### **Design Challenge: Actuarial Model Web Application**

#### **Objective:**

Design a web application using ReactJS and .NET that wraps an actuarial model microservice written in Python. The model code makes use of a pipeline architecture where separate tasks consume and produce data. Generally, a single task in a specific pipeline is run at a time. The .NET backend will orchestrate the pipelines using Azure Storage queues and pass user information from the UI to the tasks. The focus of this challenge is on the design and architecture of the UI layer, but it will count in your favor if your design is backed up with code snippets.

#### **Requirements:**

1. **Backend (ASP.NET Core):**
   * Design the necessary endpoints to:
     + Return the UI schema and data schema.
     + Manage the UI state.
     + Start a pipeline with user data.
     + Return the status of the pipeline.
2. **Frontend (ReactJS):**
   * **Components:**
     + Design a home page with panels for user input and displaying results.
     + Implement a wizard to guide the user through multiple steps, running pipelines and updating the UI based on the pipeline status.
     + Ensure the UI is dynamic and can render based on the configuration provided by the backend.
   * **State Management:**
     + Design how the UI schema and data schema will be fetched from the backend.
     + Plan how the UI state will be stored and updated in the backend.
     + Outline how user input will be handled and passed to the backend to start pipelines.
     + Describe how the UI will be updated based on the pipeline status and results.
3. **Pipeline Orchestration:**
   * Design the use of Azure Storage queues to manage the pipeline tasks.
   * Plan how user data from the UI will be passed to the pipeline tasks.
   * Describe how the UI will be updated based on the pipeline status and results.

#### **Deliverables:**

* **Architecture Diagram:** A high-level diagram showing the overall architecture of the application, including the backend, frontend, and pipeline orchestration.
* **Component Design:** Detailed design of the React components, including how they interact with each other and the backend.
* **State Management Plan:** A plan for managing the UI state, including how it will be stored, updated, and synchronized with the backend.
* **Pipeline Orchestration Design:** A detailed design of how the pipelines will be orchestrated using Azure Storage queues, including how user data will be passed and results will be handled.
* **Sample code:** Code snippets demonstrating key design decisions.

#### **Evaluation Criteria:**

* **Design Quality:** Clarity, completeness, and feasibility of the design.
* **Architecture:** Proper separation of concerns and use of design patterns.
* **Functionality:** How well the design meets the requirements.
* **UI/UX:** Intuitive and user-friendly interface design.
* **Scalability:** Consideration of scalability and performance in the design.