CAPSTONE PROJECT: BANKING SYSTEMS

1. SYSTEM REQUIREMENT SPECIFICATION:

1.1. Software Requirements:

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

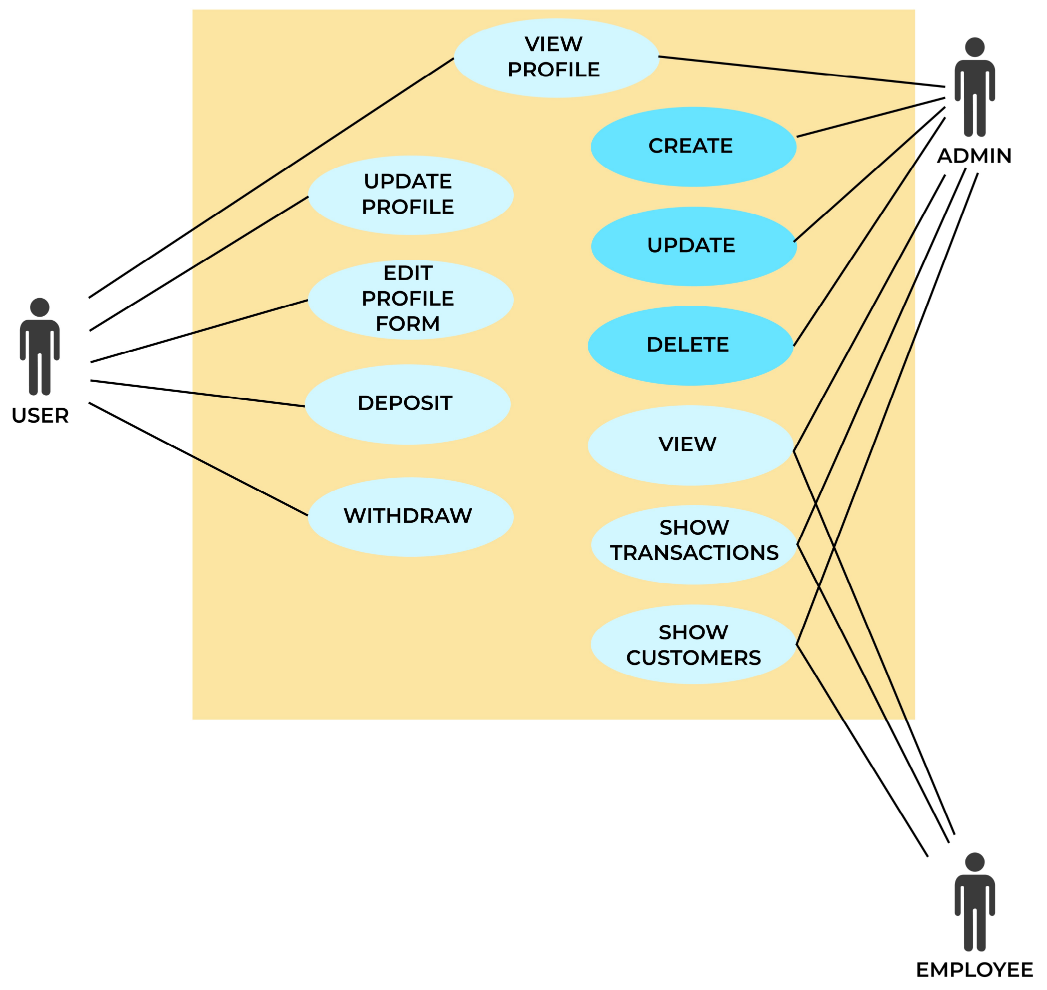
1.2. Hardware Requirements:

- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Processor: Intel core^{TM} i5
- Installed Memory (RAM): 8.00 GB

2. 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.

3. 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:

a) User

b) Admin

c) Employee

4. FLOW OF EVENTS:

- **Basic flow**

4.1. Withdraw:

4.1.1. User enters customer Id.

4.1.2. Bank Database validates the user.

4.1.3. On success user can withdraw money.

4.2. Deposit:

4.2.1. User enters customer Id.

4.2.2. Bank Database validates the user.

4.2.3. On success user can deposit money.

4.3. View Profile

4.3.1. User/Admin/Employee enters Id.

4.3.2. Bank Database validates the user.

4.3.3. On success User/Admin/Employee can view profile. (They can also change their own profile detail)

4.4. View Transactions:

4.4.1. Admin/Employee logs in

4.4.2. Bank Database validates the Admin/Employee.

4.4.3. After successful validation, Admin/Employee can view all transaction details.

4.5. Show Customers:

4.5.1. Admin/Employee logs in

4.5.2. Bank Database validates the Admin/Employee.

4.5.3. After successful validation, Admin/Employee can view all customers and their details.

- **Alternate Flow:**

4.6. 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.

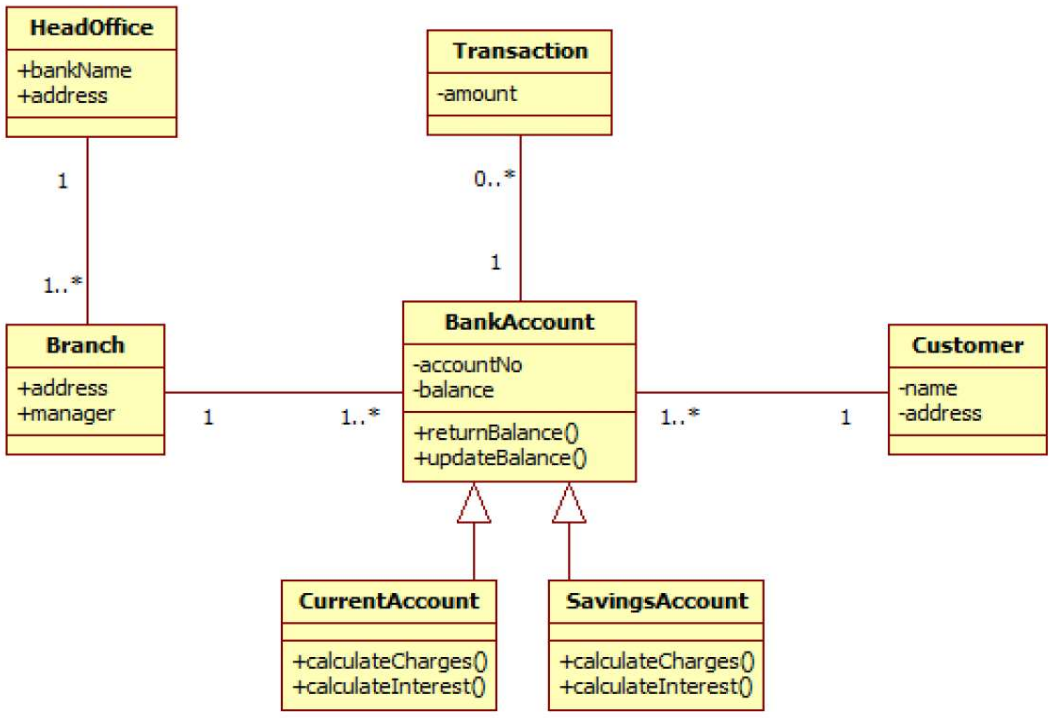
- **Pre-conditions:**

4.7. The user must have a valid account in the bank.

- **Post-conditions:**

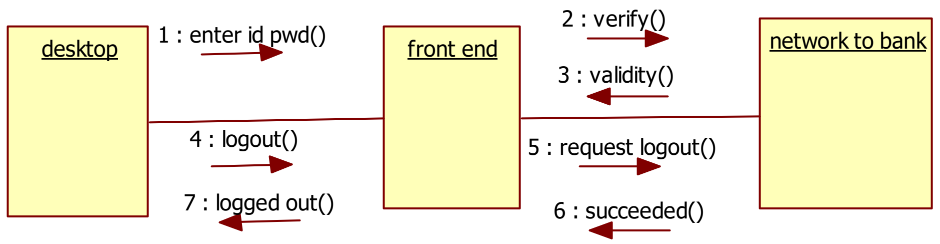
4.8. The database is modified after transaction.

