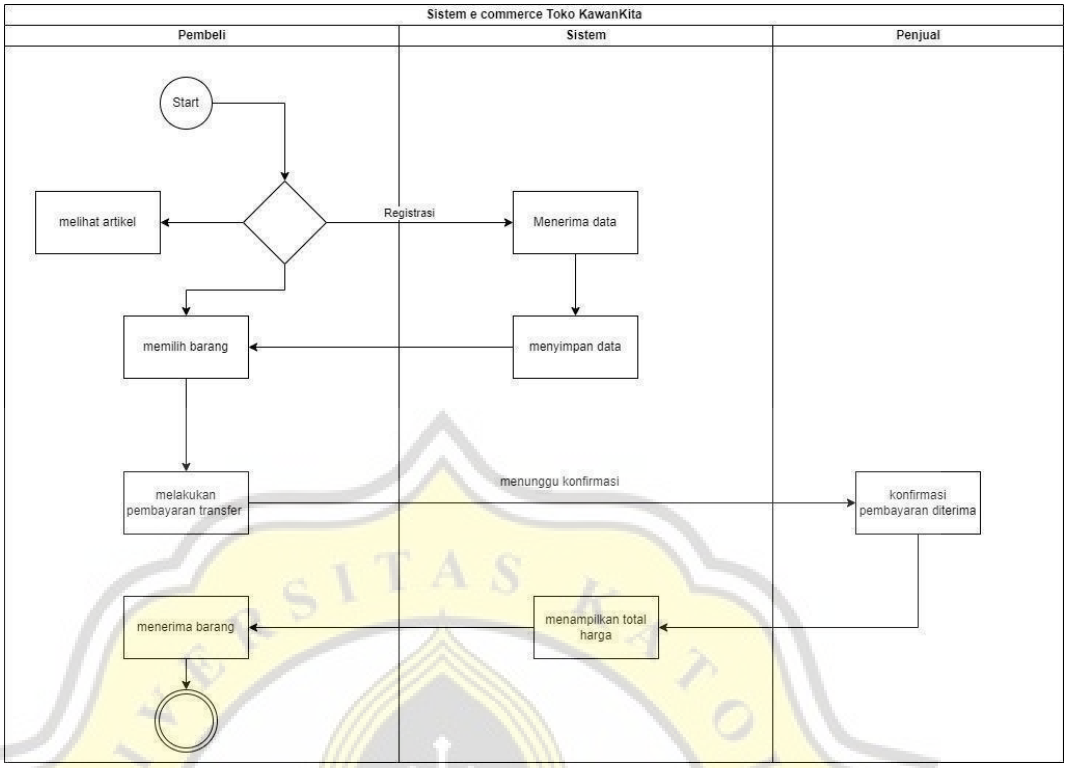
BAB IV PEMBAHASAN DAN HASIL PENELITIAN

4.1 Metode pengembagan website

Metode dalam pengembangan website ini yang digunakan menggunakan metode yang dinamakan “waterfall” dimana langkah demi langkah saling berkaitan dan berkelanjutan. Dalam langkah pertama yang diambil untuk menganalisa bagaimana website ini dibuat adalah dengan melakukan proses komunikasi dengan pemilik “toko sepeda Kawan Kita” untuk menentukan rangka dari website ini. Langkah kedua adalah dengan membuat desain dari website ini dari mulai bagian menganalisa proses bisnis, membuat tabel ERD, serta membuat desain tampilan atau UI/UX website. Dan langkah terakhir yang dilakukan dalam pembuatan website adalah testing untuk mengecek berbagai bug yang ada sehingga dapat diperbaiki dan dilakukan maintenance.

4.2 Proses bisnis website

Dalam pembuatan suatu website perlu diketahui bagaimana akan terjadinya proses bisnis dari user dari pertama hingga selesai untuk memudahkan memetakan bagaimana website akan dibuat dan memudahkan untuk pembuatan desain database dari website tersebut.Pada gambar 4.1 di bawah memperlihatkan bagaimana proses bisnis dari website Toko Sepeda Kawan Kita, disini terlihat interaksi antara pembeli dan system, system dengan penjual, dan penjual dengan pembeli.



Gambar 4.1 Proses Bisnis website Toko Kawan Kita

Diawal proses bisnis ini adalah Ketika pembeli membuka website ditandai dengan label Start disitu pembeli dapat memilih untuk melakukan registrasi atau tidak, Ketika pembeli tidak memilih untuk melakukan regis pembeli hanya dapat melihat dari segi tampilan artikel dan untuk melanjutkan proses transaksi Pembeli atau user harus melakukan proses registrasi untuk validasi data pembeli.

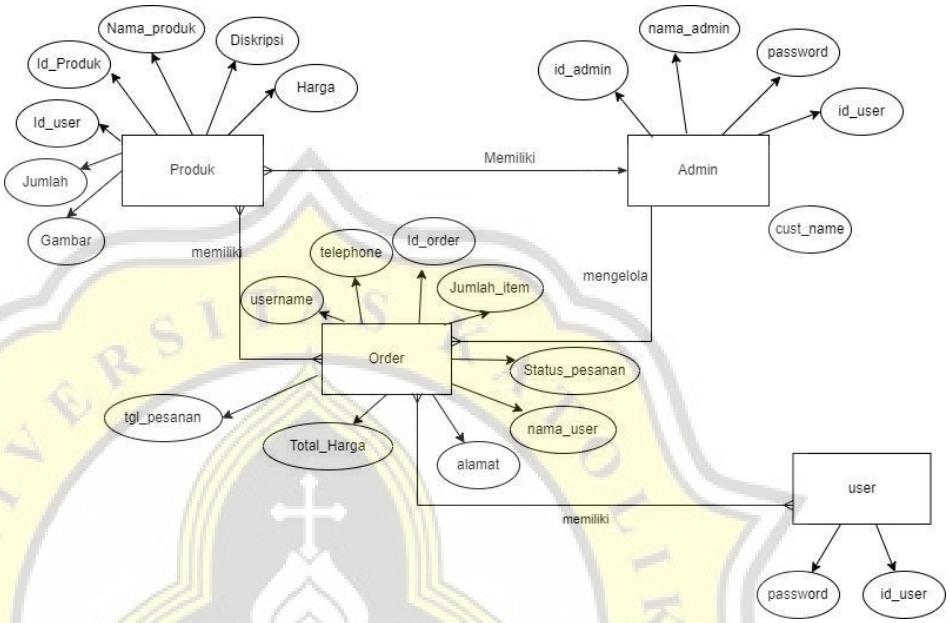
4.3 Perancangan aplikasi

Perancangan aplikasi ini bertujuan melakukan pencatatan dan melakukan perencanaan kedepannya untuk pembuatan aplikasi proses ini dilakukan setelah kita mengetahui proses bisnis bagaimana yang akan dilakukan system. Dari sumber proses bisnis Gambar 4.1 Proses Bisnis website Toko Kawan *Kita* kita dapat melakukan perancangan awal disini setelah itu kita melakukan perancangan dimulai dengan perancangan UI/UX untuk pengguna atau user.

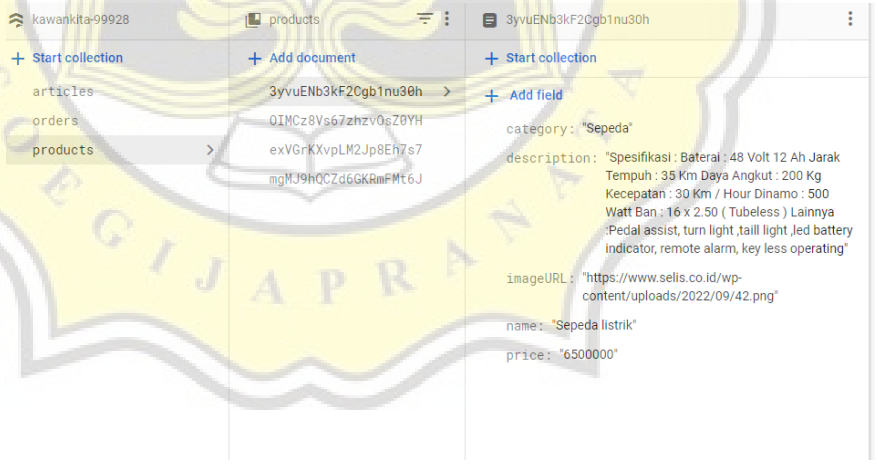
4.1 Pembuatan ERD

Dalam pembuatan website diperlukan juga rancangan daripada ERD untuk memudahkan pemetaan data di database

nantinya. Gambar 4.2 di bawah merupakan bagaimana hubungan antara entitas terhubung satu sama lainnya dan contoh rancangan di dalam Firebase bisa dilihat di gambar dibawah.



Gambar 4.2 Rancangan ERD

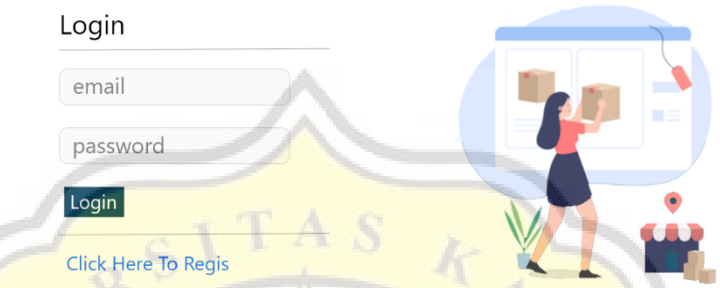


Gambar 4.3 Tabel database dari firebase

4.2 Perancangan UI dan UX

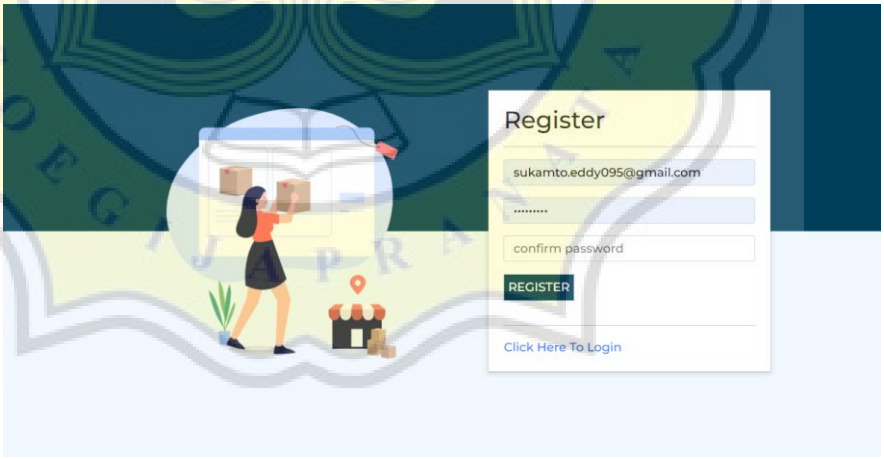
Perancangan User interface dan User experience bertujuan untuk mempermudahkan user untuk menggunakan website ini, karena website ini menggunakan konsep e-commerce maka

mengedepankan kenyamanan dan keamanan user sangat diutamakan. Gambar 4.4 di bawah ini merupakan tampilan UI halaman login mengenai bagaimana tampilan login user menggunakan template dari react.



