**INTRODUCTION**

**INTRODUCTION TO PROJECT**

A **Mphasisbank** is a financial institution which accepts deposits, pays interest on pre-defined rates, deposite money, makes loans, and often acts as an intermediary in financial transactions. It also provides other financial services to its customers.

**PROJECT OF MODULES**

* **Home**
* **Customer Registration**
* **Employee Registration**
* **Login**
* **Service**
* **Loan**
* **Balance**
* **Feedback**

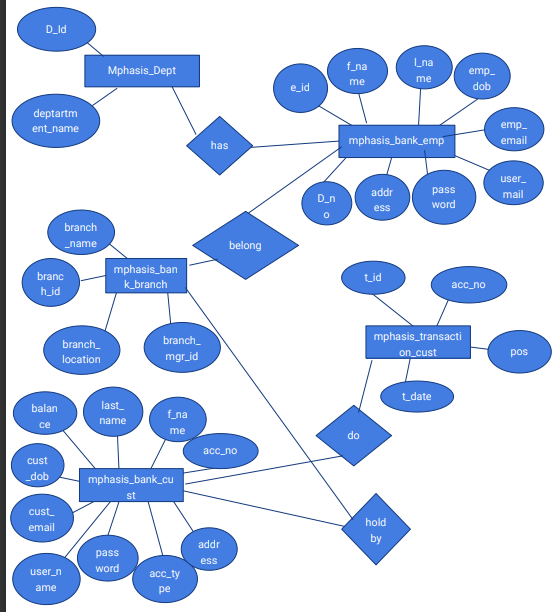
**Aims & Objectives**

The **aim** of the **banking** system is to provide security and confidence in the economy.

* Providing services.
* Currency issue.
* Creation of transaction media.
* Receiving deposit.
* Making loan.
* Ensuring safety.
* Investment.

MPHASIS BANK SCHEMA

ER DIAGRAM

****

CREATING SCHEMA:

**Step1:-Connect to data base**

Conn system/password;

**Step2:-Create new user in oracle**

Create user bank identified by mphasis123;

**step3:-Grant permission**

Grant dba to bank;

**step4:-Disconnect to system and connect to new user**

Conn bank/mphasis123;

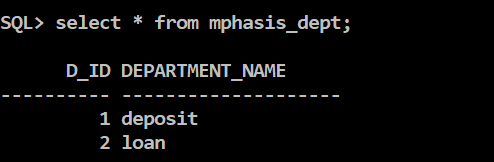
**Step5:-Create mphasis\_dept table**

CREATE TABLE MPHASIS\_DEPT (D\_ID NUMBER PRIMARY KEY,DEPARTMENT\_NAME VARCHAR2(20));

insert into MPHASIS\_DEPT values(1,'deposit');

insert into MPHASIS\_DEPT values(2,'loan');

Select \* from mphasis\_dept;



**Step6:-Create mphasis\_bank\_emp table and sequence**

CREATE TABLE MPHASIS\_BANK\_EMP (

E\_ID NUMBER PRIMARY KEY,

F\_NAME VARCHAR2 (20),

L\_NAME VARCHAR2 (20),

EMP\_DOB VARCHAR2 (20),

EMP\_EMAIL VARCHAR2 (30),

USER\_NAME VARCHAR2 (20) UNIQUE NOT NULL,

PASSWORD CHAR (32),

ADDRESS VARCHAR2 (40),

D\_NO NUMBER,

CONSTRAINT FK\_DEPT\_ID FOREIGN KEY (D\_NO) REFERENCES MPHASIS\_DEPT (D\_ID));

CREATE SEQUENCE EMPLOYEE\_ID\_SEQ

START WITH 1

INCREMENT BY 1;

INSERT INTO MPHASIS\_BANK\_EMP VALUES (EMPLOYEE\_ID\_SEQ.NEXTVAL,'HARIKA','Y',TO\_DATE ('01/01/1990 22:30:50', 'DD/MM/YYYY HH24:MI:SS'),

' [HARIKA @ MAIL.COM](mailto:HARIKA@MAIL.COM) ','HARIKA','HARIKA123','RJY',1);

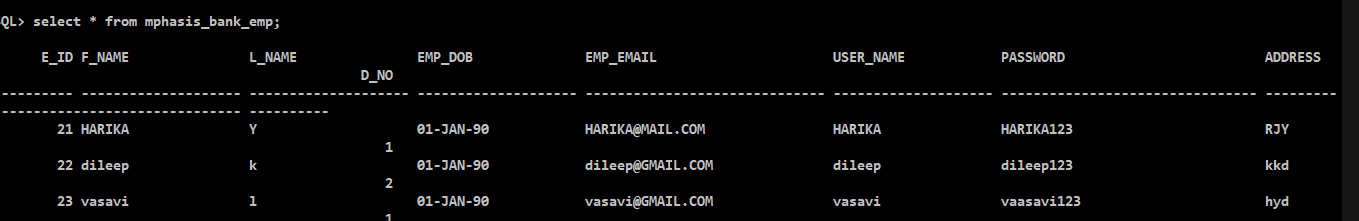
INSERT INTO MPHASIS\_BANK\_EMP VALUES (EMPLOYEE\_ID\_SEQ.NEXTVAL,'dileep','k',TO\_DATE ('01/01/1990 22:30:50', 'DD/MM/YYYY HH24:MI:SS'),'dileep@GMAIL.COM','dileep',

'dileep123','kkd',2);

INSERT INTO MPHASIS\_BANK\_EMP VALUES (EMPLOYEE\_ID\_SEQ.NEXTVAL,'vasavi','l',TO\_DATE ('01/01/1990 22:30:50', 'DD/MM/YYYY HH24:MI:SS'),'vasavi@GMAIL.COM','vasavi',

'vaasavi123','hyd',1);

Select \* from MPHASIS\_BANK\_EMP;



**Step7:-create mphasis\_bank\_branch table and sequence**

CREATE TABLE MPHASIS\_BANK\_BRANCH (

BRANCH\_ID NUMBER PRIMARY KEY,

BRANCH\_NAME VARCHAR2 (50),

BRANCH\_LOCATION VARCHAR2 (50),

BRANCH\_MGR\_ID NUMBER,

CONSTRAINT FK\_BRANCH\_MGR FOREIGN KEY (BRANCH\_MGR\_ID) REFERENCES MPHASIS\_BANK\_EMP (E\_ID));

CREATE SEQUENCE BRANCH\_ID\_SEQ

START WITH 103

INCREMENT BY 1;

INSERT INTO MPHASIS\_BANK\_BRANCH VALUES

(101,'kukatpally','hyderabad',21);

INSERT INTO MPHASIS\_BANK\_BRANCH VALUES

(102,'madhapur','hyderabad',22);

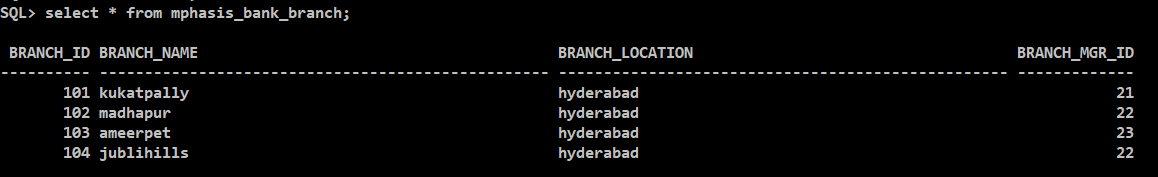
INSERT INTO MPHASIS\_BANK\_BRANCH VALUES

(BRANCH\_ID\_SEQ.NEXTVAL,'ameerpet','hyderabad',23);

INSERT INTO MPHASIS\_BANK\_BRANCH VALUES

(BRANCH\_ID\_SEQ.NEXTVAL,'jublihills','hyderabad',22);

Select \* from MPHASIS\_BANK\_BRANCH;



**Step 8:-Create mphasis\_bank\_cust table and sequence**

CREATE TABLE MPHASIS\_BANK\_CUST (

ACCOUNT\_NO NUMBER,

F\_NAME VARCHAR2 (20),

L\_NAME VARCHAR2 (20),

CUST\_DOB DATE,

CUST\_EMAIL VARCHAR2 (20),

USER\_NAME VARCHAR2 (20) UNIQUE NOT NULL,

PASSWORD CHAR (32),

ADDRESS VARCHAR2 (40),

ACCOUNT\_TYPE VARCHAR2 (20),

BALLANCE NUMBER (7,2),

DATE\_CREATED DATE DEFAULT SYSDATE,

CONSTRAINT PK\_MPHASIS\_BANK\_CUST PRIMARY KEY (ACCOUNT\_NO),

CONSTRAINT MPHASIS\_ACC\_TYPE CHECK (ACCOUNT\_TYPE IN ('SAVINGS', 'FIXED DEPOSIT', 'LOAN')));

