##### Insights into Visible: Analysis and Implementation

**DSN4096-CAPSTONE PROJECT PHASE-II**

###### ***Submitted by***

##### ISHITA VERMA

**(20BCE10395)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

# COMPUTER SCIENCE AND ENGINEERING

****

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH - 466114**

##### MAY 2024

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“Insights into Visible: Analysis and Implementation”** is the bonafide work of “**ISHITA VERMA (20BCE10395)”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

Dr J. Manikandan,, Mr. Venkata Avinash Sainni

Senior Assistant Professor (Gr-2) Associate Director

School of Computing Science and Engineering Software Development

VIT BHOPAL UNIVERSITY VERIZON

The Internship Examination is held on May 2,2024.

**ACKNOWLEDGEMENT**

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I would like to express my sincere gratitude to Verizon for offering me the opportunity to work as an SDE intern.

I would like to thank my supervisor Mr. Venkata Avinash Sainni, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I wish to express my heartfelt gratitude to Dr. J. Manikandan, Head of the Department, School of Computing Science and Engineering for much of his valuable support and encouragement in carrying out this work.

I would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**LIST OF ABBREVIATIONS**

1. SDE Software Development Engineer
2. VZ Verizon
3. CSS Cascading Style Sheet
4. DB Database
5. UI User Interface
6. UX User Experience
7. CRUD Create, Read, Update, Delete
8. CI/CD Continuous Integration/Continuous Deployment
9. API Application Programming Interface
10. JS JavaScript
11. REST Representational State Transfer
12. JWT JSON Web Token
13. SQL Structured Query Language
14. JSON JavaScript Object Notation

**LIST OF FIGURES AND GRAPHS**

| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

**ABSTRACT**

Embarking on an enriching journey as a Software Development Engineering (SDE) intern at Verizon (VZ), I have had the privilege of immersing myself in a stimulating environment of learning and professional development. The internship commenced with a structured curriculum encompassing foundational and technical training sessions, meticulously designed to equip interns with the requisite knowledge and skills to excel in their roles.

Following the initial training phase, interns were grouped into teams and assigned diverse projects aligned with Verizon's technological ecosystem. These projects provided a hands-on opportunity to apply theoretical concepts to real-world scenarios, fostering collaboration, innovation, and problem-solving within a supportive team environment.

As an SDE intern, I have actively engaged in project development, leveraging my technical acumen to contribute meaningfully to the team's objectives. Through continuous iteration and feedback cycles, I have honed my coding abilities, expanded my proficiency in various technology stacks, and cultivated a deeper understanding of software development best practices.

Moreover, the internship experience extends beyond technical aspects, encompassing invaluable opportunities for personal and professional growth. Engaging in cross-functional collaboration, participating in mentorship programs, and attending networking events have facilitated the development of essential soft skills such as communication, teamwork, and adaptability.

As the internship progresses, the transition into functional training further enriches the learning experience, providing insights into Verizon's operational framework and strategic objectives. This comprehensive approach to training underscores Verizon's commitment to nurturing talent and fostering a culture of continuous learning and development.

In this report, I reflect on the myriad experiences, challenges, and triumphs encountered during my internship at Verizon. Through a structured narrative, I delve into the projects undertaken, technical competencies acquired, and the broader lessons gleaned from this immersive experience. As I continue to immerse myself in the vibrant ecosystem of Verizon, I am grateful for the opportunities bestowed upon me and eagerly anticipate the contributions I can make to the organization and beyond.

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | List of Abbreviations  List of Figures and Graphs  Abstract | 5  6  7 |
| 1 | **CHAPTER-1:**  **PROJECT DESCRIPTION AND OUTLINE** Introduction 1.2 Motivation for the work  1.3 About Introduction to the project including techniques  1.5 Problem Statement  1.6 Objective of the work  1.7 Organization of the project  1.8 Summary | 1  .  .  . |
| 2 | **CHAPTER-2:**  **RELATED WORK INVESTIGATION**  2.1 Introduction  2.2 Core area of the project  2.3 Existing Approaches/Methods  2.3.1 Approaches/Methods -1  2.3.2 Approaches/Methods -2  2.3.3 Approaches/Methods -3  2.4 Pros and cons of the stated Approaches/Methods  2.5 Issues/observations from investigation  2.6 Summary |  |
| 3 | **CHAPTER-3:**  **REQUIREMENT ARTIFACTS**  3.1 Introduction  3.2 Hardware and Software requirements  3.3 Specific Project requirements  3.3.1 Data requirement  3.3.2 Functions requirement  3.3.3 Performance and security requirement  3.3.4 Look and Feel Requirements  3.3.5 ………  3.4 Summary |  |
| 4 | **CHAPTER-4:**  **DESIGN METHODOLOGY AND ITS NOVELTY**  4.1 Methodology and goal  4.2 Functional modules design and analysis  4.3 Software Architectural designs  4.4 Subsystem services  4.5 User Interface designs  4.6 Summary |  |
| 5 | **CHAPTER-5:**  **TECHNICAL IMPLEMENTATION & ANALYSIS**  5.1Outline  5.2 Technical coding and code solutions  5.3 Working Layout of Forms  5.4 Prototype submission  5.5 Test and validation  5.6 Performance Analysis(Graphs/Charts)  5.7 Summary |  |
| 6 | **CHAPTER-6:**  **PROJECT OUTCOME AND APPLICABILITY**  6.1Outline  6.2 key implementations outlines of the System  6.3 Significant project outcomes  6.4 Project applicability on Real-world applications  6.4 Inference |  |
| 7 | **CHAPTER-7:**  **CONCLUSIONS AND RECOMMENDATION**  7.1Outline  7.2 Limitation/Constraints of the System  7.3 Future Enhancements  7.4 Inference |  |
|  | Appendix A  Appendix B  References  ***Note: List of References should be written as per IEEE/Springer reference format. (Specimen attached)*** |  |

**CHAPTER-1:**

**PROJECT DESCRIPTION AND OUTLINE**

# 1.1 Introduction

# 

# my tech stack : React, APIGEE, Java, SQL, Spring boot Based Microservices

Components/Services

User Login

User Authentication

Home page

User Profile

Transaction History

Transactions

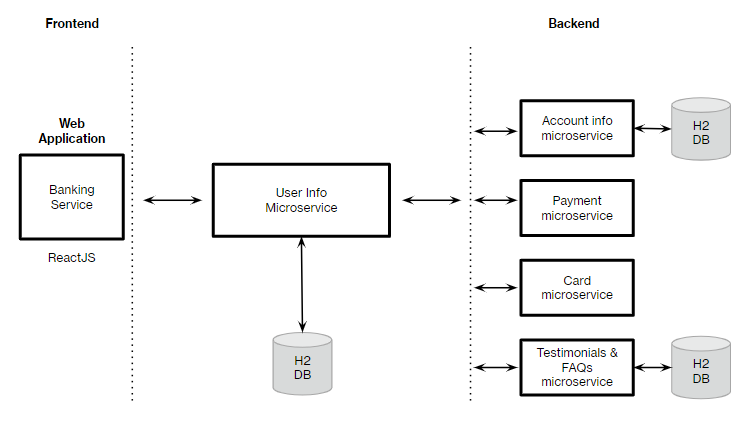
Testimonials

FAQs

Cards

workflow:

1. User logs into the banking platform using their credentials.
2. Upon successful login,the user is presented with their account dashboard.
3. User can navigate to specific services they need such as transactions, cards application or view testimonials.
4. User can view their account details in Profile section.
5. In the Transactions section, the user initiates fund transfer or payments by providing recipient details and transaction amounts.
6. In the bank card section, the user can add their card details.
7. The user logs out of the platform after completing their tasks.



User Validation using JWT Tokens :

JWT Token Generation

a. Login : Users authenticate by providing their credentials .

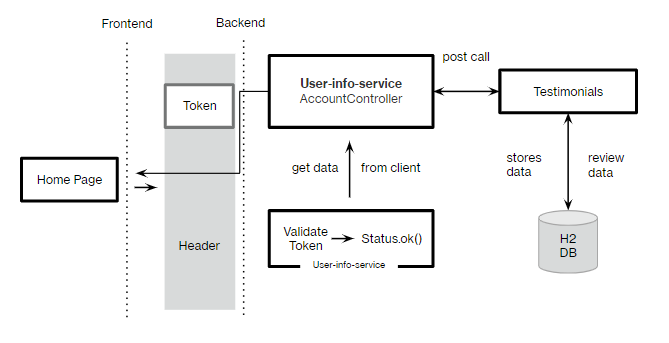
b. Token Generation : Upon successful authentication , the server generates a JWT token.

c. Token Storage: Users stores the JWT token in local storage .

Token Validation

a. Authorizaton Header : Users must include the token in the Authorization header of their API requests.

b. Token Expiry: Tokens have an expiration time to ensure security.



