ARISGLOBAL SOFTWARE PRIVATE LIMITED

KRS Road, Mysore 570016, Karnataka, India

A Synopsis Report on

"**Bank Management System**"

By

1.Charithra S

2. M Roshini

3. Manjula B

4. Rachana V

5. Tilak Jinachandra Nayak

6. Varshitha MS

UNDER THE GUIDENCE OF

"Sarat Chandra Sarangi"

"Sri Ranga Raju K.R"

ABSTRACT

The Bank Management System is an application for maintaining a person's account in a bank. In This application we have tried to show the working of a banking account system and cover the basic functionality of a Bank Management System. The main aim of this project is to develop software for Bank Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software.

This application mainly focuses on the admin part, this ensures the admin can create the account, view user details, update and delete the user details. Bank Management System can easily accompanied with the help of this application.

ACKNOWLEDGEMENT

We are extremely thankful to Sarat Chandra Sarangi and Sri Ranga Raju K.R from Network lab for providing us the academic ambience and laboratory facilities to work, and everlasting motivation to carry out this work and shaping our careers.

We extend our gratitude to trainer's Saurabh Chaudhary, Sakshi Mohinia and Samantha Beaty, for There support and guidance over the entire course of work. We take this opportunity to thank all our Colleagues, Teammates who always stood by us in difficult situations also helped us in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to our parents who were a constant source of encouragement and stood by us as pillar of strength for completing this work and course successfully.

Table of Contents:

Overview

In this project, we designed admin portal for BANK MANAGEMENT SYSTEM. Admin logins to the system and manage all the functionalities of Banking Management System. Admin can add, edit, delete, and view the records of Customer, Accounts, Saving Accounts, Balance Admin can manage all the user details.

Design Overview

The following goals were kept in mind while designing the new system:

i) To reduce the manual works required to be done in the existing system.

ii)To avoid errors inherent in the manual works and hence make the output consistent and correct.

iii)To improve the management of permanent information of the hotel by keeping it is properly structured tables and to provide facilities to update this information as efficient as possible.

iv)To make the system complete menu-driven and hence user-friendly. This is necessary so that even non-programmers could use the system effectively and system could act as catalyst in achieving objective.

v)To design the system in such a way that reduces feature maintenance and enhancement times and efforts.

vi)To make the system reliable, understandable, and cost effective.

vi)To make the system reliable, understandable, and cost effective.

1. Introduction

Bank account system involves maintaining of account related information. This requires greater accuracy, speed that is why the proposed system is the computerization of the existing system. The computerization system does the job monitoring the record in easy and effective manner as stated below:

- Efficiently handles customer, account related data.
- Monitor transaction and makes related information.
- Keeps records of customer account detail and other information.
- Generates reports.

Account system involved maintaining data related different customer and his transaction. This required greater accuracy, speed that is why the proposed system is the computerization of the existing system. The computerized system does the job of the monitoring the information easy and effective manner.

The main aim of this “Bank Management System” is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like banking. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the application software.

Anybody who is an Account holder in this bank can become a member of Bank Management System.

2. Objective

The main Objective of the Banking management system is to provide security to the user account data. So, to provide security to the user data we gave access only to the admin. The admin can see all the user list, update it, and delete the information. so that the admin can create account, instantly for the users. Here, we are using efficient MySQL database to store and retrieve data easily without loss of data while processing the information.

The main Objective of the Bank management system is to provide security to the user account data. So, to provide security to the user data we gave access only to the admin. The admin can see all the user list, update it, and delete the information. so that the admin can create account, instantly for the users. Here, we are using efficient MySQL database to store and retrieve data easily without loss of data while processing the information.

3. Requirements

Tools:

Spring Tool Suite (STS)

Spring Tools 4 is the next generation of spring tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Their IDE.

TS IDE (Spring Tool Suite) is an eclipse-based IDE (Integrated development environment) for developing spring applications. It provides massive support for the implement, run, deploy, debug the Spring applications. Further, it allows us to build large applications from scratch.

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

Visual Studio Code

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A preview build was released shortly thereafter.

Features

Data collection: Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled. In addition, because of the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected. According to Microsoft, the data is shared with Microsoft-controlled affiliates and subsidiaries, although law enforcement may request it as part of a legal process.

Language Support: Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

Version Control: Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature you must link Visual Studio Code to any supported version control system (Git, Subversion, Perforce, etc.). This allows you to create repositories as well as make push and pull requests directly from the Visual Studio Code program.

Source-code editor

A source-code editor is a text editor program designed specifically for editing source code of computer programs. It may be a standalone application, or it may be built into an integrated development environment.

Angular CLI

The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell. Angular CLI helps developers to create projects easily and quickly. As we know already, Angular CLI tool is used for development and built on top of Node.js, installed from NPM

Install the CLI using the npm package manager:

npm install -g @angular/cl

Xampp

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database,

and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

Features

XAMPP is regularly updated to the latest releases of Apache, MariaDB, PHP and Perl. It also Comes with a number of other module including OpenSSL, phpMyAdmin, MediaWiki, Joomla, WordPress and more. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another. XAMPP is offered in both a full and a standard version (Smaller version).

Free and open-source software

Free and open-source software (FOSS) is software that is both free software and open-source software where anyone is freely licensed to use, copy, study, and change the software in any way, and the source.

Languages

HTML

HTML is the language for describing the structure of Web pages. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spreadsheets, video clips, sound clips, and other applications directly in their documents.

With HTML, authors describe the structure of pages using *markup*. The *elements* of the language label piece of content such as “paragraph,” “list,” “table,” and so on
W3C HTML

CSS CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments.

Java:

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

High-level programming language in computer science, a high-level programming language is a programming language with strong abstraction from the details of the computer. In contrast to low-level programming languages, it may use natural.

Bootstrap:

Bootstrap is a potent front-end framework used to create modern websites and web apps. It's open-source and free to use yet features numerous HTML and CSS templates for UI interface elements such as buttons and forms. Bootstrap also supports JavaScript extensions. Software engineers use Bootstrap for several different reasons. It is easy to set up and master, it has a lot of components, a good grid system, styling for many HTML elements ranging from typography to buttons, as well as support of JavaScript plugins, making it even more flexible. Bootstrap is great for creating layouts, as its responsive CSS is designed to conform to different devices. It can be employed to ensure consistency, eliminate cross-browser issues, and so on.

MySQL:

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to implement, manage, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, data-driven B2B services. Its open-source nature, stability, and rich feature set paired with ongoing development.

Uniting Business Intelligence with Data Analytics for Data-Driven Insights
Business intelligence and data analytics work together for the best results.