CREATE SEQUENCE CUST\_ACCOUNT\_SEQ

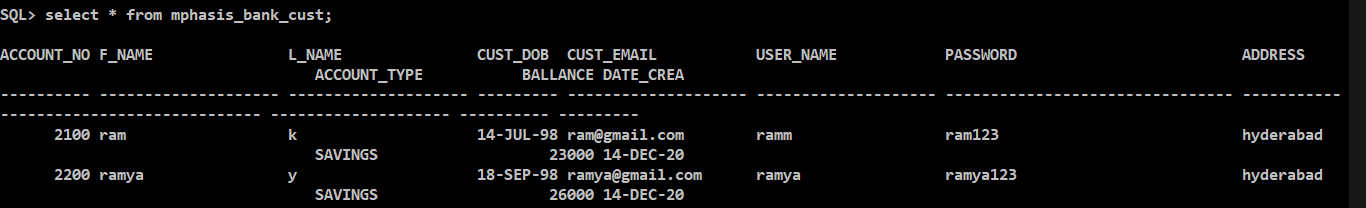
START WITH 100

INCREMENT BY 100;

INSERT INTO MPHASIS\_BANK\_CUST VALUES(CUST\_ACCOUNT\_SEQ.NEXTVAL,'ram','k','14-jul-1998','ram@gmail.com','ramm','ram123','hyderabad','SAVINGS',23000,'14-dec-2020');

INSERT INTO MPHASIS\_BANK\_CUST VALUES(CUST\_ACCOUNT\_SEQ.NEXTVAL,'ramya','y','18-sep-1998','ramya@gmail.com','ramya','ramya123','hyderabad','SAVINGS',26000,'14-dec-2020');

Select \* from MPHASIS\_BANK\_CUST;



**Step 8:- Create mphasis\_transaction\_cust table and sequences**

CREATE TABLE MPHASIS\_TRANSACTION\_CUST (

T\_ID NUMBER,

ACC\_NO NUMBER,

POS NUMBER,

T\_DATE DATE DEFAULT SYSDATE,

AMOUNT NUMBER (7,2),

CONSTRAINT PK\_MPHASIS\_TRANSACTION\_CUST PRIMARY KEY (T\_ID),

CONSTRAINT FK\_ACC\_NO FOREIGN KEY (ACC\_NO) REFERENCES MPHASIS\_BANK\_CUST (ACCOUNT\_NO));

CREATE SEQUENCE CUST\_TRANSACTION\_SEQ

START WITH 100

INCREMENT BY 1;

CREATE SEQUENCE CUST\_TRANSACTION\_POS\_SEQ

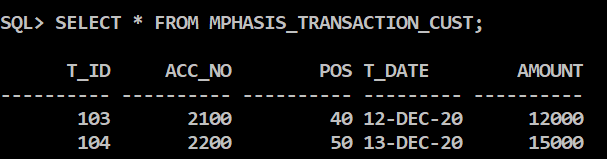
START WITH 10

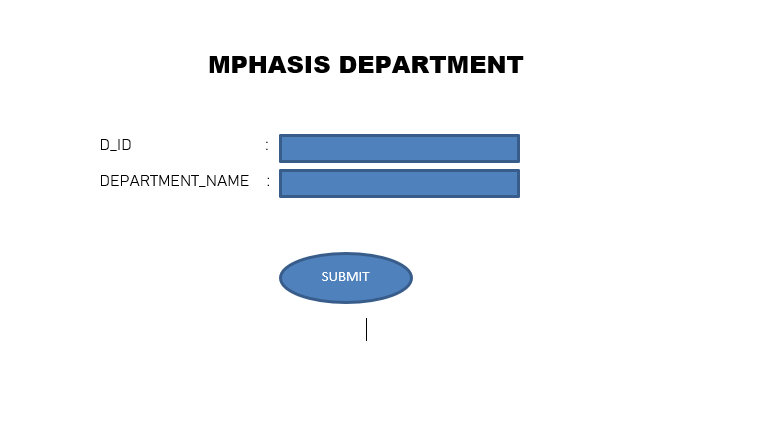
INCREMENT BY 10;

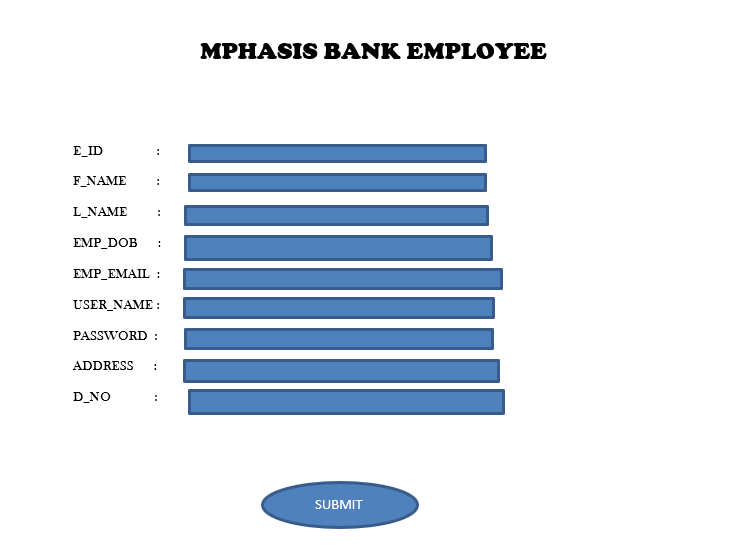
insert into MPHASIS\_TRANSACTION\_CUST values (CUST\_TRANSACTION\_SEQ.NEXTVAL,2100,CUST\_TRANSACTION\_POS\_SEQ.NEXTVAL,'12-dec-2020',12000);

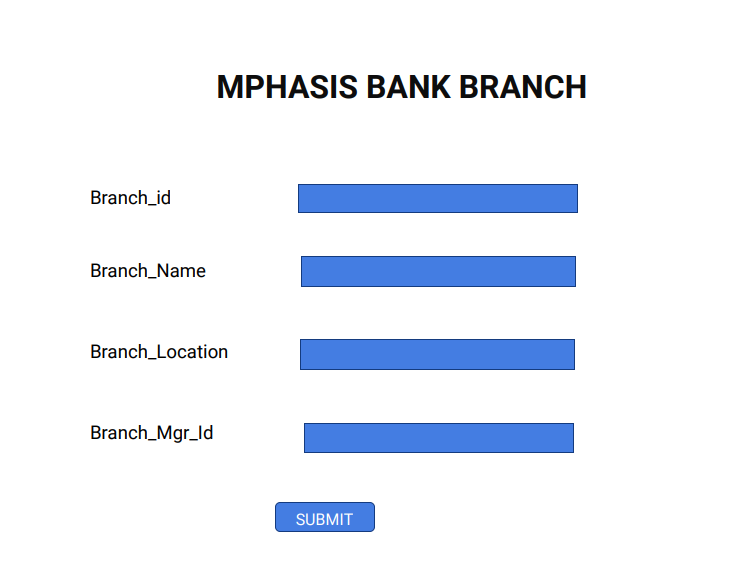
insert into MPHASIS\_TRANSACTION\_CUST values (CUST\_TRANSACTION\_SEQ.NEXTVAL,2200,CUST\_TRANSACTION\_POS\_SEQ.NEXTVAL,'13-dec-2020',15000);

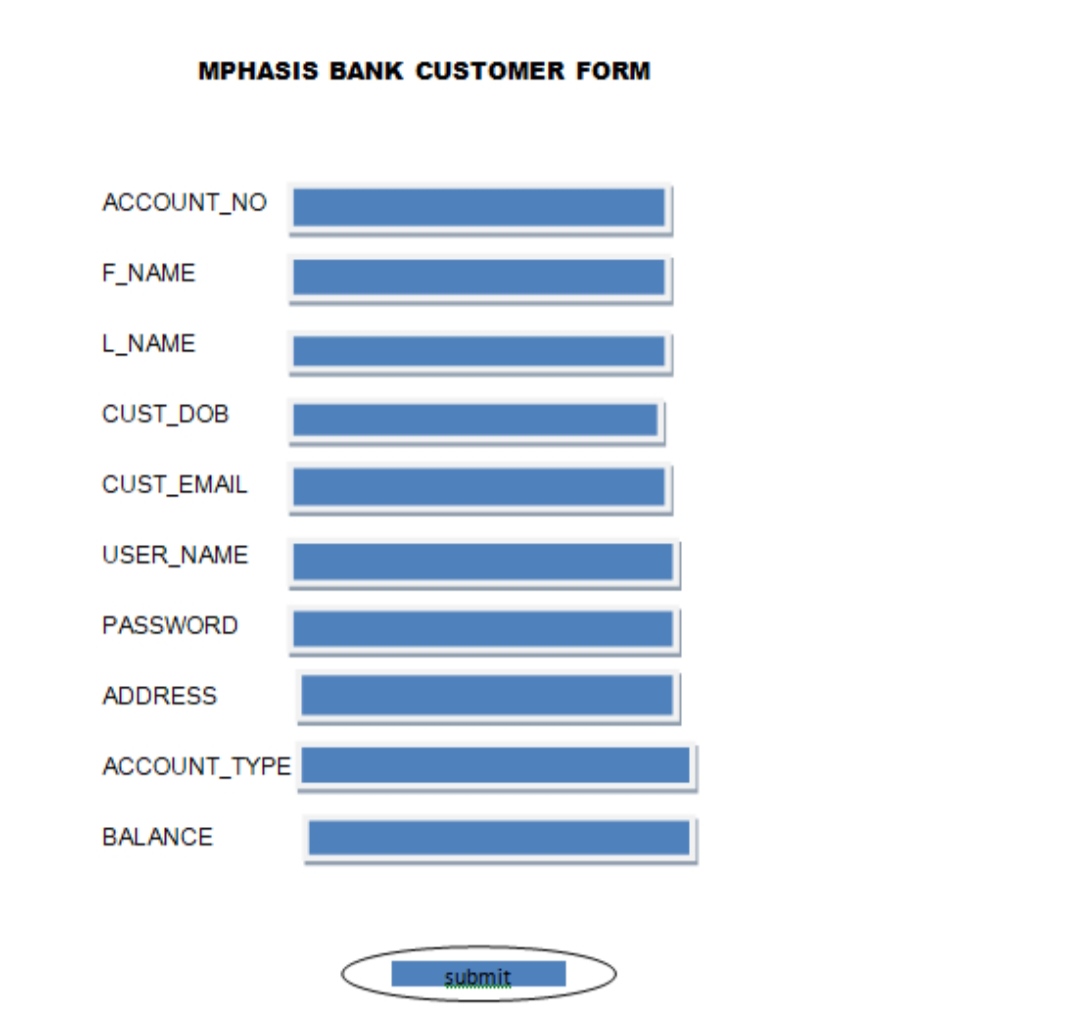
SELECT \* FROM MPHASIS\_TRANSACTION\_CUST;

