# Implementasi *Progressive Web App* pada Aplikasi Manajemen Data *Dropship*

**Skripi**


Oleh

**RIFALDI KUSNAWAN**

NIM : 11160910000081


**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SYARIH HIDAYATULLAH**

**JAKARTA**

**1443 H – 2021**

# Implementasi *Progressive Web App* pada Aplikasi Manajemen *Data Dropship*

**Skripi**

Diajukan sebagai salah satu syarat untuk
memperoleh gelar Sarjana Ilmu Komputer (S.Kom)


Oleh

**RIFALDI KUSNAWAN**

NIM : 11160910000081

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SYARIH HIDAYATULLAH**

**JAKARTA**



**1443 H – 2021**

## PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa:

1. Skripsi ini merupakan hasil karya asli saya yang diajukan untuk memenuhi salah satu persyaratan memperoleh gelar Strata 1 di UIN Syarif Hidayatullah Jakarta.
2. Semua sumber yang saya gunakan dalam penulisan ini telah saya cantumkan sesuai dengan ketentuan yang berlaku di UIN Syarif Hidayatullah Jakarta.
3. Jika dikemudian hari terbukti bahwa karya ini bukan hasil karya asli saya atau merupakan hasil jiplakan dari karya orang lain, maka saya bersedia menerima sanksi yang berlaku di UIN Syarif Hidayatullah Jakarta.

Jakarta, 15 Oktober 2021


Rifaldi Kusanwan

11160910000081

## LEMBAR PERSETUJUAN PUBLIKASI KARYA SKRIPSI

Sebagai sivitas akademik UIN Syarif Hidayatullah Jakarta, saya yang bertanda tangan di bawah ini:

| | |
|---|---|
| Nama | : RIFALDI KUSNAWAN |
| NIM | : 11160910000081 |
| Program Studi | : Teknik Informatika |
| Fakultas | : Sains dan Teknologi |
| Jenis Karya | : Skripsi |

Demi pembuatan ilmu pengetahuan menyetujui untuk memberikan kepada UIN Syarif Hidayatullah Jakarta **Hak Bebas Royalti Non-ekslusif *(Non-Exlusive Royalti Fee)*** atas karya ilmiah yang berjudul :

**"IMPLEMENTASI *PROGRESSIVE WEB APP* PADA APLIKASI MANAJEMEN DATA *DROPSHIP*"**

Dengan Hak Bebas Royalti Non-Ekslusif ini UIN Syarif Hidayatullah Jakarta berhak menyimpan, mengalih media / formatkan, mengelola dalam bentuk pangakalan data, merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemilih Hak Cipta. Demikian pernyataan ini saya buat sebenarnya.

Dibuat di: Jakarta

Pada Tanggal: 15 Oktober 2021

**Yang Menyatakan**

**( RIFALDI KUSNAWAN)**

| | | |
|---|---|---|
| **Nama** | **:** | **RIFALDI KUSNAWAN** |
| **Program Studi** | **:** | **Teknik Informatika** |
| **Judul** | **:** | **Implementasi *Progressive Web App* pada Aplikasi Manajemen Data *Dropship*** |


## ABSTRAK

Tren bisnis digital selama pandemi Covid-19 mengalami kenaikan, salah satu bisnis digital yang dapat dijalankan pada kondisi pandemi salah satunya adalah bisnis drophip, bisnis ini memiliki risiko yang cukup rendah karena tidak memerlukan modal dan persiapan infrastruktur yang besar. Pengelolaan data *dropship* tanpa sistem manajemen data dapat membuat *dropshipper* akan sulit untuk mengelola dan melihat data *customer*, produk dan *supplier*. Selain itu dengan tidak sistem adanya manajemen data akan sangat besar kemungkinan *dropshipper* melakukan kesalahan pemesanan dan pengiriman produk. Penelitian ini akan membahas pembuatan aplikasi manajemen data pada *dropship* untuk menutupi masalah pengelolaan data *dropship*, aplikasi yang dikebangkan akan berbasis web menggunakan *Progressive Web App* (PWA) sehingga aplikasi dapat memiliki peforma lebih baik dengan adanya *cache* dan tampilan yang menyerupai aplikasi *mobile*.



| | |
|---|---|
| **Kata Kunci** | : *dropship*, *progressive web app*, pwa, vuejs, firebase, *service worker* |
| **Jumlah Pustaka** | : 42 (24 Jurnal + 6 Buku + 12 Artikel) |
| **Jumlah Halaman** | : VI Bab + xvii Halaman + 117 Halaman + 76 Gambar + 19 Tabel |

| | | |
|---|---|---|
| **Name** | **:** | **RIFALDI KUSNAWAN** |
| **Department** | **:** | **Informatic Engineerings** |
| **Title** | **:** | ***Implementation Progressive Web App for Dropship Data Management*** |

## ABSTRACT

The trend of digital business during the Covid-19 pandemic has increased, one of the digital businesses that can be run during a pandemic is the dropshipping business, this business has a fairly low risk because it does not require large capital and infrastructure preparation. Dropshipping data management without a data management system can make it difficult for dropshippers to manage and view customer, product and supplier data. In addition, in the absence of a data management system, it is very likely that the dropshipper will make an error in ordering and shipping products. This study will discuss the creation of a data management application on dropship to cover dropship data management problems, the application developed will be web-based using a Progressive Web App (PWA) so that the application can have better performance with a cache and display that resembles a mobile application..

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

# KATA PENGANTAR

Syukur Alhamdulillah kami panjatkan puji dan syukur kepada Allah SWT atas segala nikmat dan rahmat yang telah diberikan kepada kami, baik secara material dan moral melalui hamba-Nya sehingga penulisan tugas akhir skripsi ini dapat diselesaikan. Penulisan skripsi ini dilakukan sebagai salah satu syarat untuk mendapatkan gelar Sarjana Komputer pada Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta. Saya ucapkan terima kasih kepada seluruh pihak yang telah membantu dan terlibat selama masa perkuliahan dan penulisan skripsi ini, semoga Allah mengganti dan membalas kebaikan dan waktu yang telah diluangkannya, Aamiin. Oleh karena itu saya saya ucapkan terima kasih kepada:

1. Prof. Dr. Amany Lubis, MA selaku rektor UIN Syarif Hidayatullah.
2. Ir. Nashrul Hakiem, S.Si., M.T., Ph.D selaku Dekan Fakultas Sains dan Teknologi.
3. Dr. Imam Marzuki Shofi, MT selaku Ketua Program Studi Teknik Informatika dan dosen pembimbing akademik.
4. Husni Teja Sukmana, S.T., M.Sc, Ph.D dan Nurhayati.Ph.D selaku dosen pembimbing yang telah meluangkan waktunya dan mengarahkan penulis dalam penyusuhan skripsi ini dari awal hingga akhir.
5. Fitri Mintarsih, M.Kom dan Nurul Faizah Rozy, MTI, selaku dosen penguji yang telah memberikan penilaian dan masukan.
6. Seluruh dosen dan staff UIN Jakarta, khususnya Fakultas Sains dan Teknologi.
7. Orang Tua ku yang telah sabar dan memberikan dukungan.
8. Ririn Rianti yang selalu memberikan semangat dan inspirasi bagiku.s

Sekian yang dapat saya sampaikan di dalam Kata Pengantar ini, semoga Allah membalas kebaikan dari seluruh pihak yang terlibat. Semoga skripsi ini dapat memberikan kebaikan dan manfaat bagi kita semua, Aamiin.

Jakarta, 15 Oktober 2021

**RIFALDI KUSNAWAN**
NIM: 11160910000081

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

Pasar *mobile* begitu luas seiring dengan pertumbuhan penggunanya, di Indonesia sendiri pengguna internet mencapai 202 juta pada 2021 dengan total pengguna smartphone 98% berbanding 74% dengan pengguna perangkat komputer atau laptop. Dari sekian banyak pengguna perangkat *mobile* tersebut dapat diambil rata – rata penggunaan selama 5 jam sehari, hal tersebut menjadi alasan yang medasari jika banyak perusahaan dengan produk digital mereka masuk ke dalam pasar pengguna perangkat *mobile* (Simon Kemp, 2021).

Jurnal yang ditulis oleh (Lestari & Trisnadoli, 2017) menyebutkan bahwa *platform* yang digunakan untuk pengembangan perangkat lunak umumnya dapat dibagi menjadi tiga *platform* utama, yaitu *desktop*, *mobile* dan *web*. Setiap *platform* tersebut memiliki berbagai sistem operasi yang berbeda serta bahasa pemrograman yang berbeda dalam pengembangan aplikasinya. *Platform mobile* seperti sistem operasi android, pengembangan aplikasi pada sistem operasi tersebut dapat menggunakan bahasa permrograman Java atau Kotlin, sedangkan pada sistem operasi iOS dapat menggunakan bahasa pemrograman Swift atau Objective-C (*Native, Web or Hybrid Apps? What's The Difference?*, n.d.). Sehingga aplikasi *mobile* yang dikembangkan menggunakan metode *native* membutuhkan sumber daya yang lebih besar, diperlukan keahlian khusus dan *tools* untuk membuat aplikasi native, dengan begitu akan berdampak pada biaya dan waktu pembuatan (Mercado et al., 2016). Pengembangan aplikasi yang dapat diakses oleh berbagai macam *platform* atau aplikasi *cross-platform* tersebut memberikan penghematan biaya pengembangan aplikasi, mempercepat waktu pengembangan dan memperluas distribusi aplikasi kepada pengguna dikarenakan pengembangan aplikasi tersebut menggunakan satu bahasa pemrograman yang sama (*single-codebase*) (Fredrikson, 2018).

Salah satu solusi yang bisa digunakan untuk membuat aplikasi *mobile* antara lain yaitu dengan mengembangkan aplikasi web dengan *Progressive Web App* (PWA). Dengan menggunakan PWA, pengembang dapat mengembangkan aplikasi yang dapat diakses di berbagai perangkat dengan hanya menggunakan satu *codebase* yaitu HTML, CSS dan Javascript (Rojas, 2019). Pembuatan aplikasi menggunakan *mobile web app* dengan PWA juga cenderung lebih ringan serta hasil aplikasi dapat di-*publish* atau diakses ke berbagai perangkat selain *smartphone*. Untuk memberikan pengalaman menyerupai aplikasi *native mobile,* aplikasi *web* tersebut harus menggunakan *Service Worker*. *Service worker* tersebut dapat memberikan kemampuan applikasi *web* kita dapat diakses secara *offline*, melakukan pekerjaan atau mengambil data pada saat aplikasi sedang tidak dibuka, memberikan notifikasi dan mengizinkan perangkat untuk meng-*install* aplikasi *web* tersebut (Sheppard, 2017).

Terdapat beberapa penelitian yang menggunakan teknologi *Progressive Web App* (PWA) untuk mengembangkan *cross-platform mobile,* penelitian yang ditulis oleh (Adi, 2017) dengan judul "Platform e-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps", di dalam penelitian tersebut dijelaskan bahwa penggunaan PWA memberikan efisiensi untuk me-*loading* halaman 26% lebih cepat karena terdapat *cache* yang diatur oleh *service worker, dapat* memenuhi kebutuhan fungsionalitas dalam *platform e-learning* yang dikembangkan dan memberikan tampilan yang optimal layaknya aplikasi *native* pada *mobile* baik saat terkoneksi ke internet atau sedang *offline*. Penelitian serupa dilakukan oleh (Karim, 2016) menjelaskan bahwa PWA merupakan salah satu teknologi *web* terkini untuk mengembangkan aplikasi dengan menyerupai *native mobile* melalui teknologi standar web. Penelitian yang dilakukan Karim tersebut menjelaskan bahwa PWA memiliki tiga komponen penting seperti *Manifest, Service worker* dan *App shell.* Penelitian yang dilakukan oleh (Karim, 2016) juga menjelaskan kelebihan dari PWA, antara lain *App-Like* atau memberikan pengalaman layaknya aplikasi *native* melalui interaksi dan navigasi antarmukanya, *Safe* yang berarti aman karena menggunakan protocol HTTPS dan *Connectivity-Independent* yaitu terdapat *service worker* yang membantu aplikasi agar dapat

diakses ketika tidak terkoneksi jaringan internet. Disamping kelebihannya, PWA juga memiliki kekurangan seperti keterbatasan akses terhadap sensor dan komponen perangkat keras yang ada pada perangkat *mobile*.

PWA memiliki beberapa keunggulan dibandingkan metode pembuatan aplikasi mobile lainnya yaitu *Hybrid* (Cordova/Phonegap) dan *Interpreted* (React Native), hal tersebut dilakukan oleh Andreas dalam penelitiannya dengan membandingkan metrik – metrik pada hasil pengujian aplikasi. Perbandingan tersebut menunjukan menunjukan bahwa PWA memiliki ukuran yang lebih kecil yakni hanya 104KB. Selain ukuran aplikasinya, Andreas juga membandingkan peforma yang dapat dicapai pada aplikasi tersebut dimana PWA dapat diakses tercepat dengan waktu launch hanya 230ms (Biørn-Hansen et al., 2017). Untuk melakukan implementasi dan melihat hasil uji dari penggunaan PWA, penulis melakukan mengambil salah satu objek masalah yaitu terhadap pengembangan aplikasi untuk menjawab solusi yang dihadapi penulis ketika melakukan proses bisnis *dropship*.

Label 1. 1 Perbandingan Peforma Hybrid, Intrepreted dan PWA
Sumber: (Biørn-Hansen et al., 2017)

| Metrik | Hybrid | Intrepreted | PWA |
|---|---|---|---|
| Ukuran Intalasi | 4.35Mb | 16.39Mb | 104Kb |
| Waktu Memulai | 860ms | 246ms | 230ms |

Penulis menjalankan bisnis digital dengan mengadopsi model bisnis *dropship*, model bisnis ini dipilih karena memberikan kemudahan bagi mahasiswa seperti penulis untuk menjalankan bisnis digital. Model bisnis ini dipilih oleh penulis karena keterbatasan modal dan akses produk secara langsung terhadap produsen atau pemilik produk. *Dropship* mempermudah pelaku bisnis seperti mahasiswa yang memiliki modal minim, dikarenakan model bisnis *dropship* tidak memerlukan modal banyak untuk melakukan stok barang dan sarana pendukung seperti gudang. Penulis melakukan wawancara dengan salah satu pelaku *dropshipper* dan mengobservasi secara langsung dengan melakukan *dropship*,

terdapat beberapa masalah operasional dalam menjalankan bisnis tersebut, masalah tersebut antara lain, (1) bisnis yang berjalan tidak menggunakan basis data yang dapat mempermudah mencatat pesanan masuk dan pesanan yang sedang diproses sehingga sering kali terjadi pesanan ganda, (2) tidak bisa melihat data *customer* dan histori pembelian pada satu aplikasi terpusat, (3) terjadi kesulitan menemukan barang yang dijual karena tidak adanya pencatatan data terkait produk dan *supplier* yang menyediakan produk. Penelitian ini diharapkan dapat menjawab masalah tersebut,penulis mencoba membuat prototipe aplikasi manajemen data *dropship*, aplikasi tersebut akan mengimplementasikan *Progressive Web App* agar dapat melihat seberapa baik peforma PWA serta hasil dari implementasi tersebut sebagai alternatif pengembangan aplikasi *mobile*.

Berdasarkan latar belakang yang dijelaskan di atas, penulis tertarik untuk melakukan penelitan dengan judul "***Implementasi Progressive Web App pada Aplikasi Manajemen Data Dropship***".

## 1.2. Rumusan Masalah

Berdasarkan masalah yang sudah dijelaskan pada latar belakang penulisan, maka penulis merumuskan masalah menjadi sebagai berikut:

1. Bagaimana membangun aplikasi dropship dengan menggunakan Progressive Web App?
2. Bagaimana hasil uji dari aplikasi yang menggunakan Progressive Web App
3. Apakah service woker pada Progressive Web App dapat memberikan kemampuan *offline* pada aplikasi berbasis web?

## 1.3. Batasan Masalah

Penelitian dibatasi lingkup penelitiannya agar isi penelitian ini bisa berfokus pada masalah dan solusi terhadap rumusan masalah yang dijelaskan sebelumnya dan tidak menyimpang dari ranah penelitian yang ditargetkan:

1. Aplikasi yang dikembangkan berupa aplikasi CRUD (Create, Read, Update Delete).

2. Penyimpanan server menggunakan layanan Firebase Firestore dan Firebase Storage.
3. Aplikasi diuji menggunakan browser pada komputer dan perangkat *mobile* android.
4. Menggunakan teknologi web seperti html, css dan javascript.
5. *Framework user interface* menggunakan vuejs dan vuex sebagai *state management*.
6. *Library Workbox* digunakan untuk membantu menerapkan *Service Worker* pada *Progressive Web App*.
7. Komputer sudah ter-*install nodejs.*
8. Penelitian tidak membandingkan dengan aplikasi yang dikembangakan secara *native.*
9. Aplikasi tidak menggunakan integrasi terhadap *platform* seperti marketplace.
10. Aplikasi ini digunakan oleh dropshipper untuk pencatatan data.
11. Penelitian ini menitikberatkan pada pemanfaatan PWA, proses bisnis *dropshipper* sebagai salah satu objek penerapan dari PWA.

## 1.4. Tujuan Penelitian

Adapun tujuan dasar penulisan karya ilmiah ini yaitu untuk mengembangkan aplikasi yang dapat membantu proses penyimpanan data *supplier*, *customer*, produk dan pesanan pada proses bisnis *dropship*. Adapun tujuan khusus dari penelitian ini antara lain:

1. Membuat aplikasi manajemen data dropship dengan teknologi berbasis web menggunakan Progressive Web App.
2. Mengetahui hasil implementasi dari pembuatan aplikasi menggunakan Progressive Web App dan Vuejs.
3. Mengetahui hasil uji peforma pada aplikasi yang menggunakan Progressive Web App.

## 1.5. Manfaat Penelitian

### 1.5.1. Bagi Penulis

1. Sebagai salah satu syarat untuk lulus Pendidikan Strata Satu (S1).
2. Meningkatkat pemahaman dan mempraktikan ilmu yang sudah dipelajari selama menempuh pendidikan kuliah.

### 1.5.2. Bagi Universitas

1. Menambah daftar koleksi dan referensi UIN Syarif Hidayatullah Jakarta
2. Sebagai alat ukur dan evaluasi pendidikan yang dijalankan selama masa perkuliahan.

### 1.5.3. Masyarakat

1. Memberikan alternatif pembuatan aplikasi *mobile*.
2. Memberikan pemahan tentang Vuejs dan *Progressive Web App*.
3. Memberikan solusi pencatatan data pada pelaku bisnis *dropship*.

## 1.6. Metodologi Penelitian

### 1.6.1. Metode Pengumpulan Data

1. Studi Pustaka
2. Observasi
3. Wawancara

### 1.6.2. Metode Pengembangan Sistem

Dalam mengembangkan aplikasi menggunakan Vuejs dan PWA, penulis menggunakan metode siklus pengembangan Rapid Application Development (RAD), dengan beberapa tahapan sebagai berikut:

1. Tahap *Requirement Planning*
2. Tahap *Design System*
3. Tahap *Implementation*

## 1.7. Sistematika Penulisan

**BAB I PENDAHULUAN**

Bab ini menjelaskan latar belakang masalah yang mendasari penelitian, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penulisan, dan sistematika penulisan penelitian.

**BAB II LANDASAN TEORI**

Bab ini menjelaskan landasan teori yang berkaitan dengan penelitian yang akan dibahas mulai dari pengertian, fungsi dan manfaat.

**BAB III METODOLOGI PENELITIAN**

Bab ini menjelaskan tentang metode dan tahapan yang digunakan peneliti dalam menyusun penelitian. Metode yang digunakan terbagi menjadi pengambilan data dan pengembangan sistem. Selain itu di dalam bab ini akan menggambarkan kerangka alur penelitian ketika menyusun penelitian ini.

**BAB IV IMPLEMENTASI**

Bab ini menjelaskan secara detil hasil rancangan yang dikembangkan, struktur aplikasi, penerapan *vuejs* dan *service worker* yang diterapkan pada aplikasi manajemen data *dropship*.

**BAB V HASIL DAN PEMBAHASAN**

Bab ini menampilkan hasil implentasi. Membahas pengujian dan hasil uji aplikasi.

**BAB VI PENUTUP**

Bab ini berisi kesimpulan jawaban dari rumusan malah dan saran dari penulis agar penelitian serupa bisa menutupi kekurangan penulisan sebelumnya.

# BAB II
# LANDASAN TEORI

## 2.1. *Dropship*

### 2.1.1. Definisi *Dropship*

Penelitian ini ditulis ketika pandemi virus Covid-19 mewabah, hal tersebut menyebabkan penurunan kegiatan ekonomi secara drastis dan menimbulkan pemutusan pekerja yang besar. Wabah tersebut menjadikan tren bisnis *online* semakin diminati, salah satunya adalah skema bisnis *online dropship* karena dinilai cocok untuk kondisi tersebut. *Dropship* merupakan skema bisnis atau proses penjualan barang dengan bermodalkan foto dan deskripsi produk tanpa perlu melakukan penyetokan barang sendiri, produk yang dijual didapatkan dengan membuat kesepakatan melalui penyedia barang atau *supplier* (Syafii, 2013). Skema *dropship* menggunakan media digital sebagai media utama untuk melakukan promosi dan mencari konsumen, media digital yang umumnya digunakan sebagai sarana adalah media sosial seperti Facebook, Instagram dan Whatsapp. *Dropshipper* akan memasang foto, spesifikasi dan deskripsi produk ke media promosi untuk menarik konsumen (Hadi, 2018).

### 2.1.2. Mekanisme Transaksi *Dropship*

Transaksi *dropship* dilakukan melalui tiga pihak yaitu *konsumen*, *dropshipper* dan *supplier*. 1) Konsumen adalah pembeli barang yang dipromosikan oleh *dropshipper* sebelumnya, 2) *dropshipper* atau pelaku *dropship* umumnya berperan sebagai pencari dan perantara bagi konsumen untuk mendapatkan produk yang disediakan *supplier* dan 3) *supplier* adalah pemilik produk sesungguhnya yang telah dipercaya oleh *dropshipper* untuk memenuhi pengiriman barang (Nauval, 2018). *Dropshipper* berperan penting bagi *supplier*, supplier dapat diuntungkan dengan promosi dan

pemasaran yang dilakukan oleh dropshipper (Feri, 2014). Berikut adalah mekanisme transaksi skema dropship (Agency, 2012):

1. Tahap awal memulai dropship adalah dengan mencari penyedia produk yang dapat diajak untuk bekerja sama menjadi *supplier*. Mencari supplier sangat penting, karena supplier merupakan pelaku utama untuk memenuhi pesanan dari konsumen.
2. Melakukan pemasaran menggunakan media digital dengan memajang foto dan informasi lengkap mengenai produk yang dipromosikan untuk menarik calon konsumen.
3. Apabila konsumen tertarik dan ingin membeli produk atau yang *dropshipper* pasarkan, *dropshipper* akan menanyakan ketersediaan dan harga kepada *supplier*.
4. Infomasi pembelian oleh konsumen akan diteruskan oleh *dropshipper* kepada supplier, pesanan akan diproses supplier ketika *dropshipper* sudah mengirimkan sejumlah biaya pesanan dan supplier tidak perlu tau harga yang dijual oleh *dropshipper* (Syafii, 2013) .
5. *Supplier* memenuhi pesanan konsumen dengan cara mengirim barang sesuai informasi pesanan atas nama *dropshipper*.

### 2.1.3. Kelebihan dan Kekurangan *Dropship*

Skema ini memiliki beberapa kelebihan dan kelemahan seperti yang perlu diketahui sebelum melakukannya. Berikut adalah kelebihan dari skema *dropship* (Ratna Patria, 2020):

1. **Modal sedikit**, untuk memulai *dropship*, tidak diperlukan modal yang besar bahkan tidak perlu modal sama sekali, yang diperlukan hanyalah teknik pemasaran yang baik dan media sosial.
2. **Mudah dan praktis,** kelebihan utama *dropship* adalah tidak perlu mengurus barang, barang yang dijual akan diatur oleh *supplier* seluruhnya.

3. **Produk beragam**, produk yang ditawarkan ke calon konsumen dapat lebih beragam dan mengikuti tren, karena dropshipper tidak perlu membeli barang sehingga tidak perlu menghabiskan stok barang sebelumnya.
4. **Praktis dan fleksibel**, kegiatan pemasaran, transaksi dan pemenuhan pemasaran seluruhnya memanfaatkan jaringan internet sehingga dapat dilakukan di manapun.

Selain kelebihan yang sudah dijelaskan sebelumnya, skema ini memiliki beberapa kelemahan yang menjadi dasar penelitian ini dilakukan sebagai berikut:

1. **Tidak bisa memantau produk**, karena dropship tidak memiliki produk sendiri, dropshipper tidak bisa memantau stok, kualitas dan harga produk secara langsung setiap saat.
2. **Keuntungan kecil**, *dropshipper* tidak mendapatkan potongan harga karena *dropshipper* membeli secara eceran.
3. **Sulit untuk melakukan *after sale***, karena tidak memiliki akses langsung terhadap produk, maka sulit untuk menjamin kualitas produk dan melakukan pengembalian barang.

## 2.2. JSON

JSON (*Javascript Object Notation*) adalah struktur data dari bagian konsep sintaksis stuktur data objek bahasa pemrograman Javascript. JSON memiliki struktur yang sehingga mudah untuk memahami penulisan dan membaca data JSON oleh manusia. Tipe data JSON umumnya dipakai untuk bertukar informasi pada aplikasi web. Struktur pada JSON terdiri atas dua bagian, yaitu:

1. Kumpulan nilai yang saling berpasangan, *value* yang dimaksud bisa berupa objek, maupun data umum seperti *string*, *number* dan *boolean*.
2. Setiap data objek dipisahkan menggunakan koma “ , “.
3. Untuk menampung objek menggunakan kurung kurawal “ { } “
4. Untuk larik (*array*) yang menampung data menggunakan “ [ ] “

Berikut contoh penulisan data JSON:

```
{
        "id": 1,
        "first_name": "Cacilie",
         "last_name": "Danielsky",
        "email": "cdanielsky0@comsenz.com",
        "gender": "Genderfluid",
        "ip_address": "59.228.207.168"
}
```

### 2.3. *Local Storage*

Penyimpanan pada sisi *browser* dapat disebut *Local Storage*, penyimpanan ini merupakan set dari API (*Application Programming Interface*) yang disediakan oleh HTML 5. Local storage memungkinkan pengembang aplikasi untuk menyimpan data yang dibutuhkan aplikasi *client* atau aplikasi yang berjalan di sisi browser. Data tersebut disimpan berupa *string* dan berfungsi untuk meningkatkan peforma dan pengalaman pengunaan aplikasi *web* tersebut (Jemel & Serhrouchni, 2014).

### 2.4. *Cache Storage*

Pada dasarnya *cache* merupakan salinan dari sebuah data atupun berkas yang disimpan untuk keperluan peforma pada saat aplikasi dimuat. Pada *browser* atau akses *website cache storage*, *cache storage* berfungsi untuk meminimalisir latensi dan interaksi *client side* kepada *server side*. *Cache* storage sendiri memiliki penyimpanan yang terbatas, karena pada dasarnya *cache storage* hanya menyimpan aset *file* website seperti html, css, javascript dan beberapa gambar yang dibutuhkan untuk menampikan tampilan aplikasi website tersebut. secara kegunaan berbeda dengan local storage, *cache storage* menyalin *file* dan menyimpan aset *website* untuk jangka waktu tertentu (Meizhen et al., 2013).

