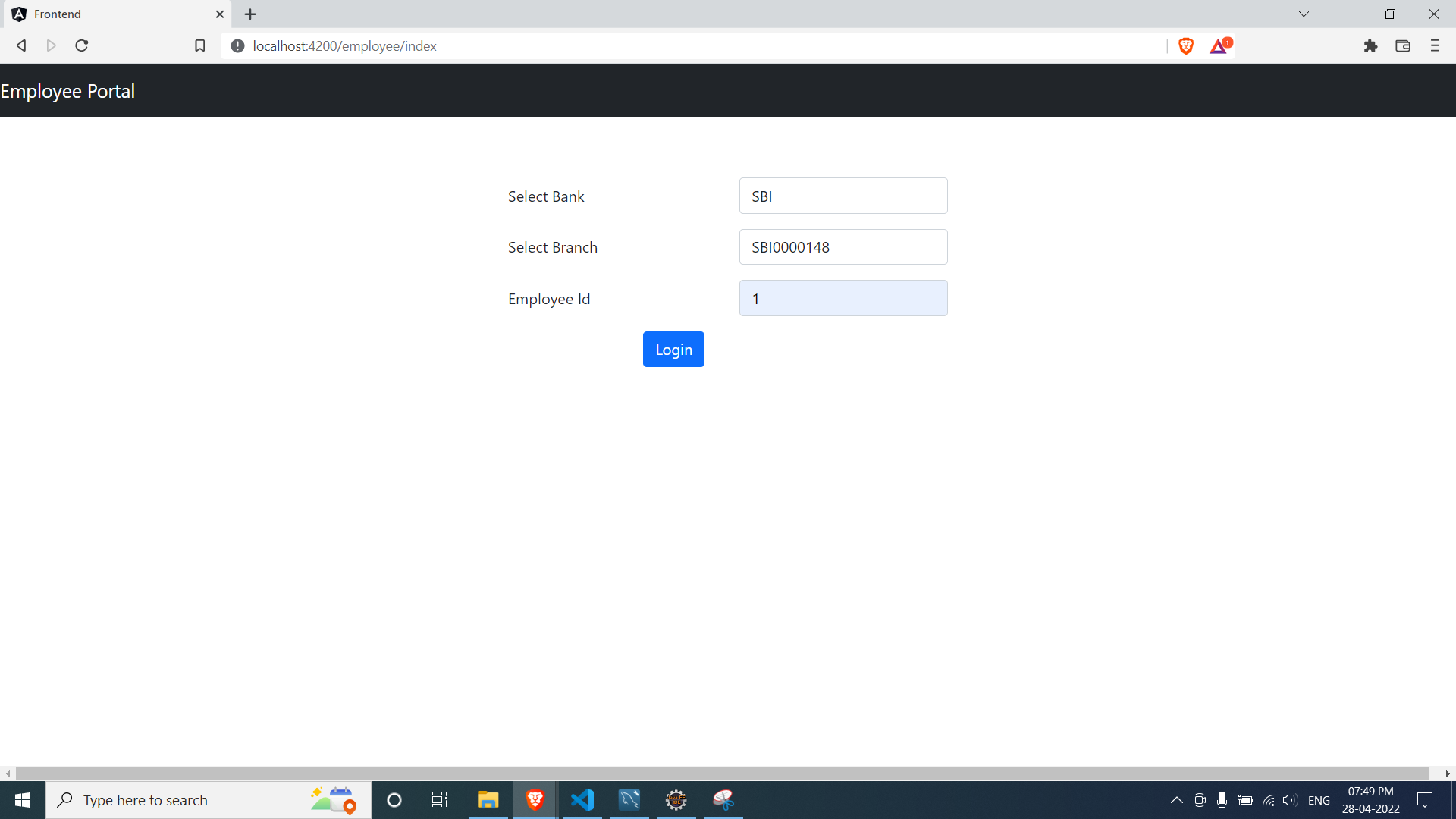
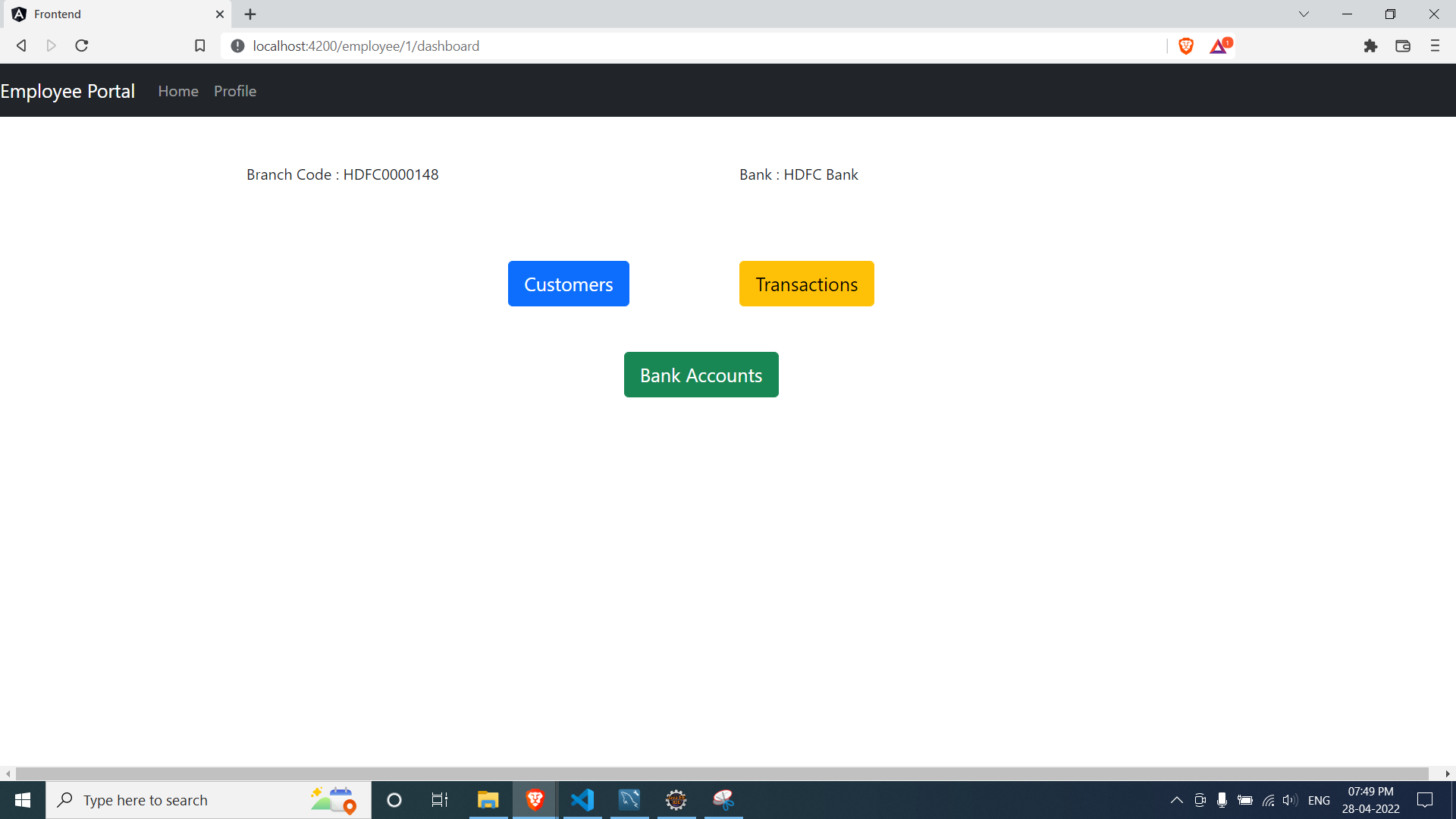
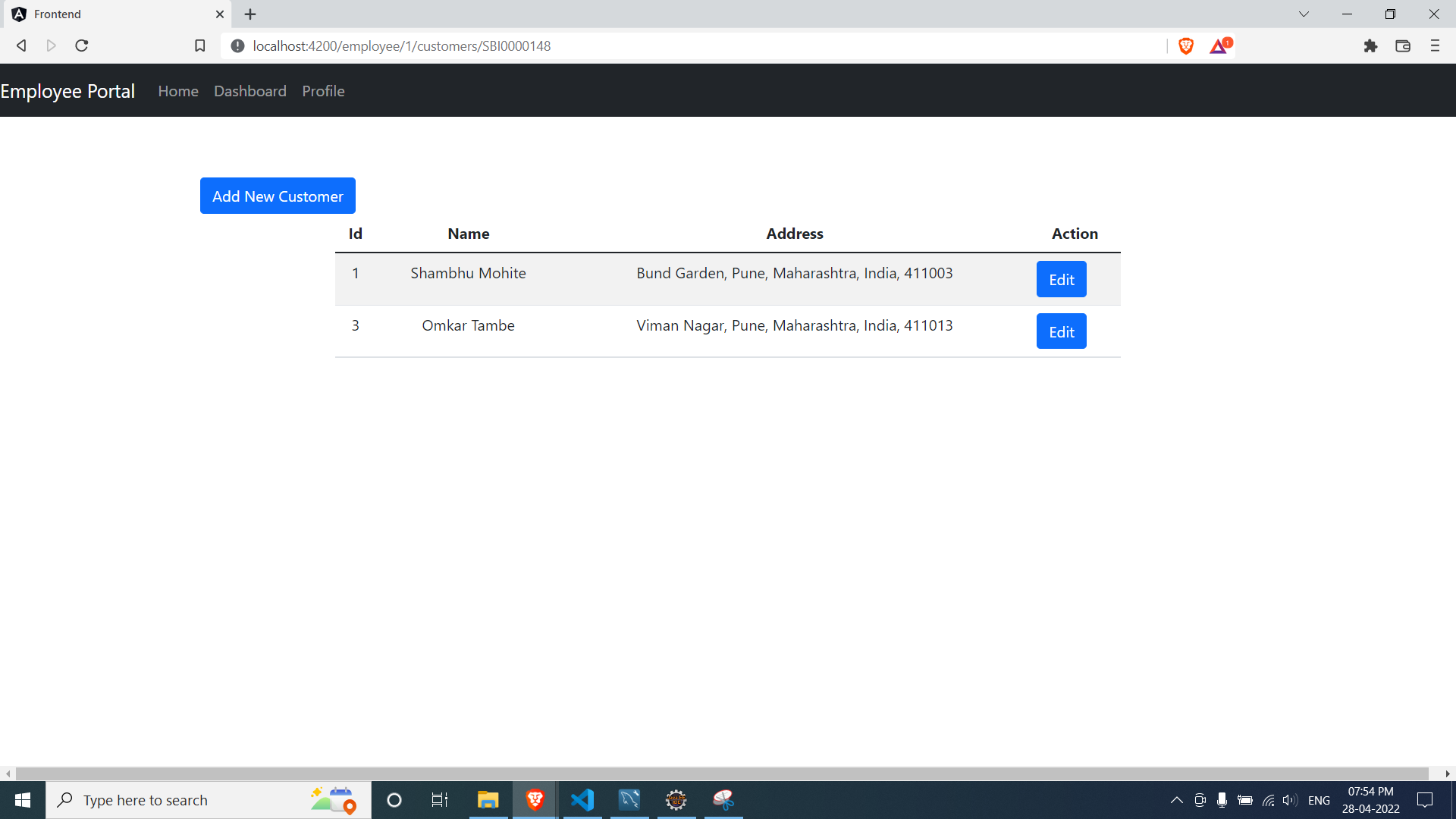
