Reflective report 340CT

The Task

The requirements for this project are to build a full-stack web application for Frequently Asked Questions theme. For this topic, I have developed a web app that a user can register to the website and then login in order to post different questions where other users can answer the question or to upvoted. After logging in, the user is redirected to the home page where the user can see all the questions asked. The user can also interact with the page by asking a question provided with a title and a summery of the question. The user can like the question, and if he wants, he can answer the question by clicking the Answer button. By pressing the button, the page will show the specific FQA with all the answers that it has.

The process

During developing the project, I have decided to use different features and techniques to deliver a quality code and web app that satisfy user needs. By taking these features I took in consideration these factors to save time, increase the performance of the application and security measurements. I put all these together in order to deliver a quality web application.

Using JWT authentication

To prevent unauthorized access to confidential data and the specific content or to transmitting information securely between parties, I decided to use JWT (JSON Web Token) as a method for authentication.

JWT is mainly used for authorization, which use token to ensure that a token follows each request to the server, which eventually the server checks for authenticity and then responds to the request. This architecture provides a very effective way to perform API requests. Tokens has many benefits compared to the cookie-based method; one of them is fine-grained access control, which can easily specify the user permission also the resources that he can have access to. In my case, I can use the token to determine which user should have access to make a post or to upvote a comment.

A disadvantage is that JWT's aren't easily revocable, so a JWT can be valid even if the user has been deleted or suspended, which in my case, a suspended user could see a private route like comment or posting until the token is canceled.

MongoDB

As a database to store the user information and the content of the web, including questions with answers and user details, I choose to use MongoDB. Being a document-oriented database, it is used to store a high volume of data. Unlike the traditional databases which use tables and rows, MongoDB utilizes collections and documents. Every database contains collection of documents and function which are like the relational databases tables. One of the advantages of MongoDB is versatility being a schema-less database, which means that one collection can hold different documents and gives the flexibility to store a variety of data types. It also speeds up application development by being easy to use and reduces the complexity of deployments. One of the limitations that MongoDB has is that it can't handle complex transactions since it is not as strong as RDBMS (Relational Database Management System)

To set up the MongoDB, I also used Mongoose, which is a library for MongoDB and Node.js that manages relationships between data and provides schema validation. This saves me from write validation code that otherwise need to do with MongoDB. It helps me to define a schema for my data models such as creating a schema for "Post" or "Comments" that follow a specific structure with pre-defined data types. A disadvantage of Mongoose is that it can slow down the performance.

React

For building the user interfaces, I decided to use one of the most known and efficient libraries for JavaScript, which is React. It is used in web development to build interactive user interfaces by using reusable UI components without reloading the page. React is a powerful tool for making front-end web application hence the benefits that React offers over other frameworks. A reason that I decided to use React is because of the simplicity which makes it easy to understand while other frameworks like Angular or Ember are referred to as 'Domain-specific Language' which are difficult to grasp right away. The only requirement for React is a basic knowledge of CSS and HTML.

But one of the best features of React is the ability to create reusable UI components or blocks that can be put together to create the page. This feature makes it easy to change or maintain the components, and it helps save time by written less code. All this enhances a faster development and make the codebase simpler. In my case, React had a big impact on my productivity and quality of code. By using reusable components such as the “Navigation bar” or “Landing” page, I could easily reuse them for each page that my website has. Another circumstance that React was helpful was when I created the “Post” component, which I used for the main page to show all the users posts. But also, for the comment section where I reused the component to show the question separately post and the answers without loading the page, which I would normally do without using React.

Anyway, without any significant drawback, just the lack of MVC or its size, React is a key factor in my project by boosting the development speed and improve the quality of the code also, makes maintainability easier. Overall with all its feature, makes React a great library to use for my future projects.

Redux

Along with React, I decided to add another popular library that is commonly used together, and that is Redux. Each component of react has its own memory called state, and Redux is a tool to manage these states. By using Redux, the state of the app is kept in a store. This allows each component to have access to any state that it needs from the Store without having to send down props from one component to another.

Redux has three building parts that I also added to the project, which are actions, reduces, and Store. Actions are events that send data from the app to the Redux store. The data can be, in my case, user interactions or from submissions. Each action is sent by using the “dispatch()” method, and they must have a type property that indicates what action to carried out. Also, it has a payload that contains the information that should be processed by the action. As an example, in my case, “delete post” actions has a type of “DELETE_POST” that, send the id which is the token generated by the user when de login to the website, as a payload. If the action didn’t go through it will send a message error. Reducers are the functions that take the current state of the app, perform the action that is assigned, and return the new state. And, the Store is the memory of the app state where you can have access to it or update it.

