***Batch-2***

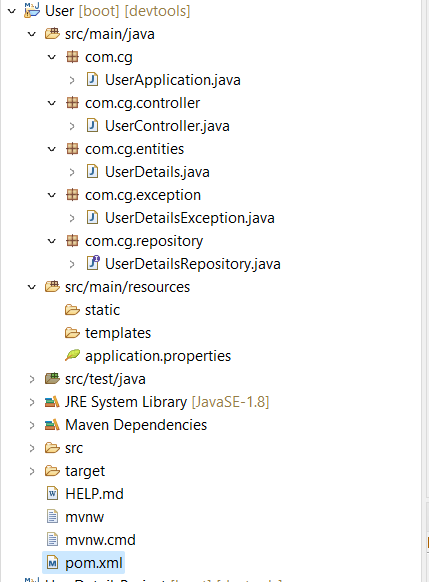
***SPRINT-2 (***Project name: User Detail)

*Batch Member:*

1. *Medapati AppaReddy(3)*
2. *Venkata janaki durgaprasad peruri(42)*
3. *Harika chalapathi(14)*
4. *kiran matta(19)*
5. *Gollapalli Usha Yasaswi(12)*
6. *Venkata Chaitanya kapa(41)*

**BACKEND(eclipse IDE)**

Eclipse directory structure:

****

**User/Src/main/java/com.cg/UserApplication.java:**

package com.cg;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class UserApplication {

public static void main(String[] args) {

SpringApplication.run(UserApplication.class, args);

}

}

**User/Src/main/java/com.cg.entities/UserDetails.java:**

package com.cg.entities;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

@Table(name = "User\_Datails")

public class UserDetails {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int id;

private String name;

private String mobile;

private String email\_id;

private String address;

private String gender;

private LocalDate date\_of\_Birth;

}

**User/Src/main/java/com.cg.repository/UserDetailsRepository.java:**

package com.cg.repository;

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

import org.springframework.stereotype.Repository;

import com.cg.entities.UserDetails;

@Repository

public interface UserDetailsRepository extends JpaRepository<UserDetails, Integer> {

}

**User/Src/main/java/com.cg.controller/UserController.java:**

package com.cg.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.transaction.annotation.Transactional;

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 com.cg.entities.UserDetails;

import com.cg.exception.UserDetailsException;

import com.cg.repository.UserDetailsRepository;

@CrossOrigin(origins = "\*")

@RestController

@RequestMapping(value ="/api/v1")

public class UserController {

@Autowired

private UserDetailsRepository Repo;

// get all students

@Transactional

@GetMapping("/user")

public List<UserDetails> getAllUserDetils(){

return Repo.findAll();

}

// create student rest api

@Transactional

@PostMapping("/user")

public UserDetails createUserdetails(@RequestBody UserDetails user) {

return Repo.save(user);

}

// get student by id rest api

@Transactional

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

public <X> ResponseEntity<UserDetails> getUserDetailsById(@PathVariable int id) {

UserDetails user = Repo.findById(id)

.orElseThrow(() -> new UserDetailsException ("Employee not exist with id :" + id));

return ResponseEntity.ok(user);

}

// updatestudent rest api

@Transactional

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

public ResponseEntity<UserDetails> updateUserDetails(@PathVariable int id, @RequestBody UserDetails userdetails){

UserDetails user = Repo.findById(id)

.orElseThrow(() -> new UserDetailsException ("Employee not exist with id :" + id));

user.setName(userdetails.getName());

user.setDate\_of\_Birth(userdetails.getDate\_of\_Birth());

user.setGender(userdetails.getGender());

user.setMobile(userdetails.getMobile());

user.setAddress(userdetails.getAddress());

com.cg.entities.UserDetails updatedUserDetails = Repo.save(user);

return ResponseEntity.ok(updatedUserDetails);

}

// delete student rest api

@Transactional

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

public ResponseEntity<HttpStatus> deleteUserDetails(@PathVariable int id){

UserDetails user = Repo.findById(id)

.orElseThrow(() -> new UserDetailsException ("Employee not exist with id :" + id));

Repo.delete(user);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

**User/Src/main/java/com.cg.exception/UserDetailsException.java:**

package com.cg.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public class UserDetailsException extends RuntimeException{

private static final long serialVersionUID = 1L;

public UserDetailsException(String message) {

super(message);

}

}

**User/Src/main/resource/application.properties:**

spring.datasource.url=jdbc:postgresql://localhost:5432/newuser

spring.datasource.username=postgres

spring.datasource.password=Asrithreddy

spring.jpa.hibernate.ddl-auto=update

spring.jpa.generate-ddl=true

**User/Target/pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.6.7</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cg</groupId>

<artifactId>User</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>User</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

