JISTE (Journal of Information System, Technology and Engineering)
Volume 1, No. 2, pp. 38-42
E-ISSN: 2987-6117
http://gemapublisher.com/index.php/jiste
Received: April 2023
Accepted: May 2023
Published: June 2023

Implementation of Progressive Web App on Dropship Data Management Application to Anticipate Product Order Errors

Hasbi Taobah Ramdani[1], Nur Ainun[2], Ahmad Muktamar B[3]

STIKes Karsa Husada Garut[1], Universitas Serambi Mekkah [2], Institut Lamaddukkelleng Sengkang [3]

Correspondence Email: hasbiners@gmail.com[1]

Abstract

Digital business trends during the COVID-19 pandemic have increased. One of the digital businesses that can be run during a pandemic, one of which is the dropship business, has a fairly low risk because it does not require large capital and infrastructure preparation. Dropship data management without a data management system can make it difficult for drop shippers to manage and view customer, product, and supplier data. Besides that, without a data management system, it is very likely that the drop shipper will make an error in ordering and shipping the product. This research will discuss the creation of a data management application for drop shipping to cover dropship data management problems. The developed application will be web-based using a Progressive Web App (PWA) so that it can have better performance with a cache and a display that resembles a mobile application.

Keywords: dropship, progressive web app, pwa, vuejs, firebase, service worker.

INTRODUCTION

The mobile market is so broad as its users grow; in Indonesia alone, internet users will reach 202 million in 2021, with a total smartphone user count of 98% compared to 74% of computer or laptop device users. Of the many mobile device users, it can be taken that the average use is for 5 hours a day; this is the underlying reason for many companies with their digital products to enter the mobile device user market. The platforms used for software development can generally be divided into three main categories: desktop, mobile, and web. Each of these platforms has different operating systems and different programming languages for application development. On mobile platforms such as the Android operating system, application development can use the Java or Kotlin programming languages, while those on the iOS operating system can use the Swift or Objective-C programming languages (Native, web, or Hybrid apps). What's The Difference? n.d. Because mobile applications developed using native methods require larger resources, special skills and tools are needed to create native applications, which will have an impact on costs and development time. The development of applications that can be accessed by various platforms, or cross-platform applications, provides savings on application development costs, speeds up development time, and expands application distribution to users because these applications use the same programming language (single codebase).

One solution that can be used to create mobile applications, among others, is to develop web applications with Progressive Web App (PWA). By using PWA, developers can develop applications that can be accessed on various devices using only one codebase, namely HTML, CSS, and Javascript. Making applications using mobile web apps with PWA also tends to be lighter, and application results can be published or accessed on various devices other than smartphones. To provide an experience resembling a native mobile application, the web application must use a Service Worker. These service workers can provide the ability for our web applications to be accessed offline, perform work or retrieve data when the application is not opened, provide notifications, and allow devices to install the web

application. PWA has several advantages compared to other mobile application development methods, namely Hybrid (Cordova/PhoneGap) and Interpreted (React Native). This was done by Andreas in his research by comparing metrics on the results of application testing. This comparison shows that PWA has a smaller size, which is only 104KB. Apart from the size of the application, Andreas also compared the performance that can be achieved in these applications, where PWA can be accessed the fastest with a launch time of only 230 ms. To carry out the implementation and see the test results from using PWA, the author uses one of the problem objects, namely application development, to answer the solutions faced by the author when carrying out the dropship business process.

The author runs a digital business by adopting the dropship business model. This business model was chosen because it makes it easy for students like writers to run a digital business. This business model was chosen by the author because of limited capital and direct product access to producers or product owners. Dropship makes it easier for business people, such as students, who have minimal capital, because the dropship business model does not require a lot of capital to stock goods and support facilities such as warehouses. The author conducted interviews with one of the drop shipper actors and observed directly by doing drop shipping that there were several operational problems in running the business. These problems included: (1) the running business does not use a database that can make it easier to record incoming orders and ongoing orders processed, so that multiple orders often occur; (2) customers cannot see customer data and purchase history in one centralized application; and (3) there are difficulties finding goods for sale because there is no data recording related to products and suppliers who provide products. This research is expected to be able to answer these problems. The authors will try to make a prototype of a dropship data management application. The application will implement a Progressive Web App so that it can see how well PWA performs and the results of this implementation as an alternative to developing mobile applications.

