Short Reflective Report

Task Management Application

Name: M.V Pubudu Piyumal Premasiri

University: University of Wolverhampton

Practical Assessment – SoftOne Group

Student ID: 2011034

Degree Title: BSc (Hons) in Computer Science

(Software Engineering)

Submission Date: 05th of July 2024

Contents

Introduction 3
Backend Development and CRUD Operations 3
Frontend Development and Integration 4
Errors Encountered 5
Conclusion 6

Introduction

The task management application project involved developing a backend using .NET Web API and a frontend using Angular. The goal was to create a full-stack application that supports CRUD operations (Create, Read, Update, Delete) for managing tasks. This report reflects on the development process, errors encountered, and the successful implementation of the backend, highlighting the CRUD operations in detail. Additionally, it discusses the technical issues encountered with CORS configuration.

Backend Development and CRUD Operations

The backend development was carried out using .NET Web API. The primary goal was to set up a robust API to handle task management operations, ensuring secure and efficient data transactions. Below are the key steps and implementations for each CRUD operation:

Create (POST):

- **Endpoint**: API/tasks
- **Description**: This endpoint handles the creation of new tasks. It accepts a JSON object containing task details and inserts it into the database.

Read (GET):

- **Endpoint**: API/tasks and API/tasks/{id}
- **Description**: The GET endpoints allow for retrieving all tasks or a specific task by its ID.

Update (PUT):

- **Endpoint**: API/tasks/{id}
- **Description**: This endpoint updates an existing task. It requires the task ID and updated.

Delete (DELETE):

- **Endpoint**: API/tasks/{id}
- **Description**: This endpoint deletes a task by its ID.

Frontend Development and Integration

The front-end was developed using Angular. The main components created were for listing tasks, and handling task creation, updates, and deletions. The integration with the backend API was achieved using Angular's HttpClient module.

1. **Components and Services**:
    - **Task Model**: Defined the structure of a task.
    - **Task Service**: Managed API calls for CRUD operations.
    - **Tasks Component**: Displayed the list of tasks and forms for task manipulation.
2. **Routing**: Set up routing to navigate between different components.
3. **Template Binding**: Used Angular’s two-way data binding to connect the frontend forms with the backend API.

Errors Encountered

1. **Namespace Error**:
    - Error: The type or namespace name 'Edit' does not exist in the namespace 'SoftOne.Pages.Tasks'
    - Solution: Corrected the namespace and class definitions to ensure proper linkage.
2. **Form Binding Issues**:
    - Error: Incorrect binding and validation attributes in Razor pages.
    - Solution: Fixed the binding attributes and ensured the form inputs were correctly linked to the model properties.
3. **CORS Configuration Issue**:
    - Error: PowerShell refused to install the CORS configuration in the ConfigureServices method.
    - Solution: Manually added the CORS policy in the Program.cs file.

Conclusion

The backend development for the task management application was successfully completed with full CRUD operations implemented. The Angular frontend was developed to interact with the backend API, though the integration process encountered a technical issue with CORS configuration, which was manually resolved. The project provided valuable experience in full-stack development, handling database operations, API development, and frontend-backend integration. Despite the challenges, the application is functional and meets the initial project requirements.