PROJECT REPORT
On
BANK APP

Submitted by

Ravikant Tatiwal – YM258

CHAPTER 1

Introduction

Banking is defined as the business activity of accepting and safeguarding money owned by other individuals and entities, and then lending out this money in order to conduct economic activities such as making profit or simply covering operating expenses.

A bank is a financial institution licensed to receive deposits and make loans. Two of the most common types of banks are commercial/retail and investment banks. Depending on type, a bank may also provide various financial services ranging from providing safe deposit boxes and currency exchange to retirement and wealth management.

This is a web-based project which is two real world entities Admin and Employee. Admin can do operations like ADD/EDIT/UPDATE new customers and employee of the bank and approve or reject transactions which holds amount of more than 2 Lakhs. Employee can do operations like change the details of the customer and can deposit, withdraw and transfer funds of customer.

This project uses Java, Spring Boot, Spring Security, JSP.

Chapter 2

objective

Objective of this project is to make a banking application which provides some basic functionalities like transfer, deposit and withdraw funds of customer. Add, update, delete customer and employee. Approve and reject pending transactions. Get all the list of customers and employees.

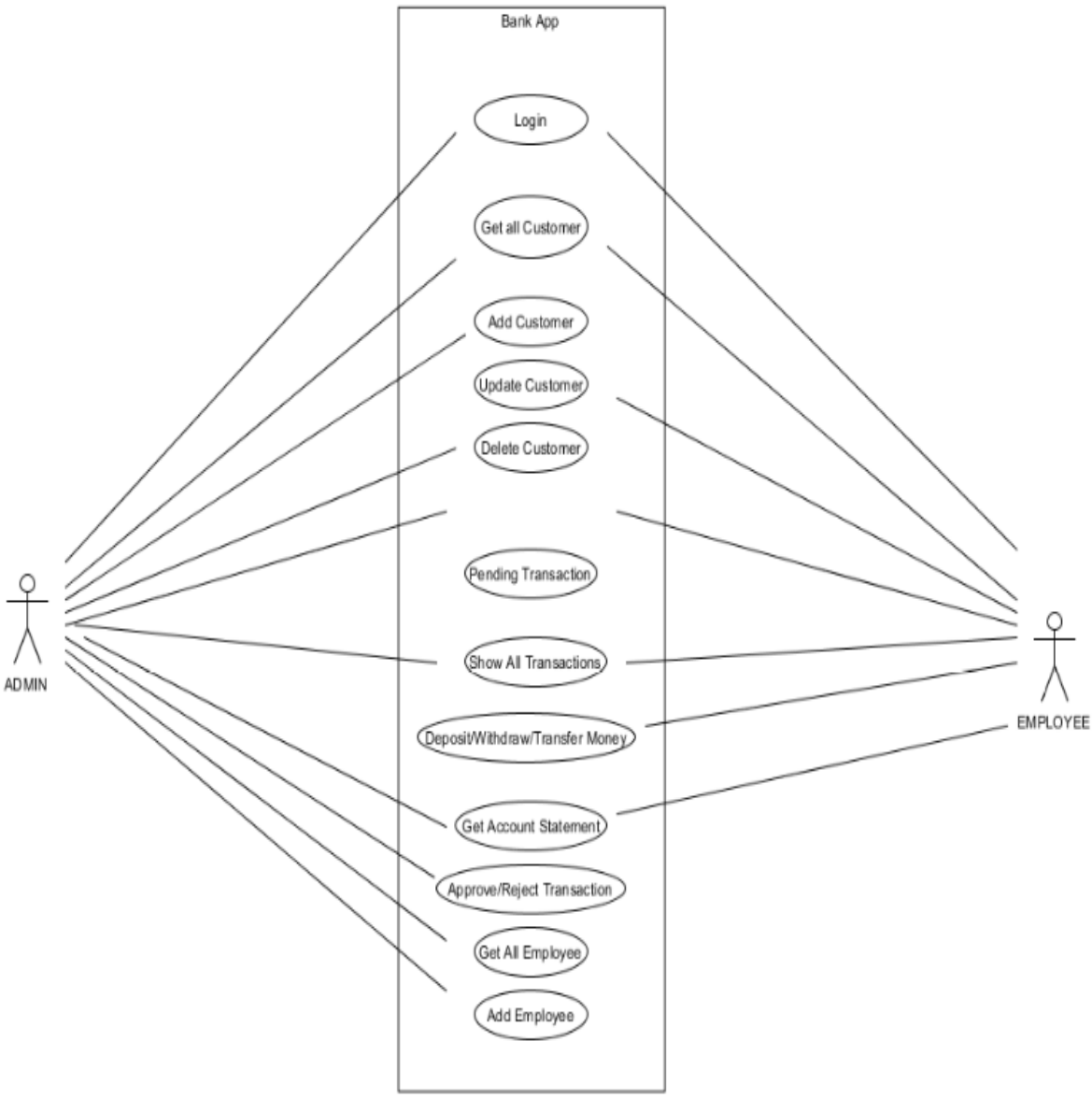
The project is designed into two modules first is for admin who can add, delete customer. Add, update, delete employee. Approve and reject transactions.

And other is Employee who can update customer. Deposit, withdraw and transfer funds for customer.

CHAPTER 4

Ooad of project

4.1 Use Case Diagram



Above shows the use case diagram of Bank application. It is a graphical depiction of employee's and admin's possible interactions with a system.

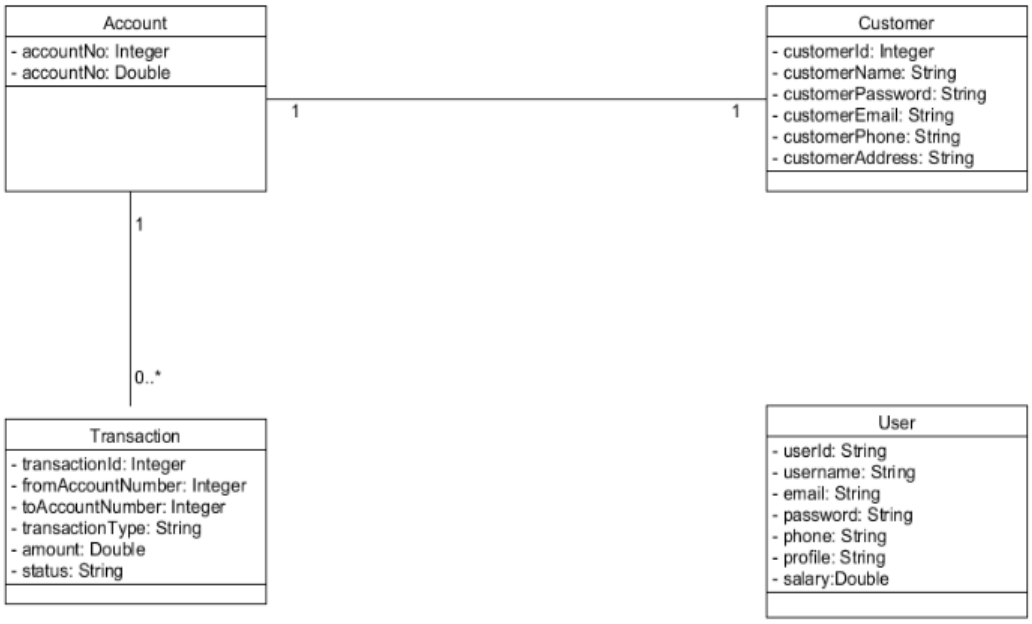
Here use case diagram describe the relationship between users and use cases. A use case is a user activity in the system. It consist of two components,

1. Actor : Actors are represented with a label naming actors role. There may be multiple actors in a diagram.
2. Use case : Represented as ellipse with a label inside, naming the use case.

There may be multiple use cases in a diagram.

Actors represent the role that a user might play where each role is represented separately. Actor and Use case names must be unique with in a diagram. A use case describes the activity that is possible. It has several instances of activity throughout it's lifetime.