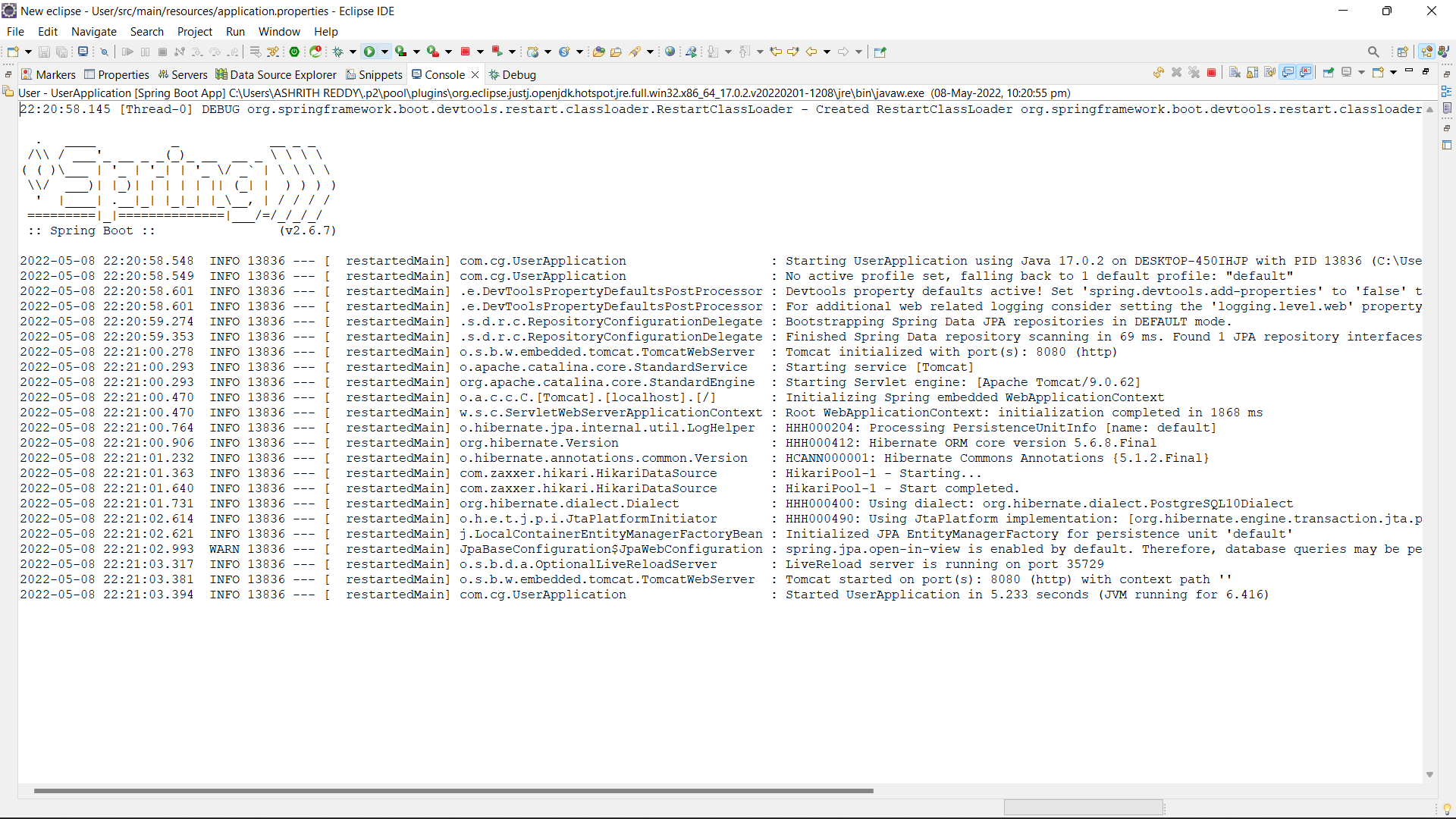
</plugin>

</plugins>

</build>

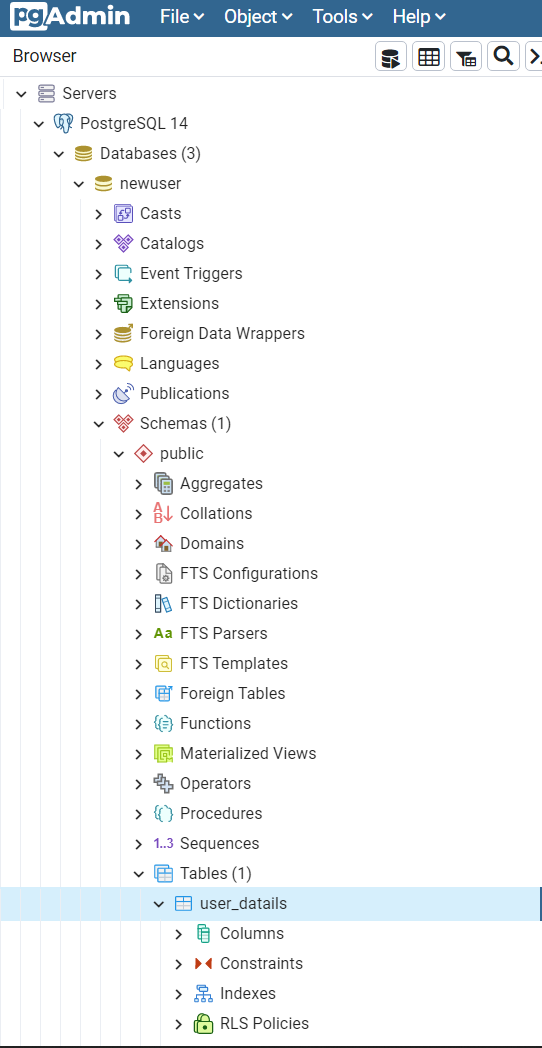
</project>

**CONSOLE:**

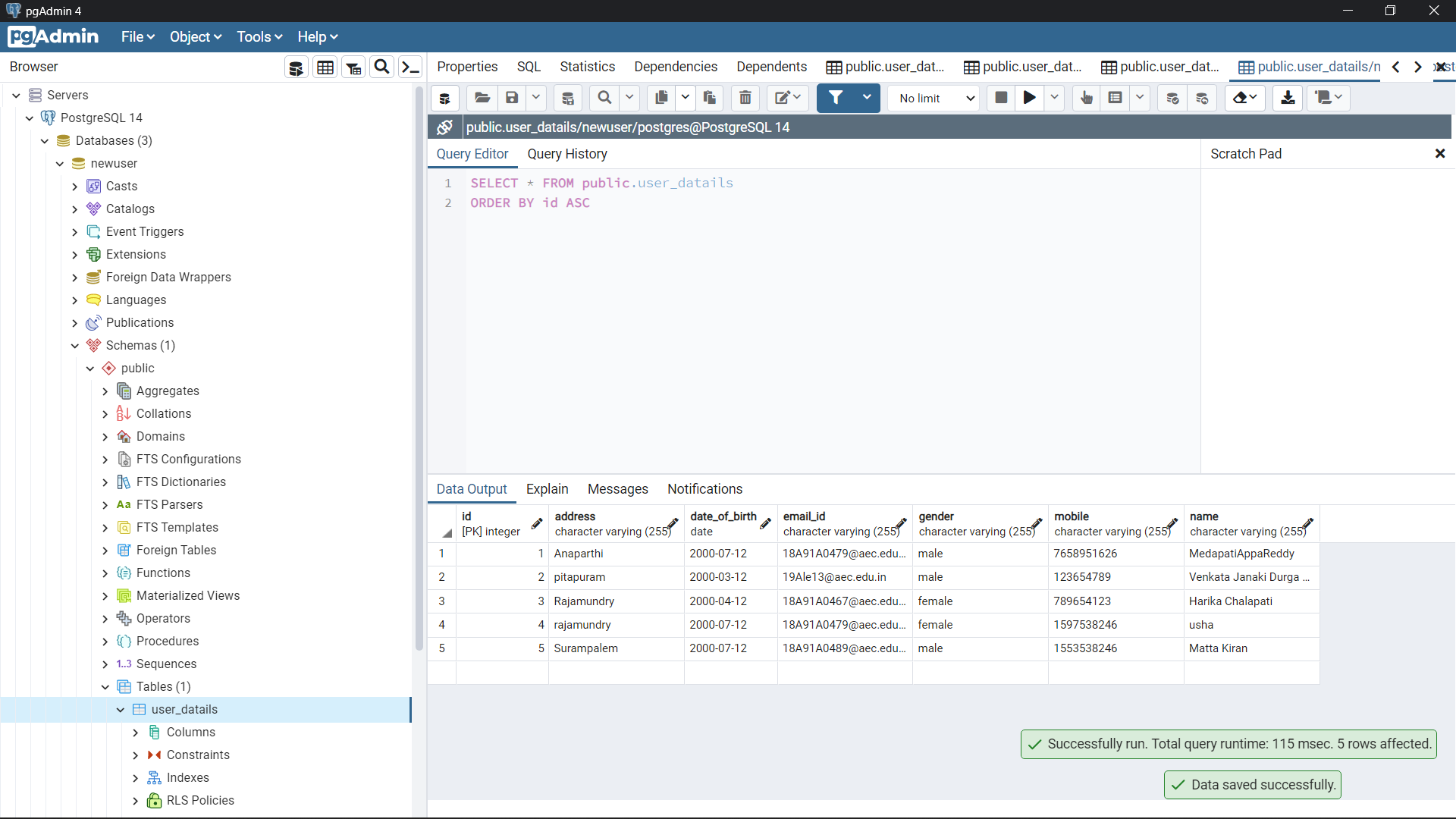
****

**DATABASE(PostgreSQL)**

Directory structure:

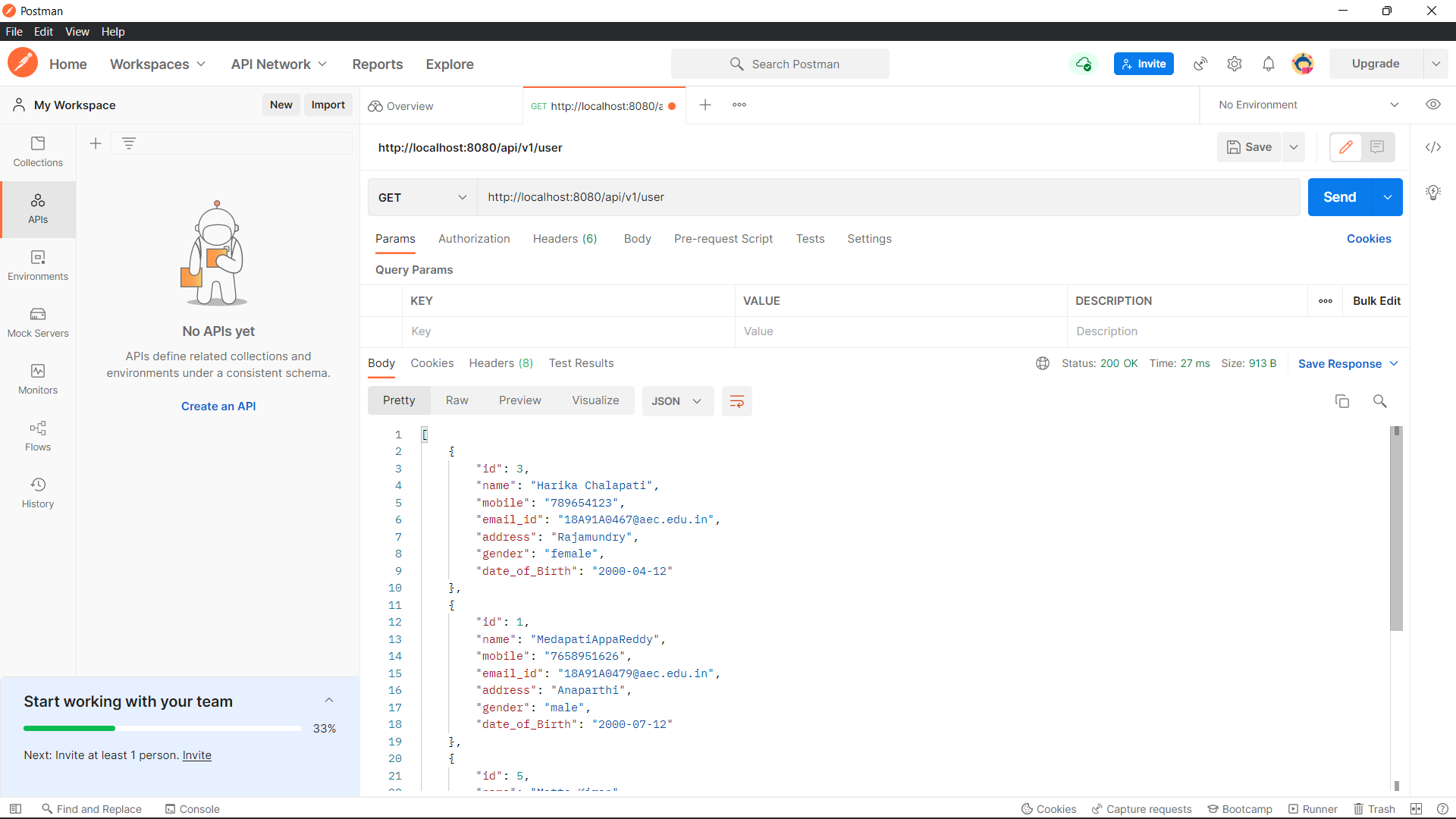
****

**DataBase Output:**

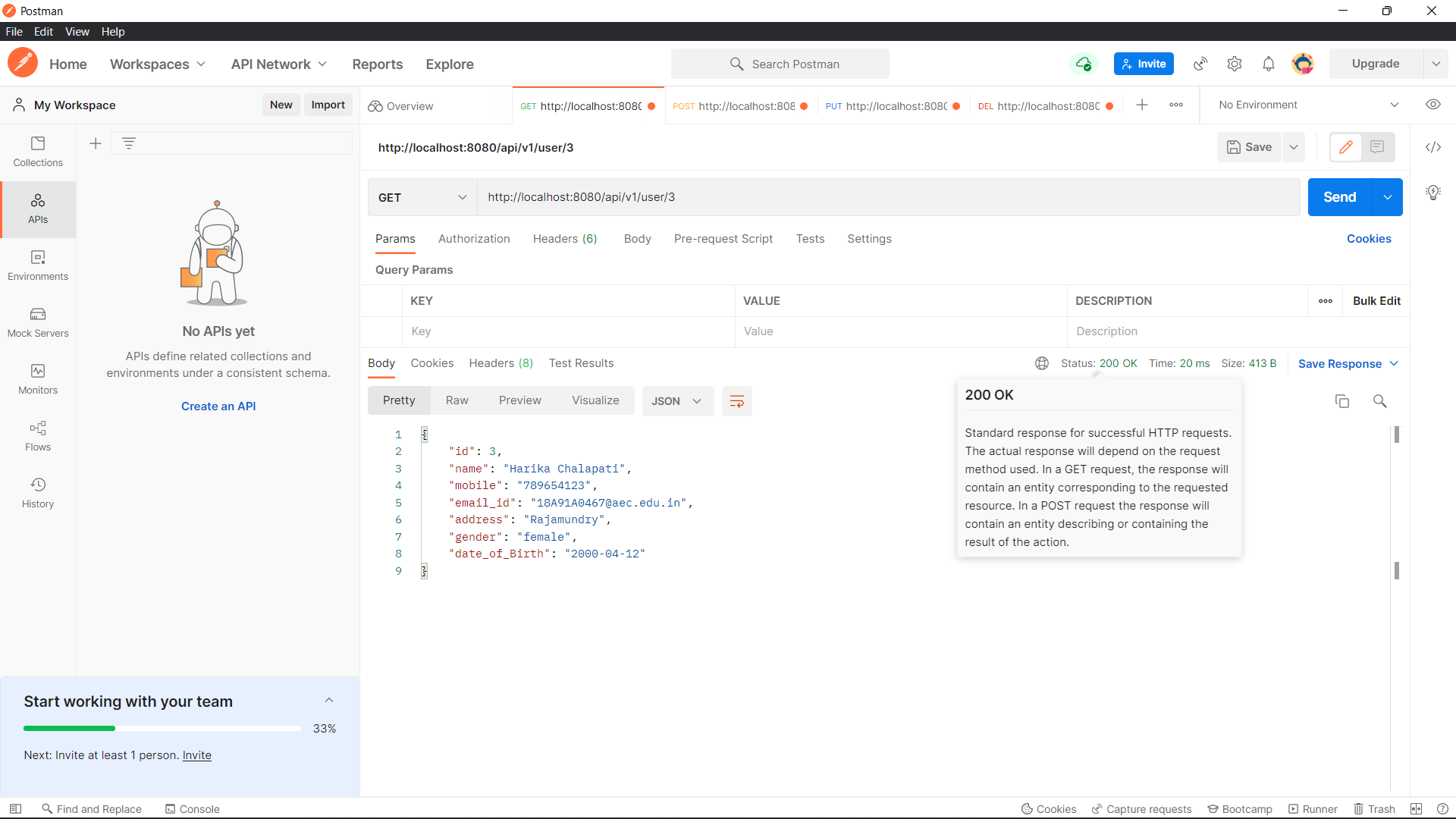
****

**POSTMAN**

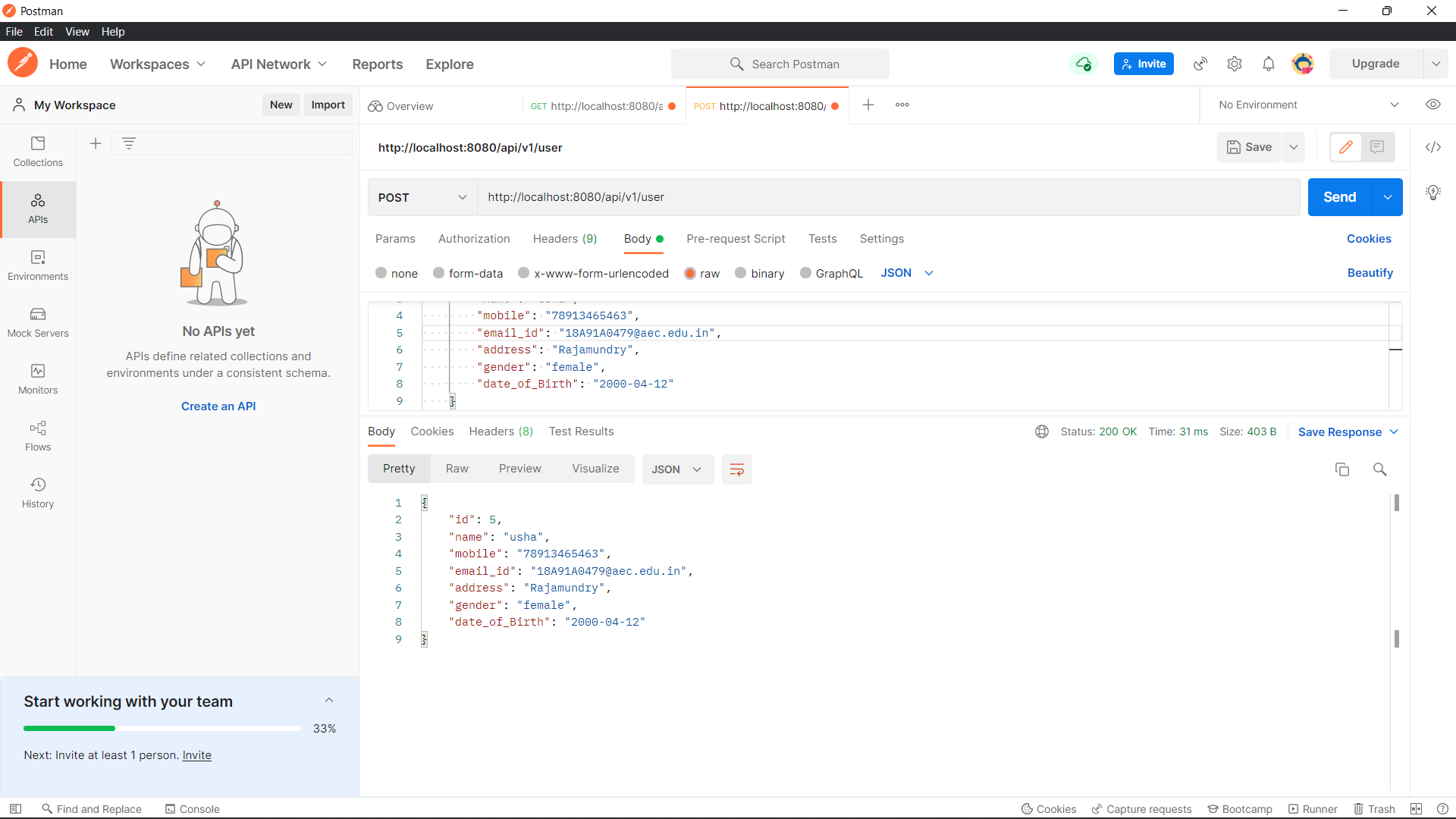
**GETUser:**

****

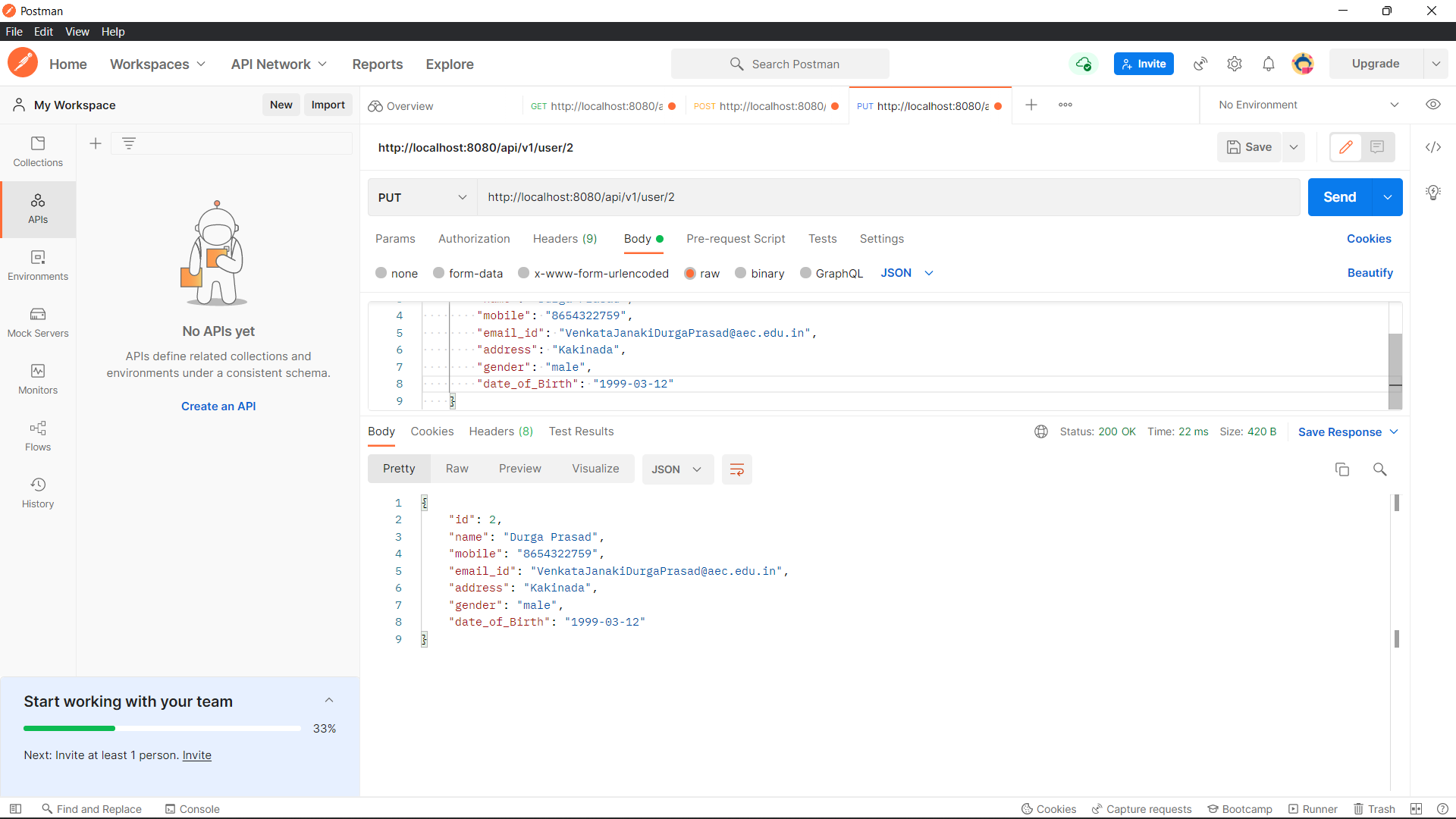
**GET User(Get user by ID):**

****

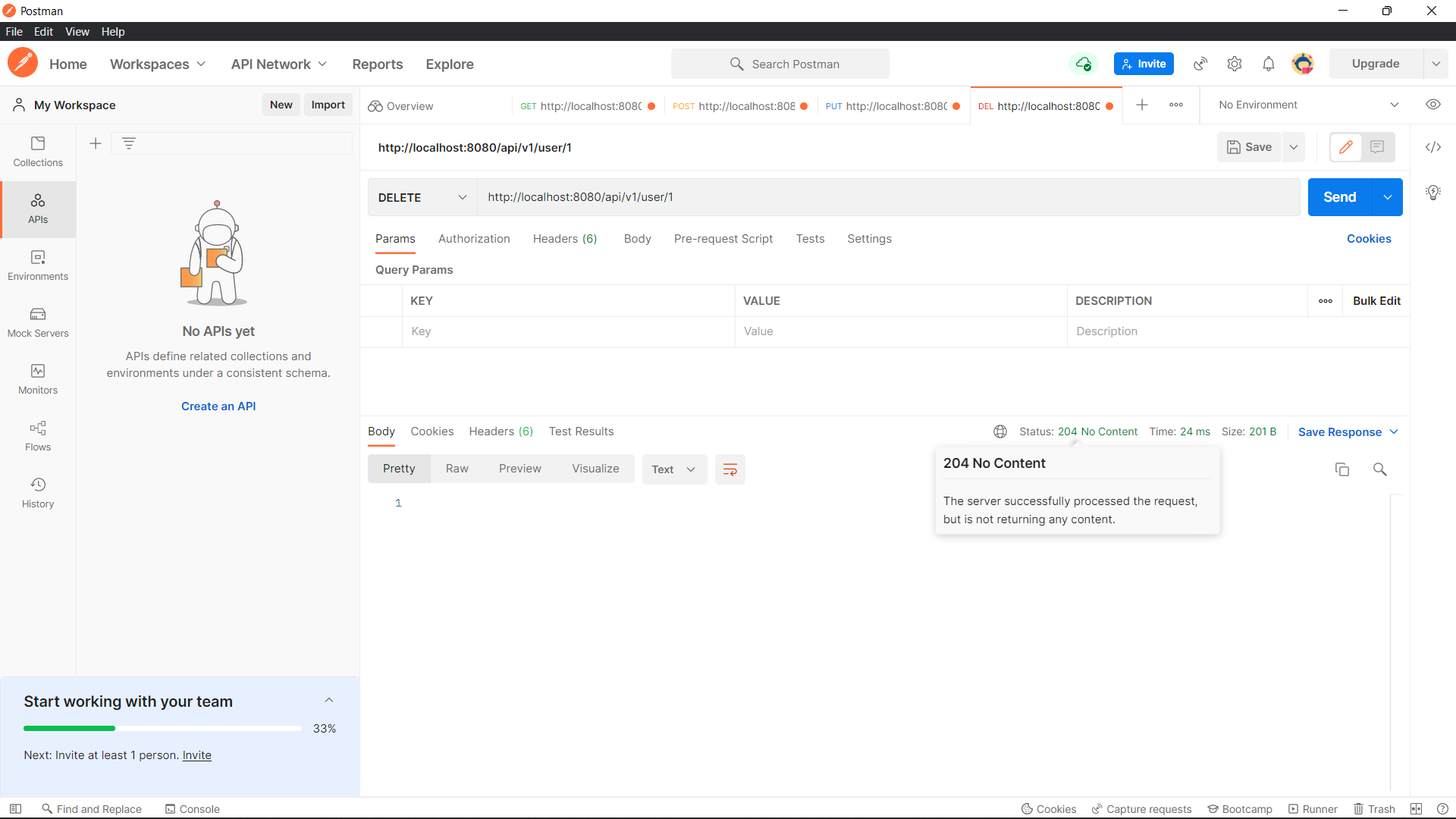
**POSTUser(add user):**

****

**PUTUser(update User):**

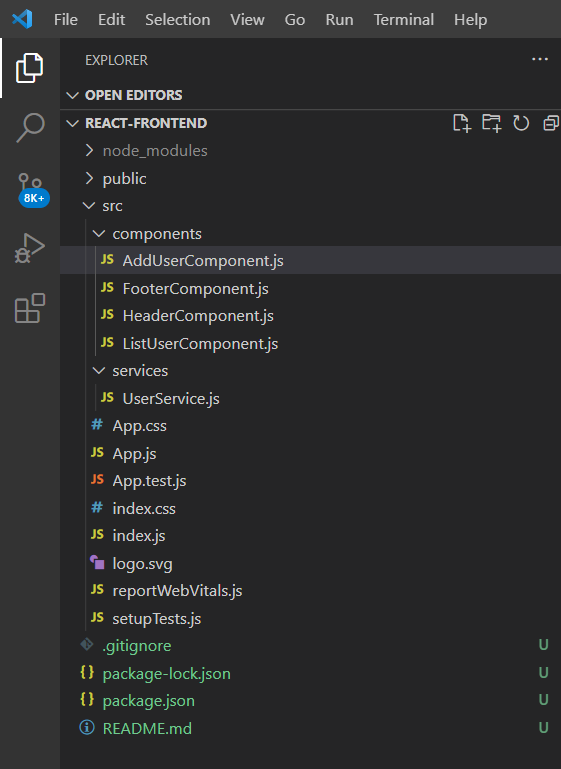
****

**DELETEUser(deleting user):**

****

**FRONTEND(react js)**

Directory structure:



**react-frontend/src/index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import 'bootstrap/dist/css/bootstrap.min.css'

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**react-frontend/src/components/ListUserComponents:**

import React,{useState,useEffect} from 'react'

import { Link } from 'react-router-dom'

import UserService from '../services/UserService'

const ListUserComponent = () => {

    const [users, setUsers] = useState([])

    useEffect(() => {

        getAllUser();

    }, [])

    const getAllUser =() => {

        UserService.getAllUser().then((Response) =>{

            setUsers(Response.data)

            console.log(Response.data)

        }).catch(error =>{

            console.log(error);

        })

    }

    const deletes = (userId) =>{

        UserService.deleteUser(userId).then((Response) => {

            getAllUser();

        }).catch(error =>{

            console.log(error);

        })

    }

  return (

    <div className='container'>

    <h2 className='text-center'>List User</h2>

    <Link to ="/add-user" className='btn btn-primary mb-2'>Add user</Link>

    <table className='table table-bordered table-striped'>

    <thead>

        <th>User Id</th>

        <th>Name</th>

        <th>mobile Number</th>

        <th>email\_id</th>

        <th>address</th>

        <th>gender</th>

        <th>date\_of\_Birth</th>

        <th>Actions</th>

    </thead>

    <tbody>

        {

            users.map(

                user =>

                <tr key={user.id}>

                    <td>{user.id}</td>

                    <td>{user.name}</td>

                    <td>{user.mobile}</td>

                    <td>{user.email\_id}</td>

                    <td>{user.address}</td>

                    <td>{user.gender}</td>

                    <td>{user.date\_of\_Birth}</td>

                    <td>

                        <Link className='btn btn-info' to={`/edit-user/${user.id}`}>Update</Link>

                        <button className='btn btn-danger' onClick={() => deletes(user.id)} style={{marginLeft: "10px"}} >Delete</button>

                    </td>

                </tr>

            )

        }

    </tbody>

    </table>

    </div>

  )

}

export default ListUserComponent

**react-frontend/src/components/AddUserComponents:**

import React,{useState , useEffect } from 'react'

import {useNavigate , useParams} from 'react-router-dom';

import UserService from '../services/UserService'

const AddUserComponent = () => {

    const [name, setName] = useState([])

    const [mobile, setMobile] = useState([])

    const [email\_id, setEmail\_id] = useState([])

    const [address, setAddress] = useState([])

    const [gender, setGender] = useState([])

    const [date\_of\_Birth, setDate\_of\_Birth] = useState([])

    const navigate = useNavigate();

    const {id}=useParams();

    const saveOrUpdateUser =(e)=>{

        e.preventDefault();

        const user ={name,mobile,email\_id,address,gender,date\_of\_Birth}

        if(id){

            UserService.updateUser(id,user).then((Response) =>{

               // console.log(Response.data)

                navigate('/');

            }).catch(error => {

                console.log(error.response.data)

            })

        }else{

            UserService.createUser(user).then((Response) =>{

                console.log(Response.data)

                navigate('/');

        }).catch(error =>{