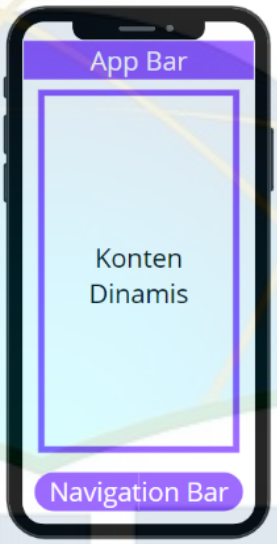
## 2.5. *Progressive Web App*

*Progressive Web App* (PWA) merupakan bagian dari cara pembuatan aplikasi berbasis website, PWA bukan sebuah teknologi baru ataupun bahasa pemrograman khusus yang diciptakan utuk membuat sebuah aplikasi. PWA menyediakan cara bagaimana mengembangkan aplikasi website yang ramah dan menyerupai aplikasi *native* pada perangkat *mobile* dengan menggunakan API maupun strategi pengembangan yang ada pada PWA (Sheppard, 2017). Pengunaan PWA dapat memberikan efisiensi pada waktu pengembangan dan sumber daya diakrenakan aplikasi yang dikembangkan berupa aplikasi web yang dapat diakses disegala perangkat melalui *browser*. Berikut karakteristik penting dalam PWA : (Addy Osmani, 2020)

1. ***Progressive***, dapat diakses menggunakan browser ataupun perangkat dari berbagai generasi, pada generasi lama fungsi standar tetap bisa diakses dan pada generasi terbaru seluruh fungsionalitas dapat diakses.
2. ***Discoverable***, koten dapat ditemukan oleh *search engine* dengan lebih baik, hal tersebut dikarenakan PWA berbasis teknologi *web* yang dapat disisipkan *metadata* dan *Manifest File* yang di dalamnya terdiri dari nama, *icon*, *splash screen*.
3. ***Installability***, dapat di-*install* di perangkat pengguna dan icon aplikasi dapat tampil pada halaman *homescreen.*
4. ***Network Independent***, tidak tergantung atas ketersediaan internet sehingga tetap dapat diakses pada kondisi jaringan buruk atau tanpa jaringan internet.
5. ***Linkable***, salah satu fitur yang dapat ditawarkan oleh aplikasi berbasis web adalah dapat dengan mudah membagikannya melalui sebuah *link* URL.
6. ***Re-engadge***, terdapat fungsi notifikasi ketika mendapat pembaharuan.
7. ***Responsive***, dapat menyesuakan tampilan berasarkan perangkat yang digunakan seperti *smartphone*, komputer, tablet bahkan televisi pintar.
8. ***Secure***, aman karena menggunakan protokol HTTPS

### 2.5.1. *App Shell*

*App Shell* perupakan struktur kerangka tampilan yang diterapkan untuk membuat tampilan aplikasi PWA, app shell terdiri atas beberapa blok komponen *user interface* yang telah disimpan sebelumnya oleh *service worker*. Hal ini memungkinkan aplikasi dapat diakses cepat diakses melalui cache, penggunaan *cache* tersebut juga memberikan kemampuan aplikasi dapat dijalankan pada keadaan *offline* (Behl, 2018). Berikut contoh gambaran struktur *App Shell* pada PWA :




Gambar 2. 1 Struktur App Shell

### 2.5.2. *Manifest File*

*Web Manifest* atau *Manifest File* adalah *metadata* yang di dalamnya terdapat informasi *icon*, nama, tema, *loading screen* dan deskripsi dari aplikasi yang akan di-*install*. *Manifest file* menyimpan informasi bagaimana aplikasi tersebut berprilaku, apakah pada saat di-*install* harus ke URL atau *routing* tertentu dan bagaimana aplikasi ditampilkan.

### 2.5.3. *Service Worker*

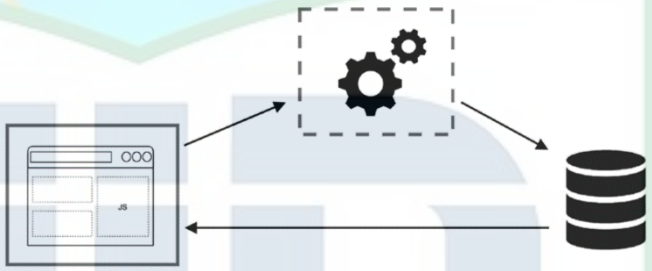
*Service Worker* adalah komponen utama dalam implementasi *Progressive Web App* (PWA), *service worker* berperan sebagian besar dari

fitur yang ditawarkan PWA tanpa service worker, PWA tidak bisa berjalan seperti yang diharapkan (Biørn-Hansen et al., 2017). *Service worker* sendiri berupa *script* (*JS File*) yang berjalan di latar belakang aplikasi. *Service worker* sendiri terpisah dengan bagian *user interface*, service worker lebih berperan mengontrol jaringan dan menangani permintaan dari sisi *client*. *Service worker* juga berperan penting untuk menyimpan aset dan data yang dibutuhkan aplikasi ke dalam *cache*.

#### 2.5.3.1. Strategi *Cache*

Untuk membuat aplikasi dapat dijalankan secara offilne, diperlukan sebuah *cache* pada aplikasi. Service worker akan mengelola *cache* tersebut ke dalam *cache storage* yang tersedia pada *browser* pengguna. Untuk memenuhi kebutuhan aplikasi, terdapat berapa pola pemakaian *cache* oleh *service worker*. Berikut pola atau strategi caching pada service worker (Hajian, 2019):

1. *Cache Only*

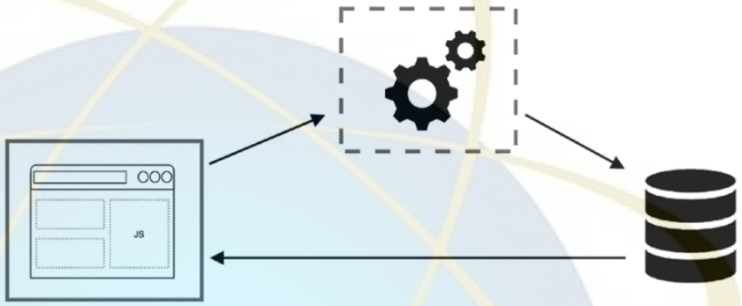


Gambar 2. 2 Cache Only

Pada *cache only* aset atau komponen tampilan akan disimpan ke dalam *cache storage*, pada saat aplikasi dijalankan *service worker* akan melakukan *request* hanya kepada *cache storage*. Cache ini bersifat *static* karena tidak melakukan request data dan aset komponen terbaru kepada server, ketika aset tidak ditemukan di dalam *cache* maka aplikasi akan gagal dijalankan.

2. *Network Only*

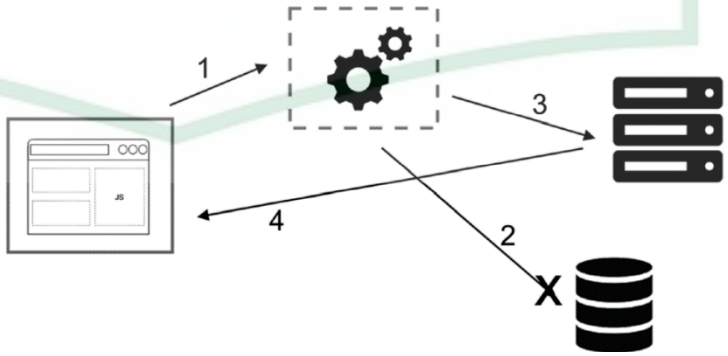
*Network only* menggunakan pola strategi dimana aplikasi wajib menggunakan data terbaru. Aplikasi yang menggunakan pola strategi *cahcing network only* berfokus pada pertukaran data atau perubahan data setiap waktu (*real-time*), contohnya adalah pada aplikasi pelelangan dan penyedia layanan jual dan beli saham yang membutuhkan data harga terbaru tiap beberapa detik.




Gambar 2. 3 Network Only

3. *Cache-First*

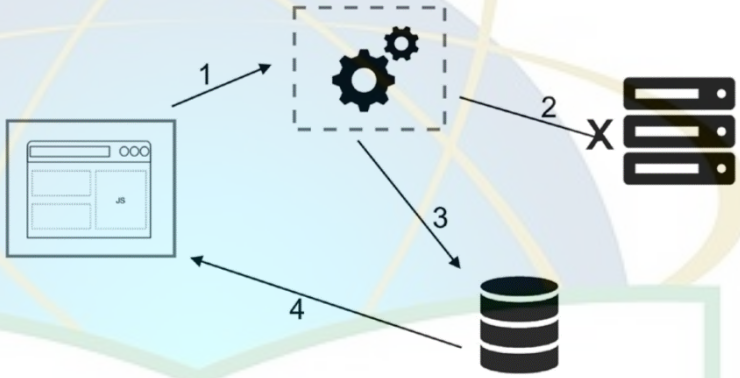
Pola strategi cache first cocok untuk digunakan pada aplikasi yang tidak sering mendapatkan perubahan, contohnya pada aplikasi PWA strategi ini dilakukan kepada aset yang dibutuhkan untuk membentuk *App Shell*. Aplikasi akan melakukan request kepada cache terlebih dahulu, jika respon dari *cache storage* gagal maka *service worker* akan mencoba mendapatkan respon dari *server.*




Gambar 2. 4 Cache First

4. *Network First*

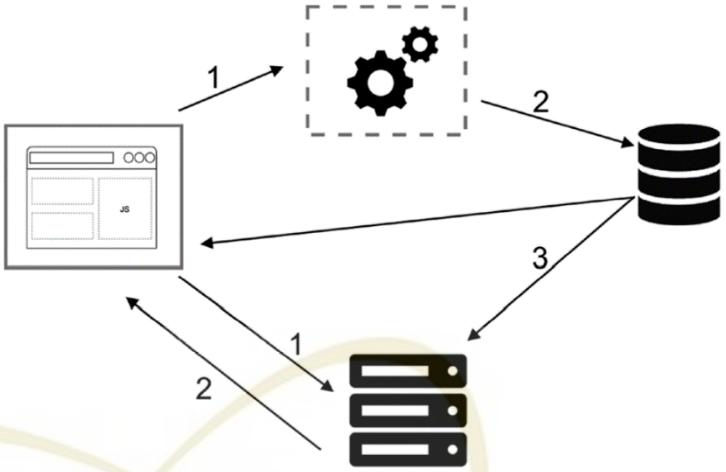
Pola *network first* merupakan kebalikan dari pola strategi cache first, jika pada cache first aplikasi akan mengutamakan mengambil data dari *cache storage*, maka *network first* akan melakukan *fetch* atau *resquest* telebih dahulu ke pada server agar mendapatkan data terbaru. Jika aplikasi gagal mendapat respon dari server maka *service worker* akan menggunakan data dan aset dari *cache*.




Gambar 2. 5 Network First

5. *Stale-While-Revalidate*

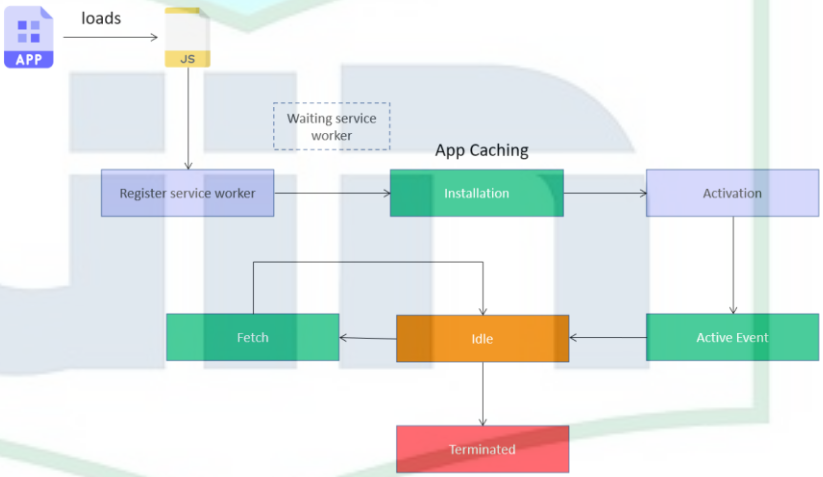
Strategi *cache* ini akan memprioritaskan data dari *cache* untuk dimuat ke dalam aplikasi, di sisi lain aplikasi juga akan melakukan *request* secara pararel terhadap *server* untuk mendapatkan data terbaru. Strategi ini sangat berguna pada aplikasi seperti sosial media, data yang ditampilkan pertama kali adalah data sebelumnya yang sudah di-*cache*, ketika pengguna melakukan *scroll* data terbaru dari respon *server* akan di-*cache* dan ditampilkan pada aplikasi.




Gambar 2. 6 Stale While Revalidate

### 2.5.3.2. Service Worker Life Cycle

*Service worker* memiliki siklus hidup yang memberikan pengembang aplikasi dalam memberikan pengalaman pengunaan terhadap penggunanya, berikut gambaran dan keterangan siklus hidup dari *service worker:*




Gambar 2. 7 Siklus hidup Service Worker

Pada gambar di atas, merupakan siklus hidup pada *service worker*. *File javascript* yang memuat *script service worker* akan melakukan registrasi *service worker* ke dalam proses *background*,

setelah teregistrasi *service worker* masuk ke tahap *install* dan pada tahap ini kita bisa menyisipkan kode yang ingin dieksekusi pada event installed. Pada *event installed, caching app shell* biasa terjadi di sini. Setelah *service worker* telah mendapatkan akses untuk mengkontrol *client* lalu masak ke tahap *activated*, di tahap ini *service worker* dapat menangani *request* yang masuk. *Service worker* tidak bisa memasuki tahap *activated* jika ada *service worker* yang masih aktif, *service worker* baru akan menunggu instalasi atau ketika seluruh proses aplikasi tersebut terhenti (Sheppard, 2017).

#### 2.5.3.3. Lighthouse

*Lighthouse* merupakan aplikasi *open source* yang terdapat di dalam *Chrome DevTools*, *tools* ini ditargetkan untuk melakukan pengujian dan pengukuran terhadap aplikasi PWA secara otomatis. *Tools* ini juga tetap bisa digunakan terhadap aplikasi non-PWA, *tools* ini melakukan audit terhadap *performance*, *accessibility*, *progressive web apps* dan SEO (Ashwin Sathian, 2020).

## 2.6. *Workbox*

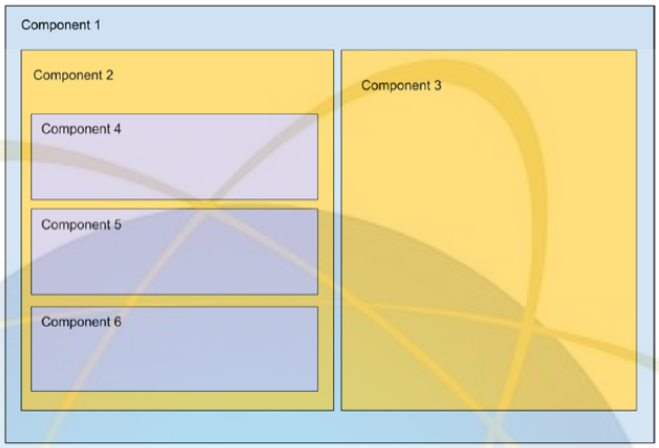
*Workbox* merupakan koleksi *library* dan *tools* yang disediakan Google untuk membantu membuat *service worker*. Workbox memberikan fleksibilitas yang lengkap untuk mengkontrol dan menambahkan fitur – fitur yang ada pada *service-worker*, dengan *workbox* kita bisa menentukan bagaimana aset *file* (html, css, js, gambar) dimuat ke dalam aplikasi dengan melalui jaringan atau *cache* yang tersedia (Sheppard, 2017).

## 2.7. Vuejs

Vue atau bisa dibaca "*view*", merupakan sebuah *framework* untuk membuat tampilan aplikasi pada aplikasi *web*. *Framework* ini dibuat oleh Evan You yang merupakan mantap pegawai Google, vue sendiri menggabungkan kelebihan dan fitru yang ada pada *framework* sejenis yaitu Angular dan React (Rojas, 2019).

### 2.7.1. *Component*

Component di merupakan elemen blok UI yang dapat dikustomisasi, component memiliki *style*, *state*, dan prilakunya sendiri. *Component* dapat digabungkan ke dalam *component* lainnya dan dapat digunakan berulang di tempat lain.



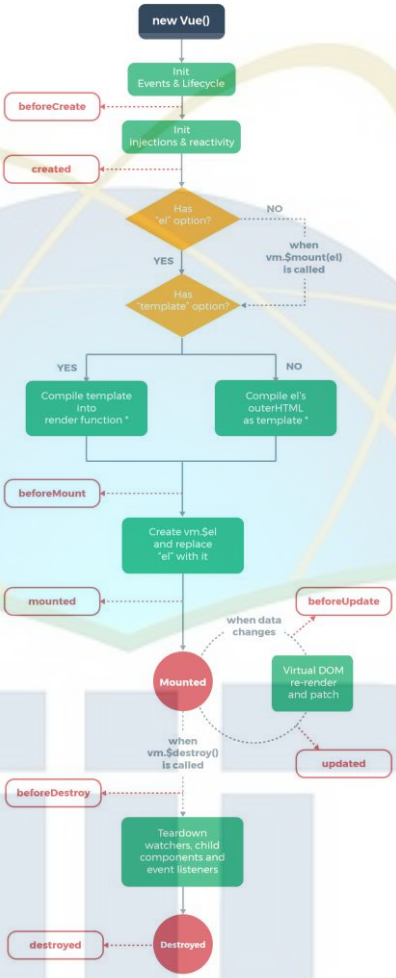

Gambar 2. 8 Web Component
Sumber: (Rojas, 2019)

### 2.7.2. *Single Page Application*

*Single Page Application* (SPA) terdiri dari beberapa *component*, *component* tersebut dapat diganti atau diperbaharui berdasarkan *request* dan interaksi pengguna. *Single Page Application* tidak perlu memperbaharui atau memuat ulang seluruh halaman untuk melakukan navigasi maupun mendapatkan konten terbaru (Jadhav et al., 2015).

### 2.7.3. *Life Cycle Hook*

Setiap blok UI (*user interface*) di vuejs adalah *component*, *component* tersebut memiliki proses pembuatan dan penghancuran ketika *component* lain dimuat dan disebut *Life Cycle Hook*. Berikut tahapan siklus hidup dari vuejs (Joshua Bemenderfer, 2020):




Gambar 2. 9 Life Cycle Hook Vuejs
Sumber: (https://v3.vuejs.org/api/options-lifecycle-hooks.html)

#### 2.7.3.1. *Creation Hooks*

Setiap *component* yang akan diinisiasi dan belum dimasukan ke dalam DOM (*Document Object Model*) akan memasuki *Creation Hooks*. *Hooks* ini memiliki dua *cycle* yaitu before *create* dan *created*. Pada saat vue dijalankan, secara langsung component akan

memasuki tahap *beforeCreate*, di *cycle* ini *state data* dan *event* belum terbentuk. *Created cycle* akan dieksekusi ketika *computed properties, watchers, events, data properties* sudah siap digunakan sehingga kita dapat melihat data dan fungsi yang terdapat di *component* melalui *vue devtools*.

#### 2.7.3.2. *Mounting Hooks*

Hook ini berjalan ketika component akan di-*render* ke dalam DOM atau terjadi perubahan data di dalam *component*. Disarankan untuk tidak melakukan *request* data melalui siklus ini, karena pada *hook* ini semua data *component* sudah terinisiasi dan *reactivity* dari vuejs tidak dapat berkerja. *Mounting hooks* terdapat dua *cycle* yaitu *beforeMount* dan *mounted.*

#### 2.7.3.3. *Updating Hooks*

*Updating hooks* dipanggil ketika adanya perubahan data atau sesuatu yang menyebabkan *component* harus di-*render* ulang. Terdapat dua *cycle* pada *hooks* ini yaitu *beforeUpdate* dan *updated*

#### 2.7.3.4. *Destruction Hooks*

*Hooks* ini dipanggil ketika component dibuang atau digantikan dengan *component* lain dari DOM. Terdapat dua *cycle* pada *hooks* ini yaitu *beforeDestroy* dan *destroyed* , pada *cycle beforeDestroy* kita bisa melakukan fungsi sebelum data di dalam *component* terhapus setelah memasuki *destroyed* seluruh data dan *component* turunannya akan dibuang dari DOM dan instansi vue

## 2.8. *Vuex*

Vuex merupakan bagian dari *framework* vuejs, vuejs terdiri dari berbagai *component* yang nantinya akan membentuk tampilan *web*. Setiap *component* tersebut dapat memiliki *state, state* tersebut menjadi cara antar *component* bertukar data. Untuk memudahkan pertukaran *state* atau data antar *component*, vuejs menyediakan *library* yaitu *vuex* sendiri. Vuex akan meyediakan sentralisasi state

pada aplikasi *vue*, dengan sentralisasi *state* ini maka *state* akan mudah diprediksi ketika terjadi perubahan dan data yang dipakai oleh antar *component* merupakan data yang sesuai. Vuex terdiri dari beberapa struktur dan fungsinya masing – masing, terdapat *state* yang menampung informasi, *getters* untuk mengembalikan data yang dimaksud, *mutation* adalah cara *state* dapat diubah secara langsung, dan *actions* tempat kita melakukan operasi *asynchronous* atau berkomunikasi dengan *server* (Anthony gore, 2017).

## 2.9. *Vuetify*

*Vuetify* merupakan UI *Framework* yang dibuat untuk *vuejs*, *vuetify* menjadi salah satu UI *framework* yang paling populer untuk membangun aplikasi *frontend* berbasis *vuejs*. *Vuetify* mengadopsi sistem desain *Material* seperti pada perangkat android yang dirancang Google, UI *framework* ini memiliki berbagai jenis *component* yang siap pakai dan memiliki banyak fitur seperti menampilkan tabel data, konfigurasi tema, dan animasi halaman (John Leider, 2021).

## 2.10. *Firebase*

*Firebase* adalah layanan komputasi dan penyimpanan yang disediakan oleh Google, *firebase* merupan bagian dari *Google Cloud Service*. *Firebase* membantu dan memudahkan pengembang untuk membuat aplikasi tanpa harus fokus terhadap infrastruktur dan *backend* aplikasi. Berikut layanan yang disediakan oleh firebase (Rojas, 2019):

### 2.10.1. *Firebase Authentication*

Aplikasi yang aman membutuhkan otentikasi untuk memasikan bahwa tersebut berhak mengakses aplikasi tersebut. Hal tersebut *firebase* menyediakan layanan otentikasi yaitu *Firebase Authentiaction*, layanan ini menyediakan metode *sign-in* menggunakan *email* dan sosial media (facebook, twitter, google) (Khedkar et al., 2017). *Firebase* akan otomatis menyimpan data setelah registrasi atau melakukan verifikasi terhadap pengguna.

### 2.10.2. *Firebase Firestore*

*Firebase Firestore* atau disebut juga *Firebase Cloud Firestore*, merupakan layanan penyimpanan data selain *firebase realtime database*. Struktur dari penyimpanan data *firestore* mirip dengan penyimpanan data non-sql. *Firebase firestore* terdiri dari *collection* dan *document*, di dalam *document* terdapat *field* yang dapat disi data. Layanan ini dapat dihubungkan ke berbagai bahasa pemrograman melalui SDK (*Software Development Kits*) yang disediakan, bahasa pemrograman tersebut antara lain javascript, java, kotlin, golang, python. *Firebase Firestore* mengenakan biaya ketika ada document yang diambil dan data yang melakukan operasi *write*, berbeda dengan *realtime database* yang menggunakan jumlah *bandwith*.

### 2.10.3. *Firebase Storage*

*Firebase storage* diperuntukan untuk menyimpan data konten perupa foto dan video, dengan adanya layanan firebase storage ini dapat memudahkan pengembang aplikasi untuk menyediakan fitur *upload* gambar dan video yang dapat diandalkan tanpa harus membuat *backend* dari awal (Chatterjee et al., 2018).

### 2.10.4. *Firebase Hosting*

Aplikasi berbasis *web* harus di-*host* agar dapat diakses melalui internet, layanan *host* telah disediakan oleh *firebase* bernama *Firebase Hosting*. Layanan *firebase hosting* dapat menyimpan *file* aplikasi *frontend* dan men-*deploy* aplikasi tersebut ke internet lewat *domain* yang disediakan *firebase*. Untuk melakukan *deploy*, kita memerlukan SDK atau menginstall *Firebase* CLI terlebih dahulu.

## 2.11. CRUD Application (*Create, Read, Update, Delete*)

CRUD adalah paradigma yang umumnya digunakan dalam mengembangkan aplikasi. Aplikasi CRUD pada dasarnya menyediakan fitur dan operasi dasar seperti membuat data, mengubah data, membaca data dan menghapus data (Truica et al.,

2013). Pada aplikasi CRUD yang akan dibuat di penelitian ini memiliki fungsi untuk mengatur pesanan, pelanggan dan produk pada *dropship*.

## 2.12. *Rapi Application Development* (RAD)

Berdasarkan kebutuhan dan batasan masalah terhadap pengembangan aplikasi yang akan diimplentasikan PWA, dibutuhkan metode pengembangan *software* atau *Software Development Life Cycle* (SDLC) yang cepat dan mudah, maka metode pengembangan *Rapid Application Development* (RAD) dinilai cocok untuk digunakan.

### 2.12.1. Siklus Pengembangan RAD

RAD cocok karena siklus pengembangannya yang singkat antara 30 – 90 hari. Metode ini menggunakan metode iteratif dan bertingkat dalam siklusnya. Metode RAD memiliki beberapa fase, yaitu (Wahyuningrum & Januarita, 2014):

1. *Requirements Planning* (Perencanaan Kebutuhan**):** Tahap ini bertujuan untuk mengidentifikasi tujuan, kebutuhan dan masalah yang mungkin timbul saat pengerjaan proyek berlangsung.
2. *Design System* (Proses Desain Sistem): fungsi desain sistem adalah untuk memberikan gambaran proses secara lebih jelas sesuai dengan hasil analisis dan tujuan. Hasil dapat berupa desain struktur sistem, tampilan, scenario tes dan *database*.
3. *Implementation* (Implementasi): tahap terakhir ini pengembang mengaplikasikan hasil desain sebelumnya ke dalam aplikasi, aplikasi tersebut akan dilakukan pengujian untuk mengetahui kesalahan dan kesesuaian terhadap tujuan dan spesifikasi sistem.

### 2.12.2. Kelebihan dan Kekurangan RAD

Metode RAD dinilai cukup baik untuk mengembangkan aplikasi dalam skala kecil dan cepat, berikut adalah kelebihan dan kekurangan metode RAD (Aswati & Siagian, 2016):

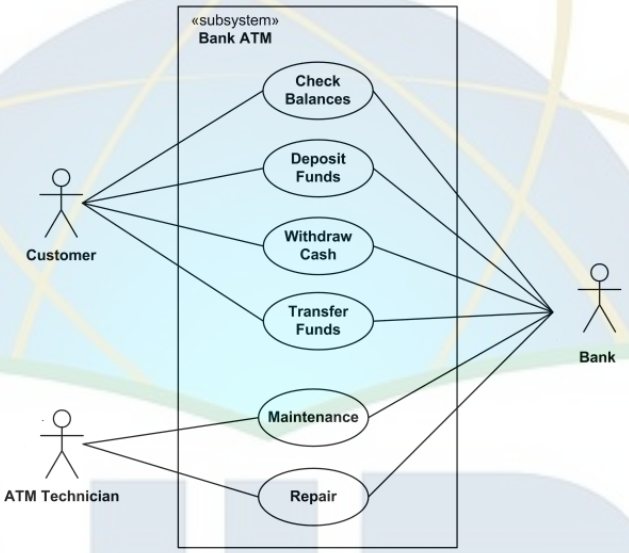
1. Kelebihan RAD
    a. Menghemat waktu dari seluruh fase pengembangan
    b. Pengembangan berfokus pada kecepatan penyelesaian proyek
    c. Perubahan desain dapat dilakukan secara fleksibel
2. Kekurangan RAD
    a. Pengerjaan yang berorientasi kecepatan, hal tersebut memberikan hasil yang tidak maksimal
    b. RAD menuntut tim untuk dinamis terhadap perubahan yang cepat dengan dituntut untuk menguasai alat dan kemampuan baru sembari melakukan pengembangan.

## 2.13. UML (*Unified Modeling Language*)

Pengembangan aplikasi membutuhkan rangcangan yang menjadi dasar acuan dalam tahap implementasi, pengembangan aplikasi tanpa rencana bisa menimbulkan kesalahan pengembangan dan akan sulit untuk menambahkan pembaharuan terhadap aplikasi tersebut. Diperlukan sebuah standar yang dapat memvisualisasi fungsi, model dan bentuk aplikasi agar dapat dipahami secara menyeluruh. UML atau *Unified Modelling Language* hadir untuk menjadi standar untuk memvisualisasikan perencanaan dan dan bentuk aplikasi yang akan dibentuk. UML sendiri berkonsep pada pemrograman berorientasi objek. UML menggambarkan pemodelan tersebut melalui bentuk diagram, berikut diagram yang tedapat dalam UML (Dharwiyanti & Wahono, 2003):

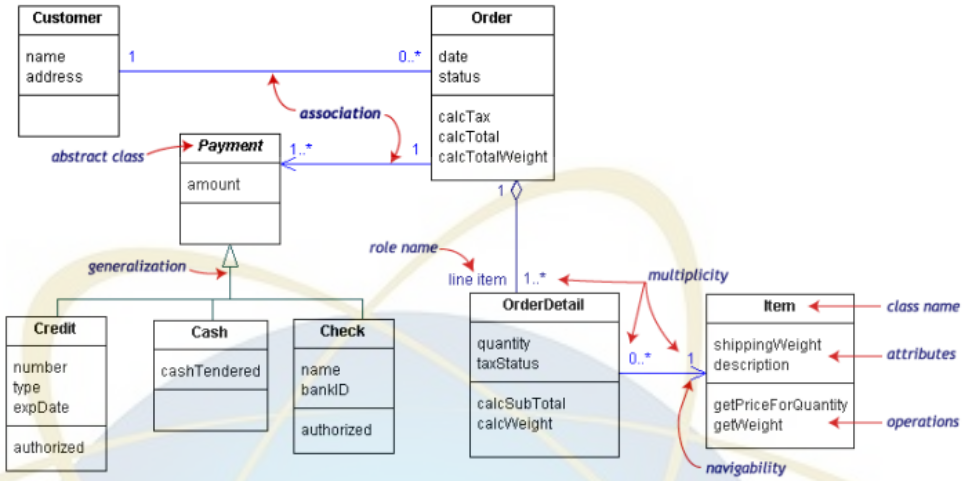
### 2.13.1. *Use Case Diagram*

Use Case Diagram mengvisualkan hubungan bagaimana atribut aktor dan sistem berinteraksi. Diagram ini dapat menjadi rujukan dalam pengembangan aplikasi yang akan dibuat, karena diagram use case dapat mendeskripsikan keseluruhan bentuk sistem yang akan dibuat. Diagram ini dapat dengan mudah dipahami oleh use dan pengembang dikarenakan diagram yang dibuat dengan beberapa bentuk. Berikut komponen yang terdapat dalam use case diagram (Martina Seidl & Marion Scholz, 2015):




Gambar 2. 10 Use Case Kasus ATM
Sumber: (https://www.dicoding.com/blog/apa-itu-uml/)

### 2.13.2. *Class Diagram*




Gambar 2. 11 Class Diagram
Sumber: (Dharwiyanti & Wahono, 2003)

Class diagram menggambarkan kelas objek yang memiliki atribut, properti dan fungsi. Tiap kelas digambarkan dengan bentuk persegi dan hubungan antar kelas digambarkan oleh garis penghubung yang menyambung dari satu kelas ke kelas lainnya. Hubungan yang terjadi pada class diagram antara lain, asosiasi, agregasi dan pewarisan.