5. CLASS DIAGRAM:

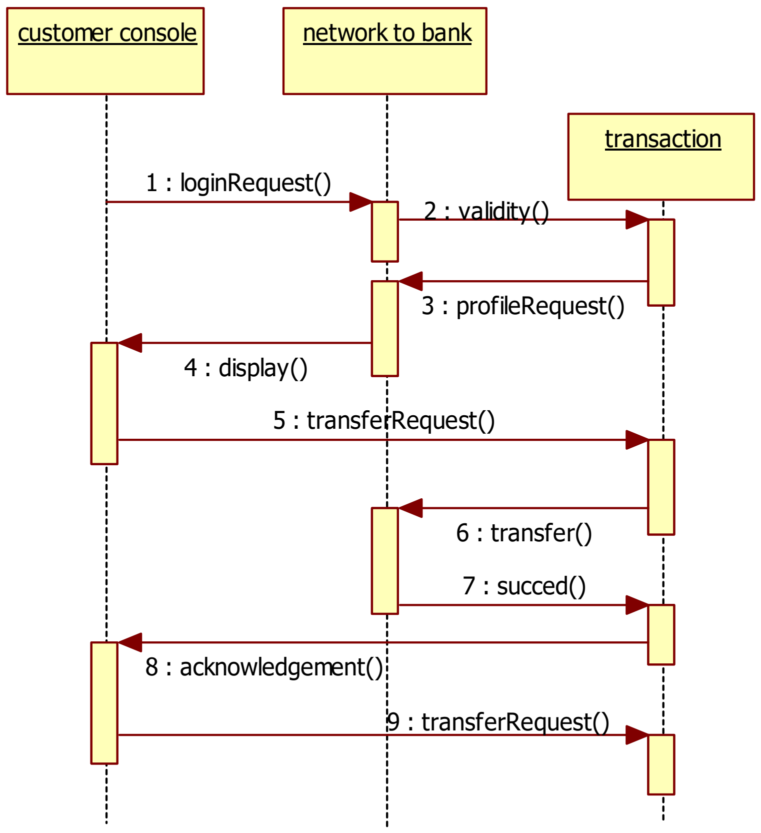


6. SEQUENCE DIAGRAM:

6.1. LOGIN & LOGOUT COLLABORATION DIAGRAM:



6.2. ONLINE TRANSACTION SEQUENCE



7. IDENTIFICATION OF RESPONSIBILITIES OF CLASSES:

7.1. 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.

7.2. 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.

7.3. 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.

7.4. 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.		

7.5. 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.

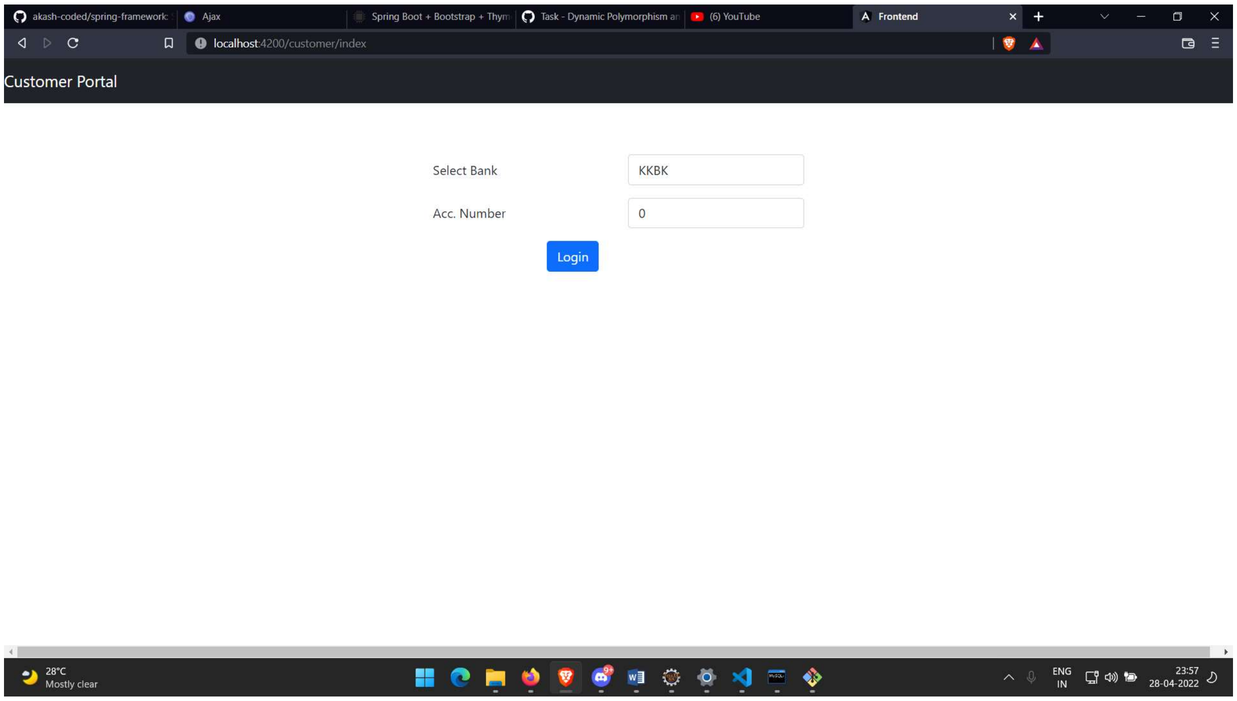
7.6. Services:

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.

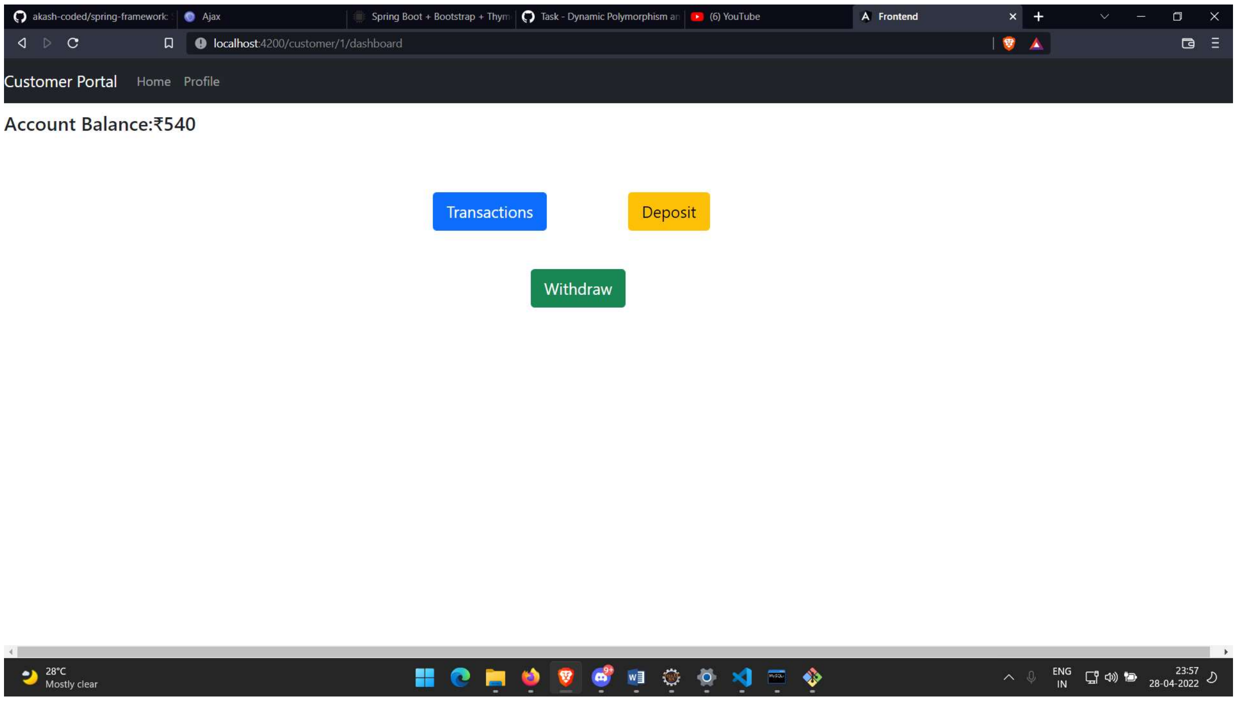
CLASS NAME: TransactionService		
RESPONSIBILITIES	DEPENDABLES	DEPENDENCIES
For depositing, withdrawing and getting all the transactions.		Transaction Model. Transaction Repository.

