Problem Statement: Create a web application with NextJS frontend and AdonisJs backend. At a high level the web application provides UI functionality to monitor and manage clusters. The backend is responsible for providing cluster information and related metrics using a data source of your choice (database/file-based-storage). The frontend provides functionality for user interaction.

Requirements:

- **Frontend**: Implement the UI using Recharts, Next.js, Tailwind. Attention to detail is important, focusing on aspects like alignment, sizing, and spacing between elements. While pixel perfection isn't necessary, we'll notice these details. Ideally this would be a separate project that invokes the other project the Backend API using Axios to get/put data.
    - **Design File**: see attached “Performance_Metrics.png”, “Edit_Snapshot_Policy_Disabled_Locking.png” and “Edit_Snapshot_Policy_Enabled_Locking.png”
    - **Side Bar:** Show application icon and Cluster Name at the header part. The main part shows two functions: “Performance Metrics” and “Edit Snapshot Policy”. The footer part shows the user information (avatar, user name). Switch user will update the whole page to show the cluster dashboard associated with the user.
    - **Time Series Graphs:** Implement a dashboard displaying time series data such as IOPS and Throughput using Recharts for data visualizations. Use an API endpoint you developed to get you the needed data. The API could get the data from a local database or even a csv or json file and return to the UI project.
    - **Revisit Snapshot Policy Schedule with Snapshot Locking:** Develop a form-based front-end interaction to set up a snapshot policy schedule with snapshot locking. Leverage the Get and Put endpoints from your API.
- **Backend**: Implement API endpoints needed by the UI as a separate project using AdonisJS. Use uuid as the identifying key for the cluster. Just one or two cluster’s worth of data surfaced by API is enough.
    - ***GET*** endpoint for retrieving the time series data
    - ***GET*** and ***PUT*** for snapshot policy config
- **Tooling**: Next.js, Tailwind, Recharts, TypeScript, AdonisJS
- **Engineering Practices**: Aim to follow good engineering practices, such as Test-Driven Development (TDD), good documentation.

Evaluate take home exercise:

Technical Proficiency:

- Code Quality: We'll assess the cleanliness, readability, and organization of your code. Pay attention to coding standards and best practices.
- Correctness: Ensure that your UI components and API endpoints function as expected, with no errors or bugs.
- Scalability: Consider how your solution would scale in terms of handling larger datasets or more complex functionality.

UI Development:

- Aesthetics: We'll evaluate the visual design and layout of your UI components. Pay attention to typography, color schemes, and overall aesthetics. See the attached CSS files “Performance_Metrics.css”, “Edit_Snapshot_Policy_Disabled_Locking.css” and “Edit_Snapshot_Policy_Enabled_Locking.css”
- Responsiveness: Ensure that your UI is responsive and works well across different devices and screen sizes.
- Testing: Unit tests and end to end tests help identify bugs and regressions.

API Development:

- RESTful Design: Evaluate the design and implementation of your API endpoints. Ensure that they follow RESTful principles and are intuitive to use.
- Data Handling: Assess how effectively you handle data retrieval and manipulation within your API. Consider factors such as performance and data integrity.
- Error Handling: Ensure that your API provides meaningful error messages and handles edge cases gracefully.

Problem-Solving Skills:

- Innovative Solutions: We'll look for innovative approaches to solving the given problems, including any creative features or functionalities you implement.
- Adaptability: Consider how well you adapt to new frameworks and technologies, as indicated by your choices in the exercise.
- Collaboration: If applicable, demonstrate your ability to work collaboratively by seeking feedback, discussing design decisions, and incorporating suggestions from others.

Documentation and Communication:

- Readme: Provide clear and concise documentation in your repository's README file, including instructions for running your projects locally.
- Code Comments: Include relevant comments throughout your code to explain your thought process and any complex logic.
- Communication Skills: If selected for further interviews, be prepared to discuss your design and implementation decisions in detail.