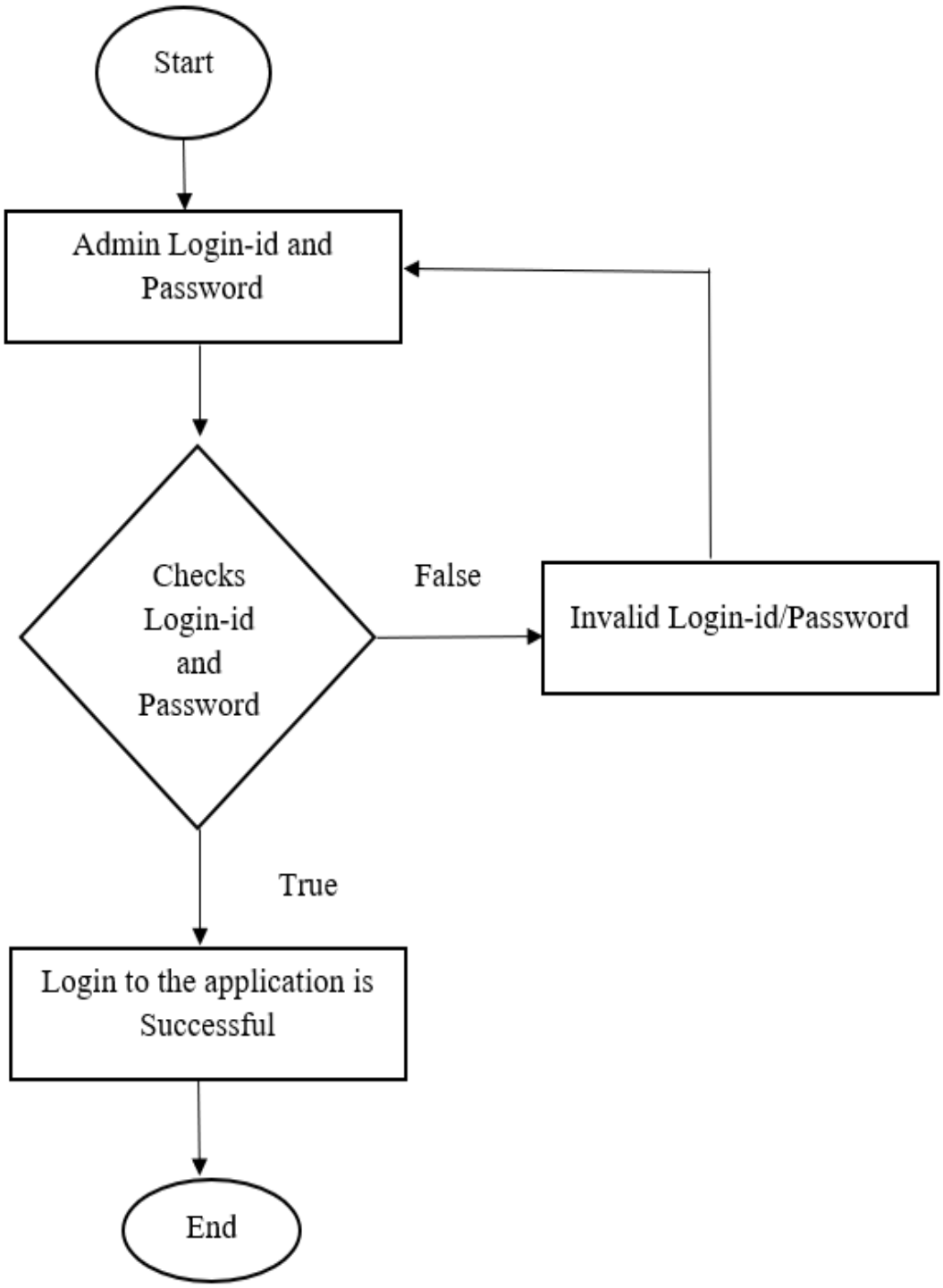
Business Requirements

- The admin should login to the system with correct user id and password.
- The existing user details will be fetched from database.
- All the user details will be displayed
- The admin can create the user account and the account number will be auto generated in the database.
- The update of the user details can be done by the admin, the corresponding account number of the user will be used to update his/her information.

- The deletion operation is like that of update, the user details can be deleted just in one click of delete button.

Class Diagram

The application consists of the following classes



4. Implementation:

Source code:

Createacc.component.css

```
@import url("https://fonts.googleapis.com/css?family=Lato:400,700");
body {
  background-color: #beb3f1;
  font-family: 'Lato', sans-serif;
  color: #4A4A4A;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  overflow: hidden;
  margin: 0;
  padding: 0;
}
h1{
  padding-left: 70px;
  padding-bottom: 30px;
  font-size: 30px;
  }
```

```
form {
  width: 350px;
  position: relative;
}
form .form-field::before {
  font-size: 20px;
  position: absolute;
  left: 15px;
  top: 17px;
  color: #888888;
  content: " ";
  display: block;
  background-size: cover;
  background-repeat: no-repeat;
}
form .form-field:nth-child(1)::before {
  width: 20px;
  height: 20px;
  top: 15px;
}
form .form-field:nth-child(2)::before {
  width: 25px;
  height: 25px;
```

```
  top: 15px;
}
form .form-field:nth-child(3)::before {
  width: 25px;
  height: 25px;
  top: 15px;
}
form .form-field {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-pack: justify;
  -ms-flex-pack: justify;
  justify-content: space-between;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  margin-bottom: 1rem;
  position: relative;
}
form input {
  font-family: inherit;
  width: 100%;
```

```
  outline: none;
  background-color: #fff;
  border-radius: 4px;
  border: none;
  display: block;
  padding: 0.9rem 0.7rem;
  box-shadow: 0px 3px 6px rgba(0, 0, 0, 0.16);
  font-size: 17px;
  color: #4A4A4A;
  text-indent: 40px;
}
form .btn {
  outline: none;
  border: none;
  cursor: pointer;
  display: inline-block;
  margin: 0 auto;
  padding: 0.9rem 2.5rem;
  text-align: center;
  background-color: #47AB11;
  color: #fff;
  border-radius: 4px;
  box-shadow: 0px 3px 6px rgba(0, 0, 0, 0.16);
```

```
  font-size: 17px;
}
```

Createacc.component.html

```
<app-navbar></app-navbar>
<body>
  <form #myForm="ngForm" (ngSubmit)="onSubmit(myForm.value)">
    <h1>Create Account</h1>
  <div class="form-field">
    <input type="name" placeholder="Name of the Person" name="name" ngModel/>
  </div>
  <div class="form-field">
    <input type="email" placeholder="Email Id" name="email" ngModel/>
</div>
    <div class="form-field">
      <input type="phno" placeholder="Phone Number" name="phno" ngModel/>
    </div>
  <div class="form-field">
    <button class="btn" type="submit">Create Account</button>
</div>
</form>
</body>
```

Createacc.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
// import * as internal from 'stream';
import { CreateaccService } from '../createacc.service';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { createacc } from '../createacc';
// import { LoginService } from '../login.service';

@Component({
  selector: 'app-createacc',
  templateUrl: './createacc.component.html',
  styleUrls: ['./createacc.component.css']
})

export class CreateaccComponent implements OnInit {
  createaccs!:createacc[];
  constructor(private createaccService:CreateaccService,private https:HttpClient) { }
  onSubmit(data:createacc)
  {
```

```
this.https.post('http://localhost:8090/api/creaacc',data).subscribe((result)=>console.warn("result",result))
;

    alert("account created successfully");

    console.warn(data);

  }

  ngOnInit(): void {

  this.createaccService.getCreateacc().subscribe((data:createacc[])=>{

    console.log(data);

    this.createaccs=data;

  });


  }

}
```

Createacc.service.ts

```
import { Injectable } from '@angular/core';

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { Observable, observable } from 'rxjs';

import { createacc } from './createacc';


@Injectable({
```

```
  providedIn: 'root'
})
export class CreateaccService {

  private baseUrl="http://localhost:8090/api/creaaac";

  constructor(private http:HttpClient) { }
  getCreateacc():Observable<createacc[]>

   {

     return  this.http.get<createacc[]>(`${this.baseUrl}`);

   }

}
```

Createacc.ts

```
export class createacc {
    name!:string;
    email!:string;
    phno!:string;
}
```

nav.component.html

```
<mat-toolbar color="primary" class="mat-elevation-z6" class="navbar">
    <span> <mat-icon>home</mat-icon>Bank Application</span>
        <div class="spacer"></div>
  </mat-toolbar>
```

Nav.component.ts

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


@Component({
  selector: 'app-nav',
  templateUrl: './nav.component.html',
  styleUrls: ['./nav.component.css']
})
export class NavComponent implements OnInit {


  constructor() { }


  ngOnInit(): void {
  }
}
```

Navbar.component.html

```
<mat-toolbar color="primary" class="mat-elevation-z6" class="navbar">
    <span> <mat-icon>home</mat-icon>Bank Application</span>
        <div class="spacer"></div>
    <a mat-button routerLink="/home">Home</a>
    <a mat-button routerLink="/createacc">Create Account</a>
    <a mat-button routerLink="/userdetails">User Details</a>
    <a mat-button routerLink="/login">Logout</a>
    <a mat-button routerLink="/deposite">About</a>
  </mat-toolbar>
<router-outlet></router-outlet>
```

Navbar.component.ts

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


@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {
```

```
  constructor() { }


  ngOnInit(): void {

  }

}
```

Signin.component.html

```
<app-nav></app-nav>

<body>

<form #myForm="ngForm" (ngSubmit)="onSubmit(myForm)">

  <h1>Sign In</h1>

  <div class="form-field">

    <input type="id" placeholder="Email / Username" name="id" id="id" required [(ngModel)]="id"/>

  </div>


  <div class="form-field">

    <input type="password" placeholder="Password" name="password" id="password" required [(ngModel)]="password"/>

  </div>


  <div class="form-field">

    <button class="btn" >login</button>

  </div>

</form>
```

```
</body>
```

Signin.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { LoginService } from '../login.service';

@Component({
  selector: 'app-signin',
  templateUrl: './signin.component.html',
  styleUrls: ['./signin.component.css']
})
export class SigninComponent implements OnInit {

  constructor(private router:Router,private login:LoginService) { }

  ngOnInit(): void {
  }
  id!:string;

  password!:string;
```

```
onSubmit(formValue:NgForm){
 console.log(formValue);
  if(this.id=="admin" && this.password=="12345678")

  {
    alert("login successfull");
  this.router.navigate(['/home']) }

  else if(this.id==" " && this.password==" ")

  {
    alert("please enter username and password!");
    }
else{
  alert("Please enter correct username and password.");
}

}

}
```

Update.component.html

```
<app-nav></app-nav>


<body>
   <form #myForm="ngForm" (ngSubmit)="onSubmit(myForm.value)">
      <h1>Update</h1>
    <div class="form-field">
      <input type="name" placeholder="Name of the Person" name="name" ngModel/>
    </div>


    <div class="form-field">
      <input type="email" placeholder="Email Id" name="email" ngModel/>
   </div>
      <div class="form-field">
        <input type="phno" placeholder="Phone Number" name="phno" ngModel/>
      </div>
      <div class="form-field">
        <input type="id" placeholder="Account Number" name="id" ngModel/>
      </div>
      <div class="form-field">
          <input type="balance" placeholder="Balance" name="balance" ngModel/>
        </div>
```

```
    <div class="form-group">

        <button type="submit" class="btn btn-primary" routerLink="/update" >Update </button>  
        <br>
           <br>

        <button type="submit" class="btn btn-primary" routerLink="/userdetails">View User details</button>

        </div>

  <!-- <button onclick="togglePopup()" class="first-button">Create Account</button> -->

</form>
</body>
```