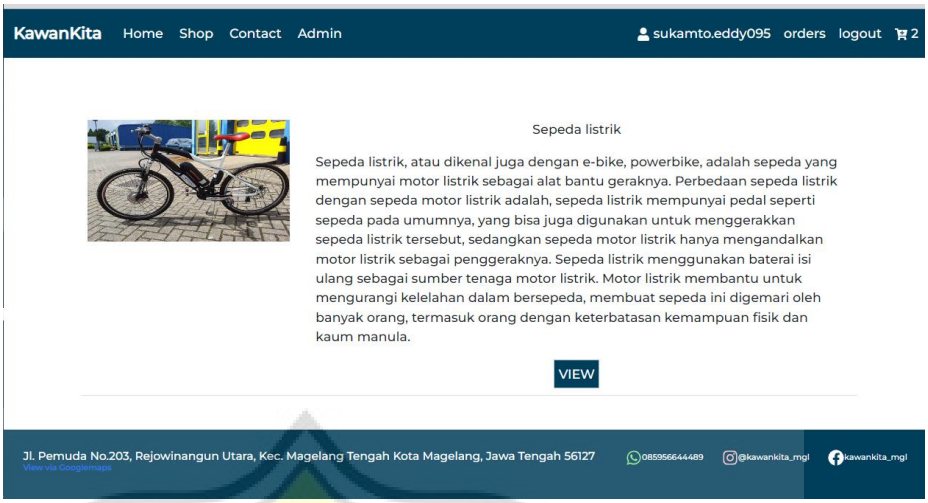
Gambar 4.4 UI halaman login

Gambar 4.5 dibawah merupakan tampilan dari halaman register untuk mendaftarkan id dan password user yang ingin masuk ke dalam website.



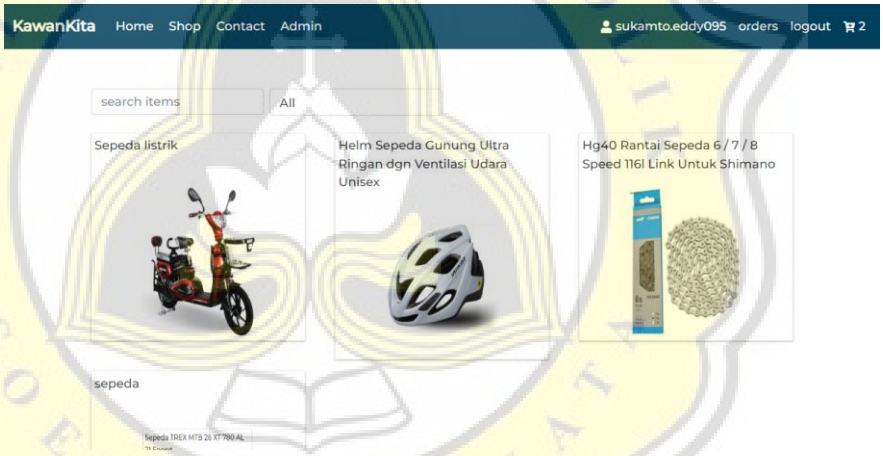
Gambar 4.5 UI halaman register

Gambar 4.6 dibawah ini merupakan menu homepage yang menunjukan halaman utama dari website ini akan menampilkan Artikel terkait sepeda seperti gambar diatas.



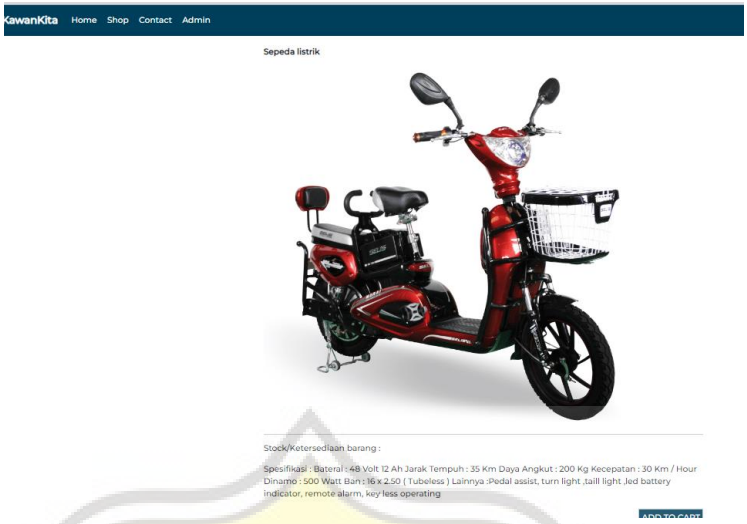
Gambar 4.6 Homepage

Gambar 4.7 di bawah ini merupakan gambar dari menu Shop dimana user akan memilih barang yang diinginkan.



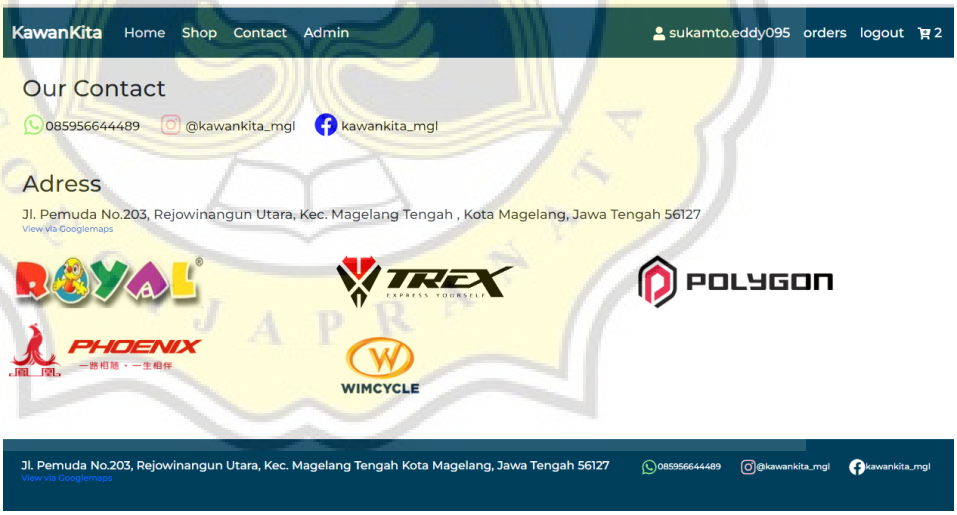
Gambar 4.7 Menu Shop

Gambar 4.8 Detail Product menunjukan detail dari produk dan ketersediaan barang yang didisplay.



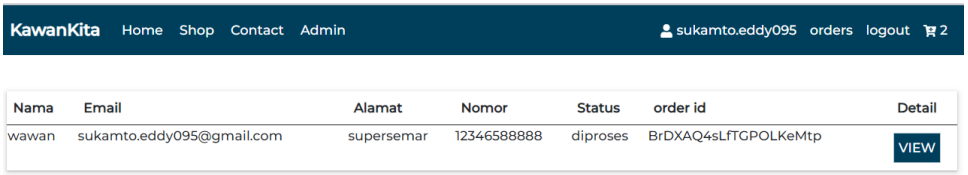
Gambar 4.8 Detail Product

Gambar 4.9 dibawah ini menunjukan menu contact yang berisikan contact dan social media yang dapat dihubungi beserta brand produk yang dijual oleh toko.



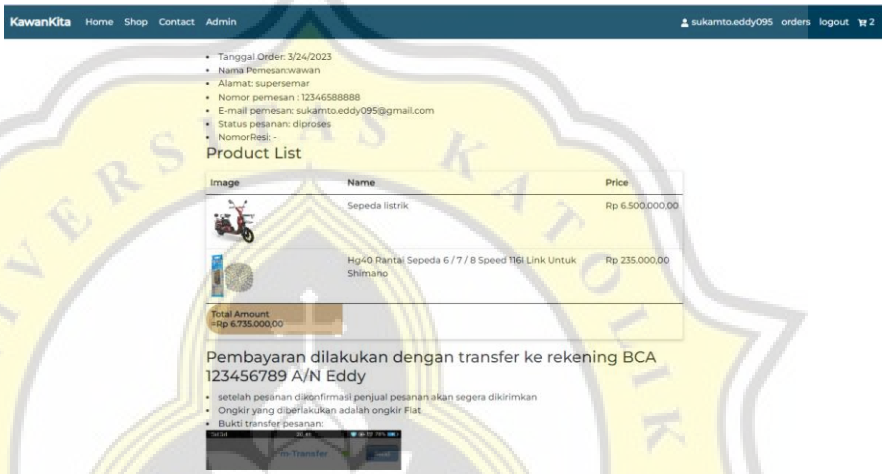
Gambar 4.9 Menu Contact

Gambar 4.10 dibawah ini merupakan menu yang menunjukan bagian list product apapun yang sudah konsumen order di website.



Gambar 4.10 Menu Orders

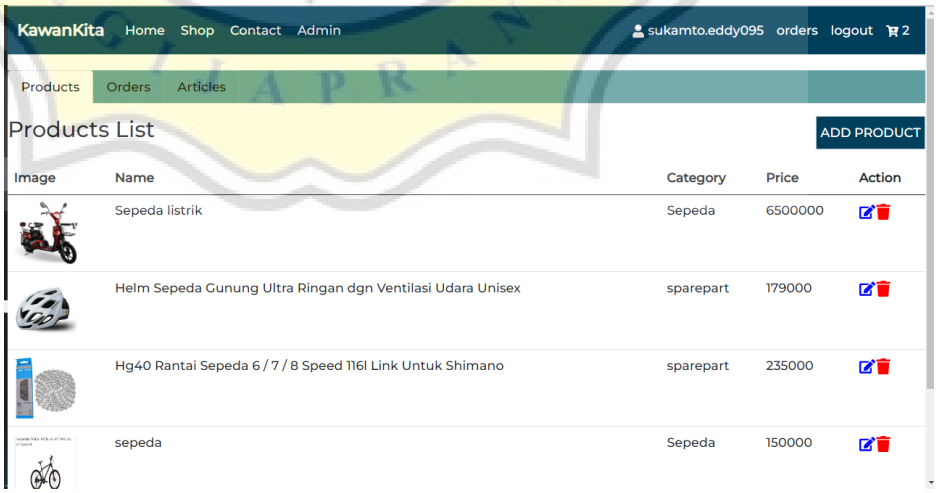
Gambar 4.11 dibawah adalah halaman dimana detail dan foto bukti dari pesanan konsumen terlihat dan.