### 2.13.3. *Activity Diagram*

*Activity* berarti aktivitas dalam bahasa Indonesia, *activity diagram* merupakan diagram yang menggambarkan alur aktivitas yang berjalan. *Activity diagram* dideskripsikan sekumpulan yang berjalan dari awal hingga akhir untuk memodelkan alur kerja dari suatu proses. Untuk menghindari kesalahan dalam memahami alur, maka dibutuhkan objek *swimlane* yang berfungsi membagi aktivitas yang dilakukan berdasarkan objek yang bertanggung jawab melakukan aktivitas dalam *swimlane*.


Gambar 2. 12 Activity Diagram
Sumber: (Komang Nova Artawan & Ketut Gede Suhartana, 2019)

## 2.14. Studi Literatur Sejenis

Tabel 2. 1 Literatur Sejenis

| No | Penelitian | Teknologi | Modul Dropship | PWA / Service Worker | Lighthouse Test | Network Test | Workbox | Frontend Framework | State Management | Mobile |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Pengembangan Aplikasi Pemesanan Layanan Kecantikan Berbasis *Progressive Web Apps* (PWA) (Studi Kasus Two Cents) (*Telaumbanua Achmad Ghazali, 2018*) | Progressive Web App | | √ | √ | | | √ | | √ |
| 2 | Platform e-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps (*Adi, 2017*) | Progressive Web App | | √ | √ | √ | | √ | | √ |
| 3 | Website Pengelolaan Dropshipper pada Toko | Website | √ | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Hikmah Putra Elektronika (Sanjaya, 2020) | | | | | | | | | |
| 4 | Perancangan Aplikasi Dropshipping Produk Smartphone Berbasis Web (Aulia & Oktavianus, 2018) | Website | √ | | | | | | | |
| 5 | Rancang Bangun Aplikasi e-Commerce Dropship Berbasis Web (Waworuntu, 2020) | Website | √ | | | | | | | √ |
| 6 | Progressive Web Apps ( PWA ) for YII Framework Enrichment (Basren, 2018) | Progressive Web App | | √ | √ | √ | | | | |
| 7 | Implementasi Progressive Web Apps (PWA) Menggunakan Laravel Dan Vue.Js dalam Pembuatan | Progressive Web App | | √ | | | | √ | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Aplikasi Penyedia Jasa Freelance (Noor & Irfan, 2020) | | | | | | | | | |
| 8 | Implementasi Progressive Web App | Progressive Web App | √ | √ | √ | √ | √ | √ | √ | √ |

# BAB III
# METODOLOGI PENELITIAN

## 3.1. Metode Pengumpulan Data

Pengumpulan data dan informasi dilakukan pada penelitian ini untuk memastikan penelitian dilakukan relevan dan mencapai tujuan yang telah ditetapkan. Penelitian ini menggunakan dua pendekatan metode pengumpulan data yaitu studi pustaka dan observasi.

### 3.1.1. Studi Pustaka

Pengumpulan data secara studi pustaka dilakukan untuk memperkuat argumen peneliti dalam melakukan penelitian berdasarkan sumber ilmiah atau penelitian sebelumnya secara tidak langsung. Metode ini dilakukan dengan mempelajari referensi dan mencari sumber yang relevan dan sesuai dengan topik penelitian. Referensi yang digunakan berupa sumber tulisan seperti buku, jurnal, situs, dan artikel. Infromasi yang didapat lalu diolah untuk menyusun latar belakang, ladasan teori, dan implementasi pembuatan dan pengujian aplikasi.

### 3.1.2. Observasi

Observasi dilakukan untuk memastikan pelaksanaan yang berjalan secara nyata terhadap topik yang dibahas, informasi yang didapat pada tahap pengumpulan data ini berupa data, fakta dan proses yang terjadi pada sistem penjualan *dropship*. Observasi dilakukan dengan melakukan langsung proses bisnis tersebut terhadap toko yang aktif berjualan secara *online* dengan model *dropship*.

### 3.1.3. Wawancara

Untuk memastikan dan mendapatkan fakta lebih luas mengenai praktik *dropship*, penulis melakukan wawancara kepada salah satu praktisi *dropshipper* pada bisnis *online* yang sudah dijalankan kurang lebih satu tahun belakangan. Hasil dari wawancara tersebut bisa dijadikan masukan

dan sebagai pertimbangan sebagai solusi yang diajukan untuk masalah yang dihadapi penulis dan praktisi model bisnis *dropship* serupa. Wawancara ini dilakukan melalui pesan singkat melalui aplikasi *whatsapp*.

## 3.2. Metode Pengembangan Sistem

Penelitian ini bertujuan untuk mengembangkan aplikasi yang dapat membantu *dropshipper* untuk mencatat data dalam proses bisnis *dropship*, selain itu aplikasi yang dikembangkan akan berbasis *website* yang diharapkan dapat menjadi alternatif pengembangan aplikasi *mobile*. Pengembangan aplikasi menggunakan *framework vuejs* dan *Progressive Web App* (PWA). Aplikasi tersebut akan mengimplementasi fungsi dasar CRUD (*create*, *read*, *update*, *delete*) dan dilakukan pengujian untuk memasitikan fungsi berjalan sesuai yang diharapkan. Aplikasi ini diharapkan dapat membantu proses bisnis dropship dan memberikan pengembangan aplikasi *mobile* dengan lebih cepat dan lebih efisien menggunakan teknologi *web*. Metode yang digunakan untuk pengembangan sistem yaitu *Rapid Application Development* (RAD), metode ini digunakan karena memiliki alur pengembangan yang mudah dan singkat sehingga cocok untuk mengembangkan aplikasi terhadap keperluan pengujian atau tidak bersifat final. Berikut penjelasan mengenai fase yang dilakukan dalam pengembangan aplikasi CRUD pada manajemen *dropship* menggunakan RAD:

### 3.2.1. *Requirements Planning* (Perencanaan Kebutuhan)

Peneliti mencari data dan informasi yang relevan terhadap sistem dropship. Informasi didapatkan melalui berbagai sumber pustaka dan melakukan observasi secara langsung dalam praktik dropship. Informasi tersebut lalu diidentifikasi sebagai acuan perancangan sistem yang akan dibuat. Berikut tahapan yang yang dilakukan pada fase ini:

1. Mengumpulkan data dan informasi yang relevan terhadap manajemen *dropship*.
2. Mengiidentifikasi fitur solusi terhadap sistem yang akan dibuat.
3. Menentukan *tools* yang digunakan dalam proses pengembangan sistem.

### 3.2.2. *Design System* (Proses Desain Sistem)

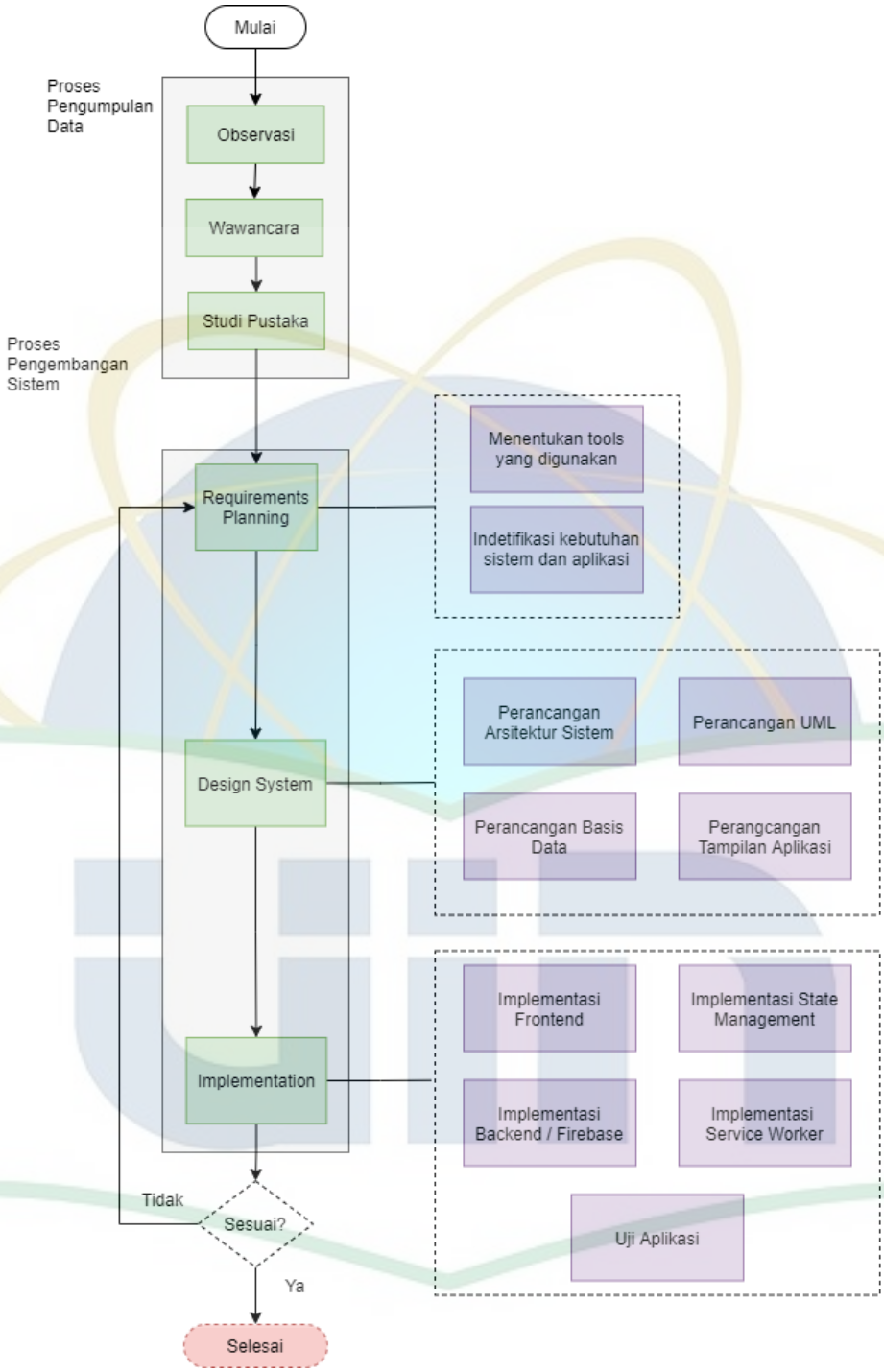
Hasil identifikasi kebutuhan yang dilakukan pada fase sebelumnya diterjemahkan ke dalam sistem melalui tahap ini, pada tahap ini dilakukan perancangan terhadap proses sistem, basis data dan tampilan sistem. Proses ini juga mendesain bagaimana alur aplikasi akan berjalan dan bagaimana *service worker* bekerja.

### 3.2.3. *Implementation (Implementasi)*

Tahap ini dilakukan pengembangan aplikasi berdasarkan desain sistem yang dibuat sebelumnya, selanjutnya aplikasi diuji untuk mengetahui kesesuaian dan ukuran aplikasi yang dihasilkan. Tahapan yang dilakukan pada fase ini diantaranya:

1. Melakukan *coding* terhadap sistem dan basis data.
2. Mengimplementasikan PWA ke dalam aplikasi.
3. Melakukan pengujian dan pengukuran, tahap ini dilakukan untuk memastikan fitur PWA, fungsi aplikasi, kinerja dan penggunaan memori oleh aplikasi.

## 3.3. Alur Penelitian

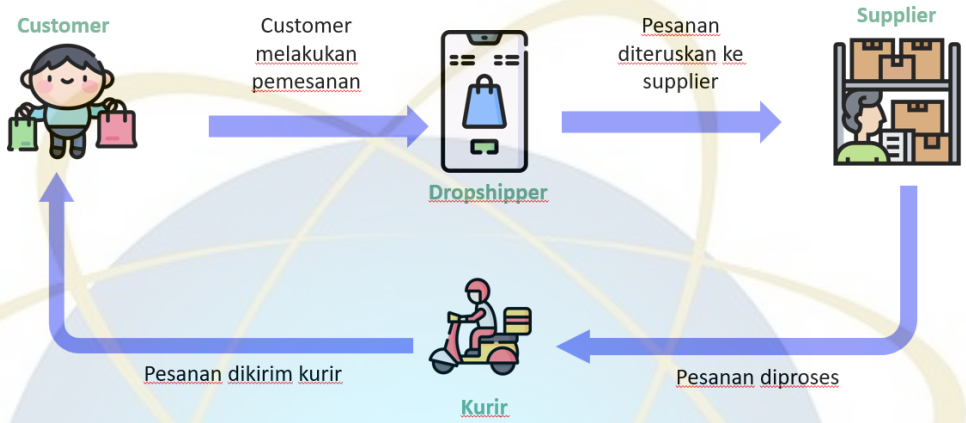



Gambar 3. 1 Alur Penelitian

# BAB IV
# IMPLEMENTASI

## 4.1. *Requirement Planning*

### Analisis Alur Sistem




Gambar 4. 1 Alur Sistem Dropship

Secara umum terdapat tiga aktor dalam transaksi dropship, yaitu customer, supplier dan dropshipper. Berikut alur sistem *dropship* yang diperoleh dari hasil *requirements planning*:

Keterangan alur sistem:

1. *Customer* melakukan pemesanan di toko online *dropshipper*.
2. *Dropshipper* menerima pesanan dan pesanan diteruskan kepada *supplier* dengan membayar sejumlah uang sesuai harga pesanan dari *supplier*.
3. *Supplier* menerima pesanan dari *dropshipper* dan memproses pesanan sesuai informasi *customer*.
4. Setelah pesanan dikirim *supplier* akan memberikan resi kepada *dropshipper* dan *dropshipper* meneruskan resi tersebut.
5. Barang terkirim ke *customer* dan *dropshipper* dapat mencairkan uang tersebut.

## 4.2. *Design System*

Pada tahapan ini dibuat pemodelan UML, basis data dan frontend. Pemodelan aplikasi disesuaikan untuk keperluan hasil uji dan target implementasi menggunakan *Vuejs* dan *Progressive Web App*.

### 4.2.1. Usulan Alur Sistem Aplikasi

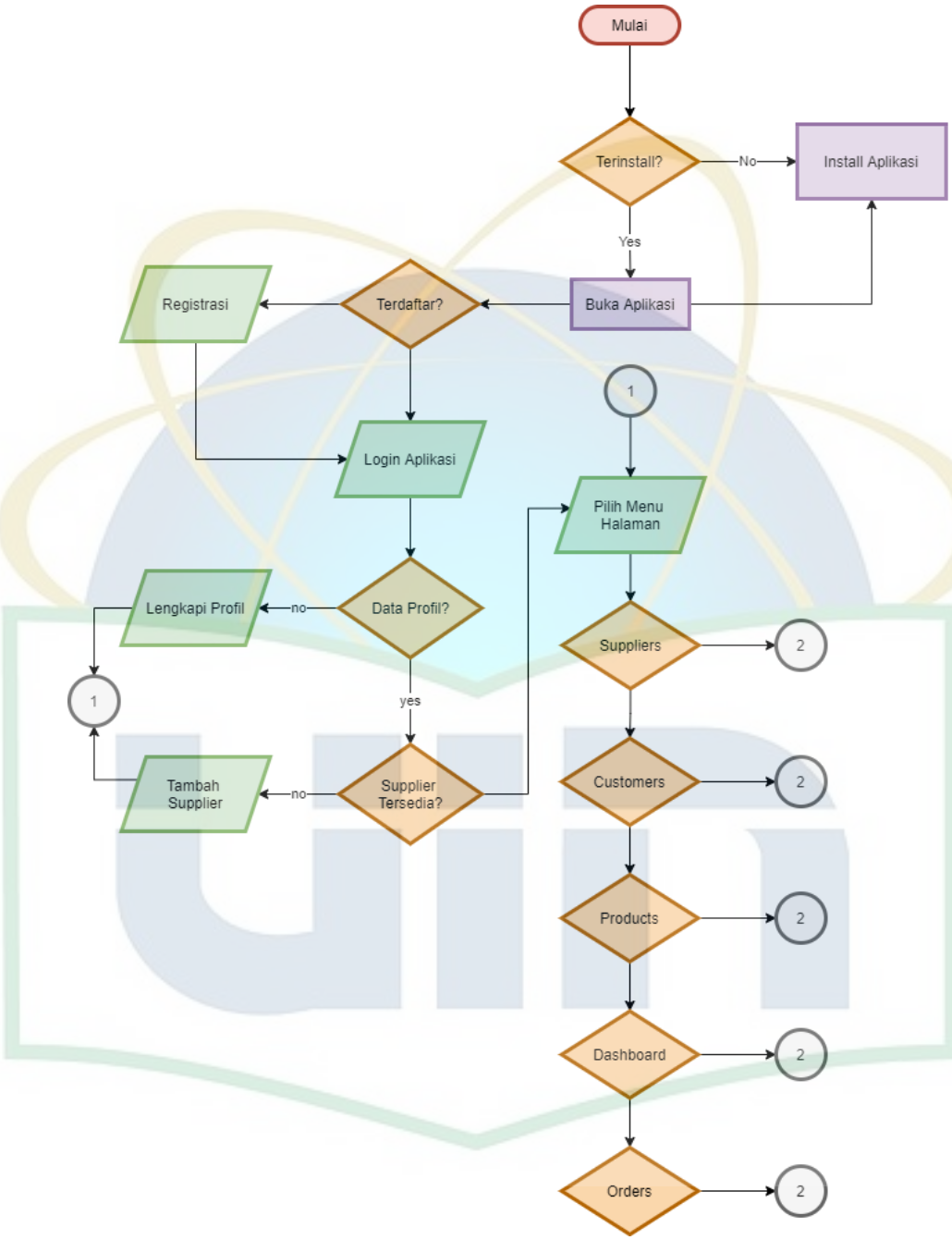
Sesuai topik topik dan hasil identifikasi kebutuhan sistem, aplikasi ini diharpakan dapat menyediakan fungsi create, read, update dan delete. Implementasi dilakukan terhadap sistem dropship, diharapkan aplikasi ini dapat melakukan pengelolaan terhadap proses bisnis *dropship*, berikut usulan alur sistem aplikasi dropship:

1. Dropshipper bertindak sebagai *user* dan *aktor* tunggal yang dapat mengakses aplikasi.
2. User baru mendaftarkan diri dengan membuat akun dan mengisi formulir yang ada untuk mendapatkan akses ke dalam aplikasi.
3. User melakukan login menggunakan password yang sudah ada
4. Setelah berhasil login, user akan disambut oleh informasi data pada halaman *Home*
5. User dapat mengupload foto dan mencantumkan link tokonya
6. User harus membuat data supplier dan produk terlebih dahulu sebelum membuat data produk
7. Data yang sudah dibuat akan tampil pada halaman masing – masing kategori
8. Ketika *user* (*dropshipper*) mendapatkan pesanan, user memasukan data pesanan dengan memilih produk, customer dan nomor pesanan (jika ada).
9. Pesanan dapat dilacak berdasarkan status yang ada, status pesanan antara lain:
    a. *To Order*, sudah diterima dan siap dipesan ke supplier
    b. *Packing*, pesanan sudah dipesan dan sedang diproses oleh *supplier*

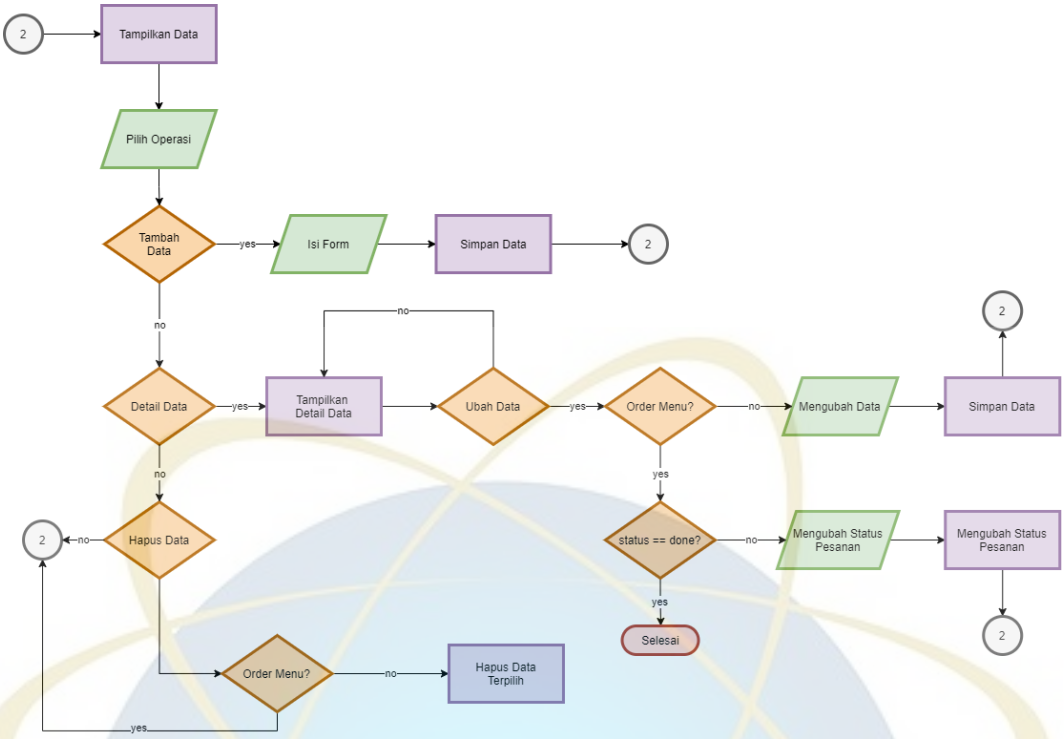
c. *Shipping*, *supplier* sudah mengirimkan resi pengiriman

d. *Done*, pesanan sudah diterima

10. Status pesanan dapat diubah oleh user di halaman pesanan.




Gambar 4. 2 Alur Aplikasi




Gambar 4. 3 Sub Alur Aplikasi Modul CRUD

### 4.2.2. Tools dan Teknologi

Berdasarkan analisis dan kebutuhan sistem, peneliti menggunakan beberapa tools yang dipakai pada tahap implementasi, diantaranya:

1. Perangkat *android*
2. *Visual studio code* sebagai teks *editor*
3. *Firebase Storage*, sebagai tempat penyimpanan gambar
4. *Firebase Firestore*, sebagai tempat meyimpan data
5. *Firebase Hosting*, tempat *host* aplikasi
6. *Vuejs*, sebagai pembuatan single page application dan app shell
7. *Vuex*, sebagai state management
8. *Progressive Web App*, sebagai alternatif pengembangan *mobile* dan membuat aplikasi berjalan *offline*

### 4.2.3. Perencanaan UML

Aplikasi yang akan dibuat adalah aplikasi yang dapat memenuhi fungsi dasar CRUD (*create*, *read*, *update*, *delete*). Aplikasi ini berfokus pada sistem manajemen yang dilakukan oleh *dropshipper*, sehingga *dropshipper* dapat mengelola toko, produk dan *supplier* dengan lebih baik. Berikut rancangan alur aplikasi yang dibuat:

#### 4.2.3.1. *Identifikasi Aktor*

Pada alur sistem dropship terdapat tiga aktor yang berperan, yaitu customer, droshipper dan supplier. Sesuai dengan perencanaan fungsi aplikasi, aktor yang akan terlibat dan dapat mengakses aplikasi hanya aktor tunggal yaitu *dropshipper*.

Tabel 4. 1 Identifikasi Aktor

| Aktor | Deskripsi |
|---|---|
| *Dropshipper* | *Dropshipper* adalah pengguna aplikasi yang dapat melakukan mengelola data pesanan, *customer*, produk dan *supplier*. |

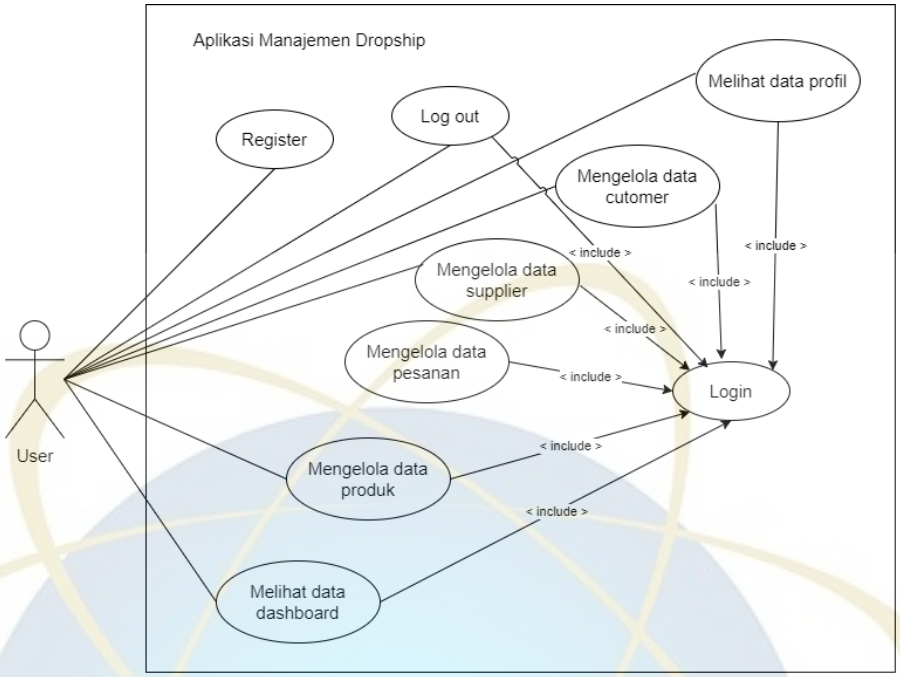
#### 4.2.3.2. Indentifikasi Use Case

Tabel 4. 2 Identifikasi Use Case

| No | Aktor | *Use Case* | Deskripsi |
|---|---|---|---|
| 1. | *User* | *Login* | Menjelaskan aktor dapat melakukan askes ke aplikasi |
| 2. | *User* | *Log Out* | Menjelaskan kegiatan aktor keluar dari aplikasi dan menghapus data *user* dari aplikasi |

| | | | |
|---|---|---|---|
| 3. | *User* | Melihat data *dashboard* | Menjelaskan aktor dapat melihat data yang terdapat di halaman utama |
| 4. | *User* | Mengelola data profil | Menjelaskan aktor dapat melihat data akun dan menghubah data akun |
| 5. | *User* | Mengelola data *supplier* | Menjelaskan aktor yang dapat melihat daftar *supplier*, membuat, mengubah dan menghapus data *supplier* |
| 6. | *User* | Mengelola data *customer* | Menjelaskan aktor yang dapat melihat daftar *customer*, membuat, mengubah dan menghapus data *customer* |
| 7. | *User* | Mengelola data produk | Menjelaskan aktor yang dapat melihat daftar produk, membuat, mengubah dan menghapus data produk |
| 8. | *User* | Mengelola data pesanan | Menjelaskan aktor yang dapat melihat daftar pesanan, membuat dan mengubah status pesanan |
| 9. | *User* | *Register* | Menjelaskan aktor membuat akun baru untuk mendapat akses ke aplikasi |

#### 4.2.3.3. Diagram *Use Case*




Gambar 4. 4 Diagram Use Case Aplikasi Dropship

#### 4.2.3.4. Skenario *Use Case*

1. *Use Case Login*

Tabel 4. 3 Use Case Login

| ***Use Case*** | *Login* |
|---|---|
| ***Actor*** | *User* |
| ***Precondition*** | Aktor telah melakukan *register* akun |
| ***Postcondition*** | Aktor berhasil *login* |
| ***Description*** | |
| Menjelaskan aktor dapat melakukan askes ke aplikasi melalui halaman *login* | |
| ***Main Scenario*** | |
| ***Actor Action*** | ***System Response*** |
| 1. Masuk ke dalam aplikasi atau memasukan halaman *Login* pada *address bar* | 2. Menampilkan halaman *Login* |

| | |
|---|---|
| 3. Mengisi data email dan *password* | 4. Melakukan validasi format data yang diisi, jika tidak sesuai muncul pesan *error* di bawah kolom *input* |
| 5. Menekan tombol *Login* | 6. Aplikasi berpindah ke halaman *Home* |

