# Project Management API

**Project Management API** built with **.NET 7** following **Clean Architecture** and **CQRS** pattern. This API provides complete project and task management functionality with advanced features like real-time updates, background jobs,and JWT authentication.

# Tech Stack

- **.NET 7** (LTS)-**ASP.NET Core Web API** - **Entity Framework Core 7** (Code-First)- **SQL Server** -  **Mediator** (CQRS Implementation)-

**JWT Authentication** with Role-Based Authorization- Signal R for Real-time Updates- Hangfire for Background Jobs- Fluent validation for Request Validation- Auto-mapperfor Object Mapping- **Swagger/Open-API**  for Documentation

# Architecture

The solution follows **Clean Architecture** :

ProjectManagementAPI  
├── ProjectManagement.API # Presentation Layer  
├── ProjectManagement.Application # Application Layer (Use Cases)  
├── ProjectManagement.Domain # Domain Layer (Core Business Logic)  
├── ProjectManagement.Infrastructure # Infrastructure Layer (Persistence, Services)  
└── ProjectManagement.Shared # Shared Utilities and DTOs

# Prerequisites

- .NET 7 SDK

- SQL Server (LocalDB or Express)

- Visual Studio 2022 or Visual Studio Code

### **2. Database Setup**

Update the connection string in appsettings.json:

json

{

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=ProjectManagementDB;Trusted\_Connection=true;TrustServerCertificate=true;"

}}

### **3. Apply Database Migrations**

bash

*# Package Manager Console*

Update-Database -Project ProjectManagement.Infrastructure -StartupProject ProjectManagement.API

*# Or via CLI*

dotnet ef database update --project ProjectManagement.Infrastructure --startup-project ProjectManagement.API

### **4. Run the Application**

bash

dotnet run --project ProjectManagement.API

The API will be available at https://localhost:44312/ (or http://localhost:5000)

## **API Documentation**

Once running, access the ****Swagger UI**** at:

https://localhost:44312/swagger

## **Authentication**

The API uses ****JWT Bearer Authentication****. To authenticate:

1.Register a new user: POST /api/Auth/register

2.Login: POST /api/Auth/login

3.Use the returned token in the Authorization header: Bearer {your-token}

### **Sample User Registration:**

json

{

"username": "Ahmed",

"email": "admin@test.com",

"password": "123456",

"firstName": "Ahmed",

"lastName": "Saad",

"role": 1}

****Available Roles:**** 1 = Admin, 2 = ProjectManager, 3 = Developer

### **API Features**

### **Core Functionality**

### User Registration & Authentication

### CRUD Operations for Projects

### CRUD Operations for Tasks

### User-Task Assignment

### Project-User Association

### Pagination, Sorting, and Filtering

### **Advanced Features**

### Real-time Task Updates (SignalR)

### Background Job Processing (Hangfire)

### Daily Email Notifications for Overdue Tasks

### In-Memory Caching for Performance

### Request Validation with Fluent Validation

### Role-Based Authorization

## **Main Endpoints**

| Method | Endpoint | Description | Authorization |
| --- | --- | --- | --- |
| POST | /api/Auth/register | Register new user | Public |
| POST | /api/Auth/login | User login | Public |
| GET | /api/Projects | Get all projects | Authenticated |
| POST | /api/Projects | Create new project | Admin, ProjectManager |
| PUT | /api/Projects/{id} | Update project | Admin, ProjectManager |
| GET | /api/Tasks | Get all tasks | Authenticated |
| POST | /api/Tasks | Create new task | Authenticated |
| PUT | /api/Tasks/{id} | Update task | Authenticated |

### **Hangfire Dashboard**

Access the Hangfire dashboard at /hangfire to monitor background jobs.

## **Testing**

Use the included ****Postman Collection**** for testing all endpoints:

Import ProjectManagementAPi.postman\_collection into Postman

* Set the base URL to your API address
* Start with the registration endpoint to create a user
* Use the login endpoint to get a JWT token
* Test other endpoints with the acquired token