            console.log(error.response.data)

        })

    }

        }

    useEffect(() => {

        UserService.getUserById(id).then((Response) =>{

            setName(Response.data.name)

            setAddress(Response.data.address)

            setEmail\_id(Response.data.email\_id)

            setGender(Response.data.gender)

            setMobile(Response.data.mobile)

            setDate\_of\_Birth(Response.data.date\_of\_Birth)

            //set(Response.data.name)

        }).catch(error =>{

            console.log(error)

        })

    }, [])

    const title=() => {

        if(id){

            return <h2 className='text-center'>Update User</h2>

        } else{

            return <h2 className='text-center'>Add User</h2>

        }

    }

  return (

    <div>

    <br /><br/>

        <div className='container'>

            <div className='row'>

                <div className='card col-md6 offset-md-3 offset-md-3'>

                    {

                        title()

                    }

                    <div className='card-body'>

                        <form>

                            <div className='form-group mb-2'>

                                <label className='form-label'>Name :</label>

                                <input type={'text'} placeholder="enter name" name='name' className='form-control' value={name} onChange ={(e) => setName(e.target.value)} />

                            </div>

                            <div className='form-group mb-2'>

                                <label className='form-label'>mobile :</label>

                                <input type={'number'} placeholder="enter mobile number" name='mobile' className='form-control' value={mobile} onChange ={(e) => setMobile(e.target.value)} />

                            </div>

                            <div className='form-group mb-2'>

                                <label className='form-label'>email\_id :</label>

                                <input type={'email'} placeholder="enter Email\_id" name='email\_id' className='form-control' value={email\_id} onChange ={(e) => setEmail\_id(e.target.value)} />

                            </div>

                            <div className='form-group mb-2'>

                                <label className='form-label'>address :</label>

                                <input type={'text'} placeholder="enter Address" name='address' className='form-control' value={address} onChange ={(e) => setAddress(e.target.value)} />

                            </div>

                            <div className='form-group mb-2'>

                                <label className='form-label'>Gender :</label>

                                <input type={'text'} placeholder="enter gender" name='gender' className='form-control' value={gender} onChange ={(e) => setGender(e.target.value)} />

                            </div>

                            <div className='form-group mb-2'>

                                <label className='form-label'>DoB :</label>

                                <input type={'date'} placeholder="enter DoB" name='date\_of\_Birth' className='form-control' value={date\_of\_Birth} onChange ={(e) => setDate\_of\_Birth(e.target.value)} />

                            </div>

                            <button className='btn btn-success' onClick={(e) =>  saveOrUpdateUser(e)}> Submit</button>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

  )

}

export default AddUserComponent

**react-frontend/src/components/HeaderComponent:**

import React from 'react'

export const HeaderComponent = () => {

  return (

    <div>

        <header>

            <nav className='navbar navbar-expand-md navbar-dark bg-dark'>

                <div>

                    <h2 className='navbar-brand'>

                        User Details

                    </h2>

                </div>

            </nav>

        </header>

    </div>

  )