2. *Use Case Register*

Tabel 4. 4 Tabel Use Case Register

| | |
|---|---|
| ***Use Case*** | *Register* |
| ***Actor*** | *User* |
| ***Precondition*** | - |
| ***Postcondition*** | Aktor berhasil terdaftar dan mendapat askes |
| ***Description*** | |
| Menjelaskan aktor membuat akun baru untuk mendapat akses ke aplikasi | |
| ***Actor Action*** | ***System Response*** |
| 1. Masuk ke dalam aplikasi atau memasukan halaman *Login* pada *address bar* | 2. Menampilkan halaman *Login* |
| 3. Menekan tombol *Register* | 4. Menampilkan *pop up* formulir registrasi akun |
| 5. Mengisi formulir registrasi | 6. Melakukan validasi format data yang diisi, jika tidak sesuai muncul pesan *error* di bawah kolom *input* |
| 7. Menekan tombol *Create Account* | 8. Menampikan pesan sukses jika akun berhasil dibuat |

3. *Use Case Melihat Data Dashboard*

Tabel 4. 5 Use Case Melihat Data Dashboard

| ***Use Case*** | *Melihat data dashboard* |
|---|---|
| ***Actor*** | *User* |
| ***Pre-condition*** | Aktor telah melakuakn login |
| ***Post-condition*** | Aktor dapat melihat halaman *Home* dan data dashboard |
| ***Description*** | |
| Menjelaskan aktor dapat melihat data yang terdapat di halaman utama | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu Home pada *bottom bar* | 2. Menampilkan halaman *Home* |

4. *Use Case Mengelola Data Profil*

Tabel 4. 6 Use Case Mengelola Data Profil

| ***Use Case*** | *Mengelola data profil* |
|---|---|
| ***Actor*** | *User* |
| ***Pre-condition*** | Aktor telah melakukan login |
| ***Post-condition*** | Aktor dapat melihat halaman *Profil* dan *link* toko tersimpan di *database* |
| ***Description*** | |
| Menjelaskan aktor dapat melihat data yang terdapat di halaman utama | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu *Home* pada *bottom bar* | 2. Menampilkan halaman *Home* |
| 3. Menekan tombol *Insert Store Link* | 4. Muncul *pop up* formulir |

| | |
|---|---|
| 5. Mengisi data formulir | |
| 6. Menekan tombol *Save* | 7. Muncul *pop up* sukses |

5. *Use Case Mengelola Data Supplier*

Tabel 4. 7 Use Case Mengelola Data Supplier

| ***Use Case*** | *Mengelola data supplier* |
|---|---|
| ***Actor*** | *User* |
| ***Pre-condition*** | Aktor sudah melakukan *login* |
| ***Post-condition*** | Data *supplier* tersimpan di dalam *firebase firestore* |
| ***Description*** | |
| Menjelaskan aktor yang dapat melihat daftar *supplier*, membuat, mengubah dan menghapus data *supplier* | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu "*Store"* pada *bottom bar* | 2. Menampilkan halaman "*Store"* |
| 3. Memilih menu "*Supplier"* | 4. Menampilkan daftar data *supplier* |
| 5. Menekan tombol dengan *icon* tambah | 6. Menampilkan halaman "*Add Supplier"* |
| 7. Mengisi data *supplier* dan menekan tombol "*Save*" | 8. Menampilkan *pop up* sukses dan kembali ke halaman "*Store*" |
| 9. Menekan kolom data *supplier* | 10. Menampilkan halaman *"Supplier Detail"* |
| 11. Menekan tombol "*edit*" | 12. Kolom formulir data *supplier* dapat diubah |

| | |
|---|---|
| 13. Mengubah data *supplier* dan menekan tombol "*Save*" | 14. Menyimpan data yang diubah ke *database* dan kembali ke halaman "*Store*" |
| 15. Menekan dan menahan kolom data *supplier* | 16. Menampilkan *pop up* konfirmasi hapus data *supplier* |
| 17. Menekan tombol "*Yes*" | 18. Data *supplier* dihapus |

6. *Use Case* Mengelola Data *Customer*

Tabel 4. 8 Use Case Mengelola Data Customer

| | |
|---|---|
| ***Use Case*** | *Mengelola data customer* |
| ***Actor*** | *User* |
| ***Pre-condition*** | Aktor sudah melakukan *login* |
| ***Post-condition*** | Data *customer* tersimpan di dalam *firebase firestore* |
| ***Description*** | |
| Menjelaskan aktor yang dapat melihat daftar *supplier*, membuat, mengubah dan menghapus data *supplier* | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu "*Store"* pada *bottom bar* | 2. Menampilkan halaman "*Store"* |
| 3. Memilih menu "*Customer"* | 4. Menampilkan daftar data *customer* |
| 5. Menekan tombol dengan *icon* tambah | 6. Menampilkan halaman "*Add Customer"* |
| 7. Mengisi data *customer* dan menekan tombol "*Save*" | Menampilkan *pop up* sukses dan kembali ke halaman "*Store*" |
| 8. Menekan kolom data *customer* | 9. Menampilkan halaman *"Supplier Detail"* |

| | |
|---|---|
| 10. Menekan tombol "*edit*" | 11. Kolom formulir data *supplier* dapat diubah |
| 12. Mengubah data *customer* dan menekan tombol "*Save*" | 13. Menyimpan data yang diubah ke *database* dan kembali ke halaman "*Store*" |
| 14. Menekan dan menahan kolom data *customer* | 15. Menampilkan *pop up* konfirmasi hapus data *customer* |

7. *Use Case* Mengelola Data Produk

Tabel 4. 9 Use Case Mengelola Data Produk

| | |
|---|---|
| ***Use Case*** | *Mengelola data* produk |
| ***Actor*** | *User* |
| ***Pre-condition*** | 1. Aktor sudah melakukan *login*<br>2. Aktor sudah membuat data *supplier* |
| ***Post-condition*** | Data produk tersimpan di dalam *firebase firestore* |
| ***Description*** | |
| Menjelaskan aktor yang dapat melihat daftar produk, membuat, mengubah dan menghapus data produk | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu "*Products"* pada *bottom bar* | 2. Menampilkan halaman "*Products"* dan daftar data produk |
| 3. Menekan tombol dengan *icon* tambah | 4. Menampilkan halaman "*Add Product"* |
| 5. Mengisi data produk dan menekan tombol "*Save*" | Menampilkan *pop up* sukses dan kembali ke halaman "*Products*" |

| | |
|---|---|
| 6. Menekan kolom data produk | 7. Menampilkan halaman *"Product Detail"* |
| 8. Menekan tombol "*edit*" | 9. Kolom formulir data produk dapat diubah |
| 10. Mengubah data produk dan menekan tombol "*Save*" | 11. Menyimpan data yang diubah ke *database* dan kembali ke halaman "*Products*" |
| 12. Menekan dan menahan kolom data produk | 13. Menampilkan *pop up* konfirmasi hapus data produk |
| 14. Menekan tombol "*Yes*" | 15. Data produk dihapus |

8. Use Case Mengelola Data Pesanan

Tabel 4. 10 Use Case Mengelola Data Pesanan

| | |
|---|---|
| ***Use Case*** | *Mengelola data* pesanan |
| ***Actor*** | *User* |
| ***Pre-condition*** | 1. Aktor sudah melakukan login<br>2. Aktor sudah membuat data *supplier*, produk da *customer* |
| ***Post-condition*** | Data pesanan tersimpan di dalam *firebase firestore* |
| ***Description*** | |
| Menjelaskan aktor yang dapat melihat daftar produk, membuat, mengubah dan menghapus data produk | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu "*Orders*" pada *bottom bar* | 2. Menampilkan halaman "*Orders*" dan daftar pesanan |
| 3. Menekan tombol dengan *icon* tambah | 4. Menampilkan halaman "Add Order" |

| | |
|---|---|
| 5. Mengisi formulir pesanan dan menekan tombol "*Checkout*" | 6. Menampilkan pop up sukses dan Kembali ke halaman "*Orders*" |
| 7. Menekan kolom data produk | 8. Menampilkan halaman *"Order Detail"* |
| 9. Melakukan konfirmasi status pesanan dan memasukan resi pesanan | 10. Menampilkan *pop up* sukses dan menampilkan halaman "*Orders*" |

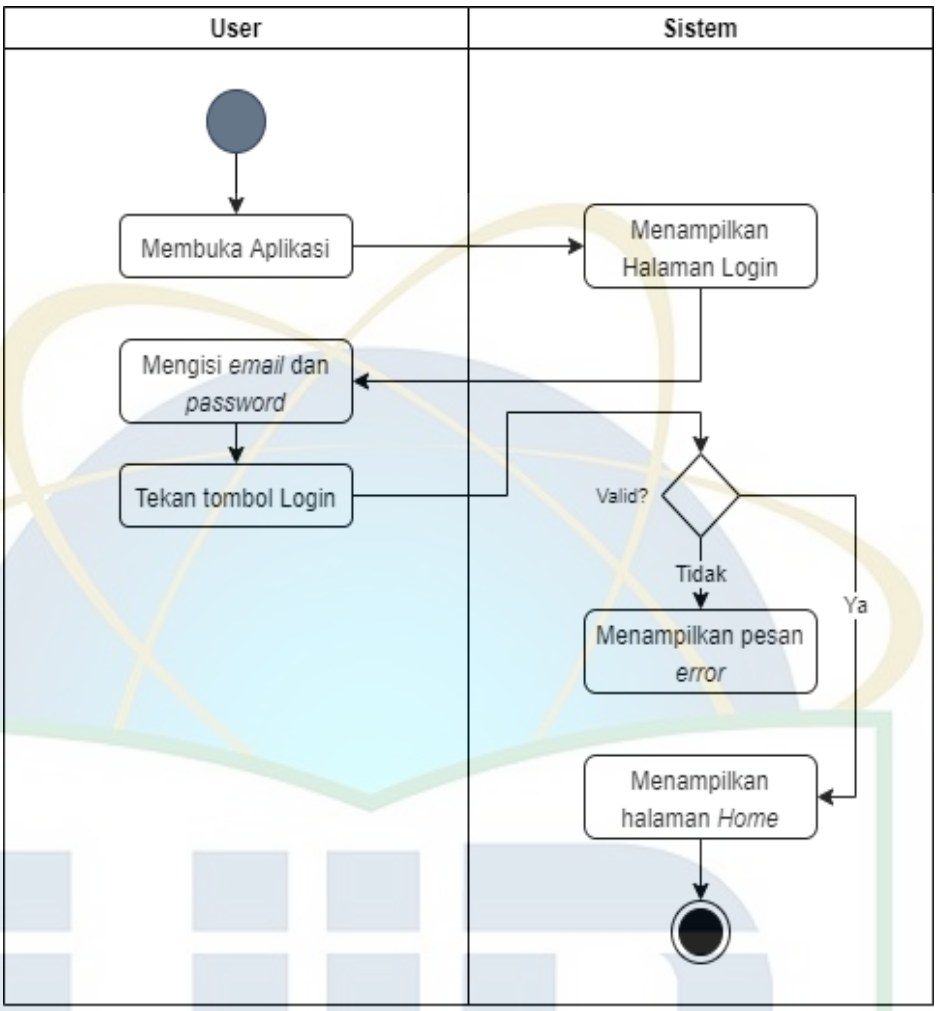
9. Use Case *Log Out*

Tabel 4. 11 Use Case Log Out

| | |
|---|---|
| ***Use Case*** | *Log Out* |
| ***Actor*** | *User* |
| ***Pre-condition*** | Aktor sudah melakukan login |
| ***Post-condition*** | *Cache* aplikasi dihapus dari aplikasi |
| ***Description*** | |
| Menjelaskan kegiatan aktor keluar dari aplikasi dan menghapus data *user* dari aplikasi | |
| ***Actor Action*** | ***System Response*** |
| 1. Menekan menu "*Profile*" pada *bottom bar* | 2. Menampilkan halaman "*Profile*" dan daftar pesanan |
| 3. Menekan tombol "*Log Out*" | 4. Menampilkan halaman "*Login*" |

