|  |  |
| --- | --- |
| Full Name | Asmita Shinde |
| Batch | MS FSD DEC 2021 Cohort |
| Project Title | **ICIN Bank** |
| Project Submission Date | 03/09/2022 |

|  |
| --- |
| **Source Code:** |
| **User Portal - Backend** |
| package com.icinbank;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class IcinbankApplication {  public static void main(String[] args) {  SpringApplication.run(IcinbankApplication.class, args);  }  } |

|  |
| --- |
| **Model Class** |
| package com.icinbank.model;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.OneToOne;  import javax.persistence.Table;  @Entity  @Table  public class Account {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long accno;  private int balance;    private String username;    @OneToOne(fetch = FetchType.LAZY)  @JoinColumn  private User user;  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getAccno() {  return accno;  }  public void setAccno(long accno) {  this.accno = accno;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }  } |

|  |
| --- |
| package com.icinbank.model;  import java.time.LocalDate;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  @Entity  @Table  public class ChequebookRequest {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long account;  private String accType;  private boolean requestStatus;  private LocalDate date;  private int no\_of\_pages;    public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }      public long getAccount() {  return account;  }  public void setAccount(long account) {  this.account = account;  }  public String getAccType() {  return accType;  }  public void setAccType(String accType) {  this.accType = accType;  }  public boolean isRequestStatus() {  return requestStatus;  }  public void setRequestStatus(boolean requestStatus) {  this.requestStatus = requestStatus;  }  public LocalDate getDate() {  return date;  }  public void setDate(LocalDate date) {  this.date = date;  }  public int getNo\_of\_pages() {  return no\_of\_pages;  }  public void setNo\_of\_pages(int no\_of\_pages) {  this.no\_of\_pages = no\_of\_pages;  }      } |

|  |
| --- |
| package com.icinbank.model;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.OneToOne;  import javax.persistence.Table;  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  @Entity  @Table  public class Saccount {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    private long accno;    private int balance;    private String username;  @OneToOne(fetch = FetchType.LAZY)  @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})  @JoinColumn  private User user;      public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getAccno() {  return accno;  }  public void setAccno(long accno) {  this.accno = accno;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }    } |

|  |
| --- |
| package com.icinbank.model;  import java.time.LocalDate;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  @Entity  @Table  public class Transfer implements Comparable<Transfer>{  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long saccount;  private long raccount;  private int amount;  private LocalDate date;    public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }    public long getSaccount() {  return saccount;  }  public void setSaccount(long saccount) {  this.saccount = saccount;  }  public long getRaccount() {  return raccount;  }  public void setRaccount(long raccount) {  this.raccount = raccount;  }  public int getAmount() {  return amount;  }  public void setAmount(int amount) {  this.amount = amount;  }  public LocalDate getDate() {  return date;  }  public void setDate(LocalDate date) {  this.date = date;  }  @Override  public int compareTo(Transfer o) {  Integer i1=this.id;  Integer i2=o.id;  return i2.compareTo(i1);  }      } |

|  |
| --- |
| package com.icinbank.model;  import java.util.Date;  import javax.persistence.\*;  import com.fasterxml.jackson.annotation.JsonFormat;  @Entity  @Table(name="\"Users\"")  public class User {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private String fname;  private String lname;  private long phone;  private String address;  private String email;  private String username;  private String password;  @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyy-MM-dd")  private Date dob;  private String identityType;  private String identity;  @Column(columnDefinition = "boolean default false")  private boolean status;  @Column(columnDefinition = "boolean default false")  private boolean authorizationStatus;  @Column(columnDefinition = "integer default 3", nullable = false)  private int featureStatus = 3;  public String getIdentityType() {  return identityType;  }  public void setIdentityType(String identityType) {  this.identityType = identityType;  }  public boolean getStatus() {  return status;  }  public void setStatus(boolean status) {  this.status = status;  }  public boolean getAuthorizationStatus() {  return authorizationStatus;  }  public void setAuthorizationStatus(boolean authorizationStatus) {  this.authorizationStatus = authorizationStatus;  }  public int getFeatureStatus() {  return featureStatus;  }  public void setFeatureStatus(int featureStatus) {  this.featureStatus = featureStatus;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getFname() {  return fname;  }  public void setFname(String fname) {  this.fname = fname;  }  public String getLname() {  return lname;  }  public void setLname(String lname) {  this.lname = lname;  }  public long getPhone() {  return phone;  }  public void setPhone(long phone) {  this.phone = phone;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public Date getDob() {  return dob;  }  public void setDob(Date dob) {  this.dob = dob;  }  public String getIdentity() {  return identity;  }  public void setIdentity(String identity) {  this.identity = identity;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  } |

|  |
| --- |
| package com.icinbank.model;  public class UserDisplay {  String username;  private long primaryAccno;  private int primaryBalance;  private long savingsAccno;  private int savingsBalance;  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public long getPrimaryAccno() {  return primaryAccno;  }  public void setPrimaryAccno(long primaryAccno) {  this.primaryAccno = primaryAccno;  }  public int getPrimaryBalance() {  return primaryBalance;  }  public void setPrimaryBalance(int primaryBalance) {  this.primaryBalance = primaryBalance;  }  public long getSavingsAccno() {  return savingsAccno;  }  public void setSavingsAccno(long savingsAccno) {  this.savingsAccno = savingsAccno;  }  public int getSavingsBalance() {  return savingsBalance;  }  public void setSavingsBalance(int savingsBalance) {  this.savingsBalance = savingsBalance;  }  public UserDisplay(String username, long primaryAccno, int primaryBalance, long savingsAccno, int savingsBalance) {  super();  this.username = username;  this.primaryAccno = primaryAccno;  this.primaryBalance = primaryBalance;  this.savingsAccno = savingsAccno;  this.savingsBalance = savingsBalance;  }  public UserDisplay() {  // TODO Auto-generated constructor stub  }  } |

|  |
| --- |
| package com.icinbank.model;  import java.time.LocalDate;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  @Entity  @Table  public class UserHistory {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long account;  private int amount;  private String action;  private LocalDate date;  public LocalDate getDate() {  return date;  }  public void setDate(LocalDate today) {  this.date = today;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }    public long getAccount() {  return account;  }  public void setAccount(long account) {  this.account = account;  }  public int getAmount() {  return amount;  }  public void setAmount(int amount) {  this.amount = amount;  }  public String getAction() {  return action;  }  public void setAction(String action) {  this.action = action;  }    } |

|  |
| --- |
| **controller** |
| package com.icinbank.controller;  import java.util.Collections;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.ResponseEntity;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.icinbank.dao.AccountRepository;  import com.icinbank.dao.SaccountRepository;  import com.icinbank.details.TransactionDetails;  import com.icinbank.details.TransferDetails;  import com.icinbank.model.Account;  import com.icinbank.model.Saccount;  import com.icinbank.model.Transfer;  import com.icinbank.model.UserHistory;  import com.icinbank.response.DepositResponse;  import com.icinbank.response.TransferResponse;  import com.icinbank.response.WithdrawResponse;  import com.icinbank.service.AccountService;  import com.icinbank.service.SaccountService;  import com.icinbank.service.TransferHistoryService;  import com.icinbank.service.UserHistoryService;  @RestController  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  public class AccountController {  @Autowired  private AccountService accountService;  @Autowired  private SaccountService savingAccService;  @Autowired  private UserHistoryService userHistoryService;  @Autowired  private TransferHistoryService transferHistoryService;  @Autowired  private AccountRepository accountRepo;  @Autowired  private SaccountRepository savingAccountRepo;  private final String ifsc = "ICIN7465";  public static boolean isprimary(long account) {  String s = Long.toString(account).substring(0, 10);  String check = "3914918201";  if (s.equals(check)) {  return true;  } else {  return false;  }  }  @GetMapping("/account/details/{account}")  public Account getAccountDetails(@PathVariable("account") int account) {  return accountService.getAccountDetails(account);  }  @PostMapping("/account/newAccount")  public ResponseEntity<Account> newAccount(@RequestBody Account account) {  return accountService.newAccount(account.getUsername());  }  @PostMapping("/saving/newAccount")  public ResponseEntity<Saccount> newAccount(@RequestBody Saccount account) {  return savingAccService.newSavingAccount(account.getUsername());  }  @PutMapping("/account/profile")  public Account updateProfile(@RequestBody Account account) {  return accountService.updateAccount(account);  }  @GetMapping("/account/getprimary/{username}")  public Account getPrimarydetails(@PathVariable("username") String username) {  return accountService.getAccount(username);  }  @GetMapping("/account/getsaving/{username}")  public Saccount getSavingdetails(@PathVariable("username") String username) {  return savingAccService.getAccount(username);  }  @PostMapping("/account/deposit")  public DepositResponse deposit(@RequestBody TransactionDetails details) {  // adao.findByUsername(adao.findByAccno(details.getAccount()).getUsername());  if (isprimary(details.getAccount())) {  return accountService.deposit(details.getAccount(), details.getAmount());  } else {  return savingAccService.deposit(details.getAccount(), details.getAmount());  }  }  @PostMapping("/account/withdraw")  public WithdrawResponse withdraw(@RequestBody TransactionDetails details) {  if (isprimary(details.getAccount())) {  return accountService.withdraw(details.getAccount(), details.getAmount());  } else {  return savingAccService.withdraw(details.getAccount(), details.getAmount());  }  }  @PostMapping("/account/transfer")  public TransferResponse transfer(@RequestBody TransferDetails details) {  try {    System.out.println("Transfer Starts");  System.out.println(details.getIfsc());  System.out.println(ifsc);  System.out.println(details.getIfsc().equals(ifsc));    System.out.println("Transfer Ends");    if (details.getIfsc().equals(ifsc)) {  Account primary = accountRepo.findByUsername(details.getUsername());  Saccount saving = savingAccountRepo.findByUsername(details.getUsername());    System.out.println(primary.getAccno());  System.out.println(details.getSavingAccount());  System.out.println(saving.getAccno());  if (primary.getAccno() == details.getSavingAccount() || saving.getAccno() == details.getSavingAccount()) {  // String len = Integer.toString(details.getSaccount());  if (isprimary(details.getSavingAccount())) {  return accountService.transfer(details.getSavingAccount(), details.getPrimaryAccount(),details.getAmount());  } else {  return savingAccService.transfer(details.getSavingAccount(), details.getPrimaryAccount(),details.getAmount());  }  } else {  TransferResponse response = new TransferResponse();  response.setSaccount(details.getSavingAccount());  response.setResponseMessage("Dear user You can only transfer funds from the accounts registered with you");  response.setTransferStatus(false);  return response;  }  } else {  TransferResponse response = new TransferResponse();  response.setSaccount(details.getSavingAccount());  response.setResponseMessage("IFSC code is incorrect");  response.setTransferStatus(false);  return response;  }  } catch (Exception e) {  TransferResponse response = new TransferResponse();  response.setSaccount(details.getSavingAccount());  response.setResponseMessage("Please provide an IFSC code");  response.setTransferStatus(false);  return response;  }  }  @GetMapping("/account/getHistory/{account}")  public List<UserHistory> getHistory(@PathVariable("account") long account) {  List<UserHistory> history = userHistoryService.getHistory(account);  Collections.reverse(history);  return history;  }  @GetMapping("/account/getTransfers/{account}")  public List<Transfer> getTransfers(@PathVariable("account") long account) {  return transferHistoryService.getTransfers(account);  }  } |

|  |
| --- |
| package com.icinbank.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.icinbank.model.ChequebookRequest;  import com.icinbank.response.ChequeResponse;  import com.icinbank.service.ChequebookService;  @RestController  //@CrossOrigin(origins = "http://localhost:4200")  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  public class ChequeBookController {  @Autowired  private ChequebookService service;  @PostMapping("/cheque/request")  public ChequeResponse createrequest(@RequestBody ChequebookRequest chequebook) {  return service.createrequest(chequebook);  }  @GetMapping("/cheque/getbyAccount/{account}")  public List<ChequebookRequest> getRequests(@PathVariable("account") long account) {  List<ChequebookRequest> list=service.getRequests(account);  //Collections.reverse(list);  //return service.getRequests(account);  return list;  }  } |

|  |
| --- |
| package com.icinbank.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.icinbank.details.LoginDetails;  import com.icinbank.response.LoginResponse;  import com.icinbank.service.impl.LoginServiceImpl;  @RestController  //@CrossOrigin(origins = "http://localhost:4200")  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  public class LoginController {    @Autowired  LoginServiceImpl service;    @PostMapping("/login")  public LoginResponse userLogin(@RequestBody LoginDetails details) {    return service.customerLogin(details);    }  } |

|  |
| --- |
| package com.icinbank.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.icinbank.details.UpdateDetails;  import com.icinbank.model.User;  import com.icinbank.model.UserDisplay;  import com.icinbank.response.UpdateResponse;  import com.icinbank.service.ProfileService;  @RestController  //@CrossOrigin(origins = "http://localhost:4200")  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  public class ProfileController {  @Autowired  private ProfileService profileService;  @PutMapping("/profile/update")  public UpdateResponse updateUser(@RequestBody UpdateDetails user) {  // TODO Auto-generated method stub  return profileService.updateUser(user);  }  @GetMapping("/profile/{username}")  public User getUser(@PathVariable("username") String username) {  // TODO Auto-generated method stub  return profileService.getUser(username);  }  @GetMapping("home/{username}")  public UserDisplay userDisplay(@PathVariable("username") String username) {  final UserDisplay currentUser = profileService.userDisplay(username);  System.out.println("Printing my data");  System.out.println(currentUser);  return currentUser;  }  } |

|  |
| --- |
| package com.icinbank.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.icinbank.model.User;  import com.icinbank.response.RegisterResponse;  import com.icinbank.service.RegistrationService;  @RestController  //@CrossOrigin(origins = "http://localhost:4200")  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  public class RegistrationController {    @Autowired  private RegistrationService service;  @PostMapping("/register")  public RegisterResponse createUser(@RequestBody User user) {  return service.createAccount(user);  }  } |

|  |
| --- |
| **DAO Class** |
| package com.icinbank.dao;  import org.springframework.data.repository.CrudRepository;  import org.springframework.stereotype.Repository;  import com.icinbank.model.Account;  @Repository  public interface AccountRepository extends CrudRepository<Account, Integer> {  public Account findByUsername(String username);  public Account findByAccno(long accno);  } |

|  |
| --- |
| package com.icinbank.dao;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import com.icinbank.model.ChequebookRequest;  public interface ChequeBookRepository extends JpaRepository<ChequebookRequest, Integer>{  public List<ChequebookRequest> findByAccount(long account);  } |

|  |
| --- |
| package com.icinbank.dao;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.icinbank.model.Saccount;  @Repository  public interface SaccountRepository extends JpaRepository<Saccount, Integer>{    public Saccount findByAccno(long accNo);  public Saccount findByUsername(String username);  } |

|  |
| --- |
| **package** com.icinbank.dao;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** org.springframework.data.jpa.repository.Query;  **import** org.springframework.stereotype.Repository;  **import** com.icinbank.model.User;  **import** com.icinbank.model.UserDisplay;  @Repository  **public** **interface** UserDisplayRepository **extends** JpaRepository<User, Integer> {  @Query("SELECT new com.icinbank.model.UserDisplay(u.username,a.accno,a.balance,s.accno,s.balance)"  + "FROM User u ,Account a,Saccount s WHERE u.username=?1 and u.username=a.username and u.username=s.username")  **public** UserDisplay getCurrentUser(String username);  } |

|  |
| --- |
| package com.icinbank.dao;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.icinbank.model.Transfer;  @Repository  public interface TransferHistoryRepository extends JpaRepository<Transfer, Integer> {  public List<Transfer>findBySaccount(long saccount);  public List<Transfer>findByRaccount(long raccount);  } |

|  |
| --- |
| package com.icinbank.dao;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.icinbank.model.UserHistory;  @Repository  public interface UserHistoryRepository extends JpaRepository<UserHistory, Integer>{  public List<UserHistory> findByAccount(long account);  } |

|  |
| --- |
| package com.icinbank.dao;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.icinbank.model.User;  @Repository  public interface UserRepository extends JpaRepository<User, Integer> {  public User findByUsername(String username);  public User findByEmail(String email);  // public User findByPancard(String pan);  public User findByPhone(long l);  } |

|  |
| --- |
| **Details Class** |
| **package** com.icinbank.details;  **public** **class** LoginDetails {  **private** String username;  **private** String password;  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  } |

|  |
| --- |
| **package** com.icinbank.details;  **public** **class** TransactionDetails {  **private** **long** account;  **private** **int** amount;  **public** **long** getAccount() {  **return** account;  }  **public** **void** setAccount(**long** account) {  **this**.account = account;  }  **public** **int** getAmount() {  **return** amount;  }  **public** **void** setAmount(**int** amount) {  **this**.amount = amount;  }  } |

|  |
| --- |
| **package** com.icinbank.details;  **public** **class** TransferDetails {  **private** **long** savingAccount;  **private** **long** primaryAccount;  **private** **int** amount;  **private** String Username;  **private** String ifsc;  **public** **long** getSavingAccount() {  **return** savingAccount;  }  **public** **void** setSavingAccount(**long** savingAccount) {  **this**.savingAccount = savingAccount;  }  **public** **long** getPrimaryAccount() {  **return** primaryAccount;  }  **public** **void** setPrimaryAccount(**long** primaryAccount) {  **this**.primaryAccount = primaryAccount;  }  **public** **int** getAmount() {  **return** amount;  }  **public** **void** setAmount(**int** amount) {  **this**.amount = amount;  }  **public** String getUsername() {  **return** Username;  }  **public** **void** setUsername(String username) {  Username = username;  }  **public** String getIfsc() {  **return** ifsc;  }  **public** **void** setIfsc(String ifsc) {  **this**.ifsc = ifsc;  }  } |