}

export default HeaderComponent

**react-frontend/src/components/FooterComponent:**

import React from 'react'

const FooterComponent = () => {

  return (

    <div>

        <footer className='footer'>

            <span className='text-muter'>Sprint-II(BATCH-2)</span>

        </footer>

    </div>

  )

}

export default FooterComponent

**react-frontend/src/services/UserService:**

import axios from 'axios'

const USER\_BASE\_REST\_API\_URL='http://localhost:8080/api/v1/user'

class UserService{

    getAllUser(){

        return axios.get(USER\_BASE\_REST\_API\_URL)

    }

    createUser(user){

        return axios.post(USER\_BASE\_REST\_API\_URL ,user)

    }

    getUserById(userId){

        return axios.get(USER\_BASE\_REST\_API\_URL + '/' + userId)

    }

    updateUser(userId,user){

        return axios.put(USER\_BASE\_REST\_API\_URL +'/' + userId, user)

    }

    deleteUser(userId){

        return axios.delete(USER\_BASE\_REST\_API\_URL +'/' + userId)

    }

}

export default new UserService();

**react-frontend/src/App.js:**

import './App.css';

import {BrowserRouter as Router,Route,Routes} from 'react-router-dom';

import FooterComponent from './components/FooterComponent';

import HeaderComponent from './components/HeaderComponent';

import ListUserComponent from './components/ListUserComponent';

import AddUserComponent from './components/AddUserComponent';

function App() {

  return (

    <div>

    <Router>

      <HeaderComponent />

      <div className='container'>

      <Routes>

        <Route exact path='/' element ={<ListUserComponent />}/>

        <Route path='/user' element ={<ListUserComponent />} />

        <Route path="/add-user" element ={<AddUserComponent />} />

        <Route path="/edit-user/:id" element ={<AddUserComponent />} />

      </Routes>

      </div>

      <FooterComponent />

      </Router>

    </div>

  );

}

export default App;

**react-frontend/package.json:**

{

  "name": "react-frontend",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@testing-library/jest-dom": "^5.16.4",

    "@testing-library/react": "^13.2.0",

    "@testing-library/user-event": "^13.5.0",

    "axios": "^0.27.2",

    "bootstrap": "^5.1.3",

    "react": "^18.1.0",

    "react-dom": "^18.1.0",

    "react-router-dom": "^6.3.0",

    "react-scripts": "5.0.1",

    "web-vitals": "^2.1.4"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

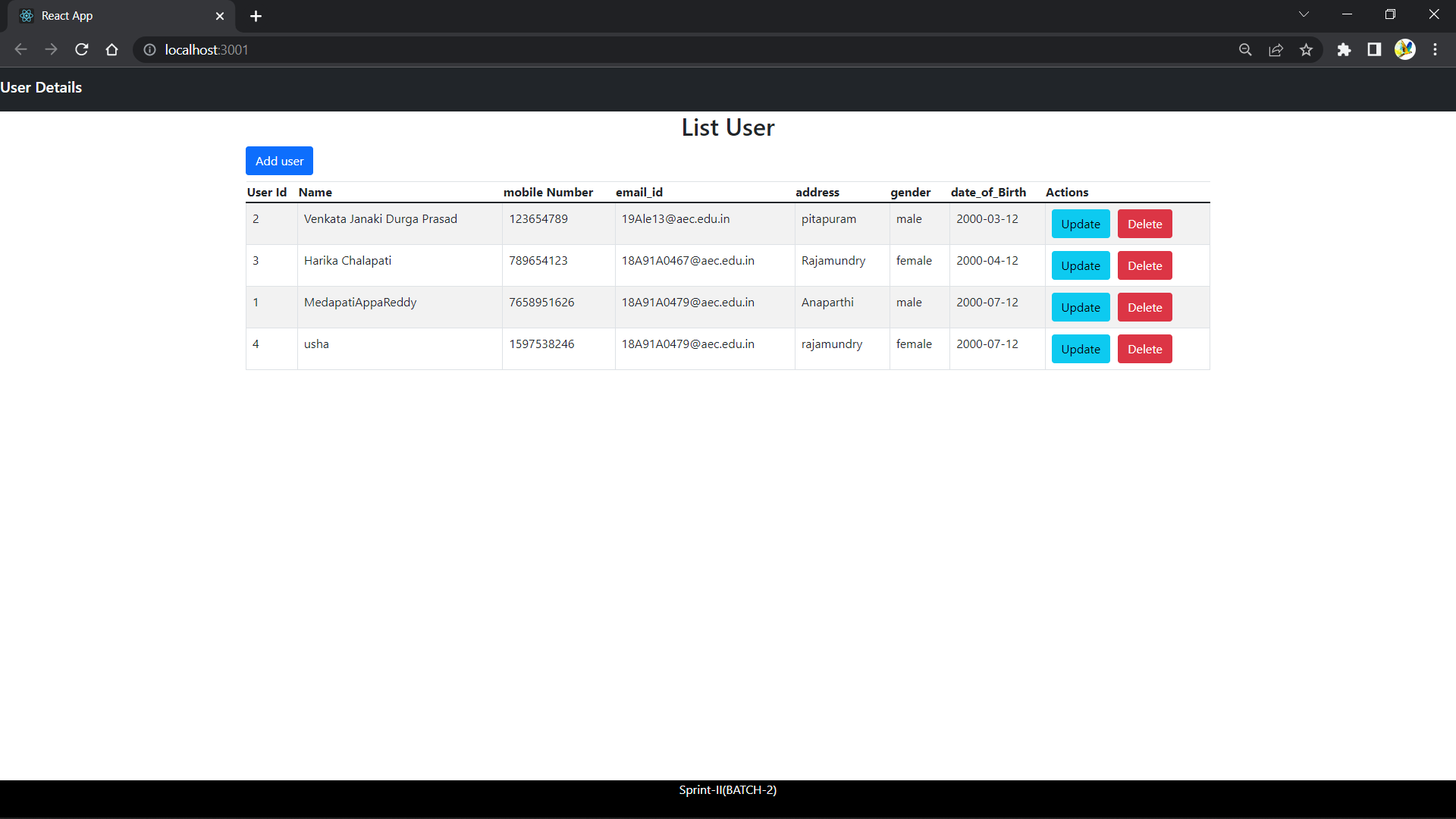
    ]

