Dynamic Reporting Tool for SEBI

Prayag Monsara
(201701257)
DA-IICT
Gandhinagar, India
201701257@daiict.ac.in

Off- Campus Mentor
Mr. Jagruthnath Kovi
KFin Technologies Private Limited
Hyderabad, India

On-Campus Mentor
Prof. Gagan Garg
DA-IICT
Gandhinagar, Gujarat

***Abstract*—In almost every profession with the help of IT solutions we can make our life easy and work experience smooth. Frontend engineering plays a very crucial role in the world of software or web application. Because at last it is the frontend end user or the client will be interacting with. They use our platform to complete their task that is why User Experience also matters a lot. Kfintech is a FinTech company it gives equal importance to finance and technology. Kfintech is in finance sector since many years, yet it always look forward to give the best tools to it's client and tries to enhance their user experience every now and then. Kfintech provides many services such as Mutual Fund Services, Corporate Registry Services, National Pension System, Global Fund Services, Global Business Services, Portfolio Management Services and many more. I got the opportunity to work in one of it's project during my six month internship.**

***Index Terms*—Frontend, HTML, CSS, JavaScript, React, Redux, Kfintech, Firebase, Gatling**

I. Introduction

KFin Technologies Private Limited is the largest registrar and transfer agency and a market leader in the financial sector providing investor servicing. As we are one of the biggest company in our field, there are series of audits that takes place by SEBI(Securities and Exchange Board of India) and tons of reports get generated by SEBI. Our clients i.e. AMCs (Asset Management Companies) also generate reports for internal usage and audits as well. Hence, we decided to develop a fullstack web application with a good User Interface (UI) which can serve all above mentioned purpose at one place with a great user experience. By this web application one can generate all type of reports in any date range with any given report values and dimensions, schedule reports at a particular time and date, see the pending reports that are getting generated, see previous report that has been generated, report a bug to developers and many more functionality. This web application would be used by SEBI, by our clients i.e. AMCs and by our company Kfintech as well.

In the beginning of the internship they planned training for me. This training was divided into two parts (i) Business Training and (ii) Technical Training. They gave business training virtually, business training was also equally important as I am from IT sector and hence understanding finance would be always helpful to tackle a problem while developing the web application. Then they call me on-site for technical training. In on-site technical training they give me the overview of all technologies that are being used by the company. After completions of training they assigned me into the frontend team. After that my internship has been divided in two parts. (i) Learning the tech stack and get exposure of development and (ii) Contributing to ongoing project

II. Tools and technologies used

A. HTML, CSS

HTML [1] and CSS [2] are the best technologies that used to build the static User Interfaces of the any web applications. HTML (Hyper Text Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. HTML has different types of tags like paragraph, division, heading, image, etc. With the help of this tags we can show our content on the web browser. And with the help of CSS (Cascading Style Sheets) we can style that content. CSS describes how elements should be rendered on screen. Some of the most used CSS features are:

- *CSS flexbox*
- *CSS grid*

B. JavaScript

If HTML defines the meaning and structure of web content and CSS styles the content on web page, then JavaScript add the logic to the web pages. It is what makes the web pages alive. It was introduced in 1995 to add logic to the web pages in the Netscape Navigator browser. Before JavaScript, the interfaces were static and had no scope of being dynamic. JavaScript solved this problem, both on the client side and the server side. JavaScript allows us to add interactivity to our web application. It can be used to enhance the functionality like validating forms, displaying animations, running the business logic.

C. React

React is a very popular frontend JavaScript library aimed to simplify the development process of user interfaces. It was initially designed and developed by Facebook for their own internal use, but was then open sourced in 2013. React has gained massive popularity over the years, with a huge developer community backing it. React provides extensive reusability, its building block being *"component"* [3]. Every component has its own state object. This state object is used to write the business logic as well as styling logic into the component. The state of the React components can be

understood as similar to the finite state machine. The rendering and unmounting of the components can be understood by the component lifecycle.

Many frameworks before React were directly manipulating the DOM on each change. This real DOM is what is kept in the browser memory and directly linked to what we see in a page. React uses the *Virtual DOM* to reduce the resources used by the browser when there is any change on a page. When a state is changed in a component, React marks that component as "dirty". Now, React updates the Virtual DOM only of the component marked as dirty. It then runs the diffing algorithm that reconciles any changes made. And finally, React updates the real DOM. Due to this only component that got change will be re render not entire webpage. And hence, React is faster than HTML CSS.

