


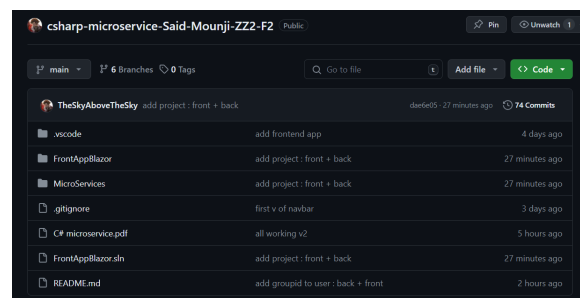
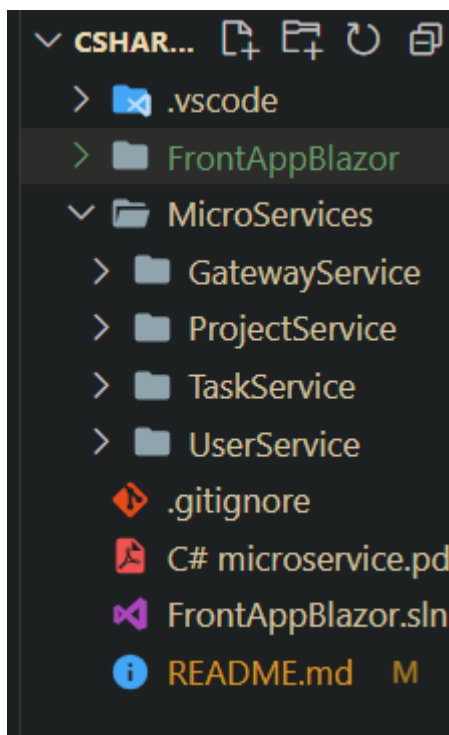
# SM Tasks

Owner	 Said Mounji
Tags	

## Github Repository

<https://github.com/TheSkyAboveTheSky/csharp-microservice-Said-Mounji-ZZ-2-F2>

## Project Overview



## User Model

- Id sous forme user-\*\*\*\*\*
- Nom

- Prenom
- Email
- Password
- Username
- Gender = ["Male", "Female"]
- Role = ["Admin", "User", "ProjectManager"]
- GroupId = [0,1,2,3,4] // 0 => No group

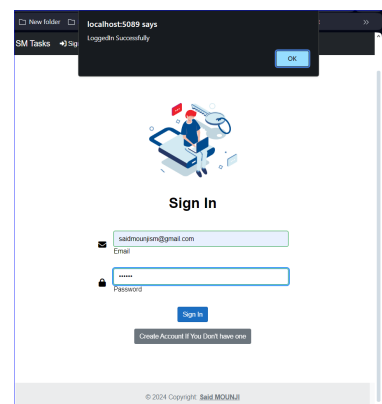
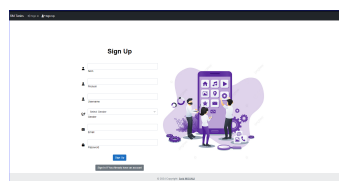
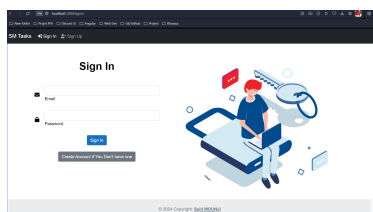
## Task Model

- Id sous forme task-\*\*\*\*\*
- Titre
- Description
- UserId sous forme user-\*\*\*\*\*
- IsChecked
- Username

## Project Model

- Id sous forme project-\*\*\*\*\*
- Nom
- Description
- GroupId = [0,1,2,3,4] // 0 => Not Affected
- Status = ["UpComing", "OnGoing", "Completed"]

