Technical Test for Fullstack Developer

Objective: To assess a Fullstack Developer's proficiency in creating a backend service and a stylish frontend using the specified libraries and technologies.

Backend Development (Node.js)

- Task 1: Setting Up the Backend (40 points)
  - Use Node.js and the library itrm-tools to create a backend service. Here's a test repository as documentation.
  - Implement an authentication system for users, including sign-in and registration functionalities.
  - Store user data securely, using any kind of tools that the user feels convenient.
  - Implement JWT token generation and verification for authenticated user access.

- Task 2: Product Endpoint (40 points)
  - Create a protected route named "products" that is only accessible to authenticated users.
  - Implement middleware to verify JWT tokens for this route.
  - Make an internal HTTPS call to the endpoint https://dummyjson.com/carts.
  - Return the response of the internal call as JSON.

- Task 3: Error Handling and Documentation (20 points)
  - Implement proper error handling for all possible scenarios.
  - Document the API endpoints, usage, and setup for future developers. You can use any automated tools that you find for this endeavor.

Frontend Development (React)

You can use the following figma as resource for your implementation:

- Task 1: Setting Up the Frontend (30 points)
  - Create a stylish React application that includes the necessary dependencies and project structure.
  - Implement routing using 'react-router' or similar libraries for a smooth user experience.
  - Develop a Login view with user authentication functionality.

- Task 2: Registration View (20 points)
  - Create a Registration view that allows new users to sign up.
  - Validate user input and provide meaningful feedback for errors.

- Task 3: Product Query View (30 points)
  - Create a view for querying products, which is only accessible to authenticated users.
  - Implement a user-friendly interface to send a request to the backend for product data.
  - Display the received data in a clear and organized manner.
  - Allow the user to search for a specific product.

- Task 4: User Authentication (20 points)
  - Implement a user authentication system using JWT tokens.
  - Ensure that users are redirected to the login page when not authenticated.

- Task 5: Error Handling and Styling (10 points)
  - Implement error handling and provide feedback to users in case of invalid input or failed requests.
  - Apply CSS/Styles to make the frontend visually appealing and user-friendly.

Bonus (10 points)

- Implement unit tests for critical parts of both the backend and frontend.
- Use 'axios' or a similar library for HTTP requests (do any of the libraries offered contain a similar functionality?).
- Implement state management for the frontend (e.g., 'Redux' or 'Context API').

Submission and Evaluation:

- The developer is required to submit both the backend and frontend code via a Git repository.
- The backend, frontend and any other services (for example databases) should be dockerized.
- The code should be well-documented and follow best practices.
- Evaluation will consider code quality, functionality, and adherence to the specified requirements.
- This technical test should provide a comprehensive assessment of a Fullstack Developer's skills in building a backend service with authentication, making internal HTTP calls, and creating a stylish React frontend with user authentication features.