Single-Page Online Shop App Development using React and Redux

Garmastewira - 13514068
Informatics/Computer Science Department
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514068@std.stei.itb.ac.id

***Abstract* — Most people surf the internet by using their mobile phones. The conventional multi-page application is not suitable for mobile phone because it will provide an unpleasant experience for the user given its responsiveness. A trending approach to solve the problem is by converting the traditional app into a single-page application. One of the most popular single-page application tool is React and Redux. Based on the online shop project that is created using React and Redux, it is proven that both libraries are a great tool for rapid single-page app development because of its clear data flow, lucid and small components, therefore providing ease for project's implementation and testing.**

***Keywords* — Chai, JavaScript, Mocha, Node.js, React, Redux, responsive, single-page application**

I. Introduction

Surfing the internet can be regarded as humans' primary needs nowadays. Specifically, people use their smartphones as their primary platform to browse the internet. A survey conducted by Kleiner Perkins Caufield & Byers states that as of 2016, the percentage of users who use mobile to browse the e-commerce is 52%. On the other hand, the percentage is much lower for people who use desktop browser, which is 41% [1].

User experience in mobile and desktop applications differs in many ways. Mobile app's nature is often attributed to responsiveness. It requires instant response upon given action, and multi-page application isn't suitable. Rendering the whole header back and forth while clicking links to navigate will give an obnoxious user experience to the users.

The rise of smartphone users is the reason why single-page application is trending. It gives smooth transition between page rendering each time a user navigates sections of the site, as the application chooses cleverly which part. It also uses dynamic loading to retrieve data from database so that it only loads when needed.

Although there are a lot of libraries and frameworks that offer support for single-web page application, this paper will only focus on React and Redux. React and Redux has proven to be the most popular tool for creating single web-page application. For this case study purpose, we will build a simple online shop project named as ShopChop that contains user management, product list, shopping cart, checkout form, and user's order history modules.

II. Literature Study

A. Single-Page Application

Single-page application is a web application that fits on a single web page with the goal of providing a user experience similar to that of desktop application. In an single-page application, HTML, CSS, and JavaScript is retrieved with a single page load [2]. Other appropriate resources that are retrieved from the database will be dynamically loaded with the help of JavaScript's AJAX feature.

B. React

React is an open-source JavaScript library for building user interfaces [3]. React is maintained by a developer team of Facebook. React introduces 3 important concepts that makes it different from other popular libraries, which are React components, one-way data flow, and virtual DOM.

Most JavaScript single-page application libraries/frameworks divide a page's structure into HTML/CSS and JavaScript components. However, React sees a component differently. A component is a part of the page that is small enough to be considered as an independent and cohesive part. Uniquely, a component will consist both HTML and JavaScript code. By treating a component like this, it is easy to define the relationship between each component, e.g. parent-children.

Another feature that makes React distinct is its one-way data flow concept. A component is defined by properties that are passed by the parent's component. A component cannot modify its properties. React itself actually provides each component the ability to have a state. Nonetheless, having an internal state is heavily discouraged. One way for a component to have a state is to integrate React with external state container libraries such as Flux and Redux.

Finally, React creates an in-memory data structure cache, computes the differences efficiently, and updates the browser's displayed DOM. This cache management is dubbed as VirtualDOM. This way, a programmer can write code as if the entire page is rendered on each change as React handles it automatically.

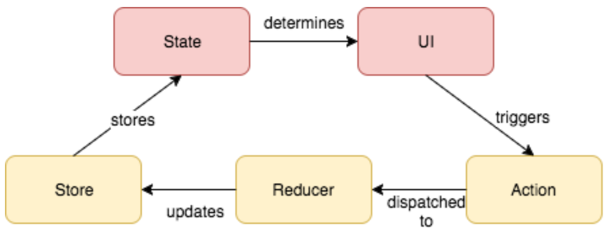

Figure 1. Redux's data flow

C. *Redux*

Redux is a predictable state container for JavaScript applications [4]. It basically helps users to manage many states that a single-page application might contain. For example, some of the complex states that are needed to be managed are selected tabs, current routes, pagination, and AJAX loading.

Redux has 3 basic components, which are store, actions, and reducers. Fig. 1 shows the data flow from Redux components (shown as yellow boxes) to the view components (shown as red boxes). Below are the details of Redux's data flow.

1. The store component stores every single state the application needs. States are often about the view components' states although not limited to. Note that there will be only a single central store for the whole application.

2. The action components are actions that are triggered upon interactions done by the user, e.g. clicking button. Actions triggered are instantly dispatched into the reducer for further processing.

3. The reducer components will receive the actions dispatched, and examining the type of each action. Each action with a particular type will be handled in a different way, and each handling might change the states stored in the store component.

The view components can be any JavaScript frameworks/libraries that support single-page application. However, Redux is often paired with React because of React's nature of dividing the whole view component into smaller components with states attached to each component.