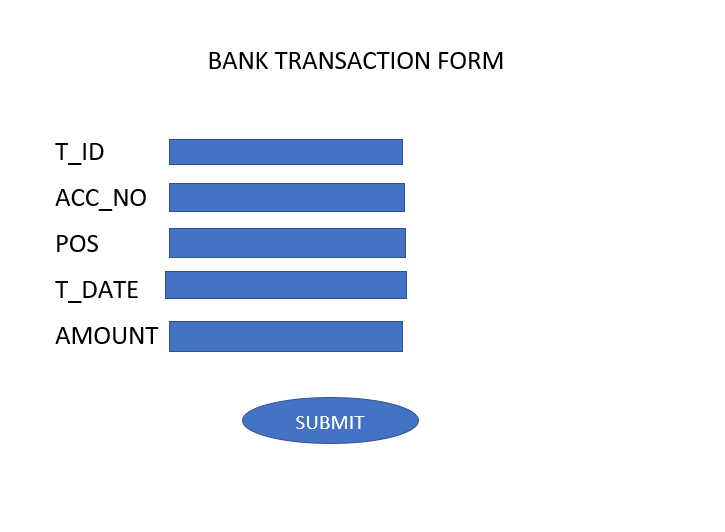


FORMS:-

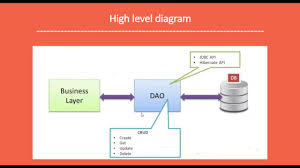








Program flow

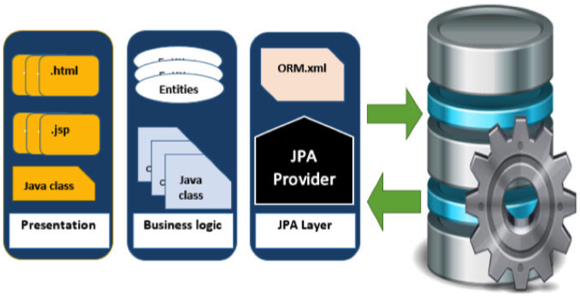


Flow diagram

JPA(java persistence api)

Java Persistence API is a collection of classes and methods to persistently store the vast amounts of data into a database which is provided by the Oracle Corporation.

To reduce the burden of writing codes for relational object management, a programmer follows the ‘JPA Provider’ framework, which allows easy interaction with database instance. Here the required framework is taken over by JPA.



Source code

Coding part(java):

Employee.java

package dileep.Model;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.Data;

@Data

@Entity

public class Employee {

@Id

public String employeeid;

public String name;

public String phone;

public String email;

public String password;

public String department;

public String employeetype;

public String getEmployeeid() {

return employeeid;

}

public void setEmployeeid(String employeeid) {

this.employeeid = employeeid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public String getEmployeetype() {

return employeetype;

}

public void setEmployeetype(String employeetype) {

this.employeetype = employeetype;

}

public Employee(String employeeid, String name, String phone, String email, String password, String department,

String employeetype) {

super();

this.employeeid = employeeid;

this.name = name;

this.phone = phone;

this.email = email;

this.password = password;

this.department = department;

this.employeetype = employeetype;

}

public Employee() {

super();

}

}

Controllercode:

package dileep.Controller;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

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.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import dileep.Model.Balance;

import dileep.Model.Customer;

import dileep.Model.Employee;

import dileep.Model.Feedback;

import dileep.Model.Loan;

import dileep.Model.Loanstatus;

import dileep.Service\_Methods.CustomerService;

import dileep.Service\_Methods.EmployeeService;

@RestController

@RequestMapping("/Bank")

@CrossOrigin(origins = "http://localhost:4200")

public class MBMLAController {

@Autowired

private EmployeeService MService;

@Autowired

private CustomerService CService;

List<Customer> listCust;

List<Employee> listEmp ;

List<Balance> listBal;

List<Loan> listLoan;

List<Loanstatus> liststatus;

@GetMapping("/all/employees")

public ResponseEntity<?> getAllEmployees() {

ResponseEntity<?> response = null ;

try {

List<Employee> list = MService.getAllEmployees();

if(list!=null && !list.isEmpty()) {

list.sort((s1,s2)->s1.getName().compareTo(s2.getName()));

response = new ResponseEntity<List<Employee>>(list, HttpStatus.OK);

} else {

response = new ResponseEntity<String>(

"No Employee Found",

HttpStatus.OK);}

} catch (Exception e) {

response = new ResponseEntity<String>(

"Unable to Fetch Students",

HttpStatus.INTERNAL\_SERVER\_ERROR); //500

e.printStackTrace();

}

return response;

}

@GetMapping("/all/loans")

public ResponseEntity<?> getAllLoans() {

ResponseEntity<?> response = null ;

try {

listLoan = CService.getAllloans();

if(listLoan!=null && !listLoan.isEmpty()) {

response = new ResponseEntity<List<Loan>>(listLoan, HttpStatus.OK);

} else {

response = new ResponseEntity<String>(

"No Employee Found",

HttpStatus.OK);

}

} catch (Exception e) {

response = new ResponseEntity<String>(

"Unable to Fetch Loans",

HttpStatus.INTERNAL\_SERVER\_ERROR); //500

e.printStackTrace();

}

return response;

}@PostMapping("/addemployee")

public Employee saveEmployee(

@RequestBody Employee employee)

{ Employee emp;

try {

emp = MService.saveEmployee(employee);

} catch (Exception e) {

emp =null;

e.printStackTrace();

}

return emp;

}

@PostMapping("/addcustomer")

public Customer saveCustomer(

@RequestBody Customer customer)

{ Customer cust;

try {

cust = CService.saveCustomer(customer);

Balance bal=new Balance(customer.customerid,500);

bal=CService.saveBalance(bal);

} catch (Exception e) {

cust =null;

e.printStackTrace();

}

return cust;

}

@PostMapping("/addfeedback")

public Feedback saveFeedback1(

@RequestBody Feedback feedback)

{ Feedback feed;

try {

feed = MService.saveFeedback(feedback);

} catch (Exception e) {

feed =null;

e.printStackTrace();

}

return feed;

}

@PostMapping("/addloanrequest")

public Loan saveLoanRequest(

@RequestBody Loan loan)

{ Loan ln;

try {

ln = CService.saveLoanrequest(loan);

Loanstatus lns=new Loanstatus(loan.customerid,loan.status);

CService.saveLoanStatus(lns);

} catch (Exception e) {

ln =null;

e.printStackTrace();

}

return ln;

}

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

public Customer getOneCustomer(

@PathVariable String id

)

{

Customer cust = null;

listCust =CService.getAllCustomers();

try {

for(Customer customer:listCust) {

if(customer.getCustomerid().equalsIgnoreCase(id)) {

cust=customer;

}

}} catch (Exception e) {

e.printStackTrace();

}return cust;

}

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

public Employee getOneEmployee(

@PathVariable String id

)

{

Employee emp = null;

listEmp =MService.getAllCustomers();

try {

for(Employee employee:listEmp) {

if(employee.getEmployeeid().equalsIgnoreCase(id)) {

emp=employee;

}

}

} catch (Exception e) {

e.printStackTrace();

}

return emp;

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

public Balance getOneCustomerBalance(

@PathVariable String id

)

{

Balance bal = null;

listBal =CService.getAllCustomersBalance();

try {

for(Balance balance:listBal) {

if(balance.getCustomerid().equalsIgnoreCase(id)) {

bal=balance;