  }

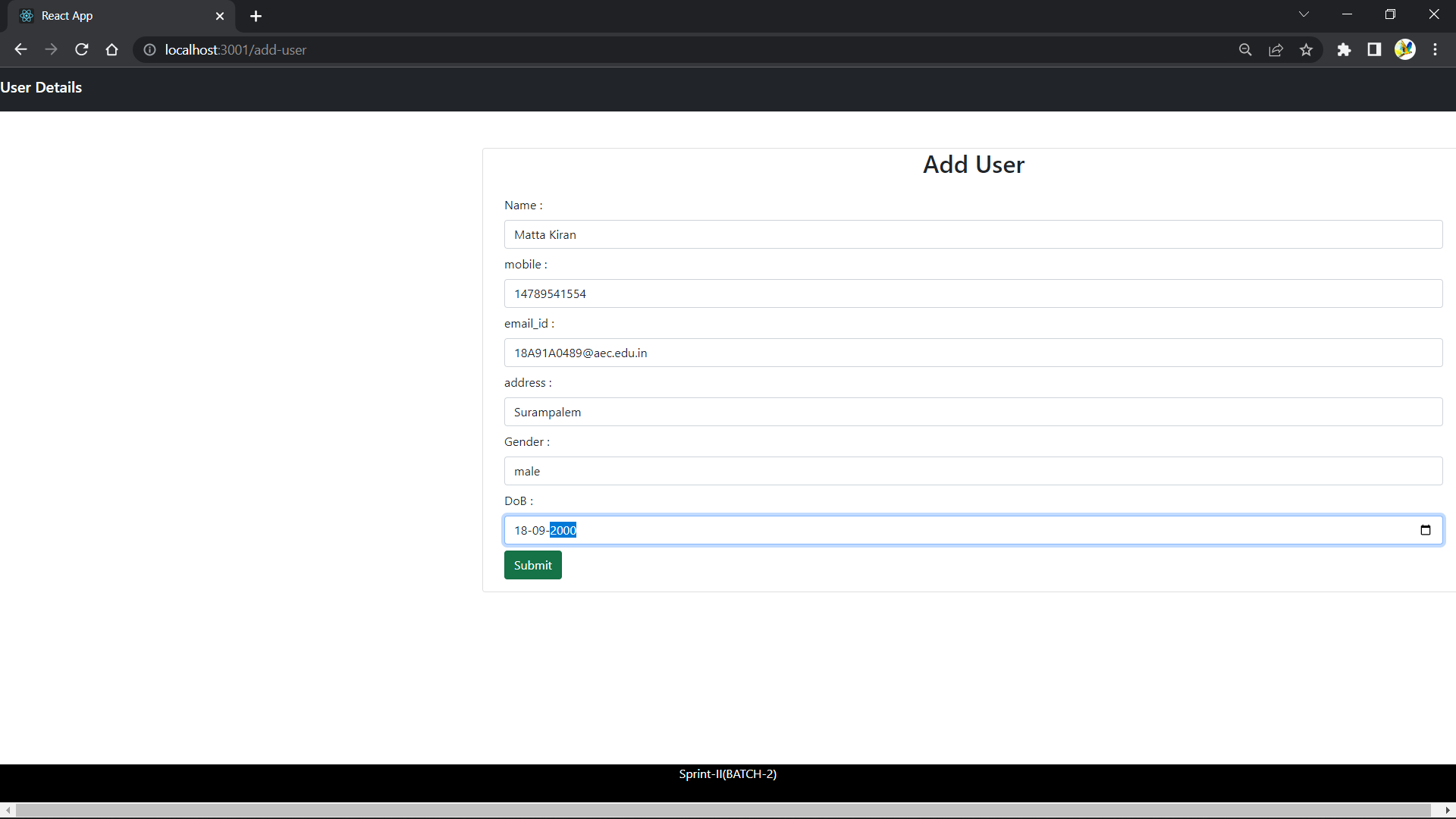
}

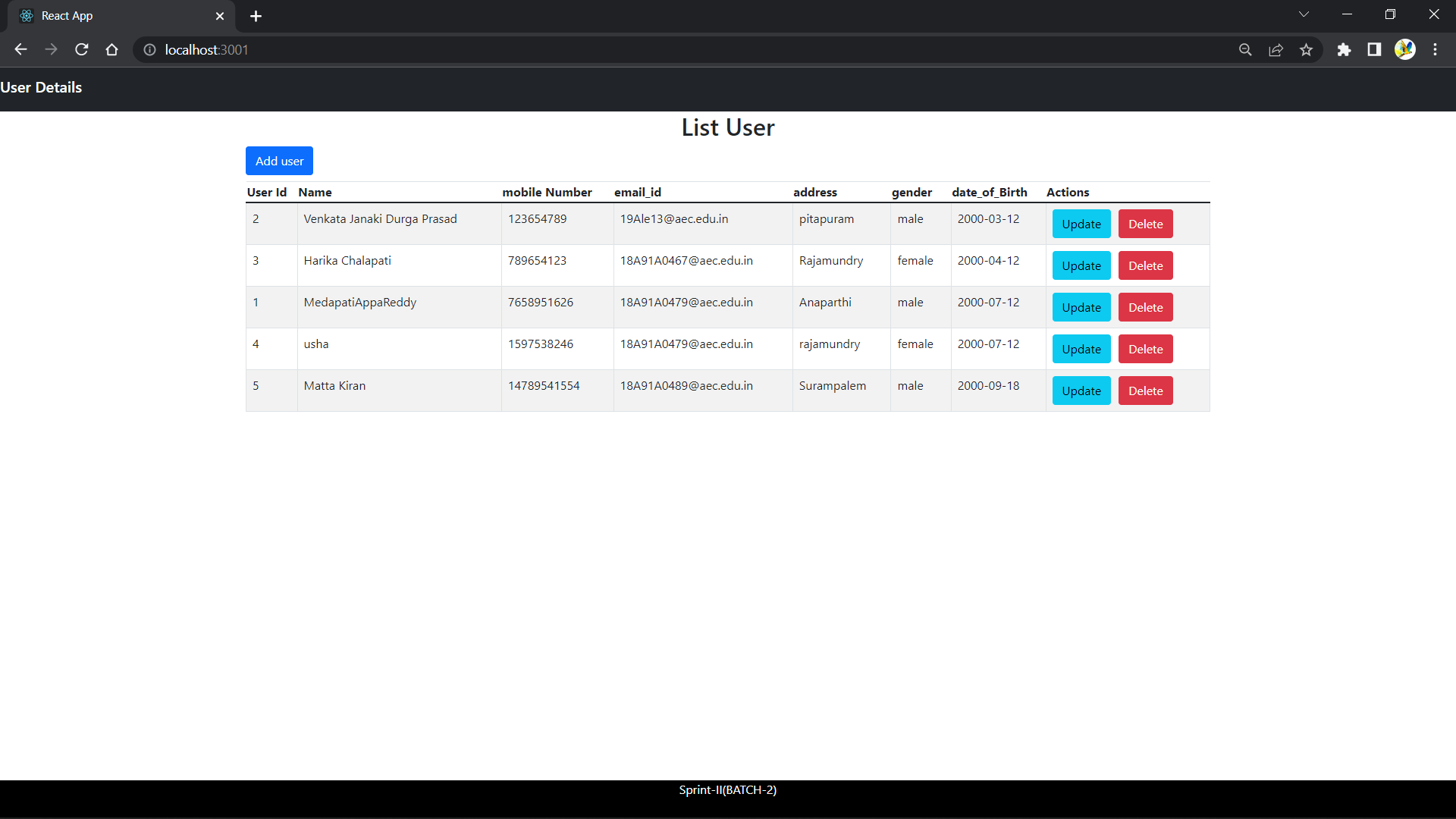
**OUTPUTS**

**List of Users:**

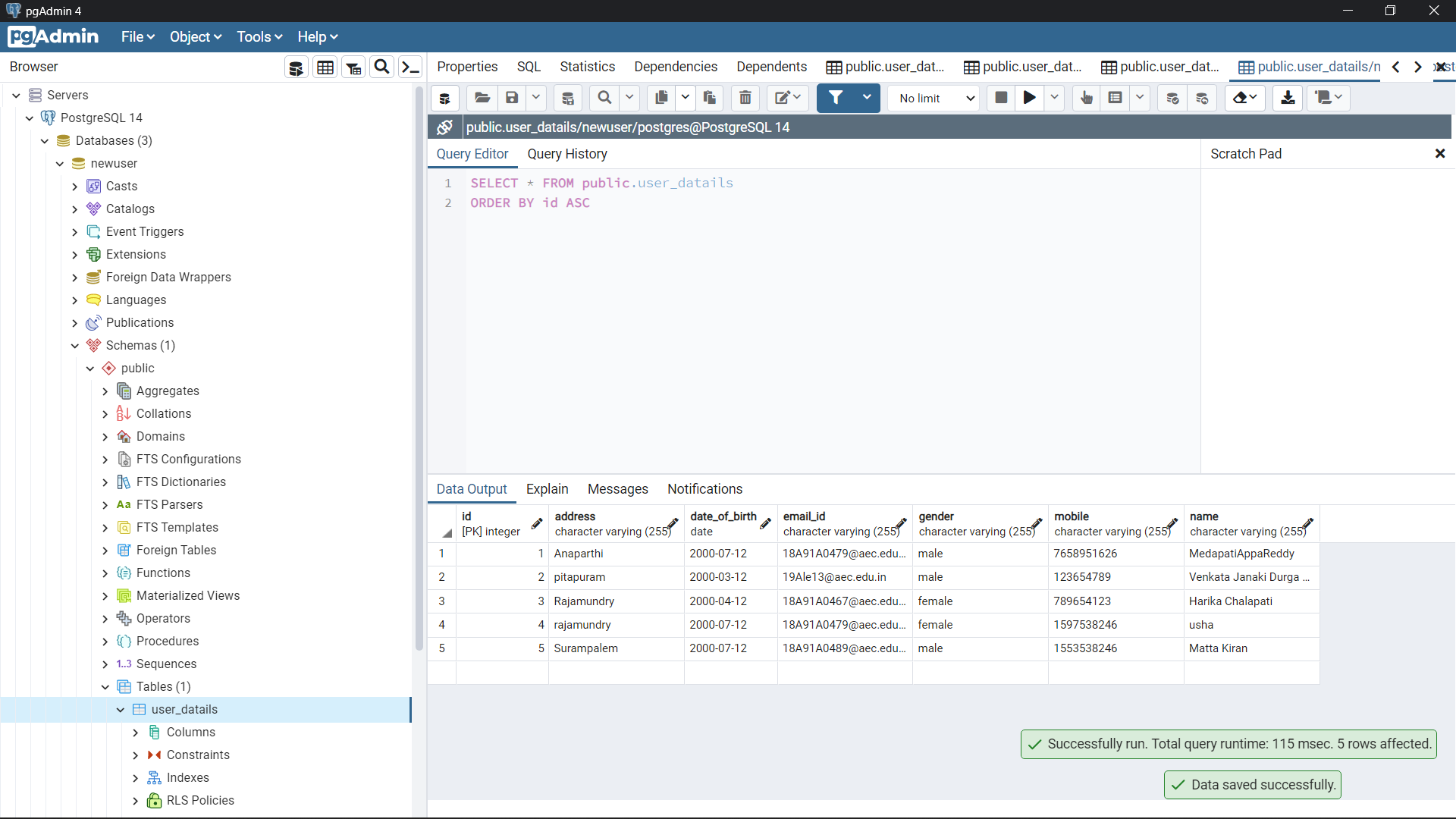
****

**Add Users:**