4.2 Class Diagram



CHAPTER 5

Technologies used in project

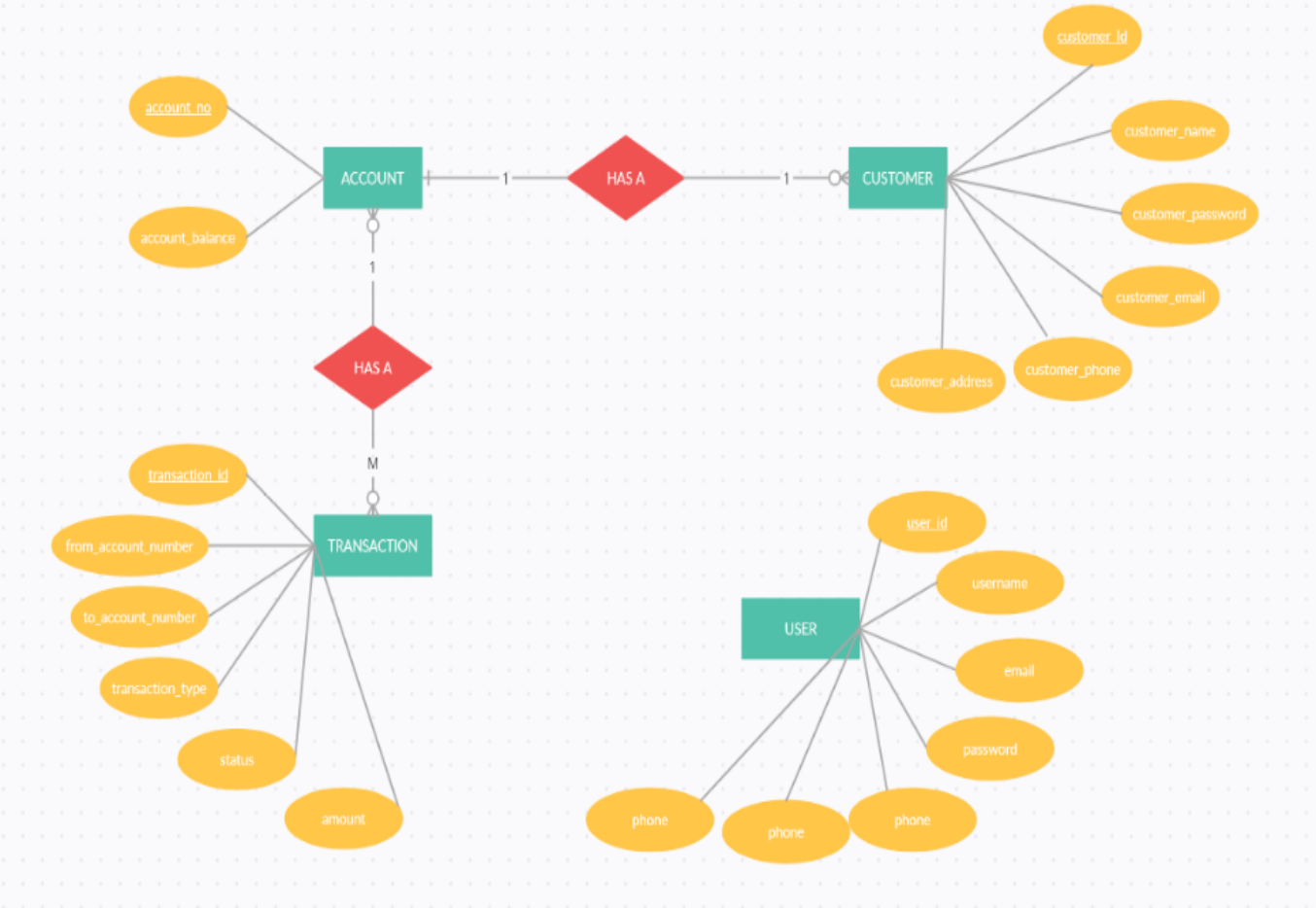
- Java 8
- Spring Boot
- Maven
- Spring Security
- Spring Data JPA
- JSR 303 Validation
- Hibernate
- JSP
- MySql-Database

