IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Comprehensive Analysis of React-Redux Development Framework

Shravan G V

Dept. of Computer Science and Engineering

R.V College of Engineering Bangalore, India

Prof. Anitha Sandeep

Dept. of Computer Science and Engineering

R.V College of Engineering Bangalore, India

Abstract: As developers while creating apps compatible on cross platforms is a tedious task and poses huge challenges before the developers, to be acquainted with cognizance specific to two or more native platforms. Besides, certain hybrid application frameworks existing in market were not able to cater similar experience to same user on all native platforms. Comprehending a solution to the underlying problem, the research work presented in this paper utilizes the advantages of React-Redux framework to create a hybrid application capable of provisioning solutions. The generated framework can be used for windows and iOS machines and the produced results are satisfactory for the users in both the platforms. The proposed methodology has been implemented using JavaScript ES6.

Index Terms **- cross platforms, end-user, React-Redux, hybrid app, frameworks.**

I. INTRODUCTION

Creating web application often tends to be a difficult job for a developer as the developer has to make one application for Android platform and one for iOS platform which requires knowledge of two different domains i.e. for Android application, the knowledge of different browser interpret react and its libraries (programming language: Kotlin or JavaScript) and for iOS mobile application (programming language: swift or JavaScript and IDE: XCode) is required. Besides, some popular hybrid web application framework made by developers were not much successful for creating the exact same experience on multiple native platforms. Comprehending a solution to the problem, this paper utilizes the advantages of React-Redux which is a framework capable of generating hybrid web applications for cross platforms with high data fetching without caching. React developed by Facebook developer's in 2015, tends to have a single place to write code in JavaScript ES6 as the programming language and build mobile application for both iOS and Android at the same time with single code written for both the native platforms. To implement complex applications, it uses another dependency with name Redux capable of managing the state in react native. Moreover, this Research paper describes the dependencies developed by other react native developers that has been utilized in this research work such as database and user interface.

The research paper tends to achieve the following objectives:

- To judge the user experience of the web application made by react redux framework as satisfactory.
- To check if the user interface formation is easy in comparison to other platforms.
- To differentiate between generation of simple and complex apps using react-redux.

i. Introduction to Proposed Application

The proposed application is generated by using react redux framework and is compatible with iOS and Android platforms. The application can cater to any user needs. This application stores all its data in its backend like other applications but with the help of redux, we can store some data on the front end and same time in fetching it. This framework can be used in multiple systems and multiple applications. The react redux has a built-in feature codebase, a run-time environment that runs same code for android and iOS and has been used by researcher to develop compatible code. The modules of web application are discussed in next section.

ii. Trends in React Redux

With the increase in the technology worldwide, many organizations have come forward to utilize efficient and faster UI frameworks. Redux is a library that can be used with any UI framework, including React, Angular, Vue and vanilla JS. Even though Redux works with all frameworks, Redux and React are commonly used together, they are independent of each other.

When we are using Redux with any other form of UI framework, we will normally use a "UI binding" library or a connecting library to tie Redux together with our UI framework, instead of directly interacting with the store from your UI code. React Redux is the original UI binding library for React**.** So reduces any use of a binding layer and provides direct connectivity.

iii. Need for React Redux in Application

a) React Redux is the official React binding for Redux. It lets your React components read data from a Redux store, and dispatch actions to central data store to update data. Developer can easily manage these states of the application easily by the help of global accessing feature.

b) When any change is made in the state, it disrupts the child components, thereby affecting the performance. However, Redux library centralizes the state management of the application. This allows the developer to use significant features of development like undo/redo, state persistence, and far more.

c) In React, it is very difficult to track the state of the application at the time of debugging process. But Redux provides an excellent developer experience that permits “time-travel debugging”, and even send complete error reports to a server.

d) React has a complex UI, where the data flow would be difficult while we are using more components to share the same data. However, Redux is flexible with all the UI layers and features a large ecosystem of add-ons to suit your requirements.

e) It is very difficult to reuse the components in React because it is tightly coupled with the root component. Redux reduces this complexity and provides global accessibility that helps to create applications that can be used frequently to check and run in several environments (client, server, and native).

II. PROBLEM STATEMENT

In UI development most libraries like React, Angular, etc. are built in a way that the components internally manage their state without any external library or tool. This works for some applications with few components, but as the application grows bigger, managing states which are accessed across components becomes a mess. An effective way should be found in an app where data is shared among components.