D. Redux

Redux is a JavaScript state management library. It can be used with other frameworks such as Angular and Vue as well, but it integrates very well with React. Very large-scale applications have a huge chunk of state for different components which makes maintainability very difficult and inconvenient. Also, we might have different components rendered in a complex hierarchy, but still using similar state. Redux can be used to solve this problem. Redux has three main constituents:

- *Store*: Redux has the concept of a "single immutable state" called a "store", which cannot be changed directly. This concept is called "single source of truth". When the user does something, then a whole new object is created that determines the new state of the application. We can listen for the events when the store updates.
- *Action*: Action is just a plain JavaScript object that explains what changed and not what needs to be done about that change. We just send this action to the store instance which updates the state of the application. The action requires us to specify a "type" field that is basically used by the reducer to take the appropriate action. Other fields in the action are optional and depend on how we want to use the action in the reducer.
- *Reducer*: Reducers are JavaScript functions that describe how the sent action needs to be processed and how the store should be changed. They recalculate the store and provide a new store for the application (hence, immutable store). A reducer function takes two arguments, previous app state and the action object that is being sent. The reducer then calculates the new state in accordance with what the action object was.

The global store is subscribed to the user interface of our application. Any change in the user interface dispatches (or sends) an action object to the reducer. The reducer then forms a new global state according to the previous state and the action object being sent. The newly created global state then triggers the change in the user interface [4].

E. Version control

Version Control makes an essential part of the development lifecycle. It makes copy of every changes made by all contributors. We can go to the any version of the code at any time. For this purpose, we are using *Git* for our project. The code is hosted in the remote repository hosting website, *Gitea*. Gitea is a painless self-hosted Git service. [5]

F. Gatling

Gatling is an open-source load and performance testing framework based on Scala, Akka and Netty. Load testing with Gatling makes it possible to anticipate high volume traffic. It simulates virtual user in our application to detect performance issues. It gives enough flexibility to customized testing script and hence we can load test according to our requirements [6].

III. Work involved

A. Amazon clone

I never been into Frontend Development. Hence, my mentor asked me to build the same User Interface as amazon website has, I will be calling it as *amazon clone*. There are two advantages behind this, one I can understand the flow of development and second I can get exposure of technologies that are being used by company in current project. Amazon clone that I build has following functionalities:

- Existing User can login to the web application
- If user does not exist then one can register and then login to web application
- User can add items to the cart and in the top bar user can see the number of items that are being added to the cart
- At the checkout page, user can see details of all the items that has been added to cart, user can remove item from the cart and total amount that user needs to pay.

Following are the snapshots of amazon clone that I build:

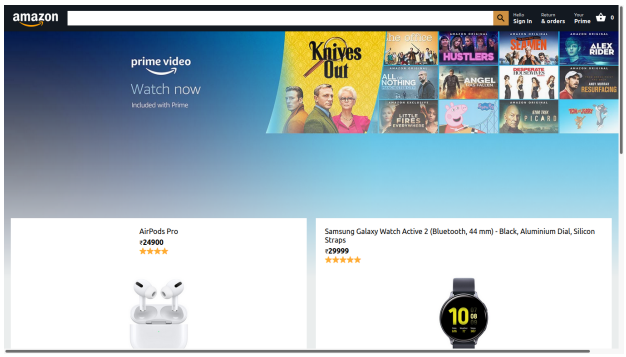

Fig. 1. Home page

Fig. 1 is where user will land when user enters the URL of the site. In Fig.2 we can see that as we scroll down on home page, top bar of the website stuck there and don't get hide by scrolling. Also user can see the number of items in the cart on top bar. In Fig.3 user can see details about items in the cart and also button to remove item from the cart. At right side of snapshot total amount of all the items in the cart is showed.