# 

1.2 Motivation for the work

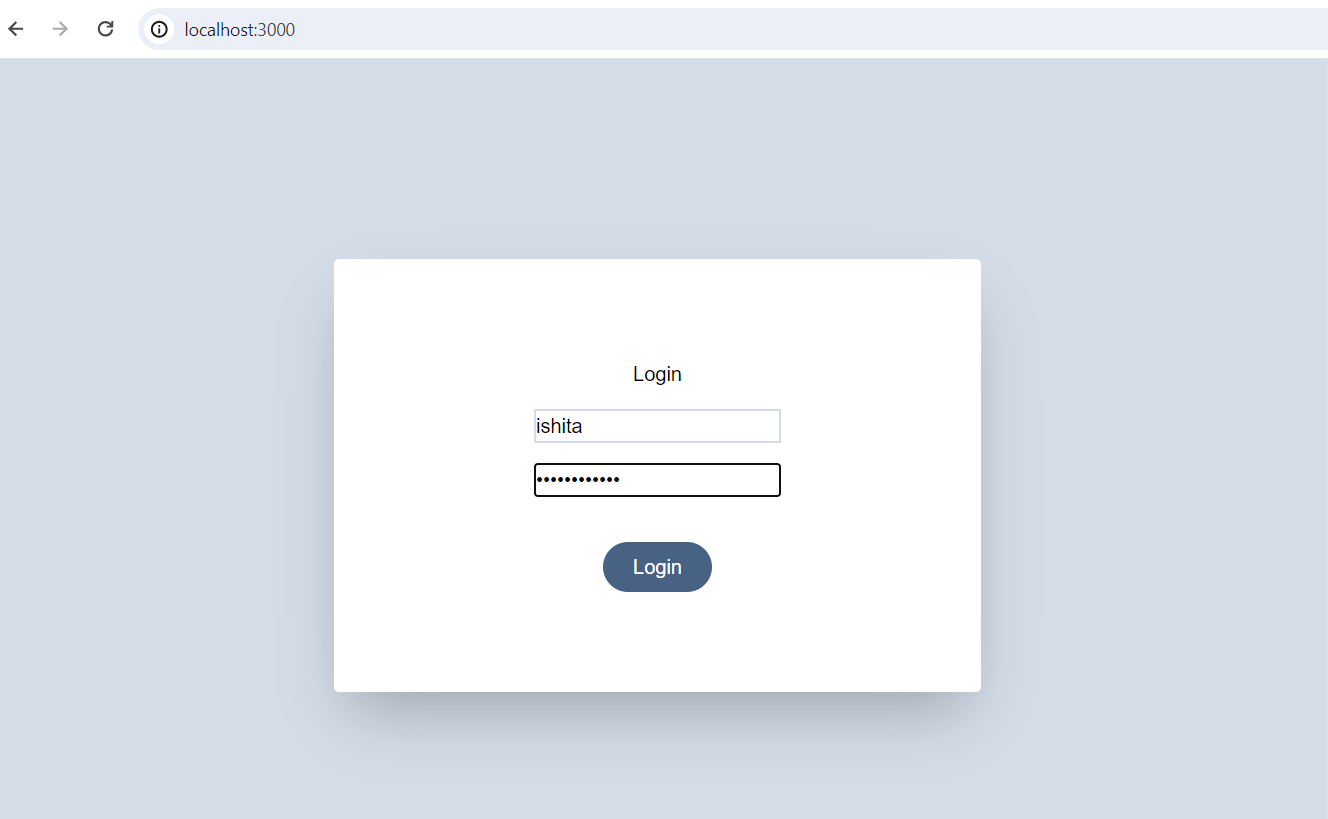
1.3 Introduction to the project including techniques