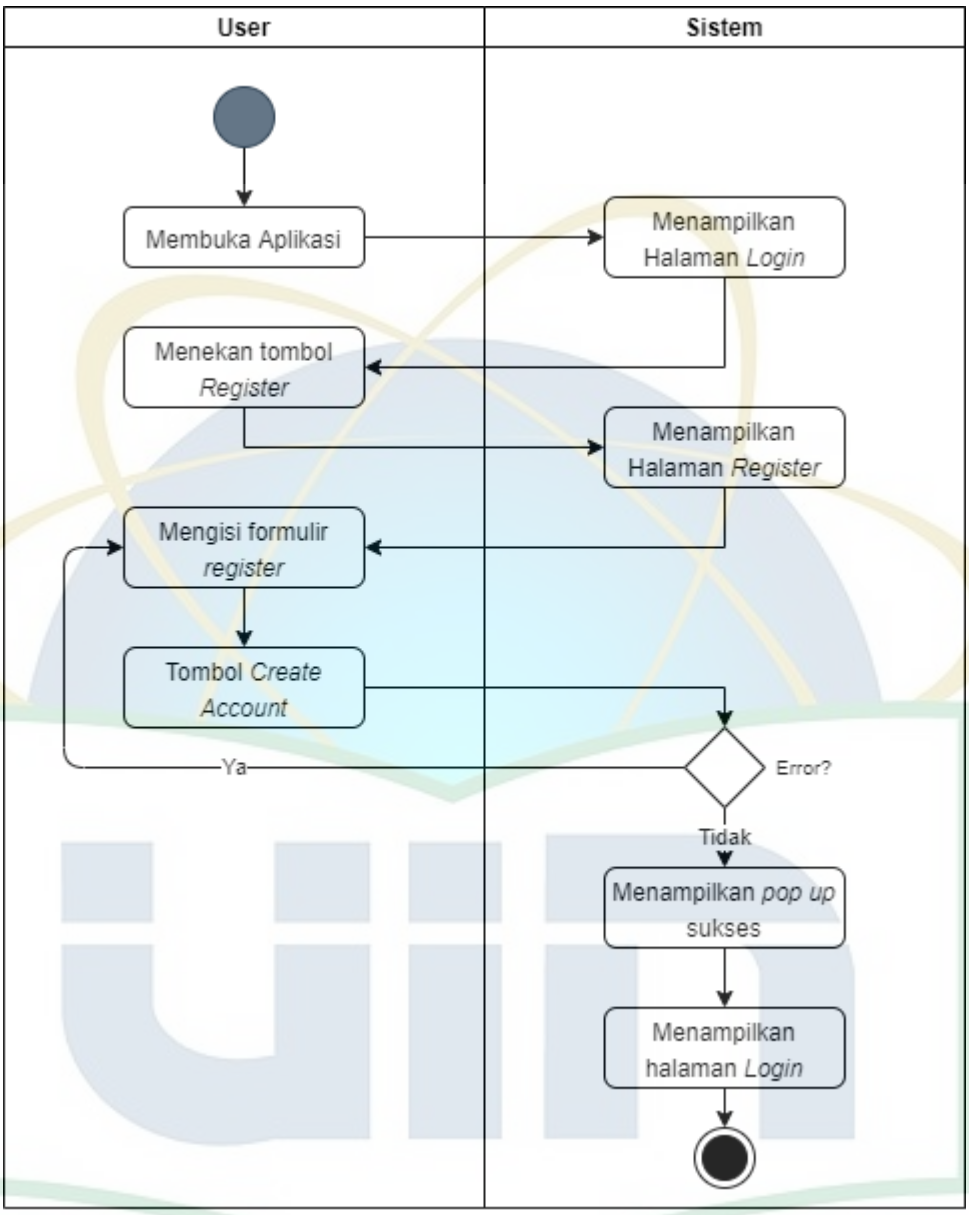
### 4.2.4. *Activity Diagram*

#### 4.2.5.1. *Activity Diagram Login*



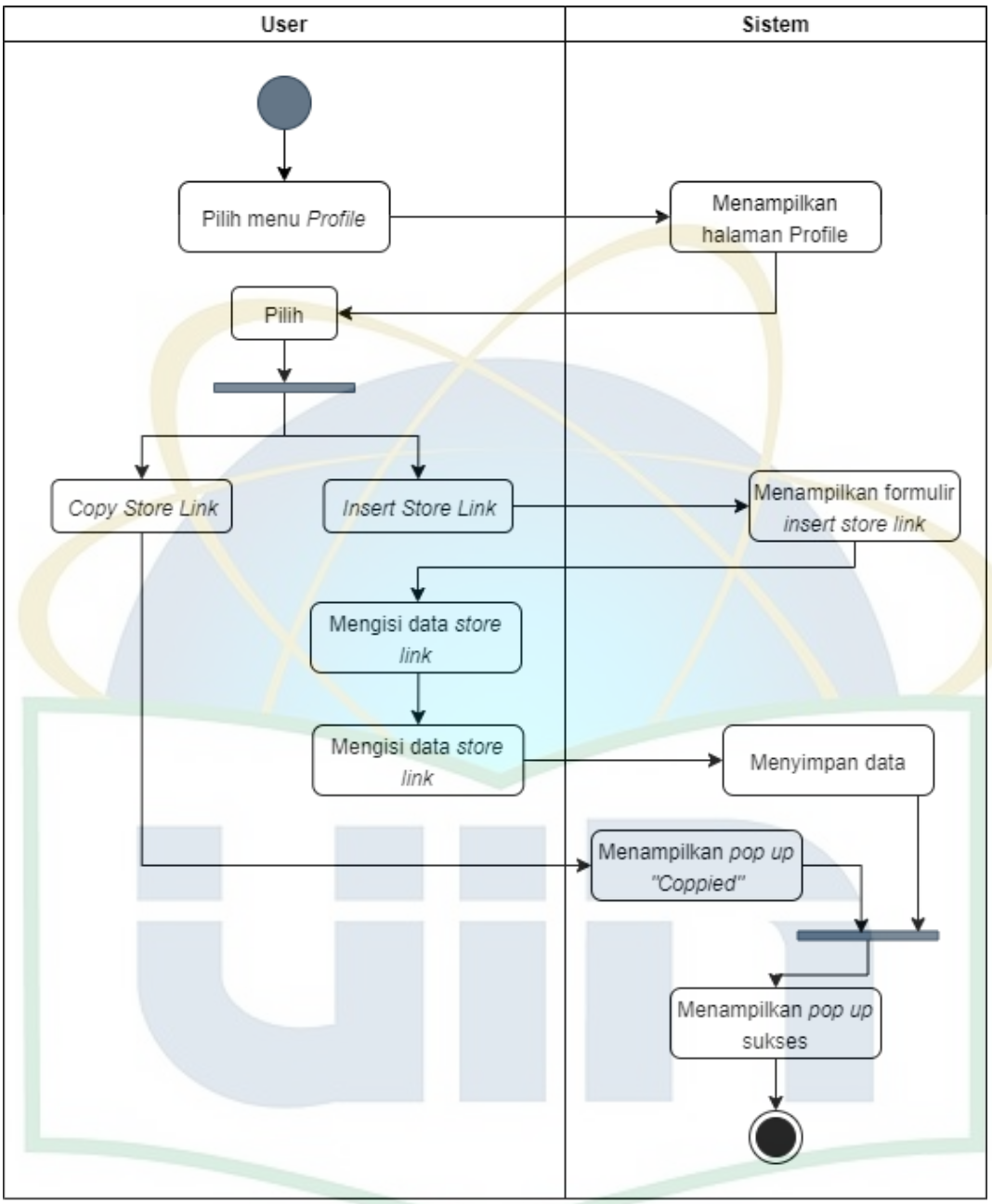

Gambar 4. 5 Activitiy Diagram Login

#### 4.2.5.2. Activity Diagram Register



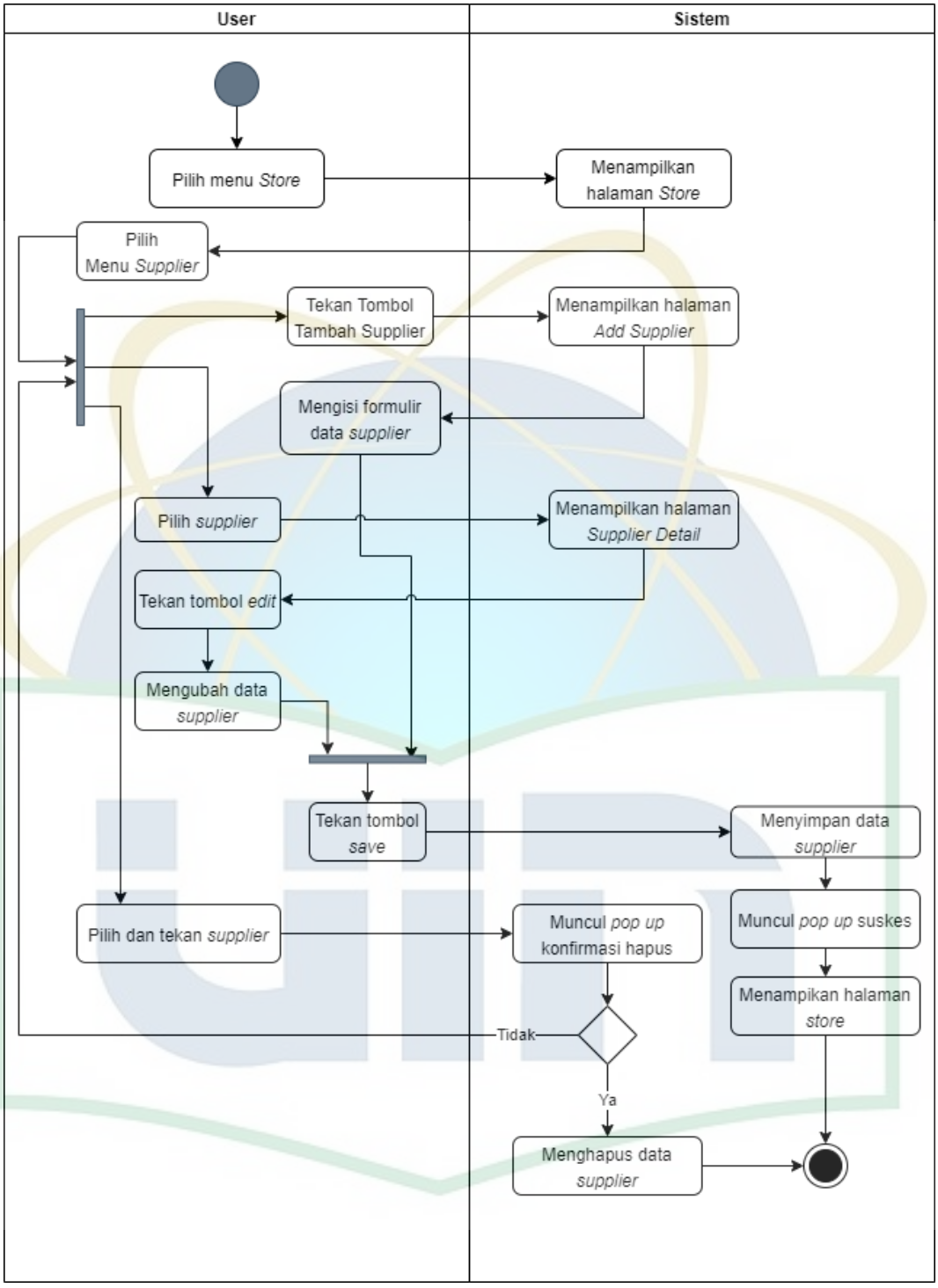

Gambar 4. 6 Activitiy Diagram Register

### 4.2.5.3. *Activity Diagram* Mengelola Data Profil



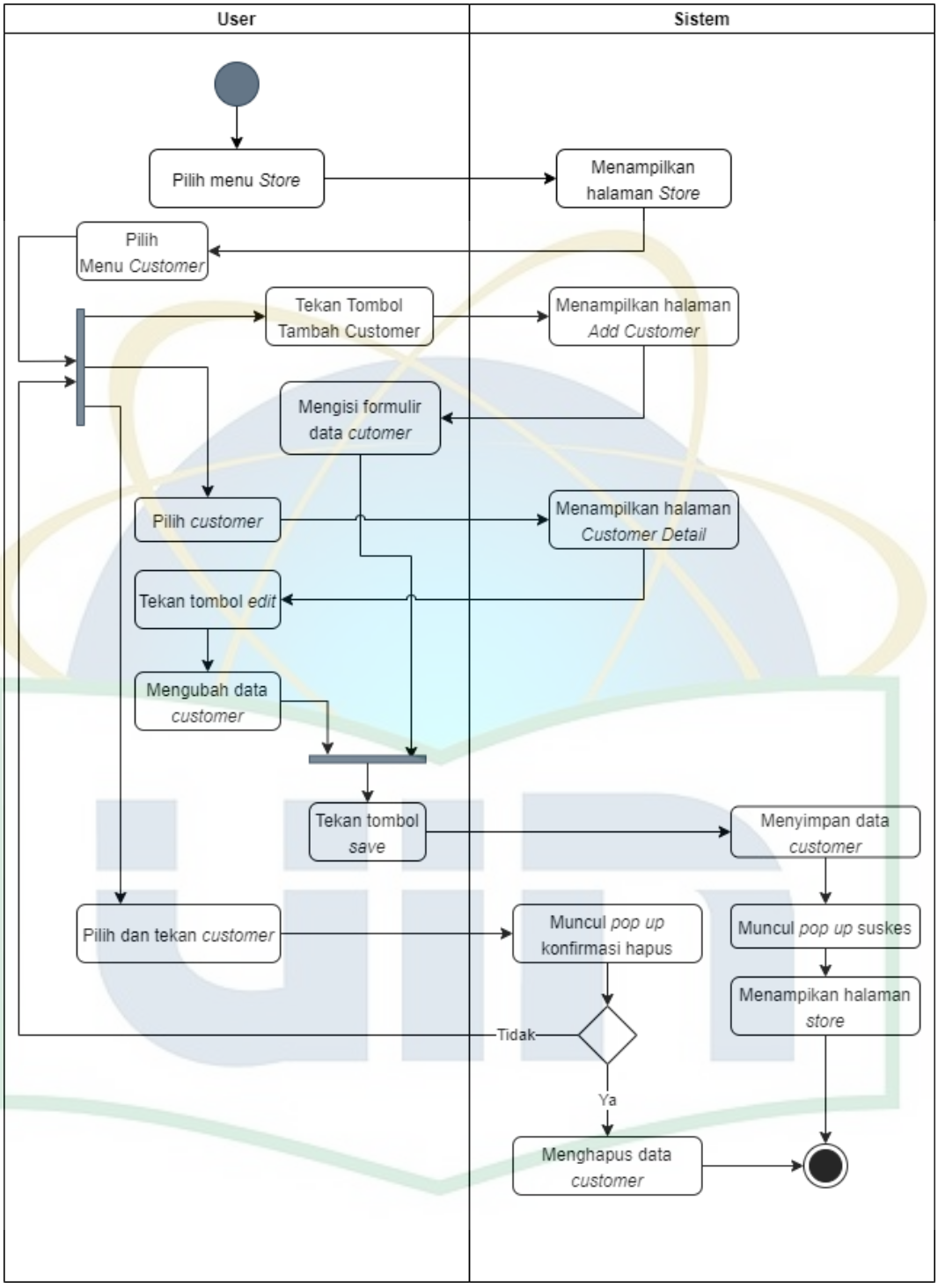

Gambar 4. 7 Activitiy Diagram Mengelola Data Profil

### 4.2.5.4. *Activity Diagram* Mengelola Data *Supplier*



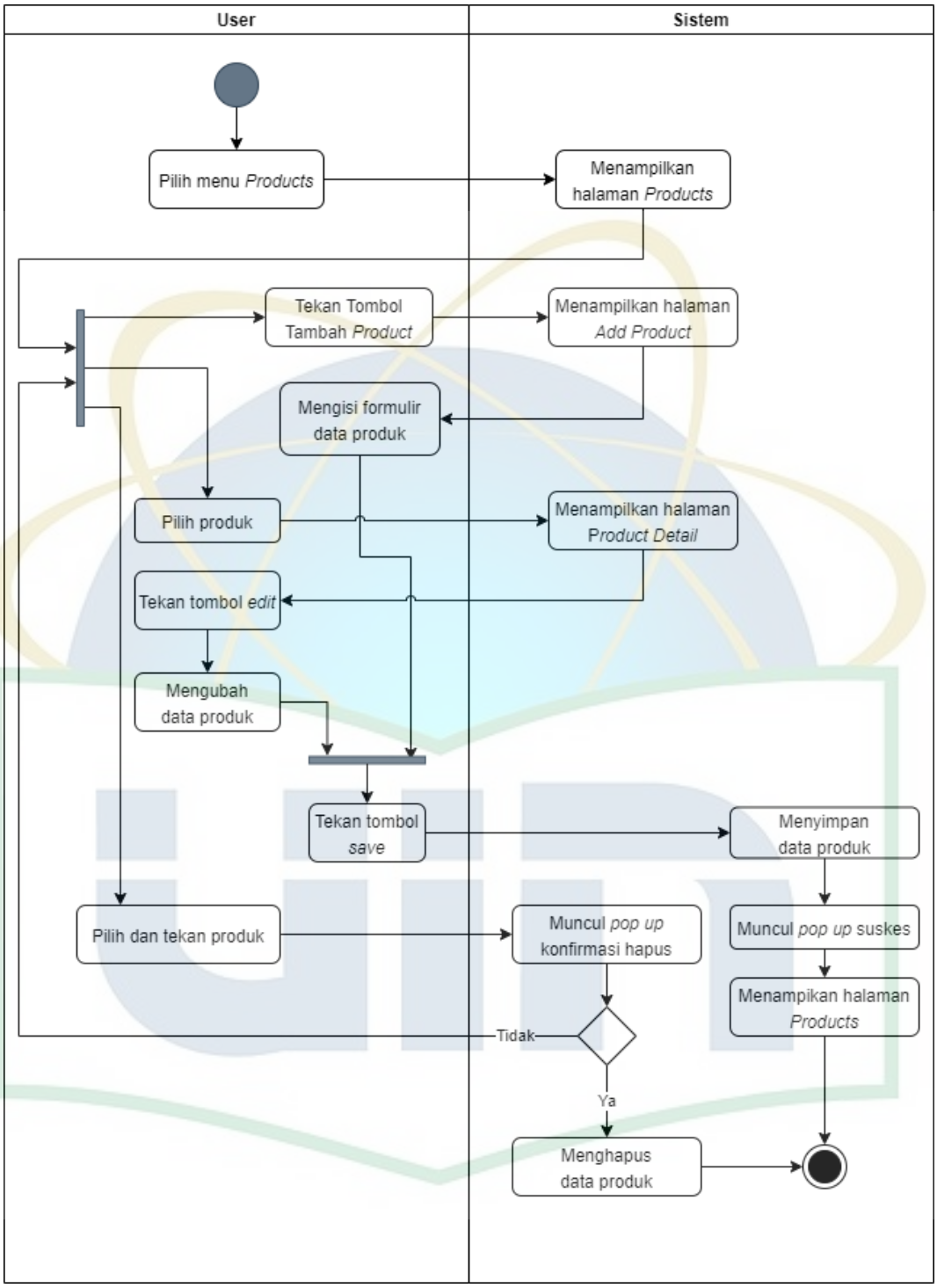

Gambar 4. 8 Activitiy Diagram Mengelola Data Supplier

### 4.2.5.5. *Activity Diagram* Mengelola Data *Customer*



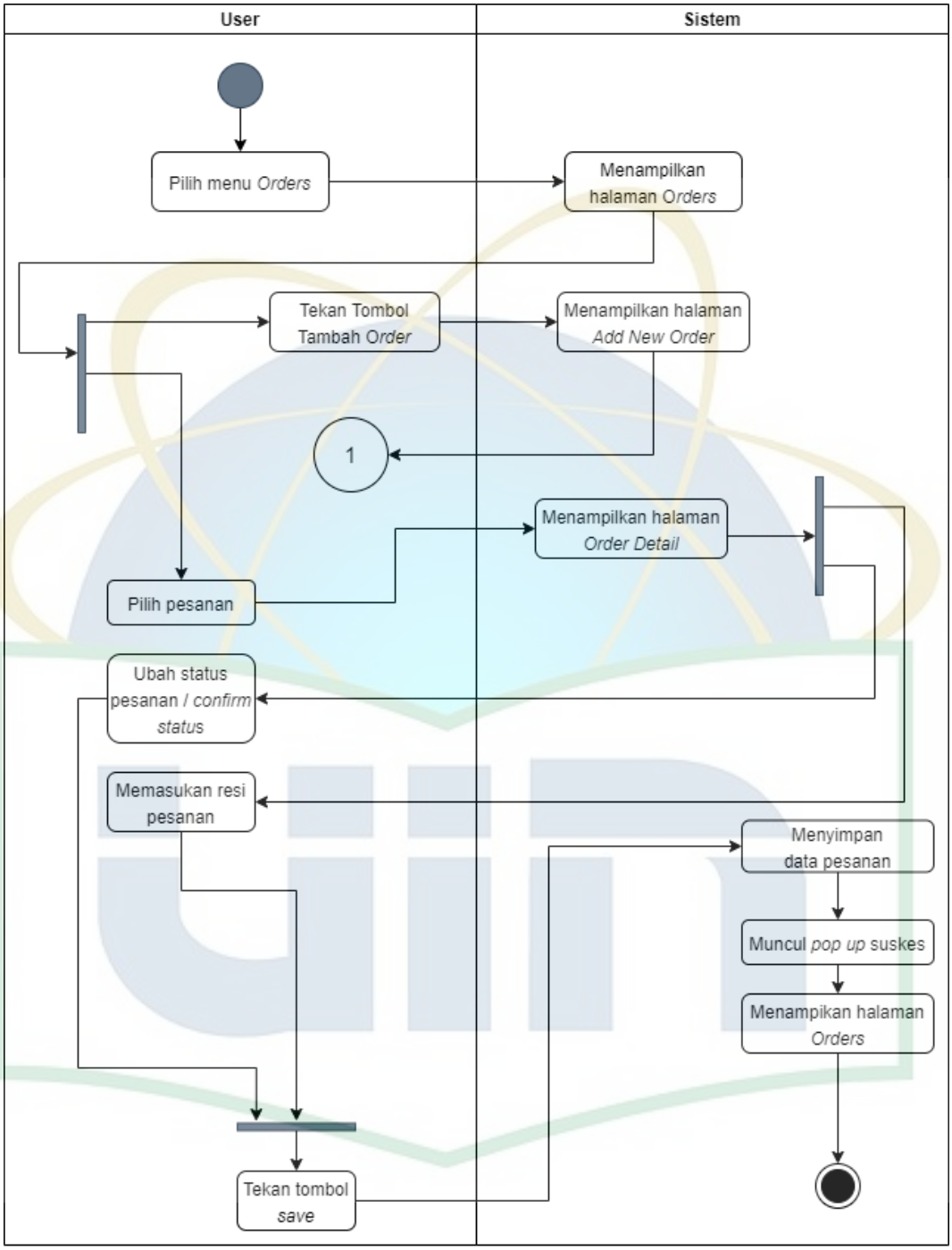

Gambar 4. 9 Activitiy Diagram Mengelola Data Customer

### 4.2.5.6. *Activity Diagram Mengelola Data Produk*

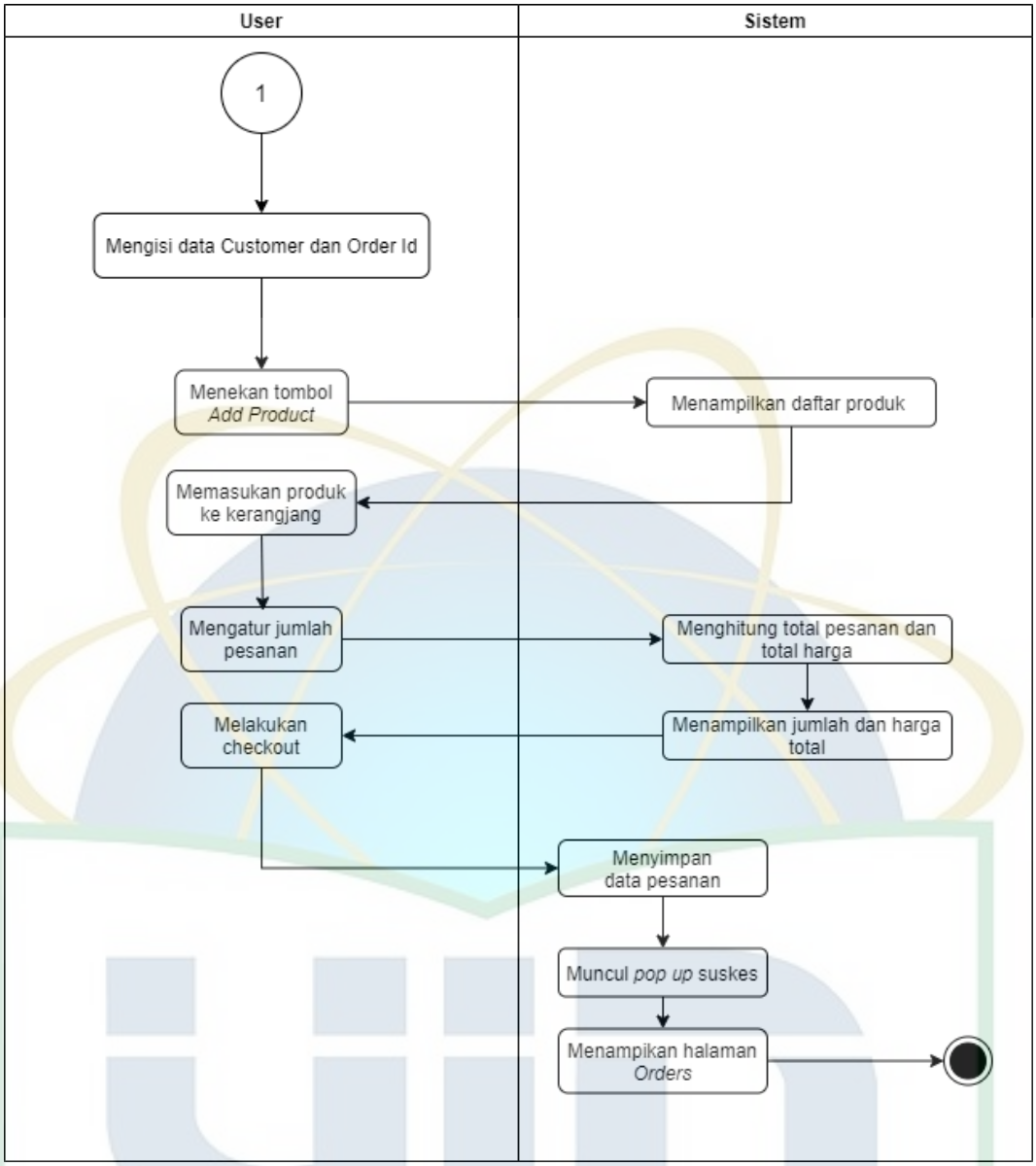



Gambar 4. 10 Activitiy Diagram Mengelola Data Produk

### 4.2.5.7. Activity Diagram Mengelola Data Pesanan



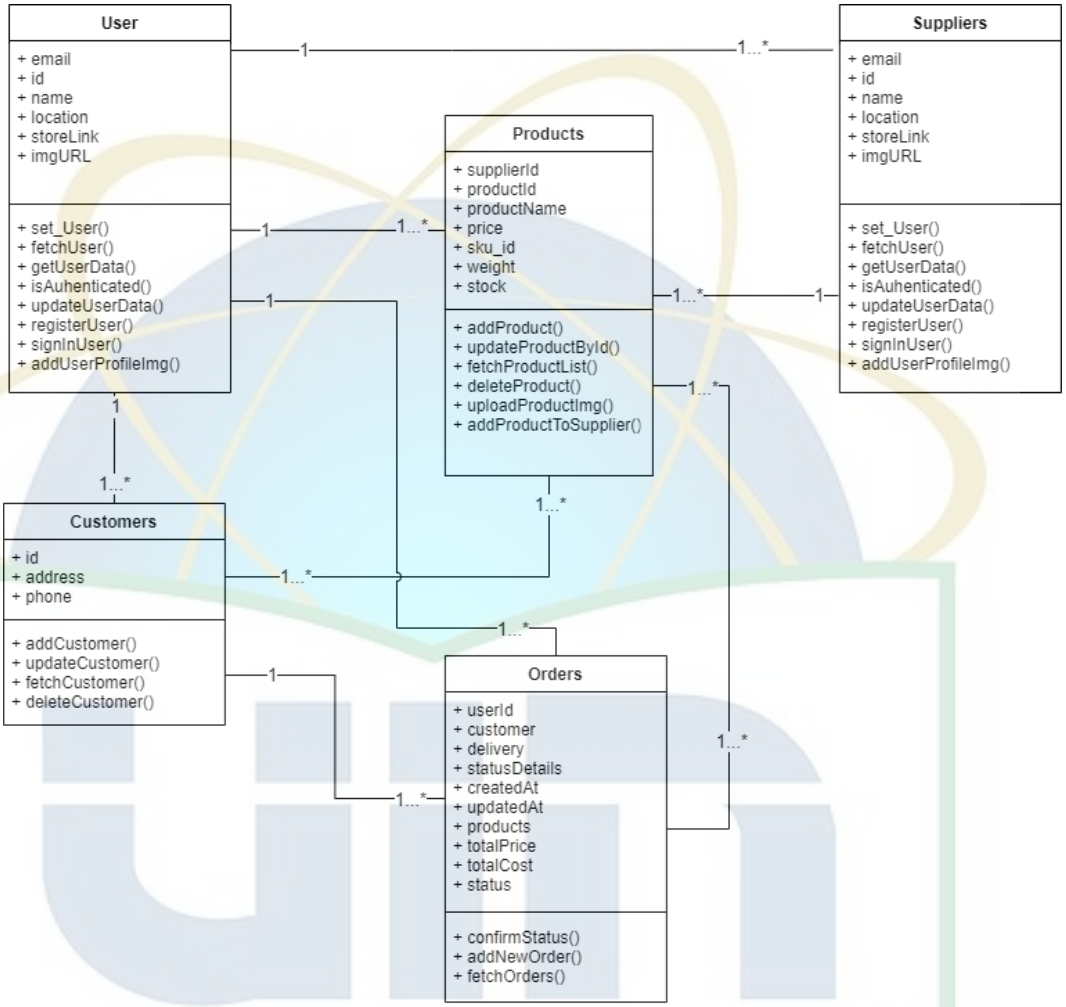

Gambar 4. 11 Activitiy Diagram Mengelola Data Pesanan




Gambar 4. 12 Activitiy Diagram Tambah Pesanan

#### 4.2.5.8. *Class Diagram*

Untuk menjelaskan struktur objek yang terdapat dalam aplikasi dibutuhkan *class diagram, class* diagram memberikan gambaran hubungan antar objek kelas dan menjelaskan atribut serta fungsi yang dapat dilakukan. Berikut struktur kelas objek yang digambarkan melalui *class diagram*:

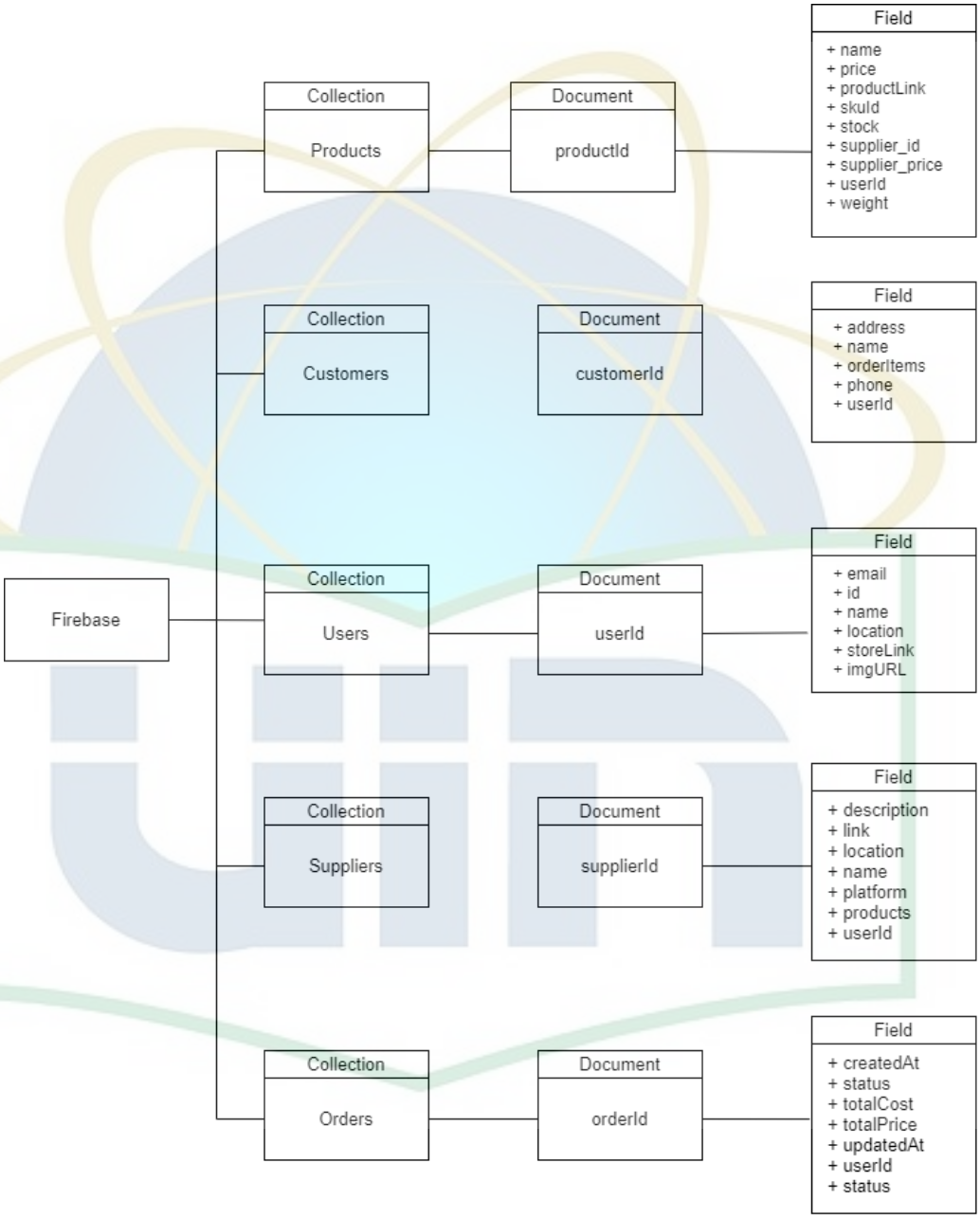



Gambar 4. 13 Class Diagram Basis Data Aplikasi Dropship

### 4.2.5. Perancangan Basis Data

Basis data yang digunakan di penelitian ini adalah *Firebase Firestore*, *database* ini memiliki struktur seperti *database* noSQL yang dimana direpresentasikan dengan pasangan *key* dan *value*. Database ini menggunakan penulisan pada JSON, data yang disimpan di firebase dapat

dilihat sebagai *document*. Kumpulan document disimpan ke dalam *collection* dapat memudahkan kita mengatur data dan membuat kueri. Perancangan basis data ini akan lebih cocok jika digambarkan menggunakan diagram pohon seperti berikut:




Gambar 4. 14 Struktur Basis Data Firebase Firestore

### 4.2.6. Perancangan Tampilan Aplikasi

#### 4.2.7.1. Tampilan Halaman *Login*


Gambar 4. 15 Desain Halaman Login

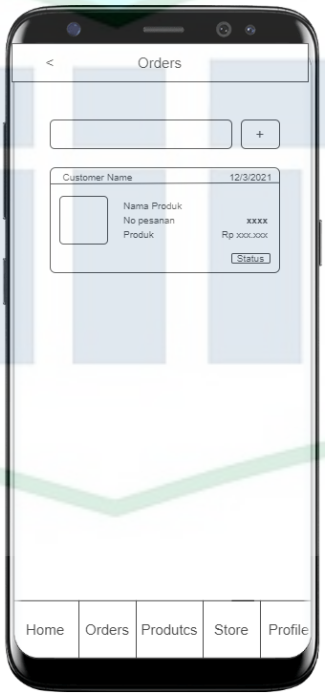
#### 4.2.7.2. Tampilan Halaman *Register*


Gambar 4. 16 Desain Halaman Register

### 4.2.7.3. Tampilan Halaman *Home*


Gambar 4. 17 Desain Halaman Home

### 4.2.7.4. Tampilan Halaman *Orders*




Gambar 4. 18 Desain Halaman Orders

### 4.2.7.5. Tampilan Halaman *Add Order*


Gambar 4. 19 Desain Halaman Add Order

### 4.2.7.6. Tampilan Halaman *Products*


Gambar 4. 20 Tampilan Halaman Products

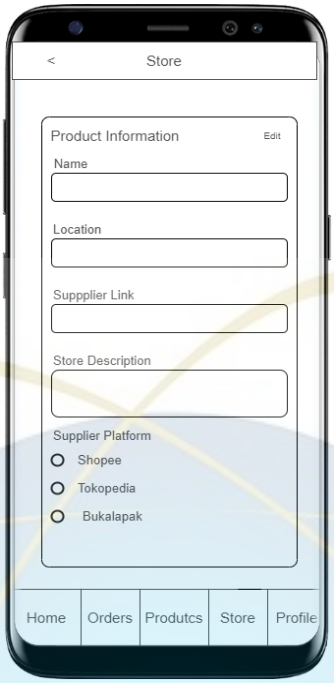
### 4.2.7.7. Tampilan Halaman *Add Product*


Gambar 4. 21 Desain Halaman Add Product

### 4.2.7.8. Tampilan Halaman *Store*


Gambar 4. 22 Desain Halaman Store

### 4.2.7.9. Tampilan Halaman *Add Supplier*




Gambar 4. 23 Desain Halaman Add Supplier

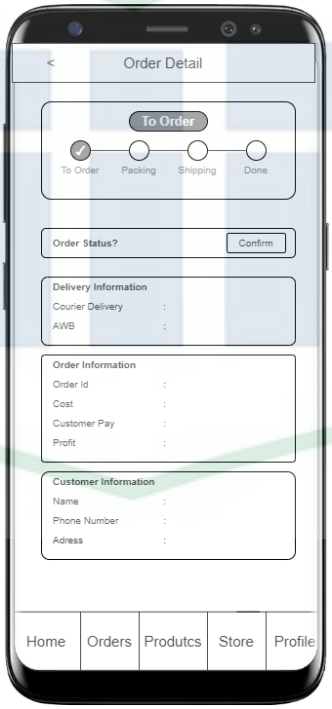
### 4.2.7.10. Tampilan Halaman *Customer*


Gambar 4. 24 Desain Halaman Add Customer

#### 4.2.7.11. Tampilan Halaman *Profile*

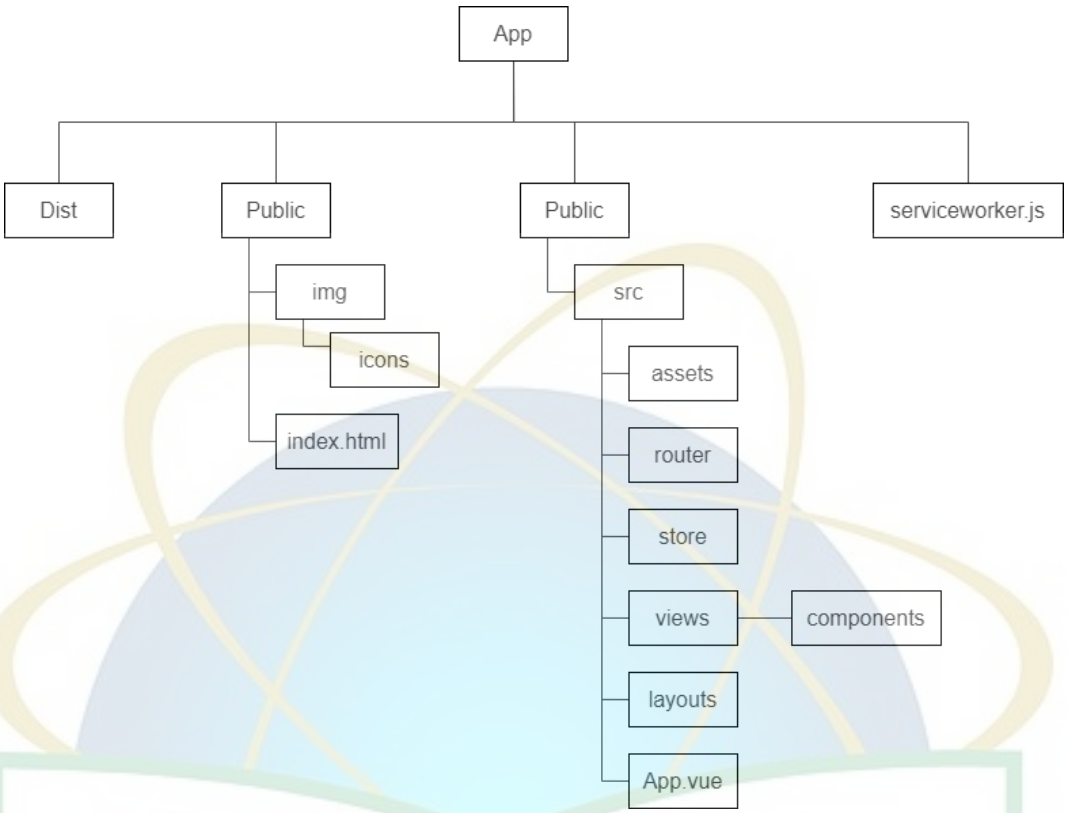

Gambar 4. 25 Desain Halaman Profile

#### 4.2.7.12. Tampilan Halaman Order Detail




Gambar 4. 26 Desain Halaman Order Detail

### 4.2.7. Perancangan Struktur *File* Aplikasi




Gambar 4. 27 Desain Struktur File Aplikasi

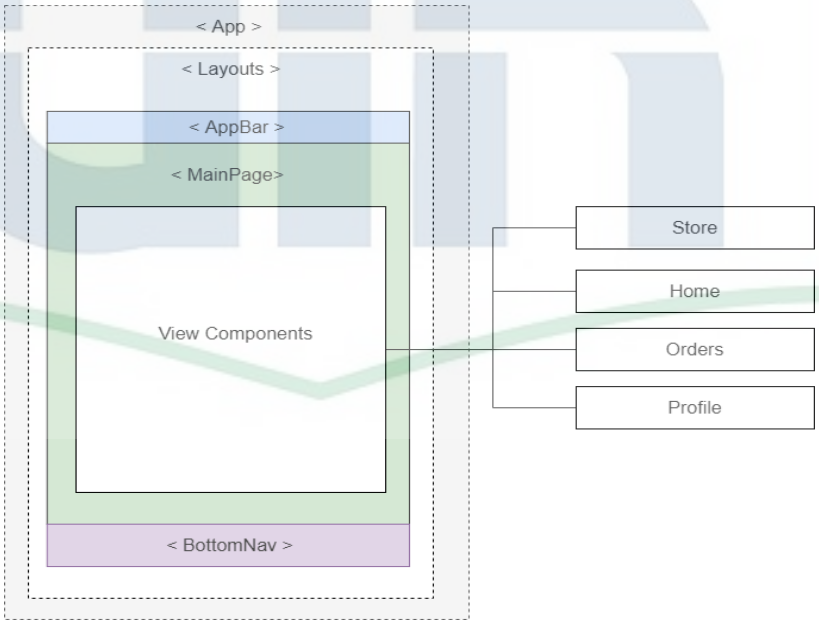
Struktur aplikasi *dropship* menggunakan *framework vuejs* dan Progressive Web App (PWA) dapat digambarkan seperti gambar pada gambar di atas. Berikut penjelasan struktru aplikasi yang ada:

1. App: Merupakan folder ini dari aplikasi
2. Dist: Tempat hasil akhir setelah aplikasi memasuki *production mode* dan *folder* yang akan di-*deploy* ke dalam *firebase hosting*
3. Public: Tempat menyimpan index.html yang nanti menjadi *file* dasar ketika melakukan *build* aplikasi vuejs dab terdapat folder image untuk menyimpan gambar *icon*
4. Src: Folder yang berisi *file* dan *folder* utama vuejs
5. Assets: Tempat menyimpan *file static*

6. Router: Tempat menangani navigasi, melakukan validasi akses aplikasi
7. Store: Tempat menyimpan state dan fungsi aplikasi. Terdapat file state.js, getters.js dan actions.js yang sudah dijelaskan pada Bab II
8. Views: Tempat halaman aplikasi disimpan terdapat folder *components* yang berperan sebagai *component* pendukung yang dipakai pada halaman aplikasi
9. Layouts: Menyimpan struktur dari aplikasi, dapat disebut App Shell dari aplikasi, akan lebih lanjut dijelaskan pada sub-bab berikutnya
10. App.vue: File ini dari vuejs, merupakan tempat berjalannya *instance* dari vuejs

### 4.2.8. Perancangan Struktur *Application Shell* (*App Shell*)

Aplikasi *mobile* indentik dengan sistem navigasi yang mulus, berpindah dari satu halaman ke halaman lain di dalam aplikasi *mobile* tidak memerlukan memuat ulang konten yang ada di dalamnya. Teknologi *web*

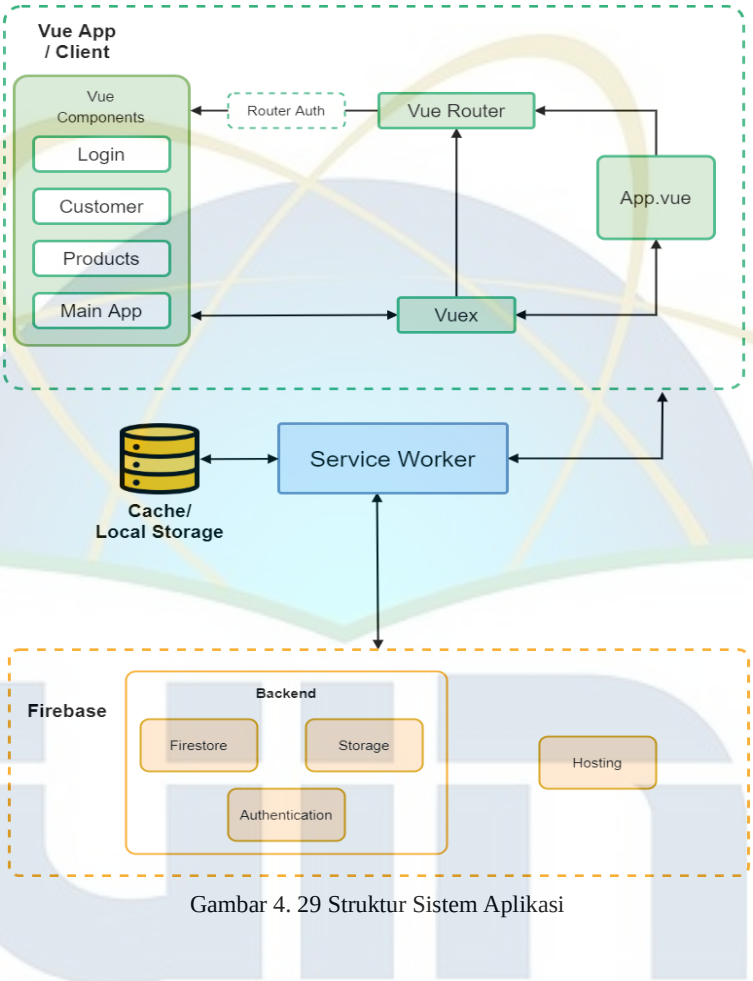



Gambar 4. 28 Desain Struktur App Shell

juga dapat memberikan pengalaman serupa dengan membuat struktur *App Shell* pada aplikasi yang kita buat, dengan menggunakan Vuejs kita bisa membuat *Single Page Application.*

## 4.3. Implementasi

### 4.3.1. Struktur Aplikasi




Gambar 4. 29 Struktur Sistem Aplikasi

Struktur pada aplikasi manajemen data dropship dibagi menjadi dua bagian utama, yaitu bagian client yang dapat dilihat oleh pengguna dan bagian *backend firebase*. Aplikasi tersebut dihubungkan oleh *service* worker untuk mengatur jaringan data keluar dan masuk aplikasi. *Service* worker juga bertugas untuk melakukan *caching* aset serta menyimpan data yang diperlukan oleh aplikasi sehingga aplikasi dapat diakses secara *offline* dan dapat di-*install* layaknya aplikasi *mobile* umumnya.