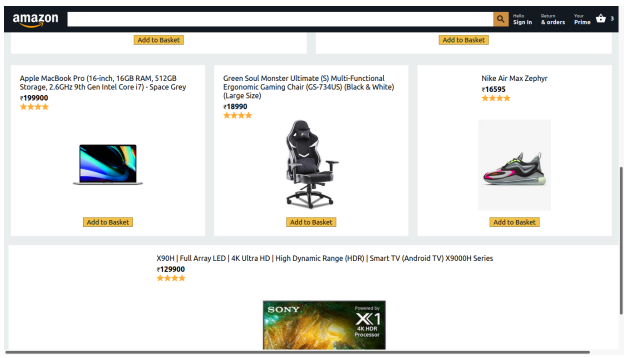

Fig. 2. Home page with stuck top bar

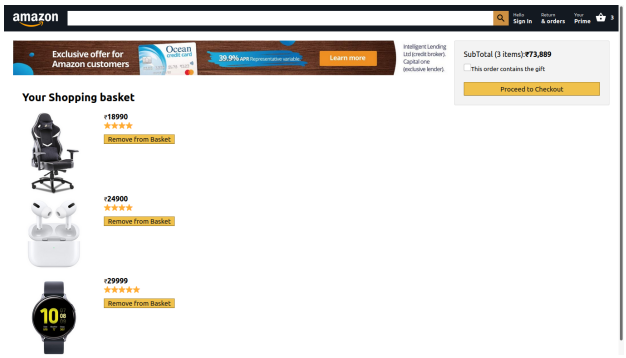

Fig. 3. Checkout page

I developed this amazon clone User Interface with React. For user authentication I used authentication service of the google firebase.

B. Daily Report Page

I learnt a lot while building this amazon clone. After this hands on i was ready for contributing to on going project of Dynamic Reporting tool. After getting the access of codebase *Daily Report* the first actual web page that I build.

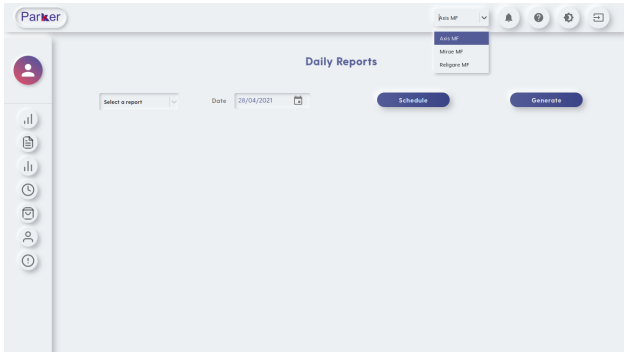

Fig. 4. Light mode

This page has following features:

- User can generate daily report of any date by selecting date from the given date picker
- After giving the report name and date user can either generate the report or schedule the report
- If user click on generate button then preview of the report will be shown at the bottom and download button will appear.
- By clicking on download button user can download the report in chosen format
- In the top bar user can select the mutual fund name for which he wants to generate or schedule a report

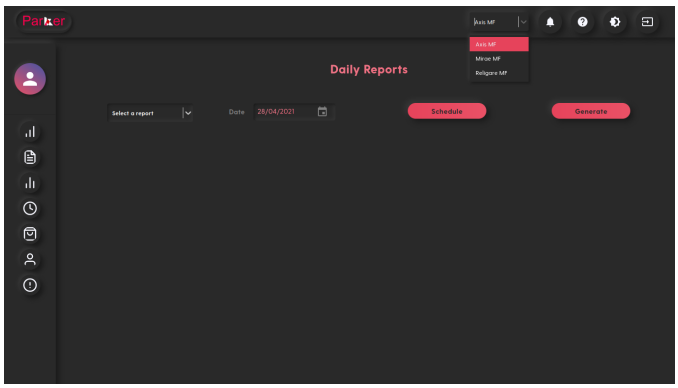

Fig. 5. Dark mode

C. Download Pop-Up

As mentioned above when user clicked on download button a pop-up will be appeared asking to select download format. Once user select the preferred report format and clicked the download button, report will be downloaded. I developed this download pop-up for every type of report in every view such as mobile view, web view, light mode and dark mode as well. I styled it in such a way that it matches with the styles of web application and persists the uniformity among the all pop ups.

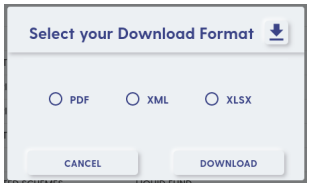

Fig. 6. Light Mode

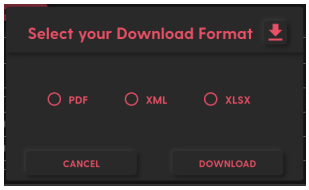

Fig. 7. Dark Mode

D. Loading Spinner

On any report building page whenever user generate the report, it will take some time to fetch the report from backend and display it on frontend.

In this mean time user must be aware of what is happening in the web application and for that reason I added a *Loading Spinner*. So now whenever user click on generate report loading spinner will spin saying loading and hence user will get idea that my report is getting load. Once report get fetched spinner will fade away and preview of the report will be shown to user. Here in Fig.8 I attached a snapshot of light mode.