|  |
| --- |
| **package** com.icinbank.details;  **public** **class** UpdateDetails {  **private** String username;  **private** String password;  **private** String newpassword;  **private** **long** phone;  **private** String address;  **private** String email;  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getNewpassword() {  **return** newpassword;  }  **public** **void** setNewpassword(String newpassword) {  **this**.newpassword = newpassword;  }  **public** **long** getPhone() {  **return** phone;  }  **public** **void** setPhone(**long** phone) {  **this**.phone = phone;  }  **public** String getAddress() {  **return** address;  }  **public** **void** setAddress(String address) {  **this**.address = address;  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  } |
| **Response Class** |
| **package** com.icinbank.response;  **public** **class** AccountResponse {  **private** String responseMessage;  **public** String getResponseMessage() {  **return** responseMessage;  }  } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** ChequeResponse {    **private** **boolean** status;  **private** String responseMessage;  **private** **long** account;  **public** **boolean** isStatus() {  **return** status;  }  **public** **void** setStatus(**boolean** status) {  **this**.status = status;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** **long** getAccount() {  **return** account;  }  **public** **void** setAccount(**long** account) {  **this**.account = account;  }      } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** DepositResponse {  **private** **boolean** depositStatus;  **private** String responseMessage;  **private** **long** account;  **public** **boolean** isDepositStatus() {  **return** depositStatus;  }  **public** **void** setDepositStatus(**boolean** depositStatus) {  **this**.depositStatus = depositStatus;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** **long** getAccount() {  **return** account;  }  **public** **void** setAccount(**long** account) {  **this**.account = account;  }  } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** LoginResponse {  **private** **boolean** loginStatus;  **private** String responseMessage;  **private** String username;    **public** **boolean** getLoginStatus() {  **return** loginStatus;  }  **public** **void** setLoginStatus(**boolean** loginStatus) {  **this**.loginStatus = loginStatus;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** RegisterResponse {  **private** **boolean** registrationStatus;  **private** String responseMessage;  **private** String username;  **public** **boolean** isRegistrationStatus() {  **return** registrationStatus;  }  **public** **void** setRegistrationStatus(**boolean** registrationStatus) {  **this**.registrationStatus = registrationStatus;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** TransferResponse {  **private** **boolean** transferStatus;  **private** String responseMessage;  **private** **long** saccount;    **public** **boolean** isTransferStatus() {  **return** transferStatus;  }  **public** **void** setTransferStatus(**boolean** transferStatus) {  **this**.transferStatus = transferStatus;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** **long** getSaccount() {  **return** saccount;  }  **public** **void** setSaccount(**long** saccount) {  **this**.saccount = saccount;  }      } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** UpdateResponse {    **public** **boolean** flag;  **public** String message;  **public** **boolean** isFlag() {  **return** flag;  }  **public** **void** setFlag(**boolean** flag) {  **this**.flag = flag;  }  **public** String getMessage() {  **return** message;  }  **public** **void** setMessage(String message) {  **this**.message = message;  }      } |

|  |
| --- |
| **package** com.icinbank.response;  **public** **class** WithdrawResponse {  **private** **boolean** withdrawStatus;  **private** String responseMessage;  **private** **long** account;  **public** **boolean** isWithdrawStatus() {  **return** withdrawStatus;  }  **public** **void** setWithdrawStatus(**boolean** withdrawStatus) {  **this**.withdrawStatus = withdrawStatus;  }  **public** String getResponseMessage() {  **return** responseMessage;  }  **public** **void** setResponseMessage(String responseMessage) {  **this**.responseMessage = responseMessage;  }  **public** **long** getAccount() {  **return** account;  }  **public** **void** setAccount(**long** account) {  **this**.account = account;  }  } |

|  |
| --- |
| **Service Class** |
| package com.icinbank.service;  import org.springframework.http.ResponseEntity;  import com.icinbank.model.Account;  import com.icinbank.response.DepositResponse;  import com.icinbank.response.TransferResponse;  import com.icinbank.response.WithdrawResponse;  public interface AccountService {  public ResponseEntity<Account> newAccount(String username);  public Account getAccount(String username);  public DepositResponse deposit(long acc,int amount);  public WithdrawResponse withdraw(long acc,int amount);  public TransferResponse transfer(long saccount,long raccount,int amount);  public Account getAccountDetails(long account);  public Account updateAccount(Account account);    } |

|  |
| --- |
| package com.icinbank.service;  import java.util.List;  import com.icinbank.model.ChequebookRequest;  import com.icinbank.response.ChequeResponse;  public interface ChequebookService {  public ChequeResponse createrequest(ChequebookRequest chequebook);  public List<ChequebookRequest> getRequests(long account);  } |

|  |
| --- |
| package com.icinbank.service;  import com.icinbank.details.LoginDetails;  import com.icinbank.response.LoginResponse;  public interface LoginService {  public LoginResponse customerLogin(LoginDetails details);  } |

|  |
| --- |
| package com.icinbank.service;  import com.icinbank.details.UpdateDetails;  import com.icinbank.model.User;  import com.icinbank.model.UserDisplay;  import com.icinbank.response.UpdateResponse;  public interface ProfileService {  public UpdateResponse updateUser(UpdateDetails user);  public User getUser(String username);  public UserDisplay userDisplay(String username);  } |

|  |
| --- |
| package com.icinbank.service;  import com.icinbank.model.User;  import com.icinbank.response.RegisterResponse;  public interface RegistrationService {  public RegisterResponse createAccount(User details);  public boolean usernameAlreadyExists(String username);  public boolean EmailAlreadyExists(String email);  public boolean PhoneAlreadyExists(long l);  } |

|  |
| --- |
| package com.icinbank.service;  import org.springframework.http.ResponseEntity;  import com.icinbank.model.Saccount;  import com.icinbank.response.DepositResponse;  import com.icinbank.response.TransferResponse;  import com.icinbank.response.WithdrawResponse;  public interface SaccountService {  public Saccount getAccountDetails(long account);  public Saccount getAccount(String username);  public ResponseEntity<Saccount> newSavingAccount(String username);  public DepositResponse deposit(long acc,int amount);  public WithdrawResponse withdraw(long acc,int amount);  public TransferResponse transfer(long saccount,long raccount,int amount);  } |

|  |
| --- |
| package com.icinbank.service;  import java.util.List;  import com.icinbank.model.Transfer;  public interface TransferHistoryService {  public Transfer addAction(long saccount, long raccount, int amount);  public List<Transfer> getTransfers(long account);  } |

|  |
| --- |
| package com.icinbank.service;  import java.util.List;  import com.icinbank.model.UserHistory;  public interface UserHistoryService {  public UserHistory addAction(long account,int amount,int balance,String action);  public List<UserHistory> getHistory(long account);  } |

|  |
| --- |
| **Service Implement Class** |
| package com.icinbank.service.impl;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpStatus;  import org.springframework.http.ResponseEntity;  import org.springframework.stereotype.Service;  import com.icinbank.dao.AccountRepository;  import com.icinbank.dao.SaccountRepository;  import com.icinbank.dao.UserRepository;  import com.icinbank.model.Account;  import com.icinbank.model.Saccount;  import com.icinbank.model.User;  import com.icinbank.response.DepositResponse;  import com.icinbank.response.TransferResponse;  import com.icinbank.response.WithdrawResponse;  import com.icinbank.service.AccountService;  import com.icinbank.service.TransferHistoryService;  import com.icinbank.service.UserHistoryService;  @Service  public class AccountServiceImpl implements AccountService {  @Autowired  private AccountRepository accountRepo;  @Autowired  private UserHistoryService userHistory;  @Autowired  private TransferHistoryService transferHistory;  @Autowired  private UserRepository userRepo;  @Autowired  private SaccountRepository savingAccRepo;  private final String bankCode = "3914";  private final String countryCode = "91";  private final String branchCode = "820";  private final String accountcode = "1";  public long generate\_saving(int userId) {  String accNo = bankCode + countryCode + branchCode + accountcode + String.valueOf(userId);  return Long.parseLong(accNo);  }  public static boolean isprimary(long account) {  String s = Long.toString(account).substring(0, 10);  String check = "3914918201";  if (s.equals(check)) {  return true;  } else {  return false;  }  }  @Override  public ResponseEntity<Account> newAccount(String username) {  final long accNo;  final Account checkAccount;  User getUser;  Account newAccount = new Account();  try {  getUser = userRepo.findByUsername(username);  accNo = generate\_saving(getUser.getId());  checkAccount = this.getAccountDetails(accNo);  if (checkAccount == null) {  newAccount.setUsername(username);  newAccount.setUser(getUser);  newAccount.setAccno(accNo);  accountRepo.save(newAccount);  return new ResponseEntity<>(newAccount, HttpStatus.CREATED);  } else {  if (checkAccount.getAccno() == accNo) {  return new ResponseEntity<>(checkAccount, HttpStatus.ALREADY\_REPORTED);  }  return new ResponseEntity<>(checkAccount, HttpStatus.ALREADY\_REPORTED);  }  } catch (Exception e) {  return new ResponseEntity<>(null, HttpStatus.INTERNAL\_SERVER\_ERROR);  }  }  @Override  public Account getAccount(String username) {  // TODO Auto-generated method stub  return accountRepo.findByUsername(username);  }  @Override  public DepositResponse deposit(long acc, int amount) {  DepositResponse response = new DepositResponse();  boolean flag = true;  try {  Account account = accountRepo.findByAccno(acc);  account.setBalance(account.getBalance() + amount);  userHistory.addAction(acc, amount, account.getBalance(), "credit");  accountRepo.save(account);  response.setResponseMessage("Rs." + amount + " successfully deposited into your account balance is now Rs."  + account.getBalance());  response.setDepositStatus(flag);  } catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setDepositStatus(flag);  }  response.setAccount(acc);  return response;  }  @Override  public WithdrawResponse withdraw(long acc, int amount) {  WithdrawResponse response = new WithdrawResponse();  boolean flag = true;  try {  Account account = accountRepo.findByAccno(acc);  User user = userRepo.findByUsername(account.getUsername());  if (user.getFeatureStatus() == 2 || user.getFeatureStatus() == 3) {  if (account.getBalance() >= amount) {  account.setBalance(account.getBalance() - amount);  userHistory.addAction(acc, amount, account.getBalance(), "debit");  accountRepo.save(account);  response.setResponseMessage("Rs." + amount  + " successfully withdrawn your account balance is now Rs." + account.getBalance());  response.setWithdrawStatus(flag);  } else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transaction");  response.setWithdrawStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("This function is not available for your account");  response.setWithdrawStatus(flag);  }  }  catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setWithdrawStatus(flag);  }  response.setAccount(acc);  return response;  }  @Override  public TransferResponse transfer(long saccount, long raccount, int amount) {  TransferResponse response = new TransferResponse();  boolean flag = true;  try {  Account senderAccount = accountRepo.findByAccno(saccount);  // String check = Long.toString(raccount);  if (isprimary(raccount)) {  Account receiverAccount = accountRepo.findByAccno(raccount);  if (senderAccount.getAccno() != receiverAccount.getAccno()) {  if (senderAccount.getBalance() > amount) {  User user = userRepo.findByUsername(senderAccount.getUsername());  if (user.getFeatureStatus() == 3) {  senderAccount.setBalance(senderAccount.getBalance() - amount);  receiverAccount.setBalance(receiverAccount.getBalance() + amount);  transferHistory.addAction(saccount, raccount, amount);  accountRepo.save(senderAccount);  accountRepo.save(receiverAccount);  response.setResponseMessage("Rs." + amount + " successfully transferred to account "  + receiverAccount.getAccno());  response.setTransferStatus(flag);  } else {  flag = false;  response.setResponseMessage("This feature is not available for your account");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transfer");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("sender and recieiver accounts are same");  response.setTransferStatus(flag);  }  } else {  Saccount receiverAccount = savingAccRepo.findByAccno(raccount);  if (senderAccount.getAccno() != receiverAccount.getAccno()) {  if (senderAccount.getBalance() > amount) {  User user = userRepo.findByUsername(senderAccount.getUsername());  if (user.getFeatureStatus() == 3) {  senderAccount.setBalance(senderAccount.getBalance() - amount);  receiverAccount.setBalance(receiverAccount.getBalance() + amount);  transferHistory.addAction(saccount, raccount, amount);  accountRepo.save(senderAccount);  savingAccRepo.save(receiverAccount);  response.setResponseMessage("Rs." + amount + " successfully transferred to account "  + receiverAccount.getAccno());  response.setTransferStatus(flag);  } else {  flag = false;  response.setResponseMessage("This function isnt available for the account");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transfer");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("sender and recieiver accounts are same");  response.setTransferStatus(flag);  }  }  }  catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setTransferStatus(flag);  }  response.setSaccount(saccount);  return response;  }  @Override  public Account getAccountDetails(long account) {  // TODO Auto-generated method stub  return accountRepo.findByAccno(account);  }  @Override  public Account updateAccount(Account account) {  // TODO Auto-generated method stub  return accountRepo.save(account);  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import java.time.LocalDate;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.icinbank.dao.AccountRepository;  import com.icinbank.dao.ChequeBookRepository;  import com.icinbank.dao.SaccountRepository;  import com.icinbank.model.Account;  import com.icinbank.model.ChequebookRequest;  import com.icinbank.model.Saccount;  import com.icinbank.response.ChequeResponse;  import com.icinbank.service.ChequebookService;  @Service  public class ChequebookServiceimpl implements ChequebookService {  @Autowired  private ChequeBookRepository dao;  @Autowired  private AccountRepository adao;  @Autowired  private SaccountRepository sdao;  @Override  public ChequeResponse createrequest(ChequebookRequest chequebook) {  ChequeResponse response = new ChequeResponse();  long account = chequebook.getAccount();  List<ChequebookRequest> prevRequests = dao.findByAccount(account);  if (!prevRequests.isEmpty()) {  for (int i = 0; i < prevRequests.size(); i++) {  if (prevRequests.get(i).isRequestStatus() == false) {  response.setResponseMessage("Your previous chequebook request is still pending.");  response.setStatus(false);  response.setAccount(account);  return response;  }  }  }  LocalDate today = LocalDate.now();  if (isprimary(account)) {  try {  Account account1 = adao.findByAccno(account);  response.setAccount(account1.getAccno());  response.setStatus(true);  response.setResponseMessage("Request submitted successfully");  chequebook.setAccType("Primary");  chequebook.setDate(today);  chequebook.setRequestStatus(false);  // dao.save(chequebook);  dao.save(chequebook);  } catch (Exception e) {  response.setAccount(account);  response.setStatus(false);  response.setResponseMessage("account number is incorrect");  }  } else {  if (isSecondary(account)) {  try {  Saccount saccount = sdao.findByAccno(account);  response.setAccount(saccount.getAccno());  response.setStatus(true);  response.setResponseMessage("Request submitted successfully");  chequebook.setRequestStatus(false);  chequebook.setAccType("Secondary");  chequebook.setDate(today);  dao.save(chequebook);  } catch (Exception e) {  response.setAccount(account);  response.setStatus(false);  response.setResponseMessage("account number is incorrect");  }  } else {  response.setAccount(account);  response.setStatus(false);  response.setResponseMessage("account number is incorrect");  }  }  return response;  }  @Override  public List<ChequebookRequest> getRequests(long account) {  return dao.findByAccount(account);  }  public static boolean isprimary(long account) {  String s = Long.toString(account).substring(0, 10);  String check = "3914918201";  if (s.equals(check)) {  return true;  } else {  return false;  }  }  public static boolean isSecondary(long account) {  String s = Long.toString(account).substring(0, 10);  String check = "3914918202";  if (s.equals(check)) {  return true;  } else {  return false;  }  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.apache.commons.codec.digest.DigestUtils;  import com.icinbank.dao.UserRepository;  import com.icinbank.details.LoginDetails;  import com.icinbank.model.User;  import com.icinbank.response.LoginResponse;  import com.icinbank.service.LoginService;  @Service  public class LoginServiceImpl implements LoginService {  @Autowired  private UserRepository dao;  @Override  public LoginResponse customerLogin(LoginDetails login) {  LoginResponse response = new LoginResponse();  boolean flag = true;  String message = "Login succesfull";  User user = null;  String hashedPassword = DigestUtils.sha256Hex(login.getPassword());  try {  user = dao.findByUsername(login.getUsername());  if (!user.getStatus()) {  flag = false;  message = "Dear Mr/Ms." + user.getFname() + " your account has been blocked for security reasons.";  }  if (!user.getAuthorizationStatus()) {  flag = false;  message = "Dear Mr/Ms." + user.getFname() + " your account has not been activated yet";  }  if (!hashedPassword.equals(user.getPassword())) {  flag = false;  message = "Username or password is incorrect";  }  } catch (Exception e) {  flag = false;  message = "Username or password is incorrect";  }  response.setLoginStatus(flag);  response.setResponseMessage(message);  try {  response.setUsername(user.getUsername());  } catch (Exception e) {  response.setUsername(login.getUsername());  }  return response;  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import org.apache.commons.codec.digest.DigestUtils;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.icinbank.dao.UserDisplayRepository;  import com.icinbank.dao.UserRepository;  import com.icinbank.details.UpdateDetails;  import com.icinbank.model.User;  import com.icinbank.model.UserDisplay;  import com.icinbank.response.UpdateResponse;  import com.icinbank.service.ProfileService;  @Service  public class ProfileServiceImpl implements ProfileService {  @Autowired  private UserRepository userRepo;  @Autowired  private UserDisplayRepository userDisplayRepo;  @Override  public UpdateResponse updateUser(UpdateDetails user) {  boolean flag = true;  UpdateResponse response = new UpdateResponse();  String message = "Update successful";  try {  int counter = 0;  User u = userRepo.findByUsername(user.getUsername());  if (user.getAddress().length() != 0) {  counter++;  u.setAddress(user.getAddress());  }  if (user.getPassword().length() != 0 && user.getNewpassword().length() != 0) {  counter++;  String hashedPassword = DigestUtils.sha256Hex(user.getPassword());  u.setPassword(hashedPassword);  }  if (user.getEmail().length() != 0) {  counter++;  u.setEmail(user.getEmail());  }  if (user.getPhone() != 0) {  counter++;  u.setPhone(user.getPhone());  }  System.out.println(counter);  if (counter > 0) {  userRepo.save(u);  } else {  flag = false;  message = "Please enter some information to update";  }  } catch (Exception e) {  flag = false;  response.setMessage("Update unsuccesful");  }  response.setMessage(message);  response.setFlag(flag);  return response;  }  @Override  public User getUser(String username) {  // TODO Auto-generated method stub  return userRepo.findByUsername(username);  }  @Override  public UserDisplay userDisplay(String username) {  UserDisplay user = userDisplayRepo.getCurrentUser(username);  return user;  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import org.apache.commons.codec.digest.DigestUtils;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.icinbank.dao.UserRepository;  import com.icinbank.model.User;  import com.icinbank.response.RegisterResponse;  import com.icinbank.service.AccountService;  import com.icinbank.service.RegistrationService;  import com.icinbank.service.SaccountService;  @Service  public class RegistrationServiceImpl implements RegistrationService {  @Autowired  private UserRepository dao;  @Autowired  private AccountService service;  @Autowired  private SaccountService sservice;  @Override  public RegisterResponse createAccount(User user) {  RegisterResponse response = new RegisterResponse();  boolean flag = true;  String message = "Registration Succesful";  if (EmailAlreadyExists(user.getEmail())) {  message = "Email already Exists";  flag = false;  }  if (PhoneAlreadyExists(user.getPhone())) {  message = "Phone number already Exists";  flag = false;  }  if (usernameAlreadyExists(user.getUsername())) {  message = "Username already Exists";  flag = false;  }  if (flag) {  String hashedPassword = DigestUtils.sha256Hex(user.getPassword());  user.setPassword(hashedPassword);  dao.save(user);  // User u=dao.findByUsername(user.getUsername());  // int id=u.getId();  // service.newAccount(user.getUsername(),id);  // sservice.newAccount(user.getUsername(),id);  }  response.setRegistrationStatus(flag);  response.setResponseMessage(message);  response.setUsername(user.getUsername());  return response;  }  @Override  public boolean usernameAlreadyExists(String username) {  try {  User u = dao.findByUsername(username);  System.out.println(u.toString());  return true;  } catch (Exception e) {  }  return false;  }  @Override  public boolean EmailAlreadyExists(String email) {  try {  User u = dao.findByEmail(email);  System.out.println(u.toString());  return true;  } catch (Exception e) {  }  return false;  }  // @Override  // public boolean PanAlreadyExists(String pan) {  // try {  // User u=dao.findByPan(pan);  // System.out.println(u.toString());  // return true;  // } catch (Exception e) {  // }  // return false;  // }  @Override  public boolean PhoneAlreadyExists(long l) {  try {  User u = dao.findByPhone(l);  System.out.println(u.toString());  return true;  } catch (Exception e) {  }  return false;  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpStatus;  import org.springframework.http.ResponseEntity;  import org.springframework.stereotype.Service;  import com.icinbank.dao.AccountRepository;  import com.icinbank.dao.SaccountRepository;  import com.icinbank.dao.UserRepository;  import com.icinbank.model.Account;  import com.icinbank.model.Saccount;  import com.icinbank.model.User;  import com.icinbank.response.DepositResponse;  import com.icinbank.response.TransferResponse;  import com.icinbank.response.WithdrawResponse;  import com.icinbank.service.SaccountService;  import com.icinbank.service.TransferHistoryService;  import com.icinbank.service.UserHistoryService;  @Service  public class SaccountImpl implements SaccountService {  @Autowired  private SaccountRepository savingAccRepo;  @Autowired  private UserHistoryService userHistory;  @Autowired  private TransferHistoryService transferHistory;  @Autowired  private AccountRepository accountRepo;  @Autowired  private UserRepository userRepo;  private final String bankCode = "3914";  private final String countryCode = "91";  private final String branchCode = "820";  private final String accountcode = "2";  @Override  public Saccount getAccount(String username) {  return savingAccRepo.findByUsername(username);  }  @Override  public Saccount getAccountDetails(long account) {  // TODO Auto-generated method stub  return savingAccRepo.findByAccno(account);  }  public long generate\_saving(int userId) {  String accNo = bankCode + countryCode + branchCode + accountcode + String.valueOf(userId);  return Long.parseLong(accNo);  }  @Override  public ResponseEntity<Saccount> newSavingAccount(String username) {  final long accNo;  final Saccount checkAccount;  User getUser;  Saccount newAccount = new Saccount();  try {  getUser = userRepo.findByUsername(username);  accNo = generate\_saving(getUser.getId());  checkAccount = this.getAccountDetails(accNo);  if (checkAccount == null) {  newAccount.setUsername(username);  newAccount.setUser(getUser);  newAccount.setAccno(accNo);  savingAccRepo.save(newAccount);  return new ResponseEntity<>(newAccount, HttpStatus.CREATED);  } else {  if (checkAccount.getAccno() == accNo) {  return new ResponseEntity<>(checkAccount, HttpStatus.ALREADY\_REPORTED);  }  return new ResponseEntity<>(checkAccount, HttpStatus.ALREADY\_REPORTED);  }  } catch (Exception e) {  return new ResponseEntity<>(null, HttpStatus.INTERNAL\_SERVER\_ERROR);  }  }  @Override  public DepositResponse deposit(long acc, int amount) {  DepositResponse response = new DepositResponse();  boolean flag = true;  try {  Saccount account = savingAccRepo.findByAccno(acc);  account.setBalance(account.getBalance() + amount);  userHistory.addAction(acc, amount, account.getBalance(), "deposit");  savingAccRepo.save(account);  response.setResponseMessage("Rs." + amount + " successfully deposited into your account balance is now Rs."  + account.getBalance());  response.setDepositStatus(flag);  } catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setDepositStatus(flag);  }  response.setAccount(acc);  return response;  }  @Override  public WithdrawResponse withdraw(long acc, int amount) {  WithdrawResponse response = new WithdrawResponse();  boolean flag = true;  try {  Saccount account = savingAccRepo.findByAccno(acc);  User user = userRepo.findByUsername(account.getUsername());  if (user.getFeatureStatus() == 2 || user.getFeatureStatus() == 3) {  if (account.getBalance() >= amount) {  account.setBalance(account.getBalance() - amount);  userHistory.addAction(acc, amount, account.getBalance(), "withdraw");  savingAccRepo.save(account);  response.setResponseMessage("Rs." + amount  + " successfully withdrawn your account balance is now Rs." + account.getBalance());  response.setWithdrawStatus(flag);  } else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transaction");  response.setWithdrawStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("This function is not available for your account");  response.setWithdrawStatus(flag);  }  } catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setWithdrawStatus(flag);  }  response.setAccount(acc);  return response;  }  @Override  public TransferResponse transfer(long saccount, long raccount, int amount) {  TransferResponse response = new TransferResponse();  boolean flag = true;  try {  Saccount senderAccount = savingAccRepo.findByAccno(saccount);  if (isprimary(raccount)) {  Account receiverAccount = accountRepo.findByAccno(raccount);  if (senderAccount.getAccno() != receiverAccount.getAccno()) {  if (senderAccount.getBalance() >= amount) {  User user = userRepo.findByUsername(senderAccount.getUsername());  if (user.getFeatureStatus() == 3) {  senderAccount.setBalance(senderAccount.getBalance() - amount);  receiverAccount.setBalance(receiverAccount.getBalance() + amount);  transferHistory.addAction(saccount, raccount, amount);  savingAccRepo.save(senderAccount);  accountRepo.save(receiverAccount);  response.setResponseMessage("Rs." + amount + " successfully transferred to account "  + receiverAccount.getAccno());  response.setTransferStatus(flag);  } else {  flag = false;  response.setResponseMessage("This feature is not available for your account");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transfer");  response.setTransferStatus(flag);  }  } else {  flag = false;  response.setResponseMessage("sender and recieiver accounts are same");  response.setTransferStatus(flag);  }  } else {  Saccount receiverAccount = savingAccRepo.findByAccno(raccount);  if (senderAccount.getAccno() != receiverAccount.getAccno()) {  if (senderAccount.getBalance() > amount) {  User user = userRepo.findByUsername(senderAccount.getUsername());  if (user.getFeatureStatus() == 3) {  senderAccount.setBalance(senderAccount.getBalance() - amount);  receiverAccount.setBalance(receiverAccount.getBalance() + amount);  transferHistory.addAction(saccount, raccount, amount);  savingAccRepo.save(senderAccount);  savingAccRepo.save(receiverAccount);  response.setResponseMessage("Rs." + amount + " successfully transferred to account "  + receiverAccount.getAccno());  response.setTransferStatus(flag);  } else {  flag = false;  response.setResponseMessage("This function isnt available for the account");  response.setTransferStatus(flag);  }  }  else {  flag = false;  response.setResponseMessage("Insufficient funds to complete the transfer");  response.setTransferStatus(flag);  }  }  else {  flag = false;  response.setResponseMessage("sender and recieiver accounts are same");  response.setTransferStatus(flag);  }  }  } catch (Exception e) {  flag = false;  response.setResponseMessage("Account number is incorrect");  response.setTransferStatus(flag);  }  response.setSaccount(saccount);  return response;  }  public static boolean isprimary(long account) {  String s = Long.toString(account).substring(0, 10);  String check = "3914918201";  if (s.equals(check)) {  return true;  } else {  return false;  }  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import java.time.LocalDate;  import java.util.ArrayList;  import java.util.Collections;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.icinbank.dao.TransferHistoryRepository;  import com.icinbank.model.Transfer;  import com.icinbank.service.TransferHistoryService;  @Service  public class TransferHistoryServiceImpl implements TransferHistoryService{  @Autowired  private TransferHistoryRepository dao;    @Override  public Transfer addAction(long saccount, long raccount, int amount) {  LocalDate today = LocalDate.now();  Transfer transfer=new Transfer();  transfer.setSaccount(saccount);  transfer.setRaccount(raccount);  transfer.setAmount(amount);  transfer.setDate(today);  return dao.save(transfer);  }  @Override  public List<Transfer> getTransfers(long account) {  List<Transfer> sender=dao.findBySaccount(account);  List<Transfer> receiver=dao.findByRaccount(account);  List<Transfer> merged=new ArrayList<>();  merged.addAll(sender);  merged.addAll(receiver);  Collections.sort(merged);  return merged;  }  } |

