IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11768

Dashboard to provide a seamless Banking Experience

Nityam Agarwal

Student, ISE, RVCE, Mysore Rd, RV Vidyaniketan, Post, Bengaluru, Karnataka, India 560059

Abstract: The aim of this work was to create a React based interface to provide users with a seamless banking experience. It is well known that currently there are several legacy applications which run using HTML/CSS at the back end and would need to be improved upon. On a similar note, we have these banking applications which have a dull user interface and are slow to process. The purpose of this paper is to provide an insight into how the banking industry could implement and integrate a better UI experience for it's customers.

Keywords: React, Dashboard, Develop, Reporting, User interface, Jules, Testing

I. INTRODUCTION

It is well known that Banks have been one of the most popular modes of finance transaction for more than a couple of centuries. With the advent of technology, banks now look forward to provide it's customers with a better user experience and save on manual work. This would lead to an improvised productivity, But most of the times, these websites developed are old fashioned and need some rectifications. This paper would showcase how one of the banking applications was developed. This paper would also showcase how a dashboard was built from bottom up and then test cases were added to it to make it more resilient. The end result would be a working React based application deployed on a Kubernetes cluster with calls made to APIs and data being fetched from it.

II. LITERATURE REVIEW

React JS is a very popular JavaScript library among other JavaScript libraries as has been mentioned in [1] In the research paper, an overview of React JS has been provided and it shows everything from the installation of the library to deploying it for general use and how React JS has been the industry leader and being maintained by Meta The authors have also emphasized on the MVC (Model View Controller) model used by React JS. The authors have also shown the applications of React and how it's faster than others like Angular

In [2], the authors have shown the benefits of test automation and which strategy would be best fit to get an optimal test solution . The strategy used by these authors improved their quality of the React- Redux apps that were deployed by them. The strategy was also able to point out which specific units of testing were needed to get the best possible results, The authors mention of using four components that are essential namely the Redux reducers, the React components , the action creators and the utility function. The author goes ahead by mentioning about the types of automated tests, the goals , the code coverage and the behavior driven development to get the automated testing. The author also shows testing done using the mocking library and the built in code coverage of Jest. It is also mentioned how the author was able to set up an environment to get the best possible results. The use of test suites and the optimal number of test cases in every test suits shows how many test cases in a test suite make it fit

The research, [3], looks into how a mobile application has been built using React based on the native , hybrid and web deployments. .Th paper then compares which of these frameworks have been giving the best results in terms of performance. After making a detailed comparison, the authors concluded that React Native has the best results and performs better than the hybrid mode when it come to deploying applications in React.

It's stated that Kubernetes has a 'shared persistent store' with it's components listening to any activity change in the relevant objects as has been stated in [4].In this paper by Google, the authors have extensively shown how containerized applications work and how Kubernetes has been the backbone of the Google Cloud Infrastructure. The paper also compares the open source container management system, Kubernetes to Borg and Omega where such frameworks were initially used

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11768

III. METHODOLOGY

A. Architecture

The architecture used to carry out the project has an index page at it's center with the navigation headers and other pages providing a seamless access to the Banking pages. The Fig 1 shows what the architecture diagram looks like. The diagram shows the dashboard as a container with all of the apps , reporting , development and data in separate JS files. The dashboard also makes use of the React libraries and other services with the React DOM Router ensuring proper flow among these pages.

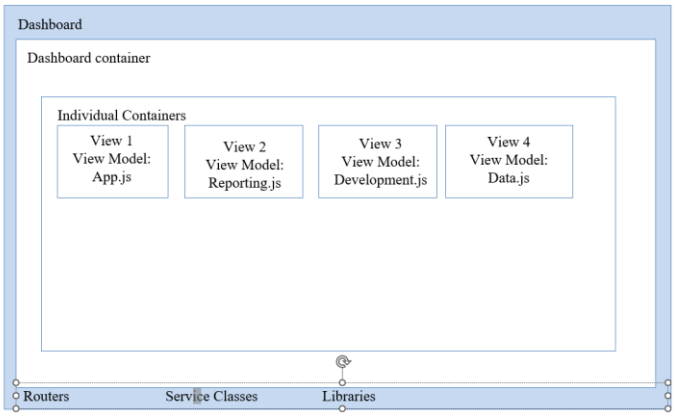

Fig. 1. Architecture Diagram for the dashboard

B. Methodology

Gathering possible updates: Initially, it was required to identify the flaws of the legacy dashboard and the features that could be implemented to improvise the dashboard. After this step, the requirements of the dashboard namely the functional requirements as user log in, gated access to specific users, deployment to a Kubernetes cluster and having all components of the legacy dashboard, and the non-functional requirements namely the performance , availability and consistency of the data were defined