## SignIn and SignUp Forms

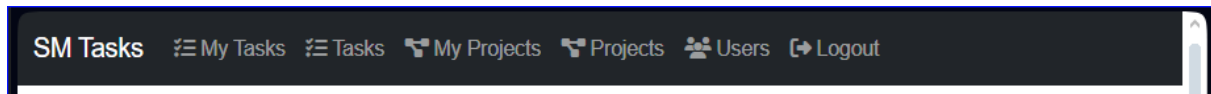


## Navbar

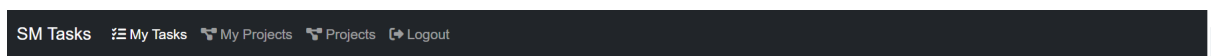
### Responsive



### Admin



### ProjectManager



### User "Normale"



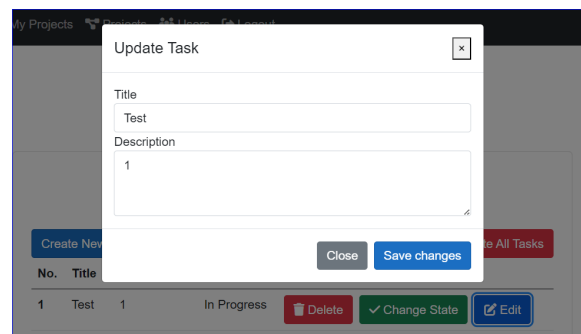
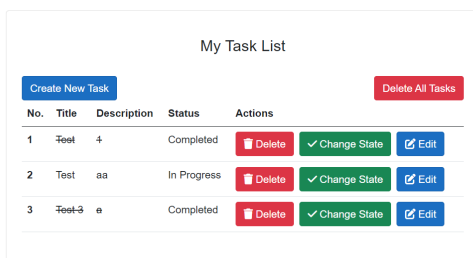
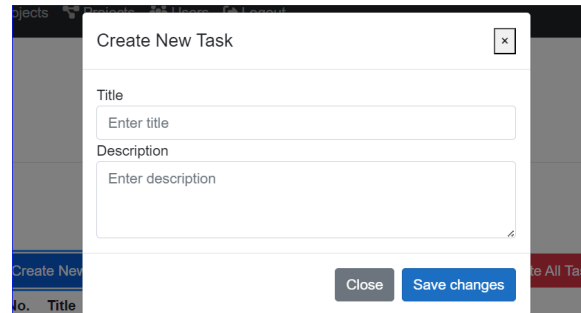
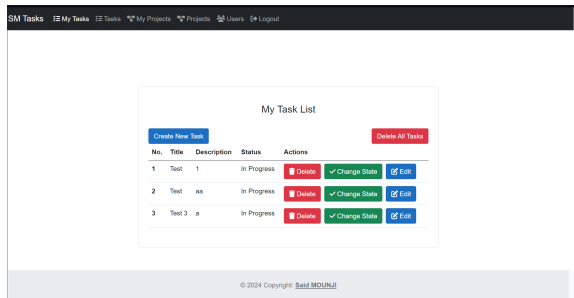
## Pages

### My Tasks

- Only Authentificated User Can Access this page
- List only the tasks linked to the user ( by his id) ( Tas

k.userId)

- Access = ["User", "ProjectManager", "Admin"]



- Create New Task => Create A New Task ( to be linked with the user by his UserId from the localStorage )
- Delete All Tasks => Delete All The Tasks linked to the User
- Delete => Delete the Task by it's id
- Edit => Edit The Task (Can't Edit the UserId)
- Change State => (Completed => In Progress , In Progress => Completed ) if Completed strike Through

## All Tasks

- Only Authenticated User Can Access this page
- List All Tasks
- Access = ["Admin"]

- Create New Task => Create A New Task ( to be linked with the user by his UserId from the localStorage )
- Delete All Tasks => Delete All The Tasks
- Delete => Delete the Task by it's id
- Edit => Edit The Task
- Change State => (Completed => In Progress , In Progress => Completed ) if Completed strike Through

SM Tasks
My Tasks
Tasks
My Projects
Projects
Users
Logout

Task List

Create New Task
Delete All Tasks

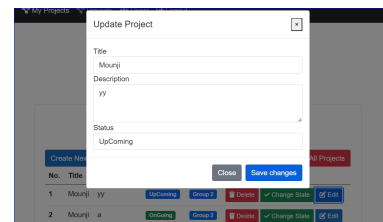
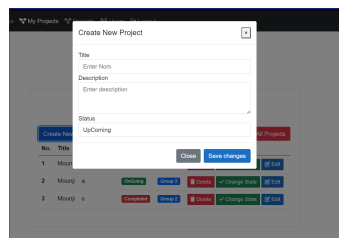
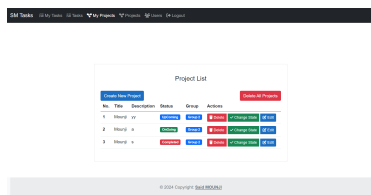
No.	Title	Description	Status	Actions
1	test	1	In Progress	Delete Change State Edit
2	test1	22	In Progress	Delete Change State Edit
3	Test	4	Completed	Delete Change State Edit
4	Test	aa	In Progress	Delete Change State Edit
5	Test-3	e	Completed	Delete Change State Edit

© 2024 Copyright: [Said MOUNJI](#)