There are some reasons that I choose Redux, and one of them is that it makes the app easy to debug because coding errors or network errors that might come up during the production are easy to understand. An example is when a user introduces the wrong credentials, by adding the Redux extension to the Chrome, you can see a plain error to the console. Another reason is that it makes maintainability easier because Redux keeps a strict structure in an organized way that makes the code easier to maintain.

But Redux has some disadvantages as well, one as mention above, is that it uses a restricted design. Compared to React, Redux is challenging to grasp away, and for someone that has no knowledge of Redux, it makes it hard to understand the code.

All these features mentioned above make Redux an excellent tool for quality code, but it also came with some drawbacks.

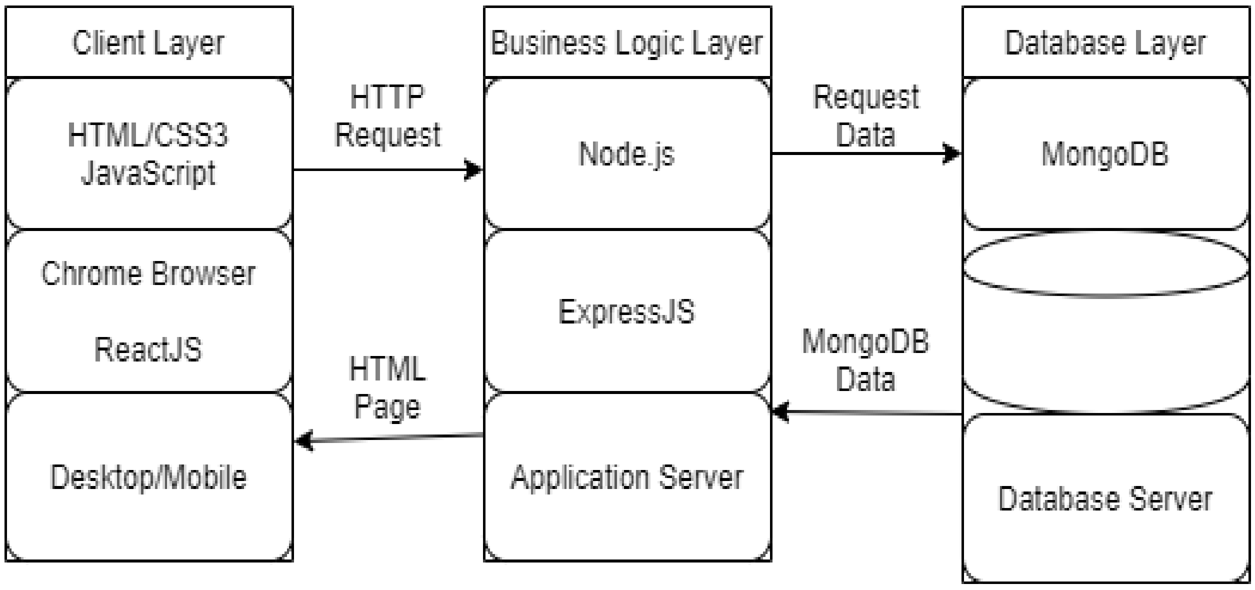
Architecture

Because I wanted to use all the features mentioned above, I had to decide which architecture will suit best. After conducting some researches, I decided that Unidirectional Data Flow Architecture will fit for this project.

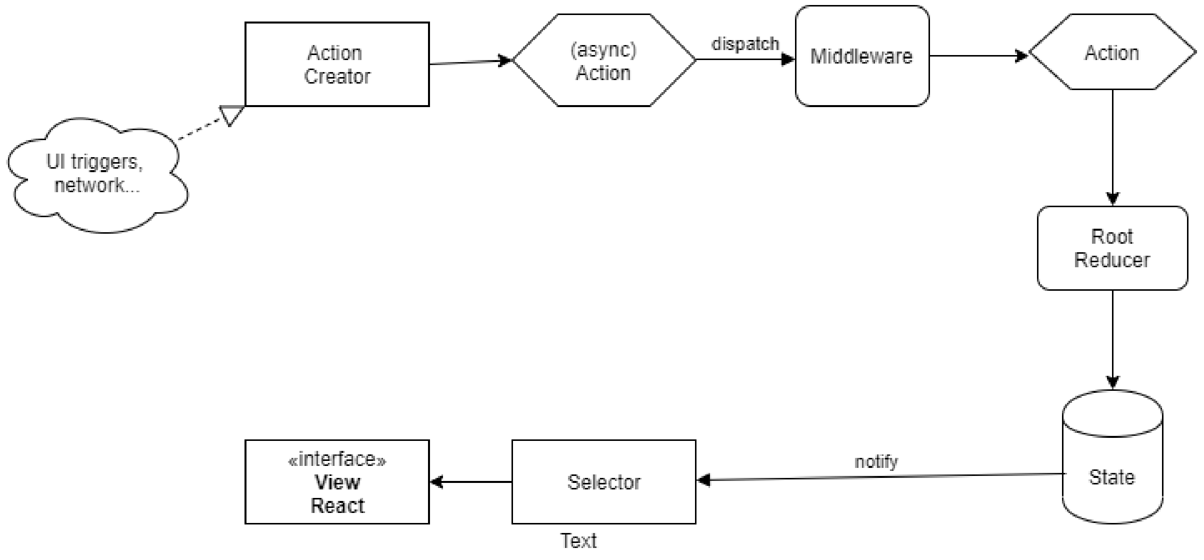
Also known as one-way data flow, which means that data follows the same lifecycle pattern.