CHAPTER 6

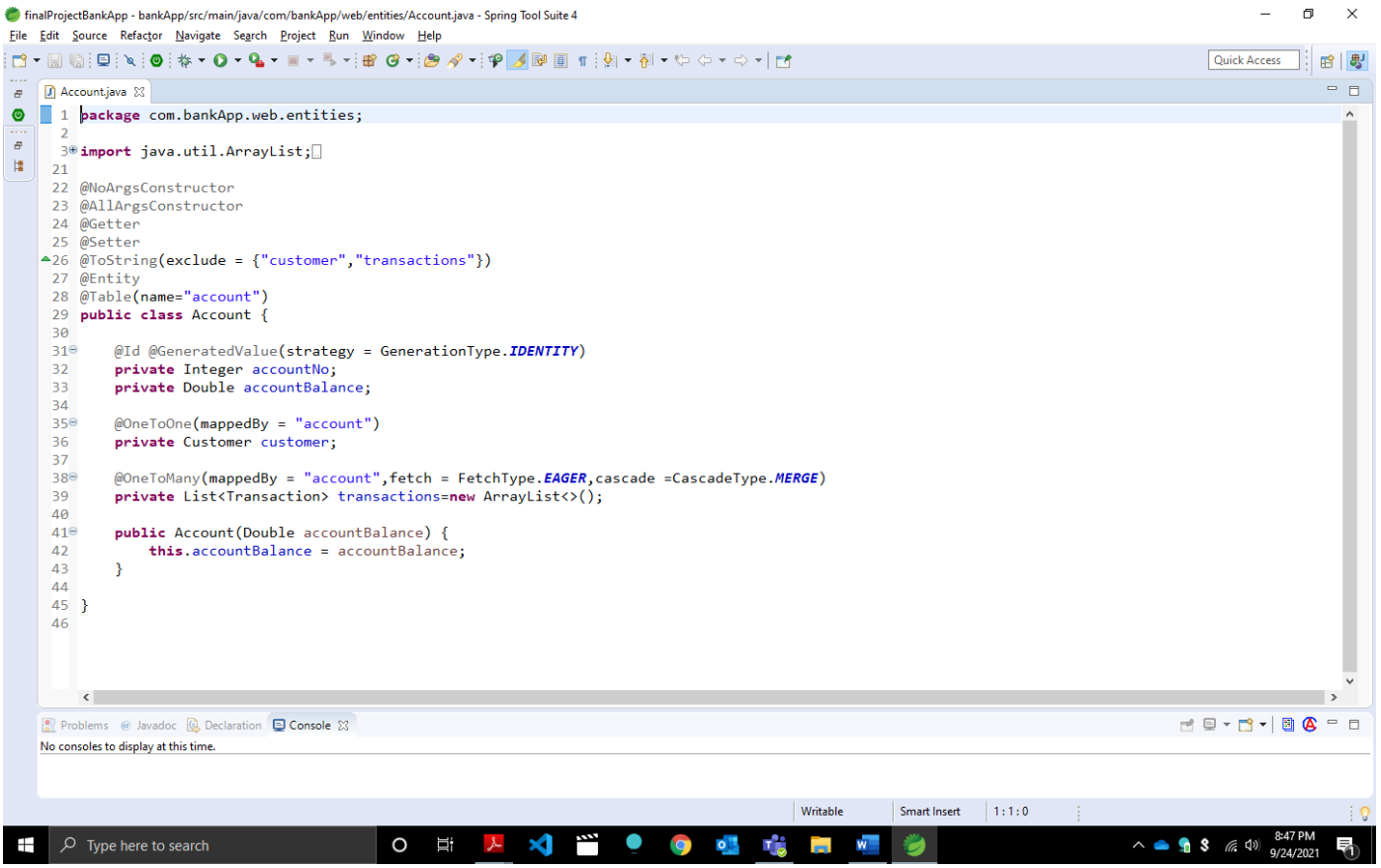
DATABASE DESIGN OF PROJECT

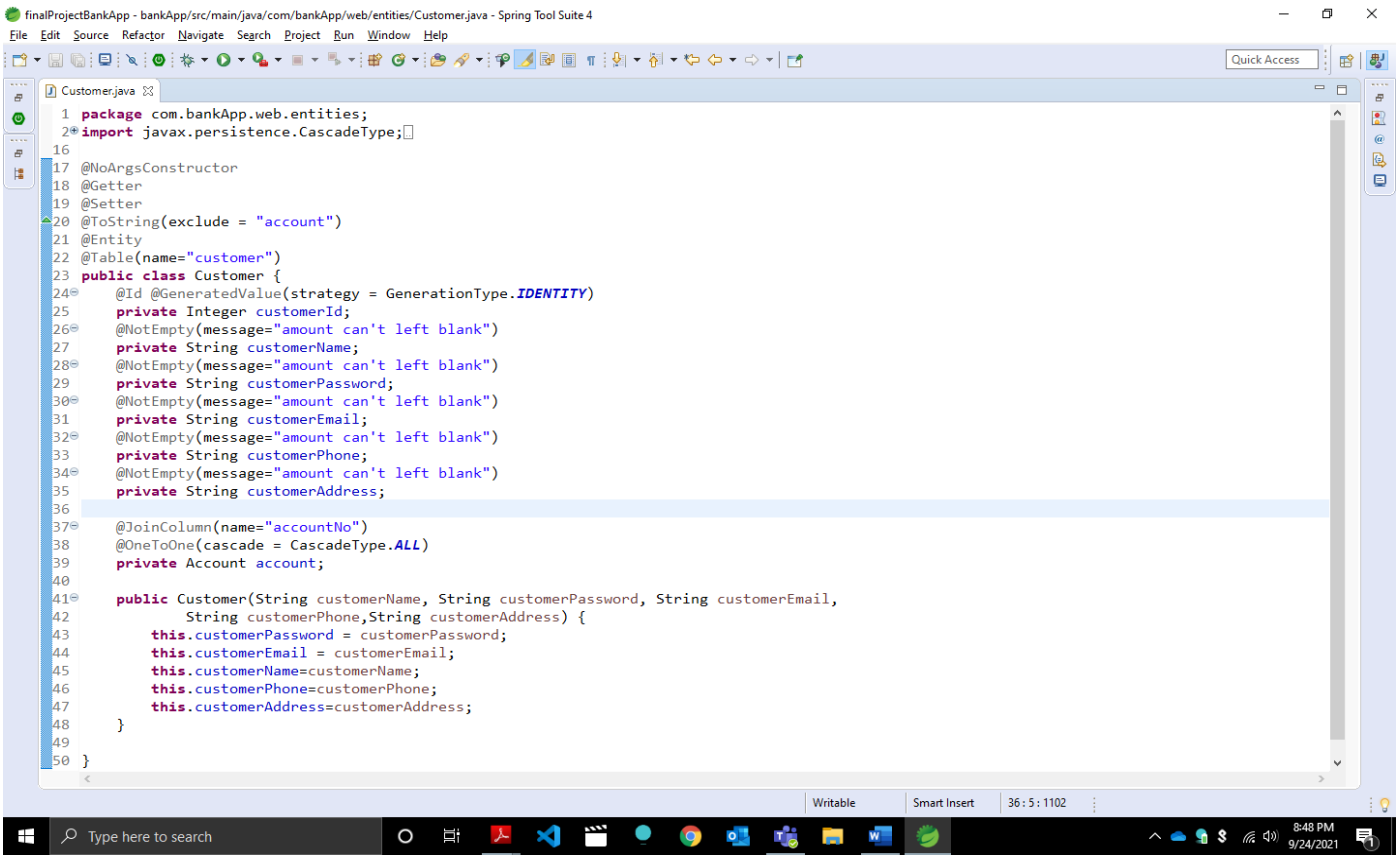
6.1 ER Diagrams

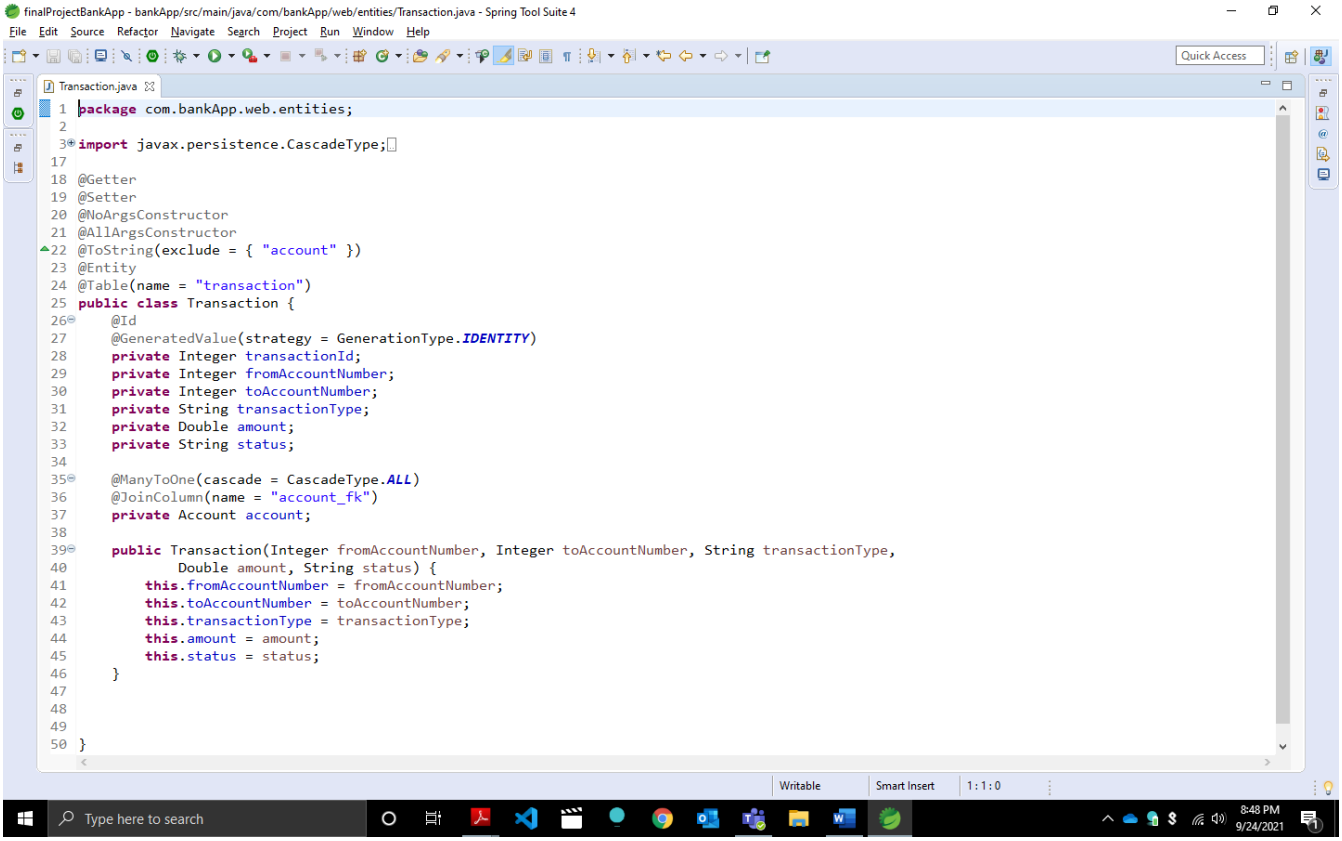
An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology system.

