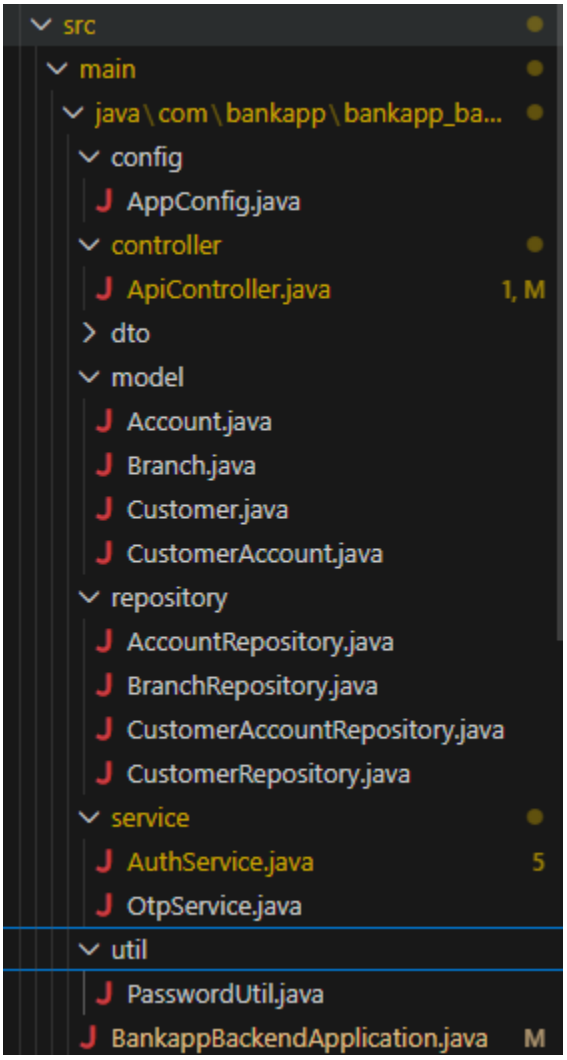
This doc is for you to understand about my file in my Banking Web application paybills features,

It is a spring boot application and frontend is done by html, css, and js

So this Is my file stucture

For the backend



This is source code for every file

AppConfig.java

```java
package com.bankapp.bankapp_backend.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
public class AppConfig {
    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

ApiController.java

```java
package com.bankapp.bankapp_backend.controller;

import com.bankapp.bankapp_backend.dto.LoginRequest;
import com.bankapp.bankapp_backend.model.Account;
import com.bankapp.bankapp_backend.model.Customer;
import com.bankapp.bankapp_backend.model.CustomerAccount;
import com.bankapp.bankapp_backend.repository.AccountRepository;
import com.bankapp.bankapp_backend.repository.CustomerAccountRepository;
import com.bankapp.bankapp_backend.repository.CustomerRepository;
import com.bankapp.bankapp_backend.service.AuthService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import jakarta.servlet.http.HttpSession;

import java.util.*;

@RestController
@RequestMapping("/api")
public class ApiController {
    @Autowired private AuthService authService;
    @Autowired private CustomerRepository customerRepo;
    @Autowired private AccountRepository accountRepo;
```

```
    @Autowired private CustomerAccountRepository customerAccountRepo;

    // LOGIN
    @PostMapping("/auth/login")
    public Map<String,Object> login(@RequestBody LoginRequest req, HttpSession
session) {
        Optional<Customer> c = authService.authenticate(req.getUsername(),
req.getPassword());
        Map<String,Object> resp = new HashMap<>();
        if (c.isPresent()) {
            session.setAttribute("customerId", c.get().getCustomerID());
            resp.put("ok", true);
            resp.put("name", c.get().getName());
            resp.put("email", c.get().getEmail());
            return resp;
        } else {
            resp.put("ok", false);
            resp.put("message", "Username or password incorrect");
            return resp;
        }
    }

    // LOGOUT
    @PostMapping("/auth/logout")
    public Map<String,Object> logout(HttpSession session) {
        session.invalidate();
        return Map.of("ok", true);
    }

    // SEND OTP (for signup or password change)
    @PostMapping("/auth/send-otp")
    public Map<String,Object> sendOtp(@RequestBody Map<String,String> body) {
        String email = body.get("email");
        boolean sent = authService.sendOtpEmail(email);
        return Map.of("ok", sent);
    }

    // VERIFY signup details (name, nic, email, accountNo) with OTP check
    @PostMapping("/auth/verify-signup")
    public Map<String,Object> verifySignup(@RequestBody Map<String,String> body,
HttpSession session) {
        String name = body.get("name");
        String nic = body.get("nic");
        String email = body.get("email");
        String accountNo = body.get("accountNo");
```

```
        String otp = body.get("otp");

        Map<String,Object> resp = new HashMap<>();

        if (!authService.verifyOtp(email, otp)) {
            resp.put("ok", false);
            resp.put("message", "OTP incorrect");
            return resp;
        }

        Optional<Customer> oc = authService.findCustomerByNameNic(name, nic);
        if (oc.isEmpty()) {
            resp.put("ok", false);
            resp.put("message", "Customer details do not match");
            return resp;
        }
        Customer c = oc.get();

        // check if username exists already
        if (c.getUsername() != null) {
            resp.put("ok", false);
            resp.put("message", "User already exists (credentials set).");
            return resp;
        }

        // verify account exists
        Optional<Account> acc = authService.findAccountByAccountNo(accountNo);
        if (acc.isEmpty()) {
            resp.put("ok", false);
            resp.put("message", "Account number not found");
            return resp;
        }

        // ensure the account belongs to the same customer in DB
(Account.CustomerID)
        if (!Objects.equals(acc.get().getCustomerID(), c.getCustomerID())) {
            resp.put("ok", false);
            resp.put("message", "Account does not belong to this customer");
            return resp;
        }

        // temp store customerId in session for next step (create credentials)
        session.setAttribute("signupCustomerId", c.getCustomerID());
        session.setAttribute("signupAccountNo", accountNo);
        resp.put("ok", true);
```

```
        return resp;
    }

    // CREATE CREDENTIALS after verify-signup
    @PostMapping("/auth/create-credentials")
    public Map<String,Object> createCredentials(@RequestBody Map<String,String>
body, HttpSession session) {
        Integer customerId = (Integer) session.getAttribute("signupCustomerId");
        String accountNo = (String) session.getAttribute("signupAccountNo");
        if (customerId == null || accountNo == null) {
            return Map.of("ok", false, "message", "No signup session present");
        }
        String username = body.get("username");
        String password = body.get("password");

        // basic checks omitted for brevity: unique username, password strength
        Optional<Customer> maybe = customerRepo.findByUsername(username);
        if (maybe.isPresent()) return Map.of("ok", false, "message", "Username
already taken");

        boolean created = authService.createCredentialsForCustomer(customerId,
username, password);
        if (!created) return Map.of("ok", false, "message", "Failed to create
credentials");

        // add customer account and mark primary
        authService.addCustomerAccount(customerId, accountNo, true);

        // clear signup session
        session.removeAttribute("signupCustomerId");
        session.removeAttribute("signupAccountNo");

        return Map.of("ok", true);
    }

    // SEND OTP for password reset identity verification
    @PostMapping("/auth/send-otp-for-reset")
    public Map<String,Object> sendOtpForReset(@RequestBody Map<String,String>
body) {
        String email = body.get("email");
        boolean sent = authService.sendOtpEmail(email);
        return Map.of("ok", sent);
    }