Update.component.ts

```
import { HttpClient } from '@angular/common/http';

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

```
import { Userdetails } from '../userdetails';

@Component({
  selector: 'app-update',
  templateUrl: './update.component.html',
  styleUrls: ['./update.component.css']
})

export class UpdateComponent implements OnInit {

  id!:string;



  constructor( private httpClient:HttpClient) { }

  ngOnInit(): void {}

  onSubmit(data:Userdetails)

  {

    this.httpClient.post('http://localhost:8090/api/updateuser/' + `${data.id}`,data).subscribe((result)=>
```

```
    {
      alert("Data updated successfully!");
      console.warn("result",this.id);

    console.warn(data);

  })

}

}
```

Update.ts

```
export class Update {
id!:BigInteger;
name!:string;
email!:string;
phno!:string;
balance!:BigInteger;

}
```

Update.service.ts

```
import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

import { Userdetails} from './userdetails';

@Injectable({
  providedIn: 'root'
})
export class UpdateService {

  private baseUrl = "http://localhost:8090/api/updateuser/";

  constructor(private http:HttpClient) { }



  getCreaacc():Observable<Userdetails[]>{

    return this.http.get<Userdetails[]>(`${this.baseUrl}`);
```

```
  }


}
```

Userdetails.component.html

```
<app-navbar></app-navbar>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script>

<body>

<div class="container">

<div class="border">

</div>

<h1>User Details</h1>

  <table class="table table-bordered table-striped">

    <thead>

    <tr>

      <!-- <th>ID</th> -->
```

```
  <th>Account Number</th>
  <th>Name</th>
  <th>Phone</th>
  <th>Email Id</th>
  <th>Balance</th>
  <th>Action</th>
</tr>
</thead>
<tbody>

  <tr *ngFor="let userdetail of userdetails">
    <td><span>{{userdetail.id}}</span></td>
  <!-- <td><span>{{userdetail.accountnumber}}</span></td> -->
  <td><span>{{userdetail.name}}</span></td>
  <td><span>{{userdetail.phno}}</span></td>
  <td><span>{{userdetail.email}}</span></td>
  <td><span>{{userdetail.balance}}</span></td>

  <td>

  <button class="btn btn-sm btn-warning" routerLink="/update">
  <i class="fa fa-edit"></i>Edit</button>
```

```
      <button class="btn btn-sm btn-danger" (click)="ondelete(userdetail)">

      <i class="fa fa-remove"></i>Delete</button>
      </td>
    </tr>
    </tbody>
  </table>
</div>
</body>
```

Userdetails.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Userdetails } from '../userdetails';
import { UserdetailsService } from '../userdetails.service';

@Component({
  selector: 'app-userdetails',
  templateUrl: './userdetails.component.html',
  styleUrls: ['./userdetails.component.css']
})
```

```
export class UserdetailsComponent implements OnInit {

  userdetails!:Userdetails[];

  constructor(private createaccService:UserdetailsService,private https:HttpClient) { }


  ngOnInit(): void {

  this.createaccService.getAllUser().subscribe((data:Userdetails[])=>{

    console.log(data);

    this.userdetails=data;

  });


  }

  ondelete(data:Userdetails){

this.https.post('http://localhost:8090/api/delete/'+`${data.id}`,data).subscribe((result)=>{console.log(res
ult)})

    alert("Do You Want to delete data?");

    window.location.reload();




  }

}
```

Application.java

```
package com.example.demo.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.demo.model.Creaacc;
	@Repository
	public interface CreaaccRepository extends JpaRepository<Creaacc, Integer> {
	}
```

CreaaacController.java

```
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.model.Creaacc;
import com.example.demo.repository.CreaaccRepository;
@CrossOrigin(origins="http://localhost:4200")
@RestController
@RequestMapping("/api")
public class CreaaacContoller {
```

```
@Autowired
CreaaccRepository repository;
@GetMapping("/Getcreaaac")
public List<Creaacc> getAllcreaaac()
{
	return repository.findAll();
}
@PostMapping("/creaacc")
public void savecreaacc(@RequestBody Creaacc cc)
{
	repository.save(cc);
}
@PostMapping("/delete/{id}")
public void deletecreaacc(@PathVariable Integer id)
{
repository.deleteById(id);
}
@PostMapping("/updateuser/{id}")

public void updatecreaacc(@PathVariable Integer id, @RequestBody Creaaccc)
{
repository.getById(id);
repository.save(c);
```

```
	}
	}
```

Creaacc.java

```
package com.example.demo.model;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="creaacc")
public class Creaacc
 {

		String name;
		@Id
		int id;
		String email;
		String phno;
		int balance;
		int accountnumber;

		public String getName() {
			return name;
		}
```

```
public void setName(String name) {
        this.name = name;
}
public int getId() {
        return id;
}
public void setId(int id) {
        this.id = id;
}
public String getEmail() {
        return email;
}
public void setEmail(String email) {
        this.email = email;
}
public String getPhno() {
        return phno;
}
public void setPhno(String phno) {
        this.phno = phno;
}
public int getBalance() {
        return balance;
```

```
	}
	public void setBalance(int balance) {
		this.balance = balance;
	}
	public int getAccountnumber() {
		return accountnumber;
	}
	public void setAccountnumber(int accountnumber) {
		this.accountnumber = accountnumber;
	}
	public Creaacc(String name, String email, String phno, int id,int accountnumber,int balance)
	{
		super();
		this.name=name;
		this.email=email;
		this.phno=phno;
		this.id=id;
		this.accountnumber=accountnumber;
		this.balance=balance;

	}
	public Creaacc()
```

```
        {

        }

        }
```

