**Problem Statement: Task Management Application**

**Objective**

Develop a web-based Task Management Application using ASP.NET Core MVC to allow users to create, view, edit, and delete tasks without a database, using in-memory storage. The application must incorporate model binding, data validation, controllers, action methods, and Razor views to ensure a user-friendly interface and data integrity.

**Requirements**

**1. Functionality**

* Users can view a list of all tasks.
* Users can create new tasks with a title, description, due date, and priority level.
* Users can edit existing tasks to update their details.
* Users can delete tasks.
* Data is stored in an in-memory list (e.g., a static List<TaskItem>) to avoid database dependencies.

**2. Model Binding**

* Automatically map form inputs to a TaskItem model for create and edit operations.

**3. Data Validation**

* Enforce validation on the TaskItem model using data annotations:
  + Title: Required, 3-100 characters.
  + Description: Optional, maximum 500 characters.
  + DueDate: Required, valid date format.
  + Priority: Required, integer between 1 and 5.
* Display validation errors for invalid input.
* Support both server-side and client-side validation.

**4. Controllers and Action Methods**

* Create a TasksController with action methods for:
  + Index: Display task list (GET).
  + Create: Show create form (GET) and handle submission (POST).
  + Edit: Show edit form (GET) and update task (POST).
  + Delete: Show confirmation page (GET) and delete task (POST).
* Use [ValidateAntiForgeryToken] on POST actions.
* Return views or redirects appropriately.

**5. Views**

* Use Razor views (.cshtml) for:
  + Index.cshtml: Table of tasks with title, description, due date, priority, and action links (Edit, Delete).
  + Create.cshtml: Form for task creation with validation messages.
  + Edit.cshtml: Form for editing tasks.
  + Delete.cshtml: Confirmation page for deletion.
* Use tag helpers (asp-for, asp-action) for binding and navigation.
* Include client-side validation scripts.

**6. Constraints**

* No database; use a static in-memory list.
* Lightweight and suitable for prototyping.
* Handle edge cases (e.g., non-existent task IDs).

**7. User Experience**

* Clean, responsive UI with Bootstrap.
* Clear error messages for invalid inputs.
* Navigation links for ease of use.
* Redirect to task list after actions.

**Deliverables**

* ASP.NET Core MVC application with:
  + TaskItem model with validation.
  + TasksController with CRUD actions.
  + Razor views for all operations.
  + In-memory storage(Collections)