React.js Retake Project Assignment

Your task is to **design** and **implement** a web application (Single Page Application) using React.js. Use a service like Kinvey or Firebase for your **back-end** or create your own with Node.js and MongoDB or a framework in another language (ASP.NET, Spring, Symfony). It can be a discussion forum, **blog system, e-commerce site, online gaming site, social network,** or any other web application of your choice.

1. Application Structure

The application should have:

- Public Part (Accessible without authentication)
- Private Part (Available for Registered Users)

1.1 Public Part

The public part of your projects should be visible **without authentication**. This public part could be the application start page, the user login, and user registration forms, as well as the public data of the users, e.g., the blog posts in a blog system, the public offers in a bid system, the products in an e-commerce system, etc.

1.2 Private Part (User Area)

Registered users should have a personal area in the web application **accessible after successful login**. This area could hold for example the user's profiles management functionality, the user's offers in a bid system, the user's posts in a blog system, the user's photos in a photo-sharing system, the user's contacts in a social network, etc.

2. General Requirements

Your Web application should use the following technologies, frameworks, and development techniques:

- At least 3 different **dynamic pages** (pages like about, contacts, etc. do not count towards that figure)
- Must have specific **views**:
  - **Catalog** – list of all created records
  - **Details** – information about a specific record
- At least one collection, different from the User collection, with all CRUD operations (create, read, update, delete)
  - **Logged in users** – create **records** and **request** to the REST API, **interaction** with the records (via Likes, Dislikes, Comments, etc.)
  - **Logged in (author)** – to be able to **Edit** / **Delete** their records
  - A **Guest** should have **access** to basic website **information** (catalog, details), but **not** to the **functional activities**
- Use React.js for the **client-side**
- Communicate to a **remote service** (via REST, sockets, GraphQL, or a similar client-server technique)
- Implement **authentication**
- Implement **client-side routing**
- Demonstrate use of programming concepts, **specific to the React library**: stateless and state full components, bound forms, synthetic events, Component Styling, etc.
- Use a **source control system**, like GitHub
- **It is required to have committed in GitHub for at least 3 days**

For students who appeared on a regular exam – they must implement a new "main" functionality! The additional Main functionality is considered another Catalog and Details views with CRUD operations that are extending your current project. This should be done for a logged-in user.

3. Other requirements

- Apply **error handling** and **data validation** to avoid crashes when invalid data is entered
- The application should be divided into **components** with **separate CSS files**.
- Brief **documentation** on the project and project architecture (as .md file)
- Demonstrate use of programming concepts - React Hooks, Context API

4. Public Project Defense

Each student will have to deliver a **public defense** of their work in front of the other students, trainers, and assistants. Students will have **only 20 minutes** for the following:

- **Demonstrate** how the application works (very shortly)
- Show the **source code** and explain how it works
- Answer **questions**

Please be **strict in the timing**! In the 20th minute, you **will be interrupted**! It is a good idea to leave **the last 5 minutes for questions**.

Be **well prepared** to present the maximum of your work for the minimum amount of time. Open **the project assets** beforehand to save time.

5. Bonuses

- Use a **state management** solution (React Redux) instead of Context API
- Write **Unit Tests** for your code
- Good UI and UX
- Use a **file storage cloud API**, e.g., **Dropbox**, **Google Drive,** or other for storing the files
- Connect to an external API, like Google Maps, AccuWeather, etc.
- **Deploy the application** in a cloud (Heroku, Firebase)
- **Bonuses depend on the complexity of the implementation**
- Anything that is not described in the assignment is a bonus if it has some practical use

6. Assessment Criteria

General Requirements – 50 %

Implementing all the general requirements will grant you a place on the defense schedule. All projects that do not have the general requirements will not be accepted for defense.

Other Requirements – 20 %

Functionality Presentation – 10 %

Adequately demonstrate the requested functionality. Know your way around the application and quickly demonstrate the code.

Answering Questions – 20 %

Answer questions about potential functionality outside the scope of the project.

Bonuses – up to 10 %

Additional functionality or libraries outside the general requirements, with motivated usage.

7. Submission Deadline

You **must** submit your project before 23:59 on **15 Apr** using a survey that will show up on **10 Apr**. A presentation schedule will be available on **18 Apr** and will include only the projects that were **submitted beforehand**. Keep in mind that after submitting your project you **are allowed** to work on it until the date of the **project defense**. **Non-submitted** projects will **NOT** be evaluated**.**

8. Restrictions

You can use **parts** (some components, routing configurations, form validation, etc...) of the **course workshop**, but you are **NOT** allowed to use the **whole workshop** as your project assignment. You are **NOT** allowed to use **HTML & CSS** structures from **JS Back-End** and **JS Applications** Courses.