**Practical Test: Building a Simple Angular and .NET Application**

Objective: Create a basic web application that allows users to view a list of items and add new items. The application should have a front-end in Angular and a back-end in .NET.

**Project Requirements:**

1. Angular Frontend:

* Create an Angular project using Angular CLI.
* Develop a component that displays a list of items.
* Implement a form to add new items to the list.
* Use Angular services to manage the data (e.g., items array).
* Style the application using CSS or a CSS framework (e.g., Bootstrap).

1. .NET Backend:

* Create a .NET Web API project (ASP.NET Core or ASP.NET Framework).
* Implement endpoints for retrieving the list of items and adding new items.
* Use a data store, such as an in-memory list, to manage the items.
* Implement proper error handling for API requests.
* Use appropriate HTTP methods and status codes.

**Instructions:**

1. Set up a development environment with Angular and .NET, ensuring all necessary tools are installed (Angular CLI, .NET SDK, etc.).
2. Create the Angular application and design the user interface according to the requirements.
3. Develop the .NET Web API for managing the data.
4. Ensure that the frontend and backend can communicate correctly through HTTP requests.
5. Test the application to make sure it displays the list of items and allows the addition of new items.
6. Provide clear documentation on how to build and run the application.

**Bonus Question: Concurrent Data Processing**

Objective: Extend the item list application to perform concurrent data processing on items, using multithreading for parallel data processing.

**Task Description:**

Extend the .NET backend and Angular frontend to simulate concurrent data processing for each item in the list. Implement a calculation of the factorial of a number on each item concurrently. The row number of each item can be used as input for the factorial calculation.

**Instructions:**

1. Create a button in Angular to call a .NET Function to perform the calculation of the factorial number for each line.
2. Implement multithreading function in the .NET backend to process items concurrently. Each thread should handle a separate item's data processing.
3. Update the Angular frontend to trigger concurrent data processing and display the results as each item's data processing is completed.
4. Ensure that the main application remains responsive during concurrent data processing.
5. Implement proper error handling for any issues that may arise during the processing.