\

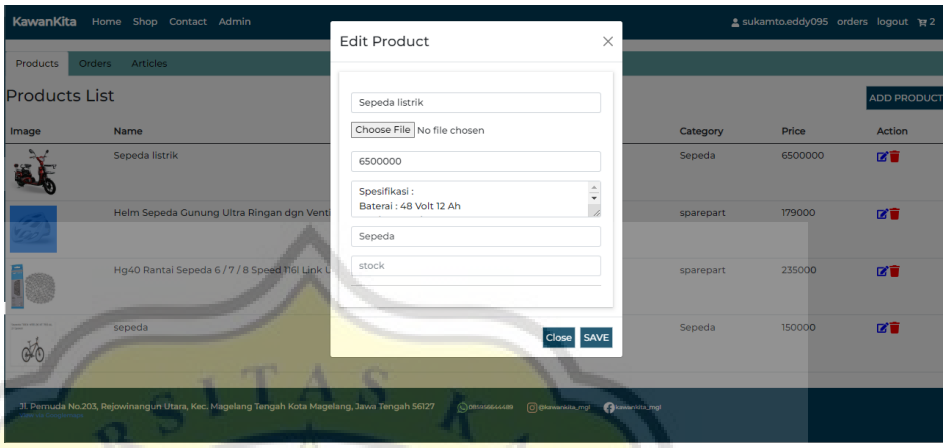
Gambar 4.11 Halaman Detail Order

Gambar 4.12 dibawah merupakan tampilan dari menu admin dalam tab product dimana admin akan mengupload produknya.



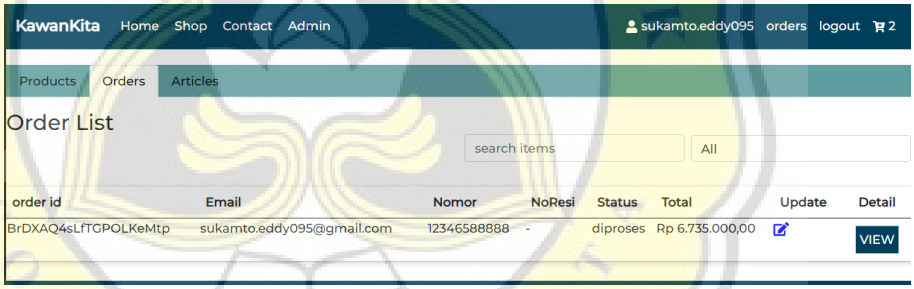
Gambar 4.12 Menu Admin Product

Gambar 4.13 dibawah ini merupakan popup form edit, dimana merupakan salah satu menu dalam dashboard admin. Pada halaman ini admin dapat mengedit product yang sudah diupload.



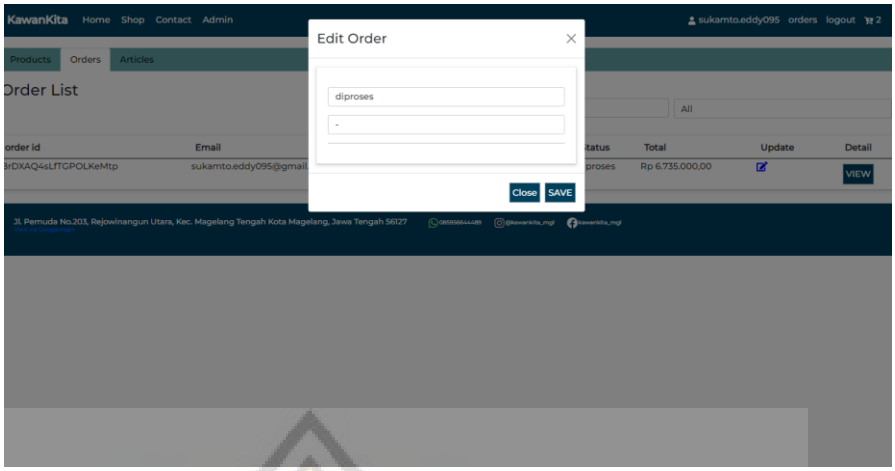
Gambar 4.13 Edit Product

Gambar 4.14 dibawah ini adalah tampilan untuk memproses order dari pembeli dari halamn admin.



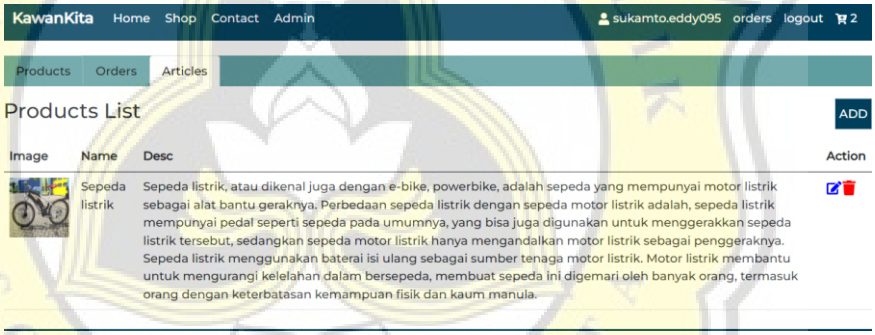
Gambar 4.14 Menu Admin Order

Gambar 4.15 dibawah menunjukan dimana Admin dapat memproses pesanan dan menambahkan nomor resi dari pesanan.



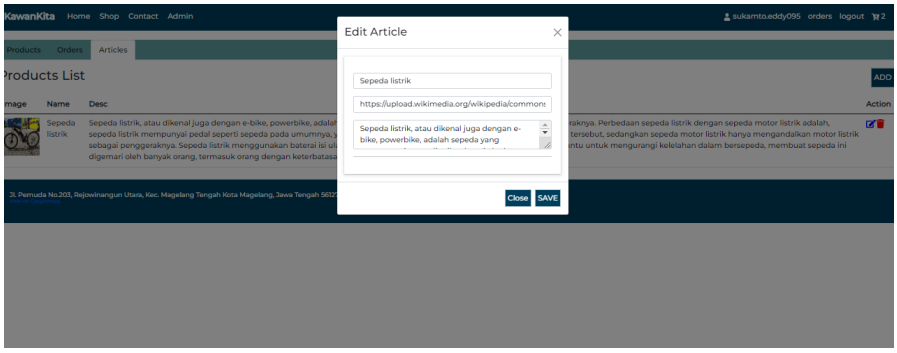
Gambar 4.15 Menu Edit Order

Gambar 4.16 dibawah merupakan menu admin articles adalah tampilan dimana admin mengupload artikel di tampilan homepage.



Gambar 4.16 Menu Admin Articles

Gambar 4.17 merupakan popup yang menunjukan form yang akan muncul ketika admin hendak mengedit artikel yang sudah didisplaykan.

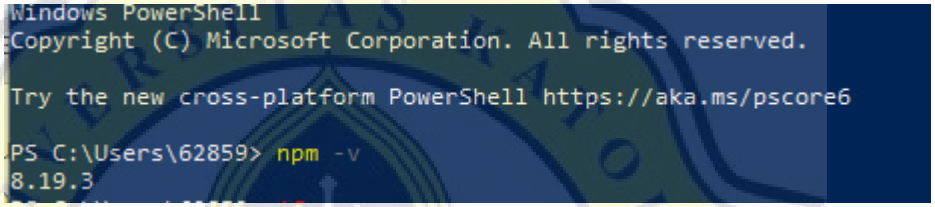


Gambar 4.17 Pop Up Edit Article

4.3 Pembuatan Aplikasi

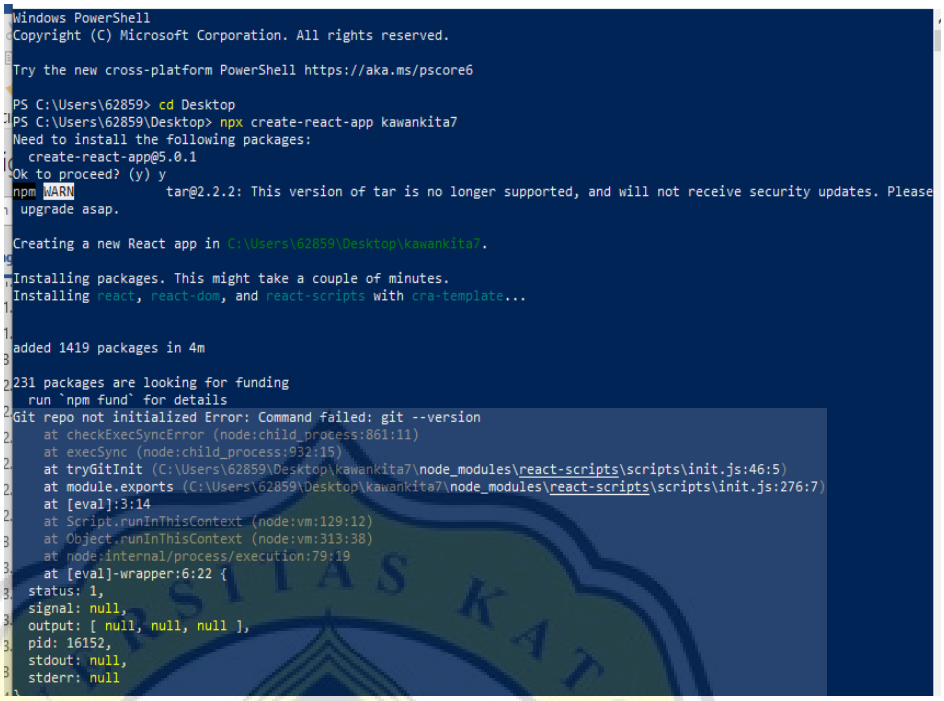
Website ini dibuat menggunakan Node Js pada back-endnya dan untuk tampilan website ini dibuat menggunakan React Js. Sedangkan dari segi database website ini menggunakan firebase sebagai penyimpanan data dan penyedia hostingnya.