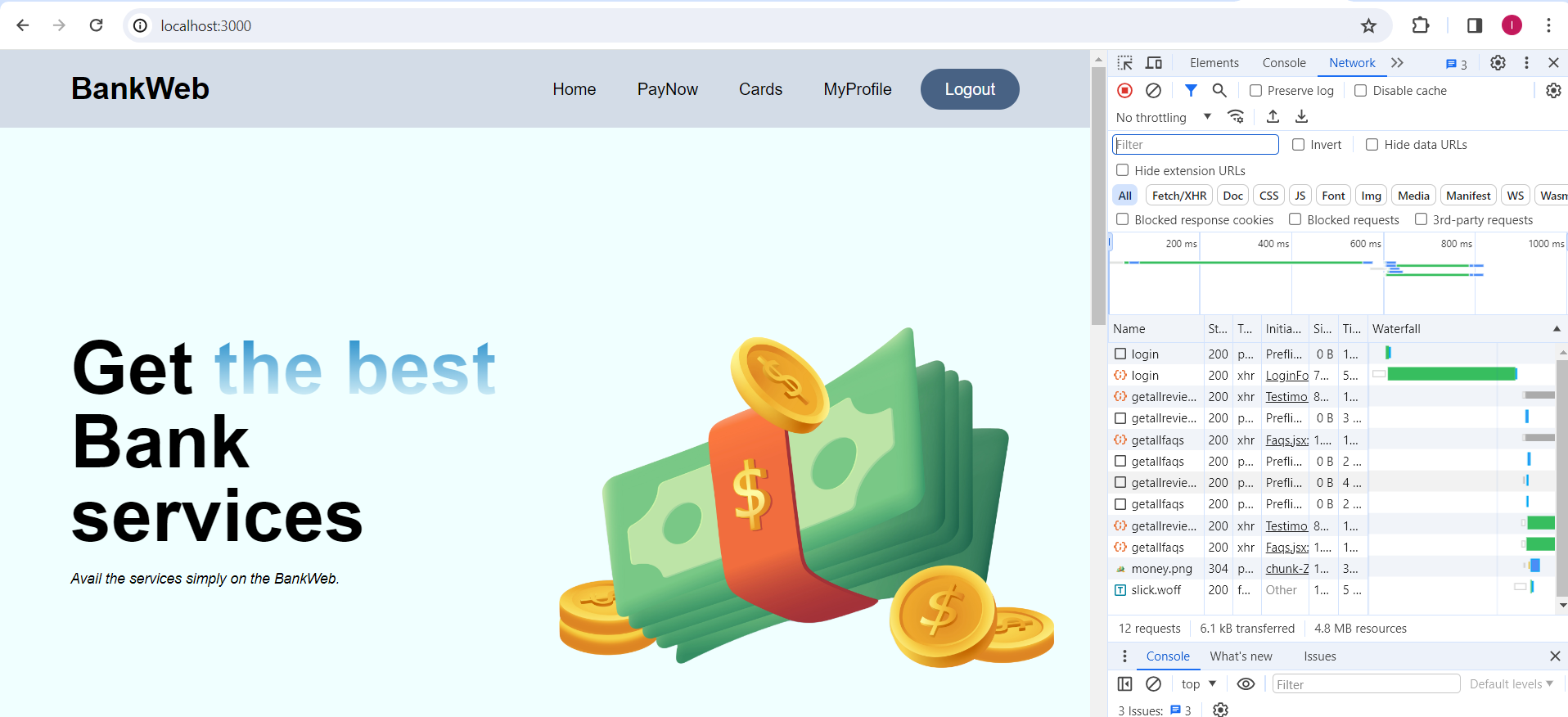
1.5 Problem Statement

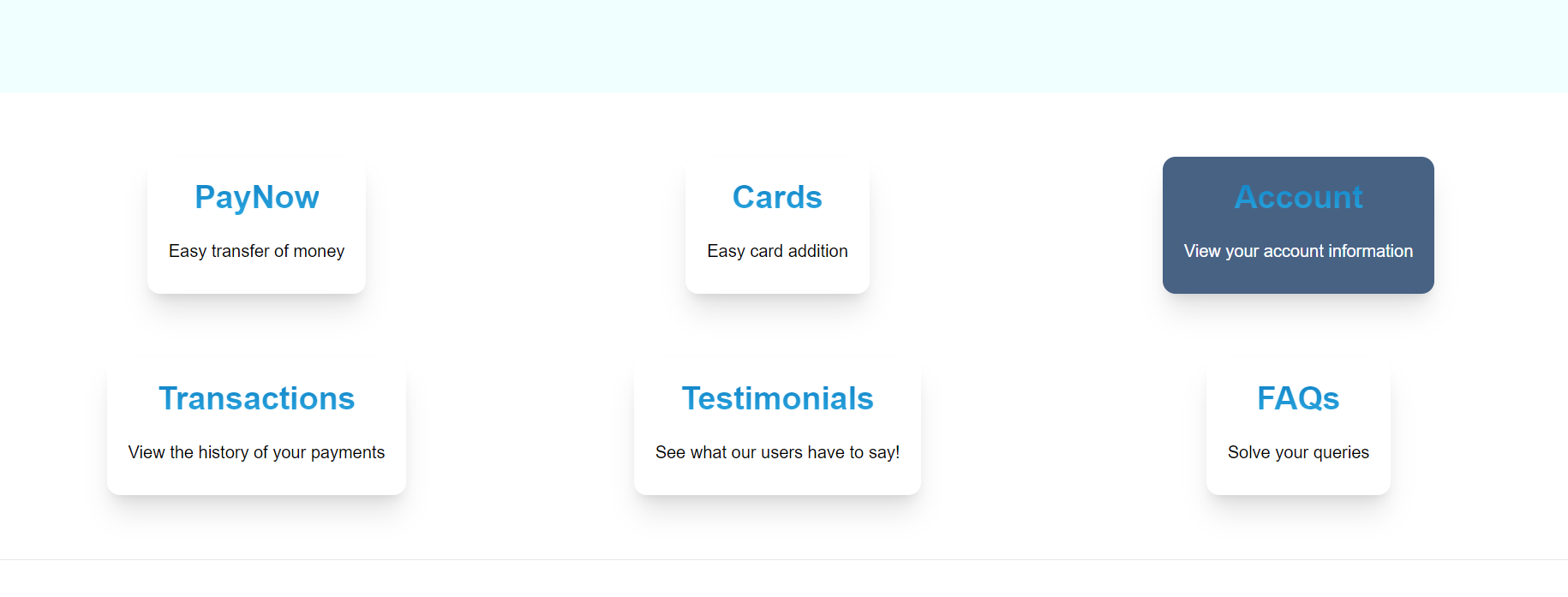
1.6 Objective of the work

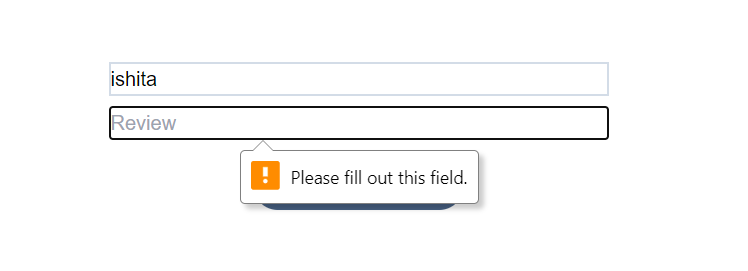
1.7 Organization of the project

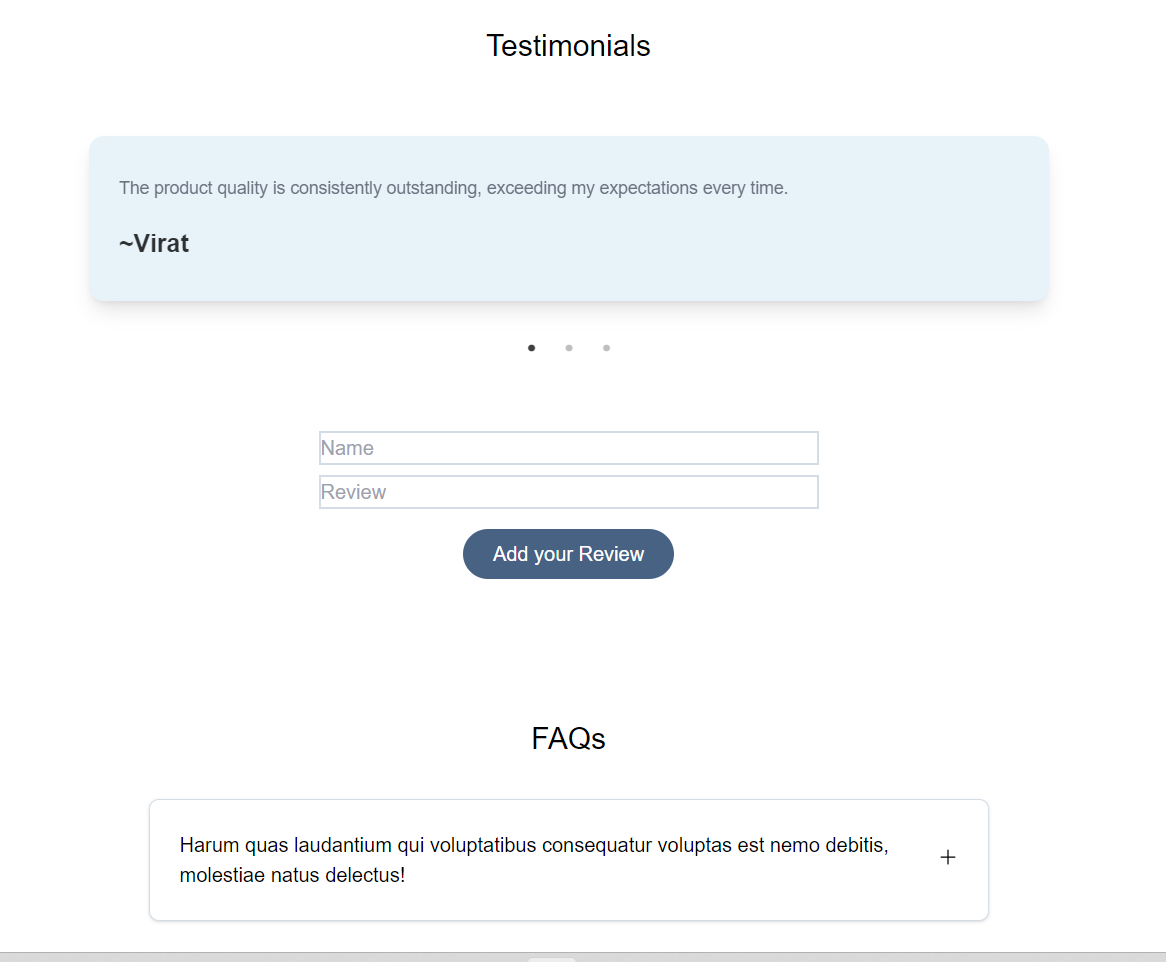
1.8 Summary

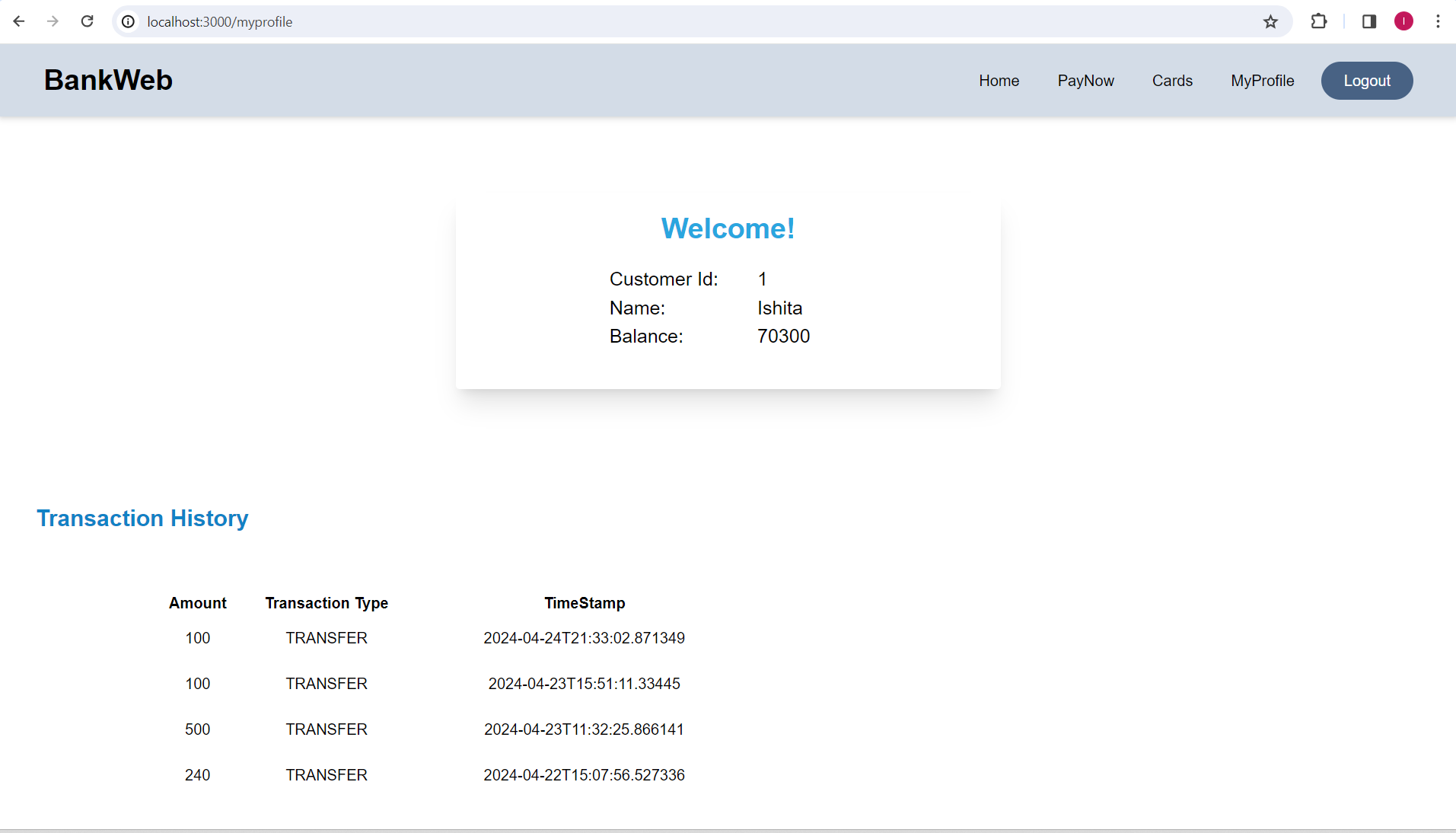


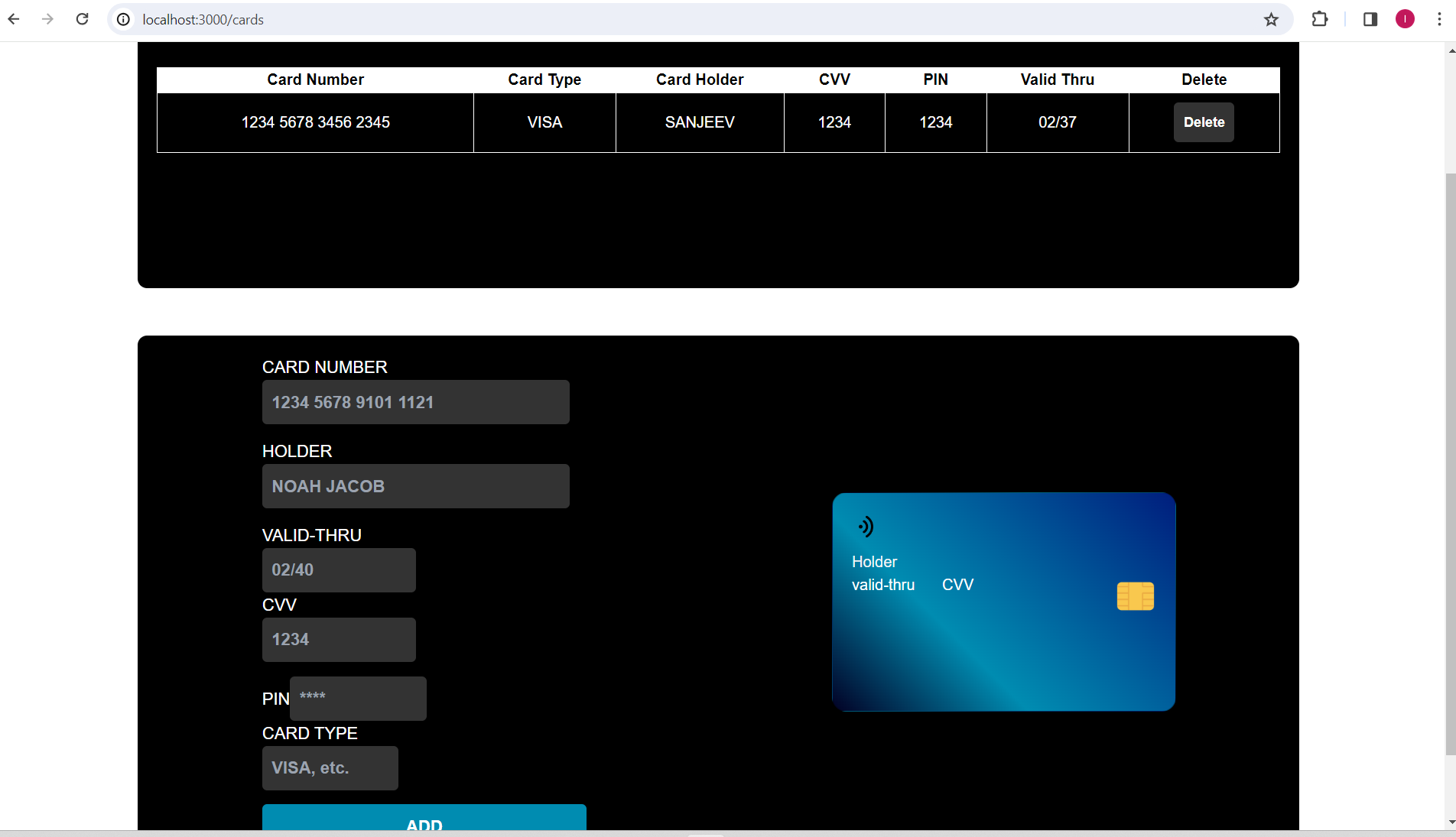


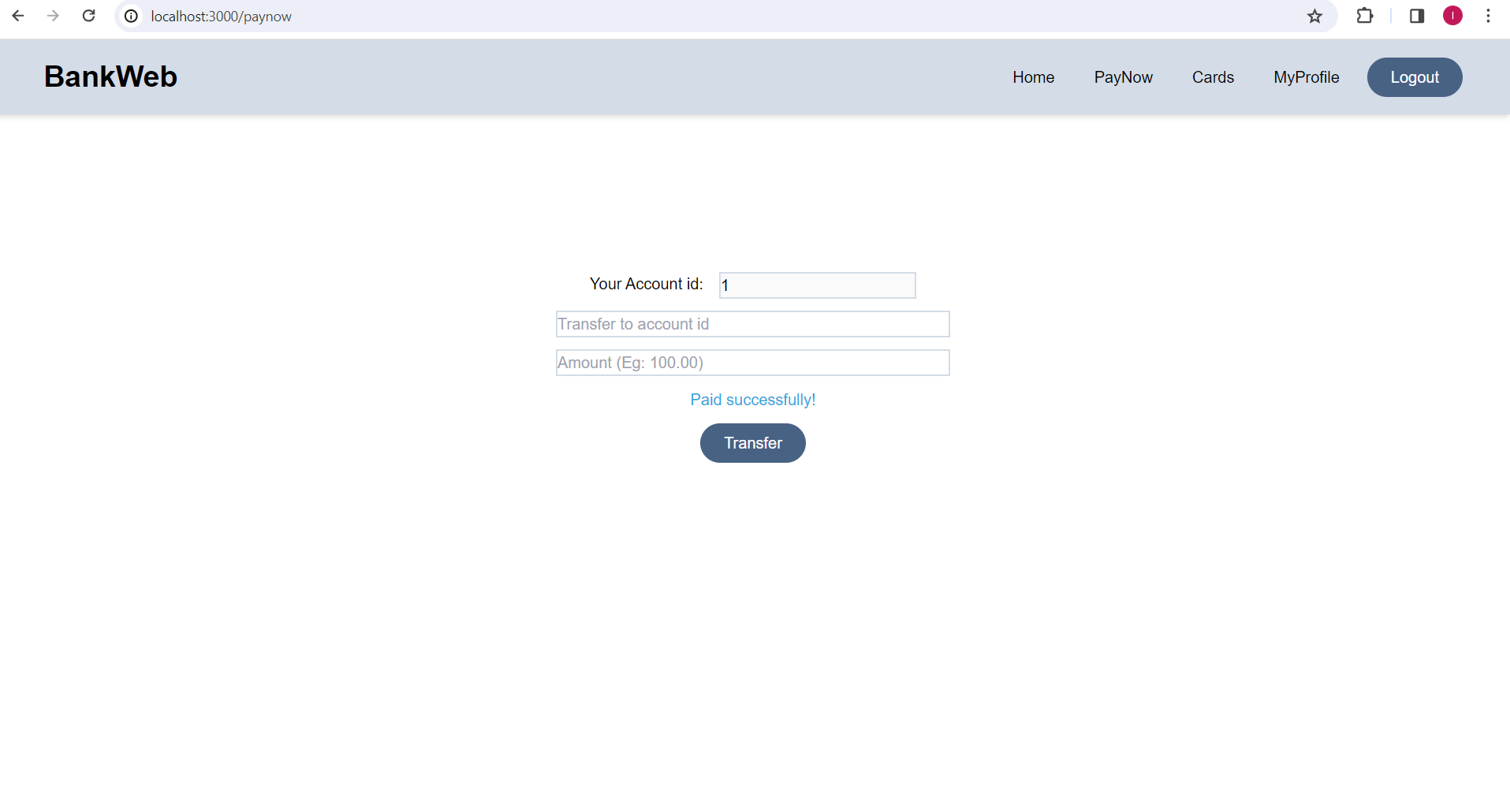


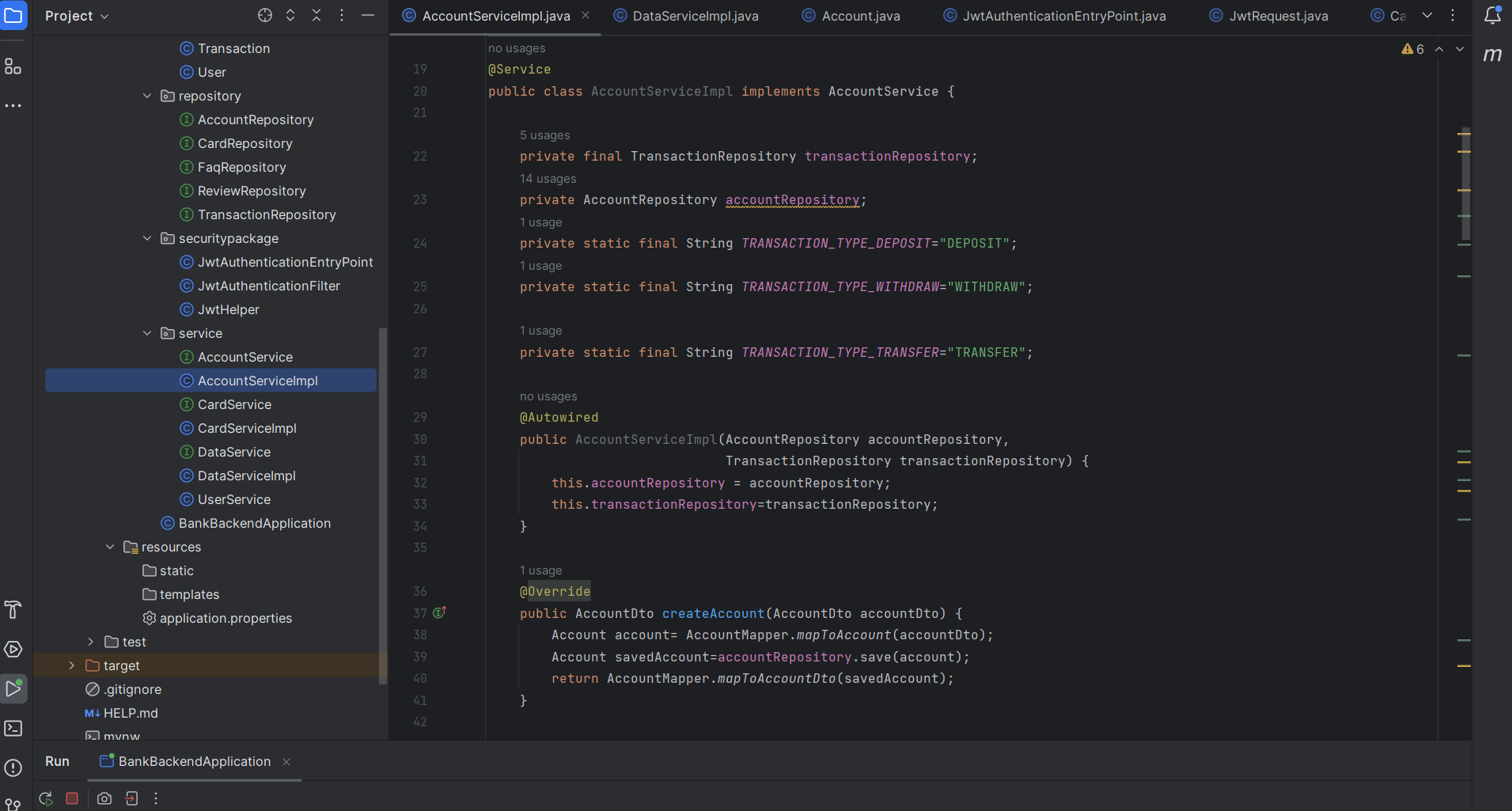












@Override

public void transferFunds(TransferFundDto transferFundDto) {

Account fromAccount = accountRepository