Proses penyimpanan data akan dilakukan oleh *firebase* pada ketiga layanan yang disediakan seperti *firebase firestore*, *firebase storage*. Selain itu digunakan juga firebase auth untuk melakukan verifikasi terhadap pengguna aplikasi. Aplikasi dapat diakses melalui internet dengan cara menaruh file dan menjalankannya pada *firebase hosting*.

Sisi client atau tampilan aplikasi, vuejs digunakan untuk memberikan kemudahan membuat aplikasi berbasis *single page application*, dengan menggunakan SPA tersebut aplikasi akan semakin terlihat mirip aplikasi umum pada perangkat *mobile*. *Client* atau *browser* akan mengambil data dari *firebase hosting* lalu *service worker* akan mengatur data yang akan ditampilkan dengan menggunakan data yang sudah ter-*cache* ataupun data yang baru diterima dari server. *Browser* akan memanggil App.vue sebagai thread utama dalam menjalankan aplikasi vuejs, selanjutnya vuejs akan manggil halaman sesuai *router* yang sudah didefiniskan. Router akan melakukan otentikasi pengguna dengan melakukan validasi data pengguna yang tersedia di vuex. Vuex akan mengatur hampir keseluruhan data yang digunakan oleh aplikasi vue, di dalam vuex data mengalami perubahan dan mengirimkan *request* terhadap *server*.

### 4.3.2. Implementasi *Vuejs*

Aplikasi ini dibuat menggunakan *framework vuejs*, framework ini memudahkan kita dalam membangung aplikasi khususnya *frontend* dengan lebih cepat dan lebih *powerful*. Karena penelitian ini diharapkan dapat memberi solusi terhadap pembuatan aplikasi *mobile* melalui teknologi *web*, vuejs menawarkan banyak kelebihan yang bisa diimplementasikan. Berikut cara mengimplementasikan vuejs ke dalam aplikasi manajemen *dropship*.

#### 4.3.2.1. Memulai Projek

Untuk membuat aplikasi berbasis vuejs tidaklah sulit, kita cukup memanggil *command* pada terminal maupun command line seperti di bawah:

Dengan menginstall vue/cli kita dapat mengakses beragam *tools, plugins* dan *extension* yang ada pada ekosistem vuejs. Selanjutnya kita perlu membuat vue project, dengan menggunakan vue-cli kita bisa membuat projek baru dengan sekali *command*, berikut *command* untuk membuat projek vuejs:

```
npm install -g @vue/cli
```

#### 4.3.2.2. Membuat *Application Shell*

*Application Shell* menjadi salah satu komponen esensial di dalam pembuatan aplikasi menggunakan *Progressive Web App*, kita bisa mengimplementasikan *app shell* menggunakan vuejs dengan membuat struktur *layout* dengan komponen yang terpisah – pisah. Berikut cara implementasi *app shell* menggunakan vuejs,

1. *Install Vuetify*

```
vue create my-app
cd my-app
```

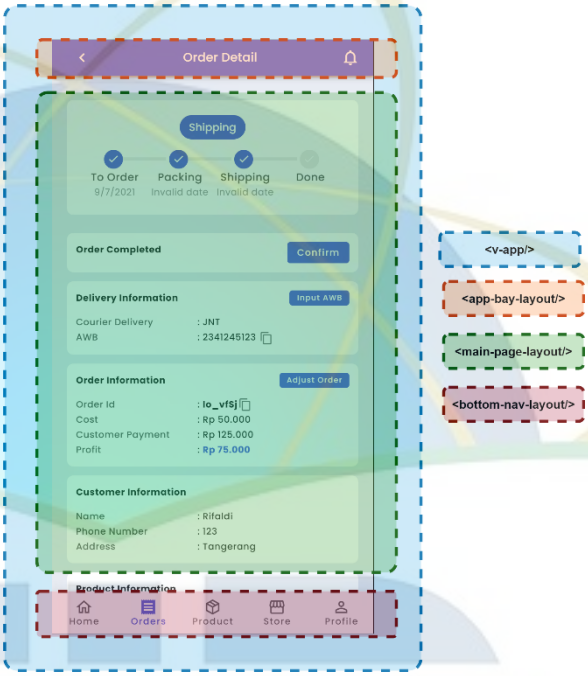
*Vuetify* menyediakan component *vuejs* yang beragam, dengan menggunakan *vuetify*, kita bisa menghemat waktu sangat banyak karena *vuetify* menyediakan sistem tema, komponen dan fungsionalitas yang siap pakai. Berikut cara meng-*install vuetify* melalui command line:

```
Vue add vuetify
```

Setelah vuetify sudah terpasang, selanjutnya kita bisa menggunakan component yang terdapat pada vuetify.

2. Membuat *Layout App Shell*

Seperti struktur app shell yang sudah dimodelkan pada gambar 4.26 kita perlu membuat folder khusus untuk menyimpan kerangka app shell tersebut. berdasarkan desain struktur file pada gambar 4.25 dalam aplikasi manajemen ini folder tersebut bernama "*layouts*". Berikut *screenshot* implementasinya:




Gambar 4. 30 Struktur App Shell pada Aplikasi

```
<template>
  <v-app id="mainContent">
    <!-- App Bar Navigation -->
    <app-bar-layout/>
    <!-- Main Page layout -->
    <main-page-layout />
    <!-- Bottom Navigation -->
    <bottom-nav-layout />
  </v-app>
</template>
```

Gambar 4. 31 Struktur App Shell pada Vuejs

### 4.3.3. Implementasi *Vuex* dan *Firebase*

Selanjutnya adalah dengan menambahkan *vuex* dan *firebase*, aplikasi kita perlu menggunakan *vuex* dan *firebase*. *Vuex* berperan sebagai pusat pangkalan data pada sistem local aplikasi kita, dengan menggunakan vuex kita dapat memprediksi perubahan data dan dapat menampikan data yang konsisten pada setiap halaman maupun *component*. Dengan menggunakan *vuex* dan *firebase*, aplikasi ini dapat melakukan fungsi basik CRUD (*create, read, update, delete*). Berikut cara implementasi *vuex* dan *firebase* ke dalam aplikasi:

1. Membuat struktur *vuex*

   *Vuex* terdiri dari beberapa bagian, seperti yang sudah dijelaskan pada bab sebelumnya. Komponen tersebut antara lain *state*, *getters*, *mutation* dan *actions*.

   a. *State*: *File* ini berisikan data yang disimpan dan digunakan di dalam aplikasi. *State* akan memberikan data yang konsisten pada setiap komponen dan data yang mengalami perubahan dapat diprediksi karena disimpan terpusat.

```
let state = {
  app: {
    notificationModal: {
      color: "",
      message: "Please Wait...",
      img: "",
      isActive: false,
      isLoading: false,
    },
  },
  user: {
    email: "",
    id: "",
    name: "",
    location: "",
    storeLink: "",
    imgURL: "",
  },
  products: {
    productList: [],
  },
 ....
};

export default state;
```

Gambar 4. 32 Implementasi Vuex State

b. *Getters*: *File* ini berfungsi untuk mengakses data dari *state* secara langsung tanpa mengubah nilai dari state tersebut. Getters menjadi solusi untuk melakukan fungsi *filter* atau mengembalikan data dengan format yang diinginkan tanpa mengubah *state*. Berikut contoh implementasi getters pada vuex:

```
const getters = {
  userData: (state) => state.user,
  isAuthenticated: (state) => (state.user.id ? true : false),
  allCustomers: (state) => state.store.customers.customerList,
  ordersByStatus: (_, getters) => (status) => {
    return getters.allOrders.filter((order) => {
      return order.status == status;
    });
  },
  getSupplierById: (_, getters) => (id) => {
    let supplierList = getters.allSuppliers;

    let selectedSupplier = supplierList.find((supplier) => supplier.id == id);

    return selectedSupplier;
  },

  productList: (state, getters) => {
    let productList = getters.allProducts;
    let supplierList = getters.allSuppliers;

    let newProductList = [];

    if (productList.length > 0 && supplierList.length > 0) {
      productList.forEach((product) => {
        let selectedSupplier = supplierList.find(
          (supplier) => supplier.id == product.supplier_id
        );

        newProductList.push(
          Object.assign(
            {
              supplier_name: selectedSupplier.name,
              supplier_location: selectedSupplier.location,
              supplier_platform: selectedSupplier.platform,
            },
            { ...product }
          )
        );
      });
    }

    return newProductList;
  },
};

export default getters;
```

Gambar 4. 33 Implementasi Vuex Getters

c. *Mutations, file* ini bekerja sebagai komponen yang dapat mengubah *state* dan tidak dianjurkan mengubah secara langsung *state* tanpa melewati *mutations.*

```
const mutations = {
  SET_products(state, payload) {
    state.products.productList = _.unionBy(
      state.products.productList,
      [payload],
      "id"
    );
  },
  UPDATE_Customer(state, payload) {
    let customers = state.store.customers.customerList;

    let selectedCustomer = customers.findIndex(
      (customer) => customer.id == payload.customerId
    );

    if (selectedCustomer != -1) {
      const customerData = (customers[selectedCustomer] = _.merge(
        customers[selectedCustomer],
        payload.updatedData
      ));

      customers[selectedCustomer] = customerData;
    }
  },

  SET_Notification(state, payload) {
    state.app.notificationModal = _.merge(state.app.notificationModal, payload);
  },

  SET_User(state, payload) {
    state.user = _.merge(state.user, payload);
  },
};

export default mutations;
```

Gambar 4. 34 Implementasi Vuex Mutations

d. Actions: Merupakan fungsi yang dapat berjalan secara asynchronous, pada *file* ini kita dapat melakukan CRUD dengan menggunakan *firebase*. Pada *file* ini juga *firebase* bekerja dan melakukan *record* data dan kueri sesuai fungsi yang ada.

```
import firebase from "firebase/app";
import {
  storage,
  productsCollection,
  suppliersCollection,
  customersCollection,
  ordersCollection,
  salesCollenction,
  usersCollenction,
} from "../plugins/firebase.js";
import { nanoid } from "nanoid";
const actions = {
  async addProductToSupplier({ commit }, payload) {
    const supplierId = payload.supplierId;
    const productId = payload.productId;

    await suppliersCollection
      .doc(supplierId)
      .update({
        products: firebase.firestore.FieldValue.arrayUnion(productId),
      })
      .then(() => {
        commit("SET_ProductToSupplier", { supplierId, productId });
      })
      .catch((error) => {
        console.error("Error removing document: ", error);
      });
  },
};
export default actions;
```

Gambar 4. 35 Implementasi Vuex Actions

### 4.3.4. Implementasi *Progressive Web App* (PWA)

Untuk memasang PWA ke dalam aplikasi vuejs, kita dapat menggunakan command yang disediakan oleh vuejs. *Command* tersebut men-*generate file* dan *config* untuk PWA ke dalam projek aplikasi vue.

```
vue add pwa
```

Dari hasil command di atas, vuejs akan men-generate file registerServiceWorker.js, berikut *code* yang dihasilkan:

```
import { register } from "register-service-worker";

if (process.env.NODE_ENV === "production") {
  register(`${process.env.BASE_URL}service-worker.js`, {
    ready() {
      console.log(
        "App is being served from cache by a service worker.\n" +
          "For more details, visit https://goo.gl/AFskqB"
      );
    },
    registered() {
      console.log("Service worker has been registered.");
    },
    cached() {
      console.log("Content has been cached for offline use.");
    },
    updatefound() {
      console.log("New content is downloading.");
    },
    updated() {
      console.log("New content is available; please refresh.");
    },
    offline() {
      console.log(
        "No internet connection found. App is running in offline mode."
      );
    },
    error(error) {
      console.error("Error during service worker registration:", error);
    },
  });
}
```

Gambar 4. 36 Implementasi Service Worker pada Vuejs

#### 4.3.4.1. Implementasi *Workbox*

Workbox adalah koleksi library yang dapat membantu kita untuk membuat *service worker*, dengan menggunakan *workbox* kita dapat melakukan *caching* pada aplikasi dengan lebih mudah dan memberikan

fleksibilitas bagaimana *service worker* kita bekerja. Berikut implementasi workbox pada aplikasi manajemen dropship:

1. Instalasi *Workbox*

   Terdapat du acara untuk memasang workbox ke dalam aplikasi vuejs, yaitu dengan menggunakan *full generate service worker* dan *inject manifest* secara manual. Dalam penelitian ini peneliti akan mengimplementasikan keduanya seperti berikut:

   a. Full Generate Service Worker

      Dengan menggunakan metode ini, workbox akan membuat file service worker dan mengurus seluruh strategi cache yang ada. Pertama kita perlu menjalankan command sebagai berikut:

      ```
      npm install workbox-webpack-plugin --save-dev
      ```

      Dengan menggunakan command di atas, ketika build aplikasi maka workbox akan secara otomatis membuat *file service worker*.

   b. Inject Manifest Service Worker

      Metode ini menggunakan *file service-worker* yang kita buat manual, *file service worker* yang sudah dibuat lalu di-*load* melalui konfigurasi *workbox* yang ada pada *file* vue.config.js. Berikut implementasi service-worker menggunakan *inject manifest:*

```
import { registerRoute } from "workbox-routing";
import { CacheFirst, StaleWhileRevalidate } from "workbox-strategies";
import { CacheableResponsePlugin } from "workbox-cacheable-response";
import { ExpirationPlugin } from "workbox-expiration";

registerRoute(
  ({ url }) => url.origin === "https://fonts.googleapis.com",
  new StaleWhileRevalidate({
    cacheName: "google-fonts-stylesheets",
  })
);

registerRoute(
  ({ url }) => url.origin === "https://fonts.gstatic.com",
  new CacheFirst({
    cacheName: "google-fonts-webfonts",
    plugins: [
      new CacheableResponsePlugin({
        statuses: [0, 200],
      }),
      new ExpirationPlugin({
        maxAgeSeconds: 60 * 60 * 24 * 365,
        maxEntries: 30,
      }),
    ],
  })
);
```

Gambar 4. 37 Implementasi Inject Service Worker pada Workbox

2. Konfigurasi *Workbox*

   Konfigurasi *workbox* terdapat di dalam *file vue.config.js*, di sini kita bisa mengkonfirguasi *workbox* sesuai keinginan, berikut keterangan konfigurasi pada *workbox*.

   a. globDirectory: mendefinisikan folder utama service-worker akan dibuat
   b. globPatterns: mendefinisikan ekstensi file yang akan dilakukan *precache*
   c. swDest: mendefinisikan tempat dan nama file *service worker*
   d. workboxPluginMode: mendefinisikan mode workbox, *injectManifest* atau *fullGenerateSW*
   e. swSrc: mendefinisikan tempat file *service worker* yang akan di-*load* ketika menggunakan mode *injectManifest*

### 4.3.5. Implementasi *Manifest File*

*Manifest file* diperlukan untuk mendefinisikan bagaimana prilaku progressive *web app* berjalan, aplikasi PWA bisa di-*install* ketika ada *manifest file* di dalamnya. Berikut implementasi *manifest file*:


Gambar 4. 38 Konfigurasi Manifest File

a. short-name: Mendefinisikan nama aplikasi pada saat di-*install* ke dalam *homescreen* atau *app-drawer*
b. theme_color: Mendefinisikan warna pada aplikasi seperti *notification bar*
c. icons: Icon apa yang digunakan dan tempat disimpannya
d. start_url: Navigasi pada awal aplikasi dijalankan
e. display: Bagaimana aplikasi ditampilkan ketika telah di-*install*, *standalone*, *fullscreen*, dan *minimal-ui*

### 4.3.6. Implementasi Host Aplikasi

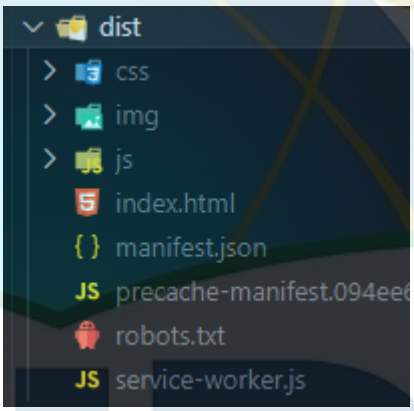
Aplikasi yang sudah dikembangkan merupakan aplikasi berbasis web, agar bisa diakses oleh pengguna diperlukan host pada aplikasi tersebut. Layanan hosting yang dipakai adalah firebase.

#### 4.3.6.1. Build Aplikasi Vuejs

Aplikasi vuejs yang sudah dikembangkan lalu di-*build* menggunakan *command* yang tersedia. Fungsi *build* ini adalah untuk menyatukan seluruh *file* atau modul dan menaruh *file* tersebut pada satu *folder*, dalam aplikasi ini hasil build akan ditaru di folder "dist" pada *root folder* projek. Berikut command untuk melakukan *build* aplikasi vuejs:

```
npm run build
```

Berikut hasil build aplikasi:




Gambar 4. 39 File hasil build aplikasi

#### 4.3.6.2. Intalasi Firebase CLI

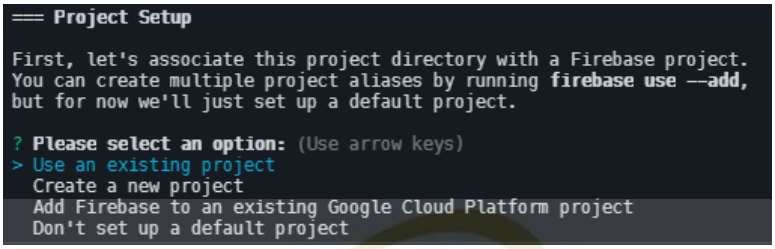
Untuk menggunakan firebase hosting, kita perlu meng-install firebase cli terlebih dahulu. Berikut command untuk meng-install firebase cli:

```
npm install -g firebase-tools
```

#### 4.3.6.3. Deploy Projek

Setelah firebase ter-install kita bisa memanggil command inisiasi firebase pada projek aplikasi manajemen dropship. Inisiasi projek

menggunakan firebase ini berfungsi untuk membuat config, rule Berikut langkah inisiasi:




Gambar 4. 40 Proses inisisasi Firebase

1. Menjalankan command “firebase init hosting”
2. Memilih folder hasil build yang sebelumnya sudah didapat, pada aplikasi ini folder yang dipilih adalah “*dist*”
3. Memilih mode *Single Page Application*

## 4.4. Pengujian Aplikasi

### 4.4.1. *Blackbox Testing*

Untuk memastikan aplikasi dapat berjalan sesuai yang diharapkan, dilakukan uji *blackbox* pada setiap fungsi CRUD yang terdapat di aplikasi manajemen *dropship*.

### 4.4.2. *Lighthouse Testing*

Pengujian menggunakan *lighthouse* dikhususkan untuk mengukur dan menguji secara lebih mendalam. Lighthouse akan memberikan hasil test yang berupa peforma, rating SEO, aksesabilitas dan progressive web app.

# BAB V
# HASIL DAN PEMBAHASAN

## 5.1. Lingkungan Pelaksanaan Pengujian

Aplikasi sebelumnya yang sudah diimplementasikan akan dilakukan pengujian dengan dengan lingkungan pengujian sebagai berikut:

### 5.1.1. Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan dan pengujian aplikasi menggunakan satu buah laptop dengan spesifikasi sebagai berikut:

1. Prosessor Intel Core i7
2. 512 SSD
3. 8 Gb RAM
4. Intel HD 630

### 5.1.2. Perangkat Lunak

1. Sistem Operasi: Windows 11
2. Microsoft Edge Version 93.0.961.38
3. Lighthouse 8.3.0

## 5.2. Hasil pengujian

### 5.2.1. Pengujian *Black Box*

Pengujian black-box dilakukan dengan cara penulis mengakses aplikasi yang sudah diimplementasi sebelumnya. Tujuan dari pengujian ini adalah untuk memastikan bahwa aplikasi yang ada dapat dijalankan dan seluruh fungsi yang ada dapat berjalan sesuai dengan hasil yang diharapkan. Berikut adalah hasil pengujian yang telah dilakuka

Tabel 5. 1 Hasil Uji Blackbox

| No | Kegiatan | Tahap Kegiatan | Hasil yang Diharapkan | Hasil |
|---|---|---|---|---|
| 1. | Register | 1. *User* nenuju halaman "*Login*" dan menekan tulisan "*Register*"<br>2. *User* mengisi formulir dan menekan tombol "*Create Account*" | Muncul notifikasi akun berhasil dibuat | Ya |
| 2. | Login | 1. *User* menuju halaman *login* dan memasukan *password* dan *email* | *User* dapat masuk ke dalam aplikasi dan menuju ke halaman *Home* | Ya |
| 3. | Melihat data profil dan memasukan link toko | 1. *User* menuju halaman "*Profile*"<br>2. Menekan tombol "*Insert Store Link*" dan mengisi data *link* toko | Muncul notifikasi akun berhasil dan *link* toko muncul di halaman "*Profile*" | Ya |
| 4. | Menambah data supplier | 1. *User* menuju halaman "*Store*"<br>2. Memilih menu "*Supplier*"<br>3. Menekan tombol tambah<br>4. Mengisi data *supplier*<br>5. Menekan tombol "*Save*" | Muncul notifikasi sukses dan menuju halaman "*Store*" | Ya |
| 5. | Mengubah data supplier | 1. *User* Menuju halaman "S*tore"*<br>2. memilih menu "*Supplier*"<br>3. Memilih kolom *supplier* yang ingin diubah<br>4. Menekan tulisan *edit*<br>5. Mengubah data yang diinginkan<br>6. Menekan tombol "Save" | Muncul notifikasi sukses data telah diubah dan menuju halaman "*Store*" | Ya |
| 6. | Menghapus data supplier | *1.* *User* Menuju halaman "S*tore"*<br>*2.* Memilih menu "*Supplier*" | Muncul notifikasi sukses data telah dihapus | Ya |

| | | | | |
|---|---|---|---|---|
| | | 3. Menekan dan tahan hingga muncul konfirmasi<br>4. Memilih "*Yes*" | | |
| 7. | Menambah data *customer* | 1. *User* menuju halaman "*Store*"<br>2. Memilih menu "*Customer*"<br>3. Menekan tombol tambah<br>4. Mengisi data *customer*<br>*5.* Menekan tombol "*Save*" | Muncul notifikasi sukses dan menuju halaman "*Store*" | Ya |
| 8. | Mengubah data *customer* | *1.* *User* Menuju halaman "S*tore"*<br>*2.* memilih menu "*Customer*"<br>*3.* Memilih kolom *customer* yang ingin diubah<br>4. Menekan tulisan *edit*<br>*5.* Mengubah data yang diinginkan<br>*6.* Menekan tombol "Save" | Muncul notifikasi sukses data telah diubah dan menuju halaman "*Store*" | Ya |
| 9. | Menghapus data *customer* | *1.* *User* Menuju halaman "S*tore"*<br>*2.* Memilih menu "*Customer*"<br>3. Menekan dan tahan hingga muncul konfirmasi<br>*4.* Memilih "*Yes*" | Muncul notifikasi sukses data telah dihapus | Ya |
| 10. | Menambah data produk | 1. *User* menuju halaman "Produk"<br>2. Menekan tombol tambah<br>3. Mengisi data Produk<br>*4.* Menekan tombol "*Save*" | Muncul notifikasi sukses dan menuju halaman "Produk" | Ya |
| 11. | Mengubah data produk | *1.* *User* Menuju halaman "Produk*"*<br>*2.* Memilih menu "Produk"<br>*3.* Memilih kolom Produk yang ingin diubah<br>4. Menekan tulisan *edit*<br>*5.* Mengubah data yang diinginkan | Muncul notifikasi sukses data telah diubah dan menuju halaman "Produk" | Ya |

| | | | | |
|---|---|---|---|---|
| | | 6. Menekan tombol “Save” | | |
| 12. | Menghapus data produk | 1. *User* Menuju halaman “S*tore”*<br>2. Memilih menu “*Customer*”<br>3. Menekan dan tahan hingga muncul konfirmasi<br>4. Memilih “*Yes*” | Muncul notifikasi sukses data telah dihapus | Ya |
| 13. | Menambah data pesanan | 1. User menuju halaman “Orders”<br>2. Menekan tombol tambah<br>3. Mengisi data customer<br>4. Memilih produk yang ada dengan menekan tombol “Add Product”<br>5. Melakukan checkout | Muncul notifikasi sukses dan menuju halaman “Orders” | Ya |
| 14. | Mengubah status pesanan | 1. User menuju halaman “Orders”<br>2. Memilih kolom pesanan<br>3. Menekan tombol confirm atau mengisi resi pengiriman | Muncul notifikasi sukses dan menuju halaman “Orders” | Ya |
| 15. | Melihat data halaman dashboard | User menuju halaman “Home” | Terdapat data dashboard ketika sudah membuat pesanan | Ya |

### 5.2.2. Pengujian *Lighthouse*

Pengujian menggunakan lighthouse dilakukan pada aplikasi yang berjalan di lokal, *tools* ini dapat diakses pada *chrome devtools*. Tools ini akan menganalisa website yang sudah dikembangkan sebelumnya dan memberikan hasil data *performance*, *Progressive Web App*, *Accesibility*, *Best Practices* dan SEO. Berikut hasil pengujian menggunakan *lighthouse:*

Tabel 5. 2 Hasil Uji Tool Lighthouse

| **Metrik** | | **Hasil** | | |
|---|---|---|---|---|
| | | Tanpa *Cache* | Dengan *Cache* | *Offline* dengan *Cache* |
| ***Peformance*** | | 29/100 | 57/100 | 61/100 |
| 1. | *First Contentful Paint* | 4.7 detik | 2.2 detik | 1.8 detik |
| 2. | *Speed Index* | 6.2 detik | 3.9 detik | 3.6 detik |
| 3. | *Largest Contentful Paint* | 5.4 detik | 2.8 detik | 2.5 detik |
| 4. | *Time to Interactive* | 10.5 detik | 7.2 detik | 6.7 detik |
| 5. | *Total Blocking Time* | 0.3 detik | 0.15 detik | 0.1 detik |
| 6. | *Cumulative Layout Shift* | 0.001 | 0 | 0 |
| ***Accessibility*** | | 83/100 | 83/100 | 83/100 |
| ***Best Practice*** | | 93/100 | 100/100 | 100/100 |
| ***SEO*** | | 92/100 | 92/100 | 92/100 |
| ***Progressive Web App*** | | 8/9 | 8/9 | 8/9 |