System.out.println(bal.balance1);

}

}} catch (Exception e) {

e.printStackTrace();

}return bal;

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

public Loanstatus getOneLoanStatus(

@PathVariable String id

)

{

Loanstatus lnst = null;

liststatus =CService.getAllloanstatus();

try {

for(Loanstatus lnst1:liststatus) {

if(lnst1.getCustomerid().equalsIgnoreCase(id)) {

lnst=lnst1;

}

}

} catch (Exception e) {

e.printStackTrace();

}return lnst;

}

@GetMapping("/getupdatedbalance/{id}/{id1}")

public Balance getUpdatedOneCustomerBalance(

@PathVariable("id") String id,@PathVariable("id1") String id1

)

{ int id2=Integer.parseInt(id1);

Balance bal = null;

int balance1;

listBal =CService.getAllCustomersBalance();

try {

for(Balance balance:listBal) {

if(balance.getCustomerid().equalsIgnoreCase(id)) {

balance1=balance.getBalance();

balance1=(balance1+id2);

balance.setBalance(balance1);

bal=CService.saveBalance(balance);

}

}} catch (Exception e) {

e.printStackTrace();

}

return bal;

}@GetMapping("/getupdatedbalance1/{id}/{id1}")

public Balance getUpdatedOneCustomerBalance1(

@PathVariable("id") String id,@PathVariable("id1") String id1

)

{ int id2=Integer.parseInt(id1);

Balance bal = null;

int balance1;

listBal =CService.getAllCustomersBalance();

try {

for(Balance balance:listBal) {

if(balance.getCustomerid().equalsIgnoreCase(id)) {

balance1=balance.getBalance();

balance1=(balance1-id2);

balance.setBalance(balance1);

bal=CService.saveBalance(balance);

}

}

} catch (Exception e) {

e.printStackTrace();

}

return bal;

}

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

public ResponseEntity<?> removeLoanRequest(

@PathVariable String id

)

{ResponseEntity<?> resp = null;

listLoan =CService.getAllloans();try {

for(Loan loan:listLoan) {

if(loan.getCustomerid().equalsIgnoreCase(id)) {

CService.deleteLoanrequest(loan);

resp= new ResponseEntity<List<Loan>>(listLoan, HttpStatus.OK);

}

}

} catch (Exception e) {

resp=null;

e.printStackTrace();

}return resp;

}@PutMapping("/updatestatus/{id}")

public ResponseEntity<?> updateStudent(

@PathVariable String id,

@RequestBody Loan loan

){ResponseEntity<?> resp =null;

liststatus =CService.getAllloanstatus();

String Status="Loan approved";

listLoan =CService.getAllloans();

try {

for(Loanstatus lnst:liststatus) {

if(lnst.getCustomerid().equalsIgnoreCase(id)) {

lnst.setStatus(Status);

CService.saveLoanStatus(lnst);

System.out.println(lnst.getStatus());

}}

} catch (Exception e) {

e.printStackTrace();

}try {

for(Loan loan1:listLoan) {

if(loan1.getCustomerid().equalsIgnoreCase(id)) {

CService.deleteLoanrequest(loan1);

resp= new ResponseEntity<List<Loan>>(listLoan, HttpStatus.OK);

}

}

} catch (Exception e) {

resp=null;

e.printStackTrace();

}return resp;

}//end

Employee Service.java

package dileep.Service\_Methods;

import java.util.List;

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

import org.springframework.stereotype.Service;

import dileep.Model.Customer;

import dileep.Model.Employee;

import dileep.Model.Feedback;

import dileep.Repository.BankRepository;

import dileep.Repository.BankRepositoryFeedback;

@Service

public class EmployeeService {

@Autowired

public BankRepository EmpRepo;

@Autowired

public BankRepositoryFeedback FeedRepo;

public List<Employee> getAllEmployees() {

return EmpRepo.findAll();

}

public Employee saveEmployee(Employee employee) {

employee=EmpRepo.save(employee);

return employee;

}

public List<Employee> getAllCustomers() {

return EmpRepo.findAll();

}

public Feedback saveFeedback(Feedback feedback) {

feedback=FeedRepo.save(feedback);

return feedback;

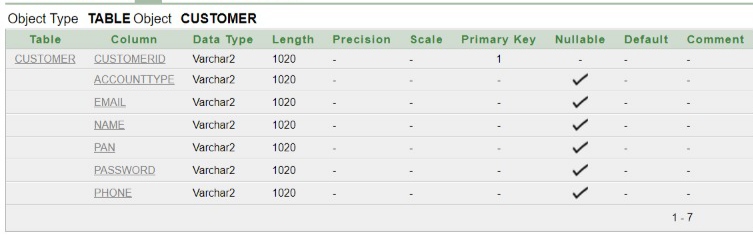
}

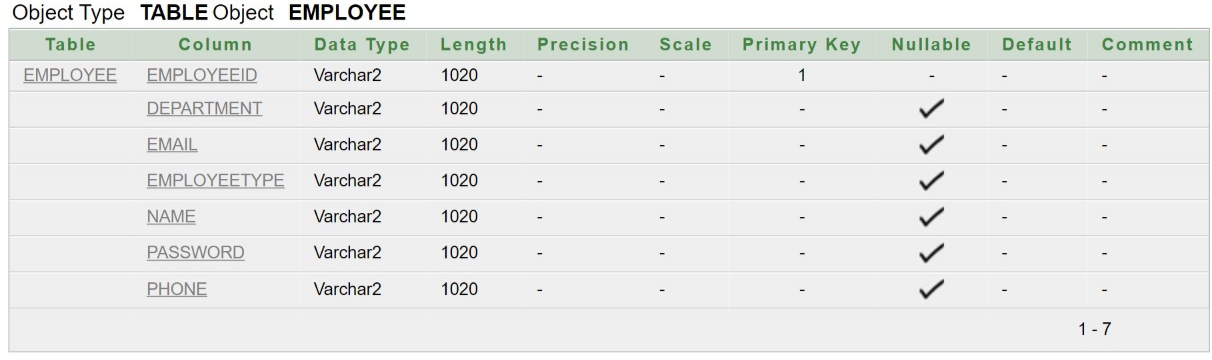
}

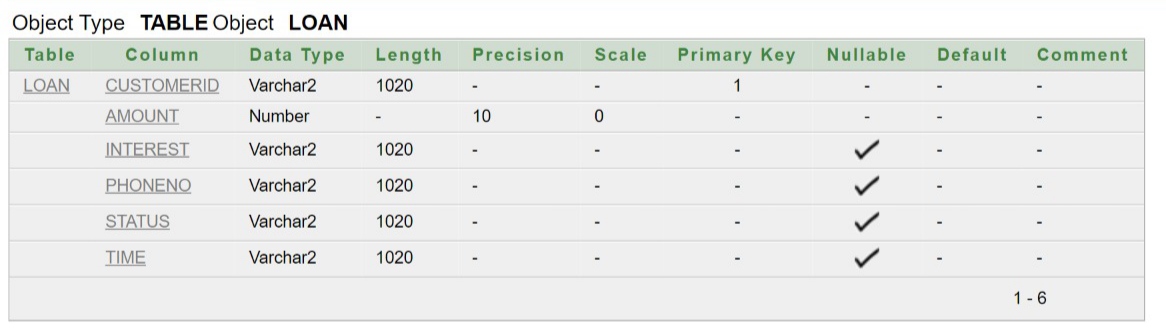
Outputs:

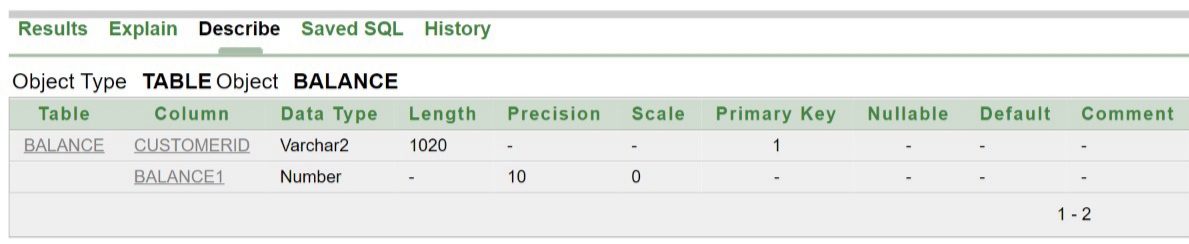
**Tables for project:**

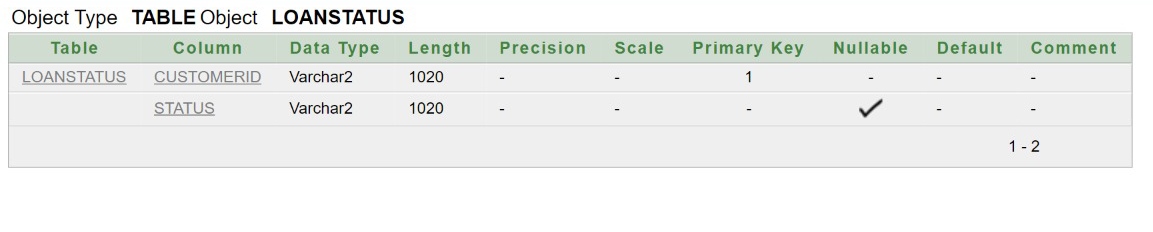
* **Go to windows->Oracle Database 10g Express…->Go to Database Home page->Database Login->SQL->SQL Commands**
* **Run.**











