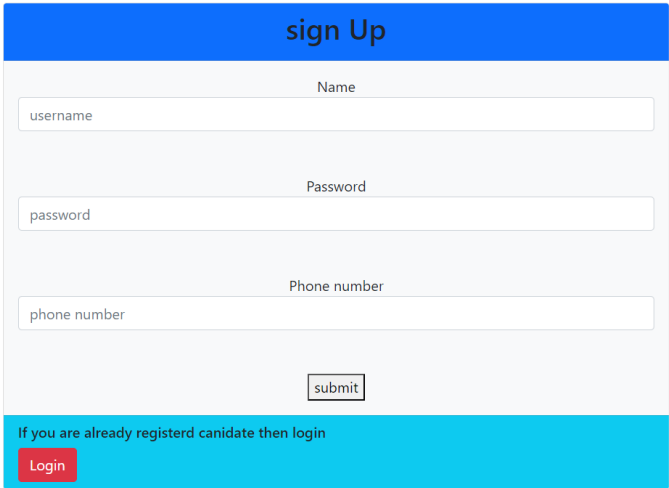


USER MANAGEMENT SYSTEM

First we will start with React.js ,By using this we can provide UI for user we need to create project by using command of npm create-react-app < App name>, Next we have to install the required dependencies like,

- 1.npm install react-router-dom
- 2.npm install -react-bootstrap5.14
- 3.npm install axios

After adding of dependencies I provided a nav bar with button of click here . After clicking of click here button we will get Registration form



The image shows a registration form titled "sign Up" in a blue header. Below the header, there are three input fields: "Name" (with placeholder "username"), "Password" (with placeholder "password"), and "Phone number" (with placeholder "phone number"). A "submit" button is located below the phone number field. At the bottom of the form, there is a light blue section with the text "If you are already registerd candidate then login" and a red "Login" button.

If you are a already registered candidate then we have go for directly for login if you are name and password are matched to you are registration details then we are directly enter to home page or if you are a new candidate then we have to go for registration after registration we are enter to home page.

NOTE : please provide the all inputs are in lower case

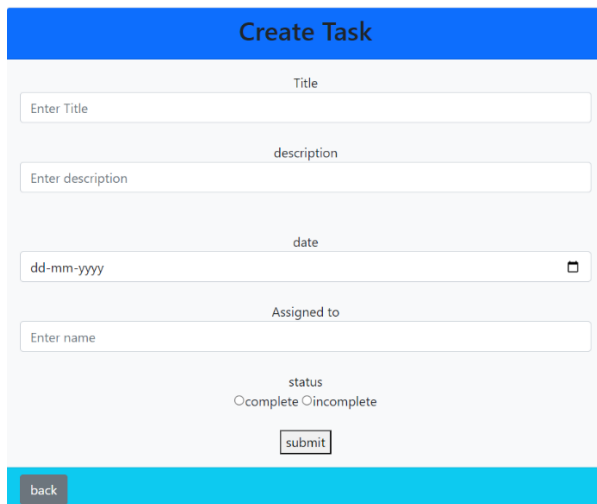
welcome user please go through the given options

Create task update task Delete task Search for task Filter task All tasks

In home page we will see some options each option perform individual tasks

By using of axios we can make the HTTP requests from browser each option have the individual HTTP request they are connected to our backend java code in java code we will perform the all data base operations.

1.If we click first option we will get task submission form

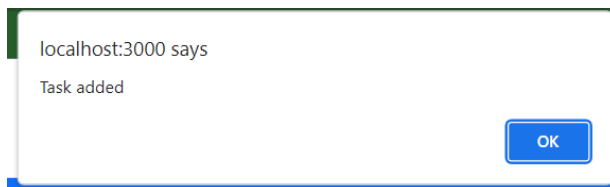


The 'Create Task' form is a web interface for adding new tasks. It features a blue header with the title 'Create Task'. Below the header, there are five input fields: 'Title' (with placeholder 'Enter Title'), 'description' (with placeholder 'Enter description'), 'date' (with placeholder 'dd-mm-yyyy' and a calendar icon), 'Assigned to' (with placeholder 'Enter name'), and 'status' (with radio buttons for 'complete' and 'incomplete'). A 'submit' button is located at the bottom of the form. A 'back' button is positioned at the bottom left of the form area.