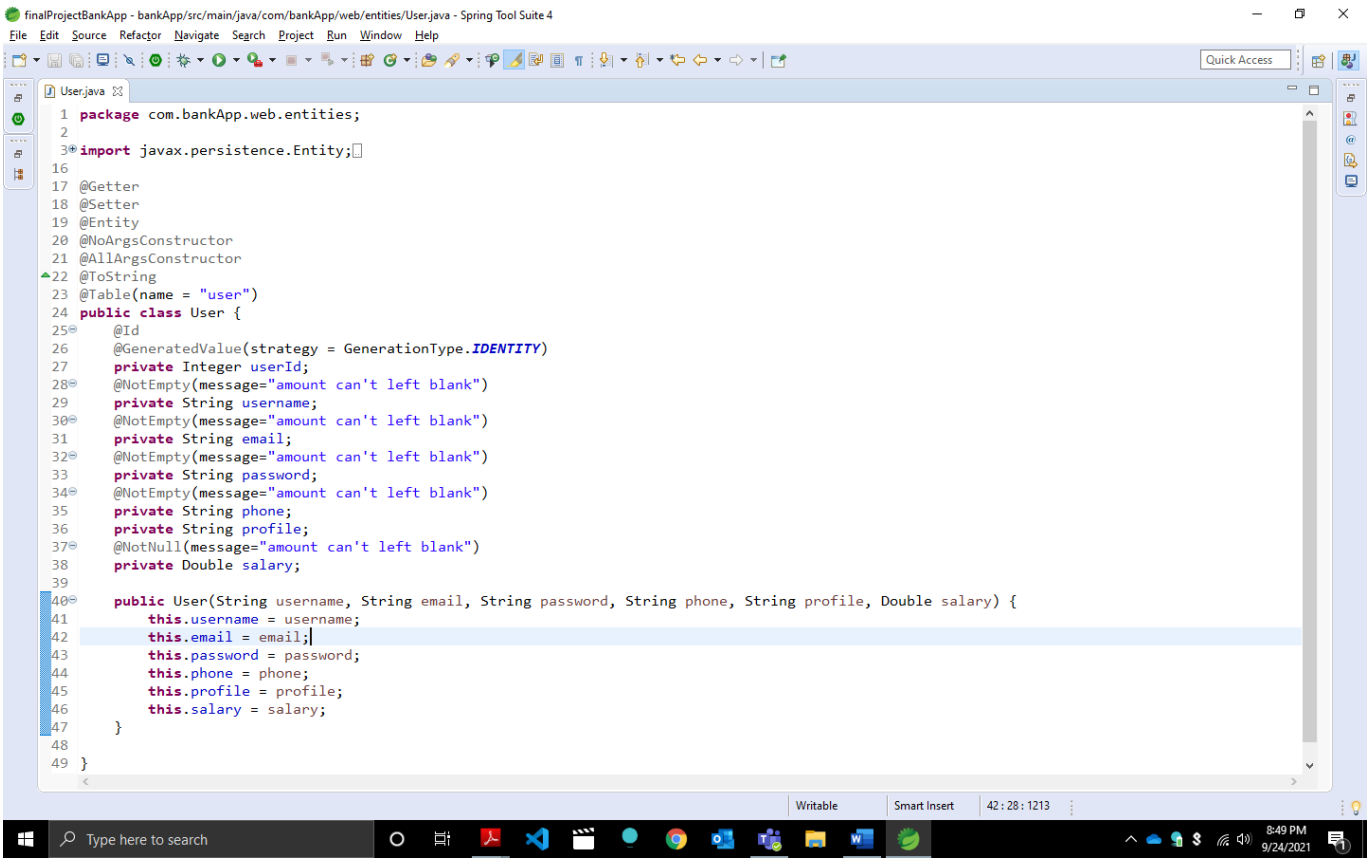


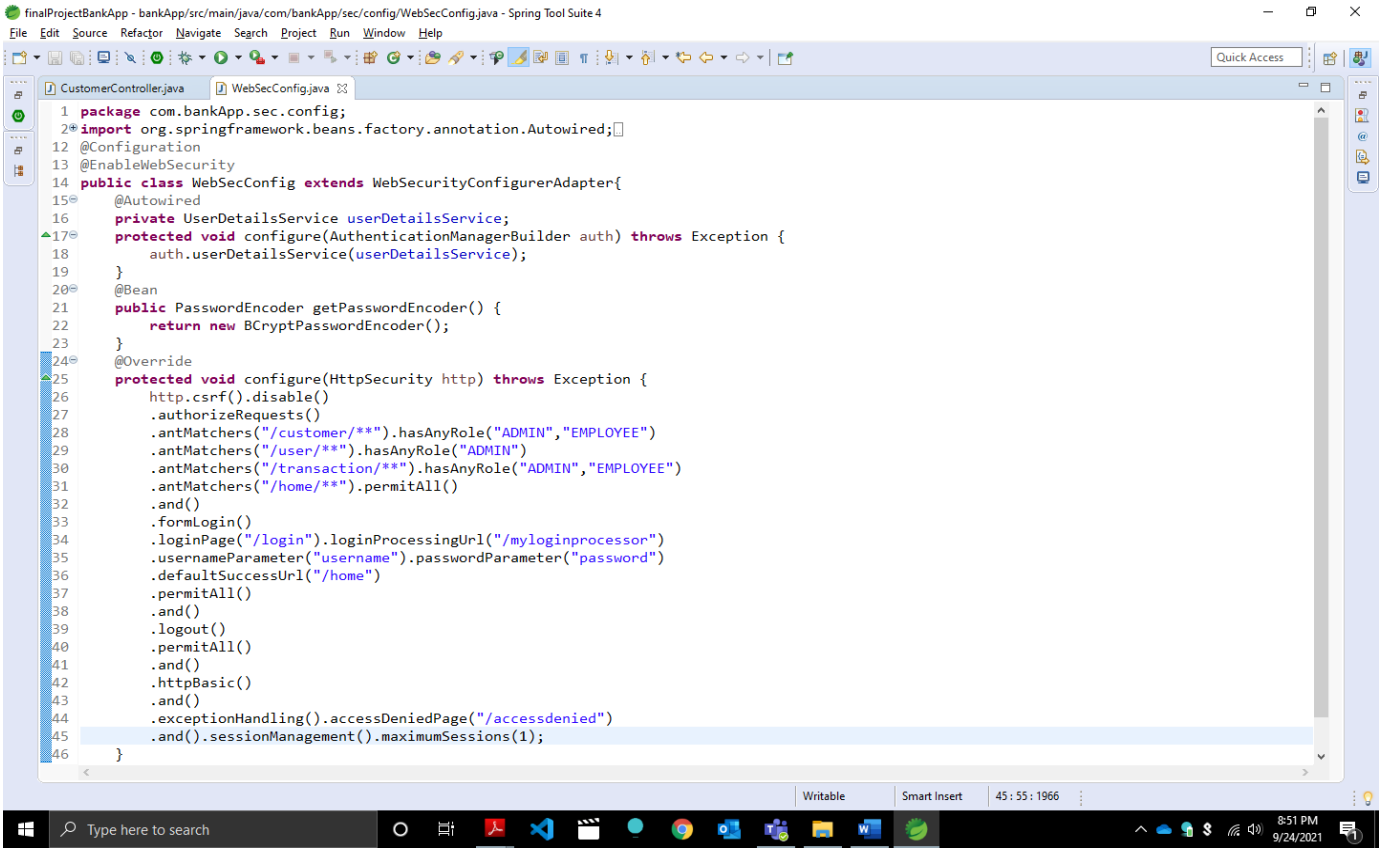
Working Screenshots

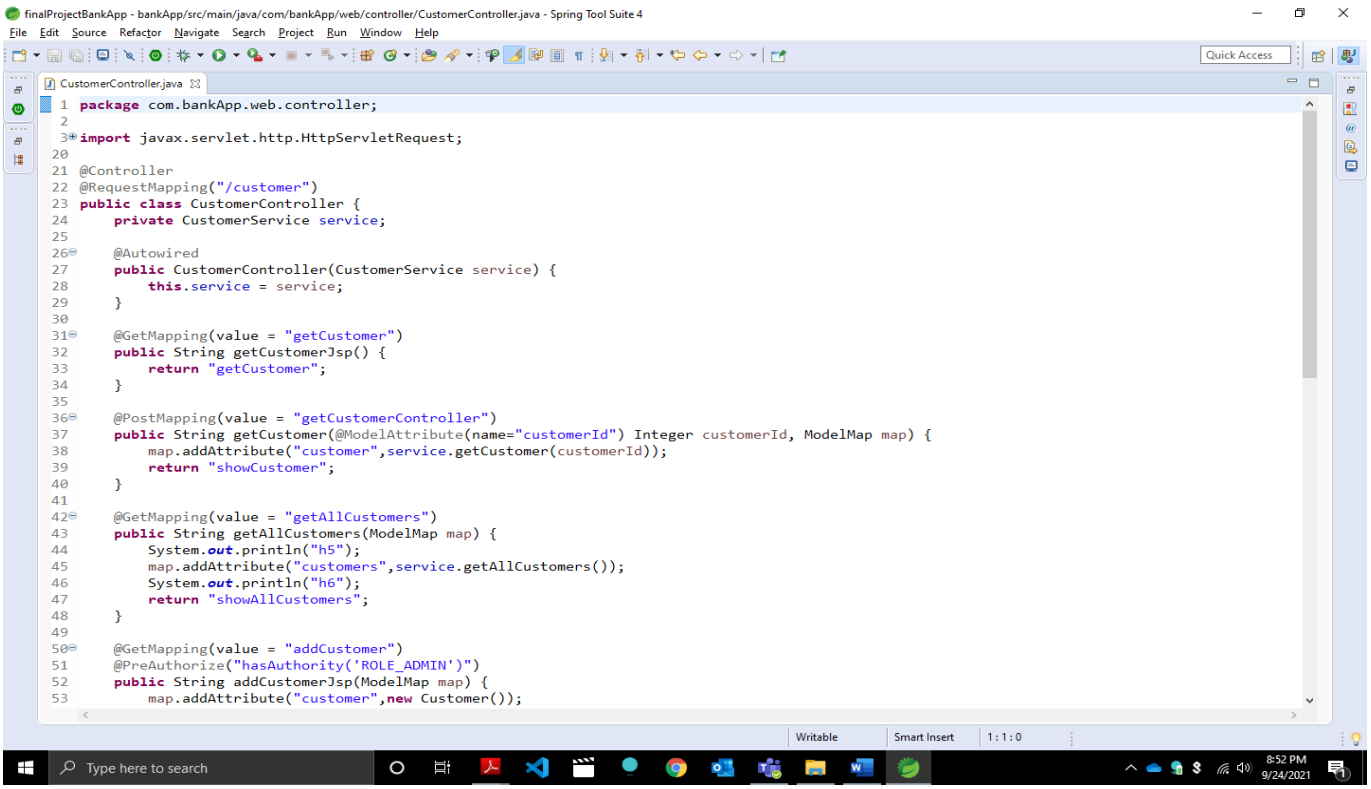

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/entities/Account.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Account.java
package com.bankApp.web.entities;
import java.util.ArrayList;
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@ToString(exclude = {"customer","transactions"})
@Entity
@Table(name="account")
public class Account {
@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer accountNo;
private Double accountBalance;
@OneToOne(mappedBy = "account")
private Customer customer;
@OneToMany(mappedBy = "account",fetch = FetchType.EAGER,cascade =CascadeType.MERGE)
private List<Transaction> transactions=new ArrayList<>();
public Account(Double accountBalance) {
this.accountBalance = accountBalance;
}
}
Problems Javadoc Declaration Console
No consoles to display at this time.
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:47 PM
9/24/2021

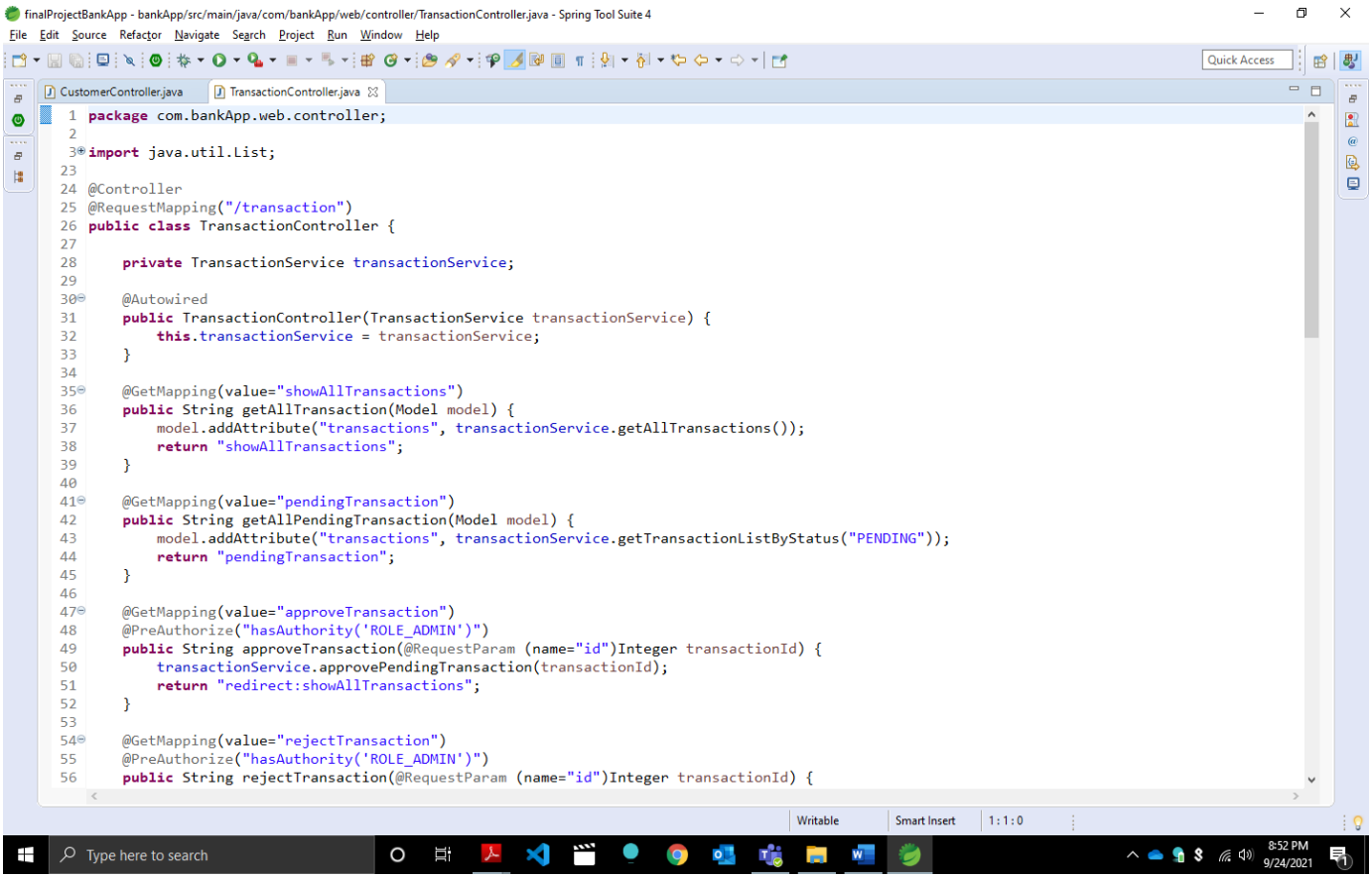

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/entities/Customer.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Customer.java
1 package com.bankApp.web.entities;
2 import javax.persistence.CascadeType;
16
17 @NoArgsConstructor
18 @Getter
19 @Setter
20 @ToString(exclude = "account")
21 @Entity
22 @Table(name="customer")
23 public class Customer {
24 @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
25 private Integer customerId;
26 @NotEmpty(message="amount can't left blank")
27 private String customerName;
28 @NotEmpty(message="amount can't left blank")
29 private String customerPassword;
30 @NotEmpty(message="amount can't left blank")
31 private String customerEmail;
32 @NotEmpty(message="amount can't left blank")
33 private String customerPhone;
34 @NotEmpty(message="amount can't left blank")
35 private String customerAddress;
36
37 @JoinColumn(name="accountNo")
38 @OneToOne(cascade = CascadeType.ALL)
39 private Account account;
40
41 public Customer(String customerName, String customerPassword, String customerEmail,
42 String customerPhone,String customerAddress) {
43 this.customerPassword = customerPassword;
44 this.customerEmail = customerEmail;
45 this.customerName=customerName;
46 this.customerPhone=customerPhone;
47 this.customerAddress=customerAddress;
48 }
49
50 }
Writable
Smart Insert
36 : 5 : 1102
Type here to search
8:48 PM
9/24/2021