8. IMPLEMENTATION:

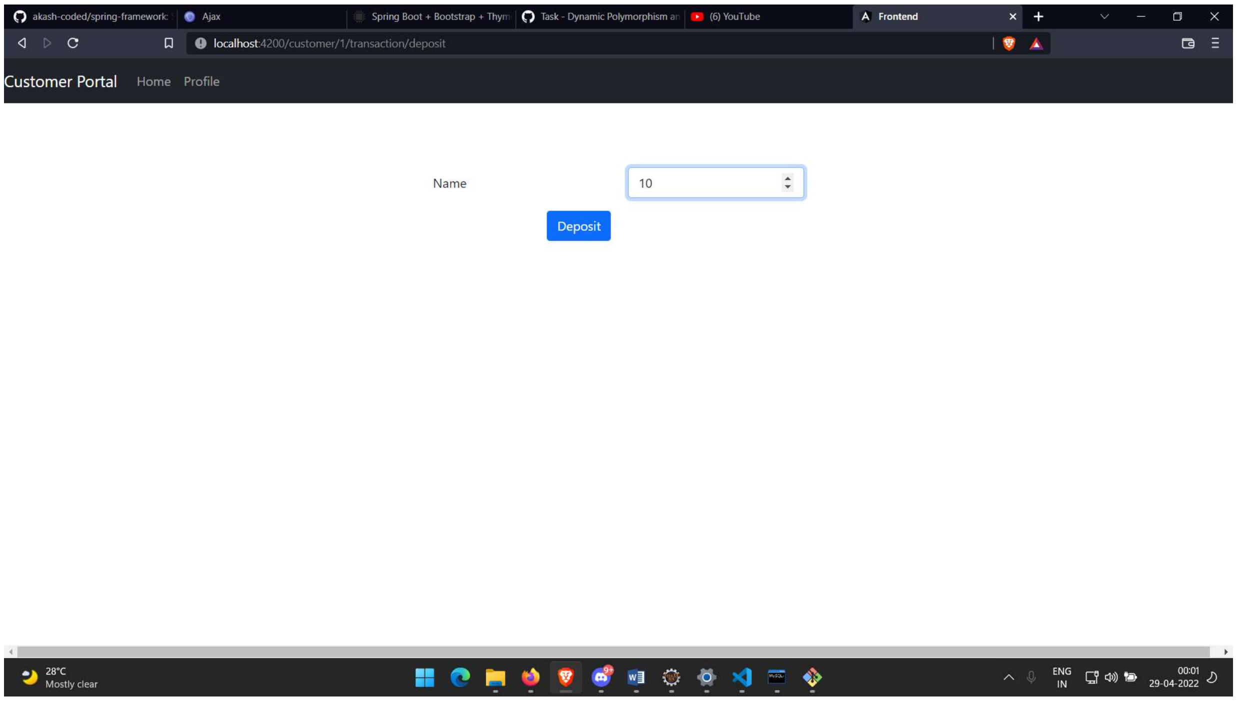

akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/customer/index
Customer Portal
Login
28°C
Mostly clear
ENG
IN
23:57
28-04-2022

Select Bank KKBK

Acc. Number 0


akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/customer/1/dashboard
Customer Portal
Home
Profile
Transactions
Deposit
Withdraw
28°C
Mostly clear
ENG
IN
23:57
28-04-2022