|  |
| --- |
| package com.icinbank.service.impl;  import java.time.LocalDate;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.icinbank.dao.UserHistoryRepository;  import com.icinbank.model.UserHistory;  import com.icinbank.service.UserHistoryService;  @Service  public class UserHistoryServiceImpl implements UserHistoryService{  @Autowired  private UserHistoryRepository dao;    @Override  public UserHistory addAction(long account, int amount, int balance, String action) {  LocalDate today = LocalDate.now();    UserHistory row=new UserHistory();  row.setAccount(account);  row.setAction(action);  row.setAmount(amount);  row.setDate(today);  return dao.save(row);  }  @Override  public List<UserHistory> getHistory(long account) {  // TODO Auto-generated method stub  return dao.findByAccount(account);  }  } |

|  |
| --- |
| **Application.properties** |
| spring.datasource.url=jdbc:postgresql://localhost:5432/icin  spring.datasource.username=postgres  spring.datasource.password=pravin@5  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true  spring.jpa.properties.hibernate.format\_sql=true  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL81Dialect  spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation= true  spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.security.SecurityAutoConfiguration |

|  |
| --- |
| **pom.xml** |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.7.2</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <groupId>com.icinbank</groupId>  <artifactId>icinbank</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>icinbank</name>  <description>Demo project for Spring Boot</description>    <properties>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>commons-codec</groupId>  <artifactId>commons-codec</artifactId>  <version>1.11</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  <exclusions>  <exclusion>  <groupId>org.junit.vintage</groupId>  <artifactId>junit-vintage-engine</artifactId>  </exclusion>  </exclusions>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

|  |
| --- |
| **Admin Portal -Backend** |
| package com.admin;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class AdminServiceApplication {  public static void main(String[] args) {  SpringApplication.run(AdminServiceApplication .class, args);  }  } |

|  |
| --- |
| **Model Class** |
| package com.admin.model;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.OneToOne;  import javax.persistence.Table;  @Entity  @Table(name = "account")  public class Account {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  @Column(name = "id")  private int id;  private long accno;  private int balance;  private String username;  @OneToOne(fetch = FetchType.LAZY)  @JoinColumn(name = "user\_id")  private User user;  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getAccno() {  return accno;  }  public void setAccno(long accno) {  this.accno = accno;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  } |

|  |
| --- |
| package com.admin.model;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.OneToOne;  import javax.persistence.Table;  @Entity  @Table(name = "account")  public class Account {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  @Column(name = "id")  private int id;  private long accno;  private int balance;  private String username;  @OneToOne(fetch = FetchType.LAZY)  @JoinColumn(name = "user\_id")  private User user;  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getAccno() {  return accno;  }  public void setAccno(long accno) {  this.accno = accno;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  } |

|  |
| --- |
| package com.admin.model;  import java.time.LocalDate;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  @Entity  @Table  public class ChequebookRequest {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long account;  private String accType;  private boolean requestStatus;  private LocalDate date;  private int no\_of\_pages;  public long getAccount() {  return account;  }  public void setAccount(long account) {  this.account = account;  }  public LocalDate getDate() {  return date;  }  public void setDate(LocalDate date) {  this.date = date;  }  public int getNo\_of\_pages() {  return no\_of\_pages;  }  public void setNo\_of\_pages(int no\_of\_pages) {  this.no\_of\_pages = no\_of\_pages;  }  public String getAccType() {  return accType;  }  public void setAccType(String accType) {  this.accType = accType;  }  public boolean getRequestStatus() {  return requestStatus;  }  public void setRequestStatus(boolean requestStatus) {  this.requestStatus = requestStatus;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  } |

|  |
| --- |
| package com.admin.model;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.OneToOne;  import javax.persistence.Table;  @Entity  @Table  public class Saccount {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    private long accno;    private int balance;    private String username;    @OneToOne(fetch=FetchType.LAZY)  @JoinColumn(name = "user\_id")  private User user;  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getAccno() {  return accno;  }  public void setAccno(long accno) {  this.accno = accno;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }    } |

|  |
| --- |
| package com.admin.model;  import java.time.LocalDate;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  @Entity  @Table  public class Transfer implements Comparable<Transfer> {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  private long saccount;  private long raccount;  private int amount;  private LocalDate date;  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public long getSaccount() {  return saccount;  }  public void setSaccount(long saccount) {  this.saccount = saccount;  }  public long getRaccount() {  return raccount;  }  public void setRaccount(long raccount) {  this.raccount = raccount;  }  public int getAmount() {  return amount;  }  public void setAmount(int amount) {  this.amount = amount;  }  public LocalDate getDate() {  return date;  }  public void setDate(LocalDate date) {  this.date = date;  }  @Override  public int compareTo(Transfer o) {  Integer i1 = this.id;  Integer i2 = o.id;  return i2.compareTo(i1);  }  } |

|  |
| --- |
| package com.admin.model;  import java.util.Date;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.OneToOne;  import javax.persistence.Table;  import com.fasterxml.jackson.annotation.JsonFormat;  @Entity  @Table(name="\"User\"")  public class User {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  @Column(name = "id")  private int id;  private String fname;  private String lname;  private long phone;  private String address;  private String email;  private String username;  private String password;  @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-mm-yyyy")  private Date dob;  private String identityType;  private String identity;  @Column(columnDefinition = "boolean default false")  private boolean status;  @Column(columnDefinition = "boolean default false")  private boolean authorizationStatus;  @Column(columnDefinition = "integer default 3", nullable = false)  private int featureStatus = 3;  @OneToOne(targetEntity = Account.class, mappedBy = "user", orphanRemoval = false, fetch = FetchType.LAZY)  private Account account;  @OneToOne(targetEntity = Saccount.class, mappedBy = "user", orphanRemoval = false, fetch = FetchType.LAZY)  private Saccount sAccount;  public Date getDob() {  return dob;  }  public void setDob(Date dob) {  this.dob = dob;  }  public String getIdentityType() {  return identityType;  }  public void setIdentityType(String identityType) {  this.identityType = identityType;  }  public String getIdentity() {  return identity;  }  public void setIdentity(String identity) {  this.identity = identity;  }  public int getFeatureStatus() {  return featureStatus;  }  public void setFeatureStatus(int featureStatus) {  this.featureStatus = featureStatus;  }  public boolean isStatus() {  return status;  }  public boolean isAuthorizationStatus() {  return authorizationStatus;  }  public void setAuthorizationStatus(boolean authorizationStatus) {  this.authorizationStatus = authorizationStatus;  }  public void setId(int id) {  this.id = id;  }  public void setStatus(boolean status) {  this.status = status;  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getFname() {  return fname;  }  public void setFname(String fname) {  this.fname = fname;  }  public String getLname() {  return lname;  }  public void setLname(String lname) {  this.lname = lname;  }  public long getPhone() {  return phone;  }  public void setPhone(long phone) {  this.phone = phone;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  } |

|  |
| --- |
| **package** com.admin.model;  **public** **class** UserDisplay {  **private** String fname;  **private** String lname;  **private** **long** phone;  **private** String username;  **private** **boolean** status;  **private** **int** featureStatus;  **private** **long** primaryAccno;  **private** **int** primaryBalance;  **private** **long** savingsAccno;  **private** **int** savingsBalance;  **public** UserDisplay(String fname, String lname, **long** phone, String username, **boolean** status, **int** featureStatus,  **long** primaryAccno, **int** primaryBalance, **long** savingsAccno, **int** savingsBalance) {  **super**();  **this**.fname = fname;  **this**.lname = lname;  **this**.phone = phone;  **this**.username = username;  **this**.status = status;  **this**.featureStatus = featureStatus;  **this**.primaryAccno = primaryAccno;  **this**.primaryBalance = primaryBalance;  **this**.savingsAccno = savingsAccno;  **this**.savingsBalance = savingsBalance;  }  **public** String getFname() {  **return** fname;  }  **public** **void** setFname(String fname) {  **this**.fname = fname;  }  **public** String getLname() {  **return** lname;  }  **public** **void** setLname(String lname) {  **this**.lname = lname;  }  **public** **long** getPhone() {  **return** phone;  }  **public** **void** setPhone(**long** phone) {  **this**.phone = phone;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** **boolean** isStatus() {  **return** status;  }  **public** **void** setStatus(**boolean** status) {  **this**.status = status;  }  **public** **long** getPrimaryAccno() {  **return** primaryAccno;  }  **public** **void** setPrimaryAccno(**long** primaryAccno) {  **this**.primaryAccno = primaryAccno;  }  **public** **int** getPrimaryBalance() {  **return** primaryBalance;  }  **public** **void** setPrimaryBalance(**int** primaryBalance) {  **this**.primaryBalance = primaryBalance;  }  **public** **long** getSavingsAccno() {  **return** savingsAccno;  }  **public** **void** setSavingsAccno(**long** savingsAccno) {  **this**.savingsAccno = savingsAccno;  }  **public** **int** getSavingsBalance() {  **return** savingsBalance;  }  **public** **void** setSavingsBalance(**int** savingsBalance) {  **this**.savingsBalance = savingsBalance;  }  **public** **int** getFeatureStatus() {  **return** featureStatus;  }  **public** **void** setFeatureStatus(**int** featureStatus) {  **this**.featureStatus = featureStatus;  }  } |

|  |
| --- |
| **Controller** |
| package com.admin.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RestController;  import com.admin.model.ChequebookRequest;  import com.admin.model.User;  import com.admin.model.UserDisplay;  import com.admin.service.impl.AdminServiceImpl;  import com.admin.service.impl.MailServiceImpl;  @RestController  @CrossOrigin  public class AdminController {    @Autowired  private AdminServiceImpl service;    @GetMapping("user/{username}/features/{featureId}")  public void setUserFeatures(@PathVariable("username") String username, @PathVariable("featureId") int featureId) {  service.setUserFeatures(username, featureId);  }    @GetMapping("user/{username}/authorize")  public void authorizeUser(@PathVariable("username") String username) {  try {  service.authorizeUser(username);  } catch (Exception e) {  e.printStackTrace();  }  }  @GetMapping("user/{username}/authorize/cancel")  public void cancelAuthorization(@PathVariable("username") String username) {  try {  //mailService.sendAuthorizeCancelEmail(username);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  service.cancelAuthorization(username);  }    @GetMapping("user/unauthorized/all")  public List<User> getAllUnauthorizedUsers()  {  return service.getAllUnauthorizedUsers();  }    @GetMapping("/user/all")  public List<UserDisplay> getAllUsers()  {  return service.getAllUsers();  }    // @GetMapping("/user/transfers/{account}")  // public List<Transfer> getTransactionList(@PathVariable("account") int account)  // {  // return service.getAllTransactions(account);  // }    @GetMapping("/chequebook/request/all")  public List<ChequebookRequest> getAllChequeBookRequests()  {  return service.getAllChequebookRequests();  }    @GetMapping("/user/{accNo}/chequebook/request/confirm")  public void confirmChequeBookRequest(@PathVariable("accNo") long accNo)  {  service.acceptChequebookRequest(accNo);  }    @GetMapping("/user/{username}/enable")  public void enableUser(@PathVariable("username") String username)  {  service.enableUser(username);  }    @GetMapping("/user/{username}/disable")  public void disableUser(@PathVariable("username") String username)  {  service.disableUser(username);  }    @GetMapping("search/user/{userDetail}")  public UserDisplay searchUser(@PathVariable("userDetail") String userDetail) {  return service.searchUser(userDetail);  }    } |