    // VERIFY identity details for password reset
```

```
    @PostMapping("/auth/verify-reset")
    public Map<String,Object> verifyReset(@RequestBody Map<String,String> body,
HttpSession session) {
        String name = body.get("name");
        String nic = body.get("nic");
        String email = body.get("email");
        String otp = body.get("otp");

        if (!authService.verifyOtp(email, otp)) return Map.of("ok", false,
"message", "OTP incorrect");

        Optional<Customer> oc = customerRepo.findByNameAndNic(name, nic);
        if (oc.isEmpty()) return Map.of("ok", false, "message", "Customer details
not match");

        // store in session
        session.setAttribute("resetCustomerId", oc.get().getCustomerID());
        return Map.of("ok", true);
    }

    // CHANGE PASSWORD after reset verification
    @PostMapping("/auth/change-password")
    public Map<String,Object> changePassword(@RequestBody Map<String,String>
body, HttpSession session) {
        Integer customerId = (Integer) session.getAttribute("resetCustomerId");
        if (customerId == null) return Map.of("ok", false, "message", "No reset
session");

        String newPassword = body.get("password");
        boolean changed = authService.changePasswordByCustomer(customerId,
newPassword);
        if (changed) {
            session.removeAttribute("resetCustomerId");
            return Map.of("ok", true);
        } else {
            return Map.of("ok", false, "message", "Failed to change password");
        }
    }

    // DASHBOARD: fetch customer and accounts
    @GetMapping("/dashboard/me")
    public Map<String,Object> dashboard(HttpSession session) {
        Integer customerId = (Integer) session.getAttribute("customerId");
        if (customerId == null) return Map.of("ok", false, "message", "Not logged
in");
```

```
        Optional<Customer> oc = customerRepo.findById(customerId);
        if (oc.isEmpty()) return Map.of("ok", false, "message", "Customer not
found");

        Customer c = oc.get();
        List<CustomerAccount> cas =
customerAccountRepo.findByCustomerID(customerId);
        List<Map<String,Object>> accounts = new ArrayList<>();
        // primary first
        cas.sort((a,b) -> Boolean.compare(b.getIsPrimary()!=null &&
b.getIsPrimary(), a.getIsPrimary()!=null && a.getIsPrimary()));
        for (CustomerAccount ca: cas) {
            Optional<Account> aopt = accountRepo.findById(ca.getAccountNo());
            if (aopt.isPresent()) {
                Account a = aopt.get();
                Map<String,Object> m = new HashMap<>();
                m.put("accountNo", a.getAccountNo());
                m.put("accountType", a.getAccountType());
                m.put("balance", a.getAccountBalance());
                m.put("isPrimary", ca.getIsPrimary());
                accounts.add(m);
            }
        }
        return Map.of("ok", true, "name", c.getName(), "email", c.getEmail(),
"accounts", accounts);
    }

}
```

Account.java

```
package com.bankapp.bankapp_backend.model;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name="Account")
public class Account {
    @Id
    @Column(length=6)
```

```
    private String accountNo;

    @Column(nullable=false)
    private Integer customerID; // original owner ID

    @Column(nullable=false)
    private String accountType; // Savings/Current/Fixed Deposit

    @Column(nullable=false)
    private Integer branchID;

    private Double accountBalance = 0.0;

    private String accountStatus = "Active";

    private LocalDateTime createdDate;
    private LocalDateTime lastUpdatedDate;

    @PrePersist
    public void prePersist() {
        createdDate = LocalDateTime.now();
        lastUpdatedDate = LocalDateTime.now();
    }
    @PreUpdate
    public void preUpdate() {
        lastUpdatedDate = LocalDateTime.now();
    }

    // getters & setters
    public String getAccountNo() { return accountNo; }
    public void setAccountNo(String accountNo) { this.accountNo = accountNo; }

    public Integer getCustomerID() { return customerID; }
    public void setCustomerID(Integer customerID) { this.customerID = customerID;
}

    public String getAccountType() { return accountType; }
    public void setAccountType(String accountType) { this.accountType =
accountType; }

    public Integer getBranchID() { return branchID; }
    public void setBranchID(Integer branchID) { this.branchID = branchID; }

    public Double getAccountBalance() { return accountBalance; }
```

```
    public void setAccountBalance(Double accountBalance) { this.accountBalance =
accountBalance; }

    public String getAccountStatus() { return accountStatus; }
    public void setAccountStatus(String accountStatus) { this.accountStatus =
accountStatus; }

    public LocalDateTime getCreatedDate() { return createdDate; }
    public LocalDateTime getLastUpdatedDate() { return lastUpdatedDate; }
}
```

Branch.java

```
package com.bankapp.bankapp_backend.model;

import jakarta.persistence.*;

@Entity
@Table(name="Branch")
public class Branch {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer branchID;

    @Column(nullable=false)
    private String branchName;

    @Column(nullable=false)
    private String city;

    private String postalCode;

    // getters & setters
    public Integer getBranchID() { return branchID; }
    public void setBranchID(Integer branchID) { this.branchID = branchID; }

    public String getBranchName() { return branchName; }
    public void setBranchName(String branchName) { this.branchName = branchName;
}

    public String getCity() { return city; }
    public void setCity(String city) { this.city = city; }
```

```
    public String getPostalCode() { return postalCode; }
    public void setPostalCode(String postalCode) { this.postalCode = postalCode;
}
}
```

Customer.java

```
package com.bankapp.bankapp_backend.model;

import jakarta.persistence.*;

@Entity
@Table(name = "Customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer customerID;

    @Column(nullable=false)
    private String name;

    @Column(nullable=false, unique=true)
    private String email;

    @Column(nullable=false, unique=true)
    private String nic;

    private String phoneNumber;

    @Column(unique=true)
    private String username;

    private String passwordHash;

    // getters and setters
    public Integer getCustomerID() { return customerID; }
    public void setCustomerID(Integer customerID) { this.customerID = customerID;
}

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
```

```
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getNic() { return nic; }
    public void setNic(String nic) { this.nic = nic; }

    public String getPhoneNumber() { return phoneNumber; }
    public void setPhoneNumber(String phoneNumber) { this.phoneNumber =
phoneNumber; }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPasswordHash() { return passwordHash; }
    public void setPasswordHash(String passwordHash) { this.passwordHash =
passwordHash; }
}
```

CustomerAccount.java

```
package com.bankapp.bankapp_backend.model;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name="CustomerAccount")
public class CustomerAccount {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer customerAccountID;

    @Column(nullable=false)
    private Integer customerID;

    @Column(length=6, nullable=false)
    private String accountNo;

    private Boolean isPrimary = false;

    private LocalDateTime addedDate;
```

```
    @PrePersist
    public void prePersist() { addedDate = LocalDateTime.now(); }

    // getters & setters
    public Integer getCustomerAccountID() { return customerAccountID; }
    public void setCustomerAccountID(Integer customerAccountID) {
this.customerAccountID = customerAccountID; }

    public Integer getCustomerID() { return customerID; }
    public void setCustomerID(Integer customerID) { this.customerID = customerID;
}

    public String getAccountNo() { return accountNo; }
    public void setAccountNo(String accountNo) { this.accountNo = accountNo; }

    public Boolean getIsPrimary() { return isPrimary; }
    public void setIsPrimary(Boolean isPrimary) { this.isPrimary = isPrimary; }

    public LocalDateTime getAddedDate() { return addedDate; }
}
```

AccountRepository.java

```
package com.bankapp.bankapp_backend.repository;

import com.bankapp.bankapp_backend.model.Account;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;
import java.util.List;

public interface AccountRepository extends JpaRepository<Account, String> {
    Optional<Account> findByAccountNo(String accountNo);
    List<Account> findByCustomerID(Integer customerId);
}
```

BrachRepository.java