III. Methods and Tools

A. *Client Side Tools*

For the application's client side (front-end), React and Redux will be used as the primary libraries. These two libraries are the foundation of the single-page application. React-Redux middleware is used to connect React's components and Redux's flow. Additionally, Twitter Bootstrap is used as a framework for the application's CSS styling and JavaScript behavior.

B. *Development Tools*

To develop the application, several middlewares are

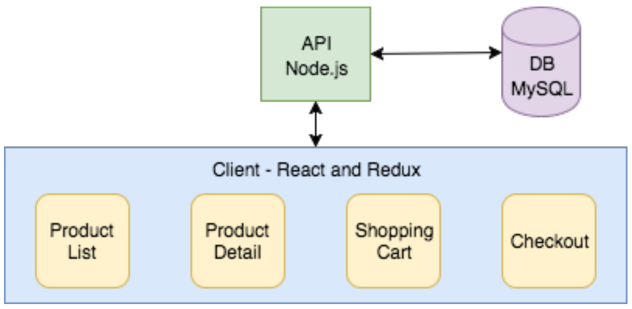

Figure 2. ShopChop online shop app's structure

needed to be managed. NPM will be used as the package manager. Webpack will be used to bundle all React and Redux JavaScript files into one JavaScript minified file. Lastly, Babel will be used to transpile the JavaScript ES6 syntax (which is recommended by React's developers due to the syntax's nature) into JavaScript ES5. This is important as most browsers can only interpret JavaScript ES5 syntax.

C. *Server Side Tools*

The online shop will use Node.js as the back-end side programming language. Express.js is also used as a framework for the routing of the server. For the database, Sequelizer will be used as a middleware to connect with the databasa management system, MySQL.

D. *Testing Tools*

For the testing part, Mocha is used as the runner and handler of each test case run. React itself has provided a utility for testing called ReactUtils. One of its core feature that will be used is its feature to simulate events like button click easily. JSDOM is used to render a component as if it is rendered in the browser while the testing is actually conducted in terminal. The last required testing tool is Chai, which is used as an assertion tool to check whether a component is rendered correctly.

E. *Online Shop Structure*

Fig. 2 shows the structure of the online shop application, named as ShopChop. ShopChop is basically divided into 2 major subsystems, which are the client side, and the API server side. The API server side is built using Node.js to handle data requests from client by retrieving it from the database which is handled using MySQL.

The client is side is the subsystem that is seen by the customers. It has 4 core pages, which are the product list, product detail, shopping cart, and checkout pages. The flow a customer buying a product is the following.

1. A customer logs in, checks the product list, sees the detail of each product that is clicked, and adds any product that is interesting to the shopping cart.

2. After finished of browsing products, a customer can do a checkout, in which the total price of all of the products are summed, including tax and delivery price.

TABLE I. Summary table of client side's implementation source code

No.	Type	Total
1	Actions	8
2	Reducers	3
3	Components	8

TABLE II. Summary table of client side's testing source code

No.	Test	No. of Files	Test Cases
1	Actions	8	16
2	Reducers	3	10
3	Components	8	15

IV. Results

A. *Client Side*

Fig. 3 shows the final results of the client side. The client side is first initialized with a single HTML file named index.html that will serve as the loader of the React and Redux JavaScript bundle file. It also serves as the loader of external CSS and JavaScript files such as Bootstrap and JQuery. Besides the index.html file, all of the React and Redux source codes before bundled are placed into a folder alongside with a main JavaScript file, and the React routing file. The user can navigate into 8 different pages based on the URL route.

Next, the client side folder has 3 main subfolders, which are components, actions, and reducers. Table I shows the number of files that are created inside each subfolder. The components are often the template of each page that can be navigated. There are only 2 types of the actions, which are fetching data and posting data to the API server. Lastly, while the reducer handles a lot of actions, it can be generally categorized 3 files, each is used to handle products, users, and orders.

B. *Server Side*

The server side consists of 8 REST routes, each of them granting user's specific request. The server is only contacted by the client side through AJAX mechanism using Axios middleware. The server side has also created 5 models using Sequelize to wrap the data retrieved from the database. Fig. 4 shows the general relationship between each model. The detail of the relationship shown is the following.

- User's shopping cart implementation is shown as a User having several AddedProducts, while each AddedProduct is connected to exactly one Product.
- User can have a lot of Orders, and each Order will contain several BoughtProducts. Each BoughtProduct is of course related to one unique product.

C. *Testing Files*

Table II shows the number of files that are created for the testing stage. The number of the testing files matches with the number of the implementation source codes. Every action test file contains 2 test cases, one is to check if the action's type is correct, and the other is to check whether the action carries the desired payload. Reducer's test cases are quite similar, in which each file should test for the action types it can handle and for any unknown action type. Components' test cases number are arbitrary, but component with form has more test cases compared to ordinary ones.

