REACT.JS - ASSESSMENT

This test is divided into 2 sections the first section is purely theoretical, to understand if you have the basic and advance concept of react.js. The second section are three tests, they divided in order of difficulty, please complete at least one of the 3 examples, but ideally,we would love to see all three or two out of three completely done.

We require you to complete this assessment within 3 working days,

THEORY OF REACT.JS

Part 1 - Basic Knowledge:

1. What is React.js?
2. What are the key features of React.js?
3. What is JSX?
4. What is a virtual DOM?
5. What is the difference between props and state?
6. What is the role of Redux in a React.js application?
7. What is the purpose of React Router?

Part 2 - Code Implementation:

1. Create a simple React component that displays "Hello World!" as text.
2. Create a form component with input fields for name and email.
3. Implement a React component that displays a list of items.
4. Use React Router to create a multi-page application with a navigation menu.

Part 3 - Advanced Concepts:

1. Explain the concept of React hooks.
2. What are higher-order components in React and how are they useful?
3. Explain the difference between server-side rendering and client-side rendering in React.
4. How would you optimize the performance of a React application?
5. What is the role of context in React?

Part 4 - Real-world Application:

1. Describe a React.js project you have worked on and your contribution to it.
2. How did you ensure the project was maintainable and scalable?
3. What challenges did you face while working on the project, and how did you overcome them?

BASIC TEST

## Building a To-Do List

Create a to-do list application using React.js that allows the user to add, remove and complete tasks. The user interface should be responsive and intuitive.

### Requirements

The application should have the following features:

* Ability to add new tasks to the list
* Ability to mark tasks as completed
* Ability to remove tasks from the list
* Display the total number of tasks remaining
* Store tasks in the browser's local storage, so that they persist after the page is reloaded

### Guidelines

* Use Create React App to set up the project
* Use functional components and hooks (e.g. useState, useEffect)
* Use CSS modules or styled components for styling
* Use a state management library such as Redux or Context API to manage the application state
* Use Git for version control and commit frequently with clear commit messages
* Write clear and concise code, and comment where necessary
* Host the application on a public Git repository and provide the link to the repository

### Bonus Points

* Add drag and drop functionality for tasks
* Allow tasks to be edited after they have been added
* Add filtering functionality to show completed tasks only or all tasks
* Add search functionality to search for tasks by name

### Evaluation

The coding challenge will be evaluated based on the following criteria:

* Code quality: clear, concise and maintainable code
* Functionality: the application should meet all the requirements listed above
* User experience: the application should be intuitive and responsive
* Bonus points: the implementation of any bonus points features will be considered when evaluating the submission

Good luck!

MID LEVEL TEST

## Challenge: Create a Submit Information Form

Create a form using React.js that allows users to submit information, such as their name, email address, and phone number. Upon submission, the form should display a success message and clear the fields. The form should also include client-side validation to ensure that all required fields are filled in and that the email address is valid.

### Requirements

The form should have the following features:

* Fields for name, email address, and phone number
* Validation to ensure that all required fields are filled in
* Validation to ensure that the email address is valid
* A submit button that sends the form data to a server (you can use a mock server or API, or use a tool like Mockoon or Postman to mock the server)
* A success message that is displayed upon successful submission
* Clear the form fields upon successful submission

### Guidelines

* Use Create React App to set up the project
* Use functional components and hooks (e.g., useState, useEffect)
* Use CSS modules or styled components for styling
* Use a form validation library like Formik or React Hook Form to handle form validation
* Use Git for version control and commit frequently with clear commit messages
* Write clear and concise code, and comment where necessary
* Host the application on a public Git repository and provide the link to the repository

### Bonus Points

* Add more fields to the form, such as address or date of birth
* Use a serverless architecture with a service like Firebase or AWS Lambda to handle form submissions
* Implement real-time validation as the user fills out the form, such as showing an error message when the user types of an invalid email address
* Add client-side error messages for each field that is invalid

### Evaluation

The coding challenge will be evaluated based on the following criteria:

* Code quality: clear, concise and maintainable code
* Functionality: the application should meet all the requirements listed above
* User experience: the application should be intuitive and responsive
* Bonus points: the implementation of any bonus points features will be taken into account when evaluating the submission

Good luck!

ADVANCED TEST

## Challenge: Dynamic Front Page with TypeScript and React

Create a dynamic front page using React.js and TypeScript that displays a list of items fetched from an API endpoint. The front page should have a search bar to filter the list of items and a sort of functionality to sort the items by different attributes.

### Requirements

The application should have the following features:

* A header section that displays the title of the page and a search bar to filter the list of items by name or any other attribute.
* A section that displays a list of items fetched from an API endpoint using Axios or any other HTTP client. The list of items should include at least the following information: id, name, description, image.
* A sort functionality that allows the user to sort the list of items by name, id, or any other attribute.
* A form to add new items to the list. The form should have fields for at least name, description, and image.
* The application should be written in TypeScript for type safety.

### Guidelines

* Use Create React App with TypeScript to set up the project
* Use functional components and hooks (e.g. useState, useEffect)
* Use CSS modules or styled components for styling
* Use Axios or any other HTTP client to fetch data from a mock API or a real API.
* Use Git for version control and commit frequently with clear commit messages
* Write clear and concise code, and comment where necessary
* Host the application on a public Git repository and provide the link to the repository

### Bonus Points

* Use pagination to display the items in smaller groups on the page.
* Add a feature that allows the user to delete items from the list.
* Add a feature that allows the user to edit existing items in the list.
* Add animations or transitions to the user interface.

### Evaluation

The coding challenge will be evaluated based on the following criteria:

* Code quality: clear, concise, and maintainable code
* Functionality: the application should meet all the requirements listed above
* User experience: the application should be intuitive and responsive
* TypeScript usage: proper use of TypeScript for type safety
* Bonus points: the implementation of any bonus points features will be considered when evaluating the submission

Good luck!