III. ARCHITECTURE OF REACT-REDUX

A typical React-Redux system is designed to support a stable web framework. It combines store data and user interface. A typical React-Redux architecture is depicted in figure 1.

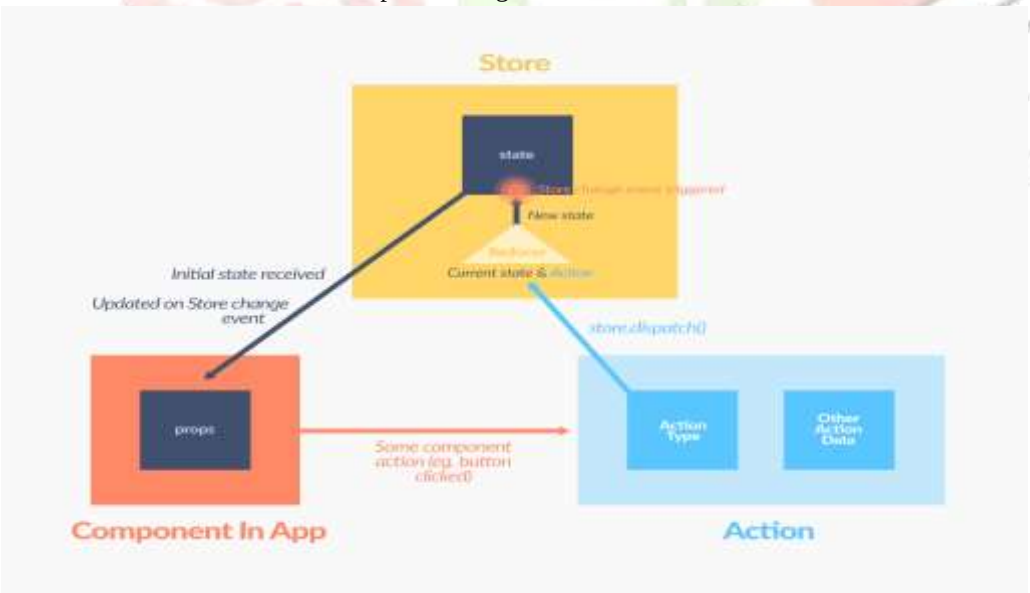

Fig. 1: Architecture of a typical React-Redux Framework

As shown in figure 1, shows the typical architecture of react redux framework.

The architecture comprises of the following components:

- **Component**: It is responsible for maintaining the metadata & providing access of metadata to other concurrently running services.
- **Template**: Responsible for the movement of data from sources to targets.

- **Container**: Generation of reports is done by this service.
- **Actions & Action Creators**: Provides a platform to compute and execute the above services.
- **Reducer**: Creates mappings between source and target.
- **Selector**: Creates workflows and manages their execution.
- **Store**: Monitors the execution of workflows.

Component:

The component is anything that is rendered with a set of props that get passed down from its parent component. In any case if it is a template, a component, and even a container is just a component. It is a set of code that uses properties and can call other components and gives them properties. A component really starts with an API.

Template:

It is a form of simple component. But it is a special kind of component that does things more specifically than the lower components do. It can be referred as this as adding opinions to the components. A component may allow any title, but the template may only pass down a single title for anything that is there.

Container:

The container is the bridge that connects react to redux. It connects it in a lot of ways. This is where the *react-redux* module is used, and it is usually called that *connect* as that is what we use it for. It takes three arguments, an object that maps state to props — mapStateToProps, an object that maps actions to dispatch *(these are used to wire events to actions)* — mapActionsToDispatch creates a response to the action and *mergeProperties* merges all the properties within and passes it to the react DOM for rendering.

Actions & Action Creators:

They can be put together because they are often referred to as each other. Often when people are referring to actions, they really mean action creators.

Action: This is an object that contains the type of action and the state that was changed because of the action.

Action Creator: This is the code that is called to create an action and send it along to the reducer.

We can think of the Action as an event for redux. When an event is fired there is an event type (like onClick, onHover, etc.) and has an event object that contains the data from the event. Well an action is the same, it has a type and the data.

