Task Management

**Code**

The code is written in .NET 9 (the latest version), it has Swagger for documentation.

Also, the code has JWT Bearer for Authentication and Entity Framework Core (In Memory Database) for the database.

For Design Patterns I've used UnitOfWork Design Pattern together with Repository Design Pattern.

The services are registered with the Dependency Injection.

**Deployment Suggestion**

**Backend Deployment**:

**SQL**: (MSSQL, PostgreSQL) is used to store project and task-related data.

**Kubernetes Cluster**: The backend API is containerized using **Docker** and deployed within a **Kubernetes cluster** (Azure, AWS, GCP) to provide **load balancing, failover, and automatic scaling**.

**Load Balancer (Nginx Ingress)**: used to distribute incoming requests efficiently across multiple backend pods (configured to automatically scale based on CPU and memory usage).

**Caching (Can be used to optimize performance)**: Rediscan be used to reduce database load and improve response times for frequently accessed data.

**Logging and Monitoring**: ELK (Elastic search, Kibana and Logstash).

**Frontend Deployment**:

The client-side application is built using React, Angular, Vue.Js or any modern framework and is deployed using CDN (Content Delivery Network)

Since the frontend apps are static using CDN will distribute them globally, it can cache files and it's faster because not each request goes to the backend only API requests.

Also, you can scale it globally (the entire world) with low latency.

**Overall Deployment Workflow:**

Using CI/CD Pipelines on code push to build and run tests on the backend code.

Containerize API using Docker, Pushing Image to Container Registry (ECR, Azure Container Registry).

Deploy the API to Kubernetes Pods (using Kubernetes Deployments and Services), in case of a bug (ERROR) we can rollback.