.findById(transferFundDto.fromAccountId())

.orElseThrow(() -> new AccountException("Account does not exists"));

// Retrieve the account to which we send the amount

Account toAccount = accountRepository.findById(transferFundDto.toAccountId())

.orElseThrow(() -> new AccountException("Account does not exists"));

if(fromAccount.getBalance()<transferFundDto.amount()){

throw new RuntimeException("Insufficient Amount");

}

// Debit the amount from fromAccount object

fromAccount.setBalance(fromAccount.getBalance() - transferFundDto.amount());

// Credit the amount to toAccount object

toAccount.setBalance(toAccount.getBalance() + transferFundDto.amount());

accountRepository.save(fromAccount);

accountRepository.save(toAccount);

Transaction transaction=new Transaction();

transaction.setAccountId(transferFundDto.fromAccountId());

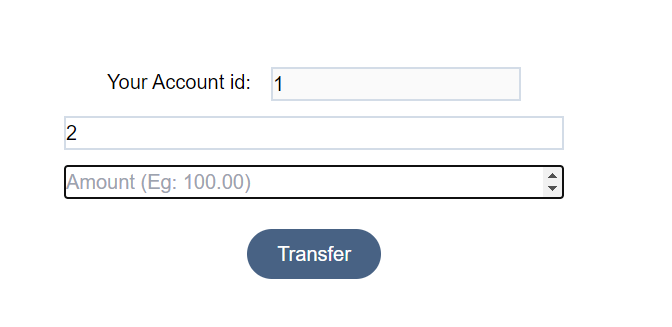
transaction.setAmount(transferFundDto.amount());

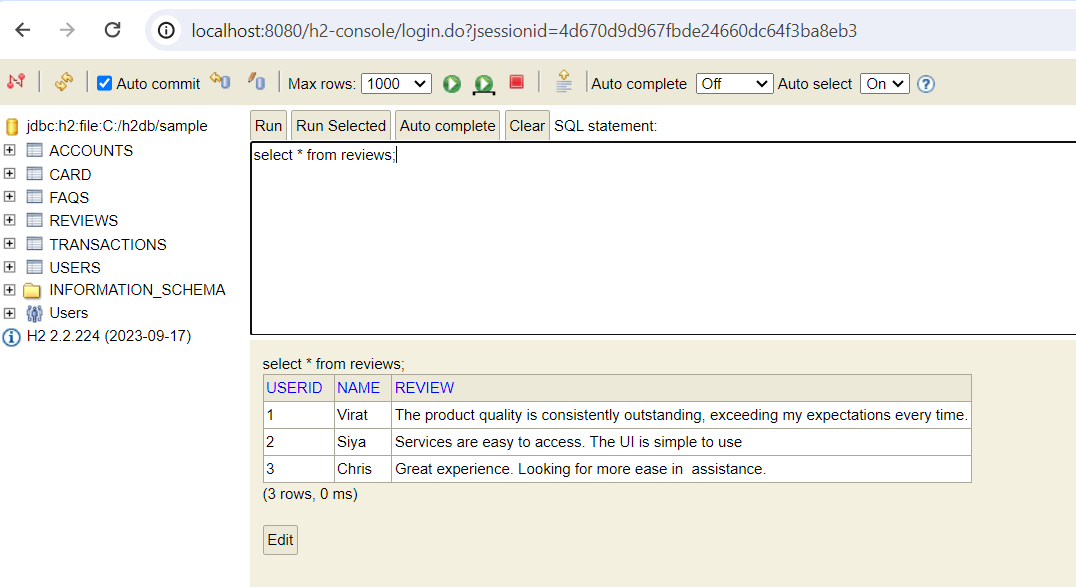
transaction.setTransactionType(*TRANSACTION\_TYPE\_TRANSFER*);

transaction.setTimestamp(LocalDateTime.*now*());

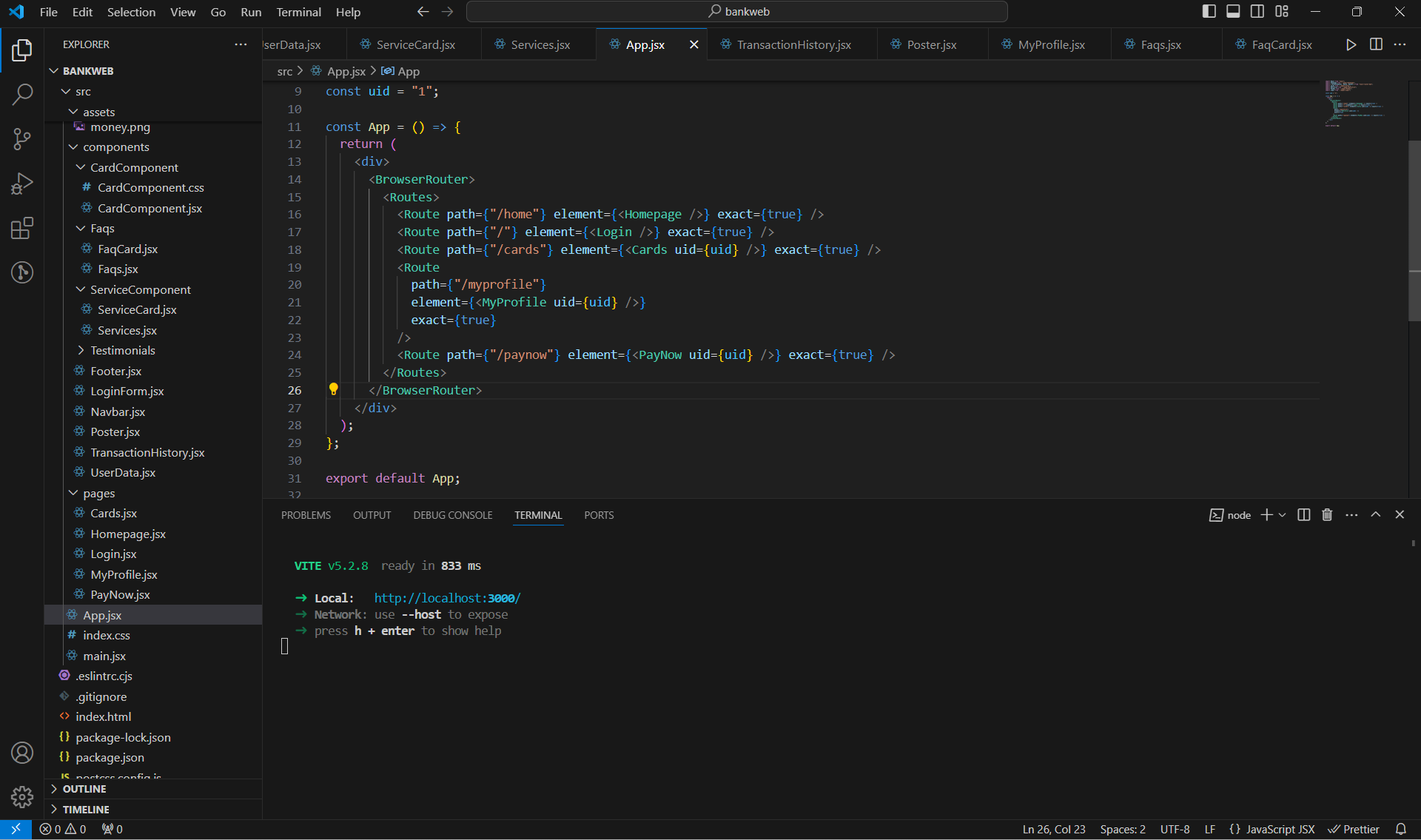
transactionRepository.save(transaction);