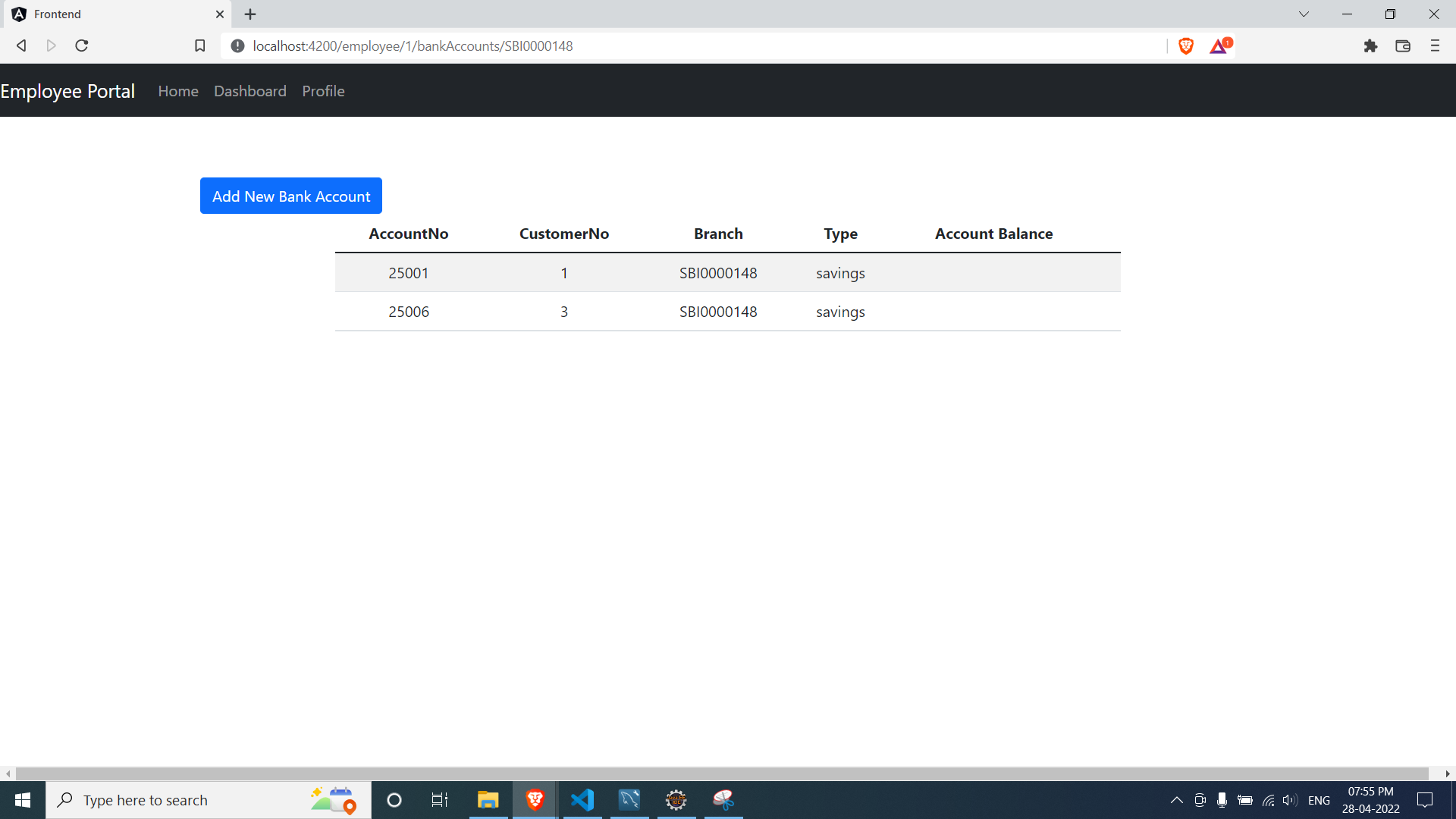
**Admin:**



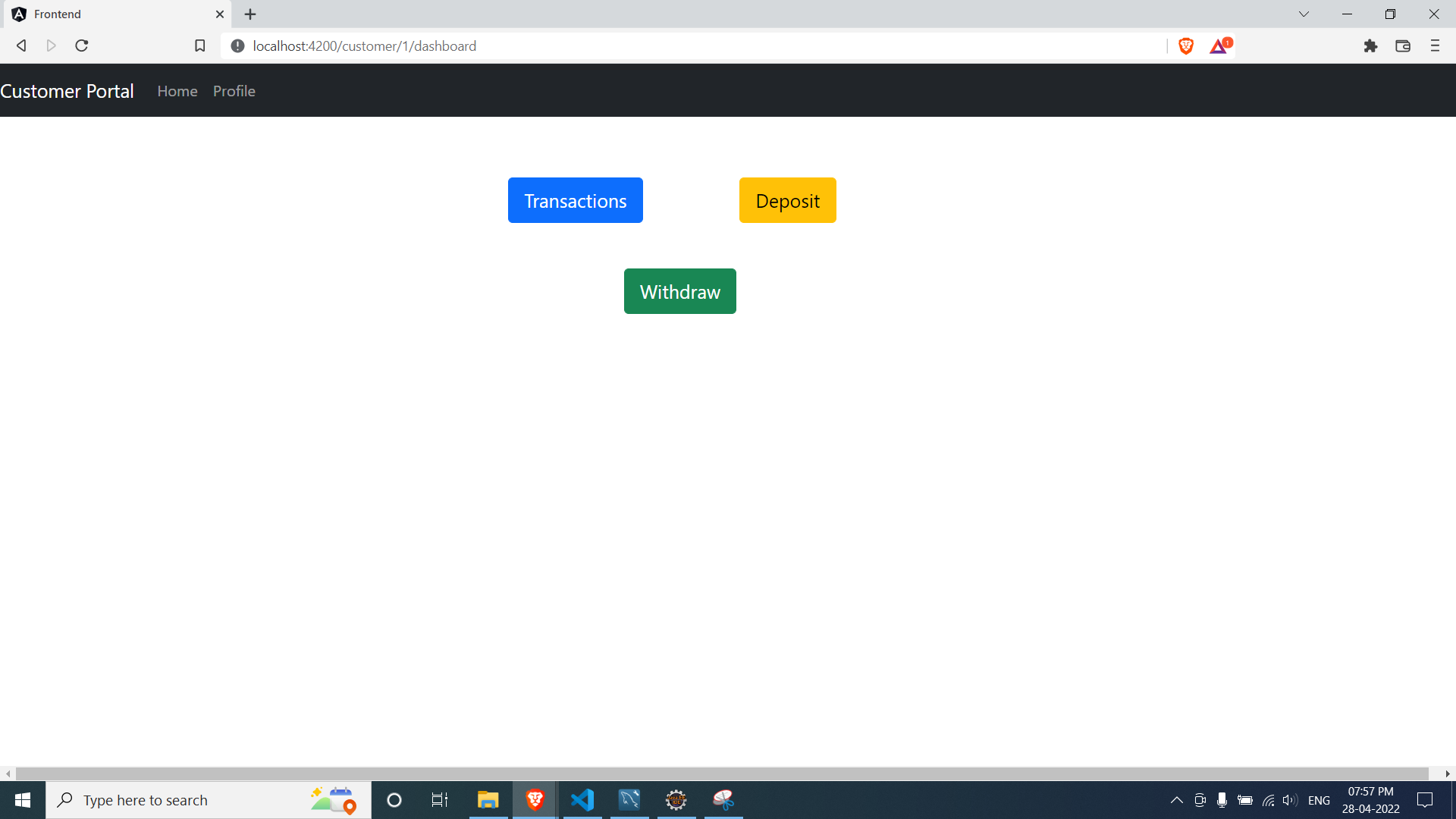