|  |
| --- |
| **DAO Class** |
| package com.admin.dao;  import org.springframework.data.repository.CrudRepository;  import org.springframework.stereotype.Repository;  import com.admin.model.Account;  @Repository  public interface AccountRepository extends CrudRepository<Account, Integer>{  public Account findByUsername(String username);  public Account findByAccno(long accno);  } |

|  |
| --- |
| package com.admin.dao;  import java.util.List;  import javax.transaction.Transactional;  import org.springframework.data.jpa.repository.Modifying;  import org.springframework.data.jpa.repository.Query;  import org.springframework.data.repository.CrudRepository;  import org.springframework.stereotype.Repository;  import com.admin.model.ChequebookRequest;  @Repository  public interface ChequeBookRequestsRepository extends CrudRepository<ChequebookRequest, Integer>{      @Modifying  @Transactional  @Query("update ChequebookRequest c set c.requestStatus=1 where c.account = ?1")  void setChequebookInfoByAccount(long accNo);    @Query("FROM ChequebookRequest c where c.requestStatus=FALSE")  public List<ChequebookRequest> findAllChequebookRequests();  } |

|  |
| --- |
| package com.admin.dao;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.admin.model.Saccount;  @Repository  public interface SaccountRepository extends JpaRepository<Saccount, Integer>{    public Saccount findByAccno(long accNo);  } |

|  |
| --- |
| package com.admin.dao;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.admin.model.Transfer;  @Repository  public interface TransferRepository extends JpaRepository<Transfer, Integer> {  public List<Transfer>findBySaccount(long saccount);  public List<Transfer>findByRaccount(long raccount);  } |

|  |
| --- |
| package com.admin.dao;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.stereotype.Repository;  import com.admin.model.User;  import com.admin.model.UserDisplay;  @Repository  public interface UserDisplayRepository extends JpaRepository<User, Integer>{    @Query("SELECT new com.admin.model.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE u.username=a.username and u.username=s.username")  public List<UserDisplay> getAllUsers();    @Query("SELECT new com.admin.model.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE u.username=?1 and u.username=a.username and u.username=s.username")  public UserDisplay getUserDetailsByUsername(String userDetail);    @Query("SELECT new com.admin.model.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE s.accno=?1 and u.username=a.username and u.username=s.username")  public UserDisplay getUserDetailsByAccountNo(long accNo);    } |

|  |
| --- |
| package com.admin.dao;  import java.util.List;  import javax.transaction.Transactional;  import org.springframework.data.jpa.repository.Modifying;  import org.springframework.data.jpa.repository.Query;  import org.springframework.data.repository.CrudRepository;  import com.admin.model.User;  public interface UserRepository extends CrudRepository<User, Integer>{    @Query("FROM User u WHERE u.username=?1")  public User findByUsername(String username);  @Modifying  @Transactional  @Query("update User u set u.status=1 where u.username = ?1")  void disableUser(String username);    @Modifying  @Transactional  @Query("update User u set u.status=0 where u.username = ?1")  void enableUser(String username);    @Modifying  @Transactional  @Query("update User u set u.authorizationStatus=1 where u.username = ?1")  void authorizeUser(String username);  @Modifying  @Transactional  @Query("delete from User u where u.username = ?1")  void cancelAuthorization(String username);  @Query("FROM User u where u.authorizationStatus=FALSE")  public List<User> findAllUnauthorizedAccounts();  @Modifying  @Transactional  @Query("update User u set u.featureStatus=?2 where u.username = ?1")  public void setUserFeatureStatus(String username, int featureId);    } |

|  |
| --- |
| **Service** |
| package com.admin.service;  import java.util.List;  import com.admin.model.ChequebookRequest;  import com.admin.model.Transfer;  import com.admin.model.User;  import com.admin.model.UserDisplay;  public interface AdminService {  public List<UserDisplay> getAllUsers();  public List<Transfer> getAllTransactions(long accountNo);  public List<ChequebookRequest> getAllChequebookRequests();  //public void acceptChequebookRequest(int accNo);  public void enableUser(String username);  public void disableUser(String username);  public void authorizeUser(String username);  public void cancelAuthorization(String username);  public List<User> getAllUnauthorizedUsers();  public void setUserFeatures(String username, int featureId);  public UserDisplay searchUser(String userDetail);  void acceptChequebookRequest(long accNo);  } |

|  |
| --- |
| **Service Implement** |
| package com.admin.service.impl;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.admin.dao.AccountRepository;  import com.admin.dao.UserRepository;  import com.admin.model.Account;  @Service  public class AccountsCreationImpl {    @Autowired  AccountRepository accountRepo;    @Autowired  private UserRepository udao;    private final String bankCode = "3914";  private final String countryCode = "91";  private final String branchCode = "820";  private final String accountcode="1";    public Account newAccount(String username,int userId) {  Account account=new Account();  account.setUsername(username);  account.setAccno(generate\_saving(userId));  account.setUser(udao.findByUsername(username));  return accountRepo.save(account);  }  public long generate\_saving(int userId) {  String accNo = bankCode+countryCode+branchCode+accountcode+String.valueOf(userId);  return Long.parseLong(accNo);  }  } |

|  |
| --- |
| package com.admin.service.impl;  import java.util.ArrayList;  import java.util.Collections;  import java.util.List;  import org.apache.commons.mail.EmailException;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.admin.dao.AccountRepository;  import com.admin.dao.ChequeBookRequestsRepository;  import com.admin.dao.SaccountRepository;  import com.admin.dao.TransferRepository;  import com.admin.dao.UserDisplayRepository;  import com.admin.dao.UserRepository;  import com.admin.model.ChequebookRequest;  import com.admin.model.Transfer;  import com.admin.model.User;  import com.admin.model.UserDisplay;  import com.admin.service.AdminService;  @Service  public class AdminServiceImpl implements AdminService{    @Autowired  private TransferRepository transDAO;    @Autowired  private ChequeBookRequestsRepository chequeBookDAO;    @Autowired  private UserRepository userDao;    @Autowired  private UserDisplayRepository displayDao;    @Autowired  private AccountRepository accDao;    @Autowired  private SaccountRepository sAccDao;    @Autowired  private MailServiceImpl mailService;    @Autowired  private AccountsCreationImpl accService;    @Autowired  private SaccountCreationImpl sAccService;    @Override  public void authorizeUser(String username) {  userDao.authorizeUser(username);  System.out.println("error here top");  User currUser = userDao.findByUsername(username);  int userId = currUser.getId();  System.out.println("error here 1");  accService.newAccount(username,userId);  System.out.println("error here 2");  sAccService.newAccount(username,userId);  }    @Override  public void cancelAuthorization(String username) {  userDao.cancelAuthorization(username);  }  @Override  public void acceptChequebookRequest(long accNo) {  String username = "";  chequeBookDAO.setChequebookInfoByAccount(accNo);  if(Long.toString(accNo).length() == 7) {  username = accDao.findByAccno(accNo).getUsername();  }  else {  username = sAccDao.findByAccno(accNo).getUsername();  }  try {  //mailService.sendChequebookConfirmedEmail(username);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  @Override  public void enableUser(String username) {  userDao.enableUser(username);    }  @Override  public void disableUser(String username) {  userDao.disableUser(username);    }  @Override  public List<UserDisplay> getAllUsers() {  return displayDao.getAllUsers();  }  @Override  public List<Transfer> getAllTransactions(long accountNo) {  List<Transfer> sender=transDAO.findBySaccount(accountNo);  List<Transfer> receiver=transDAO.findByRaccount(accountNo);  List<Transfer> merged=new ArrayList<>();  merged.addAll(sender);  merged.addAll(receiver);  Collections.sort(merged);  return merged;  }  @Override  public List<ChequebookRequest> getAllChequebookRequests() {  return chequeBookDAO.findAllChequebookRequests();  }  @Override  public List<User> getAllUnauthorizedUsers() {  return userDao.findAllUnauthorizedAccounts();  }  @Override  public void setUserFeatures(String username, int featureId) {  userDao.setUserFeatureStatus(username,featureId);    }    static boolean isNumber(String s)  {  for (int i = 0; i < s.length(); i++)  if (Character.isDigit(s.charAt(i))  == false)  return false;    return true;  }  @Override  public UserDisplay searchUser(String userDetail) {  if(isNumber(userDetail)) {  return displayDao.getUserDetailsByAccountNo(Long.parseLong(userDetail));  }  else {  return displayDao.getUserDetailsByUsername(userDetail);  }    }  } |

|  |
| --- |
| package com.admin.service.impl;  import org.apache.commons.mail.DefaultAuthenticator;  import org.apache.commons.mail.EmailException;  import org.apache.commons.mail.HtmlEmail;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.admin.dao.AccountRepository;  import com.admin.dao.UserRepository;  import com.admin.model.Account;  import com.admin.model.User;  @Service  public class MailServiceImpl {    @Autowired  UserRepository userDAO;    @Autowired  AccountRepository accountDAO;  /\*  public void sendAuthorizedEmail(String username) throws EmailException {    User user = userDAO.findByUsername(username);  String receiver = user.getEmail();  String fullName = user.getFname()+" "+user.getLname();  Account account = accountDAO.findByUsername(username);  long accountNo = account.getAccno();  int balance = account.getBalance();    HtmlEmail mail = new HtmlEmail();  mail.setSSL(true);  mail.setHostName("smtp.gmail.com");  mail.setSmtpPort(587);  mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));  mail.setFrom("icinbank.noreply@gmail.com", "ICIN");  mail.addTo(receiver, "To");  mail.setSubject(user.getFname()+", Welcome to your new ICIN Bank Account");  mail.setHtmlMsg("<h1 style='font-size:27px;color:blue'>Hi "+fullName+",</h1>"  + "<br><h2 style='font-size:21px;color:black'>Thank you for creating a ICIN Bank account. Your account has been activated,"  + " Here are your details.</h2><br>"  + "<p style='font-size:18px;'>Username: "+username+"<br><p style='font-size:18px;'>Account Number: "+accountNo+"<br>"  + "<p style='font-size:18px;'>Current Balance: Rs. "+balance+"</p>");  mail.send();  System.out.println("email sent");  }    public void sendAuthorizeCancelEmail(String username) throws EmailException {    User user = userDAO.findByUsername(username);  String receiver = user.getEmail();  String fullName = user.getFname()+" "+user.getLname();    HtmlEmail mail = new HtmlEmail();  mail.setSSL(true);  mail.setHostName("smtp.gmail.com");  mail.setSmtpPort(587);  mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));    mail.setFrom("icinbank.noreply@gmail.com", "ICIN");  mail.addTo(receiver, "To");  mail.setSubject("ICIN Bank Account Cancelled");  mail.setHtmlMsg("<h1 style='font-size:27px;color:red'>Hi "+fullName+",</h1>"  + "<br><p style='font-size:19px;color:black'>Sorry, Your account with ICIN Bank with username "+username+ " has been cancelled. Please contact nearest branch for more details.</p><br><br>");  mail.send();  System.out.println("email sent");  }    public void sendChequebookConfirmedEmail(String username) throws EmailException {    User user = userDAO.findByUsername(username);  String receiver = user.getEmail();  String fullName = user.getFname()+" "+user.getLname();    HtmlEmail mail = new HtmlEmail();  mail.setSSL(true);  mail.setHostName("smtp.gmail.com");  mail.setSmtpPort(465);  mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));  mail.setFrom("icinbank.noreply@gmail.com", "ICIN");  mail.addTo(receiver, "To");  mail.setSubject("ICIN Bank Chequebook Request Confirmed");  mail.setHtmlMsg("<h1 style='font-size:25px;color:green'>Hi "+fullName+",</h1>"  + "<br><p style='font-size:19px;color:black'>Your chequebook request has been processed and confirmed it will be sent to you via courier in 4-5 business days.</p><br><br>");  mail.send();  System.out.println("email sent");  }  \*/  } |

|  |
| --- |
| package com.admin.service.impl;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.admin.dao.SaccountRepository;  import com.admin.dao.UserRepository;  import com.admin.model.Saccount;  @Service  public class SaccountCreationImpl {    @Autowired  SaccountRepository dao;    @Autowired  UserRepository udao;    private final String bankCode = "3914";  private final String countryCode = "91";  private final String branchCode = "820";  private final String accountcode="2";    public long generate\_saving(int userId) {  String accNo = bankCode+countryCode+branchCode+accountcode+String.valueOf(userId);  return Long.parseLong(accNo);  }  public Saccount newAccount(String username, int userId) {  Saccount account =new Saccount();  account.setUsername(username);  account.setAccno(generate\_saving(userId));  account.setUser(udao.findByUsername(username));  return dao.save(account);  }  } |

|  |
| --- |
| **Application.properties** |
| server.port=8085  spring.datasource.url=jdbc:postgresql://localhost:5432/icin  spring.datasource.username=postgres  spring.datasource.password=pravin@5  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true  spring.jpa.properties.hibernate.format\_sql=true  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL81Dialect  spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation= true  spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.security.SecurityAutoConfiguration |

|  |
| --- |
| **pom.xml** |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.7.2</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <groupId>com.admin</groupId>  <artifactId>admin\_service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>admin\_service</name>  <description>Admin service for icin bank</description>  <properties>  <java.version>1.8</java.version>  </properties>    <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  <exclusions>  <exclusion>  <groupId>org.junit.vintage</groupId>  <artifactId>junit-vintage-engine</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-email</artifactId>  <version>1.5</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

|  |
| --- |
| **User Portal - Fronted** |
| //login component  <div class="bank-login">      <div class="container">          <form [formGroup]="loginForm" (ngSubmit)="onFormSubmit()" class="login-form">              <h3 class="text-center">                  <svg xmlns="http://www.w3.org/2000/svg" width="30" height="30" fill="currentColor"                      class="bi bi-person-bounding-box" viewBox="0 0 16 16">                      <path                          d="M1.5 1a.5.5 0 0 0-.5.5v3a.5.5 0 0 1-1 0v-3A1.5 1.5 0 0 1 1.5 0h3a.5.5 0 0 1 0 1h-3zM11 .5a.5.5 0 0 1 .5-.5h3A1.5 1.5 0 0 1 16 1.5v3a.5.5 0 0 1-1 0v-3a.5.5 0 0 0-.5-.5h-3a.5.5 0 0 1-.5-.5zM.5 11a.5.5 0 0 1 .5.5v3a.5.5 0 0 0 .5.5h3a.5.5 0 0 1 0 1h-3A1.5 1.5 0 0 1 0 14.5v-3a.5.5 0 0 1 .5-.5zm15 0a.5.5 0 0 1 .5.5v3a1.5 1.5 0 0 1-1.5 1.5h-3a.5.5 0 0 1 0-1h3a.5.5 0 0 0 .5-.5v-3a.5.5 0 0 1 .5-.5z" />                      <path d="M3 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1H3zm8-9a3 3 0 1 1-6 0 3 3 0 0 1 6 0z" />                  </svg>                  Login User Portal              </h3>              <hr />              <div class="form-group">                  <label for="username">Username</label>                  <input type="text" formControlName="username" class="form-control"                      [ngClass]="{ 'is-invalid': submitted && fval.username.errors }" placeholder="Enter Username" />                  <div \*ngIf="submitted && fval.username.errors" class="invalid-feedback">                      <div \*ngIf="fval.username.errors.required">Username is required</div>                  </div>              </div>              <div class="form-group">                  <label for="password">Password</label>                  <input type="password" formControlName="password" class="form-control"                      [ngClass]="{ 'is-invalid': submitted && fval.password.errors }" placeholder="Enter password here" />                  <div \*ngIf="submitted && fval.password.errors" class="invalid-feedback">                      <div \*ngIf="fval.password.errors.required">Please enter a Password</div>                  </div>              </div>              <hr />              <div class="row">                  <div class="col-lg-6"> <button [disabled]="loading" class="btn btn-success full-width">                          <span \*ngIf="loading" class="spinner-border spinner-border-sm mr-1"></span>                          Login                      </button></div>                  <div class="col-lg-6">                      <a routerLink="/register" class="btn btn-secondary full-width">Register</a>                  </div>              </div>          </form>      </div>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { Router } from '@angular/router';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { Login } from '../models';  import { LoginService } from '../login.service';  import { AuthService } from '../auth.service';  import Swal from 'sweetalert2';  @Component({    selector: 'app-login',    templateUrl: './login.component.html',    styleUrls: ['./login.component.css']  })  export class LoginComponent implements OnInit {    loginForm: FormGroup;    loading = false;    submitted = false;    returnUrl: string;    isLoginError: boolean = false;    constructor(      private formBuilder: FormBuilder,      private router: Router,      private loginService: LoginService,      private authService: AuthService    ) { }    ngOnInit() {      this.loginForm = this.formBuilder.group({        username: ['', Validators.required],        password: ['', Validators.required]      });    }    get fval() { return this.loginForm.controls; }    onFormSubmit() {      this.submitted = true;      if (this.loginForm.invalid) {        return;      }      this.loading = true;      const result: Login = Object.assign({}, this.loginForm.value);      this.loginService.loginUser(result.username, result.password).subscribe((data: any) => {        if (data.loginStatus) {          this.authService.authenticate(true);          this.authService.setUserName(data.username);            localStorage.setItem('login', data.loginStatus);          localStorage.setItem('username', data.username);          Swal.fire({            icon: 'success',            title: 'Login successful',            showConfirmButton: false,            timer: 2000          })          this.router.navigate(['/home']);        } else {          Swal.fire({            icon: 'error',            title: 'Oops...',            text: data.responseMessage,          })          this.authService.authenticate(false);          this.loading = false;        }      })    }  } |

