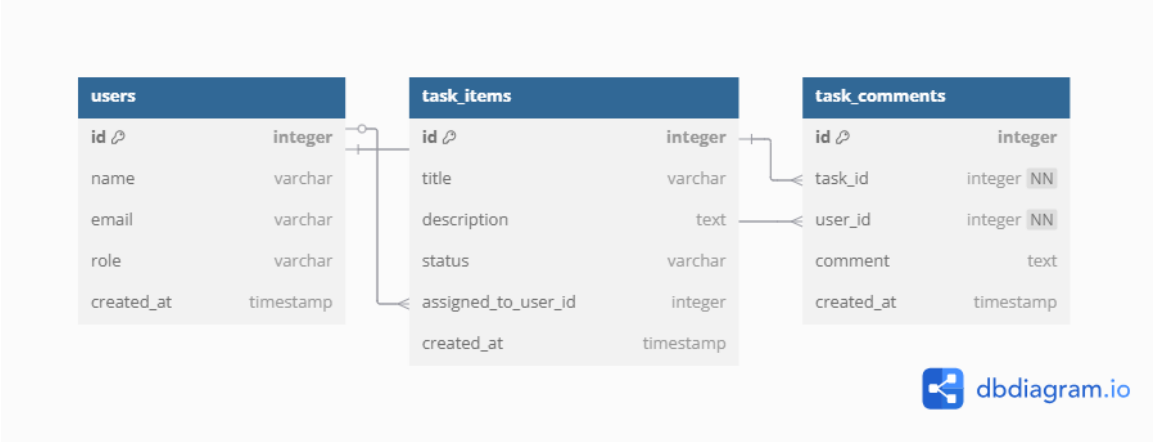
Task Management API Documentation

API Overview

The Task Management API allows users to manage tasks with the ability to:
- Create tasks
- Get tasks by ID
- Get all tasks

The API is built using ASP.NET Core with a SQL Server Express database.

ER DIAGRAM :



Models

1. Task Model

```
public class Task
{
   public int Id { get; set; }
   public string Title { get; set; }
   public string Description { get; set; }
```

```
    public int AssignedTo { get; set; }  // User ID who is assigned the task
}
```

2. TaskComment Model

```
public class TaskComment
{
    public int Id { get; set; }
    public int TaskId { get; set; }  // The task being commented on
    public int UserId { get; set; }  // The user who made the comment
    public string CommentText { get; set; }
}
```

3. User Model

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public Role Role {get; set;}
}
```

Services

TaskService

```
public class TaskService
{
    private readonly DbContext _dbContext;

    public TaskService(DbContext dbContext)
    {
        _dbContext = dbContext;
    }

    public async Task<Task> GetTask(int id)
    {
        return await _dbContext.Tasks.FirstOrDefaultAsync(t => t.Id == id);
    }
```

```
    public async Task<List<Task>> GetAllTasks()
    {
        return await _dbContext.Tasks.ToListAsync();
    }

    public async Task<Task> CreateTask(Task task)
    {
        if (task == null)
            throw new ArgumentNullException(nameof(task));

        _dbContext.Tasks.Add(task);
        await _dbContext.SaveChangesAsync();
        return task;
    }

    public async Task<TaskComment> AddComment(int taskId, TaskComment comment)
    {
        var task = await _dbContext.Tasks.FirstOrDefaultAsync(t => t.Id == taskId);
        if (task == null)
            throw new Exception("Task not found.");

        _dbContext.TaskComments.Add(comment);
        await _dbContext.SaveChangesAsync();
        return comment;
    }
}
```

Controllers

TaskController

```
[ApiController]
[Route("api/[controller]")]
public class TaskController : ControllerBase
{
    private readonly TaskService _taskService;

    public TaskController(TaskService taskService)
    {
        _taskService = taskService;
    }
```

```
  [HttpGet("{id}")]
  public async Task<ActionResult<Task>> GetTask(int id)
  {
    var task = await _taskService.GetTask(id);
    if (task == null)
      return NotFound();
    return Ok(task);
  }

  [HttpGet]
  public async Task<ActionResult<List<Task>>> GetAllTasks()
  {
    var tasks = await _taskService.GetAllTasks();
    return Ok(tasks);
  }

  [HttpPost]
  public async Task<ActionResult<Task>> CreateTask(Task task)
  {
    var createdTask = await _taskService.CreateTask(task);
    return CreatedAtAction(nameof(GetTask), new { id = createdTask.Id }, createdTask);
  }

  [HttpPost("{taskId}/comments")]
  public async Task<ActionResult<TaskComment>> AddComment(int taskId,
TaskComment comment)
  {
    var createdComment = await _taskService.AddComment(taskId, comment);
    return CreatedAtAction(nameof(AddComment), new { taskId = taskId, commentId =
createdComment.Id }, createdComment);
  }
}
```

Dockerization

Dockerfile

```
# Use the official image as a parent image
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
```

```
EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["TaskManagementAPI/TaskManagementAPI.csproj", "TaskManagementAPI/"]
RUN dotnet restore "TaskManagementAPI/TaskManagementAPI.csproj"
COPY . .
WORKDIR "/src/TaskManagementAPI"
RUN dotnet build "TaskManagementAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "TaskManagementAPI.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "TaskManagementAPI.dll"]
```

docker-compose.yml

```
version: '3.8'
services:
 task-management-api:
   build: .
   ports:
    - "5000:80"
   environment:
    - ASPNETCORE_ENVIRONMENT=Development
   depends_on:
    - db

 db:
   image: mcr.microsoft.com/mssql/server:2019-latest
   environment:
    - ACCEPT_EULA=Y
    - SA_PASSWORD=yourStrong(!)Password
   ports:
    - "1433:1433"
```

Unit Testing

Unit tests are provided in the `TaskServiceTests.cs` file. Tests cover positive and negative cases, such as task retrieval and task creation.

How to Run the API

1. Clone or download the repository.
2. Open a terminal and navigate to the project directory.
3. Build the project using the following command:
   ```bash
   dotnet build
   ```
4. Run the API with:
   ```bash
   dotnet run
   ```

docker build -t taskmanagement-api .

docker run -d -p 8080:5270 --name task-api taskmanagement-api

http://localhost:8080/swagger

   The API will be available at `http://localhost:5000`.

5. To run the unit tests:
   ```bash
   dotnet test
   ```