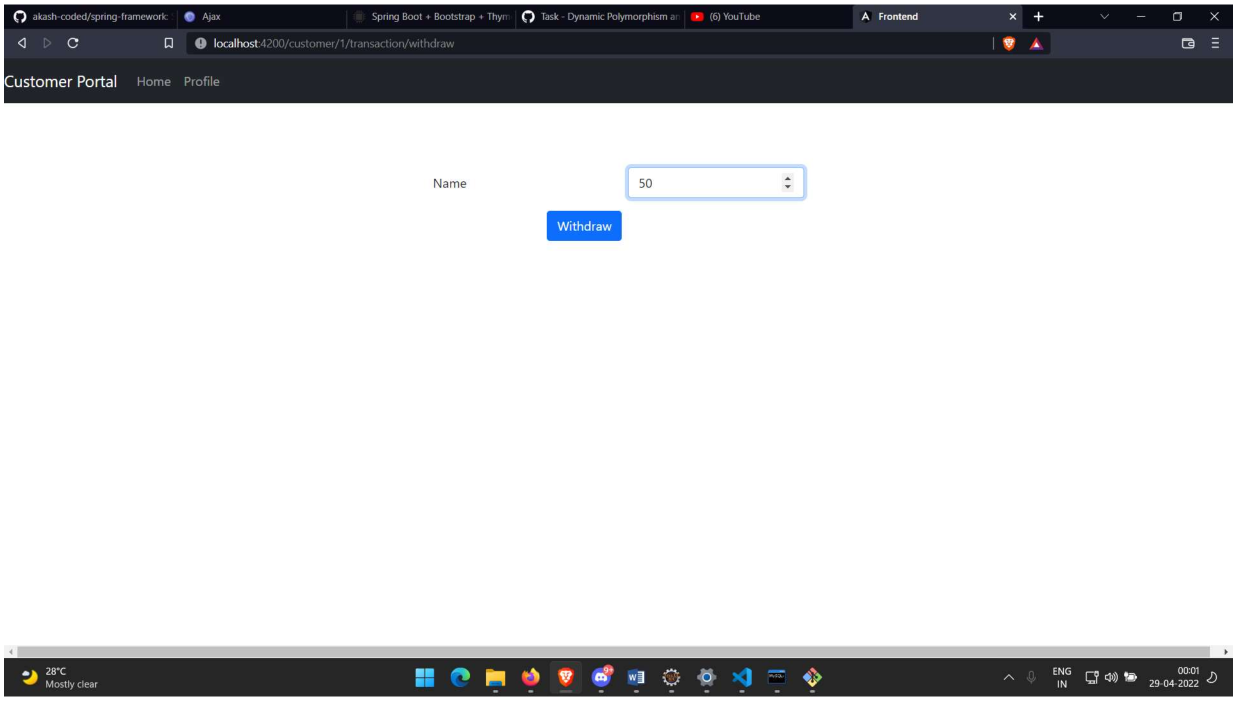
Account Balance:₹540


localhost:4200/customer/1/transaction/deposit

Customer Portal Home Profile

Name: 10

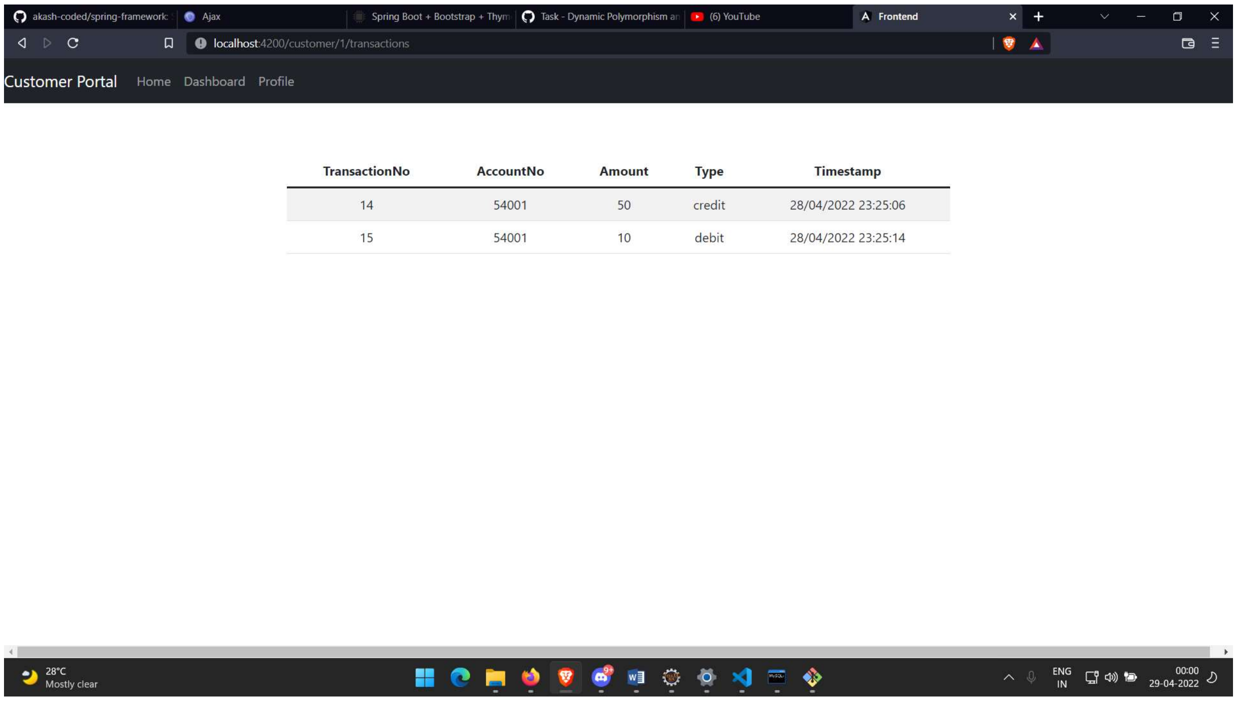
Deposit


localhost:4200/customer/1/transaction/withdraw

Customer Portal Home Profile

Name: 50

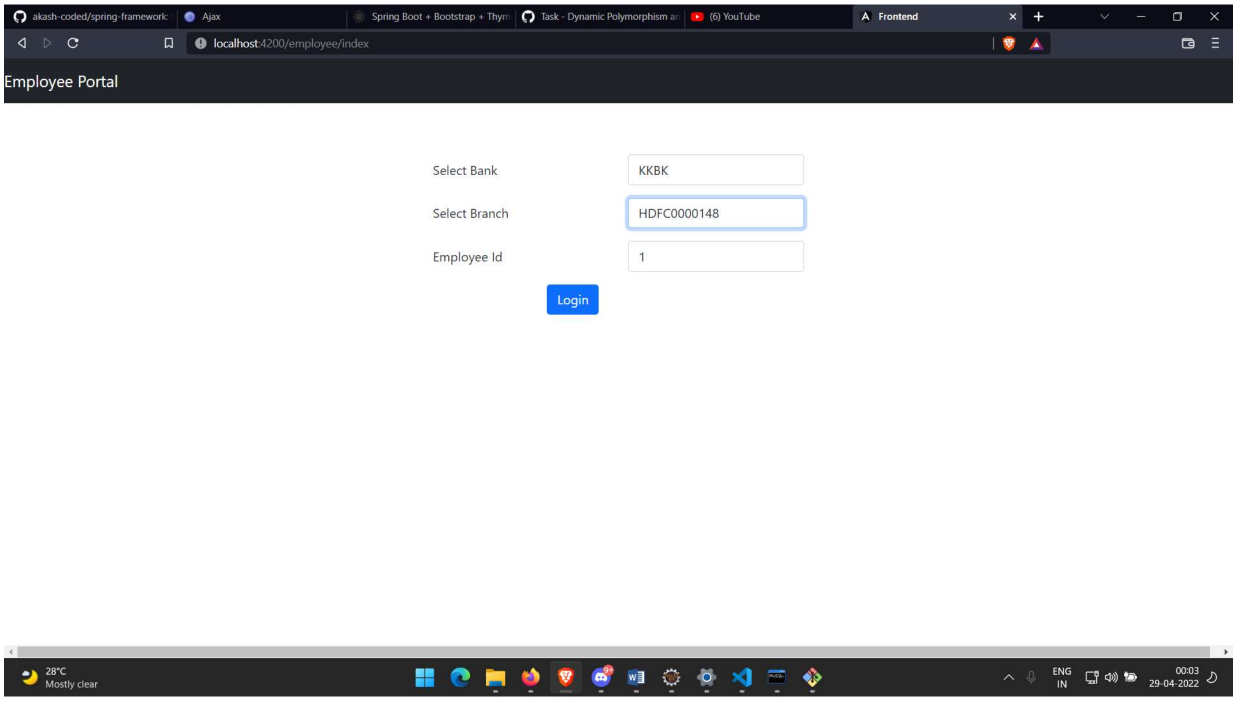
Withdraw



localhost:4200/customer/1/transactions

Customer Portal Home Dashboard Profile

TransactionNo	AccountNo	Amount	Type	Timestamp
14	54001	50	credit	28/04/2022 23:25:06
15	54001	10	debit	28/04/2022 23:25:14