}





App.js



const App = () => {

return (

<div>

<BrowserRouter>

<Routes>

<Route path={"/home"} element={<Homepage />} exact={true} />

<Route path={"/"} element={<Login />} exact={true} />

<Route path={"/cards"} element={<Cards uid={uid} />} exact={true} />

<Route

path={"/myprofile"}

element={<MyProfile uid={uid} />}

exact={true}

/>

<Route path={"/paynow"} element={<PayNow uid={uid} />} exact={true} />

</Routes>

</BrowserRouter>

</div>

);

};

export default App;

LoginForm.jsx

const handleLogin = async () => {

try {

const response = await axios.post(

"http://localhost:8080/api/auth/login",

{ username, password }

);

const { jwtToken } = response.data;

localStorage.setItem("jwtToken", jwtToken);

setLoggedIn(true); // Set logged in state to true

} catch (error) {

// Handle login error

console.error("Login failed:", error);

}

};

paynow.jsx

import React, { useState } from "react";

import axios from "axios";

import Navbar from "../components/Navbar";

const PayNow = ({ uid }) => {

const [user, setUser] = useState({

fromAccountId: uid,

toAccountId: "",

amount: "",

});

const [result, setResult] = useState("");

const handleChange = (event) => {

let value = event.target.value;

let name = event.target.name;

setUser((prevUser) => ({ ...prevUser, [name]: value }));

};

const handleSubmit = async (event) => {

event.preventDefault();

try {

const token = localStorage.getItem("jwtToken");

const response = await axios.post(

"http://localhost:8080/api/accounts/transfer",

user,

{

headers: {

Authorization: `Bearer ${token}`,

},

}

);

setUser({

fromAccountId: uid,

toAccountId: "",

amount: "",

});

setResult("Paid successfully!");

console.log("Transaction done:", response.data);

} catch (error) {

console.log("Error:", error);

}

};

return (

<>

<Navbar />

<form

onSubmit={handleSubmit}

className="items-center flex flex-col gap-3 py-40"

>

<label className="flex flex-row gap-4">

<p>Your Account id: </p>

<input

className="border-2 border-solid border-navbg min-w-[150px] md:w-[200px]"

name="fromAccountId"

value={user.fromAccountId}

placeholder="Transfer from account id"

disabled

/>

</label>

<input

className="border-2 border-solid border-navbg md:w-[400px] min-w-[300px]"

type="number"

step="1"

name="toAccountId"

value={user.toAccountId}

onChange={handleChange}

placeholder="Transfer to account id"

/>

<input

className="border-2 border-solid border-navbg md:w-[400px] min-w-[300px]"

type="number"

step="10.00"

name="amount"

value={user.amount}

onChange={handleChange}

placeholder="Amount (Eg: 100.00)"

/>

<p className="text-secondary">{result}</p>

<button

type="submit"

className="bg-navyblue text-white rounded-full py-2 px-6 hover:bg-opacity-90"

>

Transfer

</button>

</form>

</>

);

};

export default PayNow;

UserData.jsx

useEffect(() => {

const fetchUserData = async () => {

try {

const token = localStorage.getItem("jwtToken");

const response = await axios.get(

"http://localhost:8080/api/current-user",

{

headers: {

Authorization: `Bearer ${token}`,

},

}

);

setUserData(response.data);

} catch (error) {

// Handle error

console.error("Failed to fetch user data:", error);

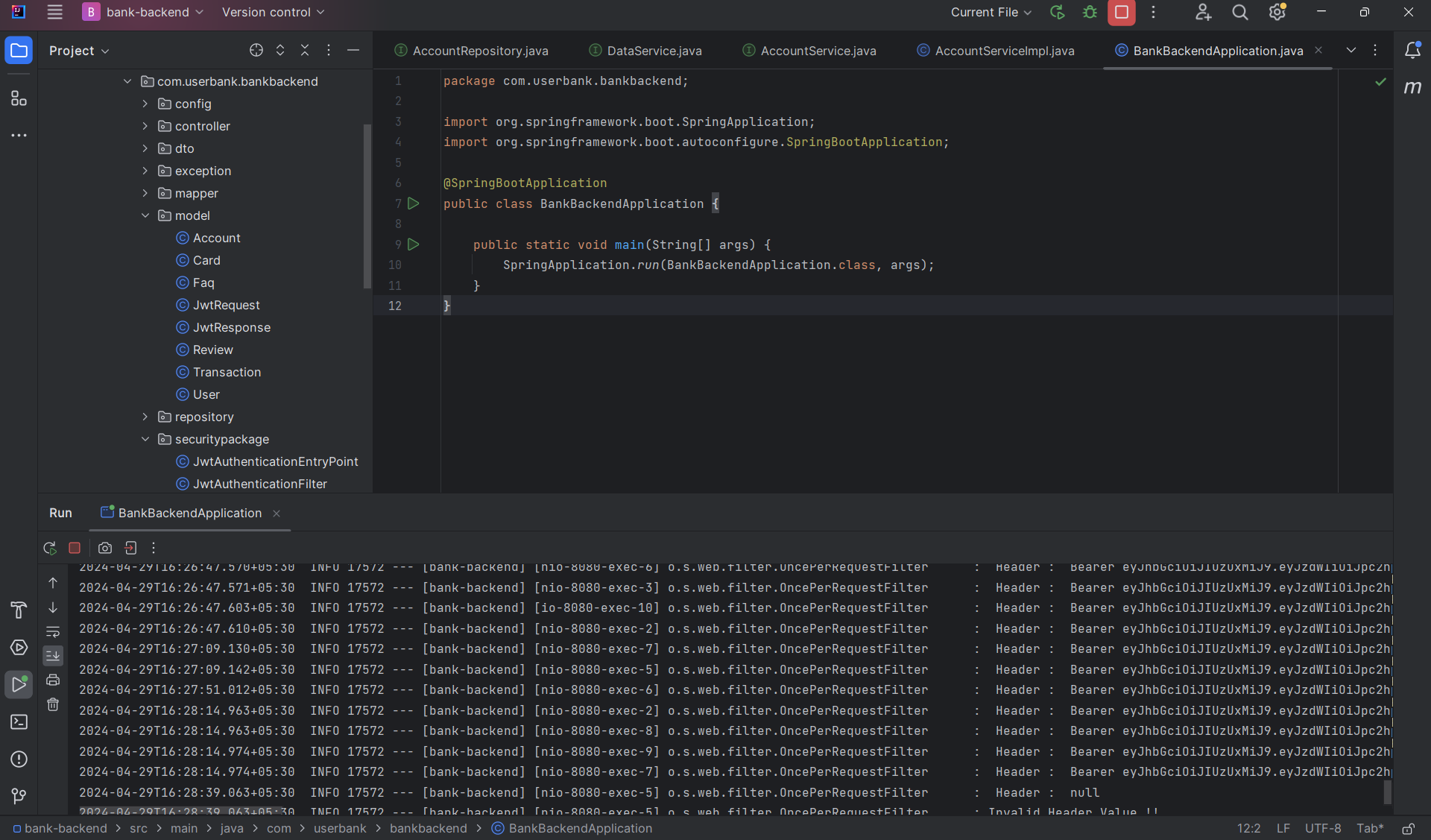
}

};

fetchUserData();

}, []);