By providing of given API we will save the tasks details in data base

```
await axios.post('http://localhost:8080/saveTask',student);  
alert('Task added')
```

After submit we will get an alert as task was submitted



After submit the Api was taken by Dispatcher servlet with request the controller class will map required method based on request.

```

@PostMapping("/saveTask")
public String se(@RequestBody Task s) {
    System.out.println(s);
    return se.sT(s);
}

```

After submission then click back button we redirect to home page.

2. If click second option it will re direct to update page in this we have to update the task based on id. In this we will update the all details of specified task. After clicking of All taska we will get id number of project

```

const re=await axios.post('http://localhost:8080/updateTask',student);
if(re.data==="no"){
    alert('no task is there on this id')
}
else{
    alert('data updated')
}

```

By providing the above API we will connect the java method in controller class

```

@PostMapping("/updateTask")
public String UTask(@RequestBody Task s) {
    if (se.get(s.getId()) == null) {
        return "no";
    }
    return se.uptask(s.getTitle(), s.getDescription(), s.getDueDate(), s.getAssignedto(), s.getStatus(),
        s.getId());
}

```

This method is going to update the all details of task. In updating process if you provide invalid id it will give on alert “There is no task on this id” otherwise it will give an alert as “task updated”. Again then click back button it will redirect to home page

3.if we click third option it will redirect to delete page in this page we delete the task from data base based on id

Delete Task

Title id

Enter Title id

submit

back

```
const re= await axios.post('http://localhost:8080/delete',student);
if(re.data=="no"){
  alert('no task is there on this id')
}
else{
  alert('data deleted')
}
```

By providing the above API we will connect the java method in controller class

```
@PostMapping("/delete")
public String deletetask(@RequestBody Task s) {
    return se.del(s.getId());
}
```

This method is going to delete the task. In deleting process if you provide invalid id it will give on alert “There is no task on this id” otherwise it will give an alert as “data deleted”.

4.By clicking of fourth option we will search the tasks based on task name, description and assigned person

search by name

task name

sub

Submit

ByDescription

ByAssidned

back for search by name

back

	Title	Description	Duedate	Status	Assignedto
1	sub	substraction	2023-09-07	incomplete	suresh

Based on providing button s we will get the data as in table form

5. By clicking of fifth on we will filter the data based on status and due date.

serch by status

status

☐complete ☐incomplete

submit

ByDuedate

back

back

After providing input based on that data will be filtered for example if we provide status as incomplete then we will get result based on incomplete status.

6. By clicking of last option we will get All results from data base

click for all task

id	Title	Description	Duedate	Status	ssignedto
2	sub	subtraction	2023-09-07	incomplete	suresh
3	shooting	firing	2023-09-04	incomplete	vamsi

This is all about front end for user.

For backend operations I use java, Spring JPA and hibernate. We need to import

=>my -SQL connector

=>spring dev tools

=>Lombok

=> spring web

In java I was provided Four packages

1. Entity package
2. Repository package
3. Service package
4. Controller package

1. In entity package we declared Two entities classes one for User registration purpose and second one is for Tasks.

By taking of Entity classes we can map the table in our data base

```
import org.hibernate.annotations.GenericGenerator;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import java.io.Serializable;

public class Task implements Serializable {
    @Id
    @GenericGenerator(name = "auto", strategy = "increment")
    @GeneratedValue(generator = "auto")
    private int id;
    private String title;
    private String description;
    private String dueDate;
    private String assignedTo;
    private String status;
    @Override
    public String toString() {
        return "Task [id=" + id + ", title=" + title + ", description=" + description + ", dueDate=" + dueDate
            + ", assignedTo=" + assignedTo + ", status=" + status + "];"
    }
}

import java.io.Serializable;

import org.hibernate.annotations.GenericGenerator;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import lombok.Data;

@Entity
@Data
public class UserEntity implements Serializable {
    @Id
    @GenericGenerator(name = "auto", strategy = "increment")
    @GeneratedValue(generator = "auto")
    private int id;
    private String name;
    private String password;
    private String phone;
    @Override
    public String toString() {
        return "UserEntity [id=" + id + ", name=" + name + ", password=" + password + ", ph=" + phone + "];"
    }
}
```