METHOD

Data and information collection were carried out in this study to ensure that the research was relevant and achieved the stated goals. This study uses two approaches to data collection methods: literature study, observation, and interviews. The method used for system development is Rapid Application Development (RAD). This method is used because it has an easy and short development flow, making it suitable for developing applications for testing purposes or not being final. The following is an explanation of the phases carried out in developing CRUD applications for dropship management using RAD: Researchers search for data and information relevant to the dropship system. Information is obtained through various library sources and direct observation of drop shipping practices. This information is then identified as a reference for the system design. The results of the identification of needs carried out in the previous phase are translated into the system through this stage. At this stage, the design of system processes, databases, and system displays is carried out. This process also designs how the application flow will run and how service workers will work. In this implementation phase, application development is carried out based on the system design made previously, and then the application is tested to determine the suitability and size of the resulting application.

RESULT AND DISCUSSION

Black-box testing is done by the author by accessing previously implemented applications. The purpose of this test is to ensure that existing applications can be run and that all existing functions can run according to the expected results. Testing using Lighthouse is carried out on applications that run locally; these tools can be accessed on Chrome devtools. These tools will analyze previously developed websites and provide performance data results for progressive web apps, accessibility, best practices, and SEO. There is data generated by Lighthouse Tools; the data consists of several metrics. These metrics are represented by scale values with values 0–100 and 0–9 in the Progressive Web App metric. The scale recorded on performance, accessibility, best practices, and SEO has a score scale ranging from 0 to 100. The value of this score has several levels, which are also represented by color. Scores ranging from 0 to 49, which are marked in red, indicate that the application has poor performance; scores 50–89 are marked in yellow, indicating that the application has sufficient performance; and scores 90–100 are marked in green, indicating that the application has good performance.

performance This metric will be measured by Lighthouse to see the speed of the application when it is accessed. This metric is accumulated from seven sub-metrics whose weights have been determined by the lighthouse tools themselves. In the previous measurement results with three scenarios, it was found that there was an increase in performance, with the highest score obtained with the scenario where the application was run offline and used a cache of 61/100. These results are in accordance with the results of previous research that explained the effect of PWA on web application development: an increase in performance from originally taking 1769 milliseconds to 85 milliseconds. This is because the files needed by the application are already cached so as to minimize the load time and render time of the application. A more detailed explanation regarding the results of the performance metrics can be seen in each sub-metric, as follows: There has been a decrease in load and rendering times in applications developed using cache and PWA by up to 2x. The first contentful measure of itself is how much time it takes for the content in the DOM (Document Object Model) to be rendered and seen by the user. This sub-metric weighs 10% and is recommended to have a load time of under 1.8 seconds, according to Google. The results obtained in this sub-metric explain how fast it takes for the content to appear as a whole on the user's screen. This sub-metric weighs 10%, and it is recommended that the application tested get results of less than 4.3 seconds on the speed index metric test. Similar tests were carried out, showing an increase in the speed of load time for content in web applications that use PWA. The test results with the PWA show a performance difference of 24% against the speed index test.

The largest contentful paint on the performance test metric has a weight of 25%. This sub-metric is similar to the first content paint sub-metric, except that it measures how quickly the main content loads onto the user's screen. The main content referred to in this sub-metric is content in the form of images, videos, and DOM components that have text in them. It is recommended that this sub-metric have a load score of less than 4 seconds. Using PWA and cache can provide much better performance. This is because the images needed by the application are already stored on the device, so the application does not need to re-request and wait for a response from the server. Time to interactive has a weight of 10% on the performance metric; the recommended time limit for applications to be able to fully interact is less than 7.3 seconds. This metric can be measured by how long the initial content can appear (the first contentful paint) and how the application can respond to user interactions. The total blocking time sub-metric has the highest weight among the other performance sub-metrics, namely 30%. This metric measures how long the application is unable to respond to interactions from the user. It is recommended that this sub-metric have a score of less than 600 milliseconds. Every website development project has standard guidelines that help website application developers avoid some of the common mistakes that often occur in application development. In Best Practices, Lighthouse will check or audit the following description of the Best Practices test results: SEO (Search Engine Optimization) This metric measures how well our application can be found and seen by search engines. This sub-metric measures how well the application can be accessed by users with disabilities. PWA metrics on Lighthouse perform audit checks to ensure that the website application meets aspects of the Progressive Web App.

