Project Management System

Project Overview

This documentation presents a comprehensive overview of the Project Management System developed with ASP .NET core web API.
The system aims to facilitate efficient task management for small companies working on projects by providing a platform for project and team-oriented collaboration.

In response to the project requirements outlined by the instructor, the Task Management System has been crafted to align with various aspects of .NET technologies, including API development, Entity Framework Code First, Dependency Injection (IoC), and potentially frontend development.

Platform Pages Description:

1. Project Dashboard: The Project Dashboard serves as the central hub for managing projects within the Task Management System. It allows authorized users, such as administrators and team leads, to perform various actions related to project management. Key functionalities include:

- **Project Creation:** Authorized users can create new projects, specifying details such as project name, description, start date, and end date.
- **Team Member Assignment:** Users can add team members to projects, assigning roles and responsibilities to each member.
- **Project Overview:** The dashboard provides an overview of all projects, including their status, progress, and assigned team members.
- **Search Functionality:** Users can search for specific projects based on criteria such as project name or team members.

2. Team Page: The Team Page offers comprehensive management of team members within the Task Management System. Key features include:

- **Team Member List:** Displays a list of all team members along with their basic details such as name, role, and contact information.
- **Add Team Member:** Administrators have the privilege to add new team members to the system, specifying their roles and permissions.
- **Member Details:** Users can view detailed information about a specific team member, including their profile, contact details, and assigned projects.
- **Filters and Sorting:** Users can filter team members based on performance metrics like "Employee of the Year".

3. Tasks Page: The Tasks Page facilitates efficient management and tracking of tasks within the Task Management System. It provides functionalities for categorizing tasks, tracking their status, and ensuring timely completion. Key features include:

- **Task Filtering:** Users can filter tasks based on project, priority, status (e.g., active, QA, validation, done), or assigned team member.
- **Task Categorization:** Tasks are categorized into different stages such as active, QA, validation, and done, allowing users to easily track their progress.
- **Project-based View:** Users can view tasks specific to a particular project, enabling focused management and collaboration.

4. Request Day Off Page: The Request Day Off Page enables team members to request time off from work, providing a streamlined process for managing leave requests. Key functionalities include:

- **Leave Duration Calculation:** The system automatically calculates the duration of the requested leave based on the specified dates.

5. Send Email Page: The Send Email Page empowers administrators to communicate effectively with team members using predefined email templates. Key features include:

- **Email Template Selection:** Administrators can choose from a set of predefined email templates tailored for common scenarios such as vacation requests or team meetings.
- **Customization Options:** Users can customize email content and recipients based on specific requirements.

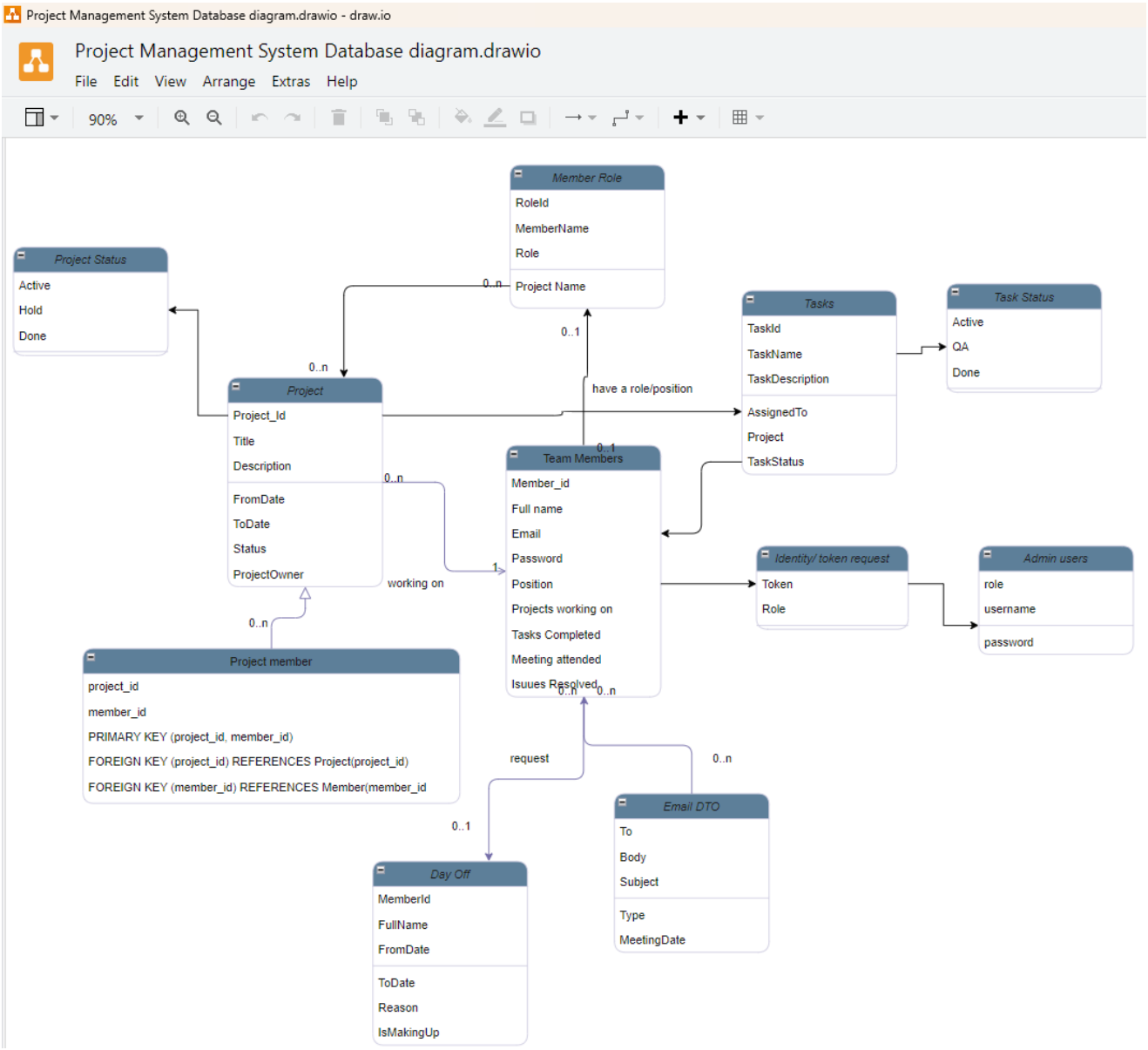
Database

First, I visualized the tables schema using Draw.io, creating a comprehensive database diagram for the Project Management System.

> You can find the detailed schema in the "Project Management System Database Diagram" in Draw.io.

Next, I developed the classes corresponding to each entity and the DbContext to interact with the database. Leveraging Entity Framework, I utilized code-first migration to generate the tables based on the defined classes, ensuring seamless integration between the application and the database.

I utilized SQL Server Management Studio (SSMS) to manage and monitor the database.



Main Tables :

Project

Field	Type	Null	Key	Default	Extra
ProjectId	int	NO	PRI	NULL	auto_increment
ProjectName	nvarchar(100)	NO		NULL	
ProjectDescr	nvarchar(max)	NO		NULL	
FromDate	datetime2	YES		NULL	
ToDate	datetime2	YES		NULL	
Status	int	NO		NULL	
ProjectOwner	nvarchar(max)	NO		NULL	

TeamMember

Field	Type	Null	Key	Default	Extra
MemberId	int	NO	PRI	NULL	auto_increment
FullName	nvarchar(max)	NO		NULL	
Email	nvarchar(max)	NO		NULL	
Password	nvarchar(max)	NO		NULL	
Position	nvarchar(max)	NO		NULL	
ProjectsComp	nvarchar(max)	NO		NULL	
MeetingsAttd	nvarchar(max)	NO		NULL	
IssuesResolv	nvarchar(max)	NO		NULL	

Email

Field Name	Type	Key	Default	Extra
EmailId	int	PRI	NULL	auto_increment
To	nvarchar(max)	NO		
Body	nvarchar(max)	NO		
Subject	nvarchar(max)	NO		
Type	nvarchar(max)	NO		
MeetingDate	datetime	YES	NULL	

Tasks

Field Name	Type	Key	Default	Extra
TaskId	int	PRI	NULL	auto_increment
Title	nvarchar(max)	NO		
Description	nvarchar(max)	NO		
AssignedTo	nvarchar(max)	NO		
Status	int	NO	NULL	
ProjectId	int		NULL	

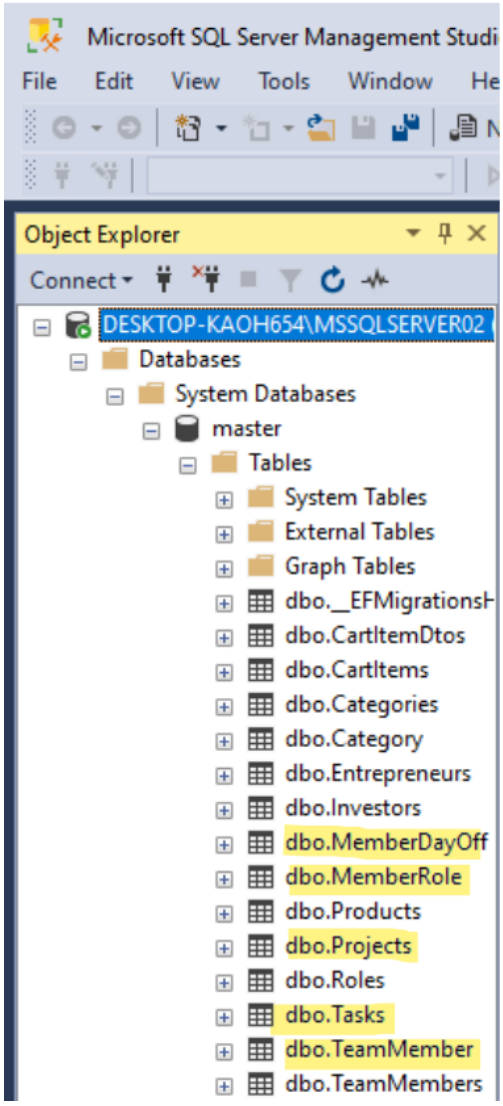
Member Day off

Field Name	Type	Key	Default	Extra
MemberId	long	PRI	NULL	
FullName	nvarchar(max)		NULL	
FromDate	datetime		NULL	
ToDate	datetime		NULL	