The data is transferred to other parts in a single direction. This makes the app more predictable and gives better control over it. The reason that I choose this approach is that combined with an only store such as Redux will provide consistency to the flow, which means that changes to state happen in a consistent manner and actions or events are also dispatched consistently. Also is more efficient and because gives an understanding where the data is coming from it make easy to debug. These methods bring many benefits for React because usually react apps nest child components within the parent component that have a container for the state of the app. And by using Unidirectional Data Flow states are moved to the view and to child component also state can be updated only by the actions, and a parent will not be affected if a state on a component is changed. The drawback of this method is that it cannot backtrace in the directions that the data came from since this method is a unidirectional way.

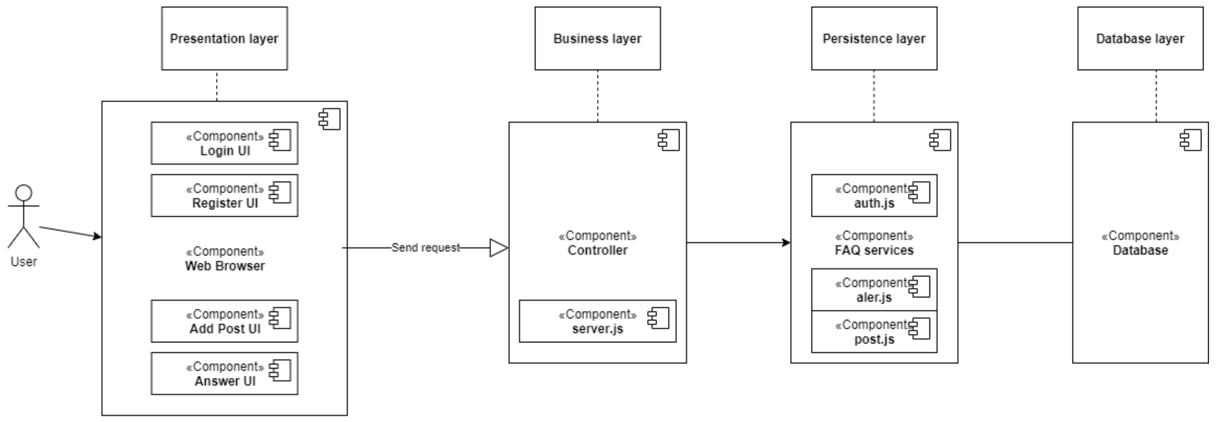
Structure Diagram



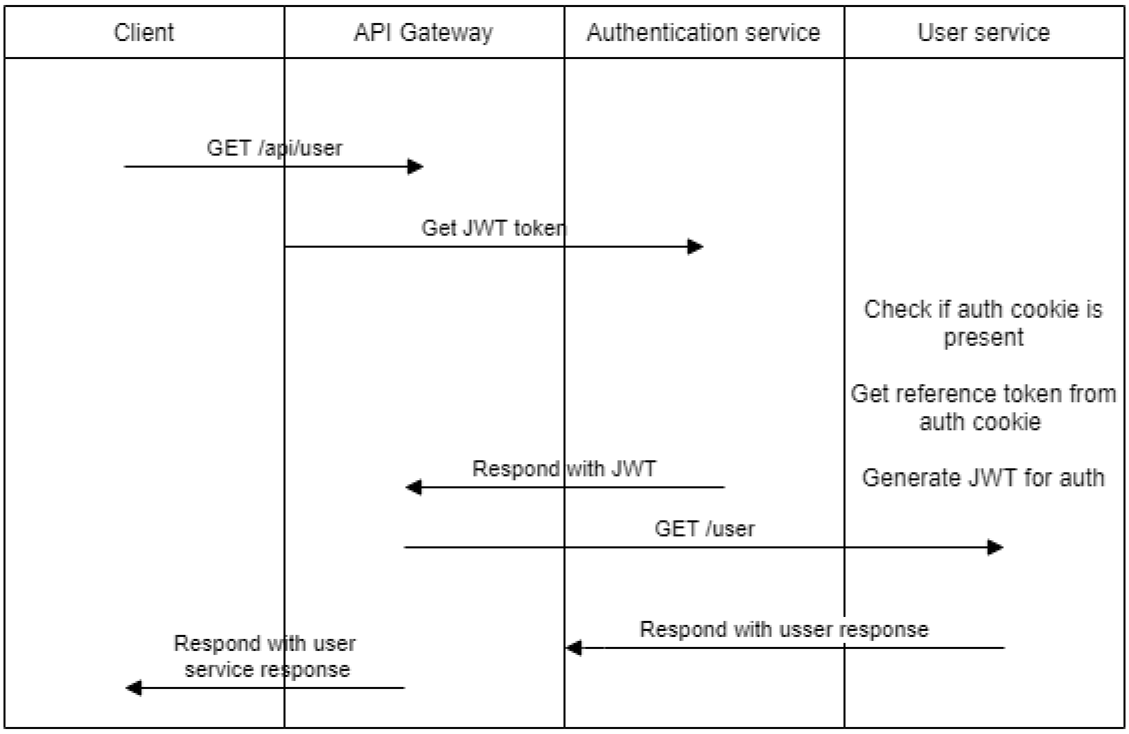
Unidirectional data flow structure



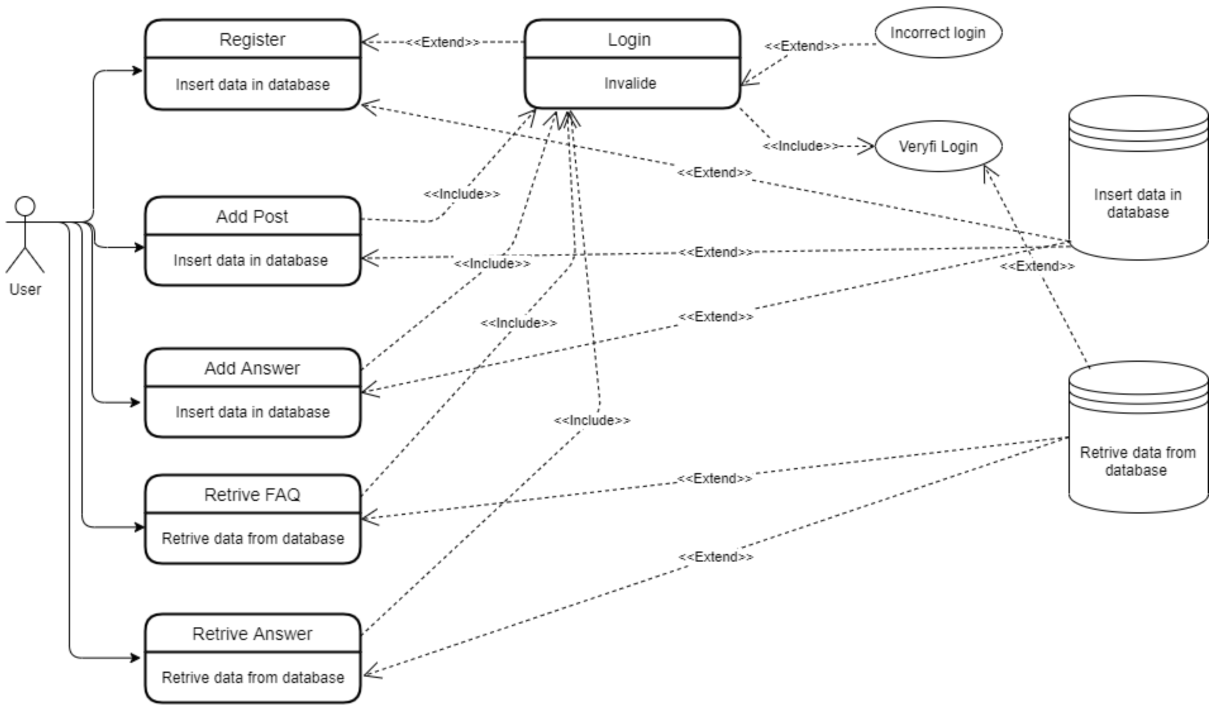
Component diagram



JSON Web Token Flow


Client
API Gateway
Authentication service
User service
GET /api/user
Get JWT token
Check if auth cookie is present
Get reference token from auth cookie
Generate JWT for auth
Respond with JWT
GET /user
Respond with usser response
Respond with user service response

Use case Diagram



The component diagram renders an example of how the app works with all the functionalities. As the controller for the functionality server.js is the control point and for the user interface React.