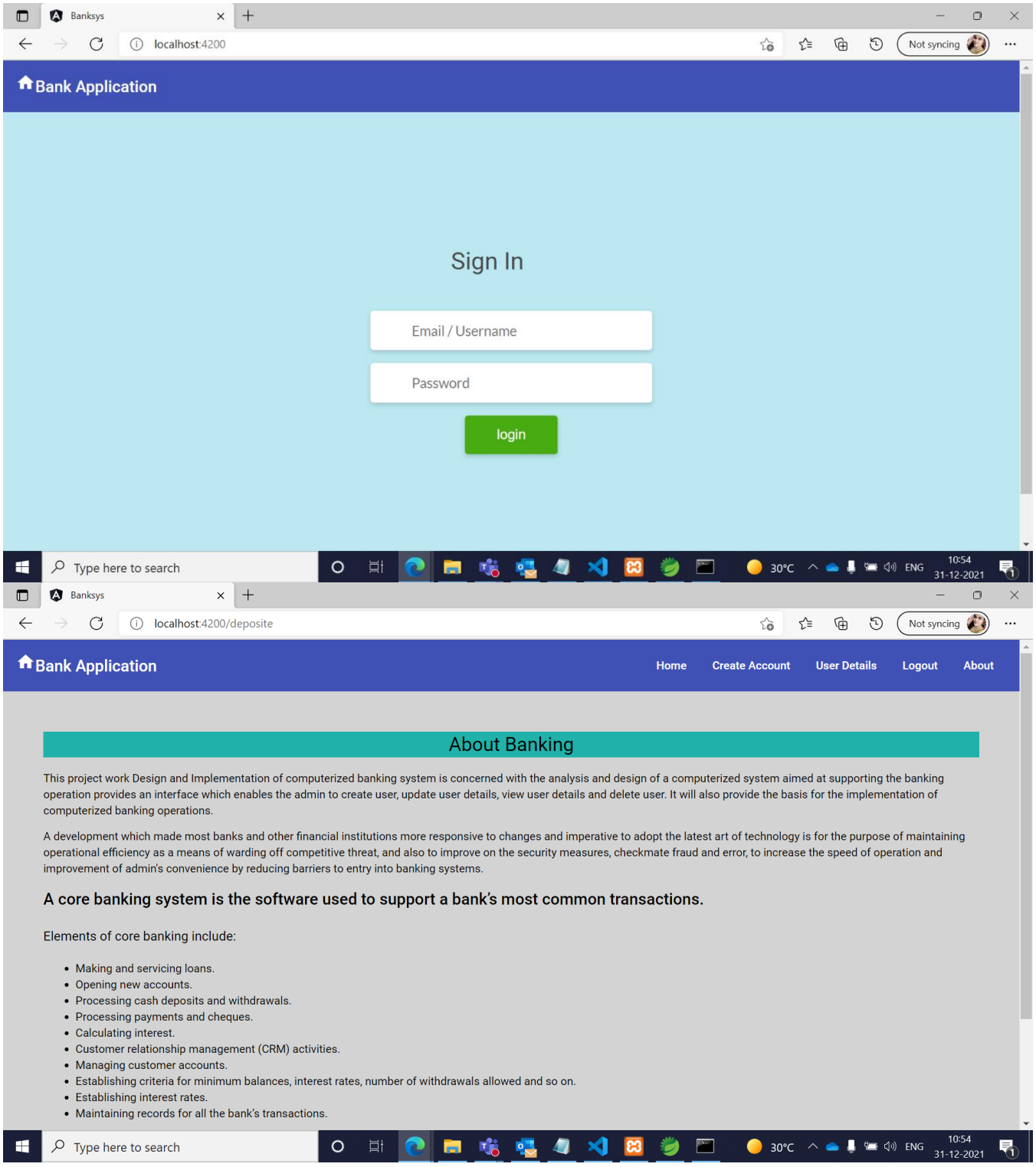
CreaaccRepository.java

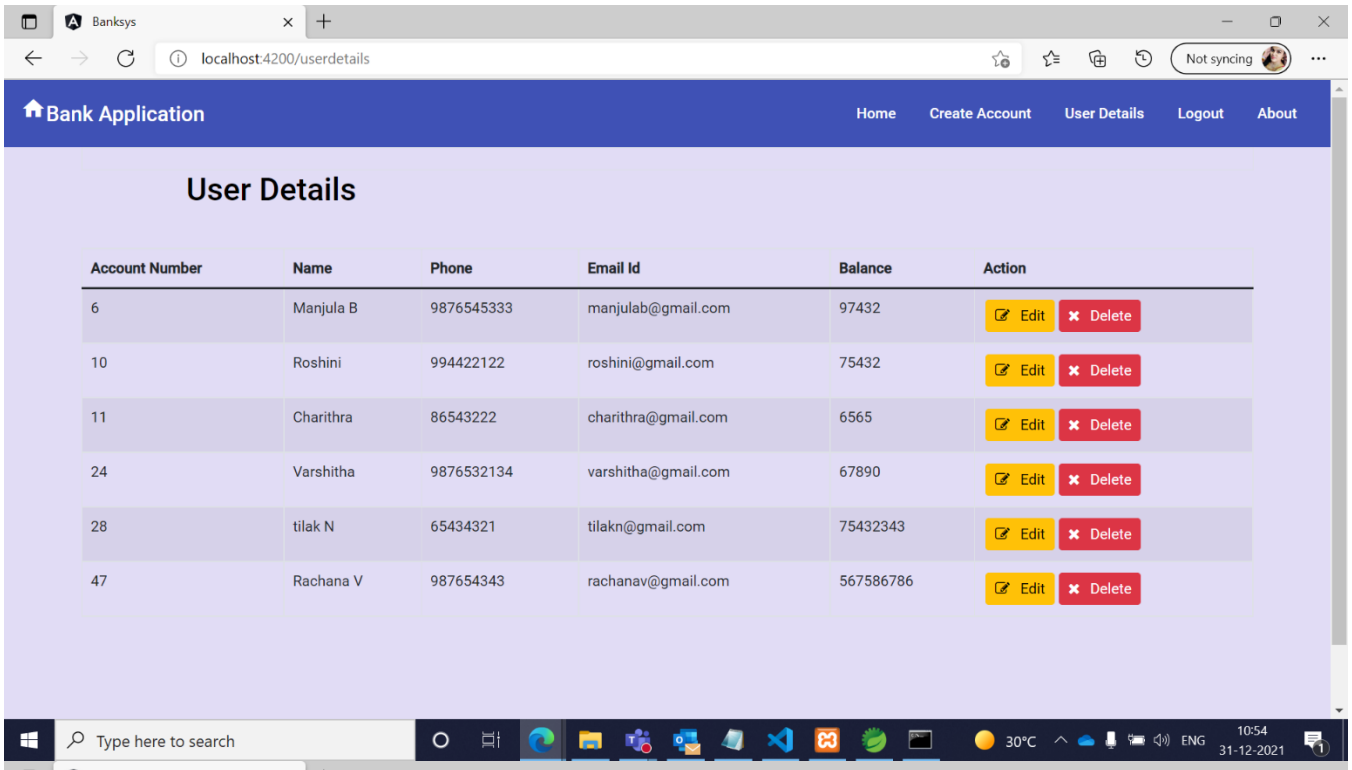
```
package com.example.demo.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.demo.model.Creaacc;

    @Repository

    public interface CreaaccRepository extends JpaRepository<Creaacc, Integer> {

    }
```