Field Name	Type	Key	Default	Extra
Reason	nvarchar(max)		NULL	
IsMakingUp	bool		NULL	
VacationDuration	int		NULL	

```
CREATE TABLE Project (
    ProjectId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    ProjectName NVARCHAR(100) NOT NULL,
    ProjectDescr NVARCHAR(MAX) NOT NULL,
    FromDate DATETIME2,
    ToDate DATETIME2,
    Status INT NOT NULL,
    ProjectOwner NVARCHAR(MAX) NOT NULL
);


CREATE TABLE TeamMember (
    MemberId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    FullName NVARCHAR(MAX) NOT NULL,
    Email NVARCHAR(MAX) NOT NULL,
    Password NVARCHAR(MAX) NOT NULL,
    Position NVARCHAR(MAX) NOT NULL,
    ProjectsComp NVARCHAR(MAX) NOT NULL,
    MeetingsAttd NVARCHAR(MAX) NOT NULL,
    IssuesResolv NVARCHAR(MAX) NOT NULL
);


CREATE TABLE Email (
    EmailId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    To NVARCHAR(MAX) NOT NULL,
    Body NVARCHAR(MAX) NOT NULL,
    Subject NVARCHAR(MAX) NOT NULL,
    Type NVARCHAR(MAX) NOT NULL,
    MeetingDate DATETIME
);


CREATE TABLE Tasks (
    TaskId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Title NVARCHAR(MAX) NOT NULL,
    Description NVARCHAR(MAX) NOT NULL,
    AssignedTo NVARCHAR(MAX) NOT NULL,
    Status INT NOT NULL,
    ProjectId INT
);
```

```
CREATE TABLE MemberDayOff (
    MemberId LONG NOT NULL,
    FullName NVARCHAR(MAX),
    FromDate DATETIME,
    ToDate DATETIME,
    Reason NVARCHAR(MAX),
    IsMakingUp BOOL,
    VacationDuration INT
);
```



APIs

0. Authorization

The IdentityController handles authentication and token generation for user access to the system. generating authentication tokens based on user claims.

1. **Constructor Injection:** The controller class injects an IConfiguration instance via its constructor. This IConfiguration object is used to access configuration settings required for token generation.
2. **GenerateToken Action:**
    - This action method handles HTTP POST requests to the "/api/identity/token" endpoint.
    - It receives a TokenRequestModel object in the request body, containing user claims.
    - The method validates the model's state and the user's claims.
    - If the claims include a valid "role" claim (either "admin" or "teamLead"), a JWT token is generated using the GenerateJwtToken method and returned as a response.
    - If the validation fails, an Unauthorized response is returned.

```
[HttpPost("token")]
0 references
public IActionResult GenerateToken([FromBody] TokenRequestModel model)
{
    if (ModelState.IsValid)
    {

        if (model.Claims.ContainsKey("role") &&
            (model.Claims["role"] == "admin" || model.Claims["role"] == "teamLead"))
        {
            var token = GenerateJwtToken(model.Claims);
            return Ok(new { token });
        }
    }

    return Unauthorized();
}
```

3. **GenerateJwtToken Method:**

```
- This private method generates a JWT token based on the provided claims.
- It retrieves the necessary security key and credentials from the
IConfiguration object.
- Using these credentials, it creates a JWT token with the specified issuer,
audience, claims, expiration time, and signing credentials.
- The generated token is returned as a string.
```

4. **TokenRequestModel:**

```
- This class defines the structure of the request model expected by the
GenerateToken action.
- It contains a property named "Claims," which is a dictionary
representing the user claims.
```

```
    1 reference
    private string GenerateJwtToken(Dictionary<string, string> claims)
    {
        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JwtSettings:Key"]));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: _configuration["JwtSettings:Issuer"],
            audience: _configuration["JwtSettings:Audience"],
            claims: claims.Select(c => new Claim(c.Key, c.Value)),
            expires: DateTime.Now.AddMinutes(30),
            signingCredentials: creds
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

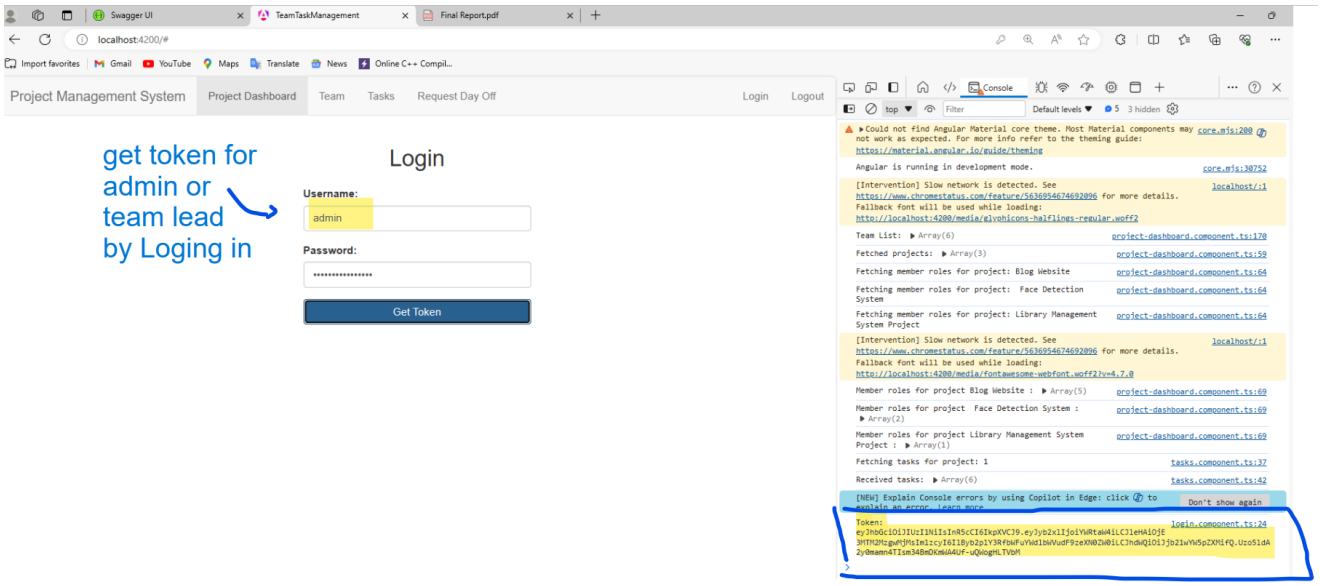
```
1 reference
public class TokenRequestModel
{
    4 references
    public Dictionary<string, string> Claims { get; set; }
}
```

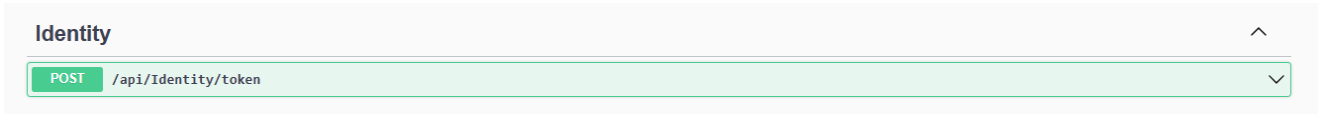
5. **Authenticated Users:**
   - Authenticated users, as denoted by the `[Authorize]` attribute, have access to the POST endpoint `PostProject(string projectName, string projectDescription, DateTime? fromDate, DateTime? toDate, ProjectStatus? status, string projectOwner)` to create a new project. However, they need to be authenticated to access this endpoint.
6. **Team Leads and Administrators:**

