# React Fundamentals Teamwork Assignment

Your task is to design and implement a web application using MongoDB, Node.js and React. Optionally, you may use Express.js and Mongoose to abstract your application. 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 15 different dynamic web pages
* Use NodeJS for the server (use an MV\* pattern)
* Use React for the client-side
* Use MongoDB as database back-end
* Create at least four tables with data with server-side paging and sorting
* Use responsive design
* It may be based on Bootstrap, Materialize or any other UI framework
* Use Passport for managing users and roles
* Your registered users should have at least one of the two roles: user and administrator
* Apply error handling and data validation to avoid crashes when invalid data is entered
* Prevent yourself from security holes (XSS, XSRF, Parameter Tampering, etc.)
* Handle correctly the special HTML characters and tags like <script>, <br />, etc.
* Use GitHub and take advantage of the branches for team collaboration.
* Brief documentation of the project and project architecture (as .md file)

## Optional Requirements

* 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

Each team will have to make a public defense of its work to the trainers (in 5-10 minutes). It includes:

* Live demonstration of the developed web application (please prepare sample data)
* Explain application structure and its back-end and front-end source code
* Show the commit logs in the source control repository to prove a contribution from all team members