**Backend**

****

**AccountCiontroller.java**

**package com.userbank.bankbackend.controller;**

**import com.userbank.bankbackend.dto.AccountDto;**

**import com.userbank.bankbackend.dto.TransactionDto;**

**import com.userbank.bankbackend.dto.TransferFundDto;**

**import com.userbank.bankbackend.service.AccountService;**

**import org.springframework.http.HttpStatus;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.web.bind.annotation.\*;**

**import java.util.List;**

**import java.util.Map;**

**@CrossOrigin("\*")**

**@RestController**

**@RequestMapping("/api/accounts")**

**public class AccountController {**

**private AccountService accountService;**

**public AccountController(AccountService accountService) {**

**this.accountService = accountService;**

**}**

**@PostMapping**

**public ResponseEntity<AccountDto> addAccount(@RequestBody AccountDto accountDto){**

**return new ResponseEntity<>(accountService.createAccount(accountDto), HttpStatus.*CREATED*);**

**}**

**@GetMapping("/{id}")**

**public ResponseEntity<AccountDto> getAccountById(@PathVariable Long id){**

**AccountDto accountDto=accountService.getAccountById(id);**

**return ResponseEntity.*ok*(accountDto);**

**}**

**@PutMapping("/{id}/deposit")**

**public ResponseEntity<AccountDto> deposit(@PathVariable Long id,**

**@RequestBody Map<String, Double> request){**

**Double amount = request.get("amount");**

**AccountDto accountDto = accountService.deposit(id, amount);**

**return ResponseEntity.*ok*(accountDto);**

**}**

**@PutMapping("/{id}/withdraw")**

**public ResponseEntity<AccountDto> withdraw(@PathVariable Long id,**

**@RequestBody Map<String, Double> request){**

**double amount = request.get("amount");**

**AccountDto accountDto = accountService.withdraw(id, amount);**

**return ResponseEntity.*ok*(accountDto);**

**}**

**@GetMapping**

**public ResponseEntity<List<AccountDto>> getAllAccounts(){**

**List<AccountDto> accounts = accountService.getAllAccounts();**

**return ResponseEntity.*ok*(accounts);**

**}**

**// Delete Account REST API**

**@DeleteMapping("/{id}")**

**public ResponseEntity<String> deleteAccount(@PathVariable Long id){**

**accountService.deleteAccount(id);**

**return ResponseEntity.*ok*("Account is deleted successfully!");**

**}**

**@PostMapping("/transfer")**

**public ResponseEntity<String> transferFund(@RequestBody TransferFundDto transferFundDto){**

**accountService.transferFunds(transferFundDto);**

**return ResponseEntity.*ok*("Transfer Successful");**

**}**

**@GetMapping("/{id}/transactions")**

**public ResponseEntity<List<TransactionDto>> fetchAccountTransactions(@PathVariable("id") Long accountId){**

**List<TransactionDto> transactions=accountService.getAccountTransactions(accountId);**

**return ResponseEntity.*ok*(transactions);**

**}**

**}**

**AuthController.java**

**package com.userbank.bankbackend.controller;**

**import com.userbank.bankbackend.model.JwtRequest;**

**import com.userbank.bankbackend.model.JwtResponse;**

**import com.userbank.bankbackend.securitypackage.JwtHelper;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.http.HttpStatus;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.authentication.AuthenticationManager;**

**import org.springframework.security.authentication.BadCredentialsException;**

**import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;**

**import org.springframework.security.core.userdetails.UserDetails;**

**import org.springframework.security.core.userdetails.UserDetailsService;**

**import org.springframework.web.bind.annotation.\*;**

**@RestController**

**@RequestMapping("/api/auth")**

**public class AuthController {**

**@Autowired**

**private UserDetailsService userDetailsService;**

**@Autowired**

**private AuthenticationManager manager;**

**@Autowired**

**private JwtHelper helper;**

**private Logger logger = LoggerFactory.*getLogger*(AuthController.class);**

**@PostMapping("/login")**

**public ResponseEntity<JwtResponse> login(@RequestBody JwtRequest request) {**

**this.doAuthenticate(request.getUsername(), request.getPassword());**

**UserDetails userDetails = userDetailsService.loadUserByUsername(request.getUsername());**

**String token = this.helper.generateToken(userDetails);**

**JwtResponse response = JwtResponse.*builder*()**

**.jwtToken(token)**

**.username(userDetails.getUsername()).build();**

**return new ResponseEntity<>(response, HttpStatus.*OK*);**

**}**

**private void doAuthenticate(String email, String password) {**

**UsernamePasswordAuthenticationToken authentication =**

**new UsernamePasswordAuthenticationToken(email, password);**

**try {**

**manager.authenticate(authentication);**

**} catch (BadCredentialsException e) {**

**throw new BadCredentialsException(" Invalid Username or Password !!");**

**}**

**}**

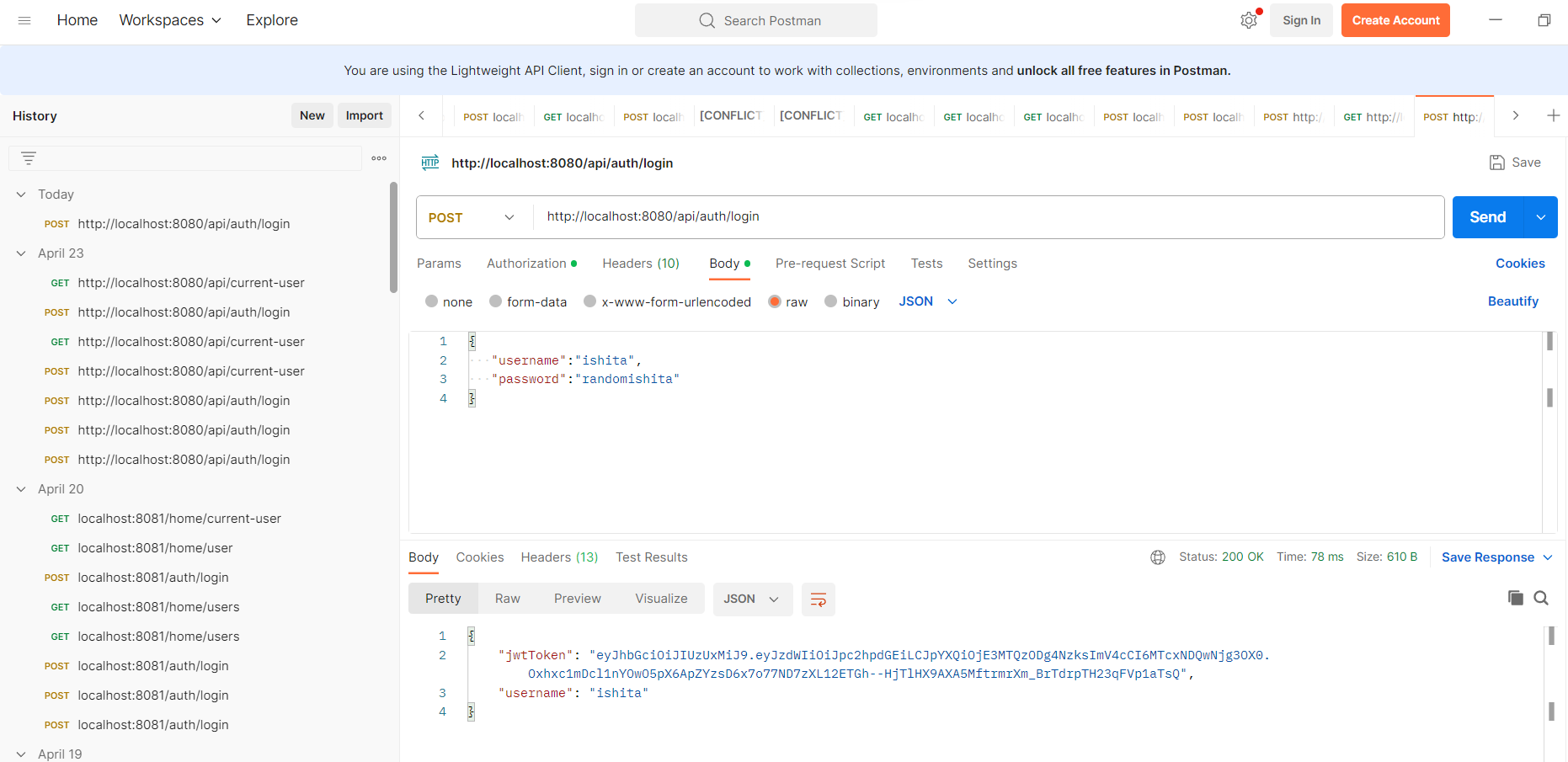
**@ExceptionHandler(BadCredentialsException.class)**

**public String exceptionHandler() {**

**return "Credentials Invalid !!";**

**}**

**}**

****

**@Component**

**public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {**

**@Override**

**public void commence(HttpServletRequest request, HttpServletResponse response, AuthenticationException authException) throws IOException, ServletException {**