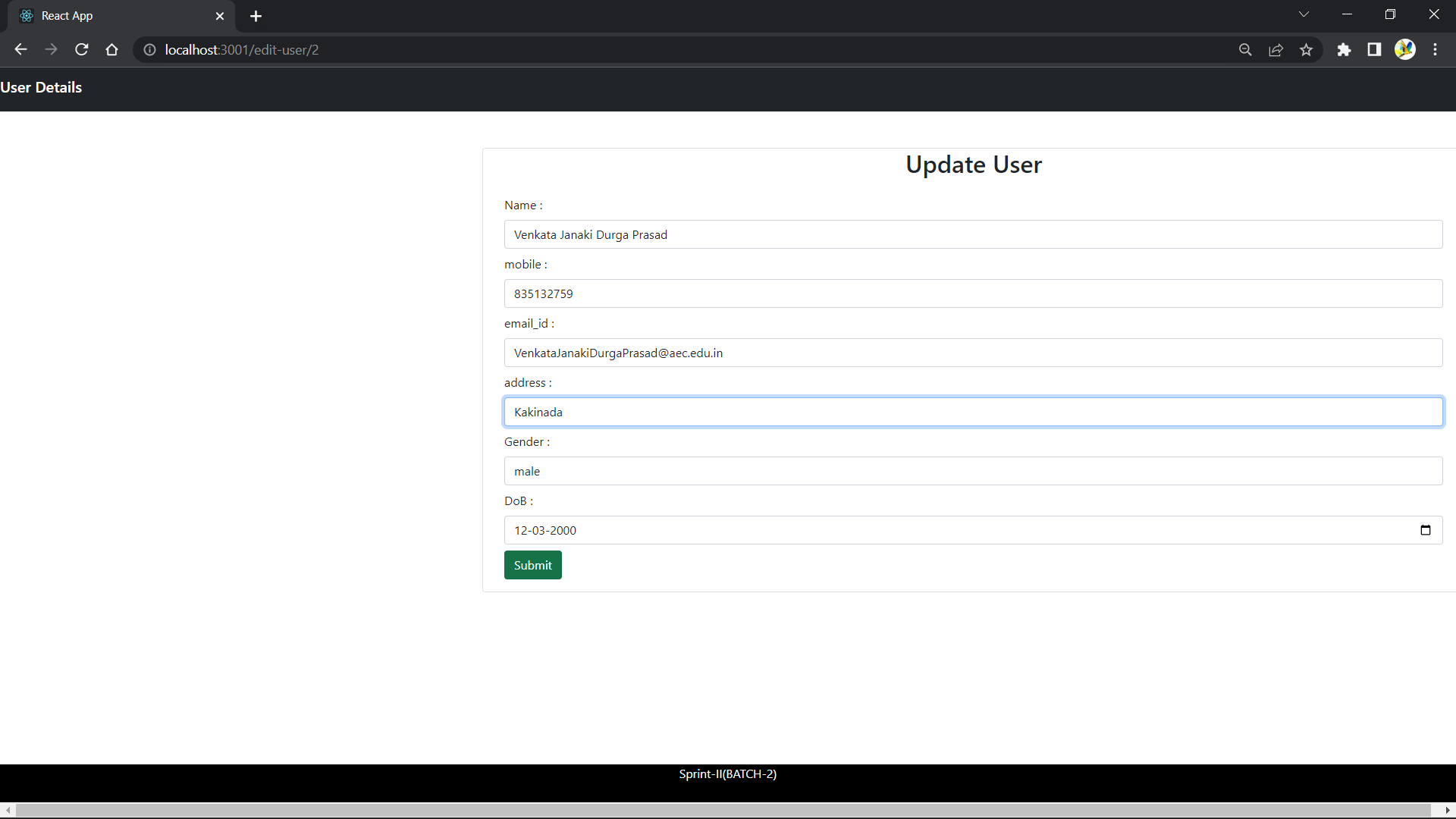
****

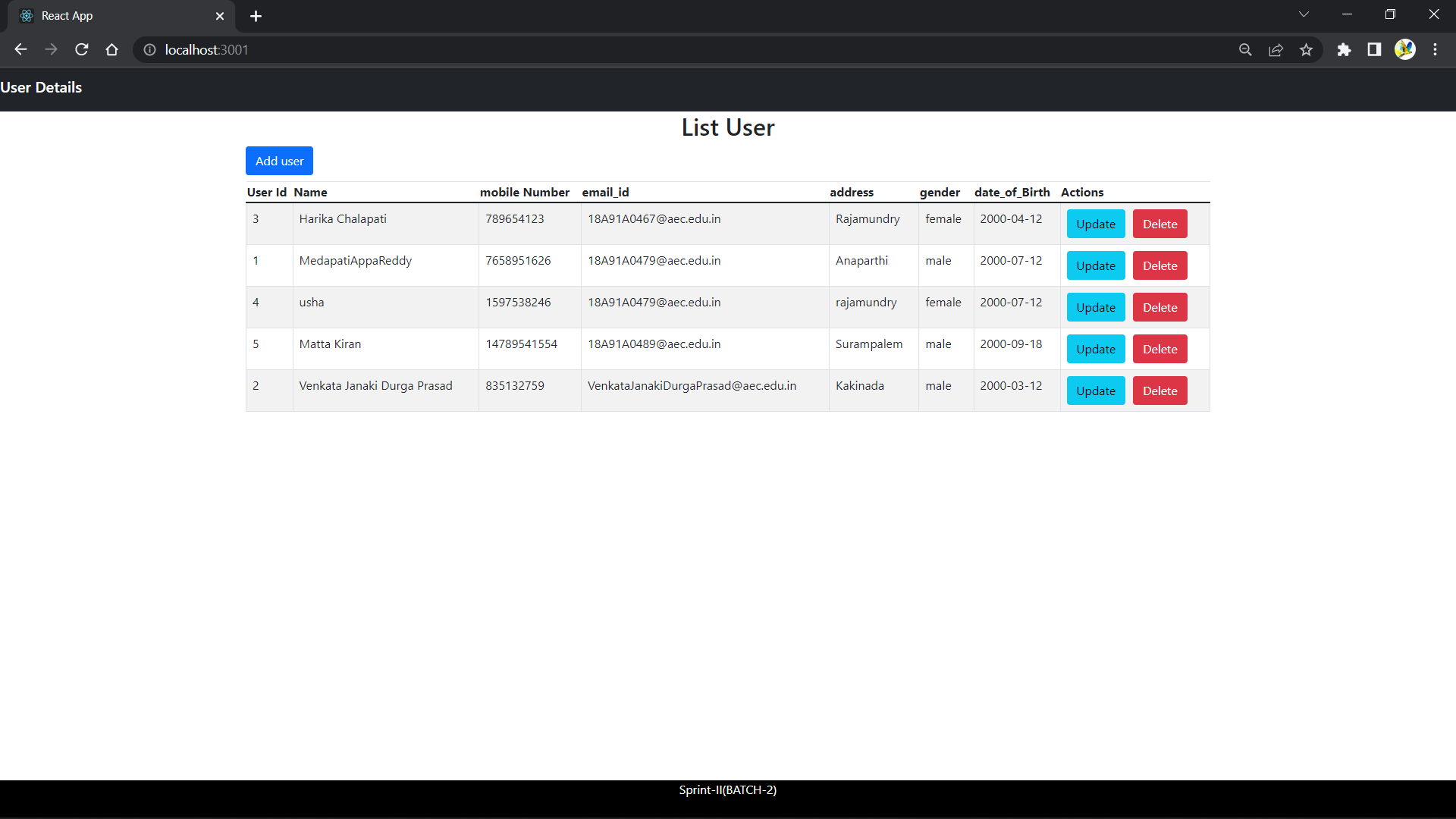
****

**DataBase Output:**

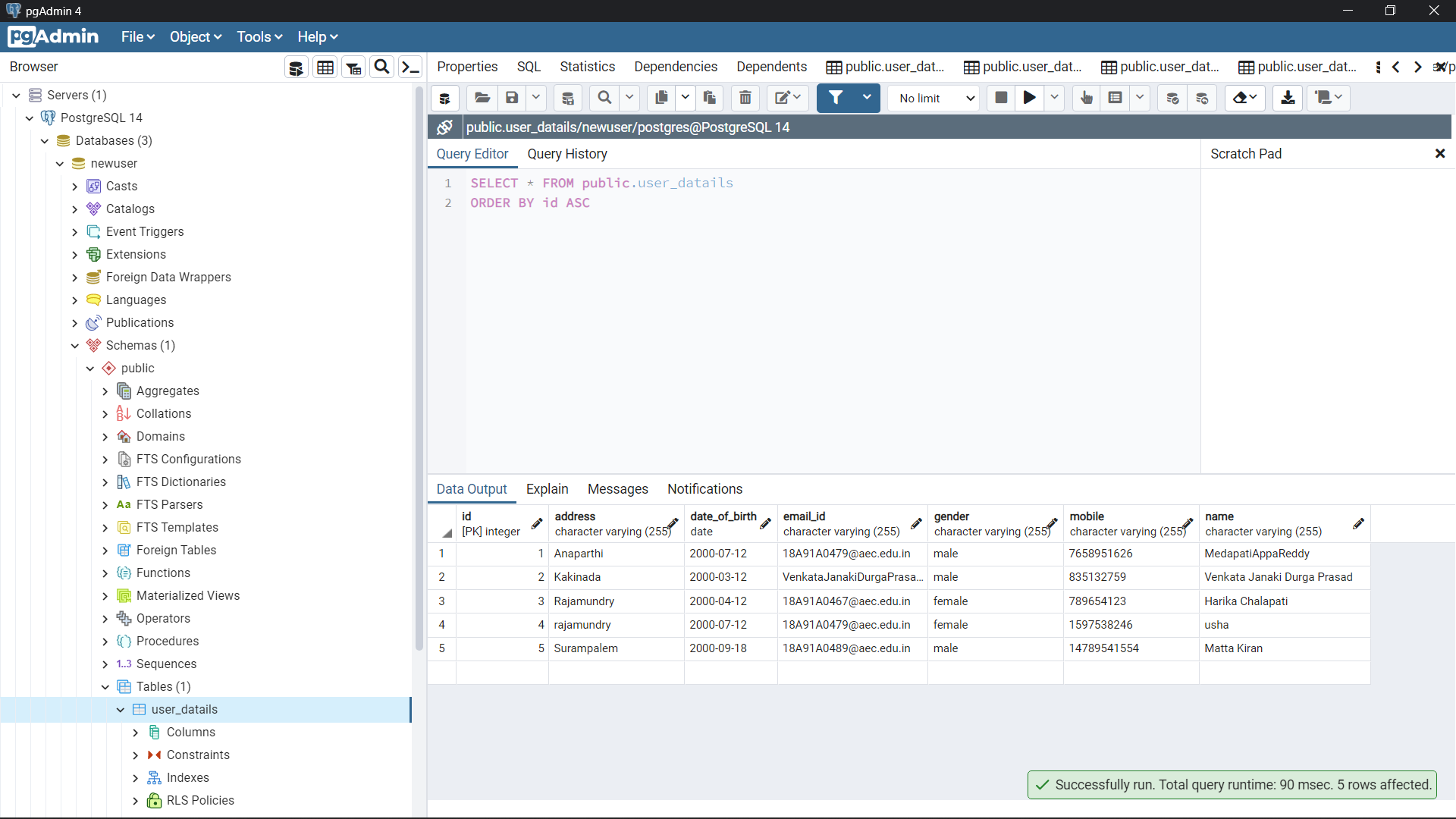
****

**Update User:**

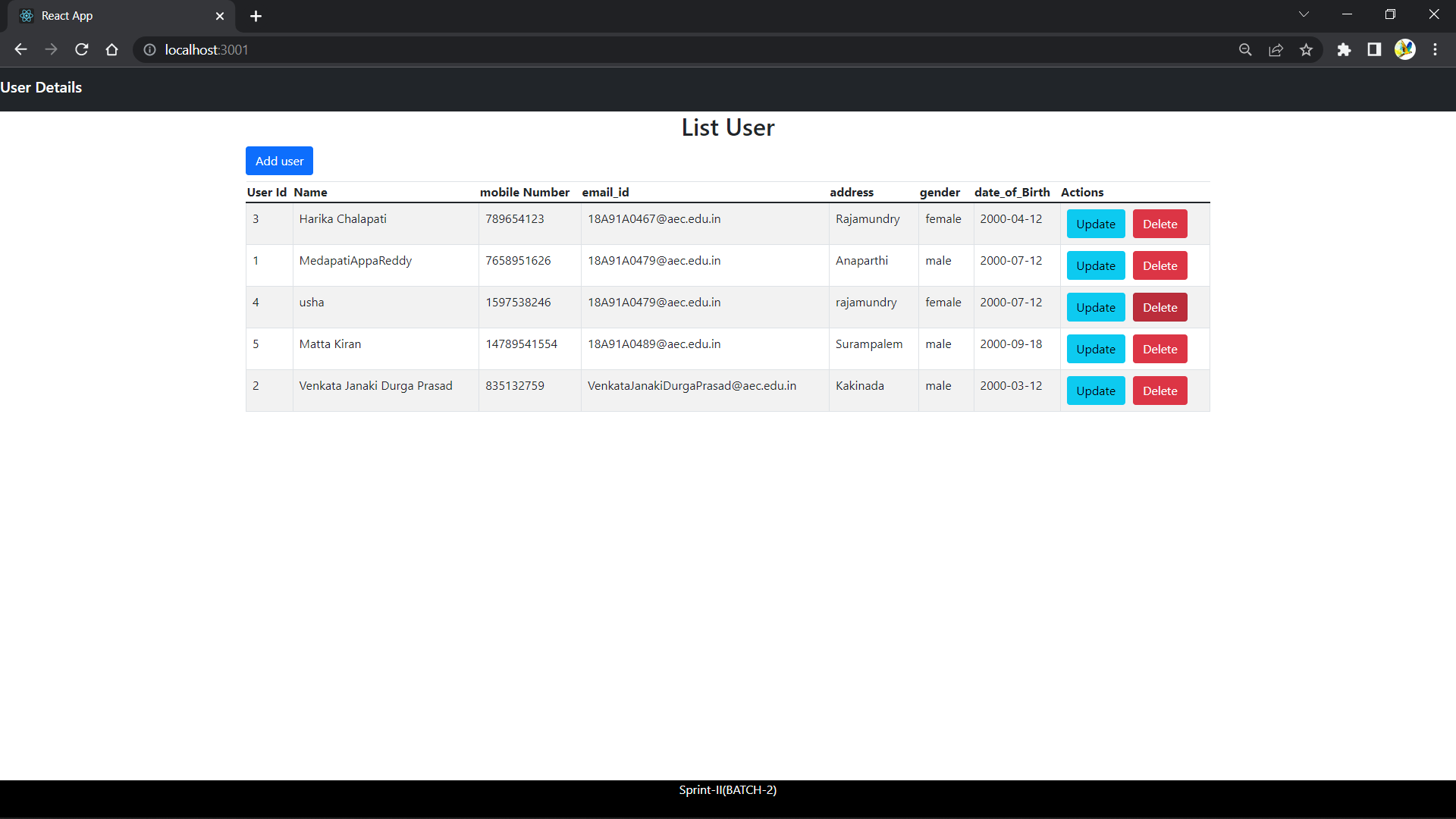