```
package com.bankapp.bankapp_backend.repository;

import com.bankapp.bankapp_backend.model.Branch;
import org.springframework.data.jpa.repository.JpaRepository;
```

```java
public interface BranchRepository extends JpaRepository<Branch, Integer> {
}
```

CustomerAccountRepository.java

```java
package com.bankapp.bankapp_backend.repository;

import com.bankapp.bankapp_backend.model.CustomerAccount;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface CustomerAccountRepository extends JpaRepository<CustomerAccount,
Integer> {
    List<CustomerAccount> findByCustomerID(Integer customerID);
    List<CustomerAccount> findByAccountNo(String accountNo);
}
```

CustomerRepository,java

```java
package com.bankapp.bankapp_backend.repository;

import com.bankapp.bankapp_backend.model.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;

public interface CustomerRepository extends JpaRepository<Customer, Integer> {
    Optional<Customer> findByUsername(String username);
    Optional<Customer> findByEmail(String email);
    Optional<Customer> findByNameAndNic(String name, String nic);
}
```

AuthService.java

```java
package com.bankapp.bankapp_backend.service;

import com.bankapp.bankapp_backend.model.Customer;
import com.bankapp.bankapp_backend.model.CustomerAccount;
```

```java
import com.bankapp.bankapp_backend.model.Account;
import com.bankapp.bankapp_backend.repository.CustomerRepository;
import com.bankapp.bankapp_backend.repository.AccountRepository;
import com.bankapp.bankapp_backend.repository.CustomerAccountRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import java.util.Optional;
import java.util.List;

@Service
public class AuthService {
    @Autowired private CustomerRepository customerRepo;
    @Autowired private AccountRepository accountRepo;
    @Autowired private CustomerAccountRepository customerAccountRepo;
    @Autowired private BCryptPasswordEncoder passwordEncoder;
    @Autowired private JavaMailSender mailSender;
    @Autowired private OtpService otpService;

    public Optional<Customer> authenticate(String username, String rawPassword) {
        Optional<Customer> c = customerRepo.findByUsername(username);
        if (c.isPresent() && c.get().getPasswordHash() != null) {
            if (passwordEncoder.matches(rawPassword, c.get().getPasswordHash()))
return c;
        }
        return Optional.empty();
    }

    public boolean sendOtpEmail(String email) {
        String otp = otpService.generateOtpFor(email);
        try {
            SimpleMailMessage m = new SimpleMailMessage();
            m.setTo(email);
            m.setFrom("novabank.noreply@gmail.com");
            m.setSubject("NovaBank OTP");
            m.setText("Your NovaBank verification code: " + otp + " (expires in 5
minutes)");
            mailSender.send(m);
            return true;
        } catch (Exception ex) {
            ex.printStackTrace();
            return false;
        }
```

```
    }

    public boolean verifyOtp(String email, String otp) {
        return otpService.verifyOtp(email, otp);
    }

    public Optional<Customer> findCustomerByNameNic(String name, String nic) {
        return customerRepo.findByNameAndNic(name, nic);
    }

    public Optional<Account> findAccountByAccountNo(String accountNo) {
        return accountRepo.findByAccountNo(accountNo);
    }

    public boolean createCredentialsForCustomer(Integer customerId, String
username, String rawPassword) {
        Optional<Customer> oc = customerRepo.findById(customerId);
        if (!oc.isPresent()) return false;
        Customer c = oc.get();
        if (c.getUsername() != null) return false; // already has username
        c.setUsername(username);
        c.setPasswordHash(passwordEncoder.encode(rawPassword));
        customerRepo.save(c);
        return true;
    }

    public boolean addCustomerAccount(Integer customerId, String accountNo,
boolean isPrimary) {
        Optional<Account> acc = accountRepo.findByAccountNo(accountNo);
        if (!acc.isPresent()) return false;
        CustomerAccount ca = new CustomerAccount();
        ca.setCustomerID(customerId);
        ca.setAccountNo(accountNo);
        ca.setIsPrimary(isPrimary);
        if (isPrimary) {
            // unset other primaries for this customer
            List<CustomerAccount> list =
customerAccountRepo.findByCustomerID(customerId);
            for (CustomerAccount other : list) {
                if (other.getIsPrimary() != null && other.getIsPrimary()) {
                    other.setIsPrimary(false);
                    customerAccountRepo.save(other);
                }
            }
        }
```

```
        customerAccountRepo.save(ca);
        return true;
    }

    public boolean changePasswordByCustomer(Integer customerId, String
newRawPassword) {
        Optional<Customer> oc = customerRepo.findById(customerId);
        if (!oc.isPresent()) return false;
        Customer c = oc.get();
        c.setPasswordHash(passwordEncoder.encode(newRawPassword));
        customerRepo.save(c);
        return true;
    }

    // fetch dashboard info
    public Customer getCustomer(Integer customerId) {
        return customerRepo.findById(customerId).orElse(null);
    }
}
```

OtpService.java

```
package com.bankapp.bankapp_backend.service;

import org.springframework.stereotype.Service;
import java.time.Instant;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.Random;

@Service
public class OtpService {
    private static class OtpEntry {
        String otp;
        Instant expiresAt;
        OtpEntry(String otp, Instant expiresAt) { this.otp = otp; this.expiresAt
= expiresAt; }
    }

    private final Map<String, OtpEntry> store = new ConcurrentHashMap<>();
    private final Random random = new Random();

    public String generateOtpFor(String email) {
```

```
        String otp = String.format("%06d", random.nextInt(1_000_000));
        Instant expires = Instant.now().plusSeconds(5 * 60); // 5 minutes
        store.put(email, new OtpEntry(otp, expires));
        return otp;
    }

    public boolean verifyOtp(String email, String otp) {
        OtpEntry e = store.get(email);
        if (e == null) return false;
        if (Instant.now().isAfter(e.expiresAt)) {
            store.remove(email);
            return false;
        }
        boolean ok = e.otp.equals(otp);
        if (ok) store.remove(email);
        return ok;
    }
}
```

PasswordUtil.java

```
package com.bankapp.bankapp_backend.util;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class PasswordUtil {
    private static final BCryptPasswordEncoder encoder = new
BCryptPasswordEncoder();

    public static String hash(String raw) {
        return encoder.encode(raw);
    }

    public static boolean matches(String raw, String hashed) {
        return encoder.matches(raw, hashed);
    }
}
```

There is also folder named dto

LoginRequest.java

```java
package com.bankapp.bankapp_backend.dto;

public class LoginRequest {
    private String username;
    private String password;

    // getters / setters
    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; }
}
```

And finally

BankAppBackendApplication.java

```java
package com.bankapp.bankapp_backend;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BankappBackendApplication {
    public static void main(String[] args) {
        SpringApplication.run(BankappBackendApplication.class, args);
    }
}
```

So this is the backend I currently have

Now I will give u the back end

For the paybills.html file

```html
<!DOCTYPE html>
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nova Bank - Bills</title>
    <link rel="stylesheet" href="css/paybills.css">
    <link
href="https://fonts.googleapis.com/css2?family=Bai+Jamjuree:wght@300;400;500;600;
700&display=swap"
        rel="stylesheet">
    <link rel="icon" type="image/png" href="img/favicon.png">
</head>

<body>
    <div class="container">
        <aside class="sidebar">
            <div class="logo-section">
                <div class="logo-circle">
                    <div class="logo-container">
                        <img src="img/j_logo.png" alt="Logo">
                    </div>
                </div>
                <h1 class="bank-name">NOVA BANK</h1>
            </div>

