Project Overview

The **Task Management System** is a web application that allows users to create, manage, and track tasks efficiently. It enables users to categorize tasks, assign due dates, update task statuses, and receive real-time notifications. The system also provides a dashboard for visualizing upcoming tasks and tasks that need attention.

This system incorporates several modern web development technologies, such as **ASP.NET Core MVC**, **ASP.NET Core Web API**, **Entity Framework Core**, **ASP.NET Core Identity**, **SignalR** for real-time notifications, and **Caching** for performance optimization.

Key Features

1. **User Registration and Authentication**:
    - Users can register, log in, and manage their accounts using **ASP.NET Core Identity**.
    - Role-based authentication (admin, user) allows for different permissions for task management.
2. **Task Management**:
    - Users can create tasks, update task details (title, description, due date, and status), and delete tasks.
    - Tasks can be categorized (e.g., "Work", "Personal", etc.) for easy management.
    - Admin users can assign tasks to specific users.
3. **Real-time Notifications**:
    - Users receive real-time updates on task changes (e.g., task assignment, status updates).
    - Notifications are sent using **SignalR**, ensuring that users are notified immediately when important changes occur.
    - Admins and assigned users are notified when tasks are due or updated.
4. **Dashboard**:
    - Users can view a dashboard displaying upcoming tasks, overdue tasks, and tasks by category.
    - The dashboard can be filtered to show tasks that are in progress, completed, or overdue.
    - Catching is used to store frequently accessed task data to improve performance.

5. **Search and Filtering**:
    - Users can search for tasks by title, category, or due date.
    - Filtering tasks by status (e.g., "Pending", "In Progress", "Completed") helps users focus on their most urgent tasks.
6. **API Endpoints**:
    - The system provides a set of RESTful APIs for managing tasks and users.
    - The API supports operations such as creating tasks, updating tasks, retrieving tasks, and assigning tasks to users.

Technologies Used

- **ASP.NET Core MVC**: For building the web front-end of the task management system.
- **ASP.NET Core Web API**: To expose backend services that can be consumed by the MVC front-end or other clients.
- **Entity Framework Core**: For interacting with the database and performing CRUD operations on tasks, users, and notifications.
- **ASP.NET Core Identity**: For managing user registration, authentication, and authorization.
- **SignalR**: For real-time communication to push task notifications to the UI.
- **Caching (In-memory cache)**: To store frequently accessed task data and improve performance.
- **SQL Server**: For data storage and management.

System Architecture

1. **Frontend (ASP.NET Core MVC)**:
    - The front-end is a web interface where users interact with the system.
    - It displays a task dashboard, task creation forms, and real-time task updates.
2. **Backend (ASP.NET Core Web API)**:
    - The API serves as the intermediary between the front-end and the database.
    - It handles task CRUD operations, user authentication, and notifications.
3. **Real-time Communication (SignalR)**:
    - SignalR is used to broadcast real-time updates to users about task changes.

    - For example, when an admin assigns a new task to a user, SignalR pushes a notification to the assigned user in real-time.

4. **Database (Entity Framework Core)**:
    - Entity Framework Core interacts with a relational database (e.g., SQL Server) to persist task, user, and notification data.
5. **Caching (In-memory caching)**:
    - Frequently accessed task data (such as tasks listed on the dashboard) is stored in memory to reduce database queries and improve performance.

User Stories

1. User Registration and Login

- **As a new user**, I want to register for an account so that I can use the task management system.
- **As a registered user**, I want to log in securely using my credentials.
- **As an admin**, I want to manage user roles and permissions (e.g., assigning tasks, viewing all users' tasks).

2. Task Creation and Management

- **As a user**, I want to create tasks with specific details (title, description, due date) and categorize them for easy tracking.
- **As a user**, I want to update the status of tasks (e.g., from "To Do" to "In Progress").
- **As a user**, I want to mark tasks as completed once done.
- **As an admin**, I want to assign tasks to users and monitor progress.

3. Real-time Notifications

- **As a user**, I want to receive real-time notifications when:
    - A task is assigned to me.
    - A task's status changes.
    - A task's due date is approaching or overdue.
- **As a user**, I want to view notifications in the UI without having to refresh the page.

4. Task Dashboard and Filtering

- **As a user**, I want to view my upcoming, in-progress, and completed tasks on the dashboard.

- **As a user**, I want to filter tasks by category, due date, and status.
- **As an admin**, I want to view all users' tasks and manage task assignments.

Database Design

Entities:

1. **User**:
    - Id (Primary Key)
    - Username
    - Email
    - PasswordHash
    - Role (Admin, User)
2. **Task**:
    - Id (Primary Key)
    - Title
    - Description
    - DueDate
    - Status (Pending, In Progress, Completed)
    - Category
    - AssignedTo (Foreign Key referencing User)
3. **Notification**:
    - Id (Primary Key)
    - UserId (Foreign Key referencing User)
    - TaskId (Foreign Key referencing Task)
    - Message
    - DateTime
4. **Category**:
    - Id (Primary Key)
    - Name

APIs and Endpoints

- **POST** /api/auth/register: Register a new user.
- **POST** /api/auth/login: Log in a user.
- **GET** /api/tasks: Get all tasks (with optional filtering by user, status, etc.).
- **GET** /api/tasks/{id}: Get details of a specific task.
- **POST** /api/tasks: Create a new task.
- **PUT** /api/tasks/{id}: Update an existing task.
- **DELETE** /api/tasks/{id}: Delete a task.
- **POST** /api/notifications: Push a notification to a user (triggered by task assignment or update).

Caching Strategy

- **Task Caching**: Frequently accessed task data (e.g., tasks that are due soon, in progress, or completed) will be cached in memory to reduce database load.
- **Category Caching**: Task categories can be cached to speed up task listing and dashboard rendering.

Real-time Notifications with SignalR

- **SignalR Hub**: A SignalR hub will manage connections between clients (users) and push notifications for tasks.
    - Notifications are pushed in real-time whenever there is an update to a task (e.g., status change, new assignment).
    - Each client will listen to notifications and display them dynamically without requiring a page refresh.