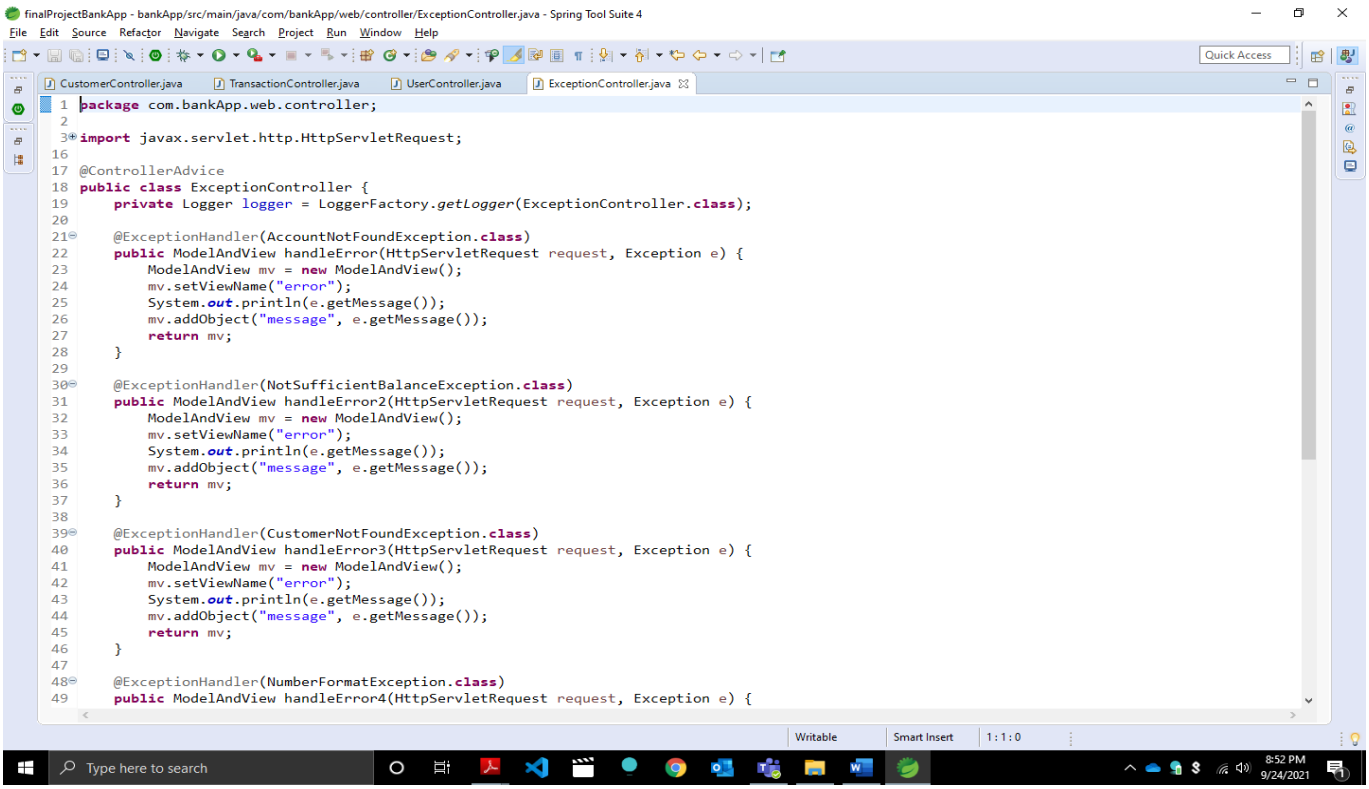

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/entities/Transaction.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Transaction.java
1 package com.bankApp.web.entities;
2
3 import javax.persistence.CascadeType;
17
18 @Getter
19 @Setter
20 @NoArgsConstructor
21 @AllArgsConstructor
22 @ToString(exclude = { "account" })
23 @Entity
24 @Table(name = "transaction")
25 public class Transaction {
26 @Id
27 @GeneratedValue(strategy = GenerationType.IDENTITY)
28 private Integer transactionId;
29 private Integer fromAccountNumber;
30 private Integer toAccountNumber;
31 private String transactionType;
32 private Double amount;
33 private String status;
34
35 @ManyToOne(cascade = CascadeType.ALL)
36 @JoinColumn(name = "account_fk")
37 private Account account;
38
39 public Transaction(Integer fromAccountNumber, Integer toAccountNumber, String transactionType,
40 Double amount, String status) {
41 this.fromAccountNumber = fromAccountNumber;
42 this.toAccountNumber = toAccountNumber;
43 this.transactionType = transactionType;
44 this.amount = amount;
45 this.status = status;
46 }
47
48
49
50 }
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:48 PM
9/24/2021

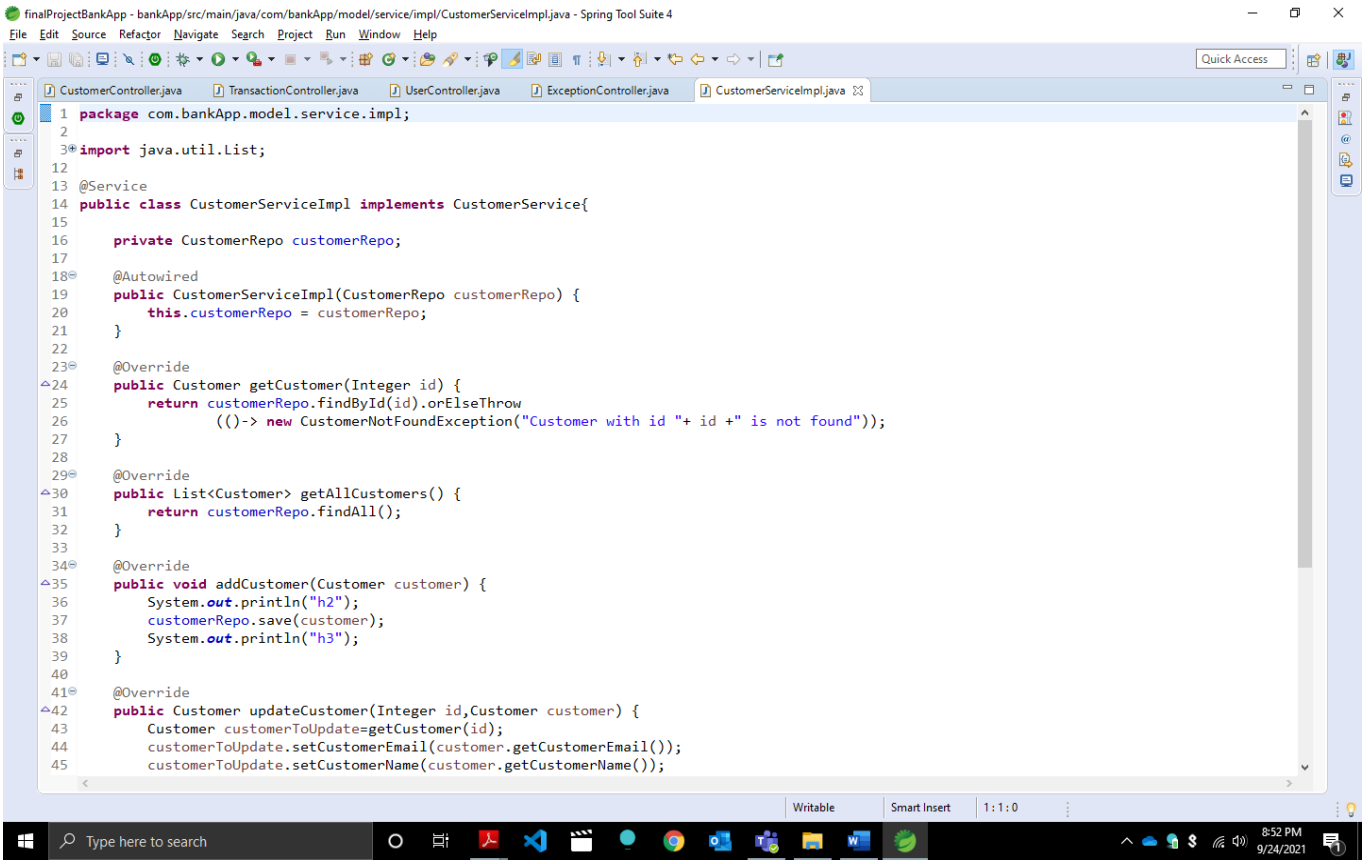

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/entities/User.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
User.java
1 package com.bankApp.web.entities;
2
3 import javax.persistence.Entity;
16
17 @Getter
18 @Setter
19 @Entity
20 @NoArgsConstructor
21 @AllArgsConstructor
22 @ToString
23 @Table(name = "user")
24 public class User {
25 @Id
26 @GeneratedValue(strategy = GenerationType.IDENTITY)
27 private Integer userId;
28 @NotEmpty(message="amount can't left blank")
29 private String username;
30 @NotEmpty(message="amount can't left blank")
31 private String email;
32 @NotEmpty(message="amount can't left blank")
33 private String password;
34 @NotEmpty(message="amount can't left blank")
35 private String phone;
36 private String profile;
37 @NotNull(message="amount can't left blank")
38 private Double salary;
39
40 public User(String username, String email, String password, String phone, String profile, Double salary) {
41 this.username = username;
42 this.email = email;
43 this.password = password;
44 this.phone = phone;
45 this.profile = profile;
46 this.salary = salary;
47 }
48
49 }
Writable
Smart Insert
42 : 28 : 1213
Type here to search
8:49 PM
9/24/2021