            <nav class="nav-menu">
                <a href="dashboard.html" class="nav-item">
                    <span>DASHBOARD</span>
                </a>
                <a href="accounts.html" class="nav-item">
                    <span>ACCOUNTS</span>
                </a>
                <a href="bank_transfer.html" class="nav-item">
                    <span>BANK TRANSFER</span>
                </a>
                <a href="pay_bills.html" class="nav-item active">
                    <span>PAY BILLS</span>
                </a>
                <a href="e_statement.html" class="nav-item">
                    <span>E-STATEMENT</span>
                </a>
            </nav>

            <div class="sidebar-footer">
```

```
            <button class="icon-btn">
                <img src="img/settings_icon.png" alt="Profile">
            </button>
            <button class="icon-btn" id="logoutBtn">
                <img src="img/logout_icon.png" alt="Profile">
            </button>
        </div>
    </aside>

    <main class="main-content">
        <header class="top-bar">
            <h2 class="page-title">Pay Bills</h2>
            <div class="header-actions">
                <button class="icon-btn">
                    <img src="img/search_icon.png" alt="Profile">
                </button>
                <button class="icon-btn">
                    <img src="img/bell_icon.png" alt="Profile">
                </button>
                <div class="profile-avatar">
                    <img src="img/pfp_img.png" alt="Profile">
                </div>
            </div>
        </header>

        <div class="pb-Container">
            <section class="utilitybtn-section">

                <h3 class="pbhead-text">Utility Billers</h3>

                <div class="utility-row">

                    <!--CEB Button-->
                    <div class="btn-wrapper">
                        <button class="utility-btn" data-form="cebForm">
                            <a href="#">
                                <img class="cebimg"
                                    src="img/cebimg.png"
                                    alt="CEB">
                            </a>
                        </button>
                        <span class="pbbtn-label">CEB</span>
                    </div>

                    <div class="btn-wrapper">
```

```
                        <!--NWSDB Button-->
                        <button class="utility-btn" data-form="nwsdbForm">
                            <a href="#">
                                <img class="nwsdbimg" src="img/nwsdbimg.png"
alt="NWSDB">
                            </a>
                        </button>
                        <span class="pbbtn-label">NWSDB</span>
                    </div>

                    <div class="btn-wrapper">
                        <!--SLT Button-->
                        <button class="utility-btn" data-form="sltform">
                            <a href="#">
                                <img class="sltimg"
                                    src="img/sltimg.png"
                                    alt="SLT">
                            </a>
                        </button>
                        <span class="pbbtn-label">SLT</span>
                    </div>

                    <div class="btn-wrapper">
                        <!--Dialog Button-->
                        <button class="utility-btn" data-form="dialogform">
                            <a href="#">
                                <img class="dialogimg"
                                    src="img/dialogimg.jpeg"
                                    alt="Dialog">

                            </a>
                        </button>
                        <span class="pbbtn-label">Dialog</span>
                    </div>

                    <div class="btn-wrapper">

                </div>


            </section>


            <div class="payinfo">
```

```
                    <!--CEB Form-->
                    <section id="cebForm" class="bill-form">


                        <div class="balance-card">
                            <div class="card-header">
                                <p class="title">Current Balance</p>
                            </div>
                            <div class="card-body">
                                <p class="amount" id="balance">Loading...</p>
                                <p class="account-name"><span
id="accountType"></span></p>
                            </div>
                        </div>
                        <h2 class="pbform-title">Ceylon Electricity Board</h2>

                        <div class="pbform-group">
                            <label for="pbbankAcc">Pay From</label>
                            <select type="text" id="accountSelect"
placeholder="Pay From">
                            </select>
                        </div>

                        <div class="pbform-group">
                            <label for="cebAmount">Amount (LKR)</label>
                            <input type="number" id="cebAmount"
placeholder="Enter payment amount">
                        </div>

                        <div class="pbform-group">
                            <label for="cebbillno">Electricity Bill Account
No</label>
                            <input type="text" id="cebbillno"
placeholder="Electricity Bill Account No">
                        </div>

                        <div class="pbform-group">
                            <label for="cebbillno">Re Enter Electricity Bill
Account Number</label>
                            <input type="text" id="recebbillno"
placeholder="Electricity Bill Account No">
                        </div>
                        <div class="pbcenter">
                            <button class="pbnext-btn"
id="cebnextBtn">Next</button>
```

```
                    </div>
                </section>


                <!--NWSDB Form-->

                <section id="nwsdbForm" class="bill-form">
                    <h2 class="pbform-title">National Water Supply and
Drainage Board</h2>

                    <div class="pbform-group">
                        <label for="pbbankAcc">Pay From</label>
                        <select type="text" id="pbbankacc" placeholder="Enter
CEB account number">
                            <option value="">-- Choose Account --</option>
                        </select>
                    </div>

                    <div class="pbform-group">
                        <label for="wbbillAmount">Amount (LKR)</label>
                        <input type="number" id="wbAmount" placeholder="Enter
payment amount">
                    </div>

                    <div class="pbform-group">
                        <label for="wbbillno">Water Bill Account No</label>
                        <input type="text" id="wbbillno" placeholder="Water
Bill Account No">
                    </div>

                    <div class="pbform-group">
                        <label for="wbbillno">Re Enter Water Bill Account
Number</label>
                        <input type="text" id="rewbbillno" placeholder="Water
Bill Account No">
                    </div>
                    <div class="pbcenter">
                        <button class="pbnext-btn">Next</button>
                    </div>
                </section>

                <!--SLT Form-->

                <section id="sltform" class="bill-form">
                    <h2 class="pbform-title">Sri Lanka Telecom</h2>
```

```
                    <div class="pbform-group">
                        <label for="pbbankAcc">Pay From</label>
                        <select type="text" id="pbbankacc" placeholder="Enter
CEB account number">
                            <option value="">-- Choose Account --</option>
                        </select>
                    </div>

                    <div class="pbform-group">
                        <label for="sltAmount">Amount (LKR)</label>
                        <input type="number" id="wbAmount" placeholder="Enter
payment amount">
                    </div>

                    <div class="pbform-group">
                        <label for="sltbillno">SLT Account No</label>
                        <input type="text" id="wbbillno" placeholder="SLT
Account No">
                    </div>

                    <div class="pbform-group">
                        <label for="sltbillno">Re Enter SLT Account
Number</label>
                        <input type="text" id="rewbbillno" placeholder="SLT
Account No">
                    </div>
                    <div class="pbcenter">
                        <button class="pbnext-btn">Next</button>
                    </div>
                </section>


                <!--Dialog Form-->

                <section id="dialogform" class="bill-form">
                    <h2 class="pbform-title">Dialog</h2>

                    <div class="pbform-group">
                        <label for="pbbankAcc">Pay From</label>
                        <select type="text" id="pbbankacc" placeholder="Enter
CEB account number">
                            <option value="">-- Choose Account --</option>
                        </select>
                    </div>
```

```
                        <div class="pbform-group">
                            <label for="dialogAmount">Amount (LKR)</label>
                            <input type="number" id="dbAmount" placeholder="Enter
payment amount">
                        </div>

                        <div class="pbform-group">
                            <label for="dbillno">Dialog Account No</label>
                            <input type="text" id="wbbillno" placeholder="Water
Bill Account No">
                        </div>

                        <div class="pbform-group">
                            <label for="dbillno">Dialog Account Number</label>
                            <input type="text" id="rewbbillno" placeholder="Water
Bill Account No">
                        </div>
                        <div class="pbcenter">
                            <button class="pbnext-btn">Next</button>
                        </div>
                    </section>


                    <div id="paydetails" class="paydetails">
                        <h3 class="pdtext">Confirm Bill Payment</h3>
                        <div class="paydetailscard">
                            <h4 class="pbhead-text">Details</h4>
                            <hr>
                            <p class="pbdetailsrow"><strong>Date & time
:</strong><span id="bpdate"></span></p>