**Angular code:**

**Customer registration.html**

<div class="col-md-6">

<h2 class="text-center">Customer Registration</h2>

<form #form="ngForm">

<div class="form-group">

<label for="name">Enter Name:</label>

<input [(ngModel)]="customer.name" placeholder="Name" name="name" class="form-control" id="name">

</div>

<div class="form-group">

<label for="phone">Enter Phone Number </label>

<input [(ngModel)]="customer.phone" placeholder="Phone Number" name="phone" class="form-control" id="phone">

</div>

<div class="form-group">

<label for="Email">Enter Email</label>

<input [(ngModel)]="customer.email" placeholder="email address" name="email" class="form-control" id="email" required #idfield="ngModel">

</div>

<div class="form-group">

<label for="pan">Enter PAN Number</label>

<input [(ngModel)]="customer.pan" placeholder="PAN Number" name="pan" class="form-control" id="pan">

</div>

<div class="form-group">

<label for="account">Enter Type Of Account you wish to Open : </label>

<div><select [(ngModel)]="customer.accounttype" name="accounttype" class="form-control">

<option value="savings">Savings</option>

<option value="loan">Loan</option>

<option value="FD">Fixed Deposit</option>

</select>

</div>

</div>

<div class="form-group">

<label for="password">Enter Your Password</label>

<input [(ngModel)]="customer.password" placeholder="password" name="password" class="form-control" id="password">

</div>

<div>

<button class="btn btn-success" (click)="createCustomer()">Create</button>

</div>

</form>

</div>

Customer registration.ts

import { Component, OnInit } from '@angular/core';

import { HttpclientService,Customer } from '../httpclient.service';

@Component({

selector: 'app-customer-register',

templateUrl: './customer-register.component.html',

styleUrls: ['./customer-register.component.css']

})

export class CustomerRegisterComponent implements OnInit {

customer:Customer = new Customer("","","","","","","");

constructor( private httpClientService: HttpclientService

) { }

ngOnInit(): void { }

createCustomer(): void {

this.customer.customerid="MBMLA"+this.customer.phone;

alert(this.customer.customerid);

this.httpClientService.createCustomer(this.customer)

.subscribe( data => {

alert(data.name+" Customer account created successfully.");

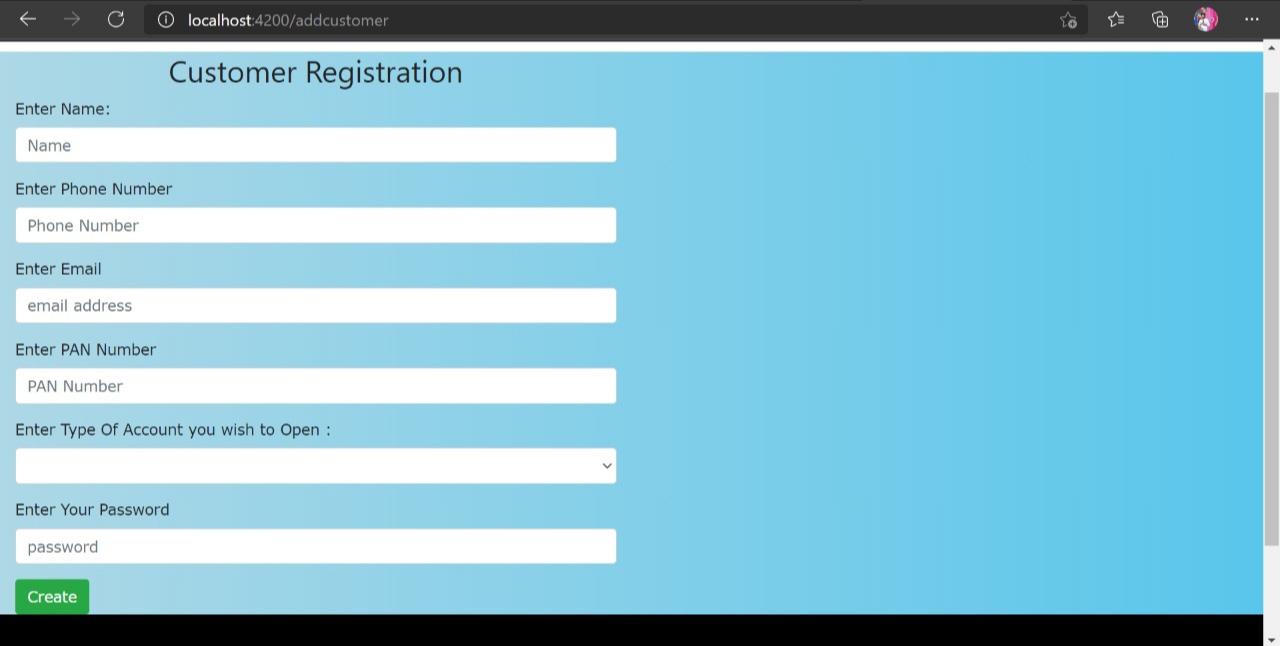
console.log("employee"+data);

});

};

}

Customer Registration output:



**Employee registration.html**

<div class="col-md-6">

<h2 class="text-center">Employee Registration</h2>

<form #form="ngForm">

<div class="form-group">

<label for="name">Enter Name:</label>

<input [(ngModel)]="employee.name" placeholder="Name" name="name" class="form-control" id="name">

</div>

<div class="form-group">

<label for="phone">Enter Phone Number </label>

<input [(ngModel)]="employee.phone" placeholder="Phone Number" name="phone" class="form-control" id="phone">

</div>

<div class="form-group">

<label for="Email">Enter Email</label>

<input [(ngModel)]="employee.email" placeholder="email address" name="email" class="form-control" id="email" required #idfield="ngModel">

</div>

<div class="form-group">

<label for="type of employee">Enter Type Of Employee : </label>

<div><select [(ngModel)]="employee.employeetype" name="employeetype" class="form-control">

<option value="Manager">Manager</option>

<option value="Employee">Employee</option>

</select>

</div>

</div>

<div class="form-group">

<label for="employee department">Enter Your Department : </label>

<div><select [(ngModel)]="employee.department" name="employeetype" class="form-control">

<option value="Deposit">Deposit</option>

<option value="Loan">Loan</option>

<option value="Manager">Manager</option>

</select>

</div>

</div>

<div class="form-group">

<label for="password">Enter Your Password</label>

<input [(ngModel)]="employee.password" placeholder="password" name="password" class="form-control" id="password">

</div>

<div>

<button class="btn btn-success" (click)="createEmployee()">Create</button>

</div>

</form>

</div>

**Employee registration.ts**

import { Component, OnInit } from '@angular/core';

import { HttpclientService,Employee } from '../httpclient.service';

@Component({

selector: 'app-employee-register',

templateUrl: './employee-register.component.html',

styleUrls: ['./employee-register.component.css']

})

export class EmployeeRegisterComponent implements OnInit {

employee:Employee = new Employee("","","","","","","");

constructor( private httpClientService: HttpclientService) { }

ngOnInit(): void {

}

createEmployee(): void {

this.employee.employeeid="MBMLA"+this.employee.phone;

alert(this.employee.employeeid)

this.httpClientService.createEmployee(this.employee)

.subscribe( data => {

alert("employee Id ="+ data.employeeid +" Employee created successfully. remember this employee ID");

console.log("employee"+data);

});

};

}

**Output:**

****

**Login.html**

<div class="col-md-6">

<h2 class="text-center">Bank User Login</h2>

<form #form="ngForm">

<div class="form-group">

<label for="type of employee">Select Type Of User : </label>

<div><select [(ngModel)]="Usertype" name="Usertype" class="form-control">

<option value="Manager">Manager</option>

<option value="Employee">Employee</option>

<option value="Customer">Customer</option>

</select>

</div>

</div>

<div class="form-group">

<label for="name">Enter {{Usertype}} Id:</label>

<input [(ngModel)]="userID" placeholder="ID" name="name" class="form-control" id="name">

</div>

<div class="form-group">

<label for="phone">Enter Password </label>

<input [(ngModel)]="password" placeholder="password" name="phone" class="form-control" id="phone">

</div>

<div>

<button class="btn btn-success" (click)="Login()">login</button>

</div>

</form>

</div>

**Login.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { EmployeeRegisterComponent } from '../employee-register/employee-register.component';

import { Customer } from '../httpclient.service';

import { HttpclientService,Employee } from '../httpclient.service';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

public Usertype:String="";

public userID:String="MBMLA";

public password:String="";

public id!:String;

public pword!:String;

constructor(private httpClientService: HttpclientService,private router: Router) { }

ngOnInit(): void {

}

Login(){

if(this.Usertype=="Customer"){

/this.getCustomer();/

this.httpClientService.getOneCustomer(this.userID).subscribe(

data => {

this.id = data.customerid;

this.pword=data.password;

if((this.id==this.userID)&&(this.pword&&this.password)){

this.router.navigate(['welcome',this.userID]);