|  |
| --- |
| .container {      margin: 5rem auto;      width: 20rem;  }  .input {      margin-bottom: 2rem;      margin-top: 2rem;  }  .bank-login {      background: #ccc;      background-image: url('../../assets/bg.jpg');      position: absolute;      width: 100%;      height: 100%;      top: 0  }  .login-form {      border: 1px solid #ccc;      padding: 20px;      background: #f1f1f1;      width: 400px;      margin-top: 200px;      border-color: #000;      opacity: 0.96;      box-shadow: #777 1px 5px 10px;  }  .login-form label {      font-weight: bold;  }  .full-width {      width: 100%;  }  ::placeholder {      /\* Chrome, Firefox, Opera, Safari 10.1+ \*/      color: #ccc;      font-size: 13px;      opacity: 1;      /\* Firefox \*/  }  :-ms-input-placeholder {      /\* Internet Explorer 10-11 \*/      color: #ccc;  }  ::-ms-input-placeholder {      /\* Microsoft Edge \*/      color: #ccc;  } |

|  |
| --- |
| //home component  <div class="container">    <div class="row">      <div class="col-lg-9">        <div class="card">          <div class="card-body box-wrap laptop-bg">            <h3>Welcome to ICIN Bank Pravin !</h3>            <p>              We offer you actions like Transfer of money, Checkbook request, Deposit,              Withdraw.            </p>            <br />            <p> This is your Saving <span class="fw-bold">A/C NO: {{ savingAcc }} </span> </p>            <p>              <a class="btn btn-info" href="/deposit"><i class="fa-solid fa-dollar"></i> Deposit Money</a>            </p>          </div>        </div>      </div>      <div class="col-lg-3">        <div class="card">          <div class="card-body box-wrap">            <div class="box-title balance"><i class="fa-solid fa-hand-holding-dollar"></i> Saving Balance</div>            <div class="box-info"> ₹ {{ savingBalanceLocal }}.00</div>            <p class="text-center">             This your saving account balace, Deposit money to increase Savings!            </p>          </div>        </div>      </div>    </div>    <div class="row mt-3">      <div class="col-lg-3">        <div class="card">          <div class="card-body box-wrap">            <div class="box-title transfer"><i class="fa-solid fa-money-bill-transfer"></i> Total Transfer</div>            <div class="box-info"> ₹ {{ savingBalanceLocal }}.00</div>            <p class="text-center">              <br/>              <a class="btn btn-warning" href="/deposit">Transfer Money</a>            </p>          </div>        </div>      </div>      <div class="col-lg-3">        <div class="card">          <div class="card-body box-wrap">            <div class="box-title withdrawl"><i class="fa-solid fa-magnifying-glass-dollar"></i> Total Withdrawl</div>            <div class="box-info"> ₹ {{ savingBalanceLocal }}.00</div>            <p class="text-center">              <br/>              <a class="btn btn-danger" href="/deposit">Withdraw Money</a>            </p>          </div>        </div>      </div>      <div class="col-lg-3">        <div class="card">          <div class="card-body box-wrap">            <div class="box-title deposit"><i class="fa-solid fa-sack-dollar"></i> Total Deposit </div>            <div class="box-info"> ₹ {{ savingBalanceLocal }}.00</div>            <p class="text-center">              <br/>              <a class="btn btn-info" href="/deposit">Deposit Money</a>            </p>          </div>        </div>      </div>      <div class="col-lg-3">        <div class="card">          <div class="card-body box-wrap">            <div class="box-title transaction"><i class="fa-solid fa-receipt"></i> Total Transaction</div>            <div class="box-info"> ₹ {{ savingBalanceLocal }}.00</div>            <p class="text-center">              <br/>              <a class="btn btn-secondary" href="/deposit">Withdraw Money</a>            </p>          </div>        </div>      </div>    </div>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { AuthService } from '../auth.service';  import { UserService } from '../user.service';  import { Router } from '@angular/router';  @Component({    selector: 'app-home',    templateUrl: './home.component.html',    styleUrls: ['./home.component.css']  })  export class HomeComponent implements OnInit {    username = localStorage.getItem("username");    savingAcc: number;    primaryAcc: number;    savingBalanceLocal: number;    primaryBalanceLocal: number;    constructor(public authService: AuthService, public userService: UserService, private router: Router) {      console.log(this.username);    }    ngOnInit(): void {      this.userService.getUser(this.username).subscribe(res => {        console.log(res)        this.savingAcc = res.savingsAccno;        this.primaryAcc = res.primaryAccno;        this.savingBalanceLocal = res.savingsBalance;        this.primaryBalanceLocal = res.primaryBalance;        localStorage.setItem("savingAccNo", this.savingAcc.toString());      });    }    displayuserdetails() {      this.userService.getUser(this.username).subscribe(() => this.ngOnInit());    }  } |

|  |
| --- |
| #userSession {    float: right;  }  .center {    left: 50%;    margin-left: auto;    margin-right: auto;  }  #userSession {    float: right;  }  .center {    left: 50%;    margin-left: auto;    margin-right: auto;  }  body .infobox {    font-weight: 400;    font-size: 20px;    color: #323232;    height: auto;    padding: 2%;    background-color: #f1ecec;    border: 1px solid #f5f5f5;    border-radius: 15px;    -webkit-box-shadow: 0px 0px 10px 0px #f5f5f5;    box-shadow: 0px 0px 10px 0px #f5f5f5;    margin: 10rem auto;    width: 35em;  }  .box-wrap {    min-height: 250px;  }  .box-wrap a {    color: #fff;  }  .laptop-bg {    background-image: url('../../assets/man-with-laptop-light.png');    background-position: bottom right;    background-repeat: no-repeat;    background-size: contain;  }  .box-title {    font-size: 1.2em;    color: #333;    padding: 10px 10px;    text-align: center;    font-weight: bold;  }  .box-info {    font-size: 1.825rem;    color: #000;    padding: 15px 10px;    text-align: center;    margin: 10px 0px;    border: 1px dashed #ccc;    border-left: none;    border-right: none;  }  .transfer {    color: #f39c12;  }  .withdrawl {    color: #e74c3c;  }  .deposit {    color: #0dcaf0;  }  .balance {    color: #696cff;  }  .transaction {    color: #2c3e50;  } |

|  |
| --- |
| // header component  <nav class="navbar navbar-expand-lg fixed-top nav-border " \*ngIf="isLoggedIn | async as isLoggedIn">      <div class="container">          <a class="navbar-brand" routerLink="/home">              <i class="fa-solid fa-building-columns"></i>              ICIN Bank</a>          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"              aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">              <span class="navbar-toggler-icon"></span>          </button>          <div class="collapse navbar-collapse" id="navbarSupportedContent">              <ul class="navbar-nav me-auto mb-2 mb-lg-0">                  <li class="nav-item">                      <a class="nav-link" routerLink="/transactionHistory">Transaction History<span                              class="sr-only"></span></a>                  </li>                  <li class="nav-item">                      <a class="nav-link" routerLink="/transferHistory">Transfer History<span class="sr-only"></span></a>                  </li>                  <li class="nav-item">                      <a class="nav-link" routerLink="/deposit">Deposit Money<span class="sr-only"></span></a>                  </li>                  <li class="nav-item">                      <a class="nav-link" routerLink="/withdraw">Withdraw Money<span class="sr-only"></span></a>                  </li>                  <li class="nav-item">                      <a class="nav-link" routerLink="/transfer">Transfer Money<span class="sr-only"></span></a>                  </li>                  <li class="nav-item">                      <a class="nav-link" routerLink="/chequebookRequest">Request Checkbook<span                              class="sr-only"></span></a>                  </li>                  <li class="nav-item dropdown">                      <a class="nav-link dropdown-toggle userName" href="Javascript:void(0)" id="dropdown04"                          data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" (click)="toggleMenu()">                          <i class="fa-solid fa-user"></i>                          {{user | async}}                      </a>                      <div class="dropdown-menu" [ngClass]="{'show': isShow}" aria-labelledby="dropdown04">                          <a class="dropdown-item" routerLink="/editProfile">Change Profile Settings</a>                          <a class="dropdown-item" routerLink="/login" (click)="logOut()">Logout</a>                      </div>                  </li>              </ul>          </div>      </div>  </nav> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { Observable } from 'rxjs';  import { AuthService } from '../auth.service';  @Component({    selector: 'app-header',    templateUrl: './header.component.html',    styleUrls: ['./header.component.css']  })  export class HeaderComponent implements OnInit {    isLoggedIn: Observable<boolean>;    user: Observable<string>;    isShow = false;    constructor(public authService: AuthService) { }    ngOnInit(): void {      this.isLoggedIn = this.authService.isLoggedIn;      this.user = this.authService.userName;    }    toggleMenu() {      this.isShow = !this.isShow;    }    logOut() {      localStorage.setItem('login', 'false');      this.isShow = !this.isShow;      this.authService.authenticate(false);    }  } |

|  |
| --- |
| nav.navbar {      margin-bottom: 25px;  }  .userName{      text-transform: capitalize;  } |

|  |
| --- |
| // footer component  <footer class="app-footer"  \*ngIf="isLoggedIn | async as isLoggedIn">    <section class="footer-section-1">      <div class="container">        <!-- Grid row -->        <div class="row">          <!-- Grid column -->          <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">            <!-- Content -->            <h6 class="text-uppercase fw-bold">              <i class="fa-solid fa-building-columns"></i>              ICIN BANK            </h6>            <hr class="mb-4 mt-0 d-inline-block mx-auto" style="width: 60px; background-color: #7c4dff; height: 2px" />            <p>              Here you can use rows and columns to organize your footer              content. Lorem ipsum dolor sit amet, consectetur adipisicing              elit.            </p>          </div>          <!-- Grid column -->          <!-- Grid column -->          <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">            <!-- Links -->            <h6 class="text-uppercase fw-bold">Products</h6>            <hr class="mb-4 mt-0 d-inline-block mx-auto" style="width: 60px; background-color: #7c4dff; height: 2px" />            <p>              <a href="#!" class="text-dark">MDBootstrap</a>            </p>            <p>              <a href="#!" class="text-dark">MDWordPress</a>            </p>            <p>              <a href="#!" class="text-dark">BrandFlow</a>            </p>            <p>              <a href="#!" class="text-dark">Bootstrap Angular</a>            </p>          </div>          <!-- Grid column -->          <!-- Grid column -->          <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">            <!-- Links -->            <h6 class="text-uppercase fw-bold">Useful links</h6>            <hr class="mb-4 mt-0 d-inline-block mx-auto" style="width: 60px; background-color: #7c4dff; height: 2px" />            <p>              <a href="#!" class="text-dark">Your Account</a>            </p>            <p>              <a href="#!" class="text-dark">Become an Affiliate</a>            </p>            <p>              <a href="#!" class="text-dark">Shipping Rates</a>            </p>            <p>              <a href="#!" class="text-dark">Help</a>            </p>          </div>          <!-- Grid column -->          <!-- Grid column -->          <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">            <!-- Links -->            <h6 class="text-uppercase fw-bold">Contact</h6>            <hr class="mb-4 mt-0 d-inline-block mx-auto" style="width: 60px; background-color: #7c4dff; height: 2px" />            <p><i class="fas fa-home mr-3"></i> New York, NY 10012, US</p>            <p><i class="fas fa-envelope mr-3"></i> info@icin.com</p>            <p><i class="fas fa-phone mr-3"></i> + 01 234 567 88</p>            <p><i class="fas fa-print mr-3"></i> + 01 234 567 89</p>          </div>          <!-- Grid column -->        </div>        <!-- Grid row -->      </div>    </section>    <section class="footer-section-2">      <div class="container">        <div class="row">          <div class="col-lg-4">            © 2022 Copyright:            <a class="text-dark" href="#">ICIN Bank</a>          </div>          <div class="col-lg-8">            <div class="float-end">              <span>Get connected with us on social networks: </span>              <span>                <a href="" class="text-white me-4">                  <i class="fab fa-facebook-f"></i>                </a>                <a href="" class="text-white me-4">                  <i class="fab fa-twitter"></i>                </a>                <a href="" class="text-white me-4">                  <i class="fab fa-google"></i>                </a>                <a href="" class="text-white me-4">                  <i class="fab fa-instagram"></i>                </a>                <a href="" class="text-white me-4">                  <i class="fab fa-linkedin"></i>                </a>                <a href="" class="text-white me-4">                  <i class="fab fa-github"></i>                </a>              </span>            </div>          </div>        </div>      </div>    </section>  </footer> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { Observable } from 'rxjs';  import { AuthService } from '../auth.service';  @Component({    selector: 'app-footer',    templateUrl: './footer.component.html',    styleUrls: ['./footer.component.css']  })  export class FooterComponent implements OnInit {    isLoggedIn: Observable<boolean>;    constructor(public authService: AuthService) { }    ngOnInit(): void {      this.isLoggedIn = this.authService.isLoggedIn;    }  } |

|  |
| --- |
| .app-footer {      background-color: #666;      padding-top: 15px;      color: #fff;  }  .footer-section-1 {      color: #fff;  }  .footer-section-2 {      padding: 15px;      color: #fff;      background-color: #21D192;  }  .footer-section-2 a, .footer-section-1 a {      color: #fff !important;  } |

|  |
| --- |
| //cheque-book-request component    <div class="container col-xs-4">      <h3 style="text-align: center">Request Cheque Book</h3>      <div class="form-group">        <div style="text-align: center" class="form-group">          <label>Account Number :&nbsp;</label> <label>{{ requestingAccNo }}</label>        </div>      </div>        <div class="form-group">        <label>ChequeBook pages</label>        <br />        <select class="form-control" (change)="$any($event.target).value">          <option \*ngFor="let p of pages" [value]="p.value">            {{ p.name }}          </option>        </select>        <br />        <button (click)="setOption()" class="btn btn-info form-control">          <span \*ngIf="loading" class="spinner-border spinner-border-sm mr-1"></span          >Request Chequebook        </button>        <br />      </div>    </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormBuilder } from '@angular/forms';  import { Router } from '@angular/router';  import { RequestService } from '../request.service';  import Swal from 'sweetalert2';  @Component({    selector: 'app-cheque-book-request',    templateUrl: './cheque-book-request.component.html',    styleUrls: ['./cheque-book-request.component.css']  })  export class ChequeBookRequestComponent implements OnInit {    requestingAccNo: number = +localStorage.getItem("savingAccNo");    selectedValue: number;    loading: boolean = false;    pages = [      { name: "20", value: 20 },      { name: "50", value: 50 },      { name: "75", value: 75 }    ]    filterSelected(value) {      this.selectedValue = value;    }    constructor(      private formBuilder: FormBuilder,      private router: Router,      private requestService: RequestService,    ) { }    ngOnInit(): void {    }    setOption() {      this.loading = true;      if (this.selectedValue == null) {        this.selectedValue = 20;      }      try {        this.requestService.insertRequest(this.requestingAccNo, +this.selectedValue).subscribe((res: any) => {          console.log(res);          this.loading = false;          if (res.status) {            Swal.fire(              {                icon: 'success',                title: 'Chequebook request placed!',                text: res.responseMessage              }            )          } else {            Swal.fire({              icon: 'error',              title: 'Oops...',              text: res.responseMessage,            })          }        });      }      catch {        this.loading = false;      }    }  } |

|  |
| --- |
| .container{      margin: 10rem auto;      width:20em;   }     .input{      margin-bottom: 2rem ;      margin-top: 2rem;  } |

|  |
| --- |
| //deposit component  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <div class="container">    <form [formGroup]="depositForm" (ngSubmit)="deposit()" method="post">      <div class="col-xs-4" style="left: 35%">        <h3 style="text-align: center">Deposit Money</h3>      </div>      <br />      <div class="form-group" style="text-align: center">        <label>Account Number : </label> <label>{{ saccountno }} </label>      </div>      <div class="form-group">        <label>Enter Amount:</label>        <br />        <input          type="text"          formControlName="amount"          class="form-control"          [ngClass]="{ 'is-invalid': submitted && fval.amount.errors }"          placeholder="Enter amount"        />        <div \*ngIf="submitted && fval.amount.errors" class="invalid-feedback">          <div \*ngIf="fval.amount.errors.required">Amount is required</div>        </div>      </div>      <div class="center1">        <button          [disabled]="loading"          class="btn btn-primary"          class="btn btn-info form-control"        >          <span            \*ngIf="loading"            class="spinner-border spinner-border-sm mr-1"          ></span>          <a (click)="deposit()" style="color: white">Deposit</a>        </button>      </div>    </form>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { Router } from '@angular/router';  import { DepositService } from '../deposit.service';  import Swal from 'sweetalert2';  @Component({    selector: 'app-deposit',    templateUrl: './deposit.component.html',    styleUrls: ['./deposit.component.css'],  })  export class DepositComponent implements OnInit {    constructor(      private formBuilder: FormBuilder,      private router: Router,      private depositService: DepositService    ) {}    depositForm: FormGroup;    loading = false;    submitted = false;    ngOnInit(): void {      const accNo = +localStorage.getItem('savingAccNo');      console.log(accNo);      this.depositForm = this.formBuilder.group({        account: accNo,        amount: ['', [Validators.required]],      });    }    get saccountno(): any {      return localStorage.getItem('savingAccNo');    }    get fval() {      return this.depositForm.controls;    }    deposit() {      this.submitted = true;      if (this.depositForm.invalid) {        return;      }      this.loading = true;      const result: any = Object.assign({}, this.depositForm.value);      // Do useful stuff with the gathered data      try {        this.depositService          .insertEntry(result.account, +result.amount)          .subscribe((data: any) => {            this.loading = false;            if (data.depositStatus) {              Swal.fire({                icon: 'success',                title: 'Transaction successful',                text: data.responseMessage,              });            } else {              Swal.fire({                icon: 'error',                title: 'Oops...',                text: data.responseMessage,              });            }          });      } catch {        this.loading = false;      }    }  } |

|  |
| --- |
| .container {      margin: 10rem auto;      width: 30rem;    }      .input {      margin-bottom: 2rem;      margin-top: 2rem;    } |

