# Online Task Management System

Project Phases:

## Project Planning and Setup:

### ~~Define project requirements and objectives.~~

### Plan the database schema and design using tools like Microsoft SQL Server Management Studio (SSMS) or Azure Data Studio.

* + 1. Defined the project database entity requirements in Project Objective and Requirements document.
    2. Created an Entity-Relationship Diagram to Visualize Data Flow, Normalization, Data Types, Constrains, and Indexes using draw.io. ERD diagram saved in repository.

### Set up the development environment, including Visual Studio 2019 for ASP.NET Core development and Azure Portal for Azure resource management.

### Create a project folder structure using a version control system like Git for code management.

## 2. Front-End Development:

### HTML: Create HTML templates for different views and components using VS Code.

### CSS: Style the HTML templates using CSS preprocessors like SASS or LESS for maintainability.

### Angular: Develop Angular components and modules using Angular CLI to scaffold the project.

### Implement responsive design using frameworks like Bootstrap or CSS Grid to ensure the application's responsiveness.

## 3. Back-End Development:

### Build ASP.NET Core MVC controllers and models to handle data and business logic, using Visual Studio.

### Implement Entity Framework Core for database operations and data modeling, defining entities and relationships.

### Set up user authentication and authorization using ASP.NET Identity, configuring roles and policies.

### Develop APIs for Vue.js components to communicate with the back end, using RESTful principles and ASP.NET Core Web API.

## 4. Database Setup:

### Create the SQL Server database using Microsoft SQL Server Management Studio or Azure Portal for Azure SQL Database.

### Set up the necessary tables and relationships based on the database design created earlier.

### Implement data seeding for initial testing and development using Entity Framework Core migrations and seed data.

## 5. User Authentication:

### Configure user registration and login functionality using ASP.NET Identity, integrating it with the front-end.

### Implement user roles and permissions using Identity roles, assigning roles to users as needed.

### Ensure secure access to user-specific data by implementing role-based access control (RBAC) in controllers and APIs.

## 6. Angular Integration:

### Integrate Angular components into the HTML templates.

### Use Angular directives and data binding to render dynamic content and handle user interactions.

### Implement client-side routing using Angular Router for a seamless single-page application (SPA) experience.

## 7. Testing:

### Perform unit testing for both front-end (Angular) and back-end (ASP.NET) components using testing libraries like Jasmine and Karma.

### Conduct integration testing to ensure all parts of the application work together smoothly, including API testing.

### Test user authentication, data retrieval, and data manipulation to verify functionality.

## 8. Documentation:

### Create documentation for the project, including a user guide and developer documentation using tools like Markdown.

### Document the database schema and data flow within the application using Entity Relationship Diagrams (ERDs).

### Provide clear instructions on how to set up and deploy the application on platforms like Azure App Service.

## 9. Deployment:

### Deploy the application to Microsoft Azure or your chosen hosting platform using Azure DevOps or other CI/CD pipelines.

### Configure Azure resources, such as Azure App Service for hosting the web application and Azure SQL Database for data storage.

### Set up domain and DNS settings if applicable to make the application accessible online.

### Perform a final round of testing in the production environment to ensure everything works as expected.

## 10. Maintenance and Optimization:

### Monitor the application's performance and user feedback using tools like Azure Application Insights.

### Continuously improve the codebase, fix bugs, and enhance features based on user feedback and business needs.

### Optimize database queries and server-side processes for better performance, considering tools like SQL Profiler.

### Stay up-to-date with security patches and updates for all components, including libraries, frameworks, and server configurations.