User Authentication: Once the possible updates were defined and the requirements were at hand, the feature of user log in was implemented based on a Kerberos authentication . If the user was a verified one, the user could move on to view the dashboard otherwise the access would be rejected. If the user had an Admin access, the user should also be able to view the buttons in a clickable state

Dashboard Component Implementation: After the verified user was logged in, the components core to the banking industry were displayed. These components like reporting (generating reports) were displayed to the user who could then perform actions like update the report or stop the processing of a report could take place. This was built using the React components of ag-grid and the button which is native to react. These components were further customized to suit the look and feel of the website.

Implementation of Action Buttons: Once the dashboard was available for viewing, if the user had an Admin access the user must be able to edit those actions and bring the changes as needed . These action buttons was able to deliver the results as intended like pausing and stopping report generation or querying the data and exporting it to the user as an excel report.

Testing the dashboard: Once the dashboard was ready, it becomes necessary to test the dashboard. This testing was done using Jest and the components were mocked and the columns of the grid were tested for their resilience. It was also tested based on the user access token type and that if the users are able to access the dashboard as desired.

Deployment to a Kubernetes cluster: The dashboard was then deployed on a docker instance after the code coverage was over 70% . Once the deployment of docker was successful and the build package had all the contents, the dashboard was then deployed to a Kubernetes cluster.

The above summary describes the steps involved in the working of the implementation of the dashboard implemented to provide a seamless banking experience in React.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11768

The Fig 2 shows the activity diagram which highlights the flow of the entire methodology implemented while implementing the dashboard. It shows how the credentials are checked for first and then how based on that the data is fetched while the Progress Bar is displayed. After the data has been fetched, all pages are displayed and proper color coding is followed before testing it and deploying to production.