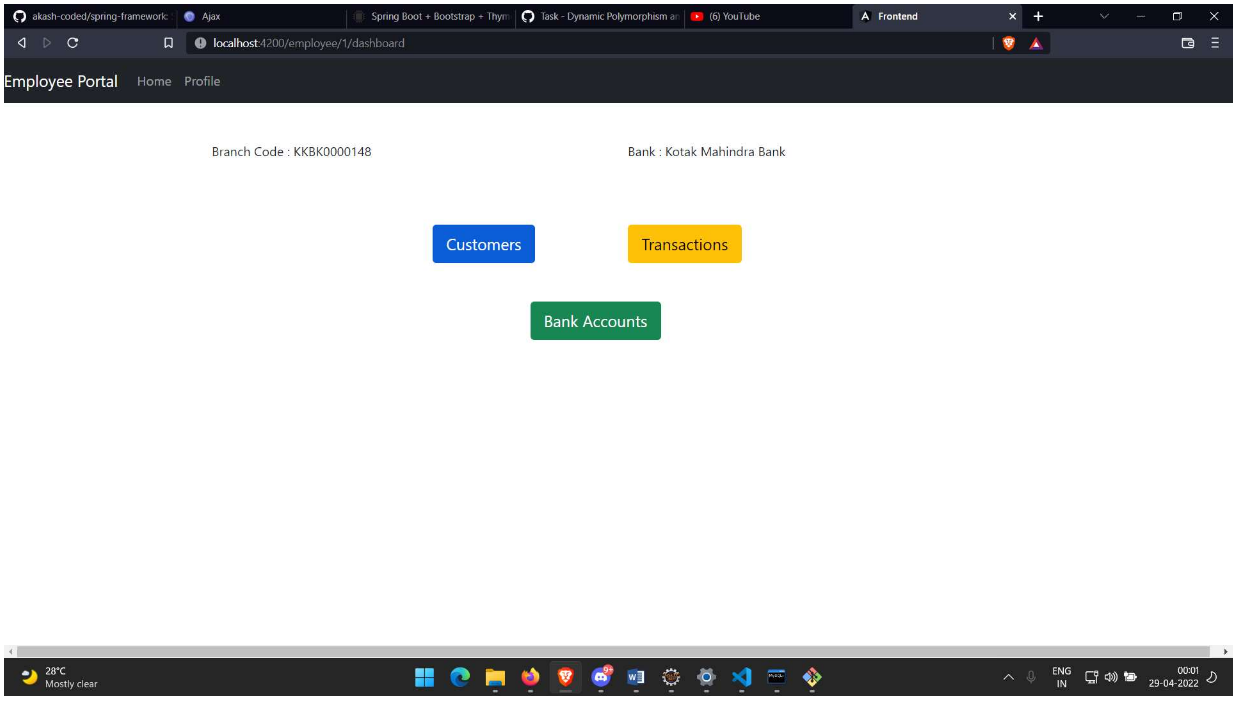


localhost:4200/employee/index

Employee Portal

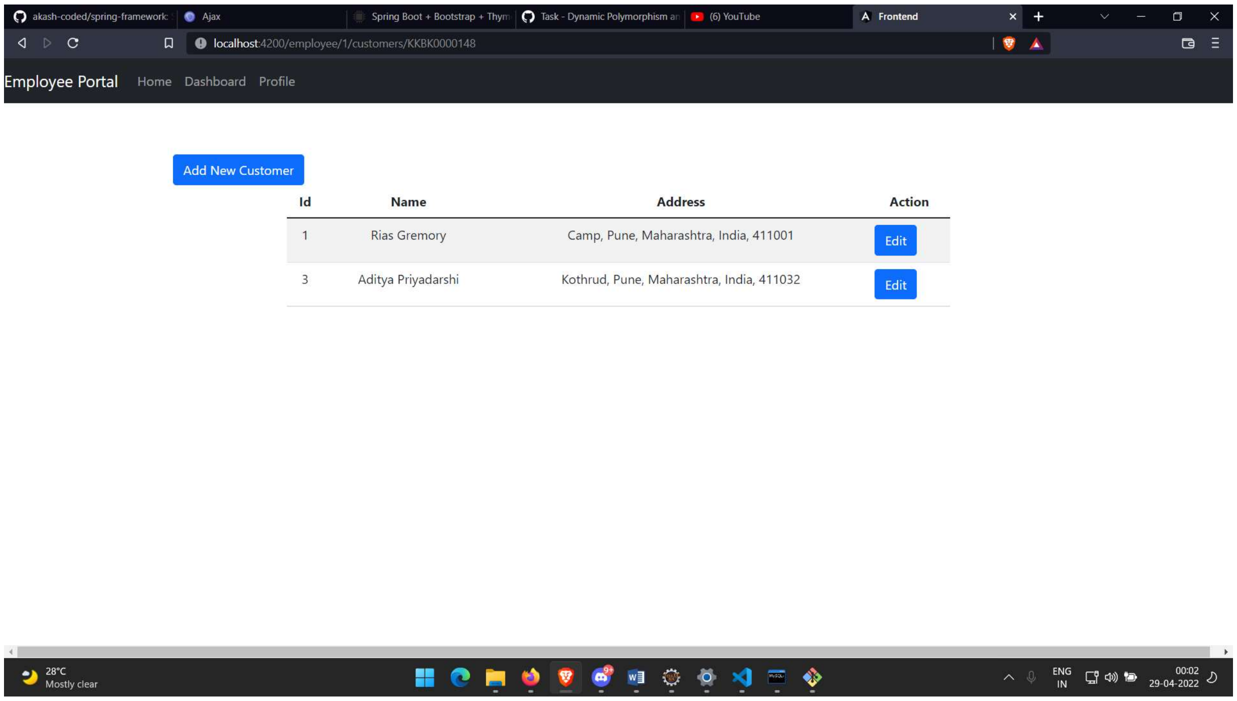
Select Bank	KKBK
Select Branch	HDFC0000148
Employee Id	1

Login

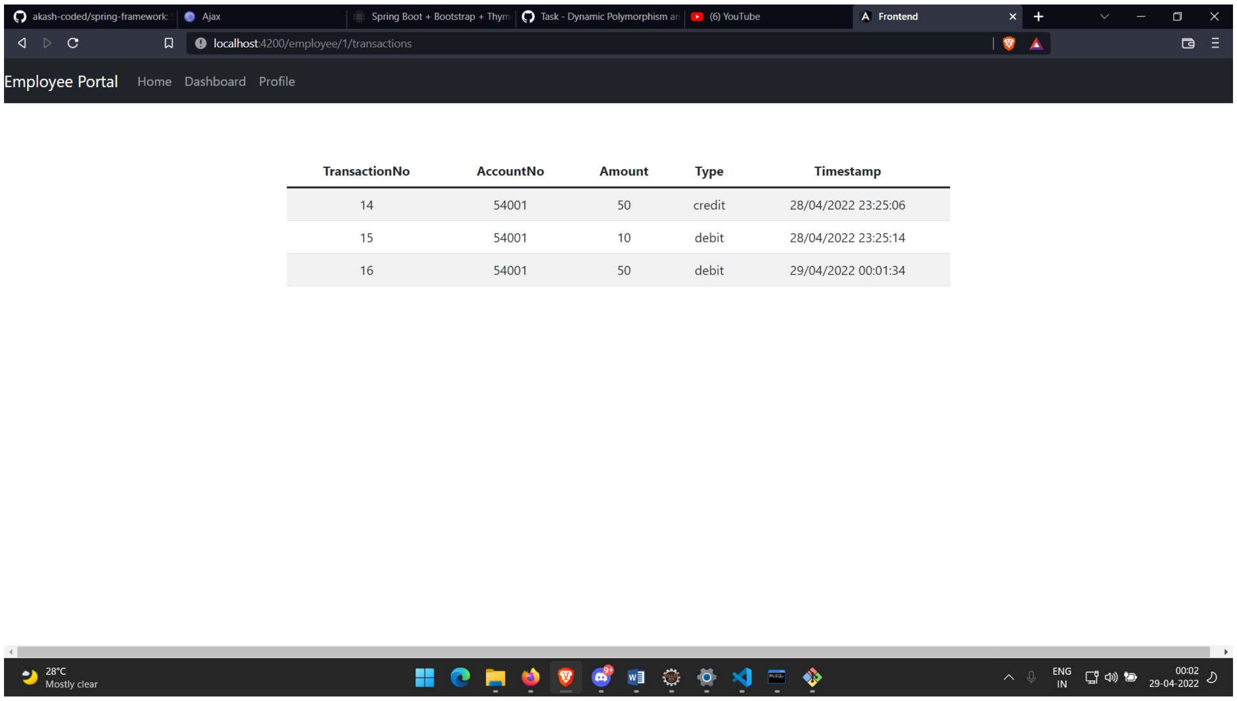

akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/employee/1/dashboard
Employee Portal
Home
Profile
Customers
Transactions
Bank Accounts
28°C
Mostly clear
ENG
IN
00:01
29-04-2022