|  |
| --- |
| //edit-profile component  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <nav class="navbar navbar-expand-sm navbar-dark bg-dark">    <a class="navbar-brand" routerLink="/home">ICIN Bank</a>    <button      class="navbar-toggler"      type="button"      data-toggle="collapse"      data-target="#navbarsExample04"      aria-controls="navbarsExample04"      aria-expanded="false"      aria-label="Toggle navigation"    >      <span class="navbar-toggler-icon"></span>    </button>    <div class="collapse navbar-collapse" id="navbarsExample04">      <ul class="navbar-nav mr-auto">        <li class="nav-item">          <a class="nav-link" routerLink="/transactionHistory"            >Transaction History<span class="sr-only"></span          ></a>        </li>        <li class="nav-item">          <a class="nav-link" routerLink="/transferHistory"            >Transfer History<span class="sr-only"></span          ></a>        </li>        <li class="nav-item">          <a class="nav-link" routerLink="/deposit"            >Deposit Money<span class="sr-only"></span          ></a>        </li>        <li class="nav-item">          <a class="nav-link" routerLink="/withdraw"            >Withdraw Money<span class="sr-only"></span          ></a>        </li>        <li class="nav-item">          <a class="nav-link" routerLink="/transfer"            >Transfer Money<span class="sr-only"></span          ></a>        </li>        <li class="nav-item">          <a class="nav-link" routerLink="/chequebookRequest"            >Request Checkbook<span class="sr-only"></span          ></a>        </li>        <li class="nav-item dropdown">          <a            class="nav-link dropdown-toggle"            href="#"            id="dropdown04"            data-toggle="dropdown"            aria-haspopup="true"            aria-expanded="false"            >User Session</a          >          <div class="dropdown-menu" aria-labelledby="dropdown04">            <a class="dropdown-item" routerLink="/editProfile"              >Change Profile Settings</a            >            <a class="dropdown-item" routerLink="/login">Logout</a>          </div>        </li>      </ul>    </div>  </nav>  <div class="container">    <h3>Edit Profile</h3>    <br />    <form [formGroup]="updateForm" (ngSubmit)="update()">      <div class="col-xs-4" style="left: 40%">        <div class="form-group">          <label for="phone">Mobile no</label>          <input            type="text"            formControlName="phone"            class="form-control"            [ngClass]="{ 'is-invalid': submitted && fval.phone.errors }"            placeholder="Enter Phone here"          />          <div \*ngIf="submitted && fval.phone.errors" class="invalid-feedback">            <div \*ngIf="fval.phone.errors.required">Mobile no is required</div>          </div>        </div>        <div class="form-group">          <label for="address">Address</label>          <input            type="text"            formControlName="address"            class="form-control"            [ngClass]="{ 'is-invalid': submitted && fval.address.errors }"            placeholder="Enter your Address"          />          <div \*ngIf="submitted && fval.address.errors" class="invalid-feedback">            <div \*ngIf="fval.address.errors.required">Address is required</div>          </div>          <br />          <div class="form-group">            <label for="email">Email</label>            <input              type="text"              formControlName="email"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.email.errors }"              placeholder="Enter email here"            />            <div \*ngIf="submitted && fval.email.errors" class="invalid-feedback">              <div \*ngIf="fval.email.errors.required">Email is required</div>              <div \*ngIf="fval.email.errors.email">Enter valid email address</div>            </div>          </div>          <div class="form-group">            <label for="password"> Previous Password</label>            <input              type="password"              formControlName="prevpassword"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.prevpassword.errors }"              placeholder="Enter Password here"            />            <div              \*ngIf="submitted && fval.prevpassword.errors"              class="invalid-feedback"            >              <div \*ngIf="fval.prevpassword.errors.required">                Password is required              </div>              <div \*ngIf="fval.prevpassword.errors.minlength">                Password must be at least 8 scharacters              </div>            </div>          </div>          <div class="form-group">            <label for="password">New password</label>            <input              type="password"              formControlName="password"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.password.errors }"              placeholder="Enter email here"            />            <div              \*ngIf="submitted && fval.password.errors"              class="invalid-feedback"            >              <div \*ngIf="fval.password.errors.required">                Password is required              </div>              <div \*ngIf="fval.password.errors.minlength">                Enter valid password              </div>            </div>          </div>        </div>        <div class="form-group" style="text-align: center">          <button [disabled]="loading" class="btn btn-primary">            <span              \*ngIf="loading"              class="spinner-border spinner-border-sm mr-1"            ></span>            Update          </button>        </div>      </div>    </form>  </div>  <div>    <router-outlet></router-outlet>  </div>  import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators, } from '@angular/forms';  import { Router } from '@angular/router';  import{UpdateService} from '../update.service';  import { UserUpdate } from '../models/userupdate';  import Swal from 'sweetalert2';  @Component({    selector: 'app-edit-profile',    templateUrl: './edit-profile.component.html',    styleUrls: ['./edit-profile.component.css']  })  export class EditProfileComponent implements OnInit {    username:string=localStorage.getItem("username");    constructor(      private formBuilder: FormBuilder,      private router: Router,      private updateService: UpdateService      ) { }    updateForm: FormGroup;    loading = false;    submitted = false;    ngOnInit() {      this.updateForm = this.formBuilder.group({        phone: [''],        address:[''],        email: ['', [Validators.email]],        prevpassword: ['', [Validators.minLength(6)]],        password:['',[Validators.minLength(6)]]    });    }    get fval() { return this.updateForm.controls; }    update(){      this.submitted = true;      // return for here if form is invalid      if (this.updateForm.invalid) {        return;      }      this.loading = true;          const result:UserUpdate= Object.assign({}, this.updateForm.value);          try{            this.updateService.update(this.username,result.phone,result.email,result.address,result.prevpassword,result.password).subscribe(              (data : any) =>{                // localStorage.clear();                // localStorage.setItem('user',JSON.stringify(data));                console.log(data);                this.loading=false;                if(data.flag==true){                  Swal.fire(                    {                      icon: 'success',                      title: 'Profile updated successfully!',                      text:data.message                    }                  )                }else{                  Swal.fire({                    icon: 'error',                    title: 'Oops...',                    text: data.message,                  })                }                this.router.navigate(['/editProfile']);                }            );          }catch{            this.loading=false;            Swal.fire({              icon: 'error',              title: 'Oops...',              text: "Something went wrong!"            })          }      }  } |

|  |
| --- |
| .container{      margin: 3rem auto;      width:20em;    }      .input{      margin-bottom: 4rem ;      margin-top: 2rem;    } |

|  |
| --- |
| //models  export class CheckBookRequest{        accNo:string;      pages:string;     } |

|  |
| --- |
| export class ChequebookResponse{      status:boolean;      responseMessage:String;      account:number;  } |

|  |
| --- |
| export \* from './user'; |

|  |
| --- |
| export class SavingAccount{      id:number;      accno:number;      balance:number;      username:String;      user:null;  } |

|  |
| --- |
| export class Transaction{      date:Date;      id:number;      action:String;      amount:number;  } |

|  |
| --- |
| export class TransferHistory{      id:number;      saccount:string;      raccount:string;      amount:number;      date:Date;  } |

|  |
| --- |
| export class User {      firstName: string;      lastName: string;      userName:string;      password: string;      dob:Date;      phone: number;      address:string;      type:string;      identity:string;      email: string;      }  export class Login {      username:string;      password: string;    } |

|  |
| --- |
| export class UserDisplay{      primaryAccno:number;      savingsAccno:number;      primaryBalance:number;      savingsBalance:number;  } |

|  |
| --- |
| export class UserDisplay{      primaryAccno:number;      savingsAccno:number;      primaryBalance:number;      savingsBalance:number;  } |

|  |
| --- |
| //register component  <div class="container">      <form [formGroup]="registerForm" (ngSubmit)="onFormSubmit()">        <div class="col-xs-4" style="left: 40%">          <br />          <h3 style="text-align: center">User Registration</h3>          <br />            <div class="form-group">            <label for="firstName">First Name</label>            <input              type="text"              formControlName="firstName"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.firstName.errors }"              placeholder="Enter First Name here"            />            <div              \*ngIf="submitted && fval.firstName.errors"              class="invalid-feedback"            >              <div \*ngIf="fval.firstName.errors.required">                First Name is required              </div>            </div>          </div>            <div class="form-group">            <label for="lastName">Last Name</label>            <input              type="text"              formControlName="lastName"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.lastName.errors }"              placeholder="Enter Last Name"            />            <div \*ngIf="submitted && fval.lastName.errors" class="invalid-feedback">              <div \*ngIf="fval.lastName.errors.required">LastName is required</div>            </div>          </div>            <div class="form-group">            <label for="userName">Username</label>            <input              type="text"              formControlName="userName"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.userName.errors }"              placeholder="Enter User Name"            />            <div \*ngIf="submitted && fval.userName.errors" class="invalid-feedback">              <div \*ngIf="fval.userName.errors.required">UserName is required</div>            </div>          </div>            <div class="form-group">            <label for="password"> Password</label>            <input              type="password"              formControlName="password"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.password.errors }"              placeholder="Enter Password here"            />            <div \*ngIf="submitted && fval.password.errors" class="invalid-feedback">              <div \*ngIf="fval.password.errors.required">Password is required</div>              <div \*ngIf="fval.password.errors.minlength">                Password must be at least 8 scharacters              </div>            </div>          </div>            <div class="form-group">            <label for="dob">DOB</label>            <input              type="date"              formControlName="dob"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.dob.errors }"            />            <div \*ngIf="submitted && fval.dob.errors" class="invalid-feedback">              <div \*ngIf="fval.dob.errors.required">Date of Birth is required</div>            </div>          </div>            <div class="form-group">            <label for="phone">Mobile No</label>            <input              type="text"              formControlName="phone"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.phone.errors }"              placeholder="Enter Phone here"            />            <div \*ngIf="submitted && fval.phone.errors" class="invalid-feedback">              <div \*ngIf="fval.phone.errors.required">Phone is required</div>            </div>          </div>            <div class="form-group">            <label for="address">Address</label>            <input              type="text"              formControlName="address"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.address.errors }"              placeholder="Enter your Address"            />            <div \*ngIf="submitted && fval.address.errors" class="invalid-feedback">              <div \*ngIf="fval.address.errors.required">Address is required</div>            </div>          </div>            <div class="form-group">            <label for="type">Choose ID Type:</label>            <select              class="form-control"              formControlName="identityType"              class="form-control"              [ngClass]="{                'is-invalid': submitted && fval.identityType.errors?.required              }"              (change)="$any($event.target).value"            >              <option \*ngFor="let t of identityType" [value]="t.value">                {{ t.name }}              </option>            </select>          </div>            <div class="form-group">            <label for="identity">Enter Id Proof no:</label>            <input              type="text"              formControlName="identity"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.identity.errors }"              placeholder="Enter Id Proof no: "            />            <div \*ngIf="submitted && fval.identity.errors" class="invalid-feedback">              <div \*ngIf="fval.identity.errors.required">Required</div>            </div>          </div>            <div class="form-group">            <label for="email">Email</label>            <input              type="text"              formControlName="email"              class="form-control"              [ngClass]="{ 'is-invalid': submitted && fval.email.errors }"              placeholder="Enter email here"            />            <div \*ngIf="submitted && fval.email.errors" class="invalid-feedback">              <div \*ngIf="fval.email.errors.required">Email is required</div>              <div \*ngIf="fval.email.errors.email">Enter valid email address</div>            </div>          </div>            <div class="center1" style="text-align: center">            <button class="btn btn-primary">Register</button>          </div>        </div>      </form> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { Router } from '@angular/router';  import { User } from '../models';  import { RegisterService } from '../register.service';  import Swal from 'sweetalert2';  @Component({    selector: 'app-register',    templateUrl: './register.component.html',  })  export class RegisterComponent implements OnInit {    constructor(      private formBuilder: FormBuilder,      private router: Router,      private registerService: RegisterService    ) {}    identityType = [      { name: 'Aadhar Card', value: 'aadhar' },      { name: 'PAN card', value: 'pancard' },      { name: 'Passport', value: 'passport' },      { name: 'Voter Id Card', value: 'voter' },    ];    registerForm: FormGroup;    loading = false;    submitted = false;    selectedOption: string;    ngOnInit() {      this.registerForm = this.formBuilder.group({        firstName: ['', Validators.required],        lastName: ['', Validators.required],        userName: ['', Validators.required],        password: ['', [Validators.required, Validators.minLength(6)]],        dob: ['', Validators.required],        phone: ['', Validators.required],        address: ['', Validators.required],        identityType: ['', Validators.required],        file: [''],        identity: ['', Validators.required],        email: ['', [Validators.required, Validators.email]],      });    }    get fval() {      return this.registerForm.controls;    }    onFormSubmit() {      this.submitted = true;      // return for here if form is invalid      if (this.registerForm.invalid) {        return;      }      this.loading = true;      const result: User = Object.assign({}, this.registerForm.value);      // Do useful stuff with the gathered data      console.log(result.firstName);      console.log(this.selectedOption);      this.registerService        .insertUser(          result.firstName,          result.lastName,          result.userName,          result.password,          result.dob,          result.phone,          result.address,          this.selectedOption,          result.identity,          result.email        )        .subscribe((data: any) => {          this.loading = false;          localStorage.clear();          localStorage.setItem('user', JSON.stringify(data));          if (data.registrationStatus) {            Swal.fire({              icon: 'success',              title: 'User registered succesfully!',              text: 'Please wait for an email for account activation!',            });          } else {            Swal.fire({              icon: 'error',              title: 'Oops...',              text: data.responseMessage,            });          }          this.router.navigate(['/login']);        });    }    filterSelected(selectedOption) {      this.selectedOption = selectedOption;      console.log('selected value = ' + selectedOption);    }  } |

|  |
| --- |
| // transaction-history  <div class="container">    <h3>Transaction History</h3>    <div class="card">      <div class="card-body">        <div class="container text-center">          <div class="row">            <div class="col">              <label> Account Type:</label> Saving            </div>            <div class="col">              <label> Saving Balance :</label> {{ savingBalance }}            </div>            <div class="col">              <label> Account Number :</label> {{ accNo }}            </div>          </div>        </div>        <hr/>        <ag-grid-angular class="ag-theme-alpine" [rowData]="transactionList" [columnDefs]="columnDefs"          style="width: auto; height: 500px" [defaultColDef]="defaultColDef"  [paginationAutoPageSize]="true"          [pagination]="true"></ag-grid-angular>      </div>    </div>    <br />  </div> |

|  |
| --- |
| import { Component, OnInit, ViewChild } from '@angular/core';  import { TransactionService } from '../transaction.service';  import { Transaction } from '../models/transaction';  import { AgGridAngular } from 'ag-grid-angular';  import { ColDef } from 'ag-grid-community';  @Component({    selector: 'app-transaction-history',    templateUrl: './transaction-history.component.html',    styleUrls: ['./transaction-history.component.css']  })  export class TransactionHistoryComponent implements OnInit {    username = localStorage.getItem("username");    accNo = JSON.parse(localStorage.getItem("savingAccNo"));    public savingBalance: number;    public transactionList: Array<Transaction>;    public columnDefs: ColDef[] = [      { field: "date" }, { field: "id" }, { field: "action" }, { field: "amount" }    ];    public defaultColDef: ColDef = {      sortable: true,      filter: true,    };    @ViewChild(AgGridAngular) agGrid!: AgGridAngular;      constructor(private transactionService: TransactionService) { }    ngOnInit(): void {      this.transactionService.getTransactions(this.accNo).subscribe((res: Transaction[]) => {        this.transactionList = res;      });      this.transactionService.getSavingAccount(this.username).subscribe(res => {        this.savingBalance = res.balance;      });    }  } |

|  |
| --- |
| .card-body label{    font-weight: bold;  } |

|  |
| --- |
| // transfer-between-account component  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <div class="container">    <form [formGroup]="transferForm" (ngSubmit)="transfer()" method="post">      <div class="col-xs-4" style="left: 35%">        <h3 style="text-align: center">Transfer Money</h3>      </div>      <br />      <div class="form-group" style="text-align: center">        <label>Account Number : </label> <label>{{ saccountno }} </label>      </div>      <div class="form-group">        <label>Enter the IFSC Code of the account to be transfered to:</label>        <br />        <input          type="text"          formControlName="ifscNo"          class="form-control"          [ngClass]="{ 'is-invalid': submitted && fval.ifscNo.errors }"          placeholder="Enter Receivers IFSC Code"        />        <div \*ngIf="submitted && fval.ifscNo.errors" class="invalid-feedback">          <div \*ngIf="fval.ifscNo.errors.required">IFSC Code is required</div>        </div>      </div>      <div class="form-group">        <label>Enter the account number you would like to transfer To:</label>        <br />        <input          type="text"          formControlName="raccountNo"          class="form-control"          [ngClass]="{ 'is-invalid': submitted && fval.raccountNo.errors }"          placeholder="Enter Receiver's account number"        />        <div \*ngIf="submitted && fval.raccountNo.errors" class="invalid-feedback">          <div \*ngIf="fval.raccountNo.errors.required">            Account number is required          </div>        </div>      </div>      <div class="form-group">        <label>Enter Amount:</label>        <br />        <input          type="text"          formControlName="amount"          class="form-control"          [ngClass]="{ 'is-invalid': submitted && fval.amount.errors }"          placeholder="Enter amount"        />        <div \*ngIf="submitted && fval.amount.errors" class="invalid-feedback">          <div \*ngIf="fval.amount.errors.required">Amount is required</div>        </div>      </div>      <div class="center1">        <button          [disabled]="loading"          class="btn btn-primary"          class="btn btn-info form-control"        >          <span            \*ngIf="loading"            class="spinner-border spinner-border-sm mr-1"          ></span>          <a (click)="transfer()" style="color: white">Transfer</a>        </button>      </div>    </form>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators, } from '@angular/forms';  import { Router } from '@angular/router';  import { TransferService } from '../transfer.service';  import Swal from 'sweetalert2';  @Component({    selector: 'app-transfer-between-accounts',    templateUrl: './transfer-between-accounts.component.html',    styleUrls: ['./transfer-between-accounts.component.css']  })  export class TransferBetweenAccountsComponent implements OnInit {    constructor( private formBuilder: FormBuilder,      private router: Router,      private transferService: TransferService) { }    transferForm: FormGroup;    loading = false;    submitted = false;      ngOnInit() {       var username:String=localStorage.getItem("username");     var accNo=+localStorage.getItem("savingAccNo");     console.log(accNo)     console.log(username)     this.transferForm = this.formBuilder.group({        username : username,        saccountNo: accNo,        ifscNo: ['', [Validators.required, Validators.minLength(8)]],        raccountNo: ['', [Validators.required]],        amount:['',[Validators.required]]      });  }  get saccountno(): any {    return localStorage.getItem('savingAccNo');  }  get fval() { return this.transferForm.controls; }    transfer(){      this.submitted = true;      if (this.transferForm.invalid) {        return;      }      this.loading = true;      const result:any = Object.assign({}, this.transferForm.value);          // Do useful stuff with the gathered data      try{        this.transferService.insertEntry(result.username,result.saccountNo,result.ifscNo,result.raccountNo,result.amount).subscribe(          (data : any) =>{           this.loading=false;           if(data.transferStatus==true){            Swal.fire({              icon: 'success',              title: 'Transaction successful',              text:data.responseMessage            })           }           else{            Swal.fire({              icon: 'error',              title: 'Oops...',              text: data.responseMessage,            })           }           }         );      }catch{        this.loading=false;      }      }  }  .container{      margin: 10rem auto;      width:30rem;   }     .input{      margin-bottom: 2rem ;      margin-top: 2rem;  } |