```
- Team leads and administrators, as denoted by the `[Authorize(Roles = "teamLead,admin")]` attribute, have access to the PUT endpoint `PutProject(int id, Project project)` to update existing project information.
```

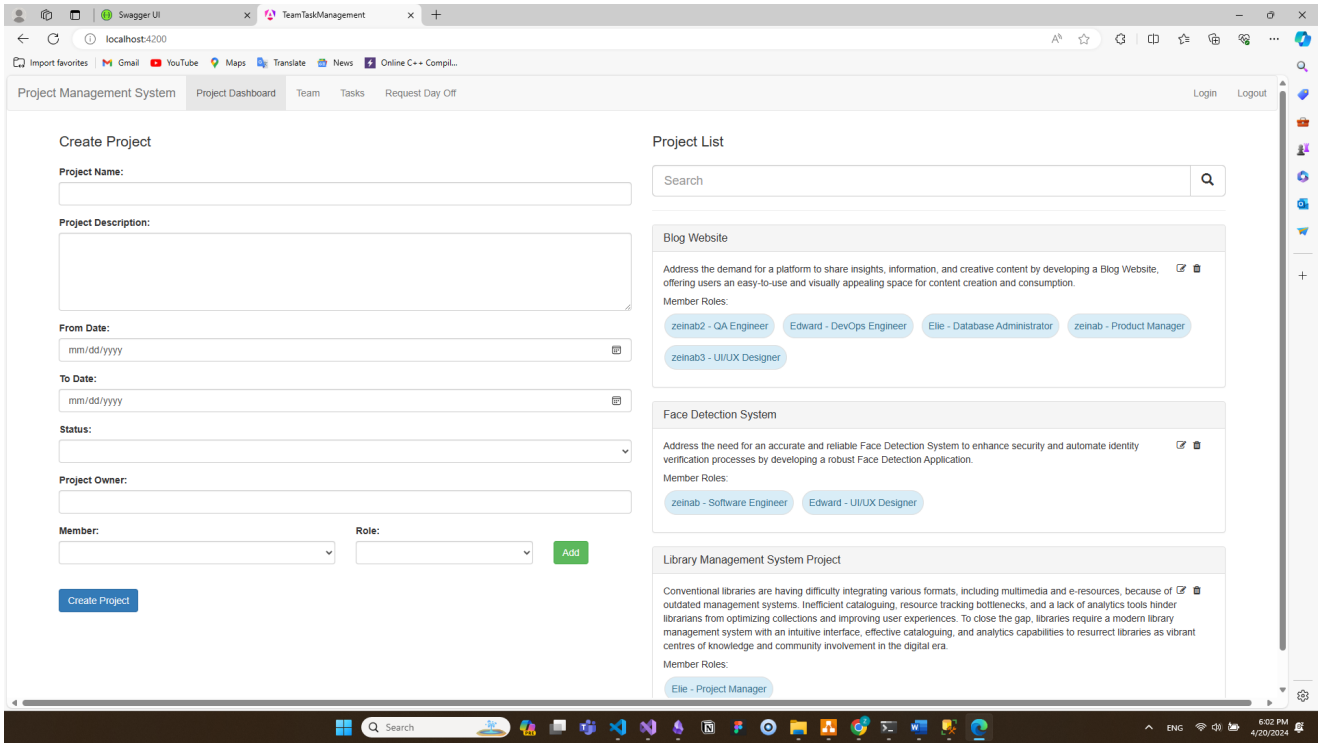
```
[HttpPut("{id}")]
[Authorize(Roles = "teamLead,admin")]
0 references
public async Task<IActionResult> PutProject(int id, Project project)
{
    if (id != project.ProjectId)
    {
        return BadRequest();
```

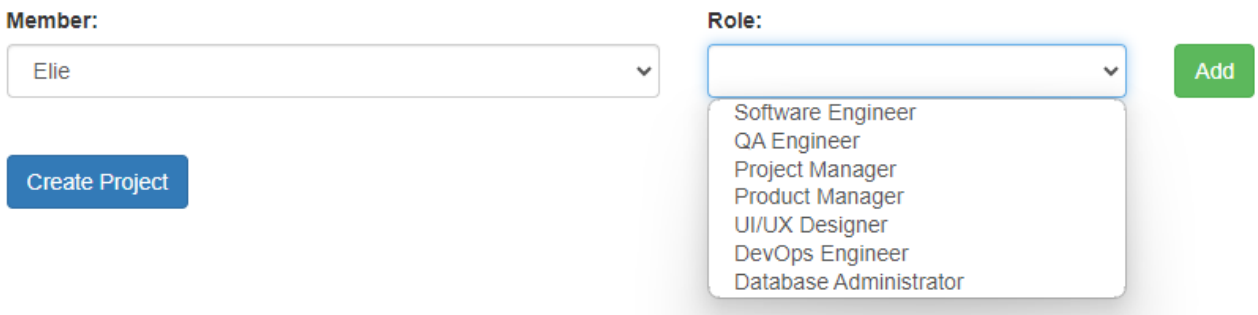
7. **Administrators:**
   - Administrators, as denoted by the `[Authorize(Roles = "admin")]` attribute, have exclusive access to the DELETE endpoint `DeleteProject(int id)` to delete a project. This endpoint is restricted to administrators only.
8. **General Queries:**
   - Users can also perform general queries using the GET endpoint `GetProjectsByName(string projectName)` to search for projects by name.

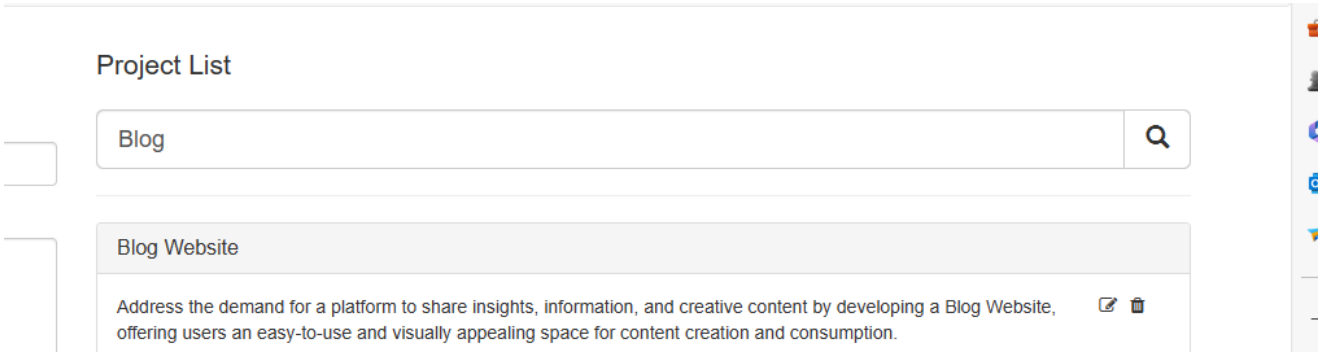




1. Project Dashboard



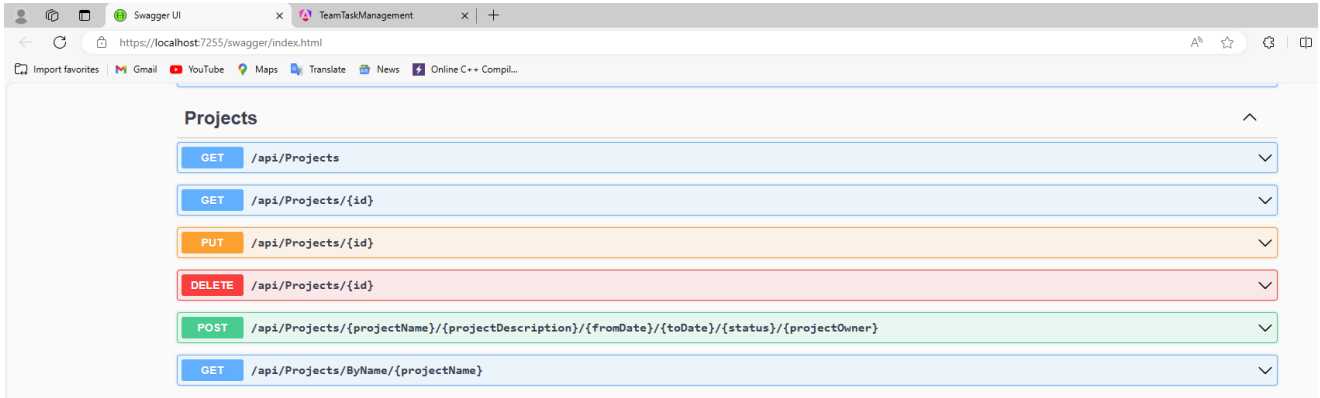




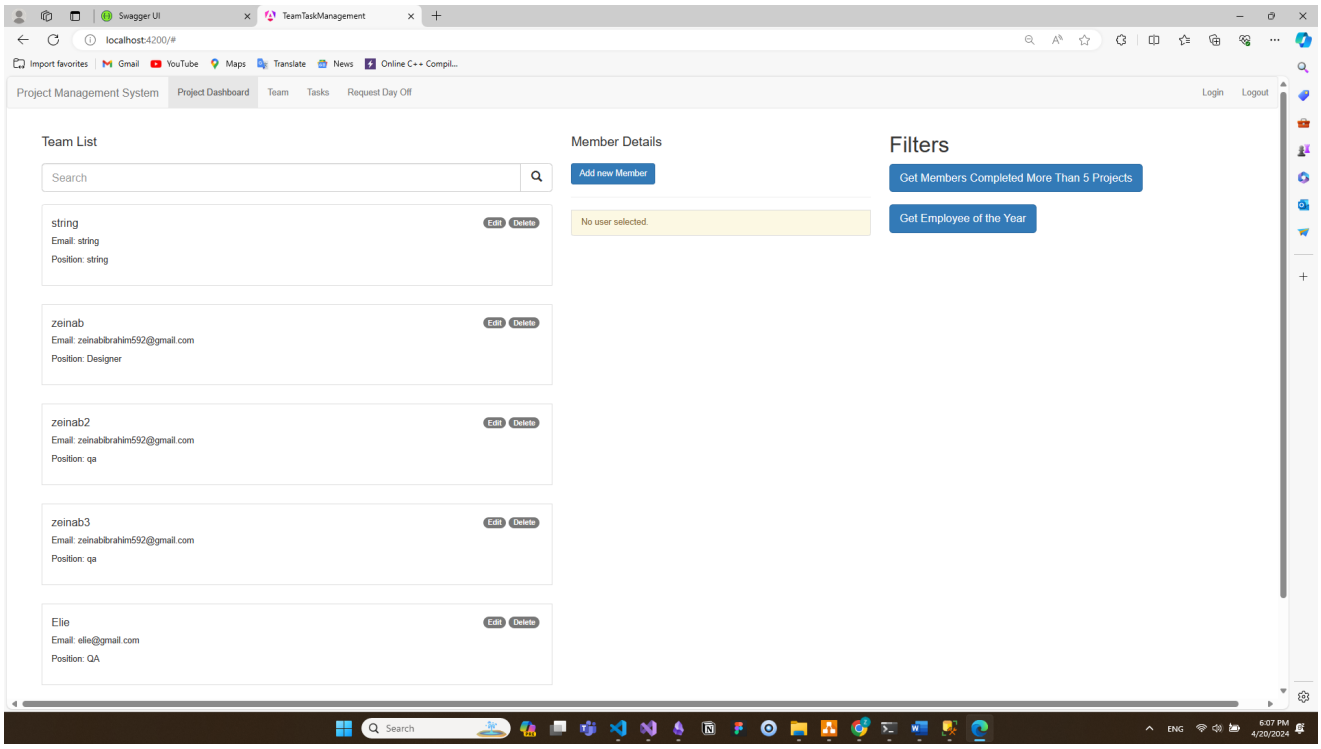
Platform Pages Description:

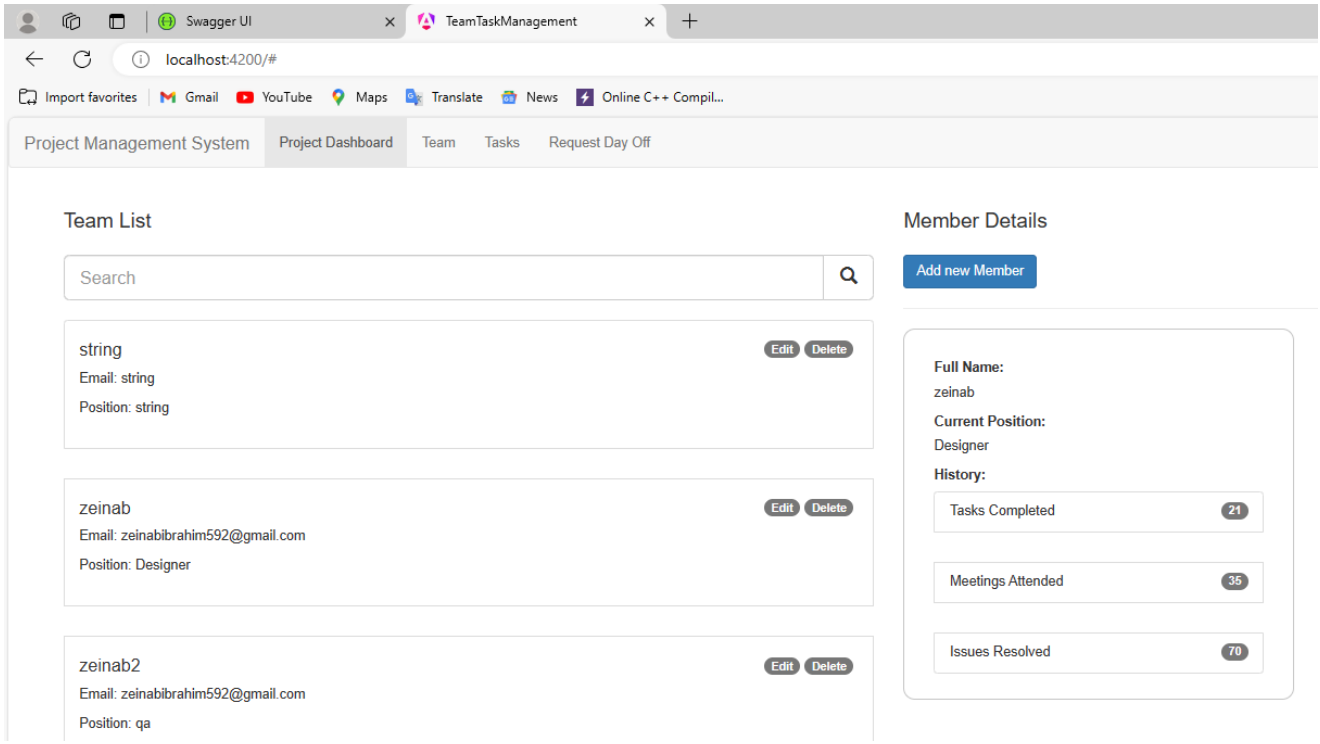
1. Project Dashboard: The Project Dashboard serves as the central hub for managing projects within the Task Management System. It allows authorized users, such as administrators and team leads, to perform various actions related to project management. Key functionalities include:

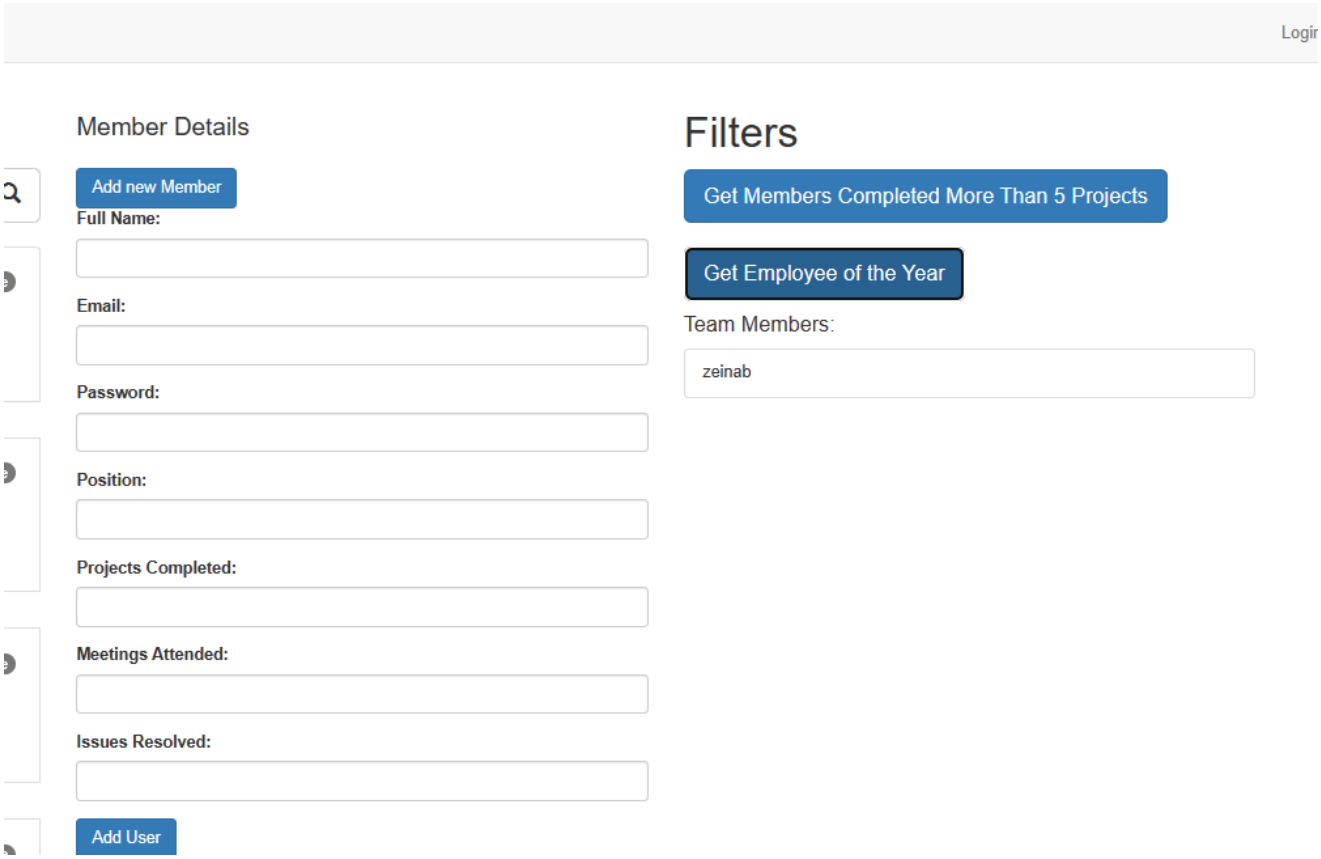