Branch Code : KKBK0000148

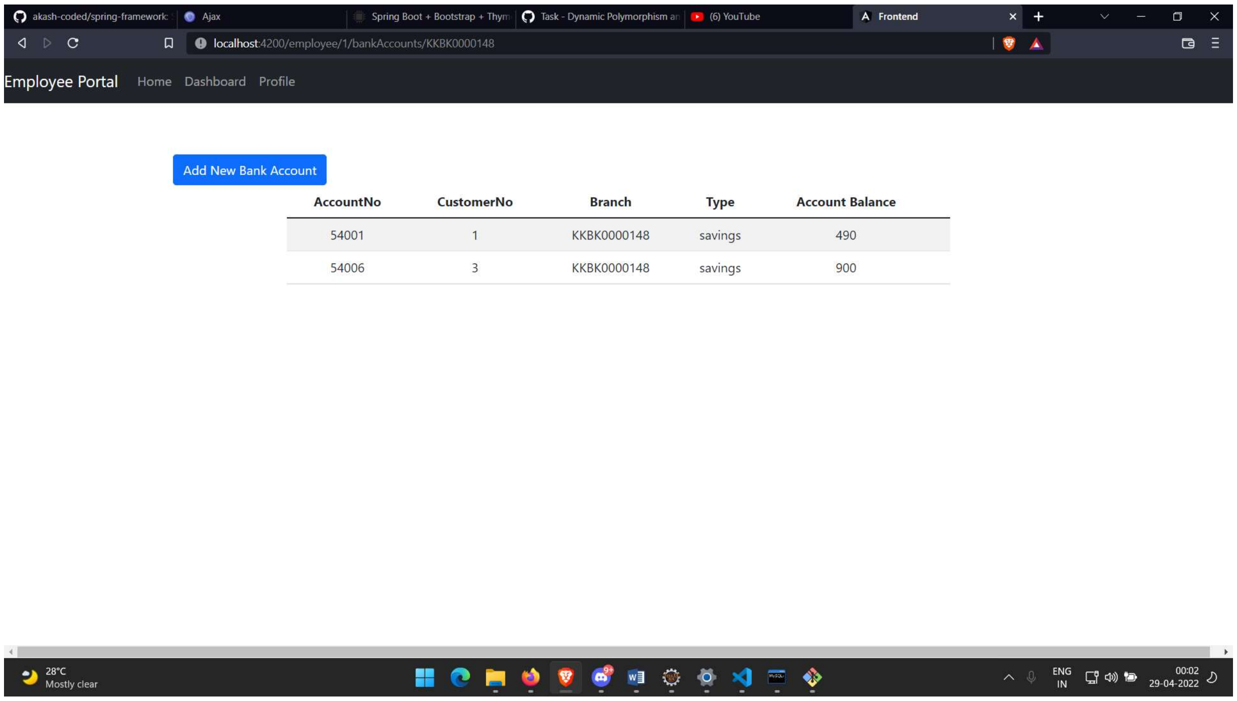
Bank : Kotak Mahindra Bank


akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/employee/1/customers/KKBK0000148
Employee Portal
Home
Dashboard
Profile
Add New Customer
28°C
Mostly clear
ENG
IN
00:02
29-04-2022

Id	Name	Address	Action
1	Rias Gremory	Camp, Pune, Maharashtra, India, 411001	Edit
3	Aditya Priyadarshi	Kothrud, Pune, Maharashtra, India, 411032	Edit

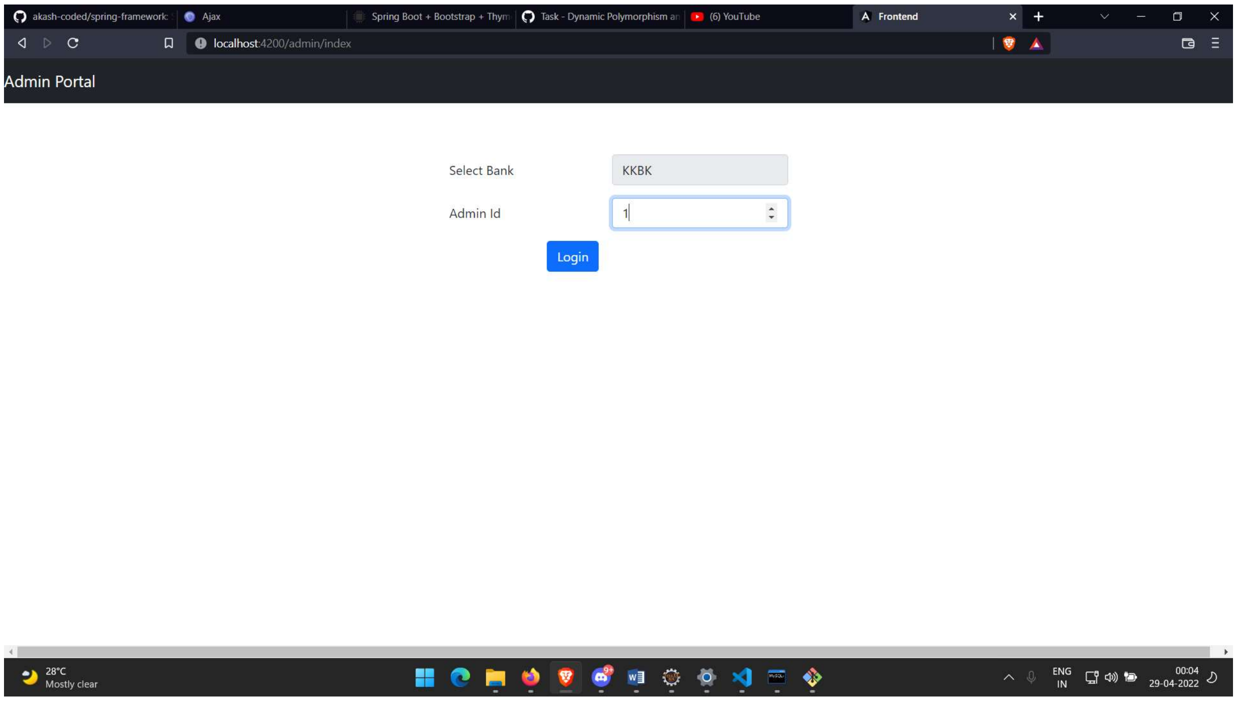

Employee Portal Home Dashboard Profile

TransactionNo	AccountNo	Amount	Type	Timestamp
14	54001	50	credit	28/04/2022 23:25:06
15	54001	10	debit	28/04/2022 23:25:14
16	54001	50	debit	29/04/2022 00:01:34


Employee Portal Home Dashboard Profile

Add New Bank Account

AccountNo	CustomerNo	Branch	Type	Account Balance
54001	1	KKBK0000148	savings	490
54006	3	KKBK0000148	savings	900