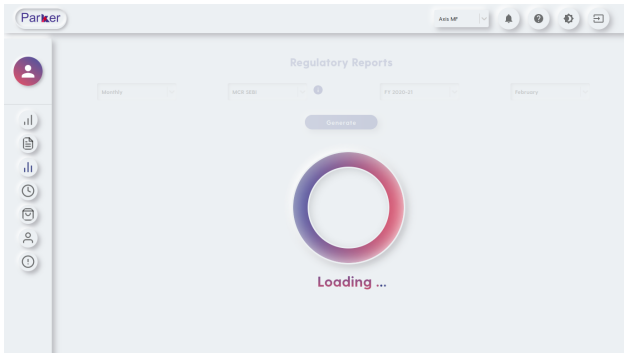

Fig. 8. Loading Spinner

E. Report Bug

As this web application is in initial release state, there might be some bugs found by the user. To report those bug to developer I designed a page called *Report Bug*. In this page user can specify URL of the bug, provide the description of bug, choose the impact and urgency of the issue, upload a screenshot of the bug, etc. Once user saves the details, developer will receive all the details provided by user and then he can resolve the bug.

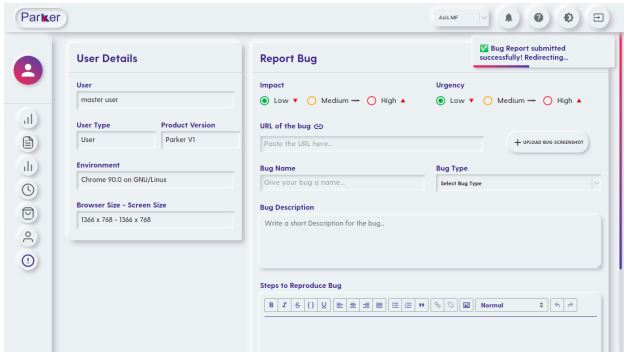

Fig. 9. Light mode

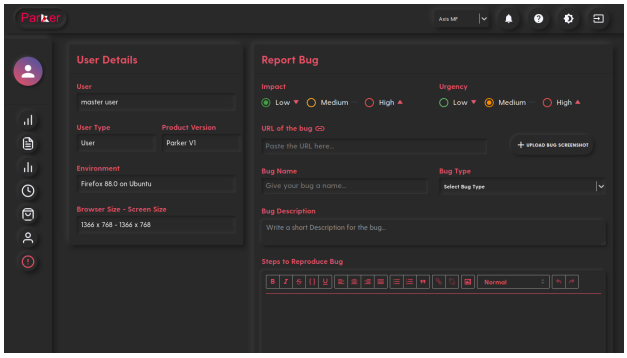

Fig. 10. Dark mode

F. Load Testing on Gatling

As this Dynamic Reporting tool is now ready from the application tier side. So, now we have to test performance of this web application. My mentor asked me to get some exposure of load and performance testing as well so he assigned this task to me. Slowdowns or crashes of websites can make everything go wrong, when lot of users are accessing our website at the same time. It damages customer satisfaction and can impact negatively on our revenues. We can tackle this problem by load testing. There are series of tutorials by Gatling academy on how to operate Gatling and test our web application while multiple concurrent users accessing it. I went through those tutorials and test our web application under different circumstances and scenarios.

IV. Difficulties Faced

- Working on an existing project is much more difficult than working on a project from scratch. Firstly, I had to understand the logic of the existing code and then write the new code.
- The codebase was large due to which finding a particular code file and understand the flow of code is a bit challenging
- After integrating a new feature, I had to rigorously test and make sure that my code fits well with the existing flow of the code and did not changes any other feature of the web application.
- Migrating from the academic level programming practices to industry level programming practices is always challenging

V. Conclusion

Kfintech gave me the total exposure of software development life cycle. From gathering user requirements, developing prototype, developing feature, change in requirements, accommodate changes, testing and then delivered. I have had hands on experience on some of the above mention phases and I witnessed other phases of software development lifecycle. I learnt the best software engineering practices. I got to know how technology is helpful to achieve the business goals especially when data is so much big and privacy is first necessity. I have made a lot of progress as a working professional and learnt to work in a team.

VI. Acknowledgment

I would like to express my gratitude to placement cell of DAIICT for providing me the opportunity to work for Kfintech. I would like to thank my mentor Jagruthnath Kovi for training and guidance in initial onsite training. I am grateful to Nikhil Paruchuru and Abhishek Kumar Lakshmisetti for continues mentoring throughout the internship. I am also thankful to my all teammates for all help and support.

References

[1] https://developer.mozilla.org/en-US/docs/Web/HTML
[2] https://developer.mozilla.org/en-US/docs/Web/CSS
[3] http://www.reactjs.org
[4] http://www.redux.js.org
[5] https://docs.gitea.io/en-us/
[6] https://gatling.io/