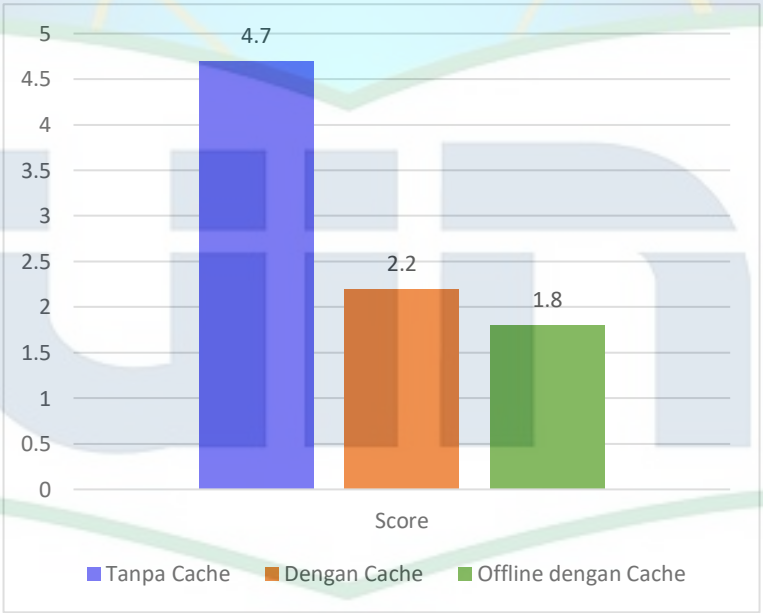
Pada tabel di atas terdapat data yang dihasilkan oleh tools *lighthouse*, data tersebut terdiri atas beberapa metrik. Metrik tersebut direpresentasikan dengan nilai skala dengan nilai 0 - 100 dan 0 - 9 pada metrik Progressive Web App. Skala yang tercatat pada peformance, accessabilty, best practice dan SEO memiliki skala skor mulai dari 0 – 100. Nilai dari skor tersebut memiliki beberapa tingkatan yang direpresentasikan juga menggunakan warna. Skor mulai dari 0 – 49 yang dirtandai dengan warna merah menyatakan bahwa aplikasi memiliki peforma yang masih kurang, skor 50 – 89 ditandai dengan warna kuning mengartikan aplikasi tersebut memiliki peforma yang cukup dan skor 90 hingga 100 ditandai warna hijau yang menyatakan bahwa aplikasi memiliki peforma yang sangat baik. Berikut keterangan lebih detail mengenai metrik yang ada pada hasil uji lighthouse:

#### 5.2.2.1. *Peformance*

Metrik ini akan diukur oleh *lighthouse* untuk melihat peforma kecepatan pada aplikasi ketika diakses. Metrik ini diakumulasikan dari 7 sub-metrik yang bobotnya sudah ditentukan

oleh *tools lighthouse* itu sendiri. Pada hasil pengukuran sebelumnya dengan tiga skenario didapatkan bahwa terdapat kenaikan peforma dengan skor tertinggi diperoleh dengan skenario aplikasi dijalankan secara *offline* dan menggunakan *cache* sebesar 61/100. Hasil tersebut sesuai dengan hasil penelitian sebelumnya yang dilakukan oleh Hanumant Pawar (2020) yang menjelaskan mengenai pengaruh PWA terhadap pengembangan aplikasi web, dalam penelitiannya Pawar menunjukan kenaikan peforma dari semula membutuhkan waktu 1769 milidetik menjadi 85 milidetik. Hal tersebut dikarenakan file yang dibutuhkan oleh aplikasi sudah ter-cache sehingga dapat meminimalisir waktu *load* dan waktu *render* dari aplikasi. Penjelasan lebih detail terkait hasil metrik peformance dapat dilihat pada masing – masing sub-metrik, sebagai berikut:

a. *First Contentful Paint*

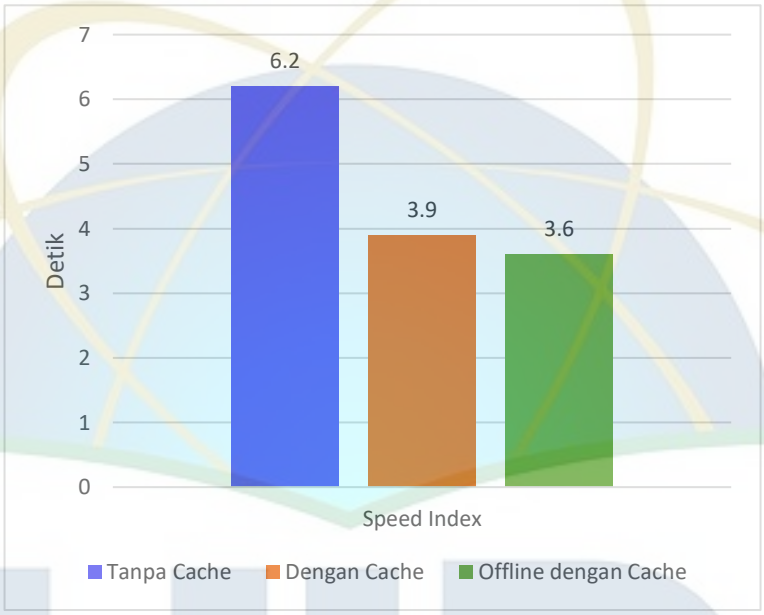



Gambar 5. 1 Hasil Uji First Contentful Paint

Pada gambar di atas terlihat terjadi penurunan waktu *load* dan *render* pada aplikasi yang dikembangkan menggunakan *cache* dan PWA hingga 2x lipat. *First contentful paint* sendiri merupakan seberapa banyak waktu yang diperlukan agar konten

pada DOM (*Docuent Object Model*) dapat di-render dan dilihat oleh pengguna. Sub-metrik ini berbobot 10% dan direkomendasikan memiliki waktu load di bawah 1.8 detik menurut Google.

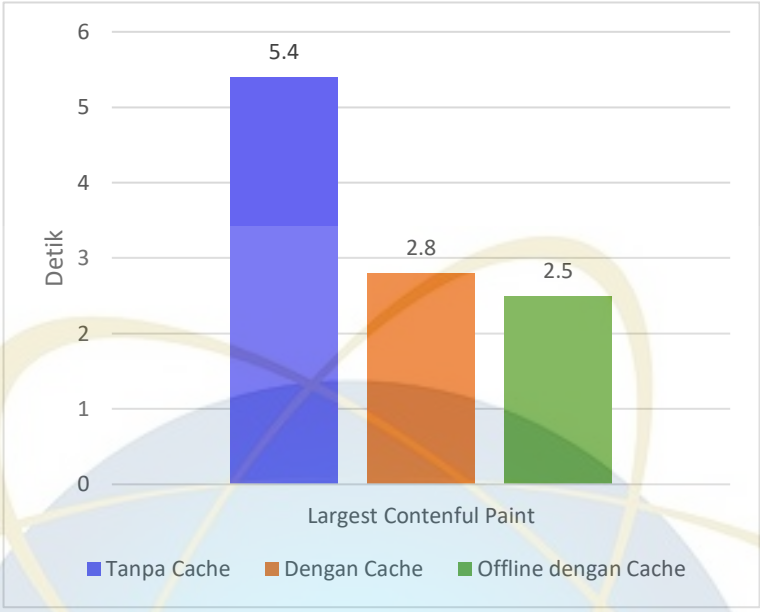
b. *Speed Index*




Gambar 5. 2 Hasil Uji Speed Index

Hasil yang didapat pada sub-metrik ini menjelaskan sebera cepat waktu yang diperlukan agar konten dapat tampil keseluruhan pada layar pengguna. Sub-metrik ini berbobot 10% dan direkomendasikan aplikasi yang diuji mendapatkan hasil kurang dari 4.3 detik pada pengujian metrik *speed index*. Pengujian serupa dilakukan oleh Tahirshah (2019) yang menunjukan terjadi kenaikan kecepatan waktu *load* terhadap konten pada aplikasi web yang menggunakan PWA, hasil uji dengan PWA tersebut menunjukan perbedaan peforma sebesar 24% terhadap uji *speed index*.

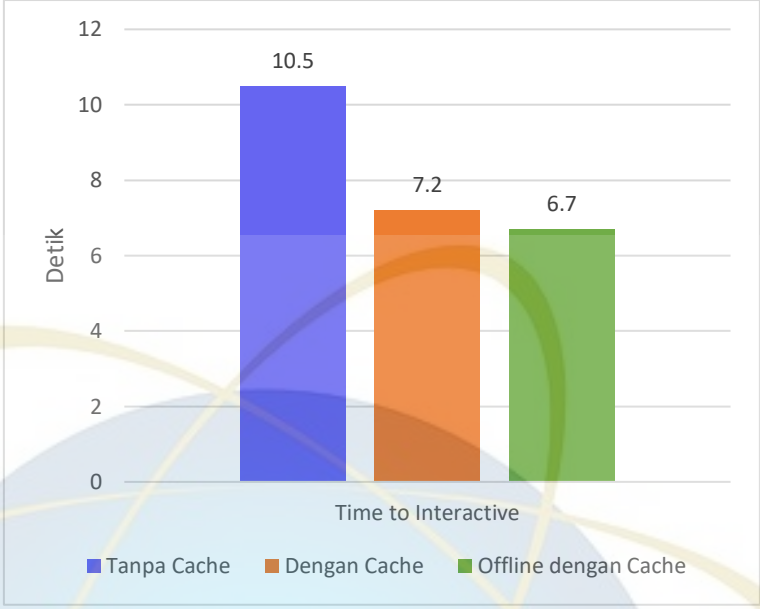
c. Largest Contentful Paint




Gambar 5. 3 Hasil Uji Largest Contentful Paint

*Largest contentful paint* pada metrik uji *peformance* memiliki bobot 25%. Sub-metrik ini mirip dengan sub-metrik *first contenful paint*, bedanya sub-metrik ini mengukur seberapa cepat konten utama dimuat ke dalam layar pengguna. Konten utama yang dimaksud pada sub-metrik ini adalah konten berupa gambar, video dan komponen DOM yang memiliki teks di dalamnya. Sub-metrik ini disarankan memiliki skor load kurang dari 4 detik (Philip Walton, 2020). Dapat dilihat pada gambar grafik di atas dengan menggunakan PWA dan *cache* dapat memberikan peforma yang jauh lebih baik, hal tersebut dikarenakan gambar yang diperlukan oleh aplikasi sudah tersimpan di dalam perangkat sehingga aplikasi tidak perlu melakukan request ulang dang menunggu respon dari *server*.

d. Time to Interactive

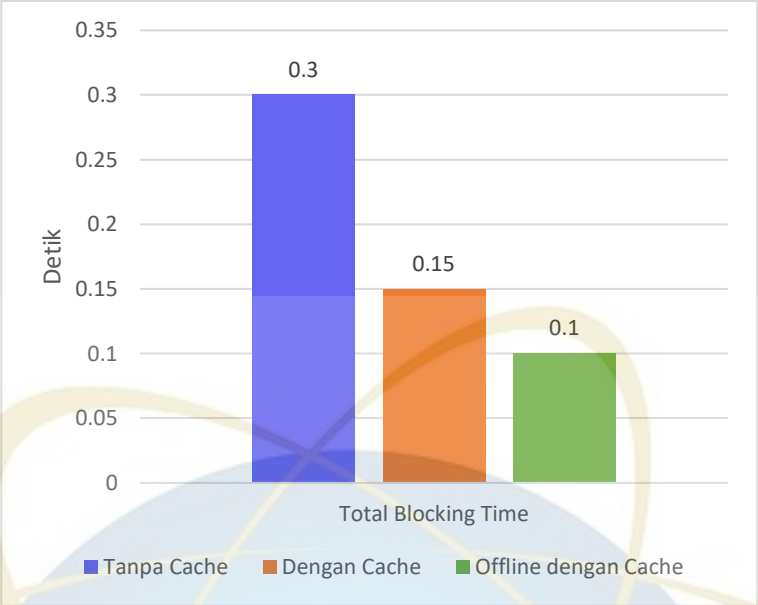



Gambar 5. 4 Hasil Uji Time to Interactive

*Time to interactive* memiliki bobot 10% pada metrik *pefromance*, batas rekomendasi waktu aplikasi untuk dapat berinteraksi secara penuh dalam waktu kurang dari 7.3 detik. Metrik ini dapat diukur dari berapa lama konten awal dapat muncul (*first contentful paint*) dan aplikasi tersebut dapat merespon interaksi dari pengguna.

e. *Total Blocking Time*

Sub-metrik *total blocking time* memiliki bobot paling besar diantara sub-metrik *peformance* lainnya, yaitu dengan bobot 30%. Metrik ini mengukur seberapa lama aplikasi tidak dapat merespon interaksi dari pengguna, sub-metrik ini disarankan agar memiliki skor kurang dari 600 milidetik.




Gambar 5. 5 Hasil Uji Total Blocking Time

#### 5.2.2.2. Best Practices

Setiap pengembangan website memiliki panduan standar yang memberikan pengembang aplikasi website agar terhindar dari beberapa masalah umum (*common mistakes*) yang sering terjadi pada pengembangan aplikasi. Pada *Best Practices*, lighthouse akan melakukan pengecekan atau audit terhadap, berikut keterangan hasil uji *Best Pracices*:

Tabel 5. 3 Hasil Uji Best Practices

| Audit | Terpenuhi |
|---|---|
| Menggunakan HTTPS | ✓ |
| Memperbolehkan untuk menyisipkan (*paste*) teks pada kolom password | ✓ |
| Memunculkan gambar dengan aspek rasio yang benar | ✓ |
| Halaman HTML memiliki *doctype* | ✓ |
| Menggunakan Cache API | ✓ |

| | |
|---|---|
| Mendeteksi *Javascript Library* | ✓ |
| Menampilkan gambar dengan resolusi yang memadai | ✓ |
| Tidak memunculkan pesan *error* pada *console log* | - |
| Menggunakan *javascript source maps* | - |

#### 5.2.2.3. SEO (*Search Engine Optimization*)

Mentrik ini mengukur seberapa baik aplikasi kita dapat ditemukan dan dilihat oleh mesin pencarian, berikut hasil uji SEO:

Tabel 5. 4 Hasil Uji SEO

| Audit | Terpenuhi |
|---|---|
| Memiliki atribut *alt* pada *tag* < *image* > | - |
| Memiliki deskripsi meta pada HTML | ✓ |
| Robo.txt yang benar | ✓ |
| HTML memiliki *tag* <*title*> | ✓ |
| Memiliki ukuran font yang standar dapat dibaca pada perangkat mobile. | - |

#### 5.2.2.4. *Accessability*

Pada sub-metrik ini mengukur seberapa baik aplikasi tersebut dapat diakses oleh pengguna disabilitas, berikut hasil uji:

Tabel 5. 5 Hasil Uji Accessability

| Audit | Terpenuhi |
|---|---|
| HTML memiliki atribut *lang* | ✓ |
| Memiliki warna *background* dan *fourground* yang kontras | ✓ |
| Memiliki atribut *alt* pada *tag* <*image*> | - |

| | |
|---|---|
| Aksesibilitas yang baik pada penamaan tombol | ✓ |

#### 5.2.2.5. PWA (*Progressive Web App*)

Metrik PWA pada lighthouse melakukan pengecekan audit bahwa aplikasi website telah memenuhi aspek dari Progressive Web App, berikut hasil uji PWA pada lighthouse:

Tabel 5. 6 Hasil Uji PWA

| Audit | Terpenuhi |
|---|---|
| Dapat di-install | ✓ |
| Memiliki service-worker dan start_url | ✓ |
| Menggunakan HTTPS | ✓ |
| Menggunakan data *splash color* pada *manifest file* | - |
| Menggunakan theme color pada address bar | ✓ |
| Dapat dijalankan secara *offline* | ✓ |
| Menyediakan *icon* pada aplikasi | ✓ |
| Dapat diakses di berbagai *browser* dan perangkat | ✓ |

### 5.2.3. Hasil Pengujian *Network*

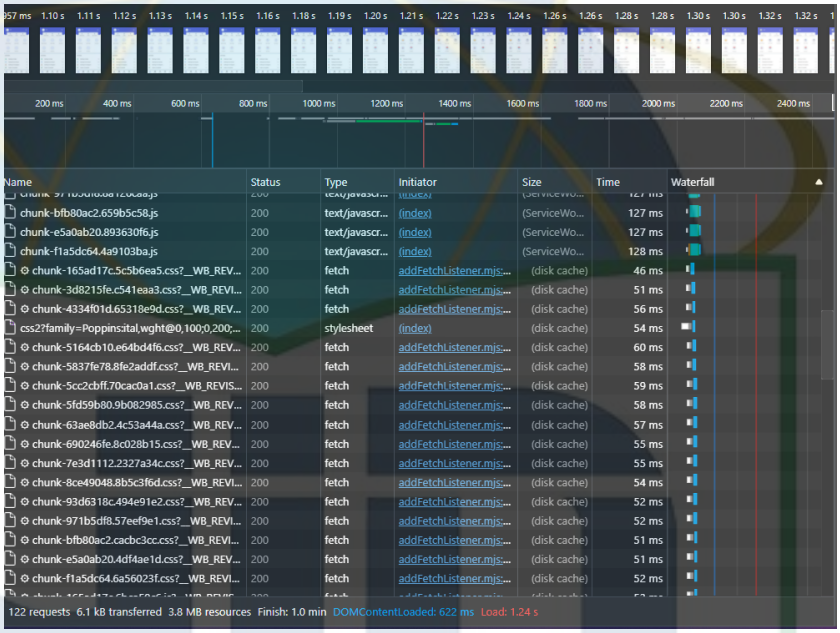
Pada sub-bab ini dilakukan uji *network* melalui *tools* yang tersedia di dalam *browser Microsoft Edge*, pengujian ini dilakukan untuk mengetahui seberapa cepat dan banyak data yang diunduh oleh pengguna. Pengujian dilakukan dengan menggunakan beberapa kondisi skenario pengujian, atara lain:

Tabel 5. 7 Skenario Pengujian Network

| No. | Nama | Kunjungan Halaman | Service Worker | Tersedia Cache | Koneksi |
|---|---|---|---|---|---|
| 1. | Skenario Pertama | Pertama Kali | Tidak Ada | Tidak Tersedia | Aktif |
| 2. | Skenario Kedua | Setelah Pertama Kali | Aktif | Tersedia | Aktif |
| 3. | Skenario Ketiga | Setelah Pertama Kali | Aktif | Tidak Tersedia | Aktif |
| 4. | Skenario Keempat | Setelah Pertama Kali | Aktif | Tersedia | Aktif 3G |
| 5. | Skenario Kelima | Setelah Pertama Kali | Aktif | Tersedia | *Offline* |

### 5.2.3.6. Skenario Pertama

Pada skenario pertama, aplikasi dibuka untuk pertama kali pada saat service worker dan cache belum tersedia.

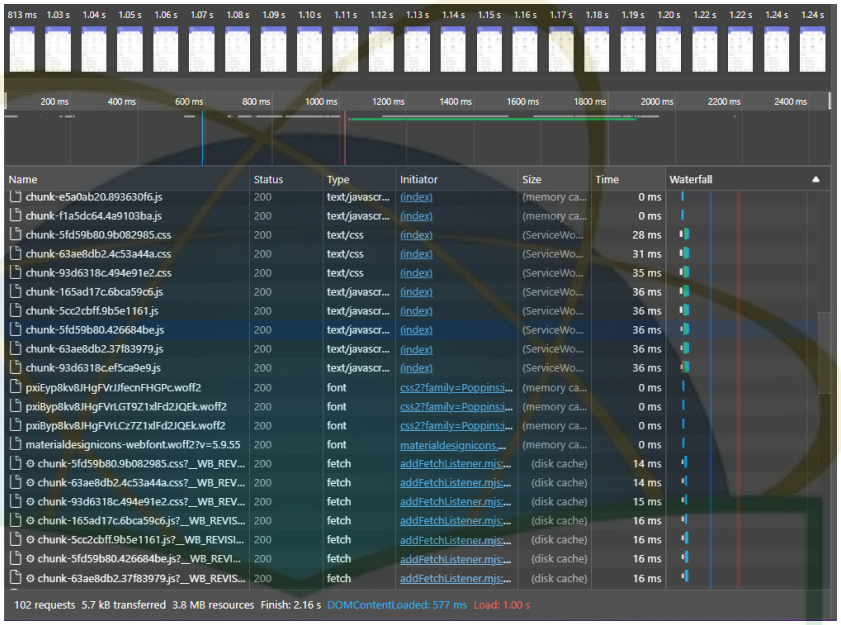



Gambar 5. 6 Hasil Uji Network Skenario Pertama

Dapat dilihat pada gambar di atas, pada saat pertama kali aplikasi dijalankan dalam keadaan cache dan service network belum tersedia aplikasi melakukan request sebanyak 122 request dengan mengirimkan data sebanyak 6.1kb dan data yang diterima oleh client sebesar 3.8Mb. Akses aplikasi dapat diselesaikan dalam waktu 1.24 detik dan proses render selama 622 milidetik.

### 5.2.3.7. Skenario Kedua

Skenario kedua dilakukan dalam kondisi aplikasi sudah dijalankan sebelumnya. Aplikasi tersebut sudah terdapat service worker dan cache yang berjalan, dengan adanya cache dan service worker diharapkan aplikasi dapat dijalankan lebih baik. Berikut hasil tes pada skenario kedua.

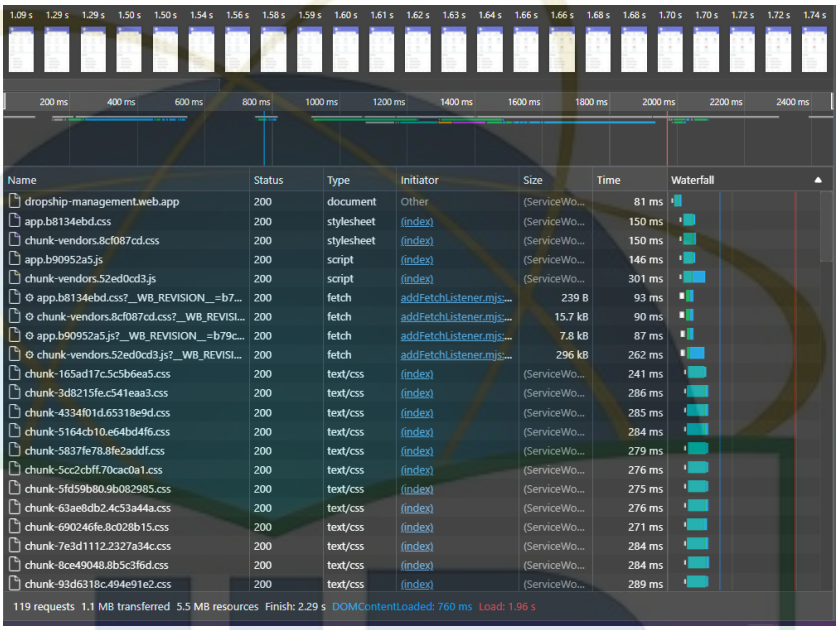



Gambar 5. 7 Hasil Uji Network Skenario Kedua

Pada skenario kedua terlihat pada gambar bahwa peforma yang didapat jauh lebih baik dari sebelum menggunakan service worker dan cache. Dapat dilihat pada gambar di atas, aplikasi hanya melakukan request sebanyak 102 request, tetapi terjadi kenaikan dalam data yang ditransfer yaitu sebesar 5.7kb. selain itu aplikasi dapat selsai di-load dalam waktu 1 detik, lebih cepat 0.24 detik dari awal aplikasi dijalankan. Waktu render juga lebih cepat sekitar 577 milidetik. Pada skenario kedua ini aset – aset yang diperlukan oleh aplikasi telah tersimpan di dalam aplikasi.

### 5.2.3.8. Skenario ketiga

Skenario ketiga dilakukan untuk mengetahui perbedaan performa pada saat tidak menggunakan *cache* tetapi *service worker* masih berjalan. Agar aplikasi tidak menggunakan *cache*, ada beberapa cara yang bisa dilakukan yaitu dengan men-*disable cache* pada *devtools* atau bisa melakukan penghapusan *cache* aplikasi. Berikut hasil uji skenario ketiga:

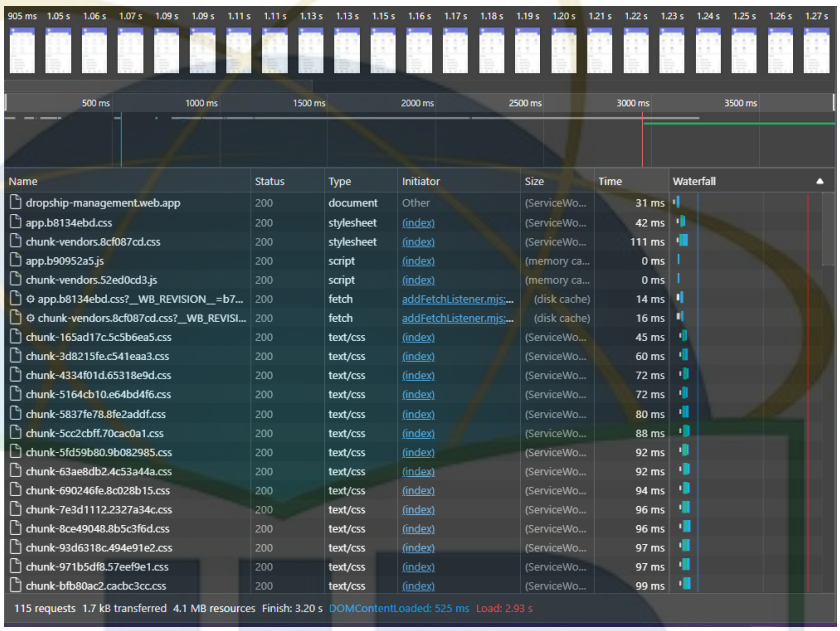



Gambar 5. 8 Hasil Uji Network Skenario Ketiga

Pada gambar di atas dapat dilihat bahwa aplikasi mencoba mengirimkan request sebesar pada skenario pertama, pada hasil skenario ketiga ini aplikasi melakukan request sebesar 119 request dan mengirim data sebesar 1.1Mb. Data aplikasi mengalami kenaikan dari 3.8Mb menjadi 5.5Mb, selain itu aplikasi membutuhkan waktu load sebesar 1.96 detik dan waktu render mencapai 760 milidetik.

### 5.2.3.9. Skenario Keempat

Pada skenario keempat aplikasi akan diakses dalam kondisi seperti pada skenario kedua, tetapi ada perbedaan pada kemampuan jaringan untuk mengakses aplikasi. Jaringan yang akan digunakan adalah pada jaringan 3G, hal ini dilakukan untuk melihat seberapa cepat aplikasi dapat diakses dalam kondisi jaringan yang tidak stabil atau buruk. Berikut hasil uji skenario keempat:

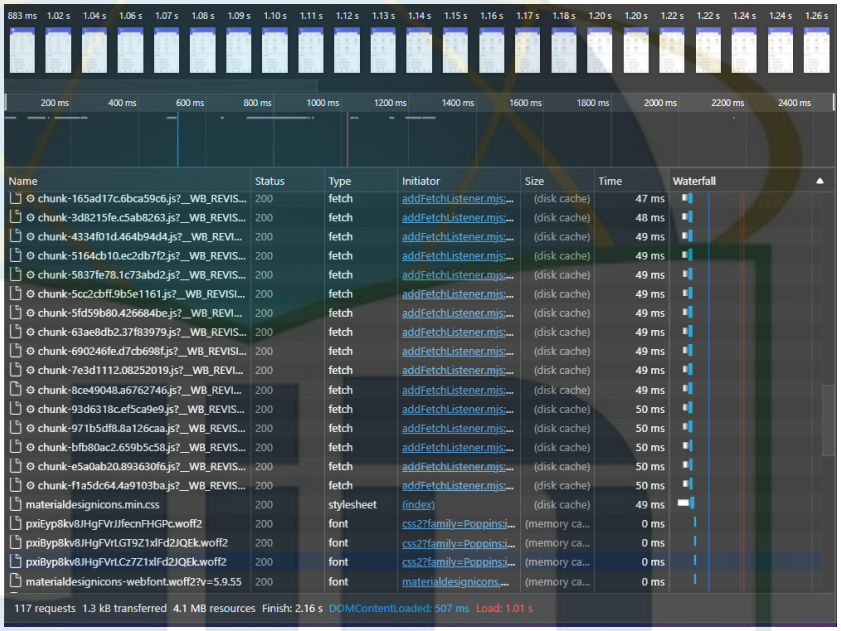



Gambar 5. 9 Hasil Uji Network Skenario Keempat

Terlihat pada gambar di atas, hasil uji skenario keempat pada jaringan lambat aplikasi tetap melakukan request sebesar 115 request dan mengirimkan data sebesar 1.7kb. Aplikasi dapat direnderdalam waktu 525 milidetik dan membutuhkan waktu *load* selama 2.93 detik. Jika dibandingkan dengan skenario ketiga, waktu render terbilang lebih cepat dikarenakan aset seperti gambar dan file static diambil dari *cache* sehingga jaringan tidak berpengaruh pada kecepatan aplikasi untuk diakses. Tetapi pada jaringan yang lambat, waktu *load* membutuhkan waktu lebih lama.