|  |
| --- |
| // transfer-history component  <table class="table table-bordered-md" style="overflow-x: auto">    <thead>      <tr>        <th style="text-align: center">Date</th>        <th style="text-align: center">Transaction Id</th>        <th style="text-align: center">Amount</th>        <th style="text-align: center">Sender Account Number</th>        <th style="text-align: center">Receiver Account Number</th>      </tr>    </thead>    <tbody>      <tr \*ngFor="let transfer of transferList">        <td style="text-align: center">{{ transfer.date }}</td>        <td style="text-align: center">{{ transfer.id }}</td>        <td style="text-align: center">{{ transfer.amount }}</td>        <td style="text-align: center">{{ transfer.saccount }}</td>        <td style="text-align: center">{{ transfer.raccount }}</td>      </tr>    </tbody>  </table> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import {TransferHistory} from '../models/transferhistory'  import { TransferhistoryService } from '../transfer-history.service';  @Component({    selector: 'app-transfer-history',    templateUrl: './transfer-history.component.html',    styleUrls: ['./transfer-history.component.css']  })  export class TransferHistoryComponent implements OnInit {    private accNo:number =+localStorage.getItem("savingAccNo");    public transferList:Array<TransferHistory>;    constructor(private transferService:TransferhistoryService) { }    ngOnInit(): void {      this.transferService.getTransferHistory(this.accNo).subscribe(res=>{        this.transferList=res;      });    }  } |

|  |
| --- |
| @media  only screen and (max-width: 1000px) {    table, thead, tbody, th, td, tr,nav {      display: block;    }    thead tr {      position: absolute;      top: -9999px;      left: -9999px;    }    tr { border: 1px solid #ccc; }    td {      border: none;      border-bottom: 1px solid #eee;      position: relative;      padding-left: 100px;      margin-left: 150px;    }    td:before {      position: absolute;      top: 12px;      left: 6px;      width: 200px;      padding-right: 40px;      white-space: nowrap;      margin-left: -150px;    }    td:nth-of-type(1):before { content: "Date"; }    td:nth-of-type(2):before { content: "Transaction Id"; }    td:nth-of-type(3):before { content: "Amount"; }    td:nth-of-type(4):before { content: "Sender Account Number"; }    td:nth-of-type(5):before { content: "Receiver Account Number"; }    } |

|  |
| --- |
| //withdraw component  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <div class="container">    <form [formGroup]="withdrawForm" (ngSubmit)="withdraw()" method="post">      <div class="col-xs-4" style="left: 35%">        <h3 style="text-align: center">Withdraw Money</h3>      </div>      <br />      <div class="form-group" style="text-align: center">        <label>Account Number : </label> <label>{{ saccountno }} </label>      </div>      <div class="form-group">        <label>Enter Amount:</label>        <br />        <input          type="text"          formControlName="amount"          class="form-control"          [ngClass]="{ 'is-invalid': submitted && fval.amount.errors }"          placeholder="Enter amount"        />        <div \*ngIf="submitted && fval.amount.errors" class="invalid-feedback">          <div \*ngIf="fval.amount.errors.required">Amount is required</div>        </div>      </div>      <div class="center1">        <button          [disabled]="loading"          class="btn btn-primary"          class="btn btn-info form-control"        >          <span            \*ngIf="loading"            class="spinner-border spinner-border-sm mr-1"          ></span>          <a (click)="withdraw()" style="color: white">Withdraw</a>        </button>      </div>    </form>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { Router } from '@angular/router';  import Swal from 'sweetalert2';  import { WithdrawService } from '../withdraw.service';  @Component({    selector: 'app-withdraw',    templateUrl: './withdraw.component.html',    styleUrls: ['./withdraw.component.css'],  })  export class WithdrawComponent implements OnInit {    constructor(      private formBuilder: FormBuilder,      private router: Router,      private withdrawService: WithdrawService    ) {}    withdrawForm: FormGroup;    loading = false;    submitted = false;    ngOnInit(): void {      var accNo = +localStorage.getItem('savingAccNo');      console.log(accNo);      this.withdrawForm = this.formBuilder.group({        account: accNo,        amount: ['', [Validators.required]],      });    }    get saccountno(): any {      return localStorage.getItem('savingAccNo');    }    get fval() {      return this.withdrawForm.controls;    }    withdraw() {      this.submitted = true;      if (this.withdrawForm.invalid) {        return;      }      this.loading = true;      const result: any = Object.assign({}, this.withdrawForm.value);      // Do useful stuff with the gathered data      try {        this.withdrawService          .insertEntry(result.account, +result.amount)          .subscribe((data: any) => {            this.loading = false;            if (data.withdrawStatus == true) {              Swal.fire({                icon: 'success',                title: 'Transaction successful',                text: data.responseMessage,              });            } else {              Swal.fire({                icon: 'error',                title: 'Oops...',                text: data.responseMessage,              });            }          });      } catch {        this.loading = false;      }    }  } |

|  |
| --- |
| .container {      margin: 10rem auto;      width: 30rem;    }      .input {      margin-bottom: 2rem;      margin-top: 2rem;    } |

|  |
| --- |
| //app component  <app-header></app-header>  <div class="app-box">      <router-outlet></router-outlet>  </div>  <app-footer></app-footer> |

|  |
| --- |
| //app.module.ts  import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';  import { ReactiveFormsModule } from '@angular/forms';  import { HttpClientModule } from '@angular/common/http';  import { BrowserAnimationsModule } from '@angular/platform-browser/animations';  import { AuthService } from './auth.service';  import { LoginService } from './login.service';  import { RegisterService } from './register.service';  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { LoginComponent } from './login/login.component';  import { RegisterComponent } from './register/register.component';  import { HomeComponent } from './home/home.component';  import { ChequeBookRequestComponent } from './cheque-book-request/cheque-book-request.component';  import { TransactionHistoryComponent } from './transaction-history/transaction-history.component';  import { TransferBetweenAccountsComponent } from './transfer-between-accounts/transfer-between-accounts.component';  import { EditProfileComponent } from './edit-profile/edit-profile.component';  import { TransferHistoryComponent } from './transfer-history/transfer-history.component';  import { DepositComponent } from './deposit/deposit.component';  import { WithdrawComponent } from './withdraw/withdraw.component';  import { FooterComponent } from './footer/footer.component';  import { HeaderComponent } from './header/header.component';  import { AgGridModule } from 'ag-grid-angular';  @NgModule({    declarations: [      AppComponent,      LoginComponent,      RegisterComponent,      HomeComponent,      ChequeBookRequestComponent,      TransactionHistoryComponent,      TransferBetweenAccountsComponent,      EditProfileComponent,      TransferHistoryComponent,      DepositComponent,      WithdrawComponent,      FooterComponent,      HeaderComponent    ],    imports: [      BrowserModule,      AppRoutingModule,      ReactiveFormsModule,      HttpClientModule,      BrowserAnimationsModule,      AgGridModule    ],    providers: [RegisterService, LoginService, AuthService],    bootstrap: [AppComponent]  })  export class AppModule { } |

|  |
| --- |
| // app-routing.module  import { NgModule } from '@angular/core';  import { Routes, RouterModule } from '@angular/router';  /\*\*Componenets \*/  import { LoginComponent } from './login/login.component';  import { RegisterComponent } from './register/register.component';  import { HomeComponent } from './home/home.component';  import { ChequeBookRequestComponent } from './cheque-book-request/cheque-book-request.component';  import { TransferBetweenAccountsComponent } from './transfer-between-accounts/transfer-between-accounts.component';  import { TransactionHistoryComponent } from './transaction-history/transaction-history.component';  import { EditProfileComponent } from './edit-profile/edit-profile.component';  import { AuthGuard } from './auth.guard.service';  import { TransferHistoryComponent } from './transfer-history/transfer-history.component';  import { DepositComponent } from './deposit/deposit.component';  import { WithdrawComponent } from './withdraw/withdraw.component';  const routes: Routes = [    {      path: 'login',      component: LoginComponent    },    {      path: 'register',      component: RegisterComponent    },    {      path: 'transfer',      component: TransferBetweenAccountsComponent,      canActivate: [AuthGuard],    },    {      path: 'deposit',      component: DepositComponent,      canActivate: [AuthGuard],    },    {      path: 'withdraw',      component: WithdrawComponent,      canActivate: [AuthGuard],    },    {      path: 'transactionHistory',      component: TransactionHistoryComponent,      canActivate: [AuthGuard],    },    {      path: 'chequebookRequest',      component: ChequeBookRequestComponent,      canActivate: [AuthGuard],    },    {      path: 'home',      component: HomeComponent,      canActivate: [AuthGuard]    },    {      path: 'transferHistory',      component: TransferHistoryComponent,      canActivate: [AuthGuard],    },    {      path: 'editProfile',      component: EditProfileComponent,      canActivate: [AuthGuard],    },    { path: '\*\*', redirectTo: '/login' },  ];  @NgModule({    imports: [RouterModule.forRoot(routes)],    exports: [RouterModule],  })  export class AppRoutingModule { } |

|  |
| --- |
| Services  Auth.guard.service.ts  import { Injectable } from '@angular/core';  import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';  import { Observable } from 'rxjs';  import { AuthService } from './auth.service';  @Injectable({    providedIn: 'root'  })  export class AuthGuard implements CanActivate {    constructor(private authService: AuthService, private router: Router) {    }    canActivate(\_next: ActivatedRouteSnapshot, \_state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {      if (this.authService.isLoggedIn) {        return true;      }      this.router.navigate(['']);      return false;    }  } |

|  |
| --- |
| // Auth.service.ts  import { Injectable } from '@angular/core';  import { Router } from '@angular/router';  import { BehaviorSubject } from 'rxjs';  @Injectable({    providedIn: 'root'  })  export class AuthService {    private loggedInStatus = JSON.parse(localStorage.getItem('login')) || false;    private isAuthenticated: BehaviorSubject<boolean> = new BehaviorSubject<boolean>(this.loggedInStatus);    private loggedInUser = localStorage.getItem('username') || "Guest";    private userInfo: BehaviorSubject<string> = new BehaviorSubject<string>(this.loggedInUser);    get isLoggedIn() {      return this.isAuthenticated.asObservable();    }    get userName() {      return this.userInfo.asObservable();    }    constructor(private router: Router) {    }    setUserName(user: string) {      this.userInfo.next(user);    }    authenticate(isAuthUser: boolean) {      if (isAuthUser) {        this.isAuthenticated.next(true);      } else {        this.isAuthenticated.next(false);        this.router.navigate(['/login']);      }    }  } |

|  |
| --- |
| // deposit.service.ts  import { HttpClient } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { environment } from '../environments/environment';  @Injectable({    providedIn: 'root',  })  export class DepositService {    constructor(private http: HttpClient) {}    insertEntry(account: string, amount: number) {      const body = {        account: account,        amount: amount,      };      console.log(body);      return this.http.post(environment.baseUrl + '/account/deposit', body);    }  } |

|  |
| --- |
| // login.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { environment } from '../environments/environment';  @Injectable({    providedIn: 'root',  })  export class LoginService {    constructor(private http: HttpClient) { }    loginUser(userName: string, password: string) {      const body = {        username: userName,        password: password,      };      return this.http.post(environment.baseUrl + '/login', body);    }  } |

|  |
| --- |
| // register.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { environment } from 'src/environments/environment';  @Injectable({    providedIn: 'root',  })  export class RegisterService {    constructor(private http: HttpClient) { }    insertUser(      firstName: string,      lastName: string,      userName: string,      password: string,      dob: Date,      phone: number,      address: string,      identityType: string,      identity: string,      email: string    ) {      const body = {        fname: firstName,        lname: lastName,        username: userName,        password: password,        dob: dob,        phone: phone,        address: address,        identityType: identityType,        identity: identity,        email: email,      };      return this.http.post(environment.baseUrl + '/register', body);    }  } |

|  |
| --- |
| // request.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { ChequebookResponse } from './models/chequebookresponse';  import { environment } from '../environments/environment';  @Injectable({    providedIn: 'root',  })  export class RequestService {    constructor(private http: HttpClient) { }    insertRequest(accNo: number, pages: number = 20) {      const body = {        account: accNo,        no\_of\_pages: pages,      };      console.log(body);      return this.http.post<ChequebookResponse>(        environment.baseUrl+ '/cheque/request',        body      );    }  } |

|  |
| --- |
| // transaction.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Transaction } from './models/transaction';  import { SavingAccount } from './models/savingaccount';  import { environment } from 'src/environments/environment';  import { Observable } from 'rxjs';  @Injectable({    providedIn: 'root',  })  export class TransactionService {    constructor(private http: HttpClient) {      }    public getTransactions(accNo): Observable<Transaction[]> {      return this.http.get<Transaction[]>(environment.baseUrl  + '/account/getHistory/' + accNo      );    }    public getSavingAccount(username): Observable<SavingAccount> {      return this.http.get<SavingAccount>(        environment.baseUrl + '/account/getsaving/' + username      );    }  }  // transfer-history.service.ts  import { Injectable } from '@angular/core';  import { TransferHistory } from './models/transferhistory';  import { HttpClient } from '@angular/common/http';  import { Observable } from 'rxjs';  import { environment } from 'src/environments/environment';  @Injectable({    providedIn: 'root',  })  export class TransferhistoryService {    private url: String;    constructor(private http: HttpClient) { }    public getTransferHistory(accNo): Observable<TransferHistory[]> {      return this.http.get<TransferHistory[]>(        environment.baseUrl+ '/account/getTransfers/' + accNo      );    }    // public getSavingAccount(username):Observable<SavingAccount>{    //   return this.http.get<SavingAccount>(this.url+"/account/getsaving/"+username);    // }  } |

|  |
| --- |
| // transfer.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { environment } from 'src/environments/environment';  @Injectable({    providedIn: 'root',  })  export class TransferService {      constructor(private http: HttpClient) { }    insertEntry(      username: string,      saccount: string,      ifscNo: string,      raccount: string,      amount: number    ) {      const body = {        username: username,        saccount: saccount,        ifsc: ifscNo,        raccount: raccount,        amount: amount,      };      console.log(body);      return this.http.post(environment.baseUrl + '/account/transfer', body);    }  } |

|  |
| --- |
| //update.service.ts  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { environment } from 'src/environments/environment';  @Injectable({    providedIn: 'root',  })  export class UpdateService {      constructor(private http: HttpClient) {}    update(      username: string,      phone: number,      email: string,      address: string,      prevpassword: string,      newpassword: string    ) {      const body = {        username: username,        phone: phone,        email: email,        address: address,        password: prevpassword,        newpassword: newpassword,      };      console.log(body);      return this.http.put( environment.baseUrl+ '/profile/update', body);    }  } |

|  |
| --- |
| // user.service.ts  import { Injectable } from '@angular/core';  import { UserDisplay } from './models/userdisplay';  import { HttpClient } from '@angular/common/http';  import { Observable } from 'rxjs';  import { environment } from '../environments/environment'  @Injectable({    providedIn: 'root',  })  export class UserService {    constructor(private http: HttpClient) {    }    public getUser(username): Observable<UserDisplay> {      return this.http.get<UserDisplay>(environment.baseUrl + '/home/' + username);    }  } |

|  |
| --- |
| // withdraw.service.ts  import { HttpClient } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { environment } from '../environments/environment'  @Injectable({    providedIn: 'root',  })  export class WithdrawService {      constructor(private http: HttpClient) {}    insertEntry(account: string, amount: number) {      const body = {        account: account,        amount: amount,      };      console.log(body);      return this.http.post(environment.baseUrl + '/account/withdraw', body);    }  } |

|  |
| --- |
| //Index.html  <!doctype html>  <html lang="en" class="full-height">  <head>    <meta charset="utf-8">    <title>ICIN Bank</title>    <base href="/">    <meta name="viewport" content="width=device-width, initial-scale=1">    <link rel="icon" type="image/x-icon" href="favicon.ico">    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>    <!-- CSS only -->    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet"      integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">    <link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/css/all.min.css" rel="stylesheet">  </head>  <body>    <app-root></app-root>  </body>  </html> |

|  |
| --- |
| // environments  export const environment = {    production: false,    baseUrl: 'http://localhost:8080'  }; |

|  |
| --- |
| **Admin Portal - fronted** |

|  |
| --- |
| //authorize-registration-component  <table class="table table-bordered">      <thead>        <tr>          <th class="alignment">First Name</th>          <th class="alignment">Last Name</th>          <th class="alignment">Phone Number</th>          <th class="alignment">Address</th>          <th class="alignment">Date of Birth</th>          <th class="alignment">Type of Verification Document</th>          <th class="alignment">Verification ID No.</th>          <th class="alignment">Email</th>          <th class="alignment">Create Account</th>          <th class="alignment">Cancel Account Creation</th>        </tr>      </thead>      <tbody>        <tr \*ngFor="let authorizeuser of authorizeusers">          <td class="alignment">{{authorizeuser.fname}}</td>          <td class="alignment">{{authorizeuser.lname}}</td>          <td class="alignment">{{authorizeuser.phone}}</td>          <td class="alignment">{{authorizeuser.address}}</td>          <td class="alignment">{{authorizeuser.dob}}</td>          <td class="alignment">{{authorizeuser.identityType}}</td>          <td class="alignment">{{authorizeuser.idNumber}}</td>          <td class="alignment">{{authorizeuser.email}}</td>          <td class="alignment"><button type="button" class="btn btn-success" (click)="authorizeAccount(authorizeuser.username)">Create</button> </td>          <td class="alignment"><button type="button" class="btn btn-danger"(click)="rejectRequest(authorizeuser.username)">Cancel</button></td>        </tr>          </tbody>    </table> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { AuthorizationService } from '../service/authorization.service'  import { AuthorizeUser } from '../model/AuthorizeUser';  @Component({    selector: 'app-authorize-registration',    templateUrl: './authorize-registration.component.html',    styleUrls: ['./authorize-registration.component.css']  })  export class AuthorizeRegistrationComponent implements OnInit {    authorizeusers:AuthorizeUser[];    constructor(public authorizeService: AuthorizationService) {      }    ngOnInit(){      this.authorizeService.getRequestData().subscribe(res=>{        console.log(res);        this.authorizeusers = res});    }    authorizeAccount(username){      this.authorizeService.authorizeAccount(username).subscribe(res=>this.ngOnInit());    }    rejectRequest(username){      this.authorizeService.rejectRequest(username).subscribe(res=>this.ngOnInit());    }  } |

