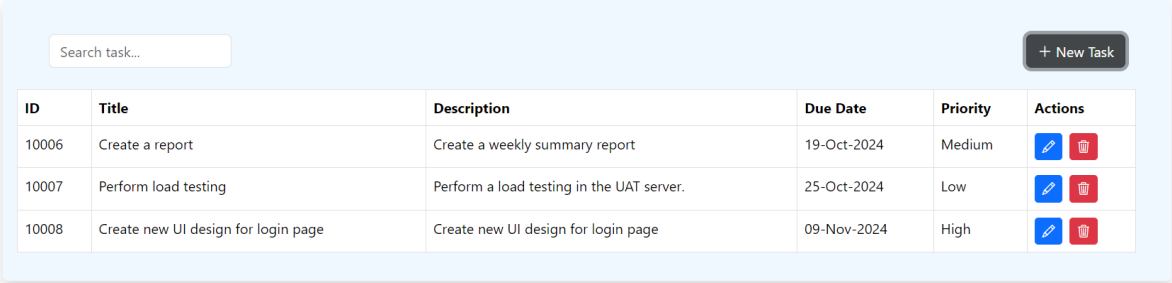
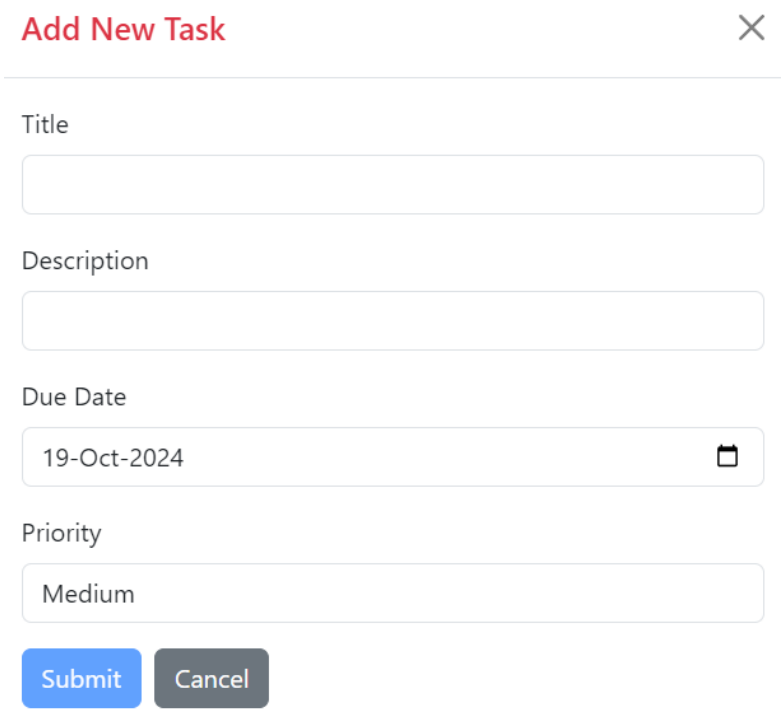
Unicom TIC - Lab work sheet.

Week 19/Oct/2024 - 20/Oct/2024

Develop a **Task Management** application with CRUD (Create, Read, Update, Delete) functionalities using **Angular** for front-end logic and **Bootstrap** for styling and **.Net Core & SQL Server** for back-end.

Sample UI screen of the application





A. Backend Development

1. Create a **.net core web-api** application named **TaskManagerAPI**
2. Install Entity Framework Core and SQL Server provider nuget packages.
3. Create a class named **TaskItem** inside the Models folder with following properties.
    - Id – int (primary key, auto generated)
    - Title – string (required)
    - Description – string
    - DueDate – datetime
    - Priority - string (required)
4. Create a **TaskContext** class in the Data folder and define a DbSet named **Tasks.**
5. Setup all required configuration in the program.cs file for db connection, swagger, CORS policy and etc.
6. Create a new database named '**TaskManager'**
7. Perform the necessary database migration.
8. Create a new Controller named '**TaskItemsController'**
9. Implement following crud functionalities in the above controller.
    - POST: Add a new task.
    - GET: Retrieve all tasks.
    - PUT: Update a task by ID.
    - DELETE: Remove a task by ID.
10. Run the backend project and test the APIs using swagger tool

B. Front-end Development

1. Use the Angular CLI to generate a new project named **task-manager**.
2. Add Bootstrap via npm and configure it in angular.json.
3. Create a TypeScript interface named task.model.ts in the src/app folder and define above mentioned properties for the task.
4. Use Angular CLI to generate the following components:
    - task-list
    - task-add
    - task-edit
5. Use Angular CLI to generate a service called task.service.ts.
6. Implement methods like getTasks(), addTask(), updateTask(), and deleteTask() inside task.service.ts.

7. Set up routes for task listing, adding, and editing in app-routing.module.ts as below.

```
const routes: Routes = [
  { path: '', component: TaskListComponent },
  { path: 'add', component: TaskAddComponent },
  { path: 'edit/:id', component: TaskEditComponent }
];
```

8. Use Bootstrap classes inside task-list.component.html to display a list of tasks.
9. Use Angular's ReactiveFormsModule to create a form in task-add.component.ts.
10. Use the task service's addTask() method when the form is submitted.
11. Add a alert window to confirm the deletion and call the deleteTask() method from the service.
12. Implement the task-edit component for updating the task.
13. Show appropriate success or failure messages to user for various actions.
14. Use date pipe to format the due date as in the list view. Example: **19-Oct-2024**
15. Implement Task Search functionalities using Angular Custom Pipe.