### 5.2.3.10. Skenario Kelima

Pada skenario ini dilakukan untuk melihat seberapa baik aplikasi dapat diakses pada kondisi tidak menggunakan jaringan hal ini juga dapat membuktikan apakah aplikasi web dengan menggunakan *Progressive Web App* dapat dijalankan tanpa menggunakan jaringan. Untuk mematikan jaringan kita bisa memutuskan jaringan terhadap perangkat atau menggunakan opsi *offline* pada *devtools*. Berikut hasil uji skenario kelima dengan mengakses aplikasi pada kondisi tidak terhubung ke internet.



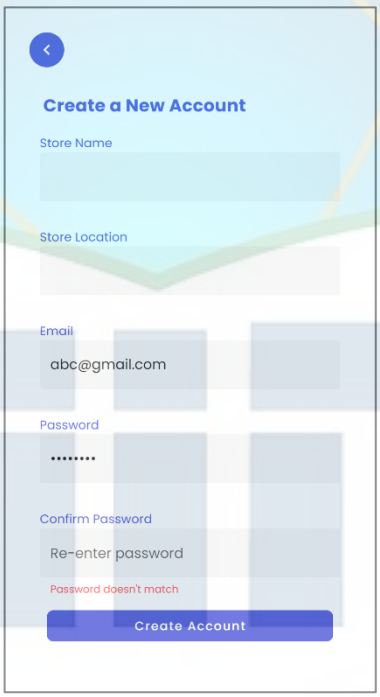

Gambar 5. 10 Hasil Uji Network Skenario Kelima

Pada gambar di atas dapat terlihat aplikasi masih bisa diakses pada kondisi tanpa jaringan, terdapat 117 *request* dan 1.3kb data yang aplikasi coba kirimkan. Karena aplikasi tidak terhubung ke jaringan internet otomatis request tersebut otomatis akan gagal, namun di sini *service worker* akan menangani request tersebut dan mengambil data dari cache. Seperti yang terlihat pada hasil uji

skenario kelima aplikasi dapat me-render dan melakukan load lebih cepat dari skenario sebelumnya, hal tersebut disebabkan aplikasi tidak menunggu respon dari *server* melaikan *service worker* langsung mengalihkan request dan mengembalikan respon dari penyimpanan lokal. Aplikasi berbasis web dengan menggunakan *Progressive Web App* dan cache dapat memberikan kenaikan peforma akses dan aplikasi tidak lagi memerlukan jaringan untuk bisa dijalankan.

## 5.3. Hasil Implementasi Tampilan Aplikasi

### 5.3.1. Tampilan Halaman *Register*

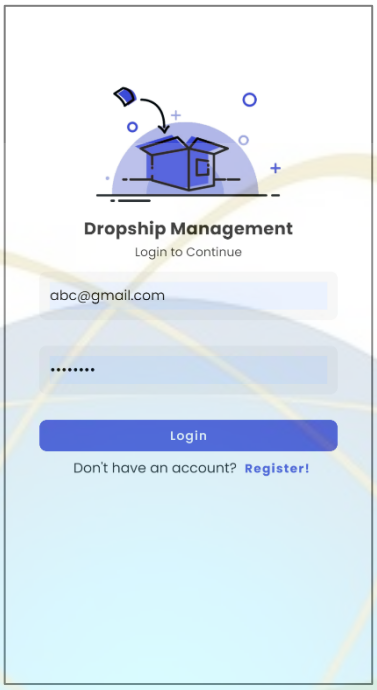



Gambar 5. 11 Tampilan Halaman Register

Terlihat pada gambar 5.11 menampilkan halaman Register, pada halaman ini pengguna baru dapat mendaftarkan akunnya sebelum untuk mendapatkan akses ke dalam aplikasi. Pengguna baru perlu mengisi beberapa formulir data seperti nama toko, lokasi toko, email dan password.

Setelah semua kolom terisi lalu pengguna perlu mengirimkan data tersebut dengan menekan tombol "*Create Account*"

### 5.3.2. Tampilan Halaman *Login*



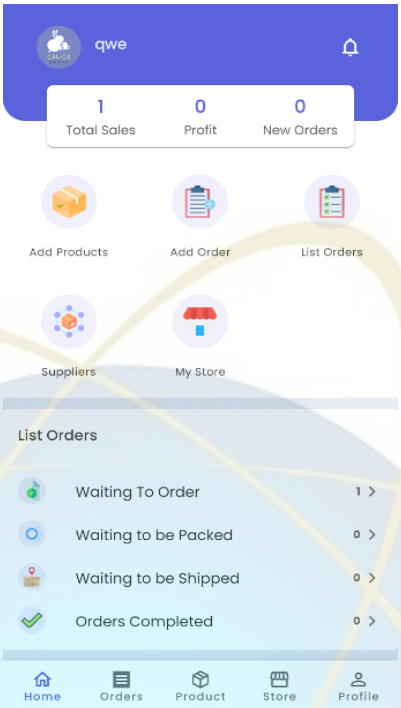

Gambar 5. 12 Tampilan Halaman Login

Setelah memiliki akun atau baru saja melakukan pendaftaran akun, maka akan diarahkan ke halaman "Login" untuk melakukan verifikasi sebelum masuk ke aplikasi utama. Halaman ini akan muncul kembali jika pengguna melakukan *logout* dari aplikasi. Setelah mengisi formulir data akun, pengguna perlu menekan tombol "Login", jika akun berhasil terverifikasi maka akan diarahkan ke halaman "Home" dan jika akun tidak dapat terverifikasi maka akan muncul keterangan *error*.

### 5.3.3. Tampilan Halaman *Home*

Setelah melakukan login seperti pada gambar 5.12, selanjutnya aplikasi akan menampilkan halaman "Home", halaman ini akan muncul tiap pengguna mengakses membuka aplikasi selama data akun masih tersimpan

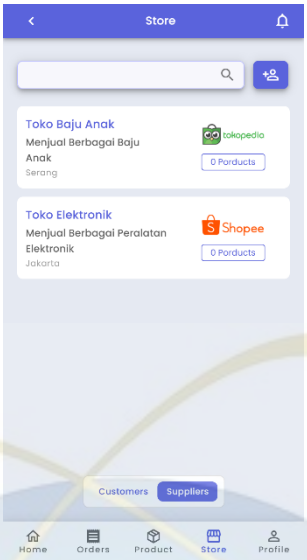
di perangkat. Di halaman ini kita bisa melihat atau mengakses secara cepat menu yang ada pada aplikasi.




Gambar 5. 13 Tampilan Halaman Home

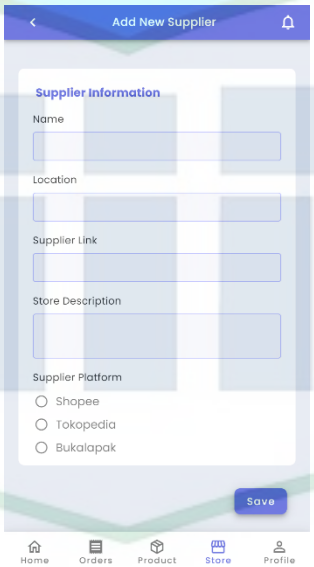
### 5.3.4. Tampilan Halaman *Supplier*

Sesuai dengan alur desain yang telah dibahas pada bab sebelumnya, direkomendasikan untuk pengguna untuk membuat data supplier terlebih dahulu. Untuk mengakses halaman "Supplier", terdapat menu navigasi yang tersedia di bawah aplikasi, pengguna bisa memilih menu bertulisan "Store" atau dapat mengakses secara cepat lewat menu "Supplier" pada menu *icon* di bawah data *dashboard*. Setelah memasuki halaman "Store", pilih menu "Supplier" yang terdapat di bawah, terdapat dua pilihan yaitu "Customer" dan "Supplier".




Gambar 5. 14 Tampilan Halaman Store

Seperti pada gambar 5.14 pengguna dapat melihat daftar supplier yang sudah didata ke dalam aplikasi. Untuk menambahkan data *supplier* baru, kita bisa menekan tombol dengan gambar orang pada samping kolom

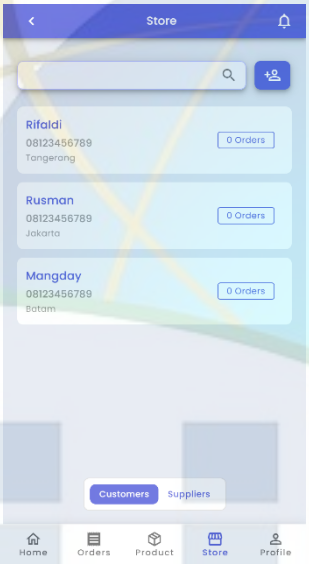



Gambar 5. 15 Tampilan Halaman Supplier

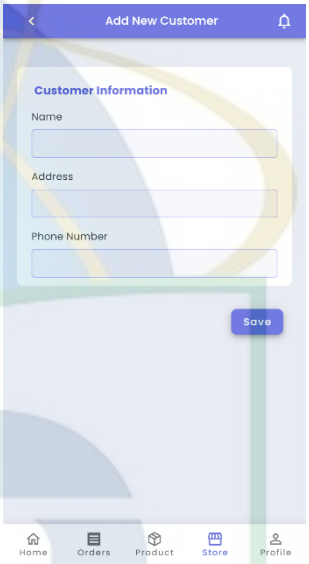
pencarian *supplier*. Pada gambar 5.15 dapat dilihat terdapat beberapa kolom yang perlu diisi oleh pengguna untuk menambahkan data supplier. Data supplier tersebut diperlukan sebelum membuat produk dan pesanan .

### 5.3.5. Tampilan Halaman *Customer*

Halaman "*Customer*" akan menampilan daftar customer yang telah melakukan transaksi pesanan ke toko pengguna. Selain menampilkan daftar nama *customer*, terdapat data berapa banyak pesanan yang telah customer lakukan sebelumnya. Selanjutnya untuk menambahkan customer baru pengguna bisa menekan tombol dengan *icon* orang di samping kolom pencarian.

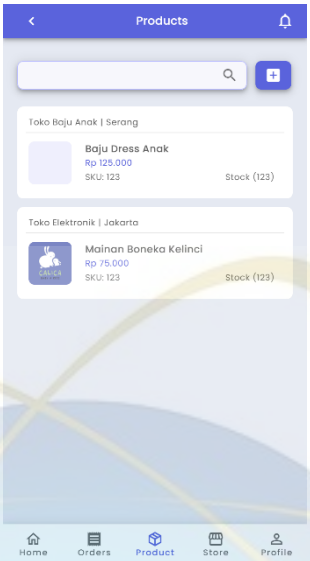



Gambar 5. 16 Tampilan Customer



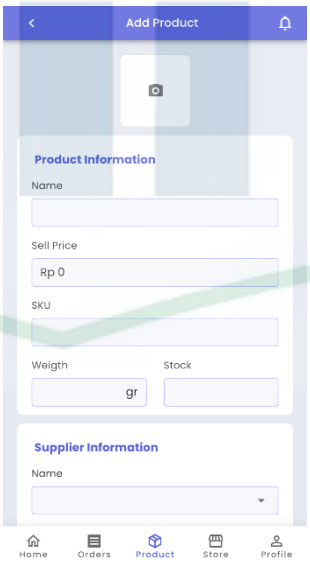

Gambar 5. 17 Tampilan Halaman Add Customer

### 5.3.6. Tampilan Halaman *Product*



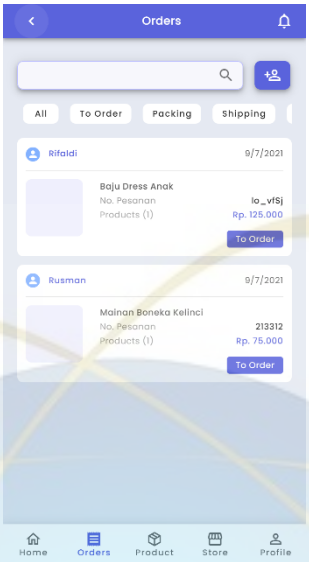

Gambar 5. 18 Tampilan Halaman Products

Seperti pada halaman Customer dan Supplier, pada halaman "Product" pengguna dapat melihat daftar produk yang dipasarkan oleh pengguna (*dropshipper*). Pada gambar 5.19 terlihat kolom produk menampilkan informasi singkat mengenai produk tersebut dan terdapat foto produk (jika ada). Untuk menambahkan produk baru dapat dilakukan dengan menekan tombol tambah pada samping kolom pencarian.



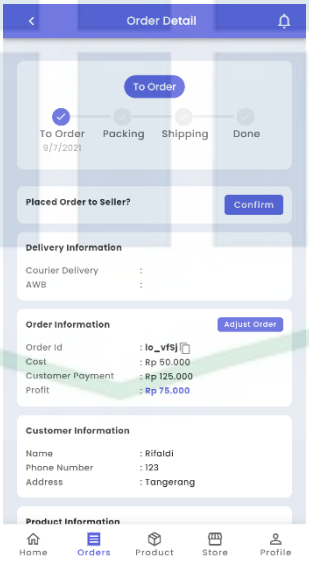

Gambar 5. 19 Tampilan Halaman Add Product

### 5.3.7. Tampilan Halaman *Orders*



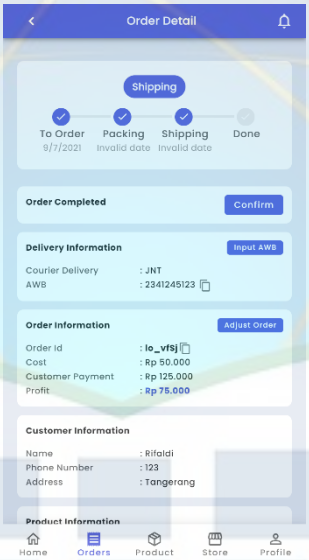

Gambar 5. 20 Tampilan Halaman Orders

Pada tampilan halaman "*Orders*" pengguna bisa melihat daftar pesanan yang masuk dan selesai. Terdapat beberapa status yang ada pada tiap pesanan. Daftar pesanan akan menampilkan informasi terkait pesanan seperti *customer*, total harga pesanan dan tanggal pesanan tersebut dibuat. Untuk informasi lebih lanjut dapat dilihat pada halaman "*Order Detail*".




Gambar 5. 21 Tampilan Halaman Order Detail

Pada gambar 5.21 dapat dilihat secara lengkap informasi terkait pesanan, terdapat status pesanan, data pengiriman, data *customer*, data *supplier* dan produk yang ada dalam pesanan. Untuk mengubah status pesanan, pengguna bisa melakukan konfirmasi pada tombol "Confirm". Tombol "Confirm" akan meminta pengguna untuk mengkonfirmasi bahwa pesanan sudah melakukan proses pemesanan ataupun pengiriman sesuai yang tercantum di aplikasi. Setelah pesanan sudah dirproses dan dikirim oleh *supplier*, pengguna (dropshipper) perlu memasukan resi yang sesuai dengan resi yang diberikan oleh *supplier* atau pihak ekspedisi.

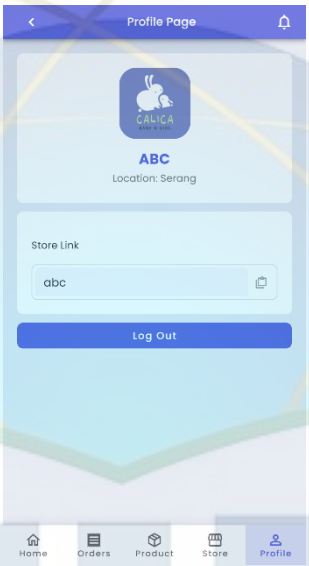



Gambar 5. 22 Perubahan Status pada Halaman Order Detail

Pesanan yang telah dimasukan resi, secara otomatis akan merubah status pesanan menjadi "Shipping". Setelah pesanan selesai atau telah sampai pada pembeli maka pengguna dapat mengkonfirmasi bahwa pesanan sudah selesai dengan menekan tombol "Confirm" , dengan demikian pesanan akan dianggap selesai dan data pesanan selesai akan secara otomatis terdata ke data *dasboard*.

### 5.3.8. Tampilan Halaman *Profile*

Pada halaman ini pengguna dapat melihat informasi mengenai akun pengguna. Data tersebut antara lain foto profil, nama toko, lokasi tokom, dan link toko. Pada awal pembuatan akun, foto profil dan link toko tidak tersedia sehingga perlu diisi terlebih dahulu. Selain itu pada halaman ini pengguna dapat mengeluarkan akunnya dari aplikasi manajemen dropship. Setelah keluar maka aplikasi akan menghapus data yang tersimpan di dalam perangkat.




Gambar 5. 23 Tampilan Halaman Profile

# BAB VI
# PENUTUP

## 5.4. Kesimpulan

Dari hasil penelitian yang dilakukan dan dijelaskan pada bab sebelumnya, penulis dapat memberikan kesimpulan terkait penelitian implementasi Progressive Web App pada pembuatan aplikasi manajemen data dropship sebagai berikut:

1. *Progressive web app* dapat digunakan sebagai alternatif pembuatan aplikasi mobile. Penggunaan PWA pada aplikasi web dapat memberikan kenaikan peforma dan memberikan kemampuan terhadap aplikasi agar bisa diakses pada kondisi tanpa jaringan internet.
2. *Dropshipper* dapat terbantu dengan adanya aplikasi manajemen data ini, sehingga data seperti pesanan, produk, customer dan supplier dapat disimpan dengan rapih menggunakan aplikasi. Selain itu dropshipper juga bisa mengetahui progress dari tiap pesanan.
3. Pengunaan PWA dan vuejs memberikan tampilan menyerupai tampilan aplikasi *mobile*.

## 5.5. Saran

Penelitian ini masih memiliki banyak kekurangan dan perlu dikembankan dari sisi penulisan dan pembuatan aplikasi. Berdasarkan penelitian yang dilakukan, terdapat beberapa saran yang dapat dilakukan untuk penelitian sejenis selanjutnya:

1. Progressive Web App memiliki banyak fitur yang dapat di-implementasikan, diharapkan penelitian sejenis dapat membahas seluruh fitur yang ada pada Progressive Web App.
2. Aplikasi manajemen *dropship* yang dikembangkan masih terbatas pada satu aktor tunggal dan masih kurang efektif untuk digunakan secara komersil, perlu integrasi terhadap toko *online* dan penyedia layanan pengiriman.

3. Aplikasi *web* yang menggunakan *Progressive Web App* dapat dijadikan APK, diharapkan pada penelitian selanjutnya aplikasi berbasis PWA tersebut dapat dihadirkan ke dalam Google Play Store.

# DAFTAR PUSTAKA


Addy Osmani. (2020). *Getting Started with Progressive Web Apps* . https://developers.google.com/web/updates/2015/12/getting-started-pwa

Adi, L. (2017). *Platform e-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps*.

Agency, B. (2012). *Dropshipping Cara Mudah Bisnis Online*. PT Elex Media Komputindo.

Anthony gore. (2017). *WTF is Vuex? A Beginner's Guide To Vuex 4*. https://vuejsdevelopers.com/2017/05/15/vue-js-what-is-vuex/

Ashwin Sathian. (2020). *Using Google Lighthouse to audit your web application*. https://flexiple.com/developers/using-google-lighthouse-to-audit-your-web-application/

Aswati, S., & Siagian, Y. (2016). Model Rapid Application Development Dalam Rancang Bangun Sistem Informasi Pemasaran Rumah (Studi Kasus : Perum Perumnas Cabang Medan. *Sesindo*, 317–324.

Aulia, I., & Oktavianus, R. (2018). Perancangan Aplikasi Dropshipping Produk Smartphone Berbasis Web. *Journal ENTER*, *1*, 517–526.

Basren, B. (2018). *Progressive Web Apps ( PWA ) for YII Framework Enrichment*. *6*(3), 193–200.

Behl, K. (2018). Architectural Pattern of Progressive Web and Background Synchronization. *International Conference on Advances in Computing and Communication Engineering*.

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T. M. (2017). Progressive web apps: The possibleweb-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

Chatterjee, N., Chakraborty, S., Decosta, A., & Nath, A. (2018). Real-time Communication Application Based on Android Using Google Firebase. *Ddfsdfdsfsd*, *6*(4), 74–79. www.ijarcsms.com

Dharwiyanti, S., & Wahono, R. S. (2003). *Pengantar Unified Modeling LAnguage (UML)*. 1–13. http://www.unej.ac.id/pdf/yanti-uml.pdf

Fredrikson, R. (2018). Emulating a Native Mobile Experience with Cross-platform Applications. In *DEGREE PROJECT COMPUTER SCIENCE AND ENGINEERING*.

Hadi, R. (2018). Analisis praktek JualBeli Dropshipping Dalam Perpektif Ekonomi Islam. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699. https://doi.org/10.1017/CBO9781107415324.004

Hajian, M. (2019). Progressive Web Apps with Angular. In *Progressive Web Apps with Angular*. https://doi.org/10.1007/978-1-4842-4448-7

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

Jadhav, M. A., Sawant, B. R., & Deshmukh, A. (2015). *Single Page Application using AngularJS*. http://mydomain.com/myseo#key=value

Jemel, M., & Serhrouchni, A. (2014). *Security Enhancement of HTML5 Local Data Storage*. http://www.w3.org/TR/IndexedDB/

John Leider. (2021). *Why you should be using Vuetify*. https://vuetifyjs.com/en/introduction/why-vuetify/

Joshua Bemenderfer. (2020). *Understanding Vue.js Lifecycle Hooks*. https://www.digitalocean.com/community/tutorials/vuejs-component-lifecycle

Karim, K. (2016). *Progressive Web Application Migrating Web Application to a Progressive Web Application*. https://play.google.com/store/apps/details?

Khedkar, S., Thube, S., Estate, W. I., W, N. M., & Naka, C. (2017). Real Time Databases for Applications. *International Research Journal of Engineering and Technology(IRJET)*, *4*(6). www.irjet.net

Komang Nova Artawan, & Ketut Gede Suhartana. (2019). View of Pengembangan Aplikasi Back-End SIM-MITRA (Sistem Informasi Manajemen Mitra Statistik) BPS Kota Denpasar. *Jurnal Elektronik Ilmu Komputer Udayana*. https://ojs.unud.ac.id/index.php/JLK/article/view/45220/29433

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.

Martina Seidl, & Marion Scholz. (2015). UML @ Classroom : An Introduction to Object-Oriented Modeling. In *Springer International Publishing Switzerland*. https://doi.org/10.1007/978-3-319-12742-2_

Meizhen, W., Yanlei, S., & Yue, T. (2013). *The Design and Implementation of LRU-Based Web Cache*.

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

*Native, Web or Hybrid Apps? What's The Difference?* (n.d.). Retrieved June 16, 2020, from https://www.mobiloud.com/blog/native-web-or-hybrid-apps/

Nauval, A. F. (2018). *Sistem Dropshipping dalam Online Shop Menurut Hukum Islam dan Undang - Undang Informasi dan Transaksi Elektronik Nomor 19 Tahun 2016*. UIN Syarif Hidayatullah Jakarta.

Noor, A. E., & Irfan, P. (2020). Implementasi Progressive Web Apps (PWA) Menggunakan Laravel Dan Vue.Js dalam Pembuatan Aplikasi Penyedia Jasa Freelance. *JTIM : Jurnal Teknologi Informasi Dan Multimedia*, *2*(3), 174–180. https://doi.org/10.35746/jtim.v2i3.109

Philip Walton. (2020). *Largest Contentful Paint (LCP)*. https://web.dev/lcp/

Ratna Patria. (2020). *Kelebihan dan Kekurangan Memulai Bisnis Dropship*. Domainesia.Com. https://www.alona.co.id/bisnis/kelebihan-dan-kekurangan-bisnis-dropship/

Rojas, C. (2019). *Building Progressive Web Applications with Vue.js: Reliable, Fast, and Engaging Apps with Vue.js*. https://books.google.co.uk/books?id=MUXEDwAAQBAJ

Sanjaya, R. (2020). Website Pengelolaan Dropshipper pada Toko Hikmah Putra Elektronika. *Techno.Com: Jurnal Teknologi Informasi*, *18*(1), 1–5.

Sheppard, D. (2017). *Beginning Progressive Web App Development Creating a Native App Experience on the Web*. https://doi.org/10.1007/978-1-4842-3090-9

Syafii, A. (2013). *Step By Step Bisnis Dropshipping dan Reseller*. Elex Media Komputindo. https://doi.org/10.1017/CBO9781107415324.004

Tahirshah, F. S. (2019). *Comparison between Progressive Web App and Regular Web App*. *june*.

Telaumbanua Achmad Ghazali. (2018). *PENGEMBANGAN APLIKASI PEMESANAN LAYANAN KECANTIKAN BERBASIS PROGRESSIVE WEB APPS (PWA)*.

Truica, C.-O., Boicea, A., & Trifan, I. (2013). *CRUD Operations in MongoDB*. *Icacsei*, 347–350. https://doi.org/10.2991/icacsei.2013.88

Wahyuningrum, T., & Januarita, D. (2014). *Perancangan Web e-Commerce dengan Metode Rapid Application Development ( RAD ) untuk Produk Unggulan*

*Desa*. *2014*(November), 81–88.

Waworuntu, A. (2020). Rancang Bangun Aplikasi e-Commerce Dropship Berbasis Web. *Ultimatics : Jurnal Teknik Informatika*, *12*(2), 118–124. https://doi.org/10.31937/ti.v12i2.1823

# LAMPIRAN

## Lampiran 1. Surat Keterangan Bimbingan Skripsi

**Lampiran 2. Transkrip Wawancara**

Hari / Tanggal : 25 Mei 2021

Narasumber : Ririn Rianti

**Penulis: Sudah berapa lama menjalani bisnis online?**

Narasumber: Toko online yang saya kelola kurang lebih sudah berjalan hampir satu tahun

**Penulis: Apakah pada awal menjalani bisnis online langsung menerapkan model dropship?**

Narasumber: Pada awal memulai bisnis online memang masih mengandalkan customer sekitar seperti teman kuliah dan saudara. Produknya ditawarkan lewat sosial media lalu dengan sistem PO, lalu mencoba membuka toko online dengan sistem dropship hingga sekarang

**Penulis: Mengapa memilih menjadi dropshipper?**

Narasumber: Karena keterbatasan modal untuk stok barang banyak dan belum bisa membuat produk sendiri.

**Penulis: Berapa banyak toko online yang tangani?**

Narasumber: Hingga sekarang baru dua toko online dan satu akun sosmed yang dikelola

**Penulis: Bagaimana efek pandemi Covid-19 terhadap toko yang ditangani?**

Narasumber: Untuk beberapa niche ada yang naik seperti produk kesehatan dan kecantikan mengalami kenaikan, tetapi untuk niche seperti fashion memang mengalami penurunan

**Penulis: Produk - produk kategori seperti apa yang dijual?**

Narasumber: Dalam satu toko bisa beberapa kategori atau niche tetapi sekarang masih fokus pada kategori kesehatan dan kecantikan

**Penulis: Bagaimana alur sistem pemesanan yang dilakukan sekarang?**

Narasumber: Untuk alurnya sendiri mirip seperti berjualan biasa, kita memasarkan produk dan mencari customer. Kalau ada customer yang membeli baru kita pesankan ke supplier lewat toko online, sebelumnya kita izin dulu untuk men-dropship kan produk dari supplier. Untuk pemesanan ke supplier bisa lewat ecommerce atau kita chat via whatsapp untuk memakai resi dari kita.

**Penulis: Kendala apa yang dialami saat menjalankan bisnis online dengan dropship?**

Narasumber: Untuk kendala kebanyakan pusing di bagian pendataan pesanan dan customer. Karena dilakukan secara manual dan sendiri kadang lupa pesanan mana saja yang sudah diproses dan belum, produk mana aja yang udah dikirim. Selain itu juga cukup ribet kalau harus ngecek data pesanan atau customer beda ecommerce harus. Kendala lainnya juga kadang lupa produk yang dicantumkan ambil dari toko mana jadi harus nyari ulang atau liat histori belanja di ecommerce itu sendiri.

**Penulis: Aplikasi seperti apa yang dibutuhkan untuk membantu proses bisnis dropship tersebut?**

Narasumber: Untuk aplikasi mungkin aplikasi yang bisa ngedata mulai dari customer, toko yang dijadikan supplier, produk yang dijual apa aja, dan yang paling penting bisa liat daftar pesanan yang masuk. Lebih enak lagi kalau bisa diakses lewat hp, karena sebagian besar proses pesanan dari customer dilakukan lewat hp.