}else{

this.router.navigate(['addcustomer']);}

}, error => {

console.log(error);

});

}

if(this.Usertype=="Employee"){

/\*this.getEmployee(); \*/

this.httpClientService.getOneEmployee(this.userID).subscribe(

data => {

this.id = data.employeeid;

this.pword=data.password;

if((this.id==this.userID)&&(this.pword==this.password)){

this.router.navigate(['welcomeemployee',this.userID]);

}else{

this.router.navigate(['addemployee']);}

}, error => {

console.log(error);

});

}

if(this.Usertype=="Manager"){

/\* this.getEmployee(); \*/

this.httpClientService.getOneEmployee(this.userID).subscribe(

data => {

this.id = data.employeeid;

this.pword=data.password;

if((this.id==this.userID)&&(this.pword&&this.password)){

this.router.navigate(['welcomemanager',this.userID]);

}else{

this.router.navigate(['addemployee']);

}

}, error => {

console.log(error);

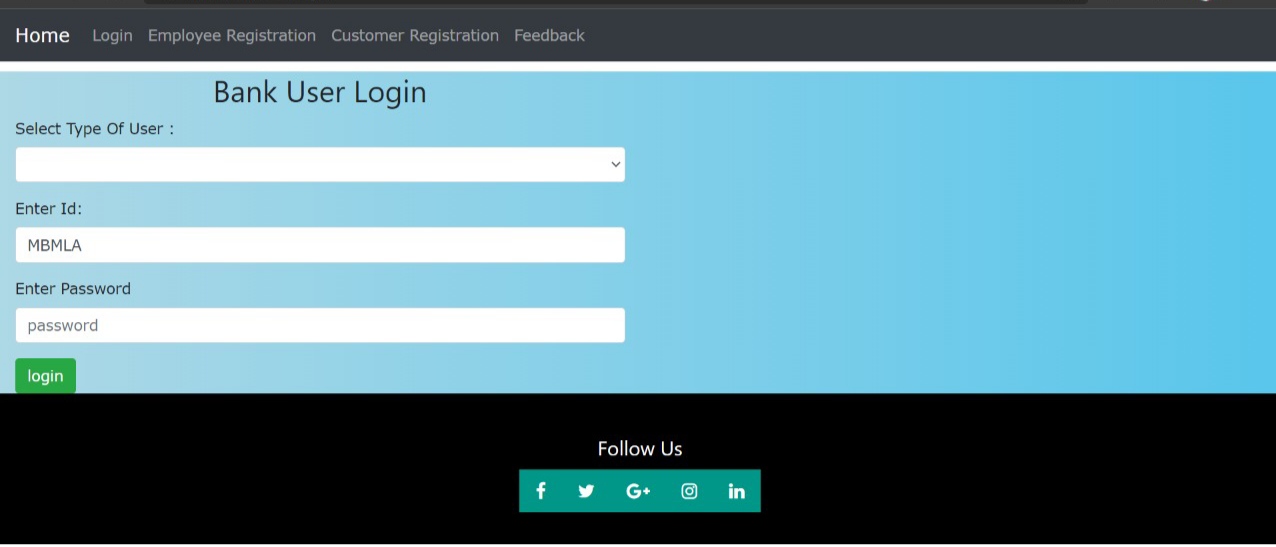
});

}

}

}

**Output:**

****

Feedback.html

<h3>Feedback Form</h3>

<form (ngSubmit)="createfeedback()">

<!-- row#1 -->

<div class="row">

<div class="col-2">

<label>Name :</label>

</div>

<div class="col-4">

<input type="text" [(ngModel)]="feedback.name" name="name" class="form-control"/>

</div>

</div>

<!-- row#2 -->

<div class="row">

<div class="col-2">

<label>Phone Number :</label>

</div>

<div class="col-4">

<input type="text" [(ngModel)]="feedback.phone" name="gender" class="form-control"/>

</div>

</div>

<!-- row#3 -->

<div class="row">

<div class="col-2">

<label>Email :</label>

</div>

<div class="col-4">

<input type="text" [(ngModel)]="feedback.email" name="email" class="form-control"/>

</div>

</div>

<div class="row">

<div class="col-2">

<label>Address :</label>

</div>

<div class="col-4">

<textarea type="textarea" [(ngModel)]="feedback.address" name="email" class="form-control"></textarea>

</div>

</div>

<div class="row">

<div class="col-2">

<label>Suggestion :</label>

</div>

<div class="col-4">

<textarea type="textarea" [(ngModel)]="feedback.suggestion" name="aadharno" class="form-control"></textarea>

</div>

</div>

<input type="submit" value="Submit feedback" class="btn btn-success"/>

</form>

**Feedback.ts**

import { Component, OnInit } from '@angular/core';

import { HttpclientService,Feedback } from '../httpclient.service';

@Component({

selector: 'app-feedback',

templateUrl: './feedback.component.html',

styleUrls: ['./feedback.component.css']

})

export class FeedbackComponent implements OnInit {

feedback:Feedback= new Feedback("","","","","");

constructor(private httpClientService: HttpclientService) { }

ngOnInit(): void {

}

createfeedback(){

this.httpClientService.createFeedback(this.feedback).subscribe(

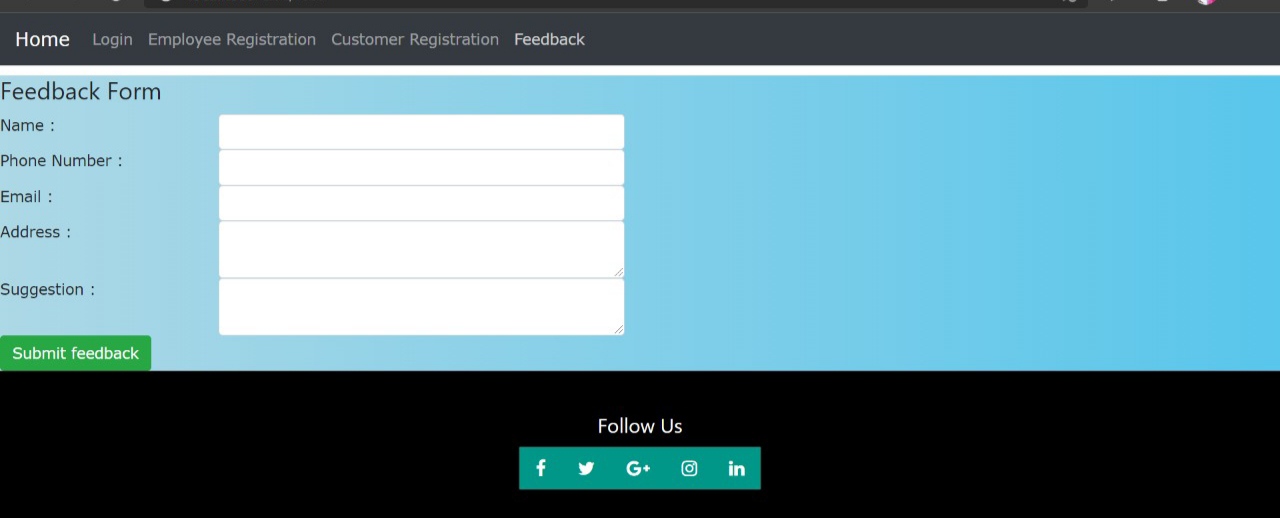
data => {

alert(data.name+" thank you for your feedback and suggestions ");

});

}}

**Output:**

****

Junit Code :

package in.bank.project;

import static org.assertj.core.api.Assertions.assertThat;

import static org.junit.jupiter.api.Assertions.assertFalse;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.junit.jupiter.api.Assertions.assertTrue;

import java.util.List;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import org.springframework.test.annotation.Rollback;

import in.bank.project.model.Customer;

import in.bank.project.repository.BankRepositoryCustomer;

@DataJpaTest

@AutoConfigureTestDatabase(replace = Replace.NONE)

public class SpringbootApplicationTests {@Autowired

private BankRepositoryCustomer CustRepo;

@Test

@Rollback(false)

public void testSaveCustomer() {

Customer customer = new Customer("MBMLA901073043","dileep","901073043","dileep@gmail.com","Dileep@212","gkfpk4425","savings");

Customer savedcustomer= CustRepo.save(customer);

assertNotNull(savedcustomer);

}

@Test

public void testGetAllCustomers() {

List<Customer> customers = (List<Customer>) CustRepo.findAll();

for ( Customer customer : customers)