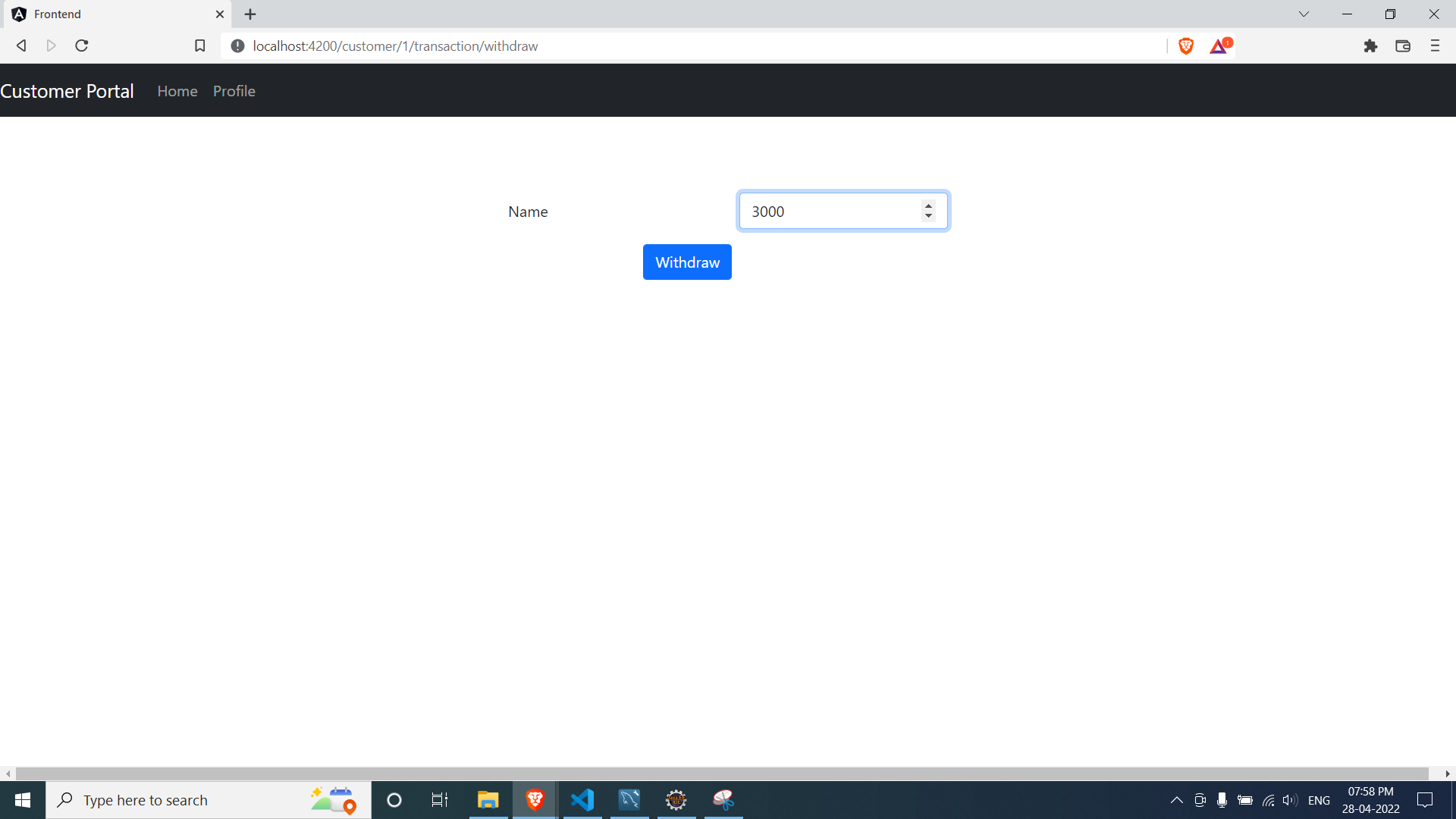
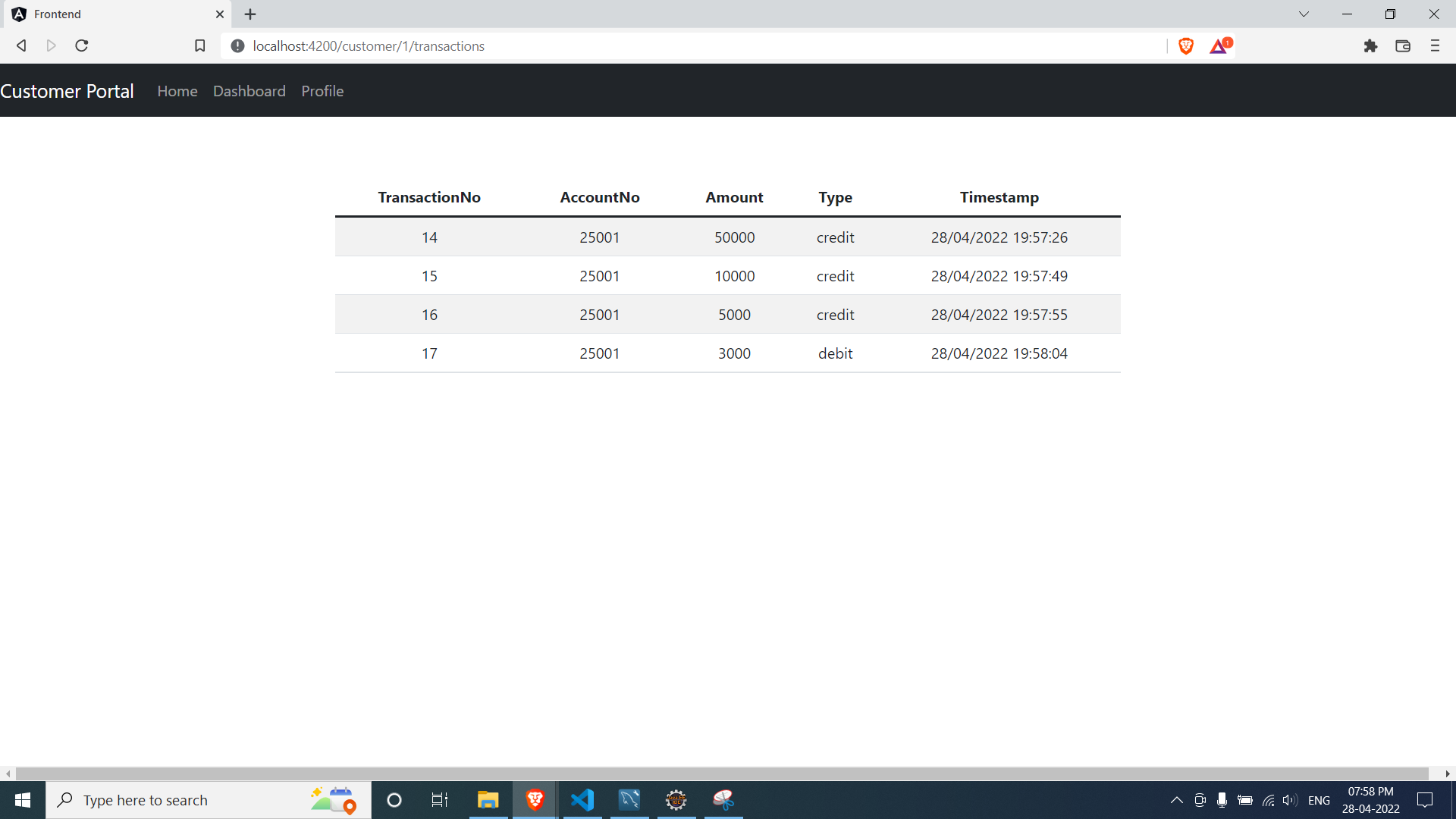