                            <p class="pbdetailsrow"><strong>Bill No :</strong>
<span id="confirmName"></span></p>
                            <p class="pbdetailsrow"><strong>Amount
:</strong><span id="confirmAmount"></span></p>
                            <hr>
                        </div>
                        <div class="paydetailsbtns">
                            <button type="button" id="backBtn"
class="pbbackbtn">Back</button>
                            <button type="button" id="pbconfirmBtn"
class="pbconfirmbtn">Confirm</button>
                        </div>
```

```
                    </div>

                    <div id="paymentsuccess " class="pbpaymentsuccess">
                        <img src="img/successimg.png" alt="Succeess"
class="paymentsuccessicon">

                        <H2 class="pbh2">Paymemt Successfull!</H2>
                        <h3 class="pbh3">Confirmation number :</h3>
                    </div>

                    <div id="paymentfailed" class="pbpaymentfailed">
                        <img src="img/failedimg.png" alt="Failed"
class="paymentfailedimg">

                        <h2 class="pbh2">Payment Failed!</h2>
                        <h3 class="pbh3"></h3>
                    </div>


                </div>


            </div>




        </main>
    </div>
    <script src="js/paybills.js"></script>
    <script src="js/dashboard.js"></script>
</body>

</html>
```

Paybills.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
```

```
    font-family: 'Bai Jamjuree', sans-serif;
    background-color: #f5f5f5;
    color: #1f2937;
}

.container {
    display: flex;
    min-height: 100vh;
}

.sidebar {
    width: 220px;
    background: #000F2E;
    color: white;
    display: flex;
    flex-direction: column;
    padding: 24px 16px;
    position: relative;
    border-top-right-radius: 30px;
    border-bottom-right-radius: 30px;
}

/* .sidebar::after {
    content: '';
    position: absolute;
    top: 0;
    right: 0;
    width: 100px;
    height: 100%;
    background: linear-gradient(90deg, transparent 0%, rgba(255, 255, 255, 0.03)
100%);
    pointer-events: none;
} */

.logo-section {
    display: flex;
    align-items: center;
    gap: 12px;
    margin-bottom: 48px;
}

.logo-circle {
    width: 48px;
    height: 48px;
    background: white;
```

```
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-shrink: 0;
}

.logo-text {
    font-size: 28px;
    font-weight: 700;
    color: #001f3f;
}

.logo-container {
    width: 48px;
    height: 48px;
    background: #003366;
    /* Circle background */
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    overflow: hidden;
    /* Crop image to the circle */
    position: relative;
    z-index: 1;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);
}

.logo-container img {
    width: 100%;
    height: 100%;
    object-fit: cover;
    /* Makes the image fill the circle nicely */
}

.bank-name {
    font-size: 16px;
    font-weight: 700;
    letter-spacing: 1px;
    white-space: nowrap;
}

.nav-icon-container {
    width: 15px;
```

```
    height: 15px;
}

.nav-icon-container img {
    width: 100%;
    height: 100%;
    object-fit: cover;
    /* Makes the image fill the circle nicely */
}

.nav-menu {
    display: flex;
    flex-direction: column;
    gap: 8px;
    flex: 1;
}

.nav-item {
    display: flex;
    align-items: center;
    gap: 12px;
    padding: 14px 20px;
    color: rgba(255, 255, 255, 0.7);
    text-decoration: none;
    font-size: 13px;
    font-weight: 500;
    letter-spacing: 0.5px;
    border-radius: 50px;
    transition: all 0.3s ease;
}

.nav-item:hover {
    background: rgba(255, 255, 255, 0.05);
    color: white;
}

.nav-item.active {
    background: #5B7AB3;
    color: white;
}

.nav-icon {
    width: 20px;
    height: 20px;
    flex-shrink: 0;
```

```css
}

.sidebar-footer {
    display: flex;
    gap: 12px;
    margin-top: 24px;
}

.icon-btn {
    background: transparent;
    border: none;
    color: rgba(255, 255, 255, 0.6);
    cursor: pointer;
    padding: 8px;
    border-radius: 8px;
    transition: all 0.3s ease;
    display: flex;
    align-items: center;
    justify-content: center;
}

.icon-btn:hover {
    background: rgba(255, 255, 255, 0.1);
    color: white;
}

.icon-btn img {
    width: 20px;
    height: 20px;
}

.main-content {
    flex: 1;
    background: #f5f5f5;
    overflow-y: auto;
}

.top-bar {
    background: white;
    padding: 20px 32px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.05);
}
```

```css
.page-title {
    font-size: 24px;
    font-weight: 800;
    color: #1f2937;
}

.header-actions {
    display: flex;
    align-items: center;
    gap: 16px;
}

.header-actions .icon-btn {
    color: #6b7280;
}

.header-actions .icon-btn:hover {
    background: #f3f4f6;
    color: #1f2937;
}

.profile-avatar {
    width: 40px;
    height: 40px;
}

.profile-avatar img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.content-grid {
    display: grid;
    grid-template-columns: 1fr 280px;
    gap: 24px;
    padding: 24px 32px;
}

.welcome-card {
    background: #E0E4EA;
    border-radius: 16px;
    display: flex;
    justify-content: space-between;
```

```
    align-items: center;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}

.welcome-content {
    flex: 1;
    padding: 32px;
}

.welcome-title {
    font-size: 20px;
    font-weight: 600;
    color: #1f2937;
    margin-bottom: 24px;
}

.balance-card {
    background: #000000;
    border-radius: 12px;
    padding: 24px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 400px;
    margin-bottom: 16px;
}

.balance-info {
    color: white;
}

.balance-label {
    font-size: 14px;
    color: rgba(255, 255, 255, 0.8);
    margin-bottom: 8px;
    align-items: center;
}

.balance-amount {
    font-size: 28px;
    font-weight: 700;
    color: #61fb24;
}

.arrow-btn {
```

```
    background: transparent;
    border: none;
    color: white;
    cursor: pointer;
    padding: 8px;
    display: flex;
    align-items: center;
    justify-content: center;
}

.arrow-btn svg {
    width: 24px;
    height: 24px;
}

.carousel-dots {
    display: flex;
    gap: 8px;
    margin-top: 8px;
}

.dot {
    width: 8px;
    height: 8px;
    border-radius: 50%;
    background: #d1d5db;
    transition: all 0.3s ease;
}

.dot.active {
    width: 24px;
    border-radius: 4px;
    background: #40b4e5;
}

.welcome-illustration {
    width: 250px;
    height: 250px;
    flex-shrink: 0;
}

.welcome-illustration img {
    width: 100%;
    height: 100%;
    object-fit: cover;
```

```
    /* Makes the image fill the circle nicely */
}

.welcome-illustration svg {
    width: 100%;
    height: 100%;
}

.saved-payees {
    background: white;
    border-radius: 16px;
    padding: 24px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}

.section-title {
    font-size: 16px;
    font-weight: 600;
    color: #1f2937;
    margin-bottom: 20px;
}

.payee-list {
    display: flex;
    flex-direction: column;
    gap: 16px;
    margin-bottom: 16px;
}

.payee-item {
    display: flex;
    align-items: center;
    gap: 12px;
}

.payee-avatar {
    width: 44px;
    height: 44px;
    border-radius: 50%;
    overflow: hidden;
    flex-shrink: 0;
}

.payee-avatar img {
    width: 100%;
```

```
    height: 100%;
    object-fit: cover;
}

.payee-info {
    flex: 1;
}

.payee-name {
    font-size: 14px;
    font-weight: 600;
    color: #1f2937;
    margin-bottom: 2px;
}

.payee-details {
    font-size: 12px;
    color: #6b7280;
}

.view-all-link {
    display: block;
    text-align: right;
    color: #1f2937;
    font-size: 13px;
    font-weight: 500;
    text-decoration: none;
    transition: color 0.3s ease;
}

.view-all-link:hover {
    color: #40b4e5;
}

.transactions-section {
    background: white;
    border-radius: 16px;
    padding: 24px 32px;
    margin: 0 32px 32px 32px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}

.transactions-list {
    display: flex;
    flex-direction: column;
```

```
    gap: 16px;
    margin-bottom: 16px;
}

.transaction-item {
    display: grid;
    grid-template-columns: 50px 140px 120px 1fr auto;
    align-items: center;
    gap: 24px;
    padding: 16px;
    border-radius: 12px;
    transition: background 0.3s ease;
}

.transaction-item:hover {
    background: #f9fafb;
}

.transaction-avatar {
    width: 44px;
    height: 44px;
    border-radius: 50%;
    overflow: hidden;
}

.transaction-avatar img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.transaction-date {
    font-size: 14px;
    color: #1f2937;
    font-weight: 500;
}

.transaction-id {
    font-size: 14px;
    color: #6b7280;
}

.transaction-amount {
    font-size: 14px;
    font-weight: 600;
```

```
    color: #1f2937;
    text-align: right;
}

.transaction-status {
    padding: 6px 16px;
    border-radius: 20px;
    font-size: 13px;
    font-weight: 500;
    text-align: center;
    min-width: 80px;
}

.transaction-status.success {
    background: #d1fae5;
    color: #065f46;
}

@media (max-width: 1200px) {
    .content-grid {
        grid-template-columns: 1fr;
    }

    .saved-payees {
        max-width: 600px;
    }
}

@media (max-width: 768px) {
    .sidebar {
        width: 80px;
        padding: 24px 12px;
    }

    .bank-name {
        display: none;
    }

    .nav-item span {
        display: none;
    }

    .nav-item {
        padding: 12px;
        justify-content: center;
```

```
    }

    .welcome-card {
        flex-direction: column;
        gap: 24px;
    }

    .welcome-illustration {
        width: 140px;
        height: 140px;
    }

    .transaction-item {
        grid-template-columns: 40px 1fr auto;
        gap: 12px;
    }

    .transaction-date,
    .transaction-id {
        display: none;
    }

    .content-grid,
    .transactions-section {
        padding: 16px;
        margin: 16px;
    }

    .top-bar {
        padding: 16px;
    }
}

@media (max-width: 600px) {
    .sidebar {