{

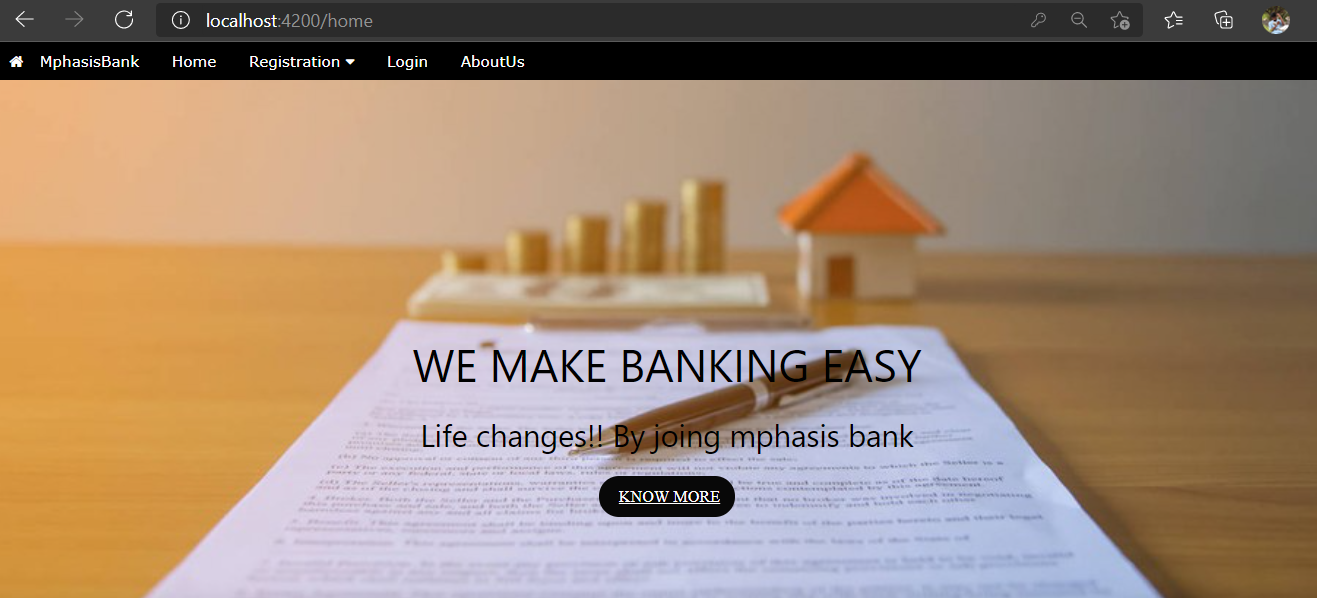
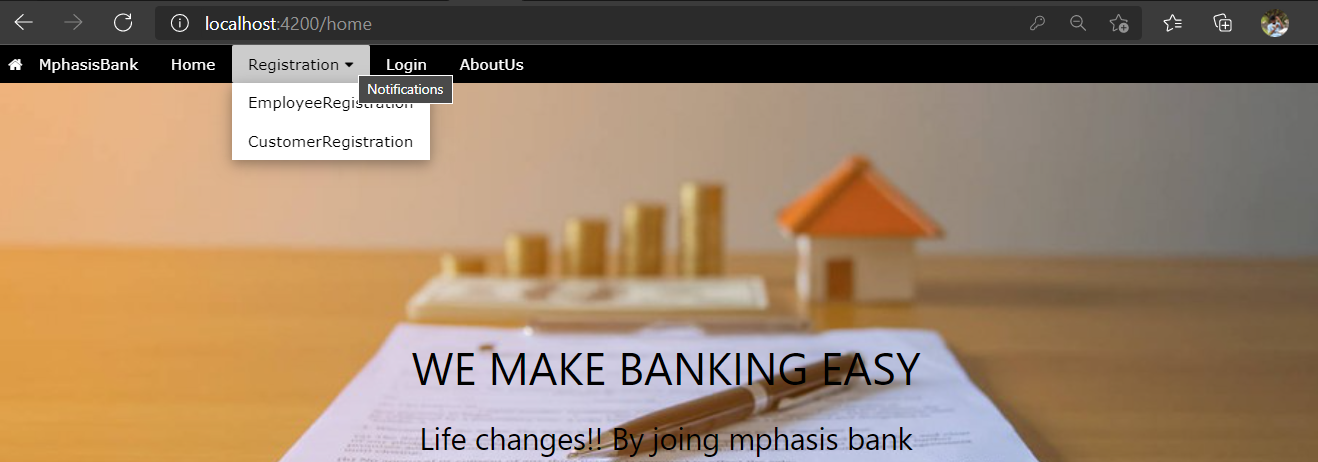
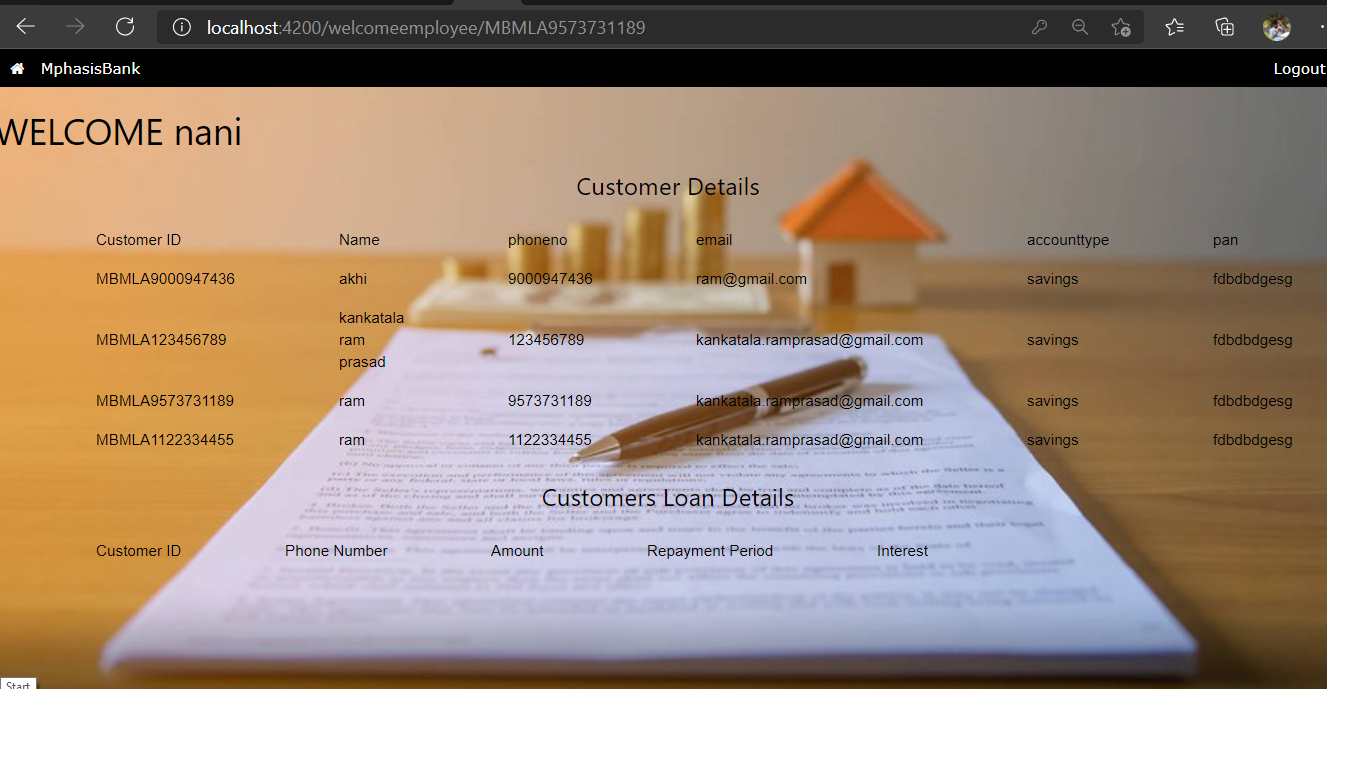
System.out.println(customer.name);