4.3.1 Penginstalan Direktori

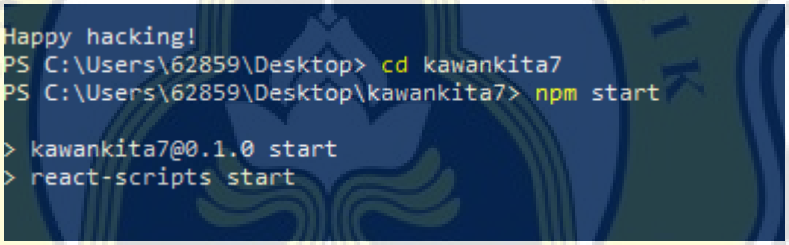


Gambar 4.16 Memeriksa versi Node.js

Hal yang dapat dilakukan sebelum penginstalan direktori framework ReactJs adalah mengecek apakah Node.Js sudah memiliki versi yang mendukung untuk melakukan penginstalan ReactJs. Gambar 4.17 Penginstalan Directory React Js menunjukan penginstalan directory React Js dengan nama kawankita7.



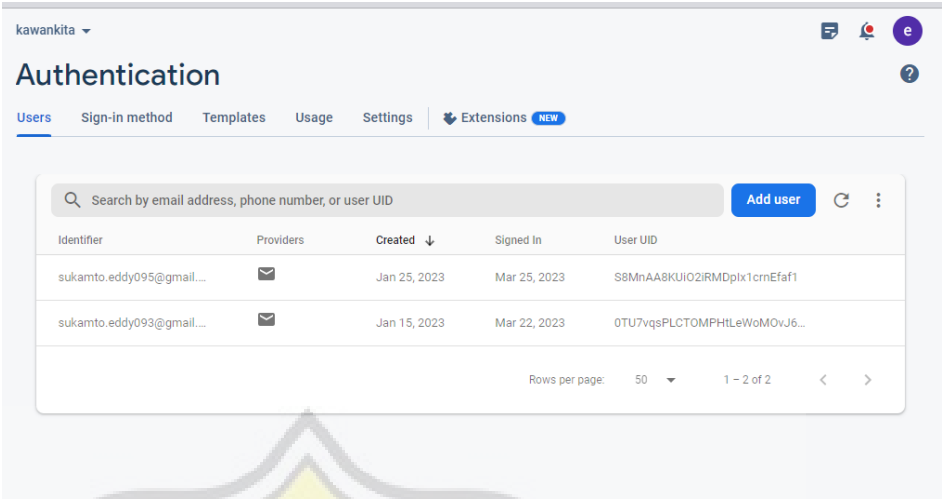
Gambar 4.17 Penginstalan Directory ReactJs



Gambar 4.18 Masuk Directory ReactJs

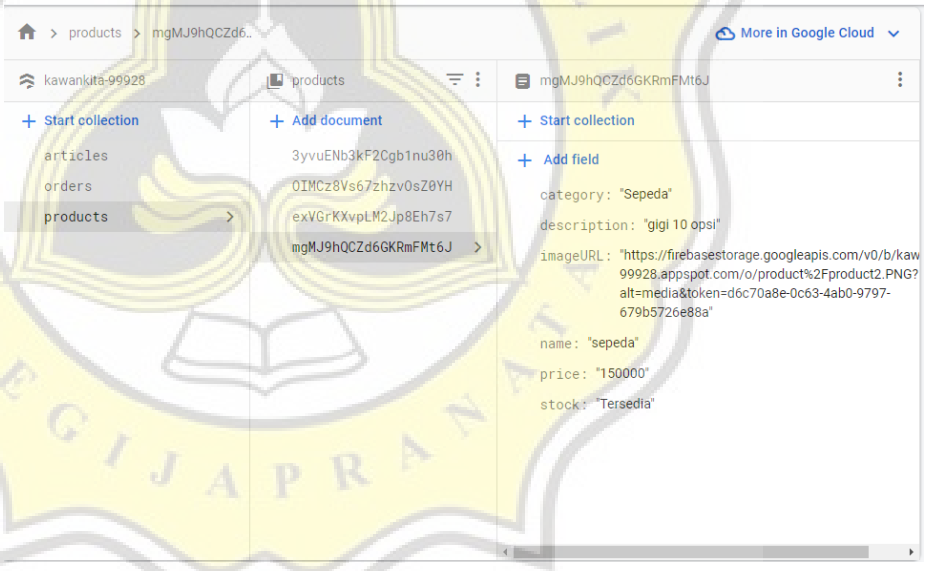
4.3.2 Database

Database yang digunakan berbasis firebase yang akan memudahkan dalam pembuatan table user dan memudahkan proses autentifikasi dari email yang sudah dibuat. gambar 4.19 Authentication User Dibawah adalah gambar dimana table berisikan user-user yang sudah mendaftar akan masuk ke dalam table authentication .



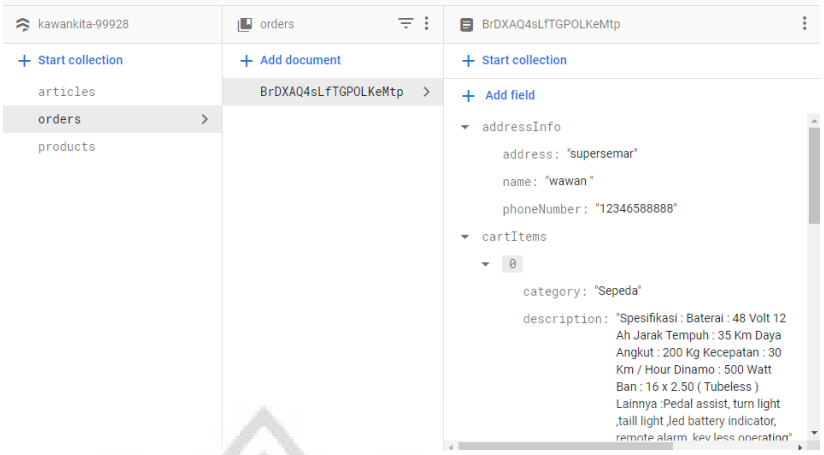
Gambar 4.19 Authentication User

Pada gambar 4.20 dibawah memperlihatkan tabel produk beserta isi dari field nya yang menunjukan isi dari produk yang disimpan dalam database



Gambar 4.20 Table Product Firestore Database

. Sedangkan pada gambar 4.21 menunjukan isi dari table order di dalam firestore database yang dimana ketika user melakukan order data akan masuk pada tabel ini.



Gambar 4.21 Table Order Firestore Database

4.3.3 Configuration project

Untuk menyambungkan project dengan firebase memerulikan configuration dari project di localhost sebelum di hosting dengan firebase.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBvBWQjmcfGZCCuJxTbYqSIoDekK6-3saM",
  authDomain: "kawankita-99928.firebaseapp.com",
  databaseURL: "https://kawankita-99928-default-rtdb.asia-southeast1
  projectId: "kawankita-99928",
  storageBucket: "kawankita-99928.appspot.com",
  messagingSenderId: "418813350230",
  appId: "1:418813350230:web:51bce9cb57d9311d51c6f7",
  measurementId: "G-V9FR1MRDMQ"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Gambar 4.22 adalah gambar yang memperlihatkan API yang digunakan dalam menyambungkan project dengan firebase. Dan untuk itu kita membuat file yang dinamakan firebaseConfig.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBvBWQjmcfGZCCuJxTbYqSIoDekK6-3saM",
  authDomain: "kawankita-99928.firebaseapp.com",
  databaseURL: "https://kawankita-99928-default-rtdb.asia-southeast1
  projectId: "kawankita-99928",
  storageBucket: "kawankita-99928.appspot.com",
  messagingSenderId: "418813350230",
  appId: "1:418813350230:web:51bce9cb57d9311d51c6f7",
  measurementId: "G-V9FR1MRDMQ"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

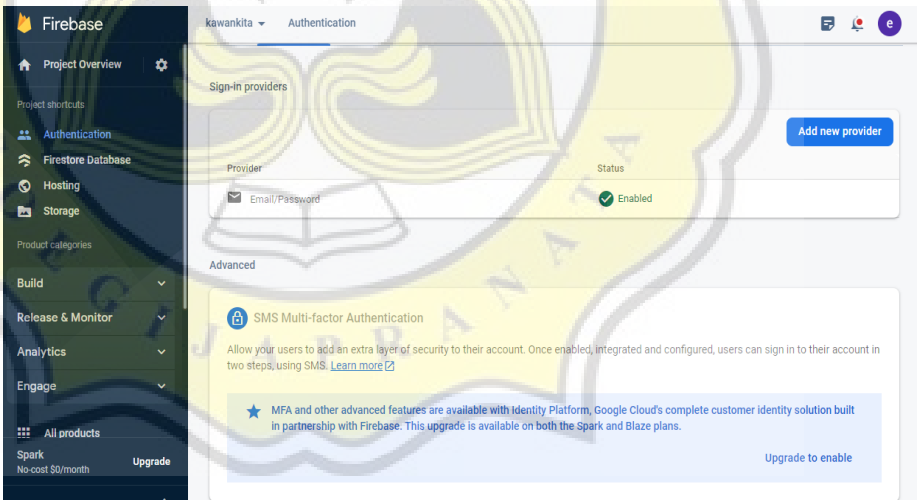
Gambar 4.22 Firebase config SDK

Pada gambar 4.23 memperlihatkan Api Key yang dideklarasikan untuk memanggil fungsi dari firebase (penyimpanan data) dan

firestore (penyimpanan data berupa file).

```
src > JS fireConfig.js > [@] firebaseConfig
1  import { initializeApp } from "firebase/app";
2  import { getFirestore } from "firebase/firestore";
3
4  const firebaseConfig = {
5    apiKey: "AIzaSyBvBWQjmcfGZCCuJxTbYqSIoDekK6-3saM",
6    authDomain: "kawankita-99928.firebaseapp.com",
7    projectId: "kawankita-99928",
8    storageBucket: "kawankita-99928.appspot.com",
9    messagingSenderId: "418813350230",
10   appId: "1:418813350230:web:51bce9cb57d9311d51c6f7",
11   measurementId: "G-V9FR1MRDMQ"
12 };
13
14 // Initialize Firebase
15 const app = initializeApp(firebaseConfig);
16
17 const fireDB = getFirestore(app)
18
19 export default fireDB
```

Gambar 4.23 function dari Firebase Api Key



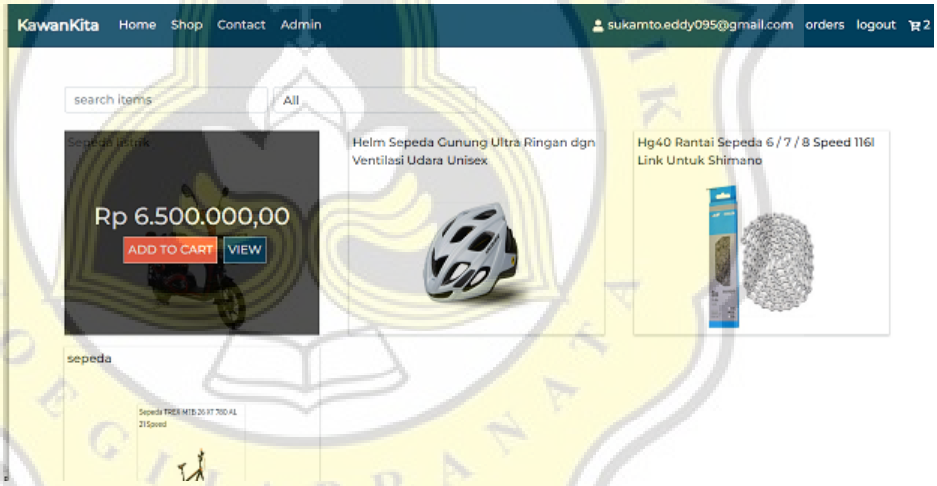
Gambar 4.24 Pengaturan fungsi auth Firebase

Gambar di atas menunjukan pengaturan di dalam firebase auth untuk penggunaan email sebagai validasi user masuk ke dalam website.