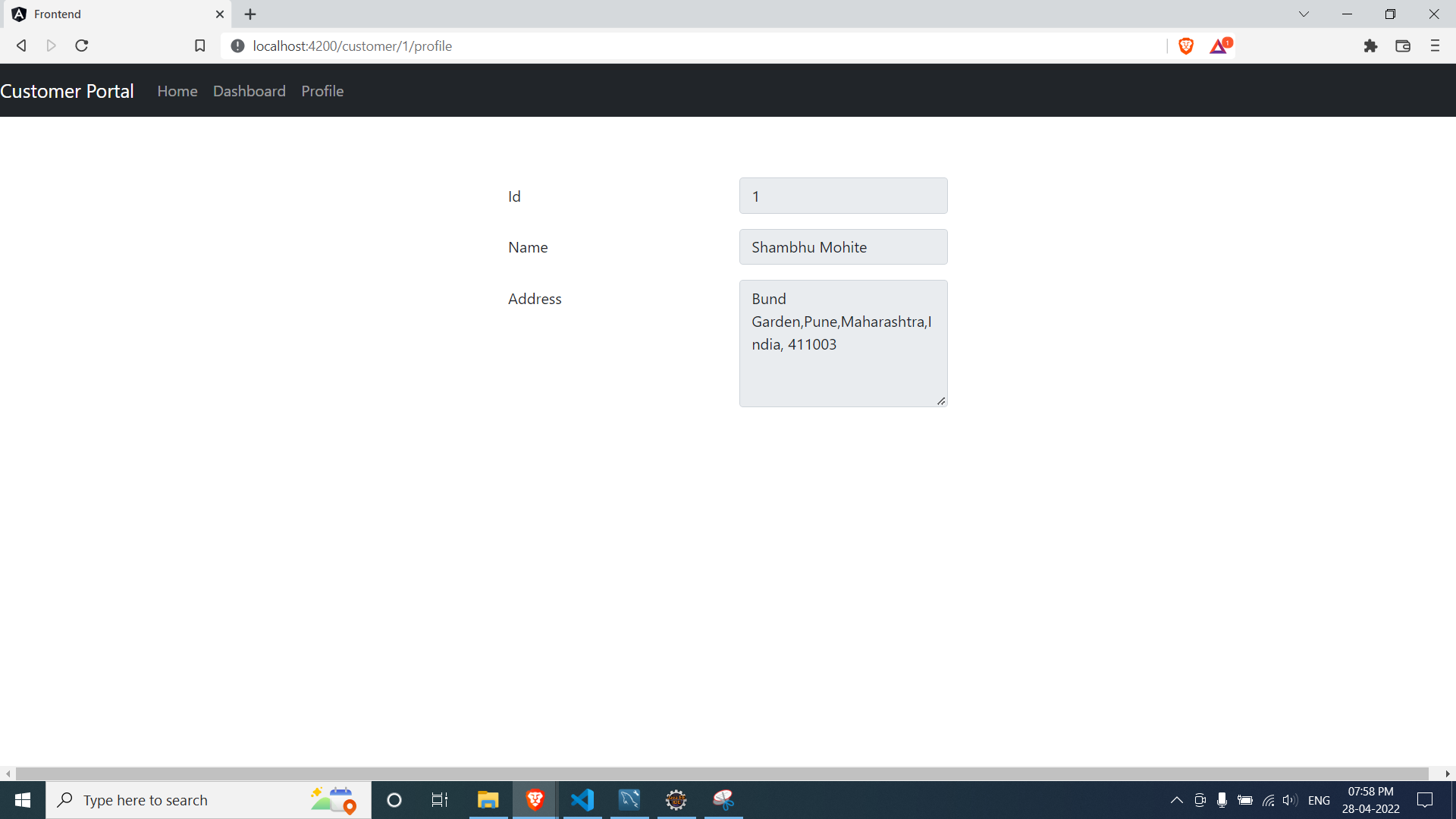
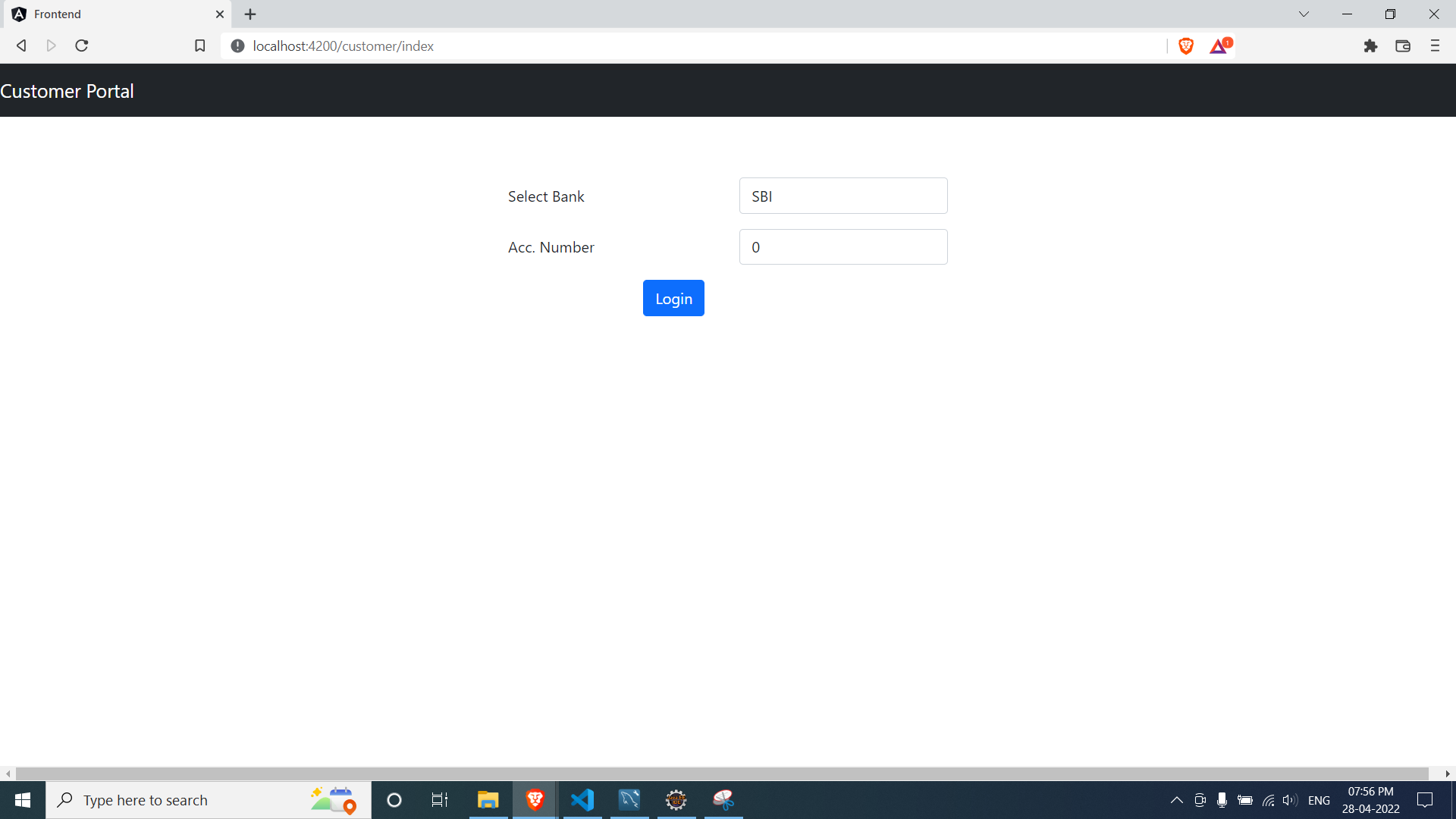
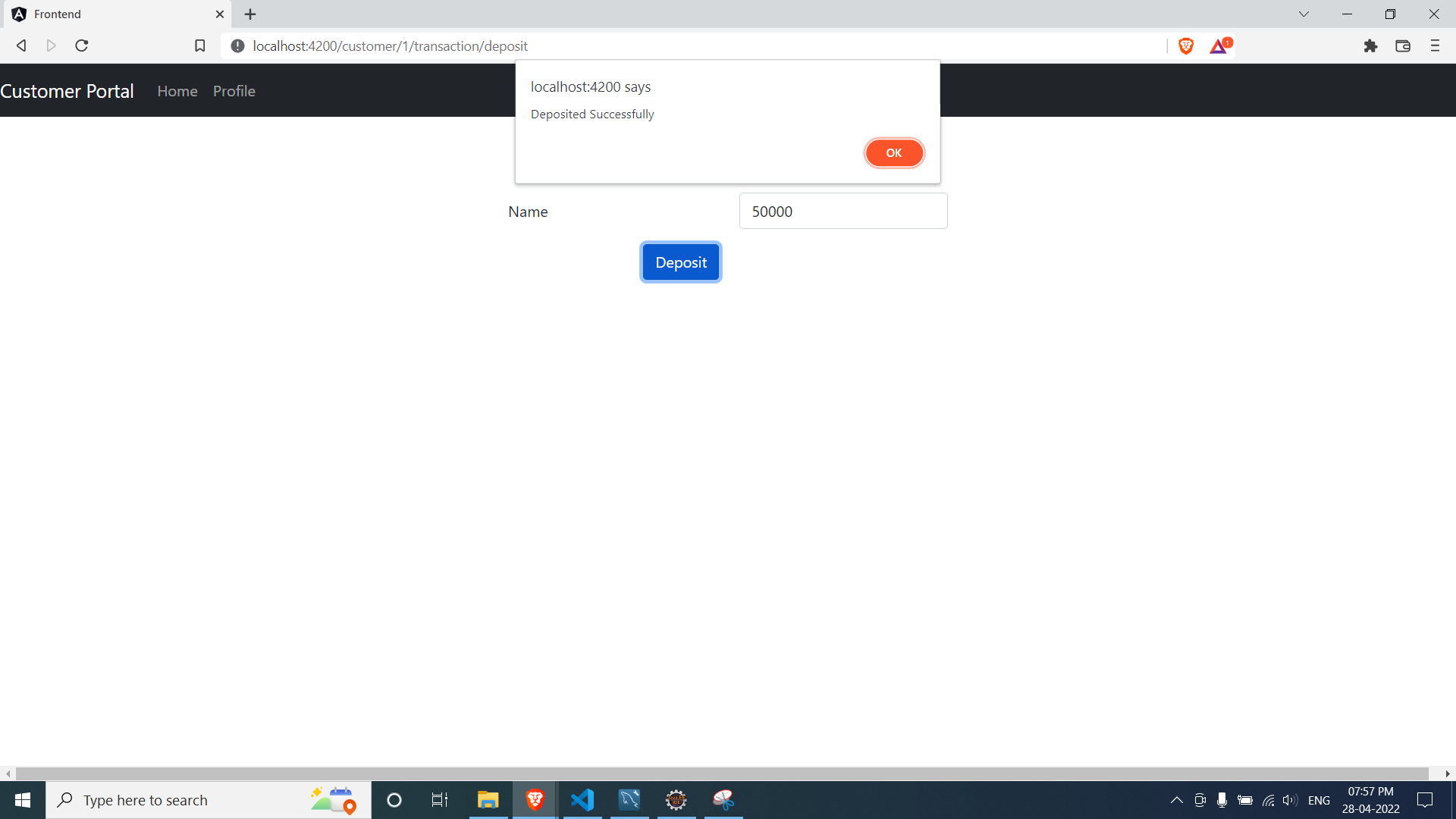
**Employee:**





**Customers:**





1. **CONCLUSION:**

This banking system allows customers to view, edit, and update their profile along with withdrawal and deposit of funds in their bank accounts. The system allows the employees and the administrator to view all transactions as well as customer details. The admin of the system can also create, update, or delete.

1. **REFERENCES:**
   1. stackoverflow.com