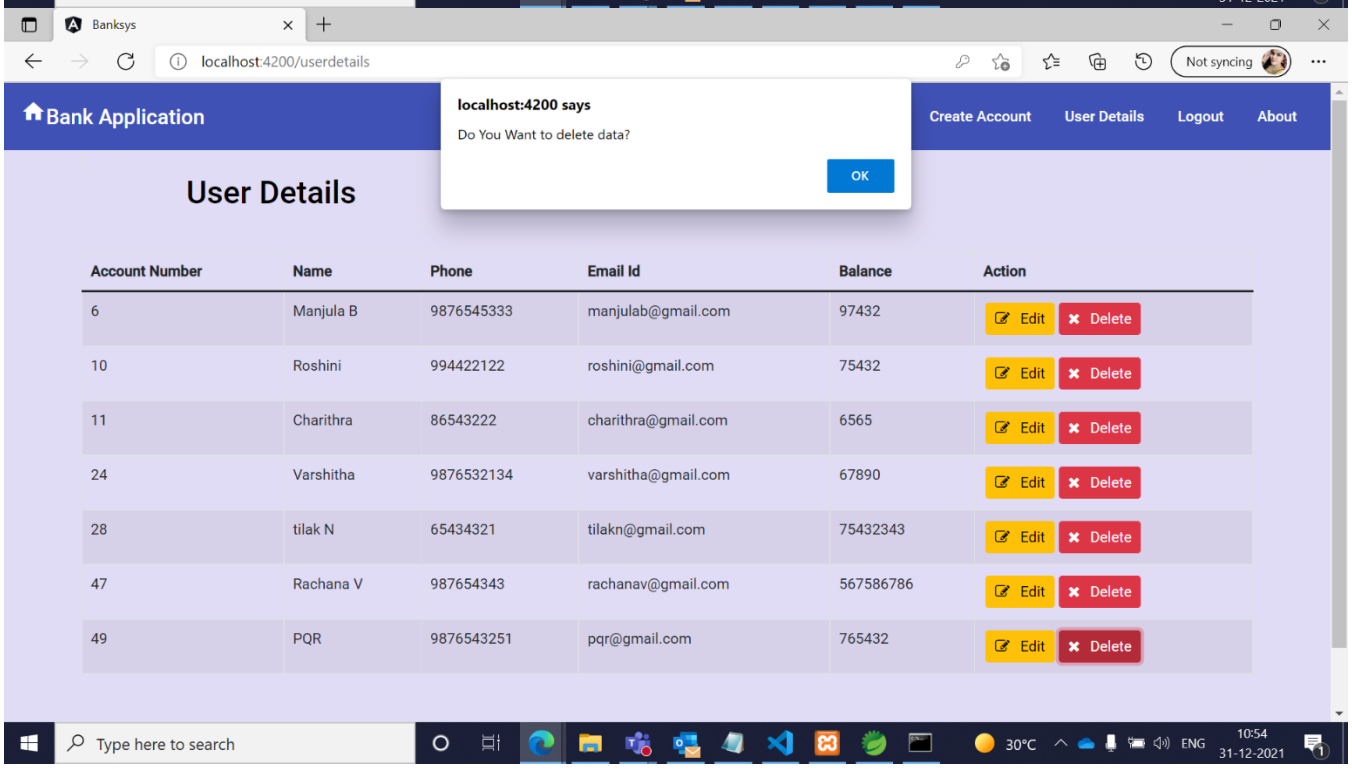
Application.properties

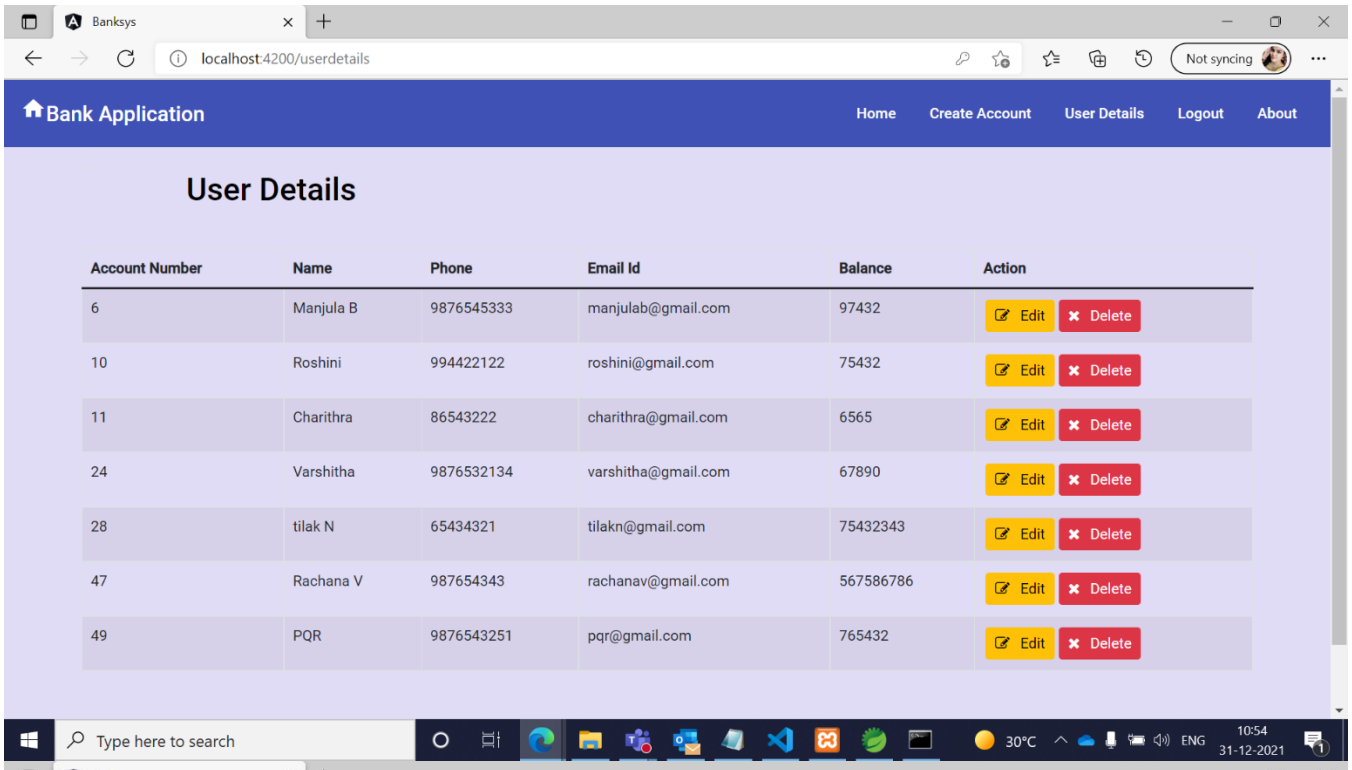
```
server.port=8090

spring.datasource.url=jdbc:mysql://localhost:3306/database

spring.datasource.username=root

spring.datasource.password=
```

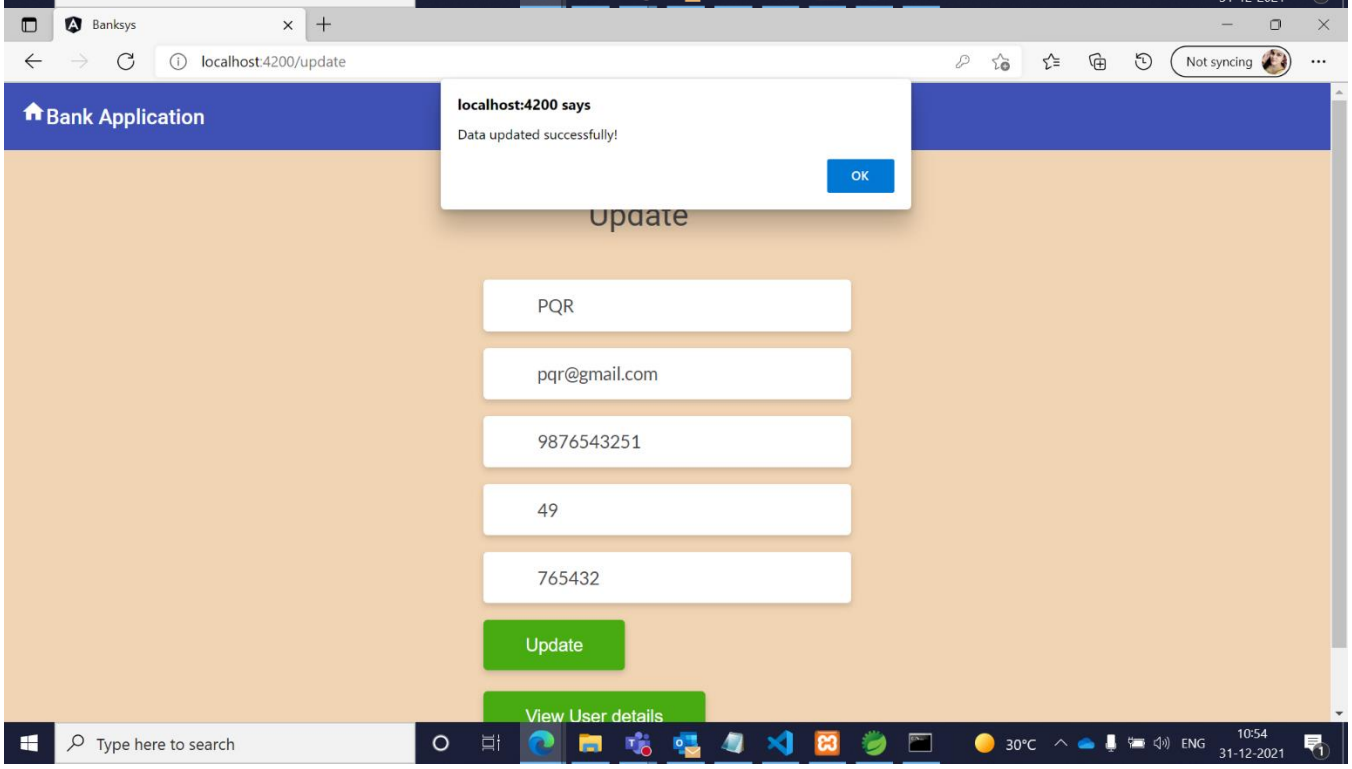
5. Output:

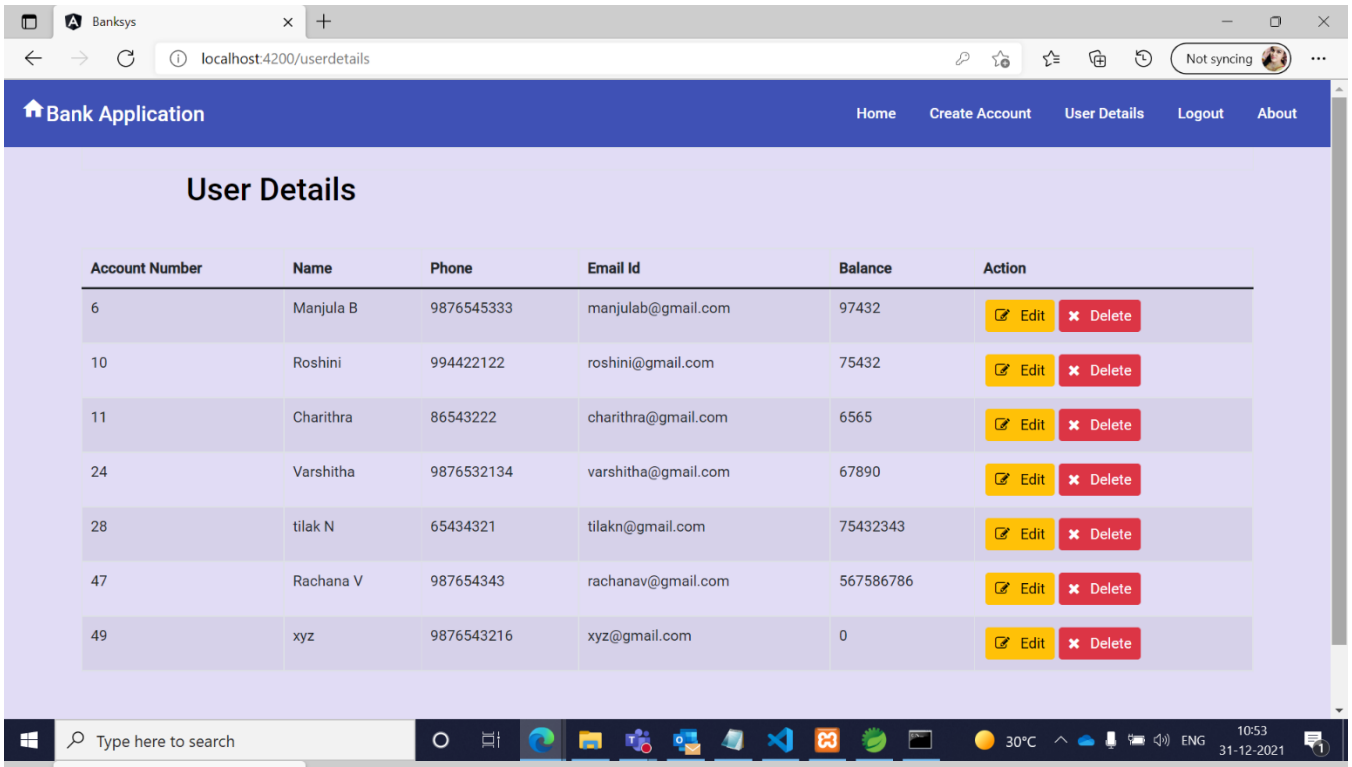

Banksys
localhost:4200
Not syncing
Bank Application
Sign In
Email / Username
Password
login
Type here to search
30°C
ENG
10:54
31-12-2021
Banksys
localhost:4200/deposite
Not syncing
Bank Application
Home
Create Account
User Details
Logout
About
About Banking
This project work Design and Implementation of computerized banking system is concerned with the analysis and design of a computerized system aimed at supporting the banking operation provides an interface which enables the admin to create user, update user details, view user details and delete user. It will also provide the basis for the implementation of computerized banking operations.
A development which made most banks and other financial institutions more responsive to changes and imperative to adopt the latest art of technology is for the purpose of maintaining operational efficiency as a means of warding off competitive threat, and also to improve on the security measures, checkmate fraud and error, to increase the speed of operation and improvement of admin's convenience by reducing barriers to entry into banking systems.
A core banking system is the software used to support a bank's most common transactions.
Elements of core banking include:
Making and servicing loans.
Opening new accounts.
Processing cash deposits and withdrawals.
Processing payments and cheques.
Calculating interest.
Customer relationship management (CRM) activities.
Managing customer accounts.
Establishing criteria for minimum balances, interest rates, number of withdrawals allowed and so on.
Establishing interest rates.
Maintaining records for all the bank's transactions.
Type here to search
30°C
ENG
10:54
31-12-2021

