Novestra

Full-Stack Upskilling Program

This assignment is designed to help you develop practical, industry-relevant skills in full-stack application development. By building a ToDo application, you will gain hands-on experience with modern technologies, including React for frontend development and .NET Core Web API for backend development.

Key Learning Outcomes:

- Master frontend and backend integration.
- Implement authentication using industry-standard practices.
- Apply Clean Architecture principles for scalable and maintainable solutions.
- Develop testing and error-handling skills.
- Enhance planning and estimation capabilities.

At Novestra, we are committed to empowering our interns to become skilled professionals ready to tackle real-world challenges.

Let’s build something amazing together!

Contents

Objective

The goal of this assignment is to provide hands-on experience in full-stack development by building a Todo Application. This project will cover frontend development using React and backend development using .NET Core Web API with Clean Architecture principles. It will also emphasize planning, estimation, implementation, and demonstrating the completed solution.Scope of the Assignment

Scope of the Assignment

Frontend Development:

- Develop a user-friendly interface using React.
- Include features such as:
    - Adding, editing, deleting, and marking tasks as completed.
    - Filtering tasks based on their status (All, Active, Completed).
    - A dashboard showing task summaries (e.g., completed vs. pending tasks).
- Implement user authentication using Auth0 or a custom JWT solution.
- Address the following React concepts and libraries:
    - Component Design: Create reusable components like TodoList, TodoItem, Filter, and Dashboard.
    - Styling: Use CSS or Tailwind CSS for layout design.
    - Hooks: Leverage hooks like useState, useEffect, and useReducer.
    - Routing: Use react-router-dom for navigation (e.g., routes for Login, Todo Dashboard, etc.).
    - State Management: Use Redux Toolkit or Context API for global state management.
    - HTTP Requests: Use Axios for efficient API communication.

Backend Development:

- Use .NET Core Web API to build a robust backend.
- Implement Clean Architecture principles for scalability and maintainability.
- Design a database schema to store users, tasks, and task statuses using Entity Framework or Dapper.

- API Endpoints:
    - User authentication (register/login/logout).
    - CRUD operations for tasks.
    - Task filtering (by status or date).
- Authentication
    - Implement user authentication using either:
        - Auth0 (preferred for simplicity).
        - Custom JWT-based solution.
- Unit Testing
    - Develop automated unit tests Back end using xUnit or MS Test to ensure code reliability, targeting a code coverage of over 80%
    - Develop automated unit tests for frontend components using Jest or React Testing Library to ensure functionality, stability, and correctness of the UI.
    - Aim for code coverage of over 80%, focusing on critical areas like component rendering, state management, and event handling.
    - Incorporate testing strategies to verify:
        - Proper rendering of components with various props and states.
        - Functionality of user interactions (e.g., button clicks, form submissions).
        - Integration of API calls and their effect on the UI.
        - Use mocking and stubbing to isolate components from external dependencies for reliable and repeatable test
- Best Practices:
    - Utilize Dependency Injection to improve testability and support maintainable code.
    - Integrate exception handling mechanisms to provide a seamless and user-friendly application experience.
    - Conduct SonarQube code analysis to identify and address defects and code smells, ensuring high-quality code standards.

Planning and Estimation

Task Breakdown

- Frontend Tasks:
    - Setting up React project.
    - Implementing task management UI (add, edit, delete, filter).
    - Creating reusable components (TodoList, TodoItem, Dashboard).
    - Setting up routing and authentication.
- Backend Tasks:
    - Setting up .NET Core Web API project.
    - Designing the database schema.
    - Implementing API endpoints.
    - Integrating authentication.
- Testing Tasks:
    - Writing unit tests for APIs and React components.

Estimation:

- Setup React project: 3 hours
- Implement task management UI: 6 hours
- API design and implementation: 8 hours
- Authentication integration: 4 hours

Project Plan

- Prepare a Gantt Chart or timeline mapping tasks to deadlines.

Task	Duration	Start Date	End Date
Setup React Project	4 hours	Day 1	Day 1
Implement Routing	6 hours	Day 1	Day 2
Develop Product Listing Page	8 hours	Day 2	Day 3
Setup .NET Core Web API	4 hours	Day 1	Day 1
Implement Authentication (JWT)	12 hours	Day 3	Day 4
Write Unit Tests	8 hours	Day 5	Day 5

Demonstration Requirements

Demo Flow

1. Present the project architecture, API design, and key features.
2. Showcase functionalities:
    - User authentication (register/login/logout).
    - Adding, editing, and deleting tasks.
    - Filtering tasks by status.
3. Provide a technical walkthrough of:
    - React components and state management.
    - API endpoints and authentication mechanism.

Q&A

Be prepared to answer questions about the design decisions and implementation.

Deliverables

1. User Stories:
    - Example: "As a user, I want to filter tasks by status so I can view only completed tasks."
2. Database Design:
    - Submit the relational database schema with ER diagrams.
3. API Design:
    - Provide RESTful API endpoint details.
4. Source Code:
    - Upload to a GitHub repository with clear documentation.
5. Estimation Plan:
    - Submit a document detailing task breakdown and project timeline.
6. Demo Presentation:
    - Prepare a presentation highlighting architecture, planning, and features.

Evaluation Criteria

1. Functionality:
    - The app must handle task CRUD operations and filtering.
2. Code Quality:
    - Follow Clean Code and SOLID principles.
3. Planning and Execution:
    - Adherence to the estimation plan.
4. Testing:
    - Include unit tests for key components and APIs.
5. Authentication:
    - Proper implementation of user authentication.

Submission

- GitHub Repository: Include all source code and documentation.
- Planning Document: Submit the task breakdown and timeline.
- Demo Video or Live Presentation: Record or present the demo live.