4.3.4 Proses Transaksi

Sebagai e-commerce fungsi utama dari website ini adalah untuk menjual produk penjual dengan perantara website ini. Dengan artian bahwa di dalam website ini proses transaksi adalah fungsi utama dari website ini yang dimana menemukan secara tidak langsung antara penjual dan pembeli.

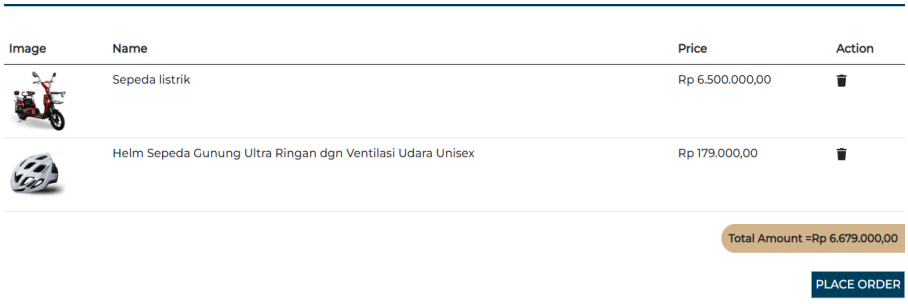
Gambar 4.25 dibawah merupakan tampilan dari menu shop yang menampilkan semua produk yang dijual oleh website. Dalam tampilan ini ketika user memilih untuk menekan tombol "add to chart" maka otomatis barang akan masuk ke chart yang sudah tersedia. Sedangkan ketika user memilih untuk menekan tombol view user akan dialihkan ke halam detail dari produk pada gambar 4.26.



Gambar 4.25 shop

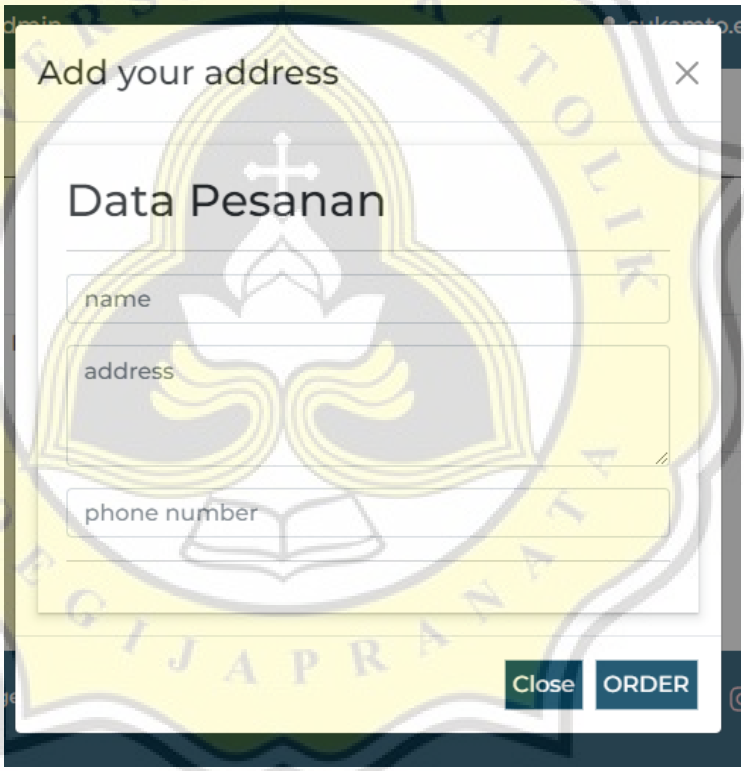
Gambar 4.26 detail product

Dalam halaman ini disajikan detail dari barang mulai dari nama, ketersediaan stok dan detail spesifikasi dari produk yang tertera di deskripsi produk tersebut dimana jika user memilih produk tersebut akan disediakan tombol “add to chart” di bawah halaman tersebut.



Gambar 4.27 chart

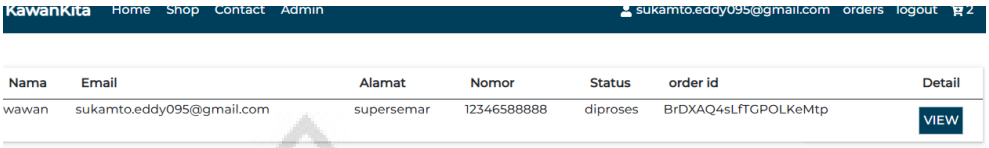
Dalam tampilan keranjang disajikan barang apa saja yang sudah dipilih untuk dimasukan ke dalam char atau user sudah memilih untuk menekan tombol “add chart”.



Gambar 4.28 popup place order

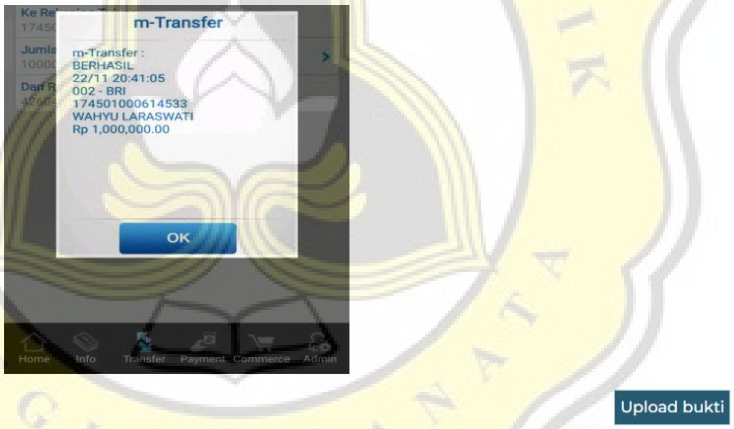
Ketika konsumen sudah yakin dengan produk yang akan dibelinya maka konsumen dapat menekan tombol place order yang akan memunculkan popup untuk konsumen mengisikan data dirinya. Setelah semua data terisi konsumen dapat menekan order untuk melanjutkan pesanan agar bisa diproses oleh penjual.

Pada gambar 4.29 dibawah merupakan tampilan dari order list dimana semua order dari konsumen akan ditampilkan. dalam list tersebut bisa terlihat status pesanan dan jika konsumen ingin melihat detail dari pesanan dan mengirimkan bukti transfernya, konsumen dapat melihat di bagian view .



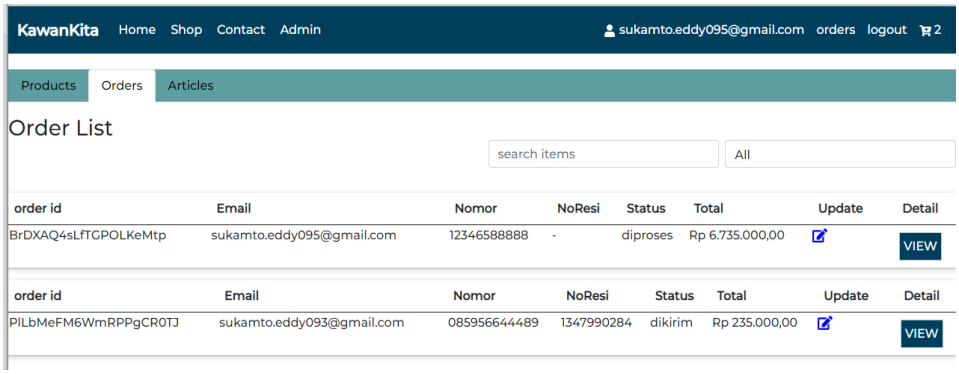
Gambar 4.29 Order List

Gambar 4.30 menunjukan ketika konsumen sudah selesai dengan mengupload foto bukti transfer. Bukti akan ditampilkan kembali di dalam menu view seperti dibawah ini.



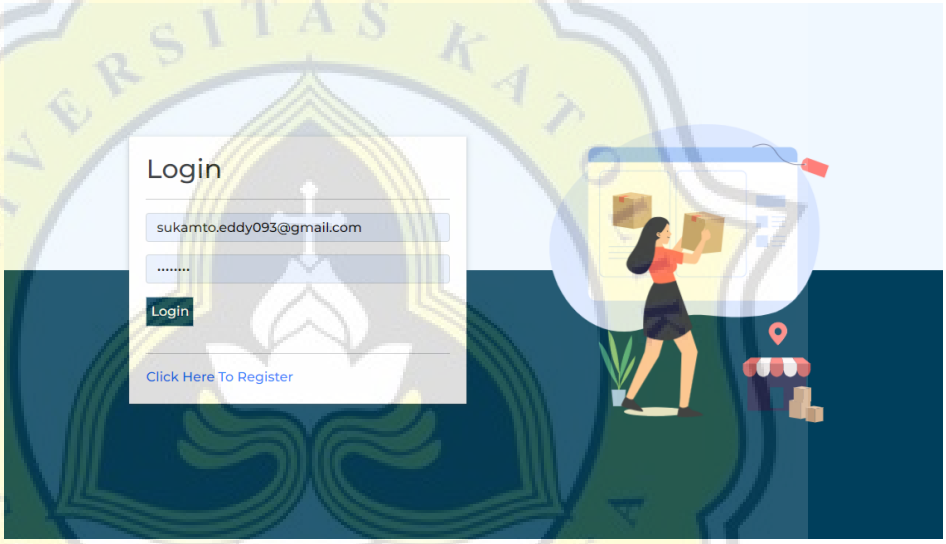
Gambar 4.30 Upload bukti

Pada gambar 4.30 dibawah merupakan gambar dimana admin akan melihat semua order yang masuk ke website tersebut. Disini admin akan melakukan pengecekan dan update terhadap order yang sudah dibayarkan. Jika memang sudah dibayarkan, Admin akan mengupdate status dari barang pesanan dari “diproses” menjadi “dikirim”. Dan admin juga akan melampirkan nomor resi pembeli yang sudah dikirimkan.