In this sub-chapter, a network test is carried out using the tools available in the Microsoft Edge browser. This test is carried out to find out how fast and how much data is downloaded by users. Testing is carried out using several test scenarios, including: In the first scenario, the application is opened for the first time when the service worker and cache are not yet available. The first time the application was run in cache and network services were not yet available, it made 122 requests, sending 6.1kb of data and receiving 3.8 mb of data from the client. Application access can be completed in 1.24 seconds, and the rendering process takes 622 milliseconds. The second scenario is carried out in conditions where the application has already been run before. The application already has a running service worker and cache; with a cache and service worker, it is hoped that the application can run better. Here are the test results for the second scenario: The performance obtained is much better than before using service workers and caches. As can be seen in the image above, the application only made 102 requests, but there was an increase in the data transferred, which was 5.7kb. Other than that, the application can be finished loading within 1 second, which is 0.24 seconds faster than the start of the application. Render time is also faster, at around 577 milliseconds. In this second scenario, the assets needed by the application have been stored in the application. The third scenario is carried out to find out the difference in performance when the service worker is not using cache but is still running.

So that the application does not use the cache, there are several ways that can be done, namely by disabling the cache in devtools or deleting the application cache. The application tries to send requests

as large as in the first scenario; in the results of this third scenario, the application makes 119 requests and sends 1.1 MB of data. Application data has increased from 3.8 MB to 5.5 MB, and besides that, the application takes 1.96 seconds to load and 760 milliseconds to render. In the fourth scenario, the application will be accessed in the same conditions as in the second scenario, but there are differences in the network's ability to access the application. The network to be used is a 3G network; this is done to see how fast the application can be accessed in unstable or bad network conditions. According to the results of the fourth scenario test on a slow network, the application still makes 115 requests and sends 1.7kb of data. The application can render in 525 milliseconds and takes 2.93 seconds to load. When compared to the third scenario, the rendering time is somewhat faster because assets such as images and static files are taken from the cache, so the network has no effect on the speed of the application to be accessed. But on a slow network, the load time may take longer.

In this scenario, it is done to see how well the application can be accessed when not using a network; this can also prove whether a web application using a Progressive Web App can be run without using a network. To turn off the network, we can disconnect the network from the device or use the offline option in devtools. The following is the result of the fifth scenario test when accessing the application when it is not connected to the internet. The application can still be accessed in conditions without a network; there are 117 requests and 1.3kb of data that the application is trying to send. Because the application is not connected to the internet network, the request will automatically fail, but here the service worker will handle the request and retrieve data from the cache. As seen in the test results of the fifth scenario, the application can render and load faster than the previous scenario. This is because the application does not wait for a response from the server, but the service worker directly diverts requests and returns responses from local storage. Web-based applications using Progressive Web Apps and cache can provide increased access performance, and applications no longer require a network to run.

The result of implementing the application display is to display the Register page. On this page, new users can register their accounts before getting access to the application. New users need to fill out several data forms, such as store name, store location, email, and password. After all the fields are filled in, the user needs to send the data by pressing the "Create Account" button. After having an account or just registering an account, you will be directed to the "Login" page to verify before entering the main application. This page will reappear if the user logs out of the application. After filling in the account data form, the user needs to press the "Login" button. If the account is successfully verified, it will be redirected to the "Home" page, and if the account cannot be verified, an error statement will appear. then the application will display the "Home" page; this page will appear every time the user accesses the application as long as the account data is still stored on the device. On this page, we can see or quickly access the menu in the application.