- **Project Creation:** Authorized users can create new projects, specifying details such as project name, description, start date, and end date.
- **Team Member Assignment:** Users can add team members to projects, assigning roles and responsibilities to each member.
- **Project Overview:** The dashboard provides an overview of all projects, including their status, progress, and assigned team members.
- **Search Functionality:** Users can search for specific projects based on criteria such as project name or team members.



2. Team

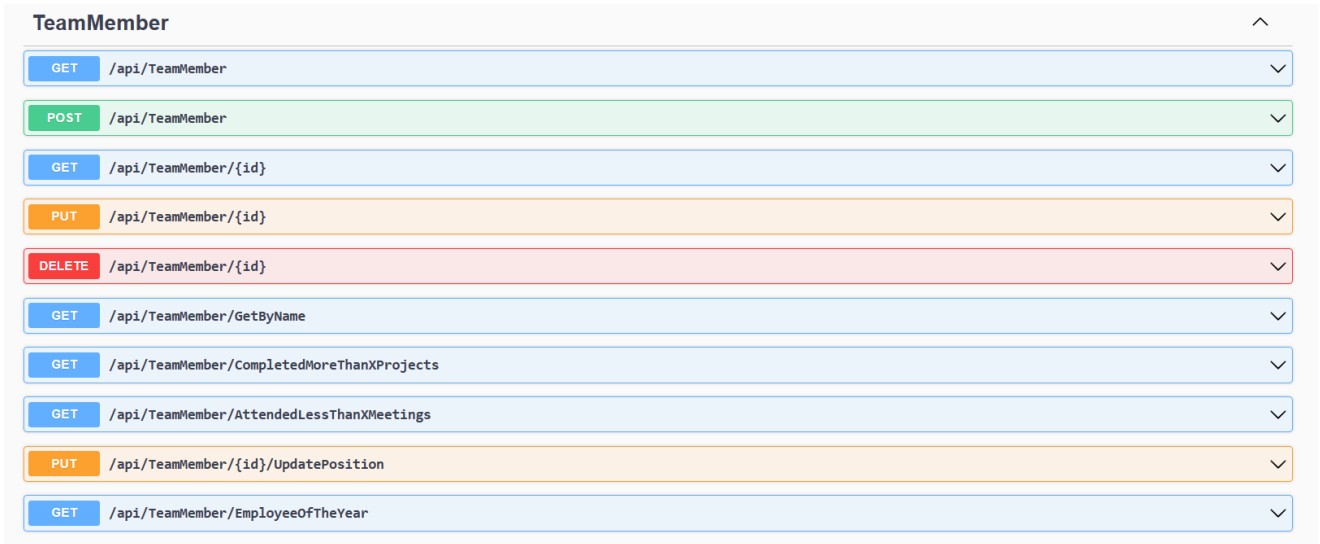

Swagger UI
TeamTaskManagement
localhost:4200/#
Import favorites
Gmail
YouTube
Maps
Translate
News
Online C++ Compil...
Project Management System
Project Dashboard
Team
Tasks
Request Day Off
Login
Logout
Team List
Search
string
Email: string
Position: string
Edit
Delete
zeinab
Email: zeinabibrahim592@gmail.com
Position: Designer
zeinab2
Email: zeinabibrahim592@gmail.com
Position: qa
zeinab3
Email: zeinabibrahim592@gmail.com
Position: qa
Elie
Email: elie@gmail.com
Position: QA
Member Details
Add new Member
No user selected.
Filters
Get Members Completed More Than 5 Projects
Get Employee of the Year


Swagger UI
TeamTaskManagement
localhost:4200/#
Import favorites
Gmail
YouTube
Maps
Translate
News
Online C++ Compil...
Project Management System
Project Dashboard
Team
Tasks
Request Day Off
Team List
Search
string
Email: string
Position: string
Edit
Delete
zeinab
Email: zeinabibrahim592@gmail.com
Position: Designer
zeinab2
Email: zeinabibrahim592@gmail.com