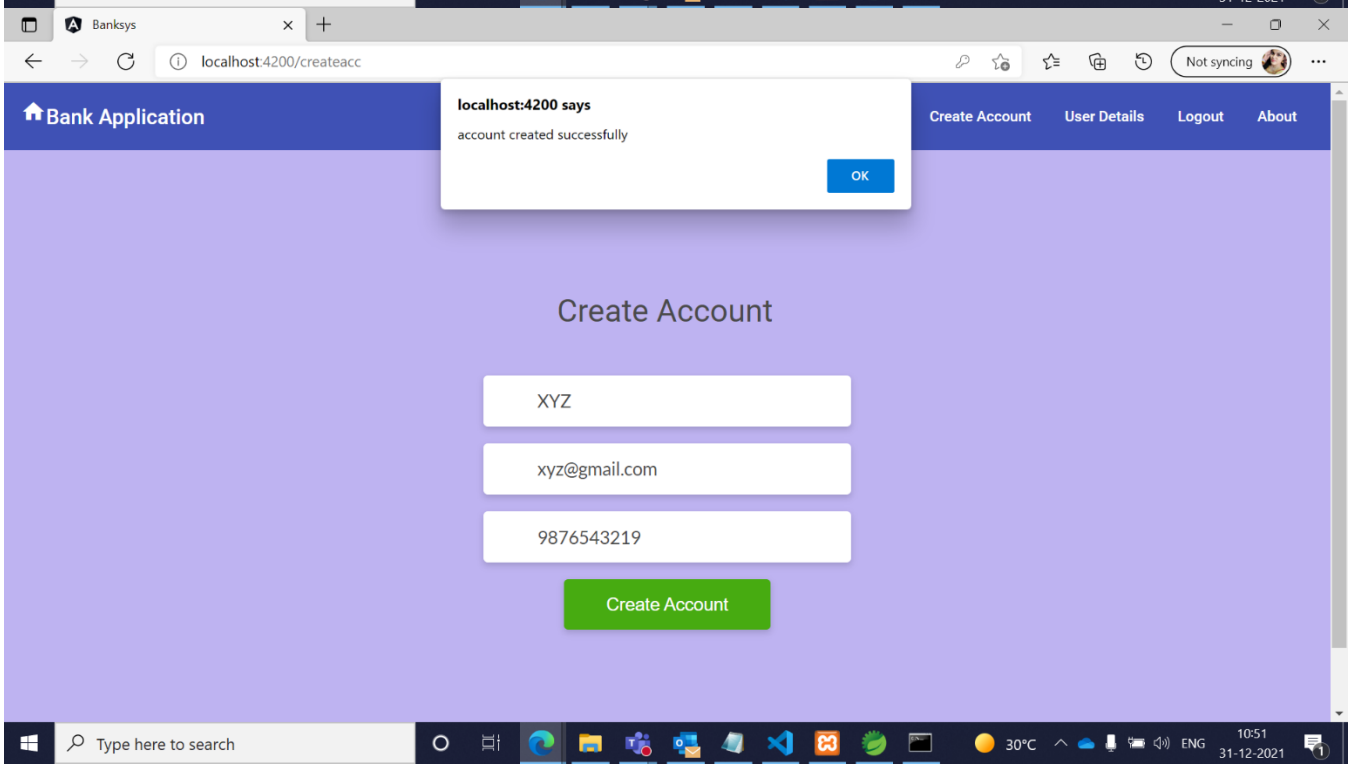

Banksys
localhost:4200/userdetails
Not syncing
Bank Application
Home
Create Account
User Details
Logout
About
User Details
Account Number
Name
Phone
Email Id
Balance
Action
6
Manjula B
9876545333
manjulab@gmail.com
97432
Edit
Delete
10
Roshini
994422122
roshini@gmail.com
75432
11
Charithra
86543222
charithra@gmail.com
6565
24
Varshitha
9876532134
varshitha@gmail.com
67890
28
tilak N
65434321
tilakn@gmail.com
75432343
47
Rachana V
987654343
rachanav@gmail.com
567586786
Type here to search
30°C
ENG
10:54
31-12-2021

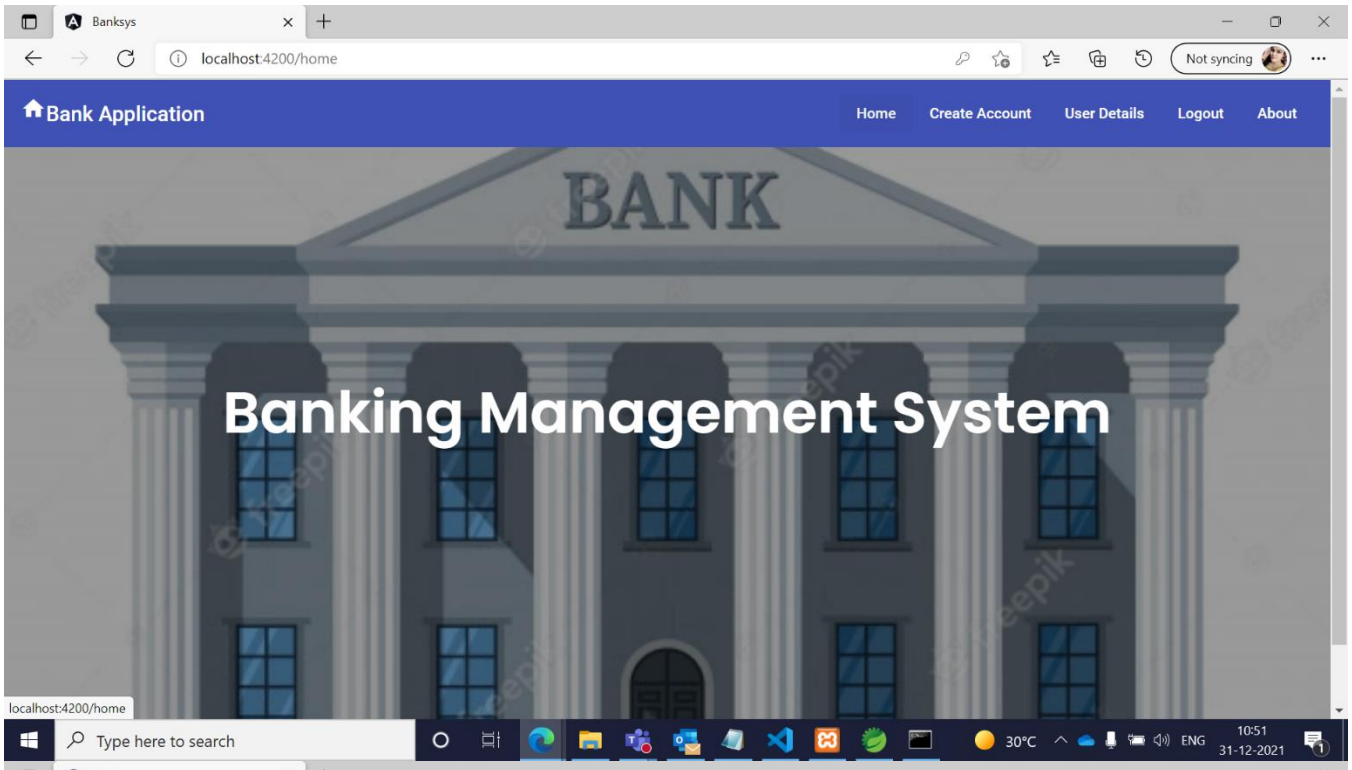

Banksys
localhost:4200/userdetails
Not syncing
localhost:4200 says
Do You Want to delete data?
OK
Bank Application
Create Account
User Details
Logout
About
User Details
Account Number
Name
Phone
Email Id
Balance
Action
6
Manjula B
9876545333
manjulab@gmail.com
97432
Edit
Delete
10
Roshini
994422122
roshini@gmail.com
75432
11
Charithra
86543222
charithra@gmail.com
6565
24
Varshitha
9876532134
varshitha@gmail.com
67890
28
tilak N
65434321
tilakn@gmail.com
75432343
47
Rachana V
987654343
rachanav@gmail.com
567586786
49
PQR
9876543251
pqr@gmail.com
765432
Type here to search
30°C
ENG
10:54
31-12-2021