****

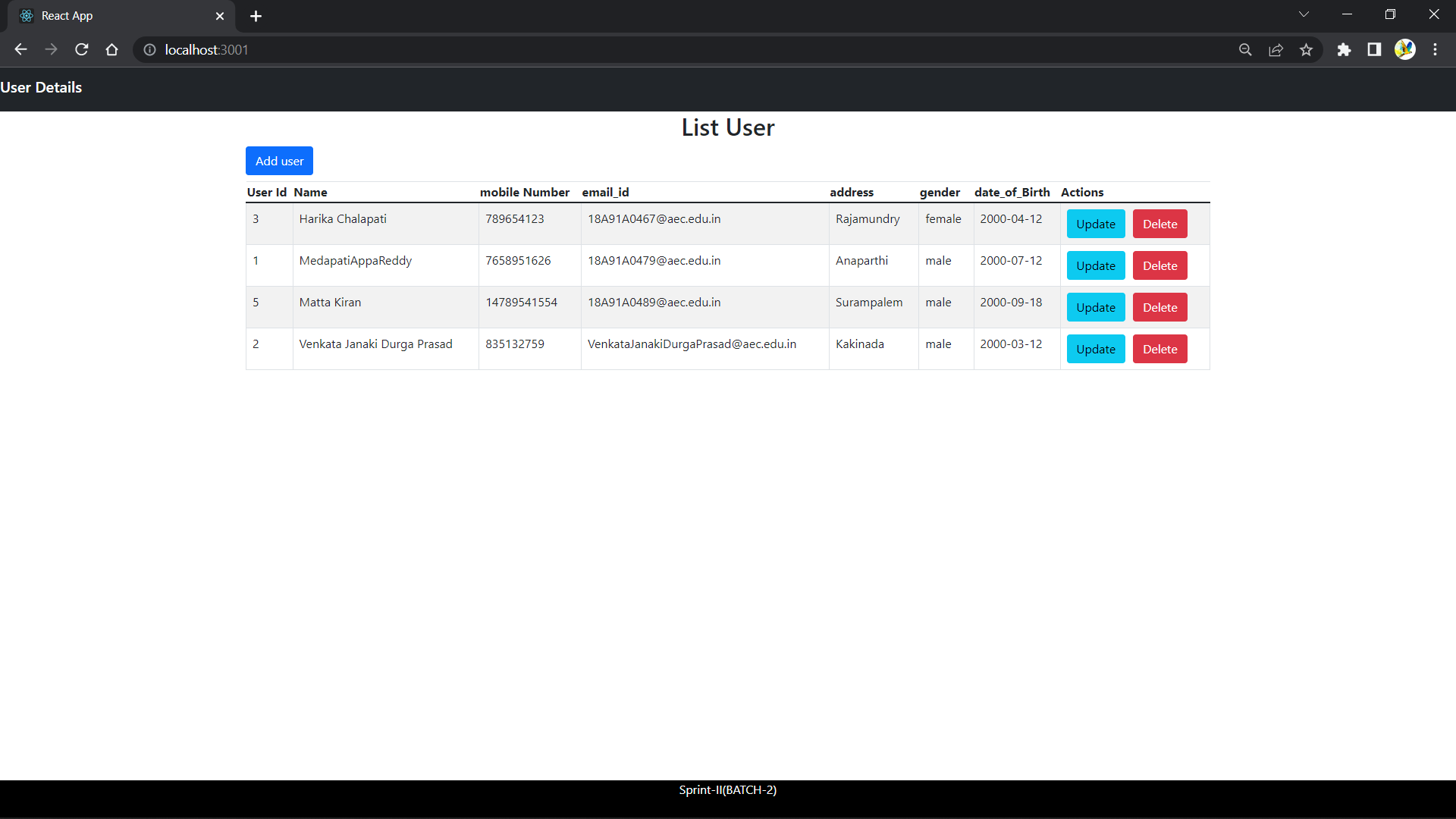
****

**Updated Database:**

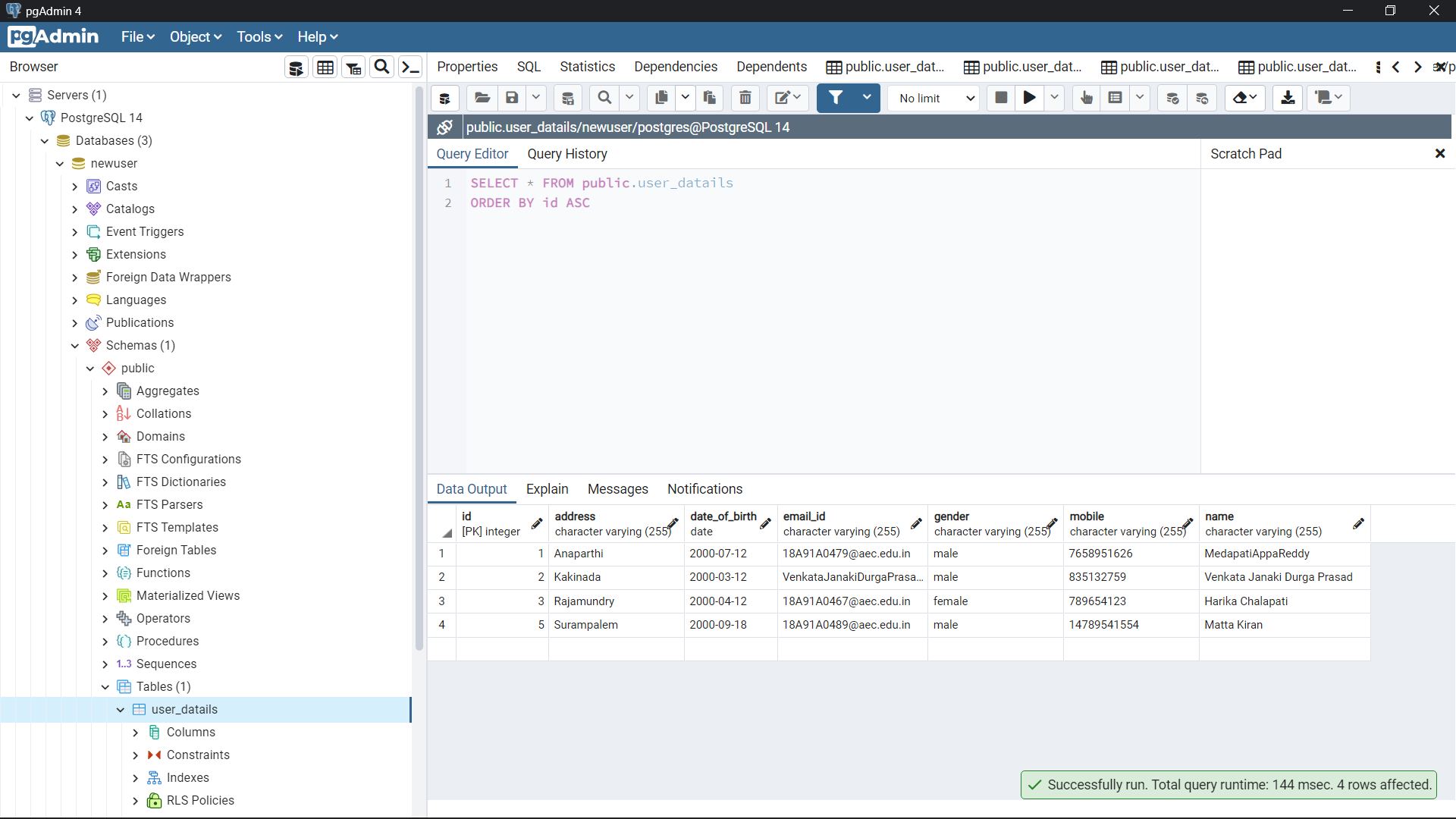
****

**Delete User:**

****

****

**DataBase Output:**

****