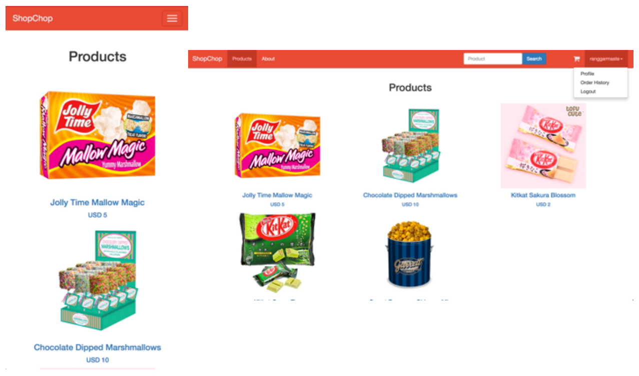

Figure 3. ShopChop product list page, seen in mobile (left) and in Desktop (right)

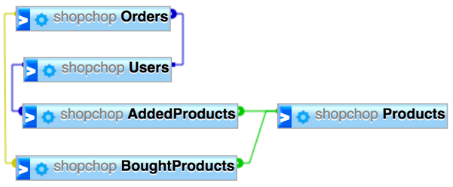

Figure 4. ShopChop database schema

D. *Overall Results*

Overall, the online shop's implementation is complete. The project has a responsive display for both mobile and desktop browsers. User can navigate through pages without the header and footer re-rendering themselves. User can register, login, add products to shopping cart, and purchase the products successfully. The testing side, however, might need additional test cases for the components to check the components' wanted behavior more thoroughly.

V. Discussion

A. *Implementation Stage*

The most obvious advantage of developing a single-page application instead of a multi-page app is the separate development between the client side and the server side. The server side is 100% free from template rendering job. It only functions as an API server which mostly only fetches data from the database. Therefore, the server side will only require a lightweight back-end framework such as Express.js, or perhaps Flask if it's developed in Python.

Moreover, the client side's composition is very tidy and structured. If the developer wants to see the data retrieval, they only need to check the actions. The application's state definition is solely defined in the reducers. Finally, the rendering job by applying the application's state is done in the rendering. Thus, the scalability of a client side using React and Redux is definitely high.

B. Testing Stage

The other advantage of building a single-page app using React and Redux is its ease and comprehension of the testing stage. Each data flow is handled by action and reducer, and both can be tested to see the correct state they should produce using small unit tests. Additionally, with JSDOM, Chai and ReactUtils, a component is not tested by its functions' output like the conventional unit testing. Instead, a component is first rendered as HTML using JSDOM, and Chai should test whether the component has the correct rendered output, e.g. has a button, the behavior if the button is clicked, etc. This is currently the new approach of testing instead of just mere functional unit testing.

C. Comparison to Other Libraries

React and Redux is absolutely not the only single page framework/library available in the internet. Other frameworks like Angular, Ember.js, Backbone.js, and Vue.js are also famous and are supported by an active global community. The strongest contender of React and Redux is Angular, mainly due to its Facebook vs. Google issue. Nevertheless, in the past 5 years, Angular has always been more popular than React according to Google Trends.

Angular uses the classic two-way data binding and HTML-JavaScript separation. Both features are often claimed to much easier to learn React's one-way data flow and HTML-JavaScript integration concept, thus probably is the reason why Angular is still unbeatable in terms of popularity. However, React and Redux's evident advantage is in the testing stage. The units that should be tested are very simple and clear. Unlike the classic approach, one must double the work to see the behavior of the JavaScript as well as the HTML output. As mentioned before, this is a clear reason why React and Redux apps is way more scalable than Angular apps.

VI. CONCLUSION

In this paper, the development of a simple online shop shows that React and Redux is perfect for rapid and tidy single-page application development. Using React and Redux makes the development process much simpler in both of the implementation stage and the testing stage.

Further projects might involve other open source packages that are build by other people to be combined with React. Another idea is to make a React Native project, which is a development tool that uses CSS, React, and Redux as its primary tool to write mobile application that are ready to be translated directly into Java Android Studio or Swift XCode language.

REFERENCES

[1] Meeker, Mary. (June 2016). *2016 Internet Trends Report*. Retrieved May 2, 2017, from http://www.kpcb.com/file/2016-internet-trends-report

[2] Flanagan, David, "JavaScript - The Definitive Guide", 5th Ed., O*'Reilly, Sebastopol, CA, 2006,* p. 497

[3] Simons, Eric. (May 2017). *What Exactly is React?*, Retrieved May 5, 2017, from https://thinkster.io/tutorials/what-exactly-is-react

[4] Bachuk, Alex. (June 2016). *Redux · An Introduction*, Retrieved May 5, 2017, from https://www.smashingmagazine.com/2016/06/an-introduction-to-redux

DECLARATION

I hereby declare that this paper is my own work, not a copy or translation of others' works, and not an act of plagiarism.

Bandung, 5 May 2017

[illegible]

Garmastewira
13514068