Position: qa
Member Details
Add new Member
Full Name:
zeinab
Current Position:
Designer
History:
Tasks Completed
21
Meetings Attended
35
Issues Resolved
70

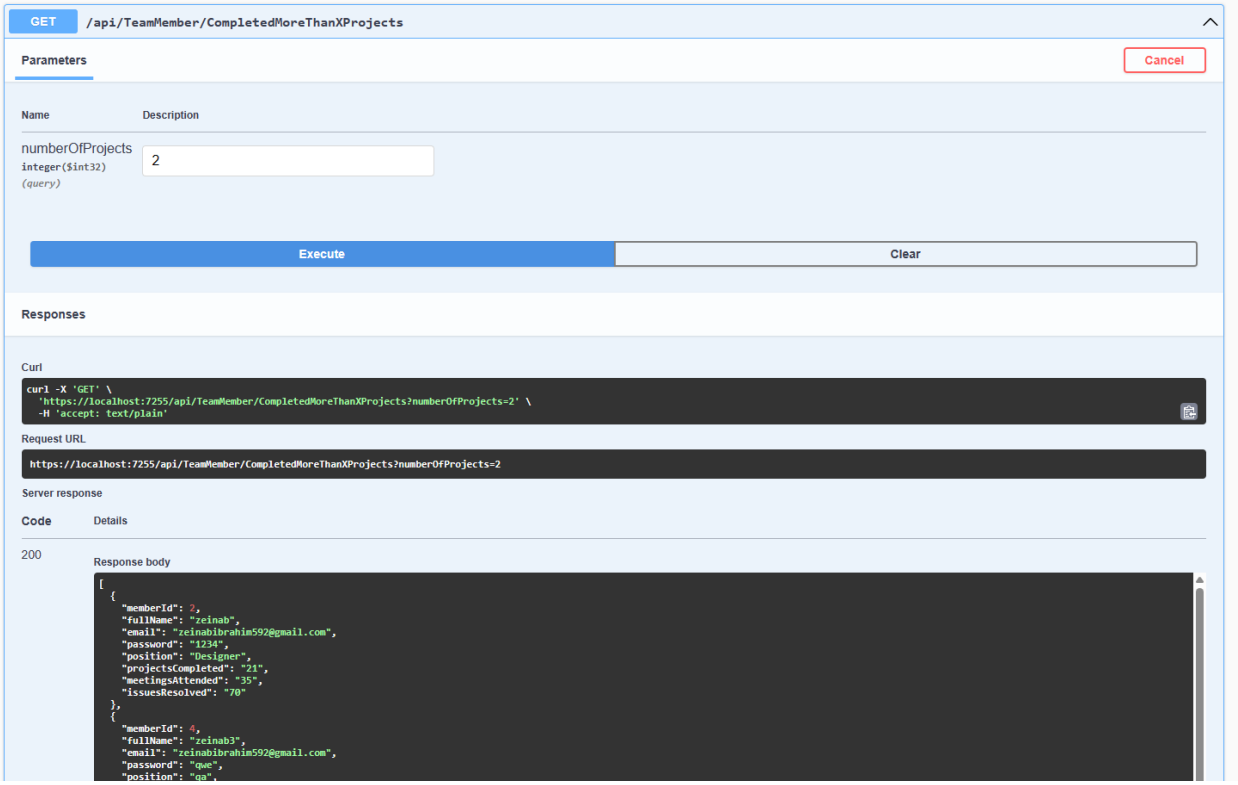


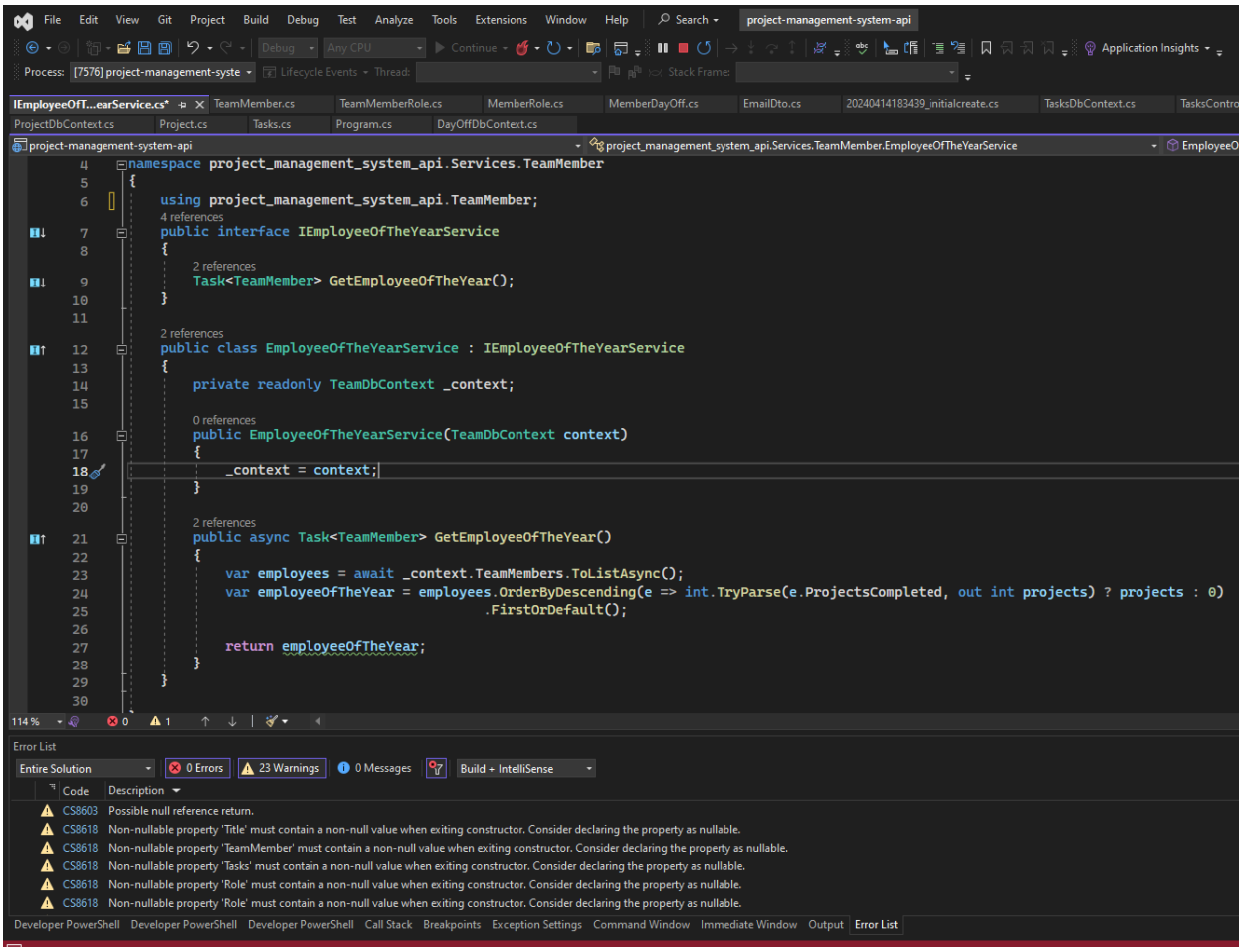
2. Team Page: The Team Page offers comprehensive management of team members within the Task Management System. Key features include:

- **Team Member List:** Displays a list of all team members along with their basic details such as name, role, and contact information.
- **Add Team Member:** Administrators have the privilege to add new team members to the system, specifying their roles and permissions.
- **Member Details:** Users can view detailed information about a specific team member, including their profile, contact details, and assigned projects.
- **Filters and Sorting:** Users can filter team members based on performance metrics like "Employee of the Year".



- Get Members That had completed more than 2 projects



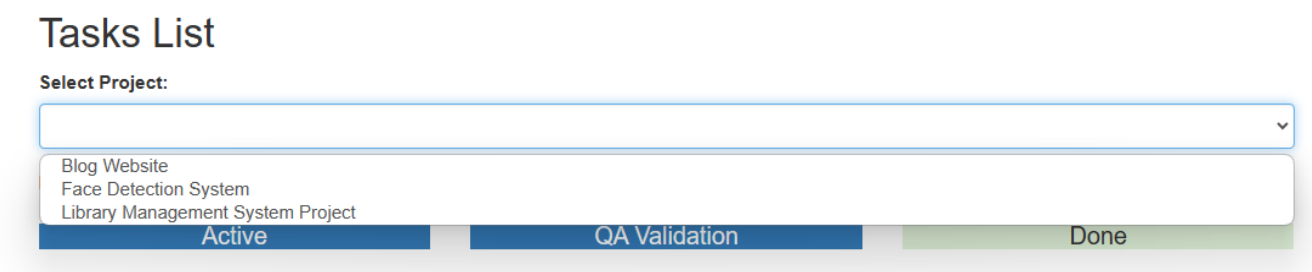


This code defines an interface and a corresponding service class for retrieving the "Employee of the Year" from a database of team members.

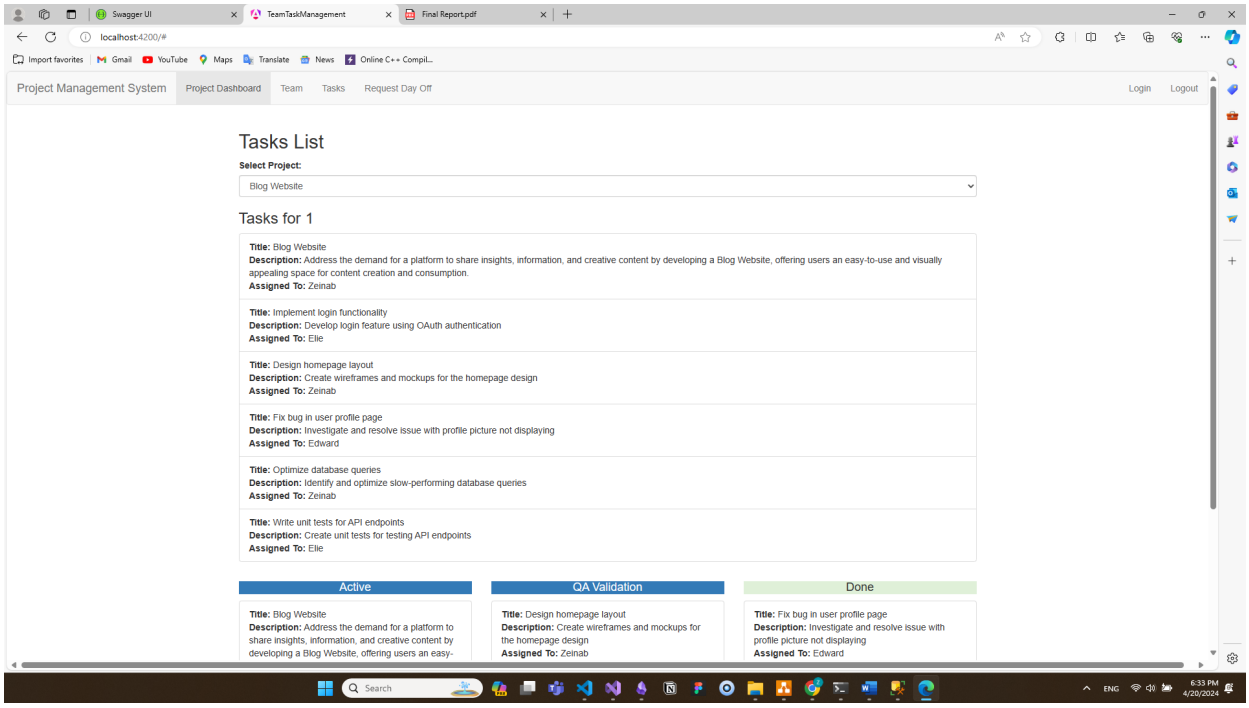
3. **EmployeeOfTheYearService Class**:

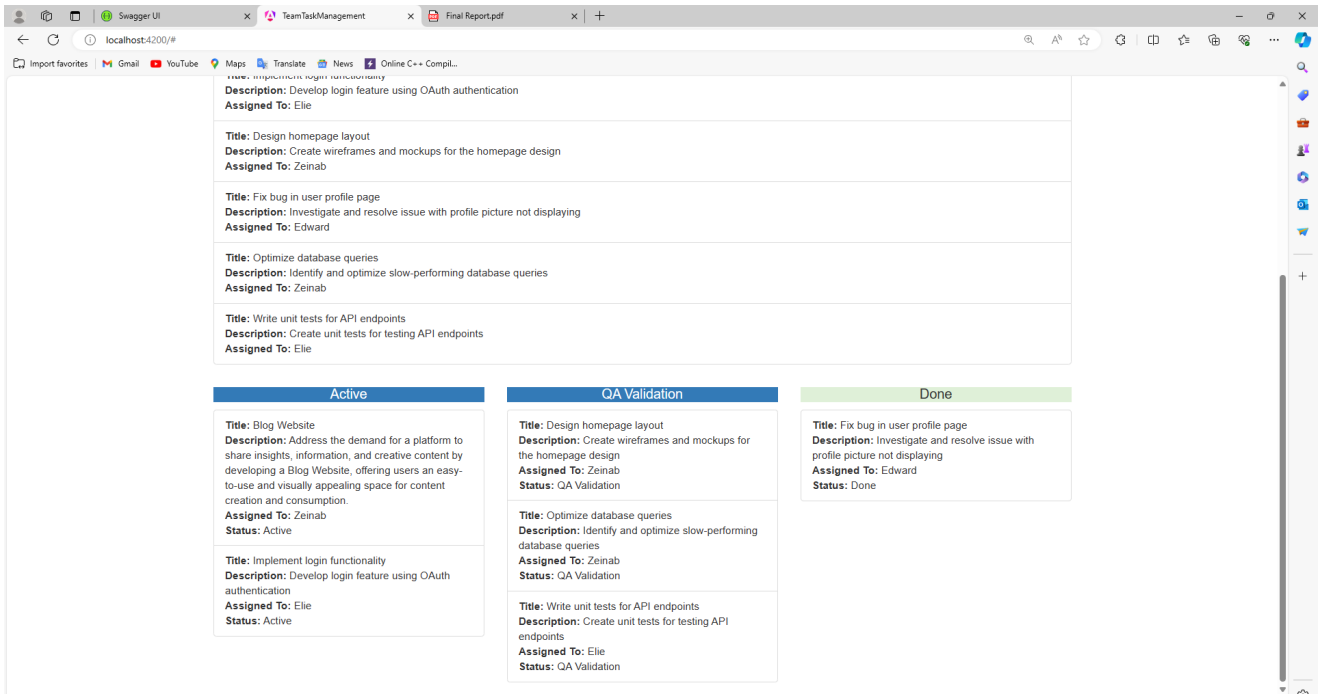
- The `EmployeeOfTheYearService` class implements the `IEmployeeOfTheYearService` interface.
- It has a constructor that takes a `TeamDbContext` instance as a parameter.