Gambar 4.31 Order List Admin

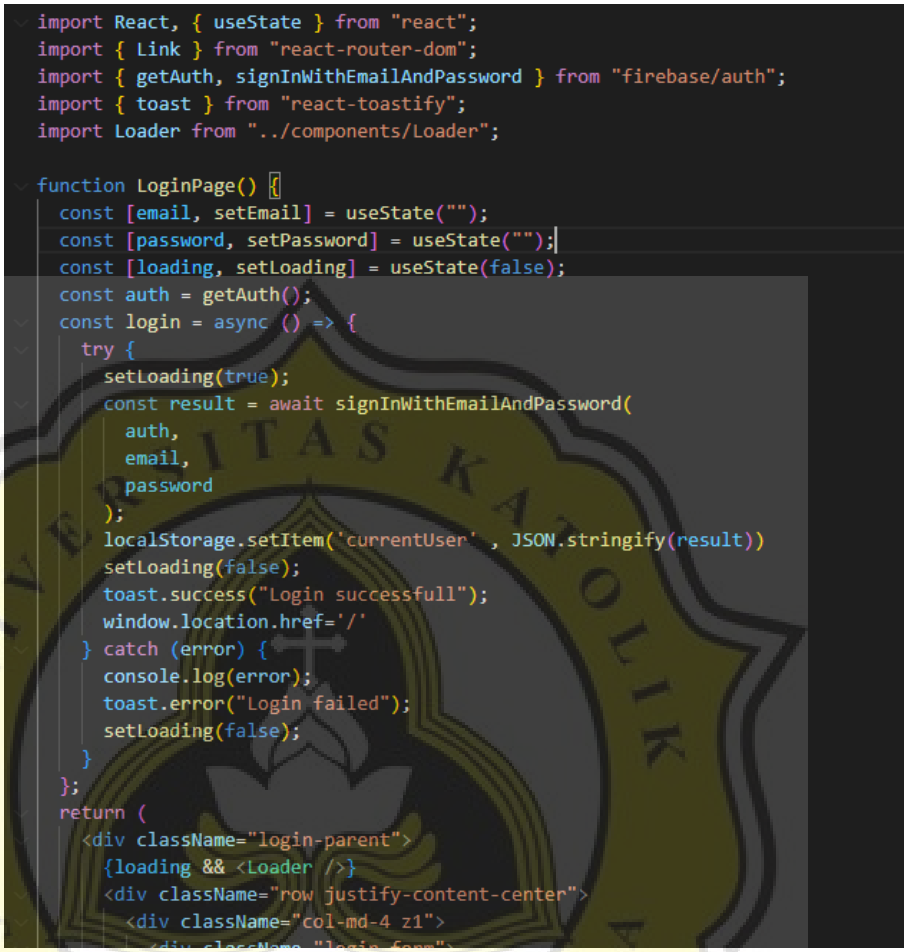
4.3.5 Halaman Login



Gambar 4.32 Halaman Login

Gambar di atas adalah gambar halaman login untuk user dimana akan melakukan login atau validasi apakah user pernah mendaftarkan emailnya atau belum.script dalam gambar 4.33 dibawah merupakan beberapa function pemanggilan dari beberapa direktori dan pendeklarasian variabel dalam pembuatan halaman login page. Dimana dibuatnya function login untuk mendapatkan data dari field input yang sudah diisikan Adapun code field input bisa dilihat pada gambar dibawah. Pada gambar 4.34 dibawah ini adalah script untuk menampilkan field input dan tombol login dimana ketika tombol login di tekan akan memasukan value atau isi field

pengautentifikasian dari firebase bahwa email dan password sudah pernah didaftarkan.



```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
import { toast } from "react-toastify";
import Loader from "../components/Loader";

function LoginPage() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);
  const auth = getAuth();
  const login = async () => {
    try {
      setLoading(true);
      const result = await signInWithEmailAndPassword(
        auth,
        email,
        password
      );
      localStorage.setItem('currentUser' , JSON.stringify(result))
      setLoading(false);
      toast.success("Login successfull");
      window.location.href='/'
    } catch (error) {
      console.log(error);
      toast.error("Login failed");
      setLoading(false);
    }
  };
  return (
    <div className="login-parent">
      {loading && <Loader />}
      <div className="row justify-content-center">
        <div className="col-md-4 z1">
```

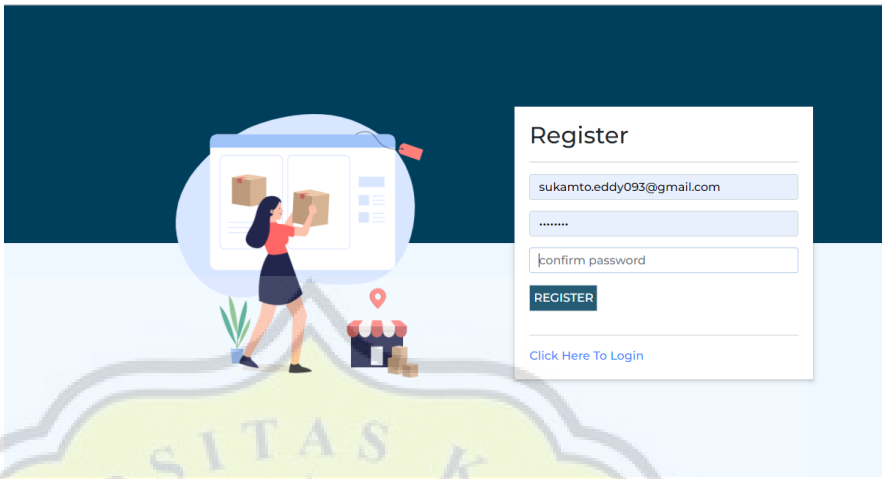
Gambar 4.33 import function dan deklarasi variable

Jika user belum pernah mendaftarkan e-mail dan password maka disediakan regispage seperti gambar dibawah ini seperti pada gambar 4.34

```
<div className="login-parent">
  {loading && <Loader />}
  <div className="row justify-content-center">
    <div className="col-md-4 z1">
      <div className="login-form">
        <h2>Login</h2>

        <hr />

        <input
          type="text"
          className="form-control"
          placeholder="email"
          value={email}
          onChange={(e) => {
            setEmail(e.target.value);
          }}
        />
        <input
          type="password"
          className="form-control"
          placeholder="password"
          value={password}
          onChange={(e) => {
            setPassword(e.target.value);
          }}
        />

        <button className="my-3" onClick={login}>Login</button>

        <hr />
```

Gambar 4.34 Code input dan button Login

pada gambar 4.35 dibawah menunjukan tampilan dalam halaman registrasi user di dalam website ini halaman ini.



Gambar 4.35 Halaman Registrasi

sedangkan gambar 4.36 di bawah ini menunjukan pemanggilan function createUserWithEmailAndPassword dalam direktori firebase/auth yang akan menjalankan function dimana ketika data yang diperoleh dari field form dan akan dimasukan ke dalam table auth sebagai member baru.

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";
import Loader from "../components/Loader";
import { toast } from "react-toastify";
function RegisterPage() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [cpassword, setCPassword] = useState("");
  const [loading, setLoading] = useState(false);
  const auth = getAuth();

  const register = async () => {
    try {
      setLoading(true);
      const result = await createUserWithEmailAndPassword(
        auth,
        email,
        password
      );
      console.log(result);
      setLoading(false);
      toast.success("Registration successfull");
      setEmail('')
      setPassword('')
      setCPassword('')
    } catch (error) {
      console.log(error);
      toast.error("Registration failed");
      setLoading(false);
    }
  };
```

Gambar 4.36 Script registrasi

Gambar 4.37 dibawah ini menunjukan script field yang akan mengambil data yang sudah diisi oleh pengguna yang

akan dikirimkan ke table firebase auth ketika tombol register di tekan

.

```
<input
  type="text"
  className="form-control"
  placeholder="email"
  value={email}
  onChange={(e) => {
    setEmail(e.target.value);
  }}
/>
<input
  type="password"
  className="form-control"
  placeholder="password"
  value={password}
  onChange={(e) => {
    setPassword(e.target.value);
  }}
/>
<input
  type="text"
  className="form-control"
  placeholder="confirm password"
  value={cpassword}
  onChange={(e) => {
    setCPassword(e.target.value);
  }}
/>

<button className="my-3" onClick={register}>
  REGISTER
</button>
```

Gambar 4.37 Fungsi register

4.3.6 Menu Website

Dalam website terdiri dari beberapa menu untuk tampilan yang menampilkan beberapa kategori halaman untuk memisahkan halaman satu dan lainnya.

Gambar 4.38 dibawah ini adalah gambar dimana ketika user memasuki website ini akan menemui halaman berisikan artikel sebagai homepage atau halaman utama website. Sedangkan pada

Gambar 4.39 dan Gambar 4.40 dibawah menunjukan dimana function ini memanggil data dimana akan dikelompokan berdasarkan id dari data tersebut.

Gambar 4.38 Menu Homepage

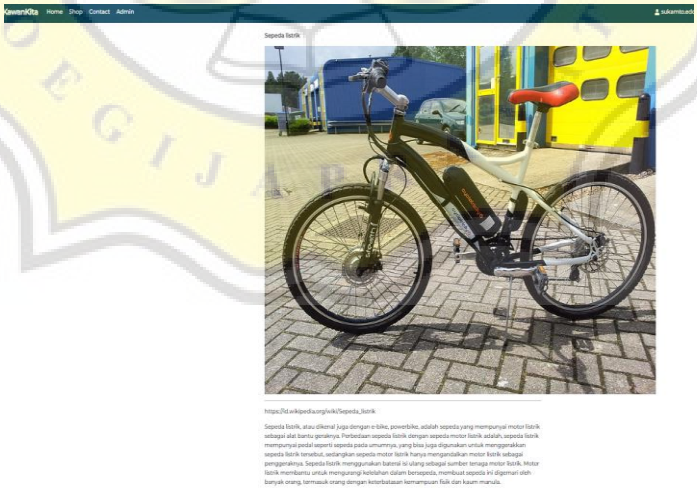
```
async function getData() {
  try {
    setLoading(true);
    const users = await getDocs(collection(fireDB, "articles"));
    const articlesArray = [];
    users.forEach((doc) => {
      const obj = {
        id: doc.id,
        ...doc.data(),
      };

      articlesArray.push(obj);
      setLoading(false);
    });

    setArticles(articlesArray);
  } catch (error) {
    console.log(error);
    setLoading(false);
  }
}
```

Gambar 4.39 Function Get data article