        position: fixed;
        left: 0;
        top: 0;
        height: 100vh;
        z-index: 10;
        transform: translateX(-100%);
        transition: 0.3s ease;
    }

    .sidebar.open {
```

```
        transform: translateX(0);
    }

    .main-content {
        margin-left: 0;
    }
}









/* Pay Bills Styles*/


.pb-Container {
    display: flex;
    gap: 15px;
    padding: 24px 32px;
    flex-wrap: wrap;
}


@media (max-width: 768px) {
    .buttoncontainer {
        flex-direction: column;

    }
}

.utilitybtn-section {
    background: #ffffff;
    height: 500px;
    border-radius: 16px;
    padding: 24px 32px;
    margin: 0 32px 32px 32px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);


}
```

```css
.pbhead-text {
    font-size: 17px;
    color: #333;
    margin-bottom: 20px;
    font-weight: 1000;

}

.utility-row {
    display: flex;
    flex-wrap: wrap;     /* allows wrapping on smaller screens */
    gap: 15px;
    flex: 1;
    justify-content: space-between;
    grid-template-columns: repeat(4, 1fr);
    max-width: 450px; /* or 600px depending on button size */
    margin: auto;
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}


.utility-btn {
    width: min(75px, 25vw);
    height: min(75px, 25vw);
    border-radius: 15px;
    display: flex;
    justify-content: center;
    align-items: center;
    border: 1px solid rgba(255, 255, 255, 0.5);
    border-right: 1px solid rgba(255, 255, 255, 0.2);
    border-bottom: 1px solid rgba(255, 255, 255, 0.2);
    box-shadow: 0 5px 45px rgba(0, 0, 0, 0.3);
    backdrop-filter: blur(2px);
    transition: 0.5s;
    overflow: hidden;
    background-color: rgba(255, 255, 255, 0.1);
}

.utility-btn:hover {
    transform: translateY(-10px);
}
```

```css
.utility-btn::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 50px;
    height: 100%;
    background-color: rgba(255, 255, 255, 0.5);
    transform: skewX(45deg) translateX(150px);
    transition: 0.5s ease;
}

.utility-btn :hover::before {
    transform: skewX(45deg) translateX(-150px);
}


@media (max-width: 600px) {
    .utility-btn {
        width: 80px;
        height: 80px;
    }

    .pbbtn-label {
        font-size: 14px;
    }
}

.btn-wrapper {
    display: flex;
    flex-direction: column;
    align-items: center;

}

.pbbtn-label {
    margin-top: 8px;
    font-size: 14px;
    font-weight: 500;
}


/*Button Imgs*/

.cebimg {
```

```
    width: 70px;
    height: 70px;
    float: center;
}

.nwsdbimg {
    width: 65px;
    height: 65  px;
    float: center;
}

.sltimg {
    width: 75px;
    height: 75px;
    float: center;
}

.dialogimg {
    width: 75px;
    height: 75px;
    float: center;
}

.bill-form {
    display: none;
    border: 1px solid #ccc;
    width: 100%;
    max-width: 1000px;
    min-width: 280px;
    height: auto;
    border-radius: 16px;
    background: #ffffff;
    padding: 24px 32px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}


.payinfo {
    display: flex;
    gap: 15px;
    padding: 24px 32px;
    grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
    flex: 2;

}
```

```css
/*Payment information form*/

.pbform-title {
    margin-top: 20px;
    margin-bottom: 30px;
    font-size: 22px;
    font-weight: 600;
    color: #333;
}

/* form structure */

.pbform-group {
    margin-bottom: 15px;
}

.pbform-group label {
    font-size: 14px;
    font-weight: 600;
    display: block;
    margin-bottom: 5px;
    color: #444;
}

.pbform-group input {
    padding: 14px 16px;
    border: none;
    border-radius: 15px;
    background: #F4F4F4;
    font-size: 15px;
    outline: none;
    transition: all 0.3s ease;
}

/* submit button */

.pbnext-btn {
    width: 60%;
    background: #274582;
    color: white;
    padding: 12px;
    font-size: 16px;
    font-weight: 600;
    border-radius: 8px;
```

```
    border: none;
    cursor: pointer;
    transition: 0.2s;
}

.pbnext-btn:hover {
    background: #005fcc;
}

@media (max-width: 700px) {
    .payinfo {
        width: 100%;
        margin-top: 20px;
    }
}

.pbform-group select {
    width: 100%;
    height: 3.5em;
    padding: 10px 12px;
    border: 1px solid #ccc;
    border-radius: 8px;
    font-size: 15px;
    background: #fff;
    cursor: pointer;
    outline: none;
    transition: 0.2s;
}

.pbform-group input{
  border: 2px solid transparent;
  width: 100%;
  height: 3.5em;
  padding-left: 0.8em;
  outline: none;
  overflow: hidden;
  background-color: #F3F3F3;
  border-radius: 10px;
  transition: all 0.5s;
}


.pbform-group input:hover,
.pbform-group input:focus {
  border: 2px solid #4A9DEC;
```

```
  box-shadow: 0px 0px 0px 7px rgb(74, 157, 236, 20%);
  background-color: white;
}


.pbform-group select:focus {
    border-color: #274582;
    box-shadow: 0 0 5px rgba(39, 69, 130, 0.3);
}

.confirmbill{
   display: none;
    border: 1px solid #ccc;
    width: 100%;
    max-width: 1000px;
    min-width: 280px;
    height: auto;
    border-radius: 16px;
    background: #ffffff;
    padding: 24px 32px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}

.paydetails{
    display: none;
    border: 1px solid #ccc;
    width: 100%;
    max-width: 1000px;
    min-width: 280px;
    height: auto;
    border-radius: 16px;
    background: #ffffff;
    padding: 24px 32px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
}

.pdtext{
    text-align: center;
    font-size: 20px;
    color: #333;
    margin-bottom: 20px;
    font-weight: 1000;
}

.pbconfirmbtn{
```

```
    width: 60%;
    background: #274582;
    color: white;
    padding: 12px;
    font-size: 16px;
    font-weight: 600;
    border-radius: 8px;
    border: none;
    cursor: pointer;
    transition: 0.2s;
}
.pbconfirmbtn:hover {
    background: #005fcc;
}

.pbbackbtn{
    width: 60%;
    background: #274582;
    color: white;
    padding: 12px;
    font-size: 16px;
    font-weight: 600;
    border-radius: 8px;
    border: none;
    cursor: pointer;
    transition: 0.2s;

}

.pbbackbtn:hover{
    background: #005fcc;
}
.paydetailscard{
  padding: 20px;
  width: 80%;
  height: 254px;
  background: rgb(255, 255, 255);
  border-radius: 1rem;
  box-shadow: rgba(50, 50, 93, 0.25) 0px 6px 12px -2px,
    rgba(0, 0, 0, 0.3) 0px 3px 7px -3px;
  transition: all ease-in-out 0.3s;
  margin: 15px 0;
  margin-left: auto;
  margin-right: auto;
  display: block;
```

```css
}

.paydetailscard:hover {
  background-color: #e4e4e4;
  box-shadow: rgba(0, 0, 0, 0.09) 0px 2px 1px, rgba(0, 0, 0, 0.09) 0px 4px 2px,
    rgba(0, 0, 0, 0.09) 0px 8px 4px, rgba(0, 0, 0, 0.09) 0px 16px 8px,
    rgba(0, 0, 0, 0.09) 0px 32px 16px;
}

.paydetailsbtns{
    display: flex;
    text-align: center;
    gap: 15px;
    padding-top: 10%;
}

.pbcenter{
    text-align: center;
}

.pbdetailsrow{
    display: flex;
    justify-content: space-between;
    padding: 8px 0;

}
.pbdetailsrow strong{
    align-self: flex-start;
    font-weight: 500;
}

#confirmAmount::before {
    content: 'Rs.'; /* Inserts 'Rs.' before the span content */
    padding-right: 3px;
}

.pbpaymentsuccess{
    display: none;
    border: 1px solid #ccc;
    width: 40%;
    max-width: 1000px;
    min-width: 280px;
    height: auto;
    border-radius: 16px;
    background: #ffffff;
```

```
    padding-top: 100px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
    position: absolute;
    margin-top: 50px;
    padding-bottom: 100px;
}
.pbpaymentsuccess.show {
    display: grid;
}

.paymentsuccessicon{
    position: absolute;
    left: 50%;
    transform: translateX(-50%);
    top: -25%;
    width: 200px;
    height: 200px;
    z-index: 10;
    border-radius:50%;
    padding: 5px;
    background-color: #32fa85 ;
    box-shadow:
    0 0 0 20px #32FA8654;

}

#pbpaymentsuccess{
position: relative;
margin-left: auto;
margin-right: auto;
width:100%;
max-width: 1000px;
}

.pbh2{
    font-size: 32px;
    font-weight: 800;
    color: #1f2937;
    display: grid;
    text-align: center;
    padding-top: 10%;
}

.pbh3{
    font-size: 14px;
```

```
    font-weight: 400;
    color: #1f2937;
    display: grid;
    text-align: center;
    padding-top: 10%;
}

.pbpaymentfailed{
    display: none;
    border: 1px solid #ccc;
    width: 40%;
    max-width: 1000px;
    min-width: 280px;
    height: auto;
    border-radius: 16px;
    background: #ffffff;
    padding-top: 100px;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.22);
    position: absolute;
    margin-top: 50px;
    padding-bottom: 100px;
}

.pbpaymentfailed.show{
  display: grid;
}
#pbpaymentfailed{
position: relative;
margin-left: auto;
margin-right: auto;
width:100%;
max-width: 1000px;
}


.paymentfailedimg{
    position: absolute;
    left: 50%;
    transform: translateX(-50%);
    top: -25%;
    width: 200px;
    height: 200px;
    z-index: 10;
    border-radius:50%;
    padding: 5px;
```

```
    background-color: #ff6666 ;
    box-shadow:
    0 0 0 20px #FFD0D0;
}


  /* 1. Card Container Styling */
  .balance-card {
    /* Dimensions and spacing */
    width: 100%;
    padding: 20px;
    margin: auto;


    /* Appearance */
    background-color: #ffffff; /* White background */
    border-radius: 12px; /* Rounded corners */
    box-shadow: 0 4px 12px rgba(92, 70, 70, 0.1); /* Subtle shadow */
    font-family: Arial, sans-serif; /* Readable font */
  }

  /* 2. Header and Title Styling */
  .card-header {
    border-bottom: 1px solid #eee; /* Separator line */
    padding-bottom: 10px;
    margin-bottom: 15px;
  }

  .title {
    margin: 0;
    color: #666; /* Gray text for the title */
    font-size: 14px;
    font-weight: 500;
    text-transform: uppercase; /* All caps for clarity */
  }

  /* 3. Body and Amount Styling */
  .amount {
    margin: 0;
    font-size: 32px; /* Large, emphasized amount */
    font-weight: bold;
    color: #007bff; /* Primary color for the balance (e.g., blue) */
  }

  .account-name {
```

```
    margin-top: 5px;
    color: #999; /* Lighter gray for account name */
    font-size: 13px;
  }
```