- The `GetEmployeeOfTheYear()` method retrieves the list of team members from the database asynchronously using Entity Framework's `ToListAsync()` method.
- It then determines the "Employee of the Year" based on a specific criteria. In this case, the employee with the highest number of completed projects (`ProjectsCompleted`) is considered the "Employee of the Year".

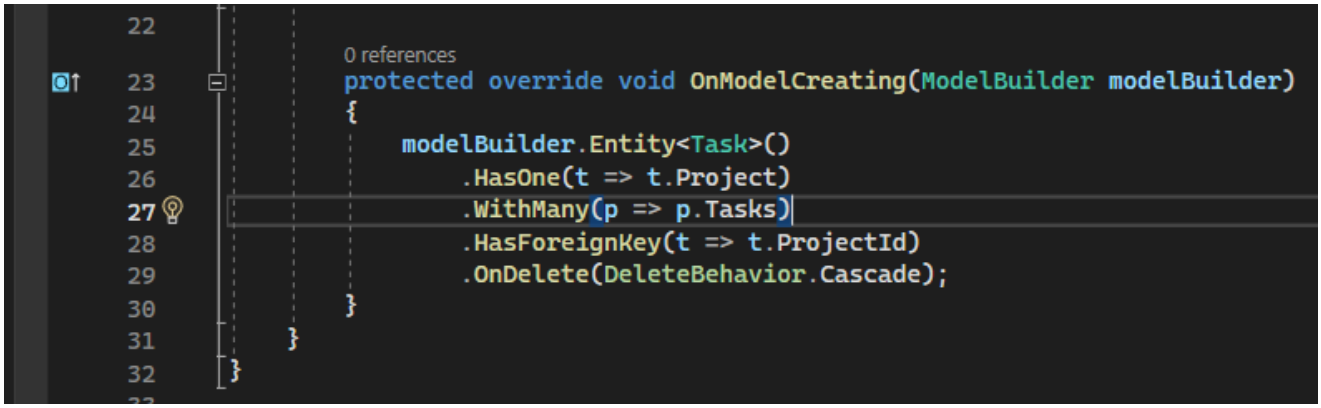
Tasks



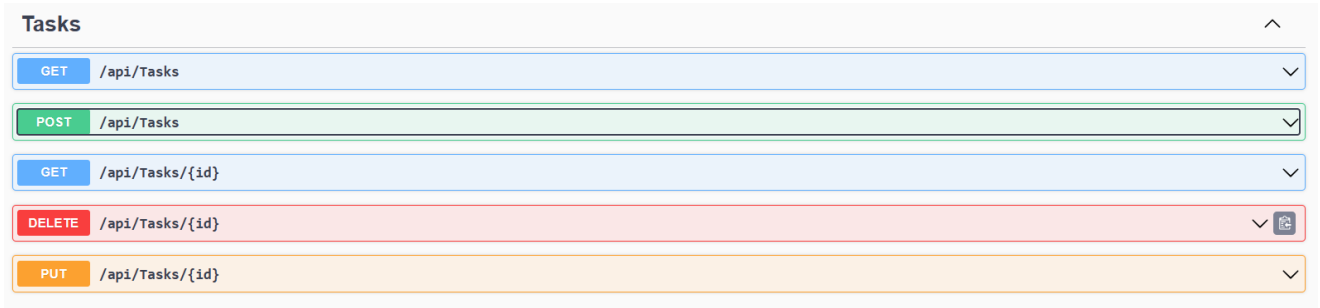
- Get Tasks for Blog Website Project



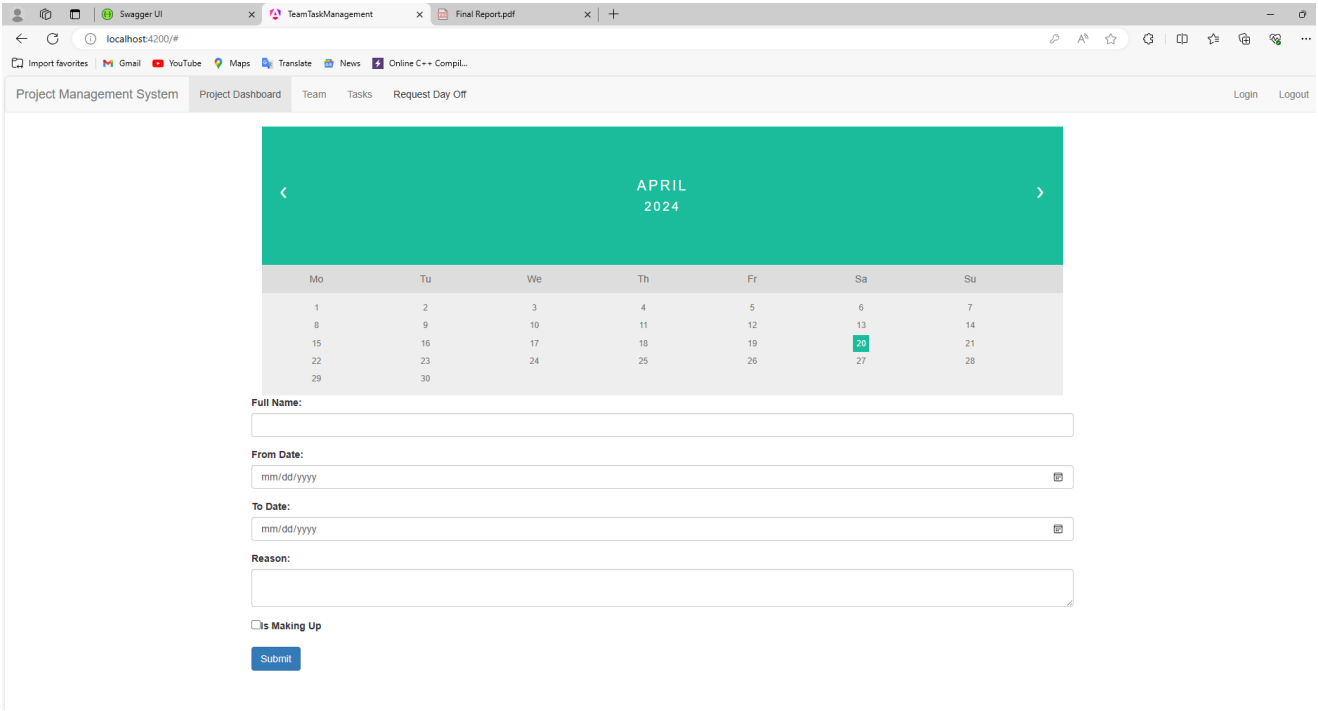




> Project has many Tasks and each task belongs to one project



Request Day Off



4. Request Day Off Page: The Request Day Off Page enables team members to request time off from work, providing a streamlined process for managing leave requests. Key functionalities include:

- **Off Days Duration Calculation:** The system automatically calculates the duration of the requested leave based on the specified dates.

1. **VacationService Class:**
   - This method calculates the duration of a vacation by subtracting the start date from the end date to obtain a TimeSpan, and then returns the total number of days in the TimeSpan plus one. Adding one ensures that the final duration includes both the start and end dates.

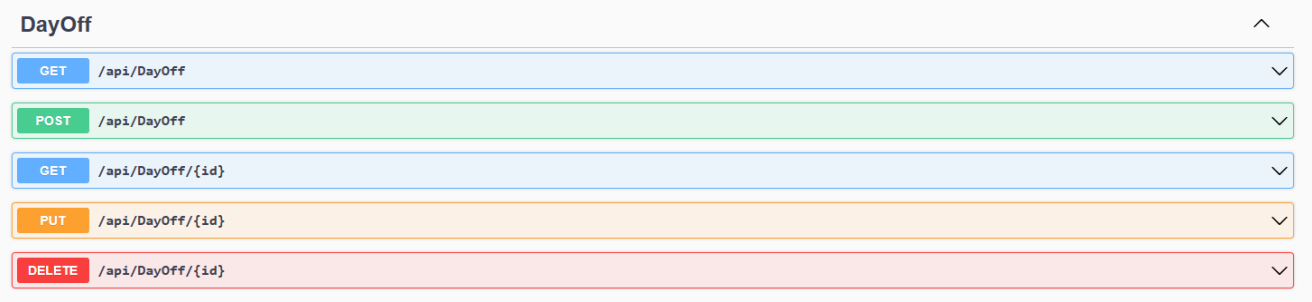
```
IVacationService.cs | MemberDayOff.cs | DayOffController.cs | ProjectController.cs | EmailController.cs | IdentityController.cs | Project.cs | Tasks.cs
project-management-system-api
project_management_system_api.Services.DaysOff.VacationService

