# React.js Fundamentals Project Assignment

Your task is to **design** and **implement** a web 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 by your choice.

The application should have:

* public part (accessible without authentication)
* private part (available for registered users) and
* administrative part (available for administrators only)

### 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.

### Private Part (User Area)

Registered users should have 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.

### Administration Part

**System administrators** should have administrative access to the system and permissions to administer all major information objects in the system, e.g. to create / edit / delete users and other administrators, to edit/ delete offers in a bid system, to edit / delete photos and album in a photo sharing system, to edit / delete posts in a blogging system, edit / delete products and categories in an e-commerce system, etc.

### 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)
* Use React.js for the **client-side**
* Communicate to a **remote service** (via REST, sockets, GraphQL, or a similar client-server technique)
* Implement **authentication** and **user roles**
* Implement **client-side routing**
* Demonstrate use of programming concepts, **specific to the React library**: stateless and statefull components, bound forms, synthetic events, etc.
* Brief **documentation** on the project and project architecture (as .md file)

### Other requirements

* Apply **error handling** and **data validation** to avoid crashes when invalid data is entered
* Prevent security exploits (XSS, XSRF, Parameter Tampering, etc.)
* Handle correctly special HTML characters and tags like <script>, <br />, etc.
* Use a **source control system**, like GitHub

### Optional Requirements

* Use **responsive design** – Bootstrap, MDL, CSS Grids or another method of your choice
* Nice looking UI, supporting of all modern and old Web browsers
* Good usability (easy to use UI)

### Deliverables

Put the following in a ZIP archive and submit it (each team member submits the same file):

* The source code
* Don't submit the NPM packages! They are not needed and take too much disk space.
* The project documentation
* Public project defense presentation

### 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 10 minutes** for the following:

* **Demonstrate** how the application works (very shortly)
* Show the **source code** and explain how it works
* Optionally you might prepare a **presentation** (3-4 slides)

Please be **strict in timing**! On the 10th minute you **will be interrupted**! It is good idea to leave **the last 2-3 minutes for questions** from the other students, trainers and assistants.

Be **well prepared** for presenting maximum of your work for minimum time. Bring your own laptop. Test it preliminary with the multimedia projector. Open the project assets beforehand to save time.

### Bonuses

* Use a **state management** library like Flux or Redux
* Deploy the application in a **cloud environment**
* 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.
* Use of features of HTML 5 like Geolocation, Local Storage, SVG, Canvas, etc.
* Anything that is not described in the assignment is a bonus if it has some practical use