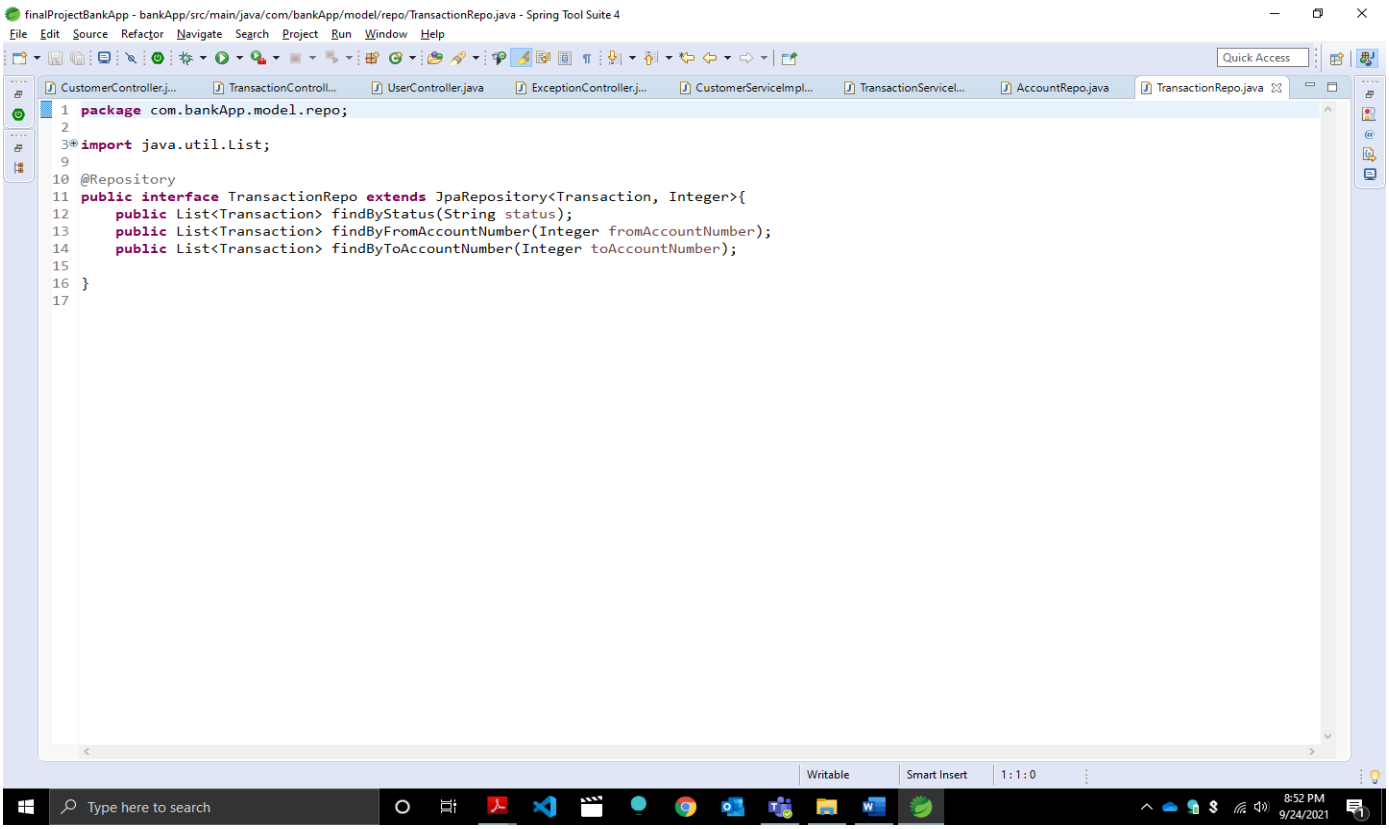

finalProjectBankApp - bankApp/src/main/java/com/bankApp/sec/config/WebSecConfig.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.java
WebSecConfig.java
1 package com.bankApp.sec.config;
2 import org.springframework.beans.factory.annotation.Autowired;
12 @Configuration
13 @EnableWebSecurity
14 public class WebSecConfig extends WebSecurityConfigurerAdapter{
15 @Autowired
16 private UserDetailsService userDetailsService;
17 protected void configure(AuthenticationManagerBuilder auth) throws Exception {
18 auth.userDetailsService(userDetailsService);
19 }
20 @Bean
21 public PasswordEncoder getPasswordEncoder() {
22 return new BCryptPasswordEncoder();
23 }
24 @Override
25 protected void configure(HttpSecurity http) throws Exception {
26 http.csrf().disable()
27 .authorizeRequests()
28 .antMatchers("/customer/**").hasAnyRole("ADMIN","EMPLOYEE")
29 .antMatchers("/user/**").hasAnyRole("ADMIN")
30 .antMatchers("/transaction/**").hasAnyRole("ADMIN","EMPLOYEE")
31 .antMatchers("/home/**").permitAll()
32 .and()
33 .formLogin()
34 .loginPage("/login").loginProcessingUrl("/myloginprocessor")
35 .usernameParameter("username").passwordParameter("password")
36 .defaultSuccessUrl("/home")
37 .permitAll()
38 .and()
39 .logout()
40 .permitAll()
41 .and()
42 .httpBasic()
43 .and()
44 .exceptionHandling().accessDeniedPage("/accessdenied")
45 .and().sessionManagement().maximumSessions(1);
46 }
Writable
Smart Insert
45 : 55 : 1966
Type here to search
8:51 PM
9/24/2021