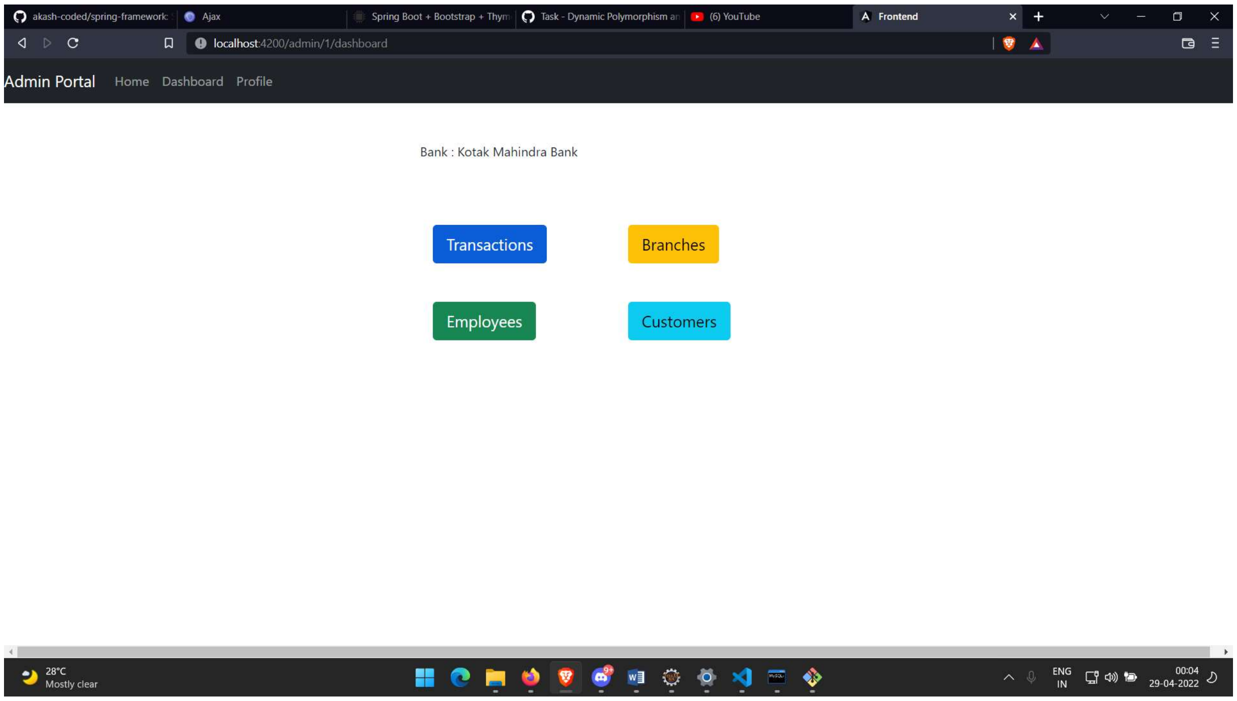

akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/admin/index
Admin Portal

Select Bank KKBK

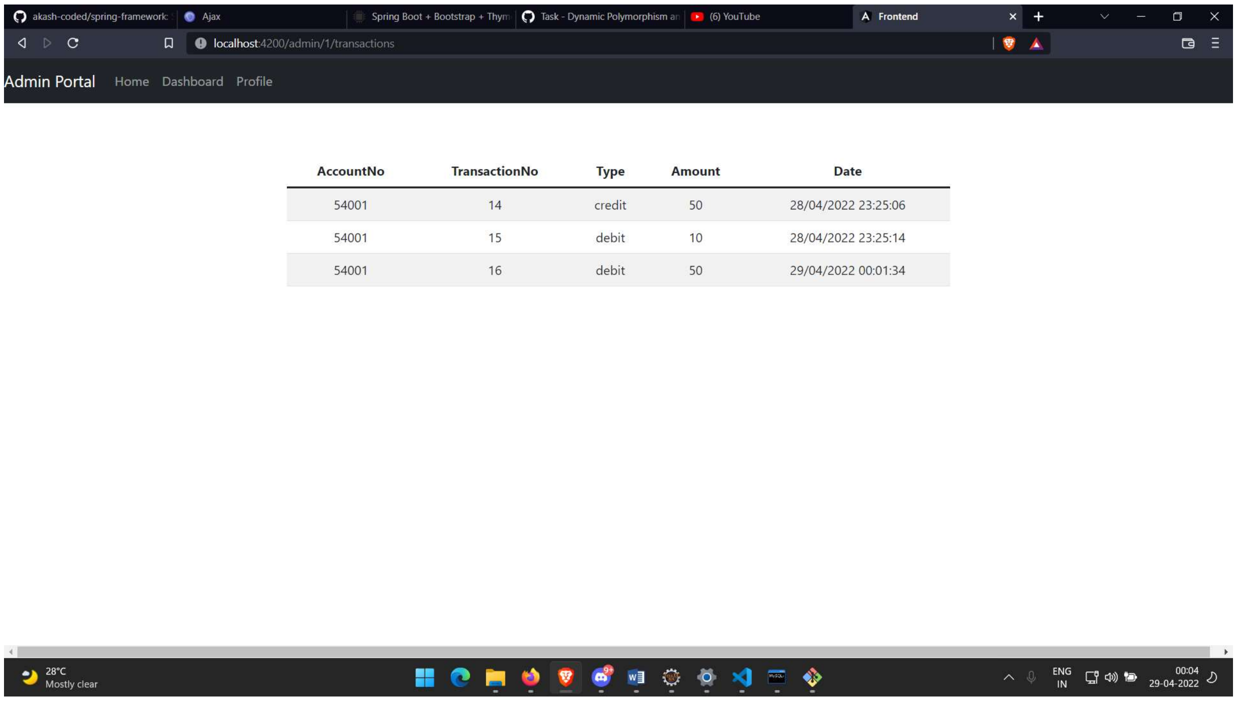
Admin Id 1

Login

28°C
Mostly clear
ENG
IN
00:04
29-04-2022


akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/admin/1/dashboard
Admin Portal
Home
Dashboard
Profile

Bank : Kotak Mahindra Bank



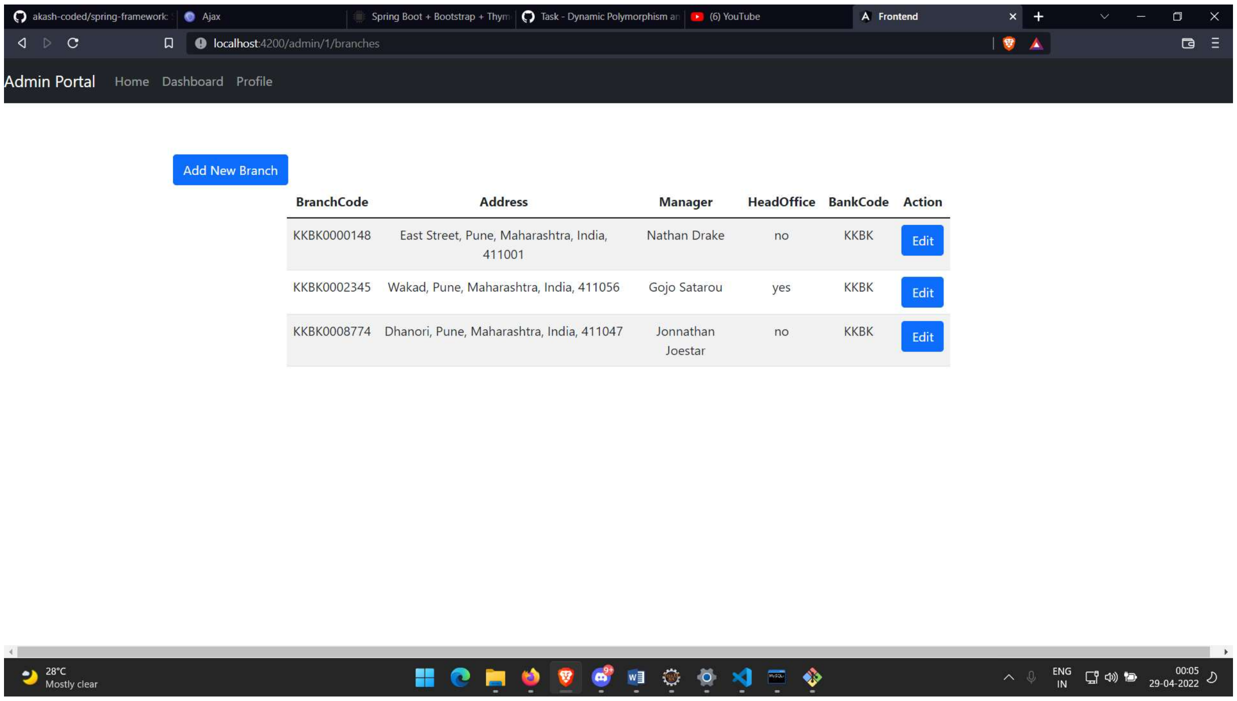
akash-coded/spring-framework | Ajax | Spring Boot + Bootstrap + Thym | Task - Dynamic Polymorphism an | (6) YouTube | Frontend

localhost:4200/admin/1/transactions

Admin Portal Home Dashboard Profile

AccountNo	TransactionNo	Type	Amount	Date
54001	14	credit	50	28/04/2022 23:25:06
54001	15	debit	10	28/04/2022 23:25:14
54001	16	debit	50	29/04/2022 00:01:34