|  |
| --- |
| .alignment{      text-align: center;  }  @media  only screen and (max-width: 1000px) {    table, thead, tbody, th, td, tr,nav {      display: block;    }    thead tr {      position: absolute;      top: -9999px;      left: -9999px;    }    tr { border: 1px solid #ccc; }    td {      border: none;      border-bottom: 1px solid #eee;      position: relative;      padding-left: 100px;      margin-left: 150px;    }    td:before {      position: absolute;      top: 12px;      left: 6px;      width: 200px;      padding-right: 40px;      white-space: nowrap;      margin-left: -150px;    }      td:nth-of-type(1):before { content: "First Name"; }    td:nth-of-type(2):before { content: "Last Name"; }    td:nth-of-type(3):before { content: "Phone Number"; }    td:nth-of-type(4):before { content: "Address"; }    td:nth-of-type(5):before { content: "Date of Birth"; }    td:nth-of-type(6):before { content: "Type of Verification Document"; }    td:nth-of-type(7):before { content: "Verification ID"; }    td:nth-of-type(8):before { content: "Create Account"; }    td:nth-of-type(9):before { content: "Cancel Account Creation"; }    } |

|  |
| --- |
| //checkbook-requests  <table class="table table-bordered-md" style="overflow-x:auto;">      <thead>        <tr style="text-align: center">          <th>Request Number</th>          <th>Type of Account</th>          <th>Account Number</th>          <th>Applied Date</th>          <th>Number of Pages</th>          <th> Confirm Request ?</th>        </tr>      </thead>      <tbody>        <tr \*ngFor="let checkbookrequest of checkbookrequests" style="text-align: center">          <td>{{checkbookrequest.id}}</td>          <td>{{checkbookrequest.accType}}</td>          <td>{{checkbookrequest.account}}</td>          <td>{{checkbookrequest.date}}</td>          <td>{{checkbookrequest.no\_of\_pages}}</td>          <td><button (click)="confirmRequest(checkbookrequest.account)" class="btn btn-primary">Confirm Request</button></td>          </tr>        </tbody>    </table> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { CheckbookService } from '../service/checkbook.service'  import { CheckbookRequest } from '../model/CheckbookRequest';  import { NavigationEnd } from '@angular/router';  @Component({    selector: 'app-checkbook-requests',    templateUrl: './checkbook-requests.component.html',    styleUrls: ['./checkbook-requests.component.css']  })  export class CheckbookRequestsComponent implements OnInit {    checkbookrequests: CheckbookRequest[];    term: string;    constructor(public checkbookService: CheckbookService) {    }    ngOnInit() {      this.checkbookService.getRequestsData().subscribe(res => {        this.checkbookrequests = res      });    }    getData() {    }    confirmRequest(account) {      this.checkbookService.confirmCheckbookService(account).subscribe(res => this.ngOnInit());    }  } |

|  |
| --- |
| @media  only screen and (max-width: 1000px) {    table, thead, tbody, th, td, tr,nav {      display: block;    }    thead tr {      position: absolute;      top: -9999px;      left: -9999px;    }    tr { border: 1px solid #ccc; }    td {      border: none;      border-bottom: 1px solid #eee;      position: relative;      padding-left: 100px;      margin-left: 150px;    }    td:before {      position: absolute;      top: 12px;      left: 6px;      width: 200px;      padding-right: 40px;      white-space: nowrap;      margin-left: -150px;    }    td:nth-of-type(1):before { content: "Request Number"; }    td:nth-of-type(2):before { content: "Type of Account"; }    td:nth-of-type(3):before { content: "Account Number"; }    td:nth-of-type(4):before { content: "Applied Date"; }    td:nth-of-type(5):before { content: "Number of Pages"; }    td:nth-of-type(6):before { content: " Confirm Request ?"; }    } |

|  |
| --- |
| //Auth.guard.ts  import { Injectable } from '@angular/core';  import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';  import { Observable } from 'rxjs';  import { AuthenticationService } from '../service/authentication/authentication.service';  @Injectable({    providedIn: 'root'  })  export class AuthGuard implements CanActivate {    constructor(private authenticationService:AuthenticationService,private router:Router){    }    canActivate(      next: ActivatedRouteSnapshot,      state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {      if(this.authenticationService.isAuthenticated){        return true;      }        this.router.navigate(['']);        return false;    }    } |

|  |
| --- |
| //login component  <div class="container">    <h1>Admin Portal</h1>    <form class="form-login" (ngSubmit)="onSubmit(loginForm)" #loginForm="ngForm">      <label for="inputUserName" class="sr-only">UserName</label>      <input type="text" id="inputUserName" name="inputUserName" class="form-control"        [ngClass]="{ 'is-invalid': submitted && inputUserName.errors }" ngModel placeholder="UserName" required autofocus        #inputUserName=ngModel>      <div \*ngIf="submitted && inputUserName.errors" class="invalid-feedback">        <span \*ngIf="inputUserName.errors?.['required']">Please enter a UserName</span>      </div>      <label for="inputPassword" class="sr-only">Password</label>      <input type="password" id="inputPassword" name="password" class="form-control"        [ngClass]="{ 'is-invalid': submitted && password.errors }" ngModel placeholder="Password" required        #password=ngModel>      <div \*ngIf="submitted && password.errors" class="invalid-feedback">        <span \*ngIf="password.errors?.['required']">Please enter a Password</span>      </div>      <br>      <button class="btn btn-lg btn-primary btn-block" type="submit">Login</button>    </form>  </div> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { NgForm } from '@angular/forms';  import { AuthenticationService } from '../service/authentication/authentication.service';  import { LoginData } from '../model/LoginData';  @Component({    selector: 'app-login',    templateUrl: './login.component.html',    styleUrls: ['./login.component.css']  })  export class LoginComponent implements OnInit {    constructor(private authenticationService: AuthenticationService) { }    submitted = false;    ngOnInit(): void {    }      onSubmit(loginForm: NgForm) {      this.submitted = true;      const Logindata = new LoginData(loginForm.value.inputUserName, loginForm.value.password);      this.authenticationService.authenticate(Logindata);    }  } |

|  |
| --- |
| .container{      margin: 10rem auto;      width:20em;   }     input{       margin-bottom: 2rem ;       margin-top: 2rem;   }     html{       background-image: linear-gradient(#009933,white);   } |

|  |
| --- |
| //models  export interface AuthorizeUser{     // id:number;      dob:string;      fname:string;      lname: string;      phone: number;      address: string;      pan: string;      email: string;      username: string;      identityType: string;      idNumber: string  } |

|  |
| --- |
| export class CheckbookRequest{      id:number;      accType: string;      username: string;      account: number;      date: string;      no\_of\_pages: number;  } |

|  |
| --- |
| export class LoginData{      username:string;      password: string;  constructor (username:string, password:string){  this.username= username;  this.password= password;  }  } |

|  |
| --- |
| export interface UserData{      fname:string;      lname: string;      username: string;      email: string;      account: number;      saccount: number;  } |

|  |
| --- |
| //Services  import { Injectable } from '@angular/core';  import { LoginData } from 'src/app/model/LoginData';  import { Router } from '@angular/router';  @Injectable({    providedIn: 'root',  })  export class AuthenticationService {    private readonly adminUser = new LoginData('admin', 'admin@123');    isAuthenticated = false;    constructor(private router: Router) { }    authenticate(login: LoginData): boolean {      if (this.checkCredentials(login)) {        this.isAuthenticated = true;        this.router.navigate(['../user-account']);        return true;      }      alert('Incorrect Login or Password');      this.router.navigate(['../']);      this.isAuthenticated = false;      return false;    }    checkCredentials(login: LoginData): boolean {      return (        this.checkEmail(login.username) && this.checkPassword(login.password)      );    }    checkEmail(email: string): boolean {      return email === this.adminUser.username;    }    checkPassword(password: string): boolean {      return password === this.adminUser.password;    }    logout() {      this.isAuthenticated = false;      this.router.navigate(['']);    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Observable } from 'rxjs';  import { AuthorizeUser } from '../model/AuthorizeUser';  @Injectable({    providedIn: 'root',  })  export class AuthorizationService {    readonly rootUrl ='http://8085/user/';    constructor(private http: HttpClient) {}    getRequestData() {      return this.http.get<AuthorizeUser[]>(this.rootUrl + '/unauthorized/all');    }    authorizeAccount(username) {      return this.http.get(this.rootUrl + username + '/authorize');    }    rejectRequest(username) {      return this.http.get(this.rootUrl + username + '/authorize/cancel');    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Observable } from 'rxjs';  import { CheckbookRequest } from '../model/CheckbookRequest';  @Injectable({    providedIn: 'root',  })  export class CheckbookService {    //readonly rootUrl = 'localhost:<port>/user/{username}/chequebook/request/confirm';    readonly rootUrl =      'http://8085/user/';    //readonly dataUrl= 'localhost:<port>/chequebook/request/all';    readonly dataUrl ='http://8085/chequebook/request/all';    private data: any = [];    constructor(private http: HttpClient) {}    confirmCheckbookService(account) {      return this.http.get(        this.rootUrl + account + '/chequebook/request/confirm'      );    }    getRequestsData(): Observable<CheckbookRequest[]> {      return this.http.get<CheckbookRequest[]>(this.dataUrl);    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  @Injectable({    providedIn: 'root'  })  export class DisableService {    readonly rootUrl ='http://8085/user/';    constructor(private http: HttpClient) {}    disableLoginService(username) {      return this.http.get(this.rootUrl + username + '/disable');    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  @Injectable({    providedIn: 'root'  })  export class EnableService {    readonly rootUrl ='http://8085/user/';    constructor(private http: HttpClient) {}    enableLoginService(username) {      return this.http.get(this.rootUrl + username + '/enable');    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  @Injectable({    providedIn: 'root'  })  export class FeaturesService {    id: number;    readonly rootUrl ='http://8085/user/';    constructor(private http: HttpClient) {}    setFeatures(username, value) {      this.id = value;      console.log(this.id);      return this.http.get(this.rootUrl + username + '/features/' + value);    }  } |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  @Injectable({    providedIn: 'root'  })  export class UsersService {    readonly url = 'http://8085/';    constructor(private http: HttpClient) {}    public getAllUsers() {      return this.http.get<any[]>(this.url + 'user/all');    }  } |

|  |
| --- |
| //user-account component  <table class="table table-bordered-md" style="overflow-x:auto;" caption="">    <thead>      <tr style="text-align: center">        <th>First Name</th>        <th>Last Name</th>        <th style=" padding-bottom: 1.5%;">UserName</th>        <th>Primary Account Number</th>        <th>Savings Account Number</th>        <th>Enable Account</th>        <th>Disable Account</th>        <th style=" padding-bottom: 1.5%;">Features Granted to the User</th>        <th style="padding-bottom: 1.5%;">Change Features</th>      </tr>    </thead>    <tbody>      <tr \*ngFor="let user of users" style="text-align: center">        <td>{{user.fname}}</td>        <td>{{user.lname}}</td>        <td>{{user.username}}</td>        <td>{{user.primaryAccno}}</td>        <td>{{user.savingsAccno}}</td>        <td><button (click)="enableLoginService(user.username)" class="btn btn-success">Enable</button></td>        <td><button (click)="disableLoginService(user.username)" class="btn btn-success">Disable</button></td>        <td>{{user.featureStatus}}</td>        <td style="text-align: center">          <select>            <option \*ngFor="let r of roles" [value]='r.value'>              {{r.name}}            </option>          </select>          <button (click)="setOption(user.username)" class="btn btn-success">Set</button>        </td>      </tr>    </tbody>  </table> |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { EnableService } from '../service/enable.service'  import { DisableService } from '../service/disable.service';  import { FeaturesService } from '../service/features.service';  import { UsersService } from '../service/users.service';  import { UserData } from '../model/UserData';  @Component({    selector: 'app-user-account',    templateUrl: './user-account.component.html',    styleUrls: ['./user-account.component.css']  })  export class UserAccountComponent implements OnInit {    roles = [      { name: " ", value: 0 },      { name: "Deposit Access", value: 1 },      { name: "Deposit + Withdraw Access", value: 2 },      { name: "Deposit + Withdraw + Transfer Access", value: 3 }    ]    users: any[]    //UserData[];    constructor(public enableService: EnableService, public disableService: DisableService, public featuresService: FeaturesService, private service: UsersService) {    }    //selectedOption: string;    //printedOption: number;    selectedValue: number;    ngOnInit() {      this.service.getAllUsers().subscribe(res => {        console.log(res)        //console.log(res[0].featureStatus)        res.forEach(element => {          console.log(element.featureStatus)          // console.log(this.roles[1].name)          if (element.featureStatus == 1) {            element.featureStatus = this.roles[1].name          }          if (element.featureStatus == 2) {            element.featureStatus = this.roles[2].name          }          if (element.featureStatus == 3) {            element.featureStatus = this.roles[3].name          }        });        this.users = res      });    }    // print() {    //   // this.printedOption = this.selectedOption;    //   // console.log(this.selectedOption);    // }    filterSelected(selectedValue) {      this.selectedValue = selectedValue      console.log('selected value= ' + selectedValue);    }    enableLoginService(username) {      console.log(username)      this.enableService.enableLoginService(username).subscribe(res => this.ngOnInit());      //this.enableService.enableLoginService();    }    disableLoginService(username) {      this.disableService.disableLoginService(username).subscribe(res => this.ngOnInit());      //this.disableService.disableLoginService();    }    setOption(username) {      this.featuresService.setFeatures(username, this.selectedValue).subscribe(res => this.ngOnInit());      //this.featuresService.setFeatures();    }    } |

|  |
| --- |
| @media  only screen and (max-width: 1000px) {    table, thead, tbody, th, td, tr,nav {      display: block;    }    thead tr {      position: absolute;      top: -9999px;      left: -9999px;    }    th{      text-align: center;    }    tr { border: 1px solid #ccc; }    td {      border: none;      border-bottom: 1px solid #eee;      position: relative;      padding-left: 100px;      margin-left: 150px;    }    td:before {      position: absolute;      top: 12px;      left: 6px;      width: 200px;      padding-right: 40px;      white-space: nowrap;      margin-left: -150px;    }    td:nth-of-type(1):before { content: "First Name"; }    td:nth-of-type(2):before { content: "Last Name"; }    td:nth-of-type(3):before { content: "UserName"; }    td:nth-of-type(4):before { content: "Current Account Number"; }    td:nth-of-type(5):before { content: "Savings Account Number"; }    td:nth-of-type(6):before { content: "Enable Account"; }    td:nth-of-type(7):before { content: "Disable Account"; }    td:nth-of-type(8):before { content: "Features Granted to the User"; }    td:nth-of-type(9):before { content: "Change Features"; }    }    .select-selected {      background-color: DodgerBlue;    }      /\* Style the arrow inside the select element: \*/    .select-selected:after {      position: absolute;      content: "";      top: 14px;      right: 10px;      width: 0;      height: 0;      border: 6px solid transparent;      border-color: #fff transparent transparent transparent;    }      /\* Point the arrow upwards when the select box is open (active): \*/    .select-selected.select-arrow-active:after {      border-color: transparent transparent #fff transparent;      top: 7px;    }      /\* style the items (options), including the selected item: \*/    .select-items div,.select-selected {      color: #ffffff;      padding: 8px 16px;      border: 1px solid transparent;      border-color: transparent transparent rgba(0, 0, 0, 0.1) transparent;      cursor: pointer;    }      /\* Style items (options): \*/    .select-items {      position: absolute;      background-color: DodgerBlue;      top: 100%;      left: 0;      right: 0;      z-index: 99;    }      /\* Hide the items when the select box is closed: \*/    .select-hide {      display: none;    }      .select-items div:hover, .same-as-selected {      background-color: rgba(0, 0, 0, 0.1);    } |

|  |
| --- |
| // app component  <div \*ngIf="authenticationService.isAuthenticated">    <nav class="navbar navbar-expand-sm navbar-dark bg-dark">      <a class="navbar-brand" routerLink="/user-account">ICIN Bank</a>      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample02"        aria-controls="navbarsExample02" aria-expanded="false" aria-label="Toggle navigation">        <span class="navbar-toggler-icon"></span>      </button>      <div class="collapse navbar-collapse" id="navbarsExample02">        <ul class="navbar-nav mr-auto">          <li class="nav-item">            <a class="nav-link" routerLink="/user-account">User Account Info</a>          </li>          <li class="nav-item">            <a class="nav-link" routerLink="/checkbook-requests">CheckBook Requests</a>          </li>          <li class="nav-item">            <a class="nav-link" routerLink="/authorize">Authorization(KYC)</a>          </li>          <li class="nav-item">            <a class="nav-link" (click)="logout()">Log Out</a>          </li>        </ul>      </div>    </nav>  </div>  <router-outlet></router-outlet> |

|  |
| --- |
| import { Component } from '@angular/core';  import{ AuthenticationService } from 'src/app/service/authentication/authentication.service';  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    title = 'Admin Portal';    constructor(public authenticationService:AuthenticationService) {    }    logout(){    this.authenticationService.logout();  }  } |

|  |
| --- |
| // app.module.ts  import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';  import { FormsModule, ReactiveFormsModule } from '@angular/forms'  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { LoginComponent } from './login/login.component';  import { EnableService } from './service/enable.service';  import { DisableService } from './service/disable.service';  import { UserAccountComponent } from './user-account/user-account.component';  import { CheckbookRequestsComponent } from './checkbook-requests/checkbook-requests.component';  import { HttpClientModule } from '@angular/common/http';  import { AuthorizeRegistrationComponent } from './authorize-registration/authorize-registration.component';  @NgModule({    declarations: [      AppComponent,      LoginComponent,      UserAccountComponent,      CheckbookRequestsComponent,      AuthorizeRegistrationComponent    ],    imports: [      BrowserModule,      AppRoutingModule,      FormsModule,      ReactiveFormsModule,      HttpClientModule    ],    providers: [EnableService, DisableService,],    bootstrap: [AppComponent]  })  export class AppModule { } |

|  |
| --- |
| // app-routing.module.ts  import { NgModule } from '@angular/core';  import { Routes, RouterModule } from '@angular/router';  import { LoginComponent } from './login/login.component';  import { UserAccountComponent } from './user-account/user-account.component';  import { CheckbookRequestsComponent } from './checkbook-requests/checkbook-requests.component';  import { AuthGuard } from './guards/auth.guard'  import { AuthorizeRegistrationComponent } from './authorize-registration/authorize-registration.component';  const routes: Routes = [    { path: 'login', component: LoginComponent },    { path: 'user-account', component: UserAccountComponent, canActivate: [AuthGuard] },    { path: 'checkbook-requests', component: CheckbookRequestsComponent, canActivate: [AuthGuard] },    { path: 'authorize', component: AuthorizeRegistrationComponent, canActivate: [AuthGuard] },    { path: '', component: LoginComponent },  ];  @NgModule({    imports: [RouterModule.forRoot(routes)],    exports: [RouterModule]  })  export class AppRoutingModule { } |

|  |
| --- |
| //Index.html  <!doctype html>  <html lang="en" class="full-height">  <head>    <meta charset="utf-8">    <title>TemplateApp</title>    <base href="/">    <meta name="viewport" content="width=device-width, initial-scale=1">    <link rel="icon" type="image/x-icon" href="favicon.ico">    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.0.4/popper.js"></script>    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">  </head>  <body>    <app-root></app-root>  </body>  </html> |