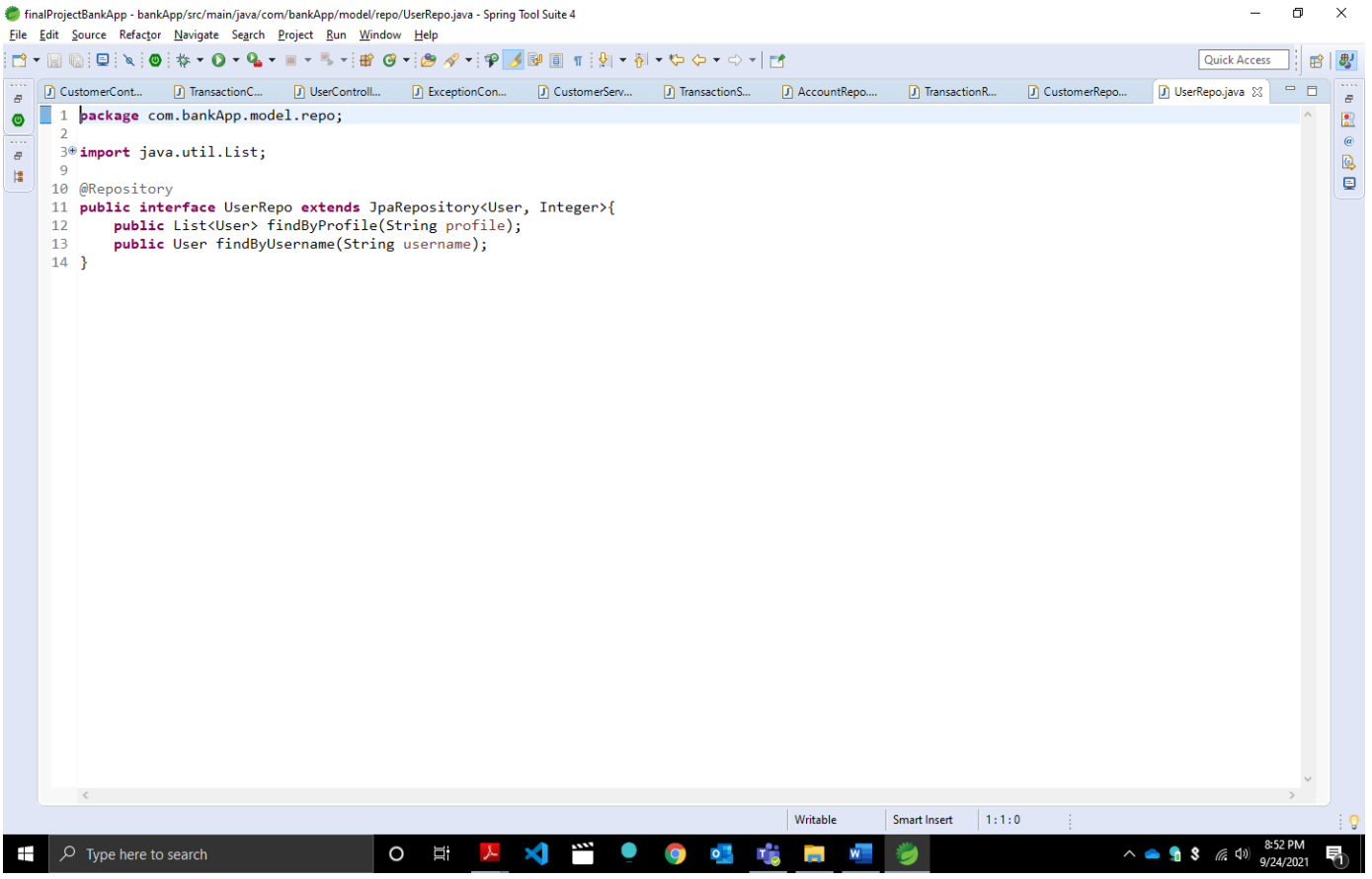

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/controller/CustomerController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.java
1 package com.bankApp.web.controller;
2
3 import javax.servlet.http.HttpServletRequest;
20
21 @Controller
22 @RequestMapping("/customer")
23 public class CustomerController {
24 private CustomerService service;
25
26 @Autowired
27 public CustomerController(CustomerService service) {
28 this.service = service;
29 }
30
31 @GetMapping(value = "getCustomer")
32 public String getCustomerJsp() {
33 return "getCustomer";
34 }
35
36 @PostMapping(value = "getCustomerController")
37 public String getCustomer(@ModelAttribute(name="customerId") Integer customerId, ModelMap map) {
38 map.addAttribute("customer",service.getCustomer(customerId));
39 return "showCustomer";
40 }
41
42 @GetMapping(value = "getAllCustomers")
43 public String getAllCustomers(ModelMap map) {
44 System.out.println("h5");
45 map.addAttribute("customers",service.getAllCustomers());
46 System.out.println("h6");
47 return "showAllCustomers";
48 }
49
50 @GetMapping(value = "addCustomer")
51 @PreAuthorize("hasAuthority('ROLE_ADMIN')")
52 public String addCustomerJsp(ModelMap map) {
53 map.addAttribute("customer",new Customer());
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:52 PM
9/24/2021

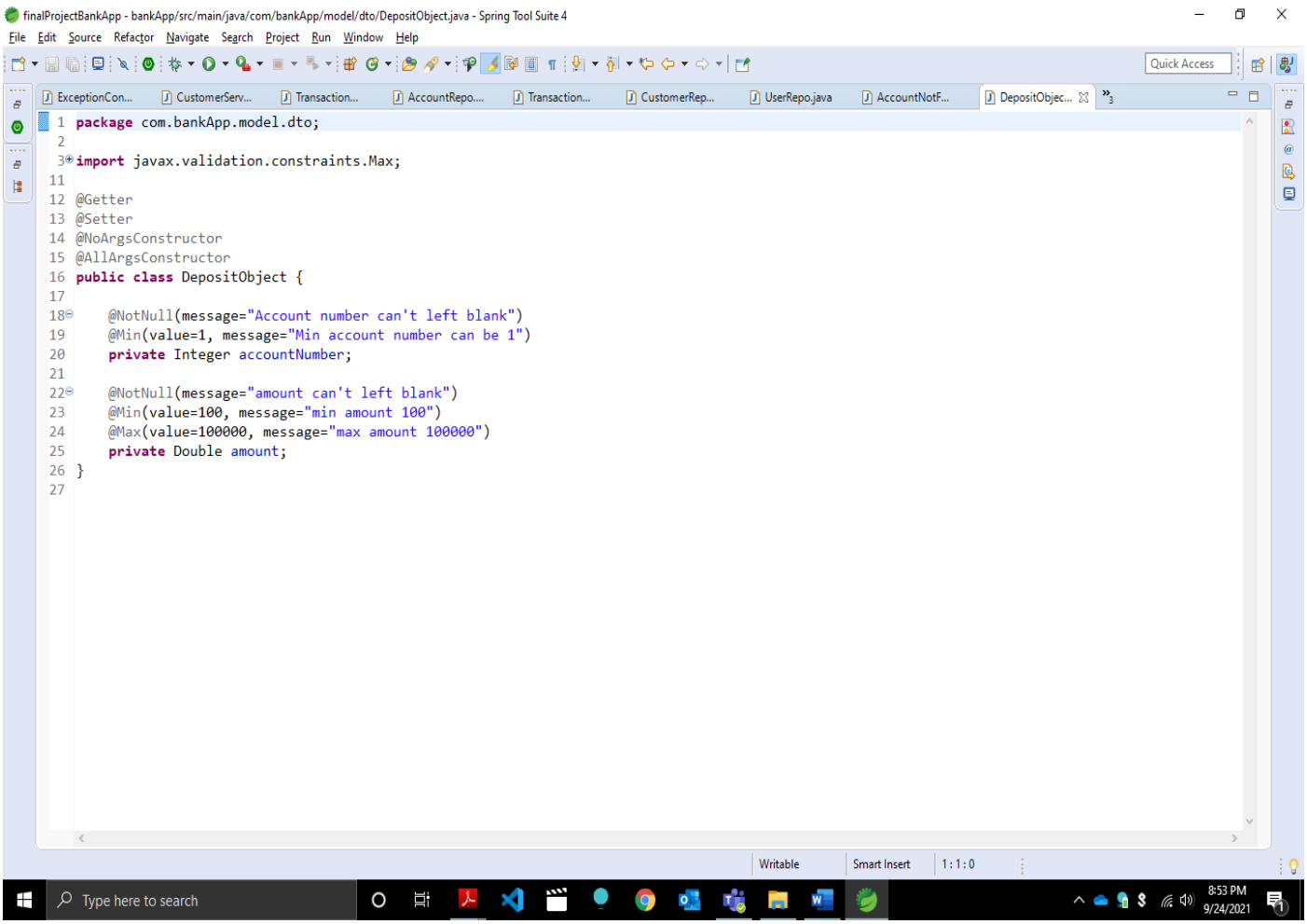

finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/controller/TransactionController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.java
TransactionController.java
1 package com.bankApp.web.controller;
2
3 import java.util.List;
23
24 @Controller
25 @RequestMapping("/transaction")
26 public class TransactionController {
27
28 private TransactionService transactionService;
29
30 @Autowired
31 public TransactionController(TransactionService transactionService) {
32 this.transactionService = transactionService;
33 }
34
35 @GetMapping(value="showAllTransactions")
36 public String getAllTransaction(Model model) {
37 model.addAttribute("transactions", transactionService.getAllTransactions());
38 return "showAllTransactions";
39 }
40
41 @GetMapping(value="pendingTransaction")
42 public String getAllPendingTransaction(Model model) {
43 model.addAttribute("transactions", transactionService.getTransactionListByStatus("PENDING"));
44 return "pendingTransaction";
45 }
46
47 @GetMapping(value="approveTransaction")
48 @PreAuthorize("hasAuthority('ROLE_ADMIN')")
49 public String approveTransaction(@RequestParam (name="id")Integer transactionId) {
50 transactionService.approvePendingTransaction(transactionId);
51 return "redirect:showAllTransactions";
52 }
53
54 @GetMapping(value="rejectTransaction")
55 @PreAuthorize("hasAuthority('ROLE_ADMIN')")
56 public String rejectTransaction(@RequestParam (name="id")Integer transactionId) {
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:52 PM
9/24/2021


finalProjectBankApp - bankApp/src/main/java/com/bankApp/web/controller/ExceptionController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.java
TransactionController.java
UserController.java
ExceptionController.java
1 package com.bankApp.web.controller;
2
3 import javax.servlet.http.HttpServletRequest;
16
17 @ControllerAdvice
18 public class ExceptionController {
19 private Logger logger = LoggerFactory.getLogger(ExceptionController.class);
20
21 @ExceptionHandler(AccountNotFoundException.class)
22 public ModelAndView handleError(HttpServletRequest request, Exception e) {
23 ModelAndView mv = new ModelAndView();
24 mv.setViewName("error");
25 System.out.println(e.getMessage());
26 mv.addObject("message", e.getMessage());
27 return mv;
28 }
29
30 @ExceptionHandler(NotSufficientBalanceException.class)
31 public ModelAndView handleError2(HttpServletRequest request, Exception e) {
32 ModelAndView mv = new ModelAndView();
33 mv.setViewName("error");
34 System.out.println(e.getMessage());
35 mv.addObject("message", e.getMessage());
36 return mv;
37 }
38
39 @ExceptionHandler(CustomerNotFoundException.class)
40 public ModelAndView handleError3(HttpServletRequest request, Exception e) {
41 ModelAndView mv = new ModelAndView();
42 mv.setViewName("error");
43 System.out.println(e.getMessage());
44 mv.addObject("message", e.getMessage());
45 return mv;
46 }
47
48 @ExceptionHandler(NumberFormatException.class)
49 public ModelAndView handleError4(HttpServletRequest request, Exception e) {
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:52 PM
9/24/2021


finalProjectBankApp - bankApp/src/main/java/com/bankApp/model/service/impl/CustomerServiceImpl.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.java
TransactionController.java
UserController.java
ExceptionController.java
CustomerServiceImpl.java
1 package com.bankApp.model.service.impl;
2
3 import java.util.List;
12
13 @Service
14 public class CustomerServiceImpl implements CustomerService{
15
16 private CustomerRepo customerRepo;
17
18 @Autowired
19 public CustomerServiceImpl(CustomerRepo customerRepo) {
20 this.customerRepo = customerRepo;
21 }
22
23 @Override
24 public Customer getCustomer(Integer id) {
25 return customerRepo.findById(id).orElseThrow
26 (()-> new CustomerNotFoundException("Customer with id "+ id +" is not found"));
27 }
28
29 @Override
30 public List<Customer> getAllCustomers() {
31 return customerRepo.findAll();
32 }
33
34 @Override
35 public void addCustomer(Customer customer) {
36 System.out.println("h2");
37 customerRepo.save(customer);
38 System.out.println("h3");
39 }
40
41 @Override
42 public Customer updateCustomer(Integer id,Customer customer) {
43 Customer customerToUpdate=getCustomer(id);
44 customerToUpdate.setCustomerEmail(customer.getCustomerEmail());
45 customerToUpdate.setCustomerName(customer.getCustomerName());
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:52 PM
9/24/2021


finalProjectBankApp - bankApp/src/main/java/com/bankApp/model/repo/TransactionRepo.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerController.j... TransactionControll... UserController.java ExceptionController.j... CustomerServiceImpl... TransactionServicel... AccountRepo.java TransactionRepo.java
1 package com.bankApp.model.repo;
2
3 import java.util.List;
9
10 @Repository
11 public interface TransactionRepo extends JpaRepository<Transaction, Integer>{
12 public List<Transaction> findByStatus(String status);
13 public List<Transaction> findByFromAccountNumber(Integer fromAccountNumber);
14 public List<Transaction> findByToAccountNumber(Integer toAccountNumber);
15
16 }
17
Writable Smart Insert 1 : 1 : 0
Type here to search
8:52 PM
9/24/2021


finalProjectBankApp - bankApp/src/main/java/com/bankApp/model/repo/UserRepo.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
CustomerCont... TransactionC... UserControll... ExceptionCon... CustomerServ... TransactionS... AccountRepo.... TransactionR... CustomerRepo... UserRepo.java
1 package com.bankApp.model.repo;
2
3 import java.util.List;
9
10 @Repository
11 public interface UserRepo extends JpaRepository<User, Integer>{
12 public List<User> findByProfile(String profile);
13 public User findByUsername(String username);
14 }
Writable Smart Insert 1 : 1 : 0
Type here to search
8:52 PM
9/24/2021