```
<div className="container">
  <div className="d-flex w-50 align-items-center my-3 justify-content-center">
  </div>
  <div className="row">
    {articles.map((article) => {
        return (
          <table className="table mt-3">
            <tbody>
          <tr>



          <td>

          <img src={article.imageURL} height="150" width="250"  />

          </td>
          <th></th>
          <td>
            <p className="product-desc">
            <p className="text-center">{article.name}</p>
            {article.description}
            </p>
            <div className="text-center">
          <button
                    onClick={() => {
                      navigate(`/ArticleInfo/${article.id}`);
                    }}
                  >
                    VIEW
                  </button>
            </div>
          </td>
```

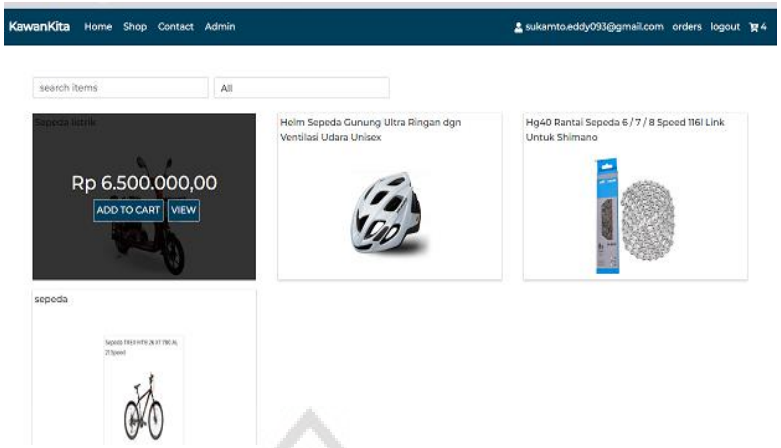
Gambar 4.40 Script tampilan homepage

Gambar 4.41 dibawah menunjukan bagaimana tampilan yang dihasilkan dari script diatas



Gambar 4.40 Tampilan menu Info Article

Gambar 4.41 merupakan tampilan dari menu shop dimana user akan melakukan pembelian atau mencari barang yang diinginkan.

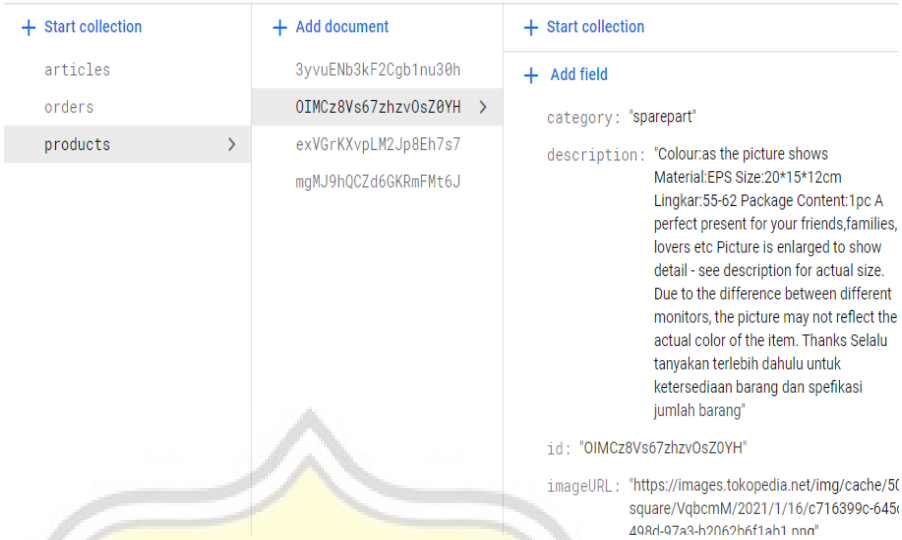


Gambar 4.41Menu Shop

Gambar 4.42memperlihatkan fungsi yang sama dengan fungsi pemanggilan data artikel pada menu Homepage, dimana data yang didapat dikelompokan berdasarkan id product yang disimpan di firebase.

```
async function getData() {
  try {
    setLoading(true);
    const users = await getDocs(collection(fireDB, "products"));
    const productsArray = [];
    users.forEach((doc) => {
      const obj = {
        id: doc.id,
        ...doc.data(),
      };

      productsArray.push(obj);
      setLoading(false);
    });

    setProducts(productsArray);
  } catch (error) {
    console.log(error);
    setLoading(false);
  }
}
```

Gambar 4.42 Function GetData Product



Gambar 4.43 Table Product

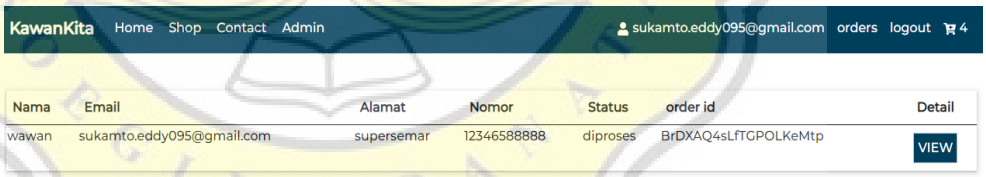
Gambar di atas merupakan tampilan tabel firebase untuk data product yang disimpan dalam firebase. Data ini yang dipanggil dan ditampilkan dalam halaman product. Gambar 4.48 di bawah ini memperlihatkan Halaman Contact yang berisikan contact person dari admin website dan beberapa sosial media yang dimiliki usaha tersebut.

Gambar 4.44 Halaman Contact

```
<table className="table mt-3">
  <thead>
    <tr>
    </tr>
  </thead>
</table>
<h3>Adress </h3>
Jl. Pemuda No.203, Rejowinangun Utara, Kec. Magelang Tengah
  , Kota Magelang, Jawa Tengah 56127
  <h5 className="navbar-brand-footer" >
  <a href="https://goo.gl/maps/4wqyVtNSCmmp1DGP6" target="_blank" rel="noopener noreferrer">
View via Googlemaps</a>
</h5>


</div>
<table className="table mt-3">
  <thead>

    <th>  <img src="logo/royal.jpg" height="80" width="250" /></th>
    <th>  <img src="logo/trex.png" height="80" width="230" /></th>
    <th>  <img src="logo/polygon.png" height="80" width="250" /></th>

  </thead>
  <tbody>
    <td><img src="logo/images.png" height="80" width="250" /></td>
    <td><img src="logo/wim.png" height="120" width="120" /></td>          </tbody>
</table>
</>
```

Gambar 4.45 Script Contact

Gambar 4.46 menunjukan tampilan untuk menu order pada tampilan website user dimna menu ini berisikan barang apa yang sudah pernah disorder melalui chart dan sudah melakukan checkout.

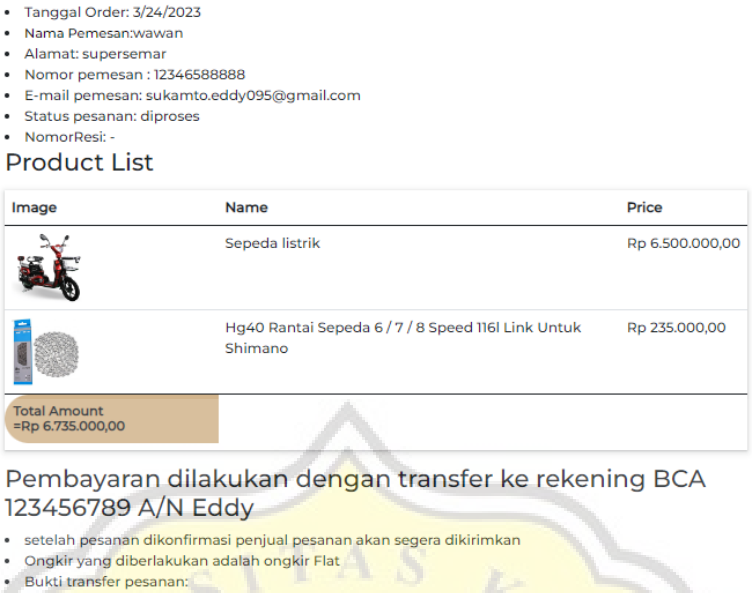


Gambar 4.46 Menu Order user

Script dibawah menunjukan tampilan yang berada di menu Order yang menunjukan semua order yang sudah dilakukan user yang di filter berdasar user id. Dan tombol view akan mengarahkan pengguna ke dalam halaman detail pesanan.

Gambar 4.47 Script menu order

Gambar 4.48 dibawah ini menampilkan bagaimana detail pesanan akan ditampilkan dan barang apa saja yang sudah dipesan oleh konsumen dalam tabel dibawah.



Gambar 4.48 Detail pesanan

Dibawah pada gambar 4.49 menunjukan script yang akan memanggil data order pengguna, Dimana order yang ditampilkan berdasarkan id order yang sudah ditetapkan.

```
<div className="row justify-content-center">
  <div className="col-md-8">
    {order && (
      <div>
        <li>Tanggal Order: {new Date(order.date.seconds * 1000).toLocaleDateString("en-US")}</
        <li>
        <a className="navbar-brand-address">Nama Pemesan:</a>{order.addressInfo.name}</li>
        <li>Alamat: {order.addressInfo.address}</li>
        <li>Nomor pemesan : {order.addressInfo.phoneNumber}</li>
        <li>E-mail pemesan: {order.email}</li>
        <li>Status pesanan: {order.status}</li>
        <li>NomorResi: {order.noresi}</li>
        <h3 className="justify-content-center" >Product List</h3>
      <table className="table mt-3 order">
        <thead>

          <tr>
            <th>Image</th>
            <th>Name</th>
            <th>Price</th>
          </tr>
        </thead>
        <tbody>
          {order.cartItems.map((item) => {
            return (
              <tr>
                <td>
                  <img src={item.imageURL} height="80" width="80" />
                </td>

                <td>{item.name}</td>
                <td> {numberFormat(item.price)}</td>
```

Gambar 4.49 Script detail order user

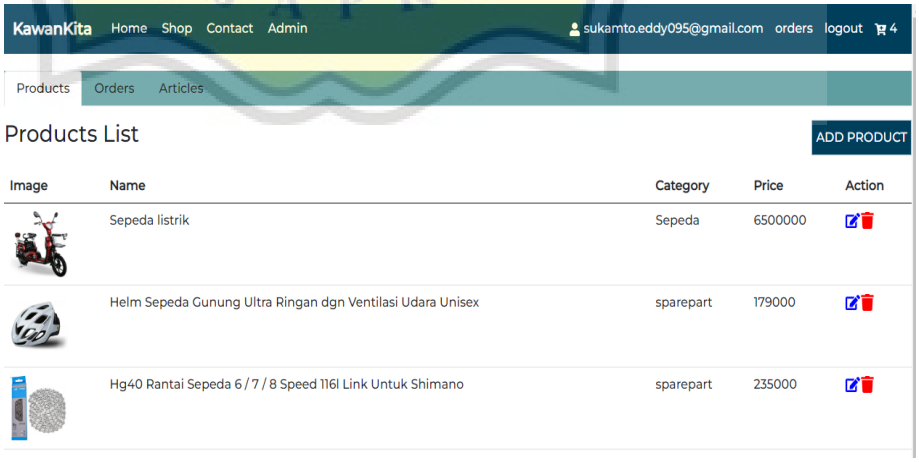
Pada gambar 4.50 dibawah script dari cart page dimana barang yang dipesan akan disimpan sementara di keranjang.