The above java codes are Entity classer for our project

2. In repository I am using two JPA repositories one for User and another for tasks.

By using of JPA repository we will get some readymade methods

```

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.jsp.UserManagementSystem.Entity.Task;

public interface TaskRepository extends JpaRepository<Task, Integer> {
    @Modifying
    @Transactional
    @Query("UPDATE Task u SET u.title=:ti,u.description=:des,u.duedate=:due,u.assignedto=:ast,u.status=:st WHERE u.id=:i ")
    void upt(@Param("ti") String title,@Param("des") String dec,@Param("due") String due,@Param("ast") String ast,@Param("st") String st,@Param("i") int id);
    List<Task> findByTitle(String tname);
    List<Task> findByDescription(String des);
    List<Task> findByAssignedto(String ass);
}

```

```

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import com.jsp.UserManagementSystem.Entity.UserEntity;

public interface UserRepository extends JpaRepository<UserEntity, Integer> {
    @Query("SELECT c from UserEntity c WHERE c.name=:name AND c.password=:password")
    UserEntity getUserObject(@Param("name")String name, @Param("password") String password);
}

```

From above first code is for Task repository and second one is for User repository

3. in controller class we will declare methods for mapping the requests . By using of controller class it takes the request from the dispatcher servlet and map the required method based on the method we will get the response from the data base the given code is for controller class

```

import java.util.List;

@RestController
@CrossOrigin("http://localhost:3000")
public class TestController {
    @Autowired
    private UserService se;

    @PostMapping("/save")
    public String save(@RequestBody UserEntity u) {
        se.saveDetails(u);
        System.out.println(u);
        return "data saved";
    }

    @PostMapping("/login")
    public boolean Login(@RequestBody UserEntity u) {
        System.out.println(u);
        UserEntity login = se.login(u.getName(), u.getPassword());
        if (login == null) {
            return false;
        }
        System.out.println(login);
        return true;
    }

    @GetMapping("/get")
    public List<UserEntity> AllDetails() {
        return se.AllUsers();
    }

    @PostMapping("/saveTask")
    public String se(@RequestBody Task s) {
        System.out.println(s);
        return se.sT(s);
    }

    @PostMapping("/updateTask")
    public String UTask(@RequestBody Task s) {
        if (se.get(s.getId()) == null) {
            return "no";
        }
        return se.uptask(s.getTitle(), s.getDescription(), s.getDuedate(), s.getAssignedto(), s.getStatus(),
            s.getId());
    }

    @PostMapping("/delete")
    public String deletetask(@RequestBody Task s) {
        return se.del(s.getId());
    }

    @GetMapping("/task")
    public void task(@RequestBody Task s) {
        System.out.println(s);
        System.out.println(se.byTask(s.getTitle()));
    }

    @GetMapping("/des")
    public void des(@RequestBody Task s) {
        System.out.println(se.byDescription(s.getDescription()));
    }

    @GetMapping("/assi")
    public void assigned(@RequestBody Task s) {
        System.out.println(se.byAssined(s.getAssignedto()));
    }

    @GetMapping("/alltasks")
    public List<Task> AllTasks() {
        System.out.println(se.ALL());
        return se.ALL();
    }
}

```

Here I have come to the end of this project on the topic I would like to share my experience

On this project. I learnt many new things on this project USERMANAGEMENT SYSTEM.

This project increased my research, thinking skill and interest in this subject