It is recommended for users to create supplier data first. To access the "Supplier" page, there is a navigation menu available under the application; users can select the menu marked "Store" or access it quickly via the "Supplier" menu on the icon menu below the data dashboard. After entering the "Store" page, select the "Supplier" menu below. There are two options, namely "Customer" and "Supplier". Users can see a list of suppliers that have been recorded in the application. To add new supplier data, we can press the button with a person's image next to the supplier search column. There are several columns that need to be filled in by the user to add supplier data. Supplier data is needed before making products and orders. The "Customer" page will display a list of customers who have made order transactions at the user's store. In addition to displaying a list of customer names, there is data on how many orders the customer has placed before. Furthermore, to add new customers, users can press the button with the person icon next to the search field. As on the Customer and Supplier pages, on the "Product" page, users can see a list of products marketed by users (drop shippers). The product column displays brief information about the product and contains a product photo (if any). To add a new product, you can do it by pressing the plus button next to the search field. On the "Orders" page, users can see a list of incoming and completed orders. There are several statuses that exist for each order. The order list will display order-related information such as the customer, the total order price, and the date the order was made. For further information, see the "Order Details" page, where complete information regarding orders can be seen, including order status, delivery data, customer data, supplier data, and products in the order. To change the order status, the user can click on the "Confirm" button. The "Confirm" button will ask the user to confirm that the order has made the order or delivery process as stated in the application. After the order has been processed and sent by the supplier, the user (drop

shipper) needs to enter a receipt that matches the receipt provided by the supplier or expeditionary party.

Orders that have been entered as receipts will automatically change the order status to "Shipping". After the order is completed or has arrived at the buyer, the user can confirm that the order has been completed by pressing the "Confirm" button; thus, the order will be considered complete, and the completed order data will automatically be recorded in the data dashboard. On this page, the user can view information about their account. This data includes profile photos, store names, store locations, and store links. At the beginning of creating an account, the profile photo and shop link are not available, so they need to be filled in first. Also on this page, users can remove their account from the dropship management application. After exiting, the application will delete the data stored on the device.

CONCLUSION

Progressive web apps can be used as an alternative to making mobile applications. The use of PWA in web applications can increase performance and provide the ability for applications to be accessed in conditions without an internet network. Drop shippers can be helped by this data management application, so that data such as orders, products, customers, and suppliers can be stored neatly using the application. In addition, the drop shipper can also find out the progress of each order. The use of PWA and VueJS gives a display resembling the appearance of a mobile application. The Progressive Web App has many features that can be implemented; it is hoped that similar research can discuss all the features in the Progressive Web App. The dropship management application developed is still limited to one single actor and is still ineffective for commercial use; it needs integration with online stores and delivery service providers. Web applications that use the Progressive Web App can be used as APKs; it is hoped that, with further research, these PWA-based applications can be presented on the Google Play Store.

REFERENCES

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T. M. (2017). Progressive web apps: The possible web-native unifier for mobile development. *WEBIST 2017 - Proceedings of the 13th International Conference on Web Information Systems and Technologies*, 344–351. https://doi.org/10.5220/0006353703440351

Hadi, R. (2018). Analysis of the practice of buying and selling Drops hipping in the Perspective of Islamic Economics. *Journal of Chemical Information and Modeling*, 53(9), 1689–1699. https://doi.org/10.1017/CBO9781107415324.004

Hanumant Pawar, S., Sahebrao Pacharne, R., Shyamkant Jadhav, D., & Deepak Sonawane, M. (2020). Impact of Progressive Web Apps on Web App Development. *International Journal of Innovative Research in Science*, 9(9). https://doi.org/10.15680/IJIRSET.2018.0709021

Lestari, I., & Trisnadoli, A. (2017). Usulan Model Kualitas Aplikasi Context Aware Mobile: Family Tracking pada Hybrid - Cross Platform. *Jurnal Komputer Terapan*, 3(2), 261–270.

Mercado, I. T., Munaiah, N., & Meneely, A. (2016). The impact of cross-platform development approaches for mobile applications from the user's perspective. *WAMA 2016 - Proceedings of the International Workshop on App Market Analytics, Co-Located with FSE 2016*, 43–49. https://doi.org/10.1145/2993259.2993268

Waworuntu, A. (2020). Design and Build Web-Based Dropship E-Commerce Applications. *Ultimatics: Jurnal Teknik Informatika*, 12(2), 118–124. https://doi.org/10.31937/ti.v12i2.1823