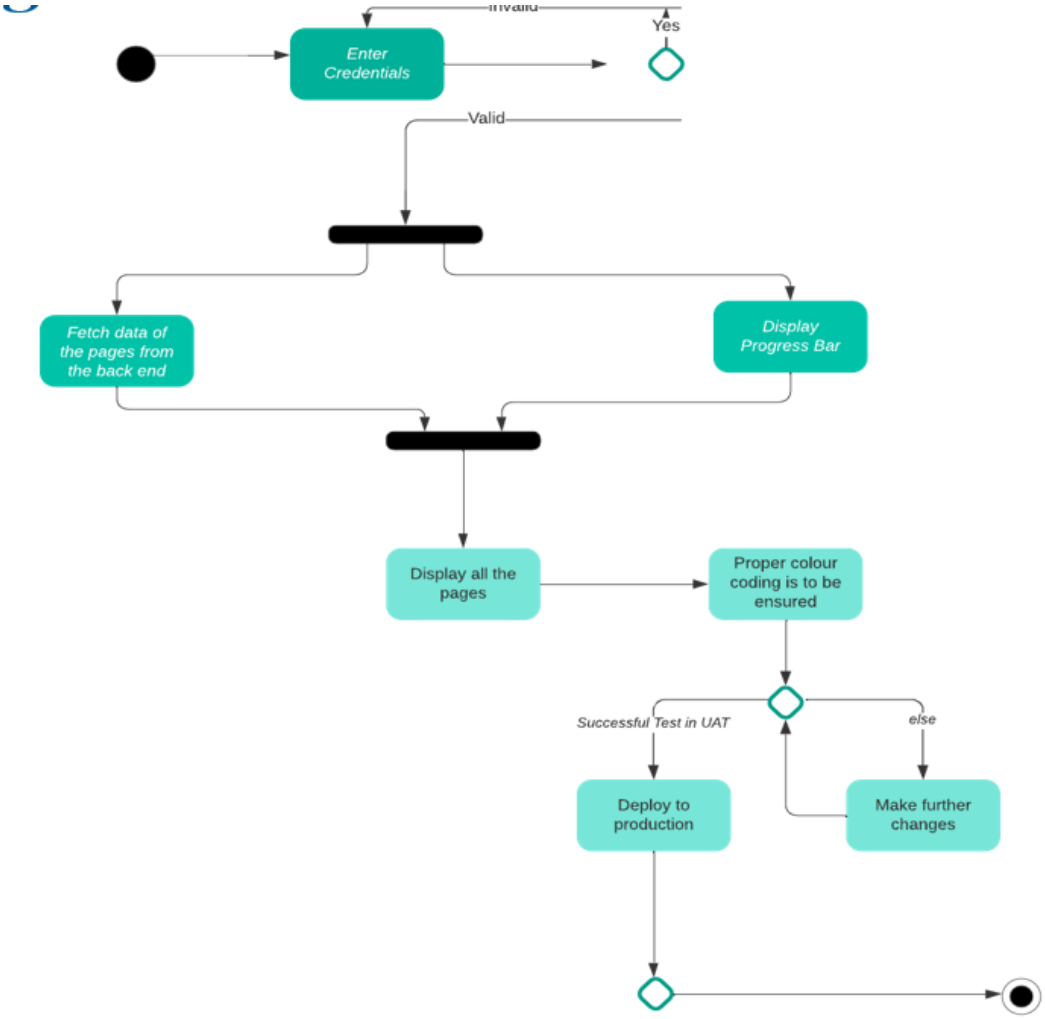

Fig. 2. Activity Diagram showing the flow for the dashboard

IV. RESULTS

The results obtained ensured that the banking experience was a robust one and provided an efficient experience to the users. The final dashboard obtained had a very smooth user interface. It was also found that with the implementation of Kubernetes cluster and React JS, the response time was decreased as compared to the legacy dashboard. The dashboard had the login page as expected and based on the role of the user, the user had a limited access. The read only user were able to view the dashboard but not make any changes to it. The users with admin access had the option to alter the data while those without any grants could not even view the dashboard .The results also ensured that the dashboard was able to deliver data consistently and was available for use to the users.

V. FUTURE DISCUSSIONS

Web technology has the power to retrieve a bank's customers with a better user experience and a smoother user interface. The area of further research could be a time versus memory tradeoff while fetching the APIs. The data fetched from the API calls could be stored locally instead of fetching them over and over but it is also necessary to have only a small size of this data stored in the cache as a larger size would again increase the latency . Hence, a further analysis on time vs memory tradeoff could be one of the possible future discussions.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022
DOI: 10.17148/IJARCCE.2022.11768

VI. CONCLUSION

It can thus be concluded that the websites provided by banks need a serious revamp and React is the best fit front end JavaScript library for the same. It can also be concluded that containerized applications are much faster and with the advent of Kubernetes by Google , it has become even more easier to implement such applications.

ACKNOWLEDGMENT

Prof. Rekha B.S (Assistant Professor) was instrumental in the effective completion of this study, and the author would like to express their gratitude to them.

REFERENCES

[1] Pratik Sharad Maratkar ; Pratibha, "*React JS - An Emerging Frontend JavaScript Library*" iconic Research And Engineering Journals 22-06-2021
[2] "*Unit Test Automation of a React-Redux Application with Jest and Enzyme*" Moroz, Bogdan ScienceDirect , CIRP Conference on Manufacturing Systems,2019
[3] Hugo Brito, Anabelo Gomes, Alvaro Santos, jorge Bernardino "*JavaScript in mobile applications: React native vs ionic vs NativeScript vs native development"* Iberian Conference on Information Systems and Technologies (CISTI) 16 June 2018. https://ieeexplore.ieee.org/document/8399283
[4] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes, Google Inc. (2016). Borg, Omega and Kubernetes. *VOLUME-14 ISSUE-1, MARCH 2016*
[5] Sanchit Aggarwal's "Modern WebDevelopment using React JS" et al. International Journal of Recent Research Aspects | March 2018, Vol. 5, Issue 1, ISSN: 2349-7688
[6] "Learning React Functional Web Development with React and Redux" by Alex Banks and Eve Porcello