And this is Js file

```
const buttons = document.querySelectorAll(".utility-btn");
        const forms = document.querySelectorAll(".bill-form");

        buttons.forEach(button => {
            button.addEventListener("click", () => {
                // hide all
                forms.forEach(form => form.style.display = "none");

                // show correct form
                const id = button.getAttribute("data-form");
                document.getElementById(id).style.display = "block";
            });
        });

        // NEXT BUTTON LOGIC
const nextButtons = document.querySelectorAll(".pbnext-btn");

nextButtons.forEach(btn => {
    btn.addEventListener("click", () => {
        // find parent form
        const currentForm = btn.closest(".bill-form");
        // hide current form
        currentForm.style.display = "none";

        // get next form ID
        const nextID = btn.getAttribute("data-next");
        // show next form
        document.getElementById(nextID).style.display = "block";
    });
});

const cebnextBtn = document.getElementById('cebnextBtn');
const backBtn = document.getElementById('backBtn');
const confirmBtn = document.getElementById('confirmBtn');
const cebForm = document.getElementById('cebForm');
const paydetails = document.getElementById('paydetails');
```

```
cebnextBtn.addEventListener('click', () => {
    const nameValue = document.getElementById('cebbillno').value;
    const amountValue = document.getElementById('cebAmount').value;

    // Hide first form, show confirmation
    cebForm.style.display = 'none';
    paydetails.style.display = 'block';

    // Fill confirmation details
    document.getElementById('confirmName').textContent = nameValue;
    document.getElementById('confirmAmount').textContent = amountValue;
});

// Back button: go back to edit
backBtn.addEventListener('click', () => {
    paydetails.style.display = 'none';
    cebForm.style.display = 'block';
});

// Confirm button: handle confirmation
pbconfirmBtn.addEventListener('click', () => {
 alert("Payment Confirmed!\nName: " +
document.getElementById('confirmName').textContent +
          "\nAmount: $" + document.getElementById('confirmAmount').textContent);

    // Optionally, reset forms
    cebForm.reset();
    paydetails.style.display = 'none';
    cebForm.style.display = 'block';
});

document.addEventListener('DOMContentLoaded', (event) => {
    const now = new Date();
    const options = {
        year: 'numeric',
        month: 'short',
        day: '2-digit',
        hour: '2-digit',
        minute: '2-digit',
        hour12: true
    };
    const formattedDateTime = now.toLocaleString('en-US', options);

    const dateSpan = document.getElementById('bpdate');
```

```
    if (dateSpan) {
        dateSpan.textContent = formattedDateTime;
    }
});

//fetch account primary data

function loadDashboard() {
    fetch("/api/dashboard/me")
        .then(res => res.json())
        .then(data => {
            if (!data.ok) {
                document.getElementById("balance").innerText = "Not logged in";
                return;
            }

            // Find primary account
            const primary = data.accounts.find(acc => acc.isPrimary === true);

            if (primary) {
                // Fill account balance
                document.getElementById("balance").innerText =
                    "Rs. " + primary.balance;

                // Fill account number
                document.getElementById("accountNo").innerText =
                    primary.accountNo;

                // Fill account type
                document.getElementById("accountType").innerText =
                    primary.accountType;

                // Fill customer (account) name from dashboard response
                document.getElementById("accountName").innerText =
                    data.name;

            } else {
                document.getElementById("balance").innerText =
                    "No accounts found";
            }
        });
}

loadDashboard();
```

```
//payinfo form select

function loadAccountSelect() {
    fetch("/api/dashboard/me")
        .then(res => res.json())
        .then(data => {

            const select = document.getElementById("accountSelect");


            data.accounts.forEach(acc => {
                const opt = document.createElement("option");
                opt.value = acc.accountNo;
                opt.textContent = `${acc.accountNo} - Rs. ${acc.balance}`;
                select.appendChild(opt);
            });
        });
}

loadAccountSelect();
```