28°C Mostly clear ENG IN 00:04 29-04-2022



akash-coded/spring-framework | Ajax | Spring Boot + Bootstrap + Thym | Task - Dynamic Polymorphism an | (6) YouTube | Frontend

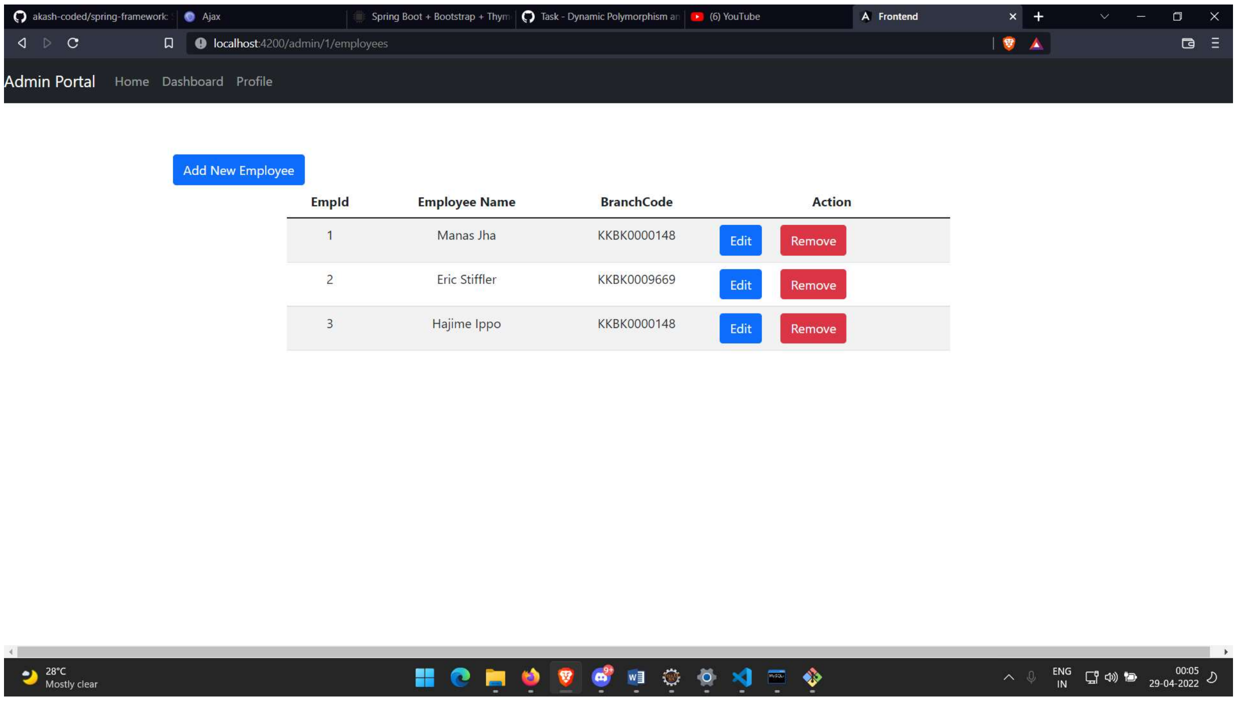
localhost:4200/admin/1/branches

Admin Portal Home Dashboard Profile

Add New Branch

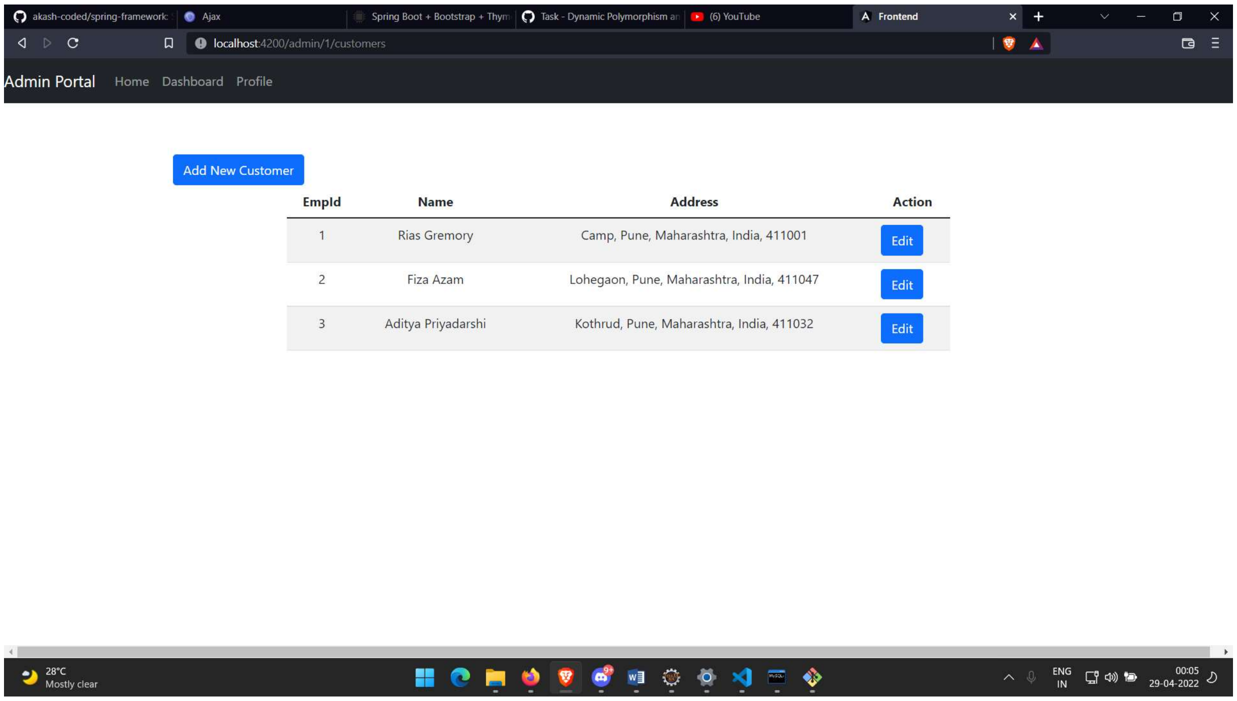
BranchCode	Address	Manager	HeadOffice	BankCode	Action
KKBK0000148	East Street, Pune, Maharashtra, India, 411001	Nathan Drake	no	KKBK	Edit
KKBK0002345	Wakad, Pune, Maharashtra, India, 411056	Gojo Satarou	yes	KKBK	Edit
KKBK0008774	Dhanori, Pune, Maharashtra, India, 411047	Jonnathan Joestar	no	KKBK	Edit

28°C Mostly clear ENG IN 00:05 29-04-2022



akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/admin/1/employees
Admin Portal
Home
Dashboard
Profile
Add New Employee
28°C
Mostly clear
ENG
IN
00:05
29-04-2022

EmpId	Employee Name	BranchCode	Action
1	Manas Jha	KKBK0000148	Edit Remove
2	Eric Stiffler	KKBK0009669	Edit Remove
3	Hajime Ippo	KKBK0000148	Edit Remove



akash-coded/spring-framework
Ajax
Spring Boot + Bootstrap + Thym
Task - Dynamic Polymorphism an
(6) YouTube
Frontend
localhost:4200/admin/1/customers
Admin Portal
Home
Dashboard
Profile
Add New Customer
28°C
Mostly clear
ENG
IN
00:05
29-04-2022

EmpId	Name	Address	Action
1	Rias Gremory	Camp, Pune, Maharashtra, India, 411001	Edit
2	Fiza Azam	Lohegaon, Pune, Maharashtra, India, 411047	Edit
3	Aditya Priyadarshi	Kothrud, Pune, Maharashtra, India, 411032	Edit

9. 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.

10. REFERENCES:

10.1. stackoverflow.com

10.2. github.com/akash-coded