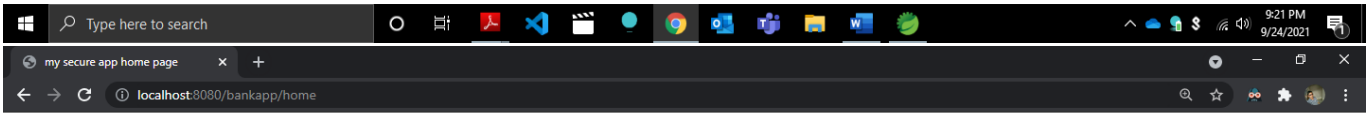

finalProjectBankApp - bankApp/src/main/java/com/bankApp/model/dto/DepositObject.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
ExceptionCon... CustomerServ... Transaction... AccountRepo.... Transaction... CustomerRep... UserRepo.java AccountNotF... DepositObjec...
1 package com.bankApp.model.dto;
2
3 import javax.validation.constraints.Max;
11
12 @Getter
13 @Setter
14 @NoArgsConstructor
15 @AllArgsConstructor
16 public class DepositObject {
17
18 @NotNull(message="Account number can't left blank")
19 @Min(value=1, message="Min account number can be 1")
20 private Integer accountNumber;
21
22 @NotNull(message="amount can't left blank")
23 @Min(value=100, message="min amount 100")
24 @Max(value=100000, message="max amount 100000")
25 private Double amount;
26 }
27
Writable
Smart Insert
1 : 1 : 0
Type here to search
8:53 PM
9/24/2021

Please login with your username and password

Username:

Password:

Submit



Home page

get a customers

get all customers

Add Customer

Show All Transactions

Pending Transaction

Withdraw Money

Deposit Money

Transfer Money

Get all employee

Add Employee

logout

Enter Customer details

Enter customer name
Enter customer password
Enter customer email
Enter customer phone number
Enter customer address
Enter customer balance
Submit

Enter Customer details

Enter customer name amount cant left blank
Enter customer password amount cant left blank
Enter customer email amount cant left blank
Enter customer phone number amount cant left blank
Enter customer address amount cant left blank
Enter customer balance
Submit

showing customers

localhost:8080/bankapp/customer/getAllCustomers

Customer Id	Customer Name	Customer Email	Customer Phone	Customer Address	Customer Account Number	Customer Account Balance	update	delete	accountStatement
1	abhishek	abhi@gmail.com	65876868	rajasthan	1	234242.0	update	delete	statement
2	himansuh	himanshu@gmail.com	4564645	hariyana	2	3232.0	update	delete	statement
4	rk	rk@gmail.com	2231986	jaipur	4	456476.0	update	delete	statement

- Add new Customer
- View all Transactions
- Withdraw Money
- Deposit Money
- Transfer Money
- View all Pending Transactions

localhost:8080/bankapp/transaction/depositMoney

Type here to search

9:22 PM
9/24/2021

Insert title here

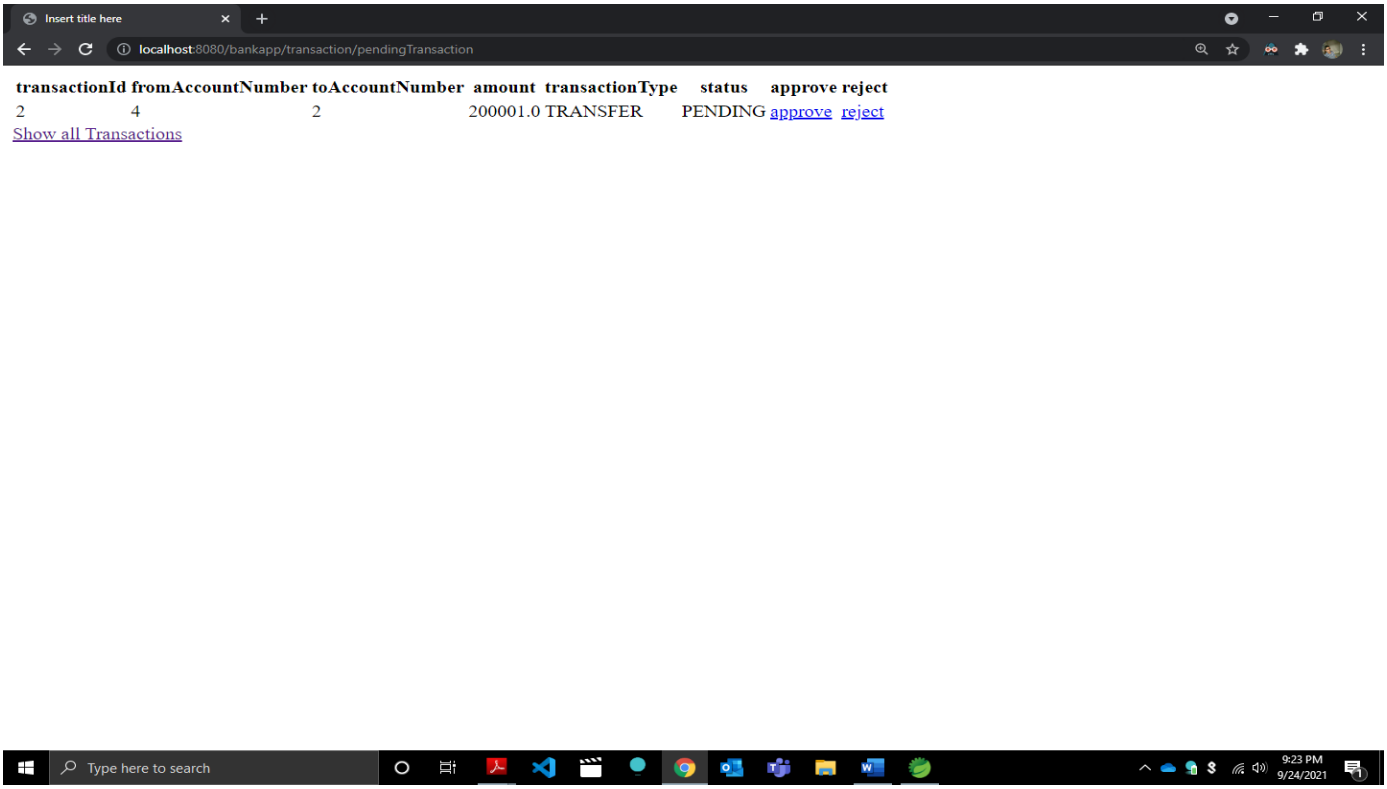
localhost:8080/bankapp/transaction/showAllTransactions

transactionId	fromAccountNumber	toAccountNumber	amount	transactionType	status
1	self	1	200.0	DEPOSIT	COMPLETED
2	4	2	200001.0	TRANSFER	PENDING

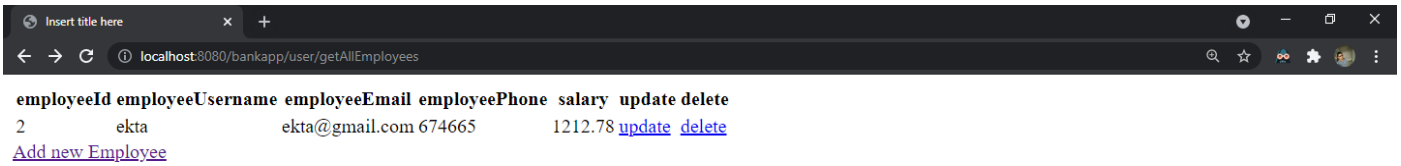
- Show all Customer
- Add new Customer
- Withdraw Money
- Deposit Money
- Transfer Money

Type here to search

9:23 PM
9/24/2021


Insert title here
localhost:8080/bankapp/transaction/pendingTransaction
transactionId fromAccountNumber toAccountNumber amount transactionType status approve reject
2 4 2 200001.0 TRANSFER PENDING approve reject
Show all Transactions

Type here to search
9:23 PM
9/24/2021


Insert title here
localhost:8080/bankapp/user/getAllEmployees
employeeId employeeUsername employeeEmail employeePhone salary update delete
2 ekta ekta@gmail.com 674665 1212.78 update delete
Add new Employee


Type here to search
9:24 PM
9/24/2021

References:

- https://stackoverflow.com/
- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
- https://spring.io/projects/spring-security
- https://docs.oracle.com/javaee/5/tutorial/doc/bnajo.html