namespace project_management_system_api.Services.DaysOff
{



        4 references
        public interface IVacationService
        {
            2 references
            int CalculateVacationDuration(DateTime fromDate, DateTime toDate);
        }

        1 reference
        public class VacationService : IVacationService
        {
            2 references
            public int CalculateVacationDuration(DateTime fromDate, DateTime toDate)
            {

                TimeSpan duration = toDate - fromDate;
                return duration.Days + 1;
            }
        }


}
```

2. **Constructor Injection:** The controller class injects a DayOffDbContext instance and an IVacationService instance via its constructor. These dependencies are used to interact with the database and perform business logic, respectively.

```
public class DayOffController : ControllerBase
{
    private readonly DayOffDbContext _context;
    private readonly IVacationService _vacationService;

    0 references
    public DayOffController(DayOffDbContext context, IVacationService vacationService)
    {
        _context = context;
        _vacationService = vacationService;
    }
```

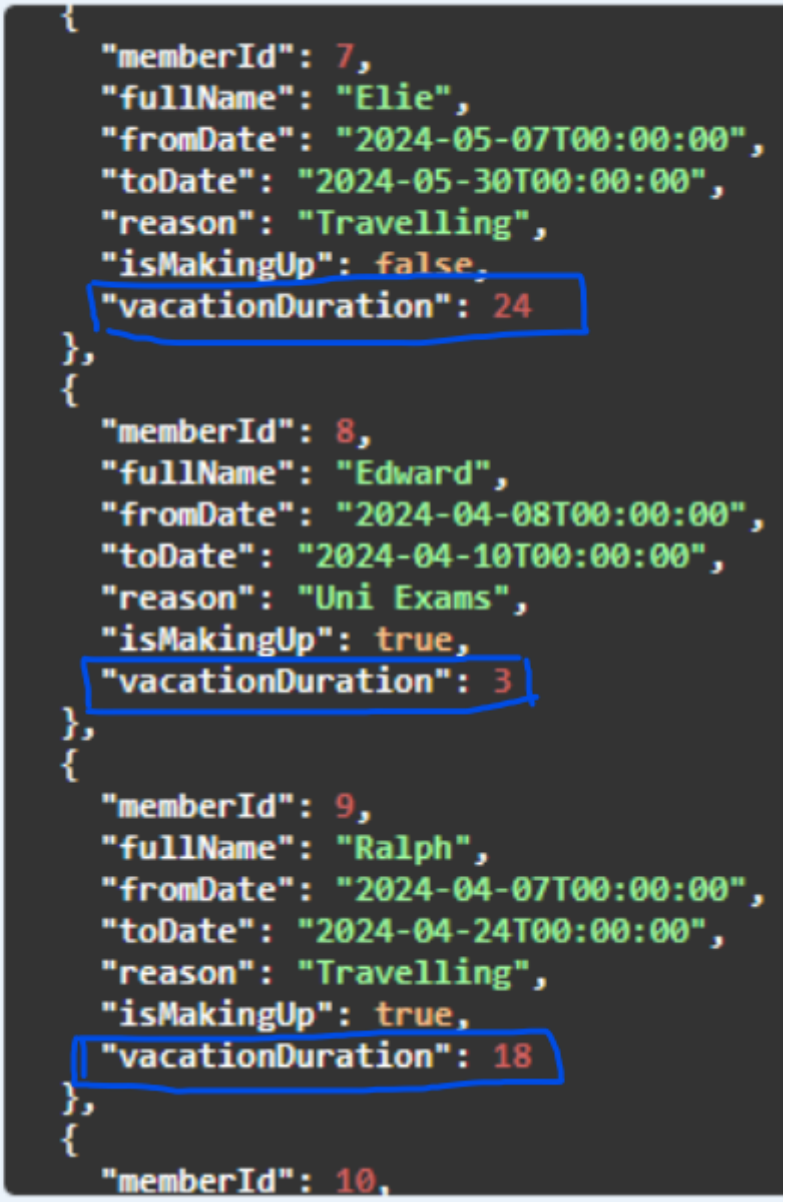
3. **Business Logic:**
   - The `PostMemberDayOff([FromBody] MemberDayOff memberDayOff)` method incorporates business logic to calculate the duration of the vacation based on the provided start and end dates. This duration is then added to the member day-off record before saving it to the database.

```
[HttpPost]
0 references
public async Task<ActionResult<MemberDayOff>> PostMemberDayOff([FromBody] MemberDayOff memberDayOff)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    int vacationDuration = _vacationService.CalculateVacationDuration(memberDayOff.FromDate, memberDayOff.ToDate);

    memberDayOff.VacationDuration = vacationDuration;

    _context.MemberDayOff.Add(memberDayOff);
    await _context.SaveChangesAsync();

    return CreatedAtAction(nameof(GetMemberDayOff), new { id = memberDayOff.MemberId }, memberDayOff);
}
```



200

Response body




```
{
  "memberId": 7,
  "fullName": "Elie",
  "fromDate": "2024-05-07T00:00:00",
  "toDate": "2024-05-30T00:00:00",
  "reason": "Travelling",
  "isMakingUp": false,
  "vacationDuration": 24
},
{
  "memberId": 8,
  "fullName": "Edward",
  "fromDate": "2024-04-08T00:00:00",
  "toDate": "2024-04-10T00:00:00",
  "reason": "Uni Exams",
  "isMakingUp": true,
  "vacationDuration": 3
},
{
  "memberId": 9,
  "fullName": "Ralph",
  "fromDate": "2024-04-07T00:00:00",
  "toDate": "2024-04-24T00:00:00",
  "reason": "Travelling",
  "isMakingUp": true,
  "vacationDuration": 18
},
{
  "memberId": 10,
```


Email

1. Code Structure: The code is organized into three main parts:

- **EmailService Class:** This class implements the `IEmailService` interface and contains methods for sending regular emails (`SendEmail`) and meeting invitations (`SendMeetingEmail`). It utilizes the MimeKit library for constructing MIME messages

and MailKit library for SMTP client functionality.