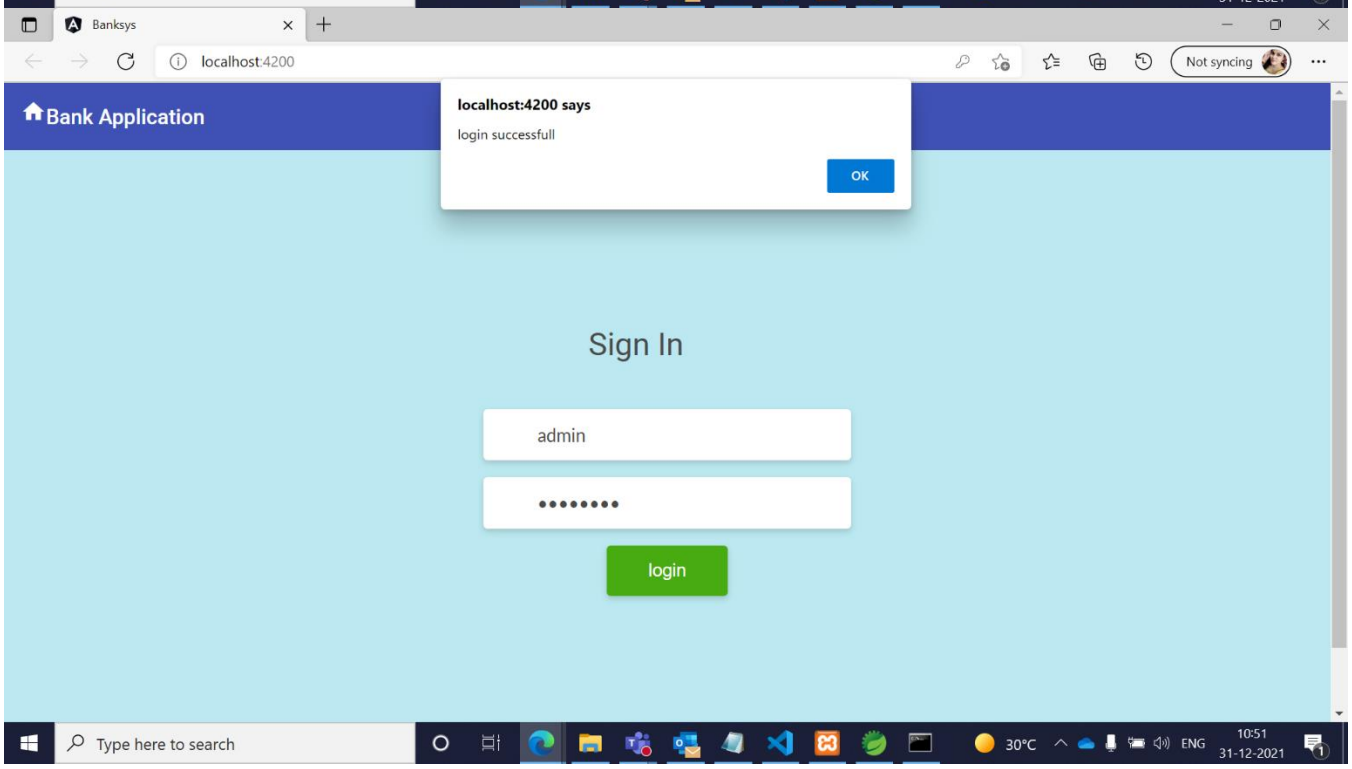

Banksys
localhost:4200/userdetails
Not syncing
Bank Application
Home
Create Account
User Details
Logout
About
User Details
Account Number
Name
Phone
Email Id
Balance
Action
6
Manjula B
9876545333
manjulab@gmail.com
97432
Edit
Delete
10
Roshini
994422122
roshini@gmail.com
75432
Edit
Delete
11
Charithra
86543222
charithra@gmail.com
6565
Edit
Delete
24
Varshitha
9876532134
varshitha@gmail.com
67890
Edit
Delete
28
tilak N
65434321
tilakn@gmail.com
75432343
Edit
Delete
47
Rachana V
987654343
rachanav@gmail.com
567586786
Edit
Delete
49
PQR
9876543251
pqr@gmail.com
765432
Edit
Delete
Type here to search
30°C
ENG
10:54
31-12-2021

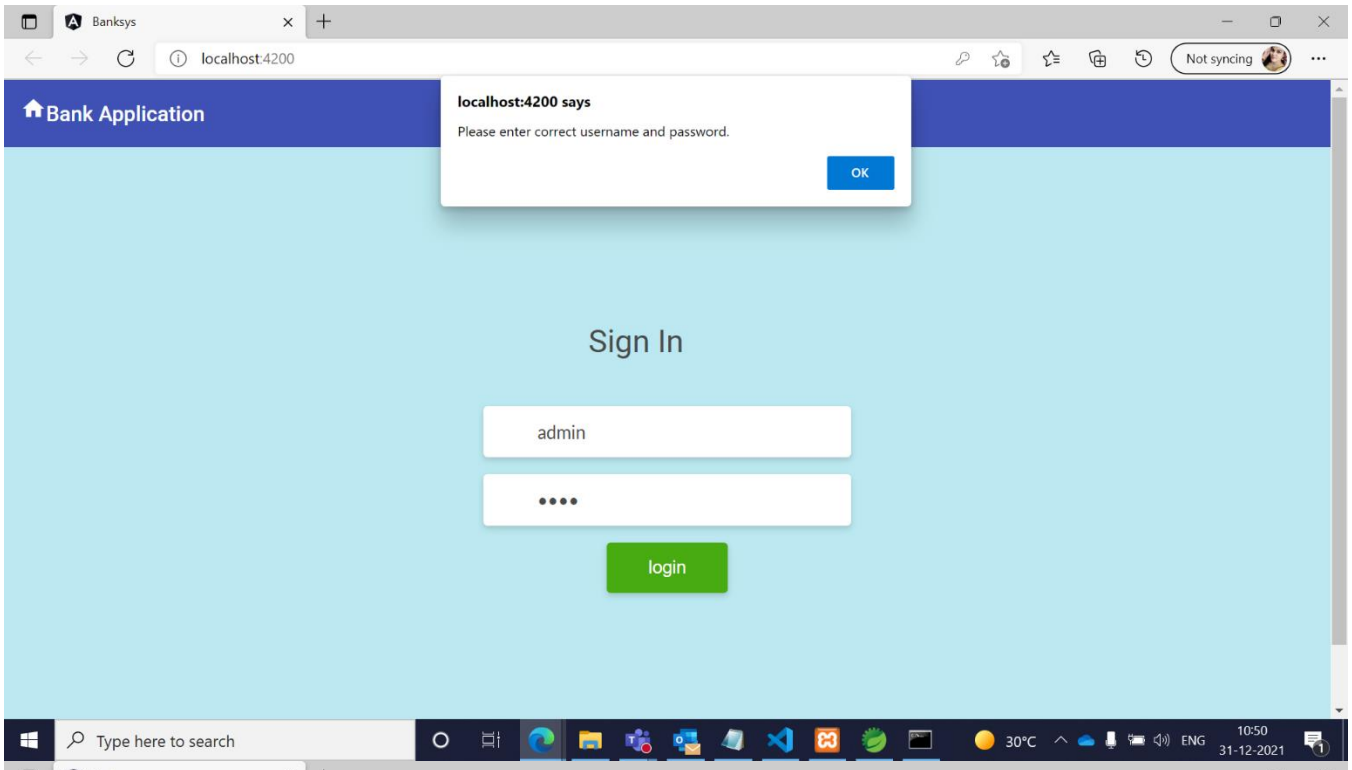

Banksys
localhost:4200/update
Not syncing
Bank Application
localhost:4200 says
Data updated successfully!
OK
Update
PQR
pqr@gmail.com
9876543251
49
765432
Update
View User details
Type here to search
30°C
ENG
10:54
31-12-2021