and this is the database schema

```
CREATE DATABASE BankingSystemDB;

USE BankingSystemDB;

CREATE TABLE Customer (
                        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
                        Name VARCHAR(100) NOT NULL,
                        Email VARCHAR(100) NOT NULL UNIQUE,
                        NIC VARCHAR(20) NOT NULL UNIQUE,
                        PhoneNumber VARCHAR(20),
                        Username VARCHAR(50) UNIQUE,
                        PasswordHash VARCHAR(255)
);

CREATE TABLE Branch (
                       BranchID INT AUTO_INCREMENT PRIMARY KEY,
```

```
                        BranchName VARCHAR(100) NOT NULL,
                        City VARCHAR(50) NOT NULL,
                        PostalCode VARCHAR(20)
);

CREATE TABLE Account (
                         AccountNo CHAR(6) PRIMARY KEY,
                         CustomerID INT NOT NULL,
                         AccountType ENUM('Savings', 'Current', 'Fixed Deposit')
NOT NULL,
                         BranchID INT NOT NULL,
                         AccountBalance DECIMAL(15,2) DEFAULT 0.00,
                         AccountStatus ENUM('Active', 'Inactive', 'Closed')
DEFAULT 'Active',
                         CreatedDate DATETIME DEFAULT CURRENT_TIMESTAMP,
                         LastUpdatedDate DATETIME DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
                         FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID),
                         FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);

CREATE TABLE CustomerAccount (
                                 CustomerAccountID INT AUTO_INCREMENT PRIMARY
KEY,
                                 CustomerID INT NOT NULL,
                                 AccountNo CHAR(6) NOT NULL,
                                 IsPrimary BOOLEAN DEFAULT FALSE,
                                 AddedDate DATETIME DEFAULT CURRENT_TIMESTAMP,
                                 FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID),
                                 FOREIGN KEY (AccountNo) REFERENCES
Account(AccountNo)
);
```

This is the data

```
    -- Insert Customers
INSERT INTO Customer (Name, Email, NIC, PhoneNumber, Username, PasswordHash)
VALUES
    ('Alice Johnson', 'alice.johnson@example.com', '123456789V', '0711234567',
NULL, NULL),
```

```
    ('Bob Smith', 'bob.smith@example.com', '987654321V', '0717654321', NULL,
NULL),
    ('Charlie Brown', 'charlie.brown@example.com', '456123789V', '0715555555',
NULL, NULL),
    ('Diana Prince', 'diana.prince@example.com', '789456123V', '0719999999',
NULL, NULL),
    ('Edward King', 'edward.king@example.com', '321654987V', '0718888888', NULL,
NULL);

-- Insert Branches
INSERT INTO Branch (BranchName, City, PostalCode)
VALUES
    ('Central Branch', 'Colombo', '00100'),
    ('Fort Branch', 'Colombo', '00200'),
    ('Kandy Branch', 'Kandy', '20000'),
    ('Galle Branch', 'Galle', '80000'),
    ('Jaffna Branch', 'Jaffna', '40000');

-- Insert Accounts
INSERT INTO Account (AccountNo, CustomerID, AccountType, BranchID,
AccountBalance, AccountStatus)
VALUES
    ('AC0001', 1, 'Savings', 1, 5000.00, 'Active'),
    ('AC0002', 1, 'Current', 2, 12000.50, 'Active'),
    ('AC0003', 1, 'Fixed Deposit', 3, 30000.00, 'Active');

-- Other customers
INSERT INTO Account (AccountNo, CustomerID, AccountType, BranchID,
AccountBalance, AccountStatus)
VALUES
    ('AC0004', 2, 'Savings', 1, 8000.00, 'Active'),
    ('AC0005', 3, 'Current', 4, 1500.75, 'Active'),
    ('AC0006', 4, 'Savings', 5, 2200.00, 'Active'),
    ('AC0007', 5, 'Fixed Deposit', 3, 50000.00, 'Active');
```

Now I have to implement the feature paybills

The this need to happened is if someone click the button in paybills page (like cebbtn,nwsdbbtn).the are getting the form that can select what account they are choosing to pay the bill and what is his account number and the amount of the bill (in the page form there are two input to enter bill number for the verification).when the user enter all the details and click next, user will get to conform the details in the next container and after that he can click the confirm button and if the bill no and the amount is in the user

database user need to see the paymentsucceess container and get a bill confirmation No ,if there is an error in the details he enterd or the amount he entered are not in his bank account he need to get the payment failed and get the errot message saying bill no wrong or insufficient balance after that is the payment successusfull the amount he enter need to reduced from the users account balance (if you can design a way to print the receipt and way to download the receipt)

Add the tables that need to have

Like bill table for when someone entered the bill no in the database the bill no is need exists or he will get and wrong bill on error . and one user can have multiple bills and if user click the cebbtn he can only pay his ceb bill , if he want pay the slt bill he need to click the sltbtn and pay the bill also if you can give me a way to show the bill history in the paybills page

Can you give me everything I need and also this is a group project and don't change anything that can effect others members (in the backend, we can add anything for the this feature)

When u give me codes and everything tell me where to paste the code too and explain the code

If u can before the payment details confirmation add the OTP email feature too

And im planning to add more bill buttons to this page too tell me how can do that too