JWT Flow diagram shows how the token is generated by the Authentication service and send it back to the user as a service response, which eventually with the token user can interact with the app.

The structure diagram is a representation of how the app is structure. Backend server and frontend client are two separate things; both have their own node_modules. On the backend, Express is installed for Node runtime and Mongoose for a convenient way to talk with MongoDB.

The entire app has multiple features, to access those features the user needs to log in or to register, and then the user can interact with the app. For every action that user does a message will be printed on the screen with a confirmation of the action deployed by the reducers.

Testing

A quality code needs to be tested in order to confirm that the unit performs as designed. For this project, I have implemented two types of testing. The first testing method that I have implemented is the unit test. After finishing functionality, I have implemented the unit test to make sure that the function does what is designed to do. To have a better-quality code, I needed to make sure that testing is covering most of the features and pass them without any trigger any errors. After implementing unit testing, I started to apply acceptance tests that are meant to test user needs and requirements also to prove that the system satisfies the acceptance criteria.

Postman

Another tool that I found to be very useful when I was developing the backend was Postman. A tool that lets me to make an API request and test and run different request methods without having a front-end already build up. It has a friendly user interface that makes it easy to use.

Vision Control

As a Version Control for this project, I have used GitHub. I started by setting up my private repository. Initial commits were made on the master branch to set up all the resources that I needed and to create a base of the project. Every commit was made when a new feature was made so I can have a track of my work in case I encounter a problem or bugs by modifying existent files. After finishing my base model, I decided to add new major features such as the "Adding Questions," so, to do that, I created a new branch with the specific name where I can implement the new feature and work separately from the master branch. By using a branch, I can have better control of my workflow, and I can organize the work more efficiently. For this project, I have created only two branches just for the significant features because being only one person working on the project, and I was developing just one feature at the time. So, there was no need for multiple branches. After I finished developing the new feature, I merged the branch to the master. While using GitHub as a Vision Control for this project, I have discovered some advantages that helped me to improve my organization and workflow by using branches. Also, by keeping track of the history of my changes, in some instances of bugs or program failure, I could easily compare my changes and revert them where it was necessary. This helped me to save time and to fix the code quickly. Furthermore, I considered that by using branches was very helpful for an individual project to keep my code and workflow clean. In contrast, for a group project or a team project, branching should be mandatory.

GITHUB: https://github.coventry.ac.uk/340CT-1920SEPJAN/moscalir-resit.git

Things to improve

Nothing is perfect, and always there is room for improvement. If I had to do the project again, surely, I would do it differently. Because this project is not a complex project with hard to solve problems, I would choose a different architecture such as Model View Controller. Because it has been around for a long time and there is a lot of support on this theme and its simplicity, it is easy to apply to any type of project, while Unidirectional Data Flow can be harder to understand. And for the front-end I would probably use Vue.js over React because Vue's documentation is largely considered better. Compared to React Vue is easier to learn and pleasant to write. Because Vue is faster and smoother than other frameworks, makes the developing process faster since it uses components and templating syntax. For Vision Control, I would probably use Git-Flow because I could have multiple versions of the web app while developing the next one.

Reference list

1.Hyperion Development (September 10, 2018) Everything You Need to Know about the MERN Stack, Available at: https://blog.hyperiondev.com/index.php/2018/09/10/everything-need-know-mern-stack/ (Accessed: 10 March 2020).

2.Ali Alhaddad (May 23, 2018) One Way Data Flow vs Unidirectional Data Flow, Available at: https://medium.com/@alialhaddad/https-medium-com-alialhaddad-redux-vs-parent-to-child-2583c8e29509 (Accessed: 19 March 2020).

3. Liz Denhup (Sep 25, 2017) Understanding unidirectional data flow in React, Available at: https://medium.com/@lizdenhup/understanding-unidirectional-data-flow-in-react-3e3524c09d8e (Accessed: 21 March 2020).

4. Margaret Rouse (August 2018) MongoDB, Available at: https://searchdatamanagement.techtarget.com/definition/MongoDB (Accessed: 26 April 2020).

5. Flavio Copes (Dec 13, 2018) JSON Web Token (JWT) explained, Available at: https://flaviocopes.com/jwt/ (Accessed: 29 March 2020).

6. Neo Ighodaro (August 31, 2018) Why use Redux? Reasons with clear examples, Available at: https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/ (Accessed: 20 April 2020).