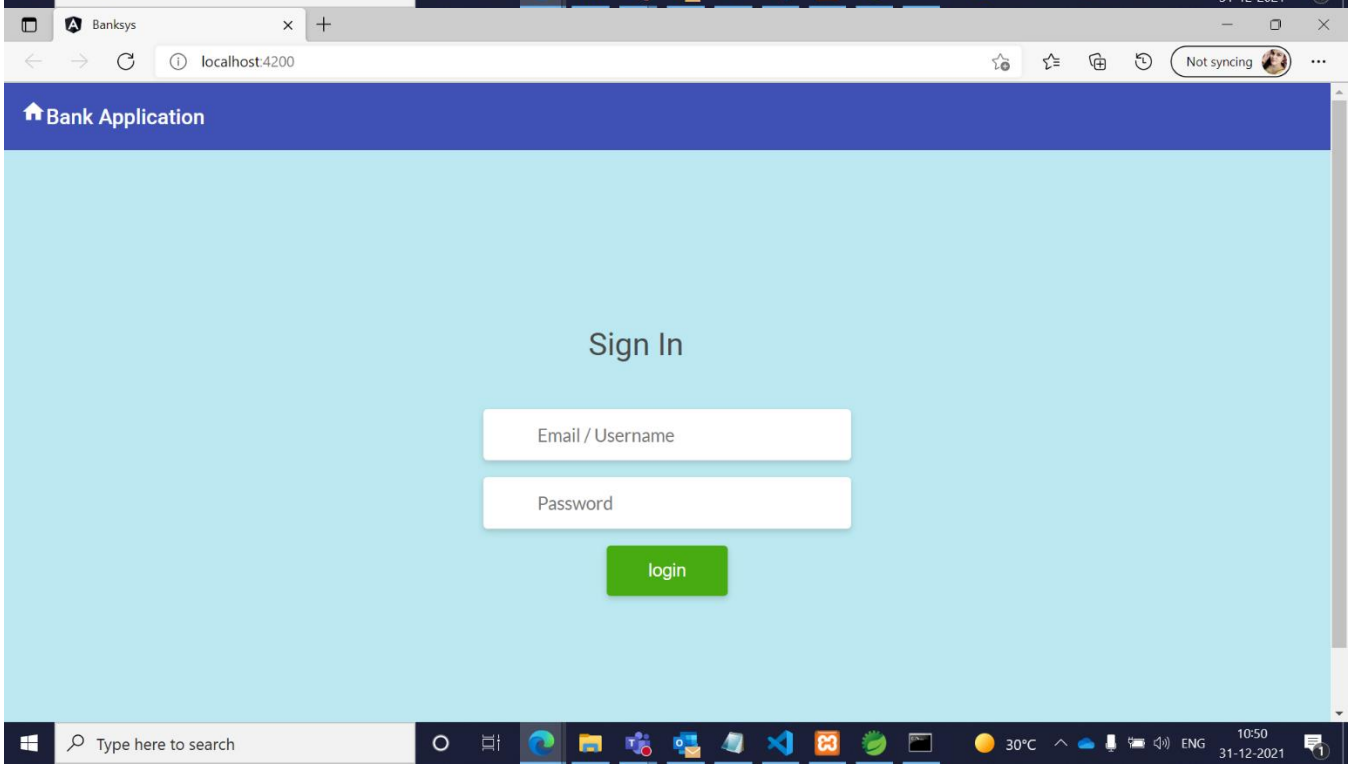

Banksys
localhost:4200/userdetails
Not syncing
Bank Application
Home
Create Account
User Details
Logout
About
User Details
Account Number
Name
Phone
Email Id
Balance
Action
6
Manjula B
9876545333
manjulab@gmail.com
97432
Edit
Delete
10
Roshini
994422122
roshini@gmail.com
75432
Edit
Delete
11
Charithra
86543222
charithra@gmail.com
6565
Edit
Delete
24
Varshitha
9876532134
varshitha@gmail.com
67890
Edit
Delete
28
tilak N
65434321
tilakn@gmail.com
75432343
Edit
Delete
47
Rachana V
987654343
rachanav@gmail.com
567586786
Edit
Delete
49
xyz
9876543216
xyz@gmail.com
0
Edit
Delete
Type here to search
30°C
ENG
10:53
31-12-2021

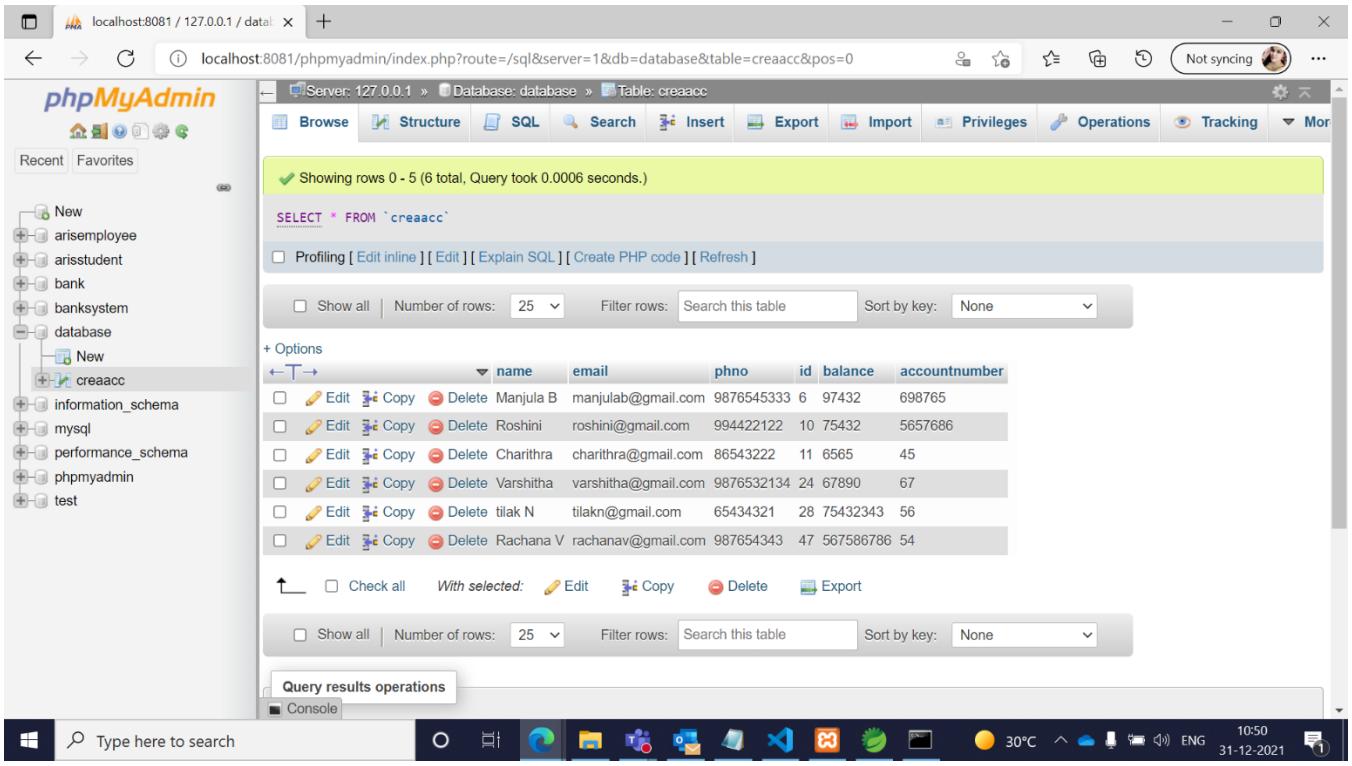

Banksys
localhost:4200/createacc
Not syncing
Bank Application
Create Account
User Details
Logout
About
localhost:4200 says
account created successfully
OK
Create Account
XYZ
xyz@gmail.com
9876543219
Create Account
Type here to search
30°C
ENG
10:51
31-12-2021


Banksys
localhost:4200/home
Not syncing
Bank Application
Home
Create Account
User Details
Logout
About
BANK
Banking Management System
localhost:4200/home
Type here to search
30°C
ENG
10:51
31-12-2021


Banksys
localhost:4200
Not syncing
Bank Application
localhost:4200 says
login successfull
OK
Sign In
admin
login
Type here to search
30°C
ENG
10:51
31-12-2021


Banksys
localhost:4200
Not syncing
Bank Application
localhost:4200 says
Please enter correct username and password.
OK
Sign In
admin
••••
login
Type here to search
30°C
ENG
10:50
31-12-2021


Banksys
localhost:4200
Not syncing
Bank Application
Sign In
Email / Username
Password
login
Type here to search
30°C
ENG
10:50
31-12-2021


localhost:8081 / 127.0.0.1 / data
localhost:8081/phpmyadmin/index.php?route=/sql&server=1&db=database&table=creaacc&pos=0
Not syncing
phpMyAdmin
Recent Favorites
New
arisemployee
arisstudent
bank
banksystem
database
New
creaacc
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: database » Table: creaacc
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking
Showing rows 0 - 5 (6 total, Query took 0.0006 seconds.)
SELECT * FROM `creaacc`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
name email phno id balance accountnumber
Edit Copy Delete Manjula B manjulab@gmail.com 9876545333 6 97432 698765
Edit Copy Delete Roshini roshini@gmail.com 994422122 10 75432 5657686
Edit Copy Delete Charithra charithra@gmail.com 86543222 11 6565 45
Edit Copy Delete Varshitha varshitha@gmail.com 9876532134 24 67890 67
Edit Copy Delete tilak N tilakn@gmail.com 65434321 28 75432343 56
Edit Copy Delete Rachana V rachanav@gmail.com 987654343 47 567586786 54
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Console
Type here to search
30°C
ENG
10:50
31-12-2021

6. Conclusion:

In our project, Bank Management System we have stored all the information about the user and provided option to create, view, modify and delete user details.

We had considered the most important requirements only, many more features and details can be added to our project in order to obtain even more user friendly applications.

7. Bibliography:

1. **https://www.tutorialspoint.com/java/java_overview.htm**

2. **https://javasterling.com/spring-boot/sts-ide/**

3. **https://spring.io/projects/spring-boot**