Reducer:

The key thing to know about a reducer is that for every dispatch every reducer is called and given the dispatched action. After the action is passed it is up to the reducer to handle it or pass it on.

Selector:

The selector is how you would get data out of your store in the container. A selector is a function that accepts Redux state as an argument and returns data that is derived from that state. Selectors can provide performance optimizations to your application and can also help you encapsulate your global state tree.

Store:

A store holds the whole states of the application in the form of a tree. The redux way to change the state inside the store is to dispatch an action on it. A store is not a class. It's just an object with a few methods on it. To create means passing the root reducing function to createStore.

IV. RELATED SURVEY

A huge number of papers and textbooks cover topics of native web development. Cross-platform development is addressed by at least some of these, although not necessarily as the main topic. To shed light on related literature, we look on such work in the following that compares several cross-platform app development frameworks. Web technology provides one means to develop apps that span platforms. Native apps are also useful for benchmarking cross-platform approaches, particularly concerning their look & feel and their performance. Another very popular way of state management on a cross-platform application is by using Angular with NGRX store for state management or React with Mobx store. Both frameworks are well established and proven technologies.

Choice of Frameworks

To ensure that the comparison and evaluation of the frameworks were as objective as possible, only frameworks leveraging JavaScript as their main programming language were chosen. For industrial application, typically some if not all but one paradigm is ruled out due to a host of preconditions. With the emergence of new frameworks, technology choice is not easy. Due to the possibility of underlying differences in frameworks using other languages, those using anything, but JavaScript were excluded. Three frameworks for cross-platform development were chosen due to their novelty, recentness, and (perceived) developer interest.

All of these frameworks have its own advantages and disadvantages, and selecting a particular framework is up to the user and their particular need for the application.

V. APPLICATIONS OF REACT-REDUX

With the advent of new technology, there has been rapid modernization, which has led to the increase in the generation of data on the UI. Many companies are now implementing this framework to help them in various apps of the business.

i. Huge front-end data:

Redux is mainly used where a reasonable amount of data gets changed over time. If the data states are not changing frequently then Redux usage is limited. It is used as a best practice for managing application state.

ii. State interpretation for complex applications:

There are some libraries or front-end technologies like React having their own application state management, so while using these libraries one should learn its inbuilt capabilities. Sometimes after developing the application, it becomes complex to understand and code, it is hard to know how the state has been changed. In this scenario, Redux is helpful and used.

iii. Scalable applications:

Some applications can scale in size and when redux is used we can simply add values to redux to scale up or down. The performance of the system is not affected with the increase in data.

VI. FUTURE SCOPE

Future work will include a longitudinal research study to investigate framework maturity, as well as an in-depth perspective on user interface design and interaction.

The react redux framework enables developers to have a stable architecture. This framework can be extended with a middle ware caching or backend data caching on the front UI. The data can from the backend can be transferred to the front end. This data can be easily managed by redux and provide an efficient solution.

VII. CONCLUSION

We have presented a study and discussed issues related to success factors for cross-platform web application development. Our work includes an in-depth assessment of react redux platform app development frameworks with industry anchored research goals. At the end of this survey we can clearly see the advantages of using react redux for certain circumstances. Management of states and props is also in the most efficient way in terms of data rate and response time. The survey together with academic literature and discussions with the case company formed the basis for the requirements and the technical implementations. Moreover, this provides insights for practitioners through mapping survey findings to common issues identified in react redux framework app development. We confirmed user experience, technical implementation, app performance, and testability to be the most common issues related to react redux framework. There are still many problems to tackle, such as the human side particularly in the form of end-user experience.

REFERENCES

[1] Vipul Kaushik, Kamal Gupta, Deepali Gupta(2018). React Native Application Development,10.1007/978-981-10-6620-7_59.

[2] Tim A. Majchrzak, Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks

[3] Available FTP: https://www.valentinog.com/blog/redux/

[4] Available FTP: https://dzone.com/articles/managing-state-in-angular-with-ngrxstore

[5] M. Miškuf and I. Zolotová, "**Architecting React Applications with Flux**," 2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, 2015, pp. 193-197. [8] Talaoui Y., Kohtamäki M., Rabetino R. (2017) Business Intelligence—Capturing an Elusive Concept. In: Kohtamäki M. (eds) Real-time Strategy and Business Intelligence. Palgrave Macmillan, Cham.