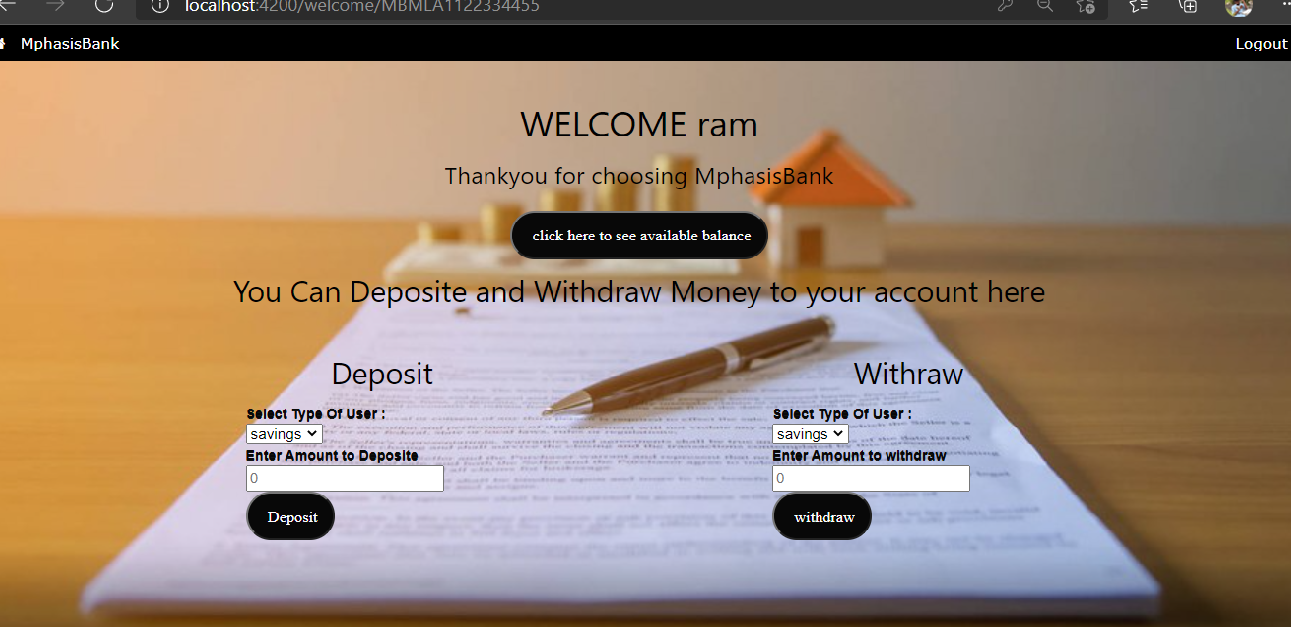
}

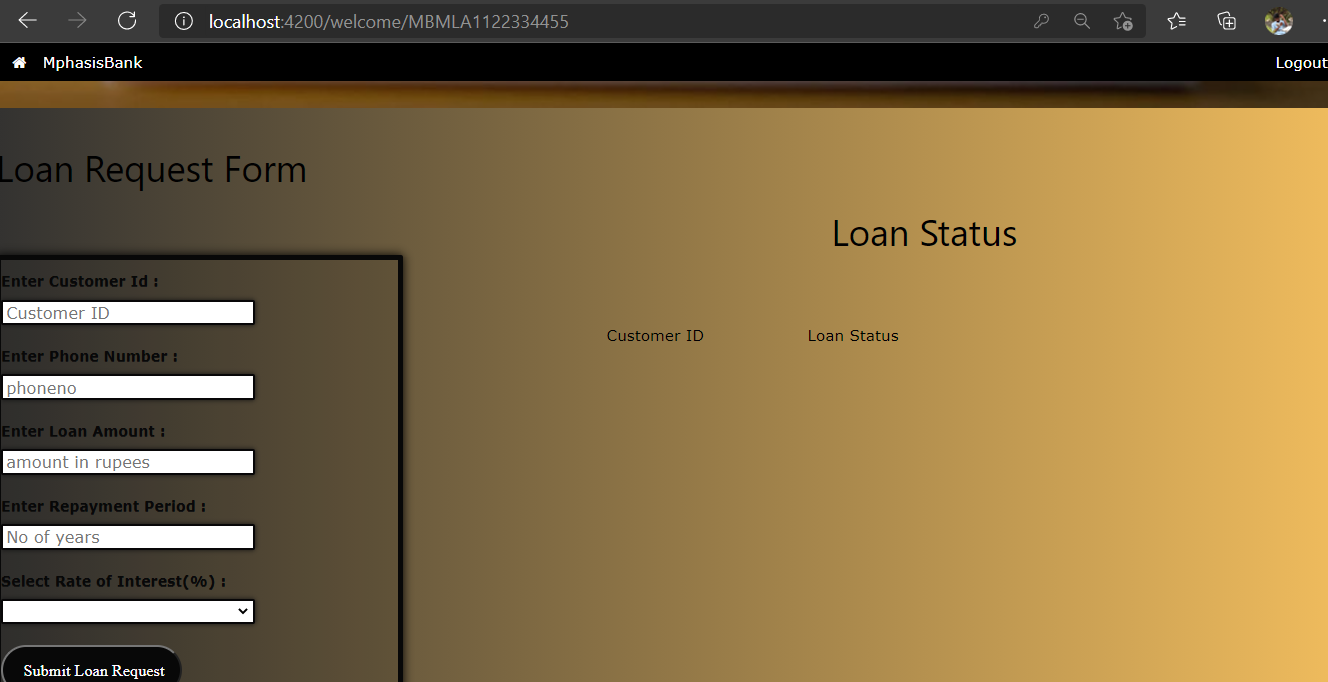
assertThat(customers).size().isGreaterThan(0);

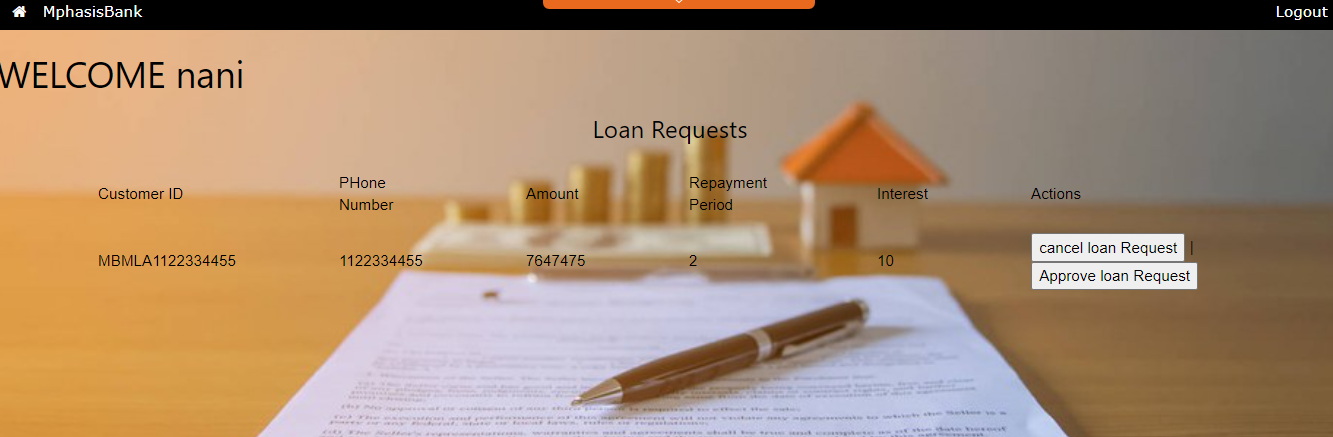
}}

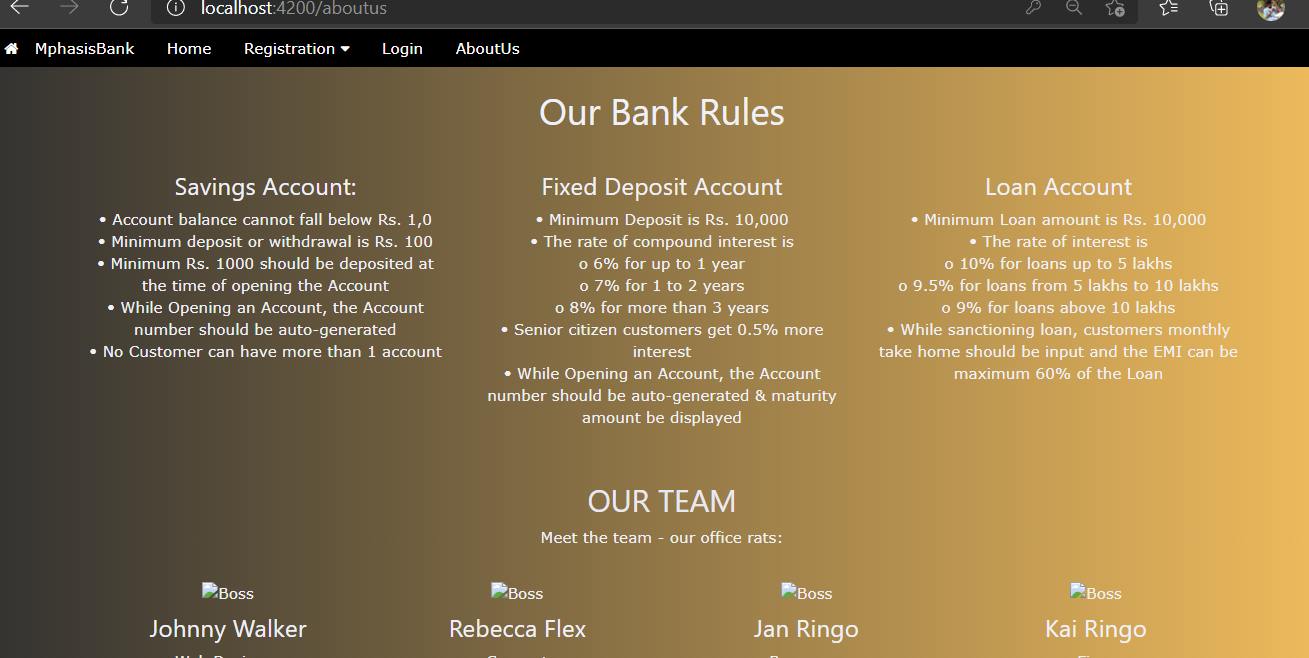
Project output:











------------------THE END-----------------