## My Projects

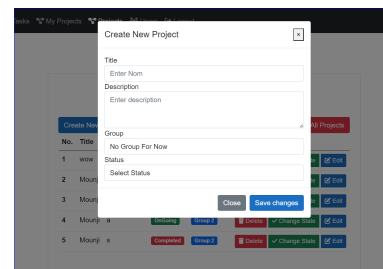
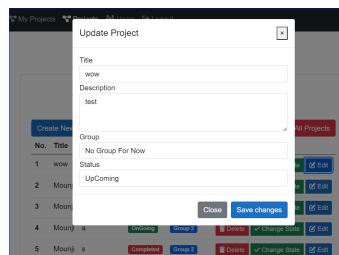
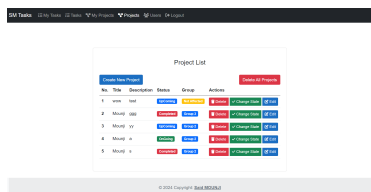
- Only Authenticated User Can Access this page
- List The Projects linked to the User ( User.GroupId == Project.GroupId)
- Access = ["Admin", "User", "ProjectManager"]

- Create New Project => Create A New Project (will be linked with the user by the group)
- Delete All Projects => Delete All The Projects linked to the user
- Delete => Delete the Task by the id
- Edit => Edit The Project ( Can't Edit the Group)
- Change State => ["UpComing", "OnGoing", "Completed"]



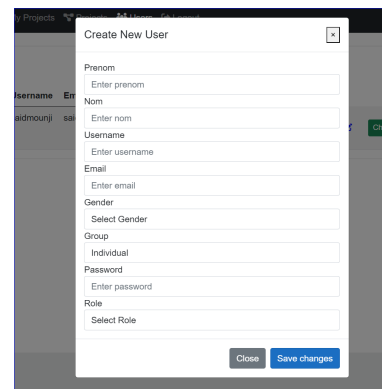
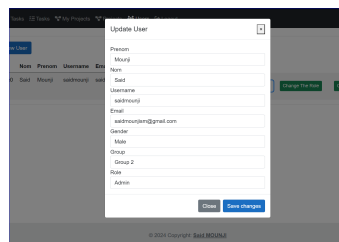
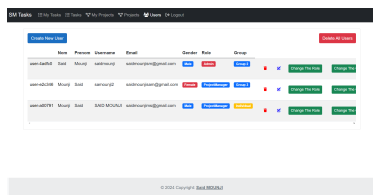
## All Projects

- Only Authenticated User Can Access this page
  - List All The Projects
  - Access = ["Admin", "ProjectManager"]
- 
- Create New Project => Create A New Project
  - Delete All Projects => Delete All The Projects
  - Delete => Delete the Task by the id
  - Edit => Edit The Project ( Can Edit the Group)
  - Change State => ["UpComing", "OnGoing", "Completed"]



## Users

- Only Authenticated User Can Access this page
- Manage The Users
- Access = ["Admin"]



## Logout() Function

```
public async System.Threading.Tasks.Task Logout()
{
    await _sessionStorage.DeleteAsync("jwt");
    await _sessionStorage.DeleteAsync("id");
    await _sessionStorage.DeleteAsync("role");
    await _sessionStorage.DeleteAsync("group");
}
```

## Authentication & Registration Functions

```
public async Task<User?> AuthenticateUser(string email, string password)
{
    var login = new UserLogin() { Email = email, Password = password };

    HttpResponseMessage response = await _httpClient.PostAsJsonAsync("http://localhost:5000/api/user/login", login);
    if (response.IsSuccessStatusCode)
    {
        var result = await response.Content.ReadFromJsonAsync<UserToken>();
        if (result != null)
        {

```

```

        await _sessionStorage.SetAsync("jwt", result.Token);
        await _sessionStorage.SetAsync("id", result.UserId.ToString());
        await _sessionStorage.SetAsync("role", result.User.Role.ToString());
        await _sessionStorage.SetAsync("group", result.UserId.ToString());

        return result.User;
    }
}
return null;
}

```

```

public async Task<User?> RegisterUser(string prenom, string nom, string email, string password, string username, string gender)
{
    var registerInfo = new User(nom, prenom, email, password, username, gender);
    HttpResponseMessage response = await _httpClient.PostAsJsonAsync("http://localhost:5000/api/User/register", registerInfo);
    if (response.IsSuccessStatusCode)
    {
        var result = await response.Content.ReadFromJsonAsync<UserToken>();
        if (result != null)
        {
            await _sessionStorage.SetAsync("jwt", result.Token);
            await _sessionStorage.SetAsync("id", result.UserId.ToString());
            await _sessionStorage.SetAsync("role", result.User.Role.ToString());
            await _sessionStorage.SetAsync("group", result.UserId.ToString());

```



```
        return result.User;
    }
}
return null;
}
```