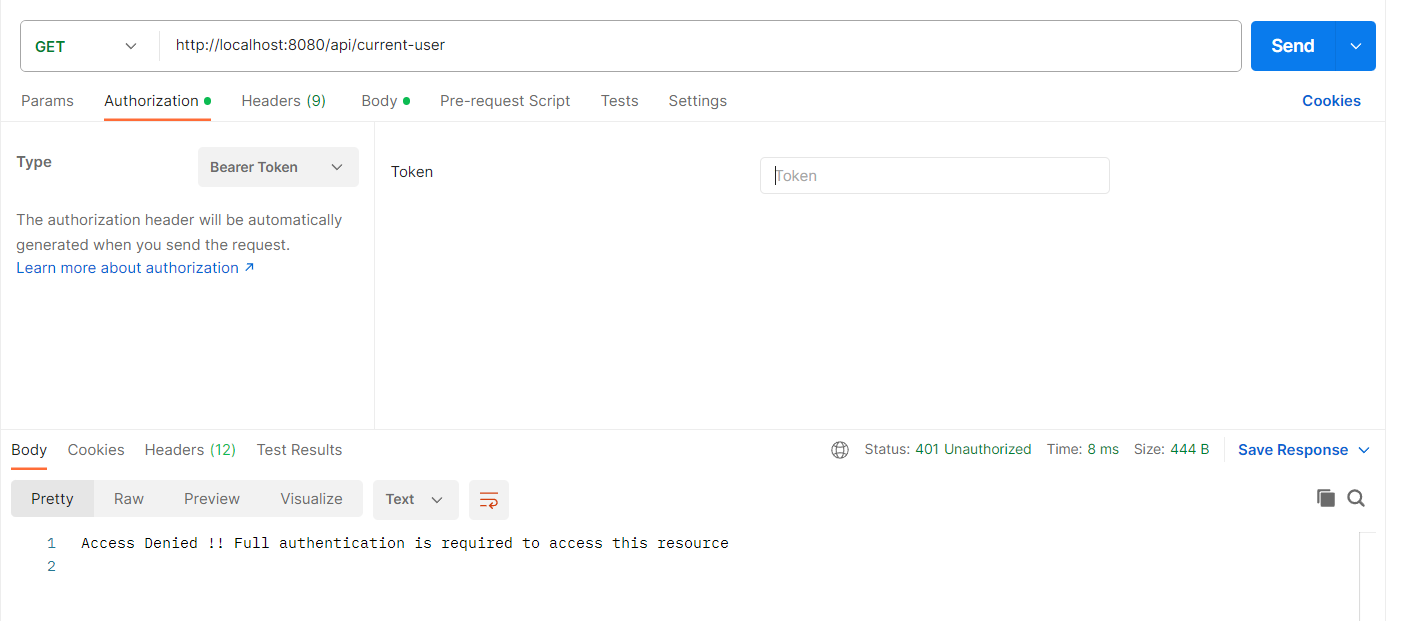
**response.setStatus(HttpServletResponse.*SC\_UNAUTHORIZED*);**

**PrintWriter writer = response.getWriter();**

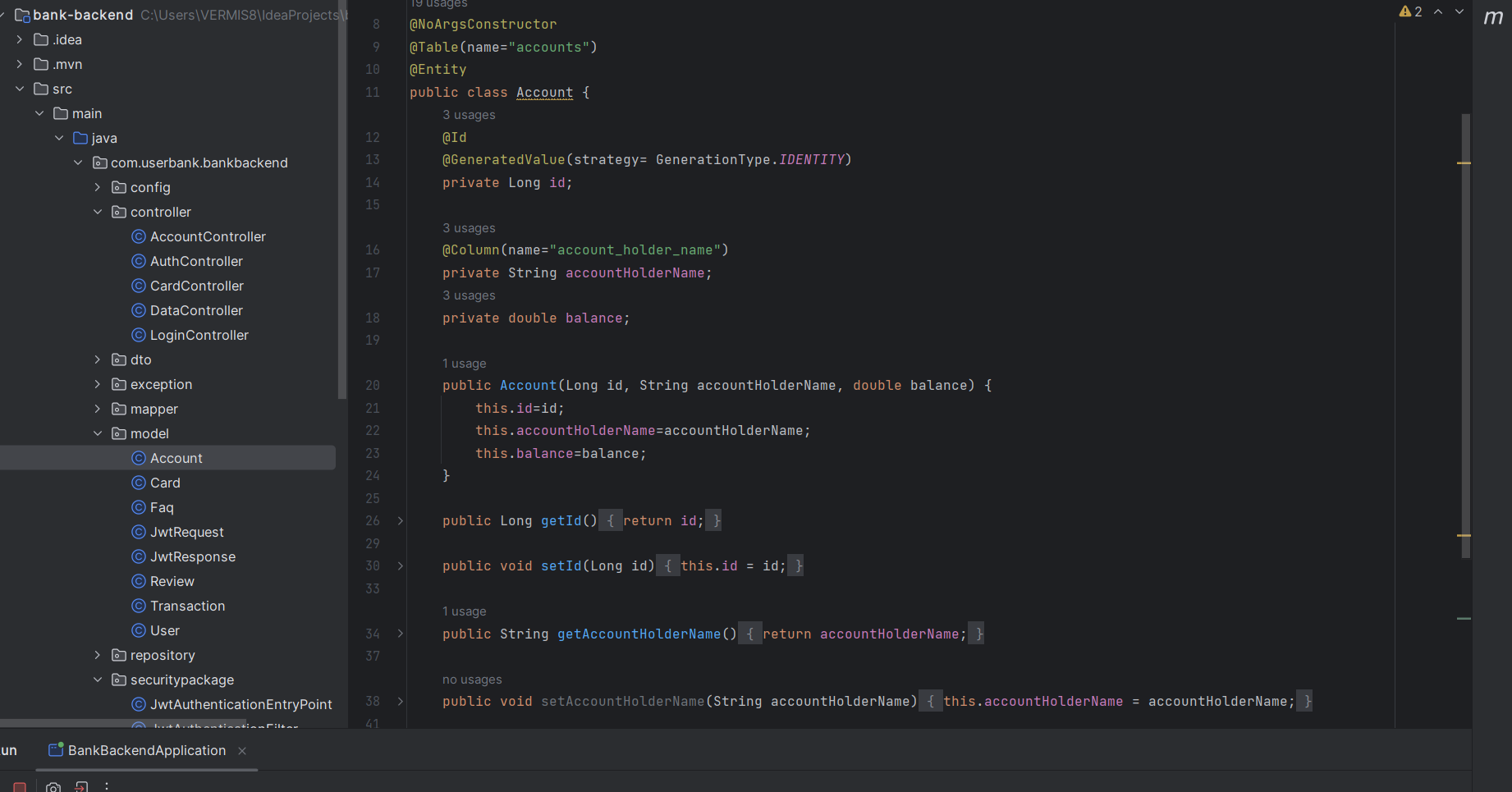
**writer.println("Access Denied !! " + authException.getMessage());**

**}**

**}**

****

**Account model:**

****

**CHAPTER-2:**

**RELATED WORK INVESTIGATION**

2.1 **Introduction**

2.2 Core area of the project

2.3 Existing Approaches/Methods

2.3.1 Approaches/Methods -1

2.3.2 Approaches/Methods -2

2.3.3 Approaches/Methods -3

2.4 Pros and cons of the stated Approaches/Methods

2.5 Issues/observations from investigation

2.6 Summary

**CHAPTER-3:**

**REQUIREMENT ARTIFACTS**

3.1 Introduction

3.2 Hardware and Software requirements

3.3 Specific Project requirements

3.3.1 Data requirement

3.3.2 Functions requirement

3.3.3 Performance and security requirement

3.3.4 Look and Feel Requirements

3.3.5 ………

3.4 Summary

**CHAPTER-4:**

**DESIGN METHODOLOGY AND ITS NOVELTY**

4.1 Methodology and goal

4.2 Functional modules design and analysis

4.3 Software Architectural designs

4.4 Subsystem services

4.5 User Interface designs

4.6 Summary

**CHAPTER-5:**

**TECHNICAL IMPLEMENTATION & ANALYSIS**

5.1Outline

5.2 Technical coding and code solutions

5.3 Working Layout of Forms

5.4 Prototype submission

5.5 Test and validation

5.6 Performance Analysis(Graphs/Charts)

5.7 Summary

**CHAPTER-6:**

**PROJECT OUTCOME AND APPLICABILITY**

6.1Outline

6.2 key implementations outlines of the System

6.3 Significant project outcomes

6.4 Project applicability on Real-world applications

6.4 Inference

**CHAPTER-7:**

**CONCLUSIONS AND RECOMMENDATION**

7.1Outline

7.2 Limitation/Constraints of the System

7.3 Future Enhancements

7.4 Inference

**REFERENCES**

1. Abdul-Wahab,S.A., Al-Alawi,S.M. and El-Zawahry, Patterns of S02 emission: a refinery case study, *Environmental modeling & software*, 2002, 17, 563-570.
2. Aggarwal A.L, Sivacoumar R. and Goyal SK Air Quality Prediction : influence of model parameters and sensitivity analysis, *Indian Journal of Environmental Protection*, 1997, 17(9), 650-655.