```
namespace project_management_system_api.Services.EmailService
{
    4 references
    public interface IEmailService
    {
        3 references
        void SendEmail(EmailDto request);
        2 references
        void SendMeetingEmail(EmailDto request);
    }
}
```

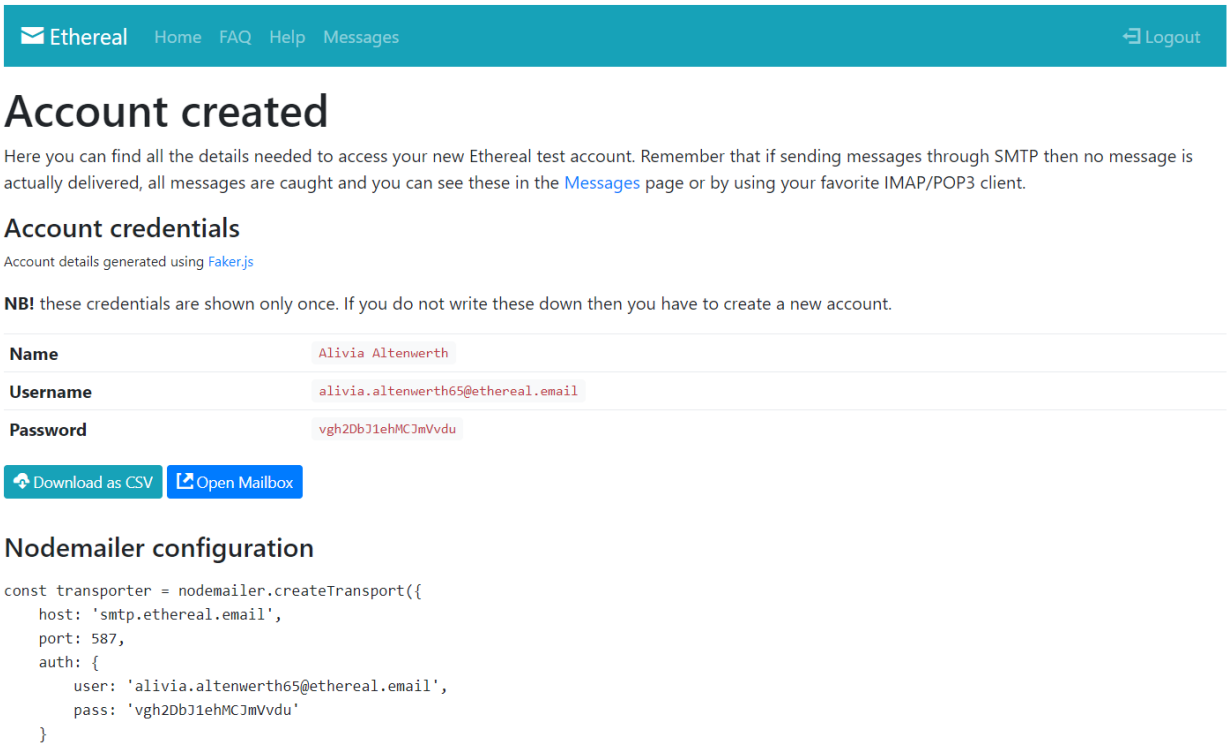
- **IEmailService Interface:** This interface defines the contract for the email service, specifying methods for sending regular emails and meeting invitations.
- **EmailDto Class:** This class represents the data transfer object (DTO) for email-related information. It includes properties such as recipient email address, email body, subject, type (not utilized in the provided code), and meeting date.

```
namespace project_management_system_api.Email
{
    7 references
    public class EmailDto
    {
        2 references
        public string To { get; set; } = string.Empty;
        5 references
        public string Body { get; set; } = string.Empty;
        4 references
        public string Subject { get; set; } = string.Empty;
        0 references
        public string Type { get; set; } = string.Empty;
        1 reference
        public DateTime MeetingDate { get; set; }

    }
}
```

3. Implementation Details:

- **Dependency Injection:** The `EmailService` class takes an `IConfiguration` object via constructor injection, allowing it to access configuration settings such as email host, username, and password.
- **Constructing MimeMessage:** Within both email sending methods, a new `MimeMessage` object is created to represent the email. Sender, recipient, subject, and body are configured accordingly.
- **SMTP Connection and Authentication:** The `SmtpClient` is used to establish a connection with the SMTP server specified in the configuration. Authentication is

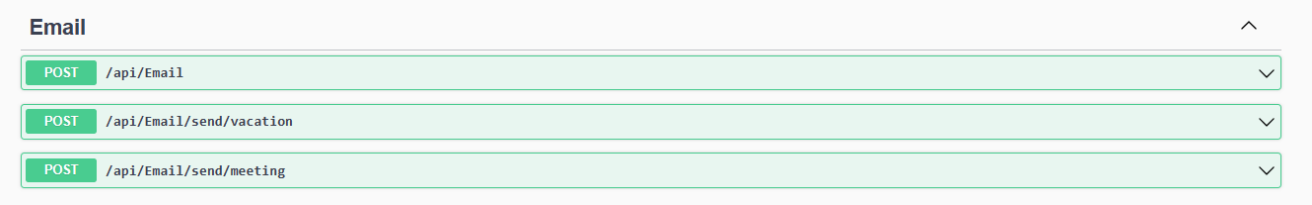
performed using the provided email username and password.



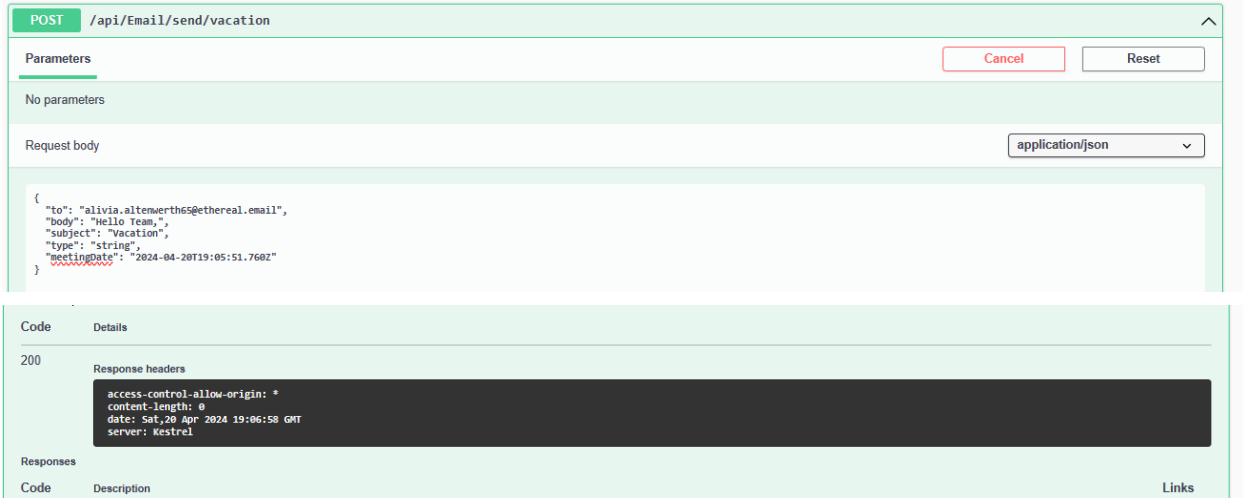
- **Sending Email:** Once the SMTP connection is established and authenticated, the `Send` method of the `SmtpClient` is used to send the constructed email message.
- **Meeting Invitation:** In the `SendMeetingEmail` method, additional information about the meeting, specifically the meeting date, is appended to the email body.

```
namespace project_management_system_api.Services.EmailService
{
    2 references
    public class EmailService : IEmailService
    {
        private readonly IConfiguration _config;

        0 references
        public EmailService(IConfiguration config) {
            _config = config;
        }
        3 references
        public void SendEmail(EmailDto request)
        {
            var email = new MimeMessage();
            email.From.Add(MailboxAddress.Parse(_config.GetSection("EmailUserName").Value));
            email.To.Add(MailboxAddress.Parse(request.To));
            email.Subject = request.Subject;
            email.Body = new TextPart(TextFormat.Html) { Text = request.Body };

            using var smtp = new SmtpClient();
            smtp.Connect(_config.GetSection("EmailHost").Value, 587, SecureSocketOptions.StartTls);
            smtp.Authenticate(_config.GetSection("EmailUserName").Value, _config.GetSection("EmailPassword").Value);
            smtp.Send(email);
            smtp.Disconnect(true);
        }
```

```
    2 references
    public void SendMeetingEmail(EmailDto request)
    {
        var email = new MimeMessage();
        email.From.Add(MailboxAddress.Parse(_config.GetSection("EmailUserName").Value));
        email.To.Add(MailboxAddress.Parse(request.To));
        email.Subject = request.Subject;
        email.Body = new TextPart(TextFormat.Html) { Text = request.Body + " The meeting will be held on " + request.MeetingDate };

        using var smtp = new SmtpClient();
        smtp.Connect(_config.GetSection("EmailHost").Value, 587, SecureSocketOptions.StartTls);
        smtp.Authenticate(_config.GetSection("EmailUserName").Value, _config.GetSection("EmailPassword").Value);
        smtp.Send(email);
        smtp.Disconnect(true);
    }
}
```



- Sending an Email Vacation



- sigin in as hilda and sending an email to alivia