```
            <td>{item.name}</td>
            <td>{numberFormat (item.price)}</td>
            <td>
              <FaTrash onClick={() => deleteFromCart(item)} />
            </td>
          </tr>
        );
      })}
    </tbody>
  </table>
```

4.50 Script Cart Page

4.3.7 Dashboard Admin

Membangun dashboard untuk admin adalah langkah pertama dalam pembuatan website ini. Untuk dashboard admin dimulai dengan pembuatan tampilan dari segi admin menggunakan ReactJs. Menyiapkan dashboard admin akan mempermudah dalam memetakan database firebase. Gambar 4.51 dibawah menunjukan Dashboard admin, dimana adalah tampilan dimana admin akan mengupload file dan data yang berkaitan dengan product, order, dan artikel yang tertampil dalam website. gambar dibawah juga menunjukan dimana dashboard admin menunjukan bagaimana admin akan menambahkan ataupun mengedit produk yang akan ditampilkan.

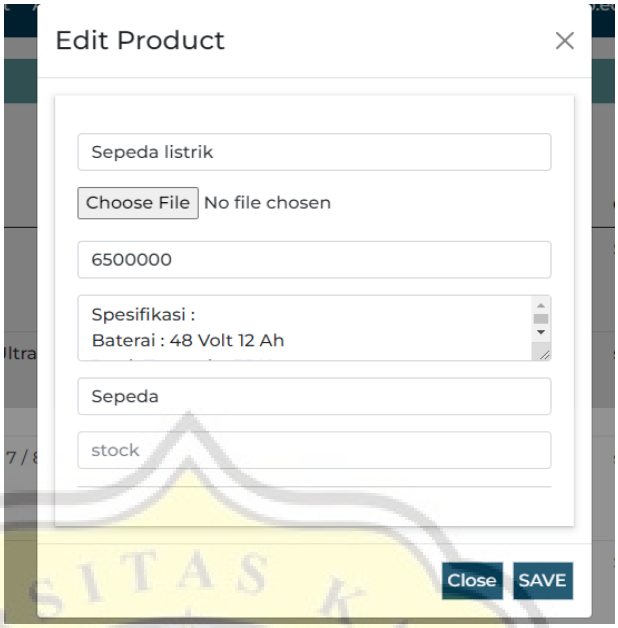


Gambar 4.51 Dashboard Admin

Gambar 4.52 dibawah merupakan script untuk menampilkan produk yang sudah di update pada table di dalam dashboard admin seperti gambar 4.52. Sedangkan pada gambar 4.53 dibawah memperlihatkan menu edit product pada dashboard admin yang memperlihatkan menu pop up edit produk yang sudah tersimpan di database. Selanjutnya Gambar 4.54 dibawah adalah gambar script untuk field input form edit product yang berisikan data yang diambil dari database lewat value field masing-masing.

```
>
  <Tab eventKey="products" title="Products">
    <div className="d-flex justify-content-between">
      <h3>Products List</h3>
      <button onClick={addHandler}>ADD PRODUCT</button>
    </div>
    <table className="table mt-3">
      <thead>
        <tr>
          <th>Image</th>
          <th>Name</th>
          <th>Category</th>
          <th>Price</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        {products.map((item) => {
          return (
            <tr>
              <td>
                <img src={item.imageURL} height="80" width="80" />
              </td>

              <td>{item.name}</td>
              <td>{item.category}</td>
              <td>{item.price}</td>
              <td>
```

Gambar 4.52 Script tampil produk dashboard admin



Gambar 4.53 Menu Edit product

Gambar 4.54 Script form edit product

Script dibawah merupakan tombol yang akan memanggil fungsi update product untuk mengubah value field database

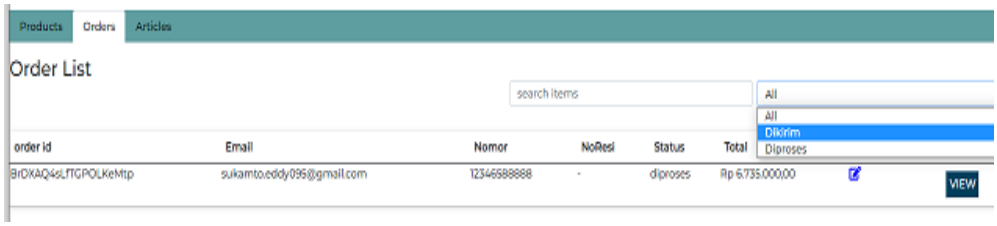
Gambar 4.55 Tombol update dan add product.

Script pada gambar 4.56 dibawah menunjukan bagaimana fungsi ini akan merubah data pada tabel product berdasarkan product.

```
const updateProduct = async () => {
  try {
    setLoading(true);
    await setDoc(doc(fireDB, "products", product.id), product);

    handleClose();
    toast.success("Product updated successfully");
    window.location.reload();
  } catch (error) {
    toast.error("Product update failed");
    setLoading(false);
  }
};
```

Gambar 4.56 Function update product

id dan diisi oleh variabel yang dimasukkan pada table edit product saat admin memasukan data dari produk yang akan di update.



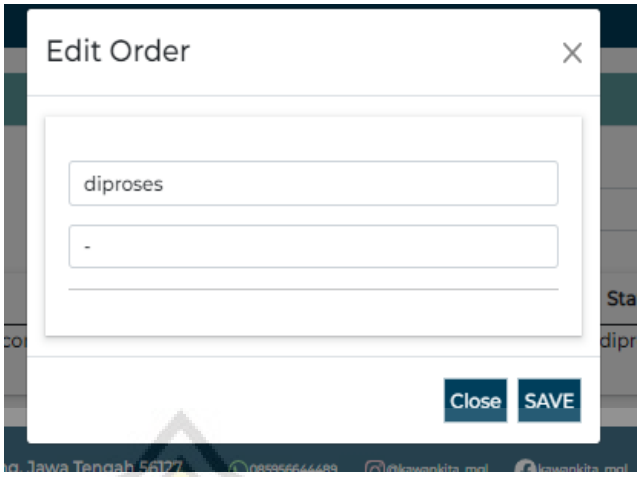
Gambar 4.57 Order dashboard admin

Gambar menu di dashboard admin ini berisikan menu tampilan pada menu order di dashboard admin. Tampilan ini bertujuan agar admin dapat memproses semua order yang masuk.

```
</div>
{orders
  .filter((obj) => obj.id.toLowerCase().includes(searchKey))
  .filter((obj) => obj.status.toLowerCase().includes(filterType))
  .map ((item) => {
  return (
    <table className="table mt-3 order">
      <thead>
        <tr>
          <th>order id</th>
          <th>Email</th>
          <th>Nomor</th>
          <th>NoResi</th>
          <th>Status</th>
          <th>Total</th>
          <th>Update</th>
          <th className="text-center">Detail</th>
        </tr>
      </thead>
      <tbody>
        <td>{item.id}</td>
        <td>{item.email}</td>
        <td>{item.addressInfo.phoneNumber}</td>
        <td>{item.noresi}</td>
        <td>{item.status}</td>
        <td>{numberFormat (item.total) }</td>

        <FaEdit
              onClick={() => editOrder(item)}
              color="blue"
              size={20}
            />
        <td>
```

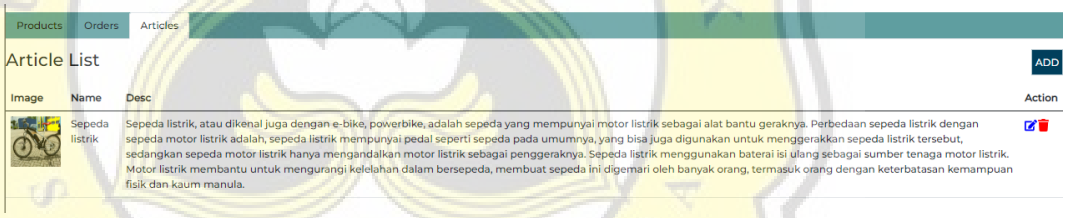
Script 4.58 Script menu order dashboard admin

Script diatas menghasilkan tampilan seperti pada gambar diatas yang menunjukan order yang masuk sesuai dengan ordered yang tertera, disini admin juga memfilter untuk mencari dimana order yang sudah dan belum terselesaikan agar dapat mempermudah pencarian order yang belum terselesaikan.

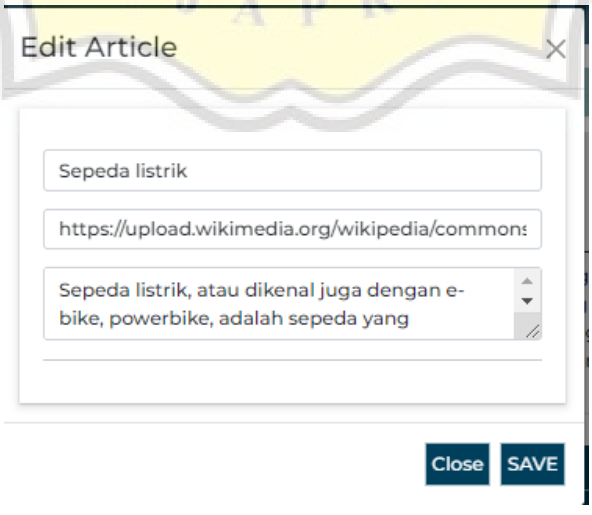


Gambar 4.59 Menu pop up edit order

Kegunaan popup ini untuk mengupdate nomor resi dan status pesanan konsumen yang masuk dalam menu order di menu dashboard admin. Gambar 4.60 dibawah ini menunjukan bagaimana article ditambahkan kedalam website melalui dashboard admin.



Gambar 4.60 Menu Article Admin Dashboard



Gambar 4.61 Menu Popup edit article

Menu popup disini berguna sebagai form edit sebagai peng-update an article jika ada perubahan yang diinginkan admin.

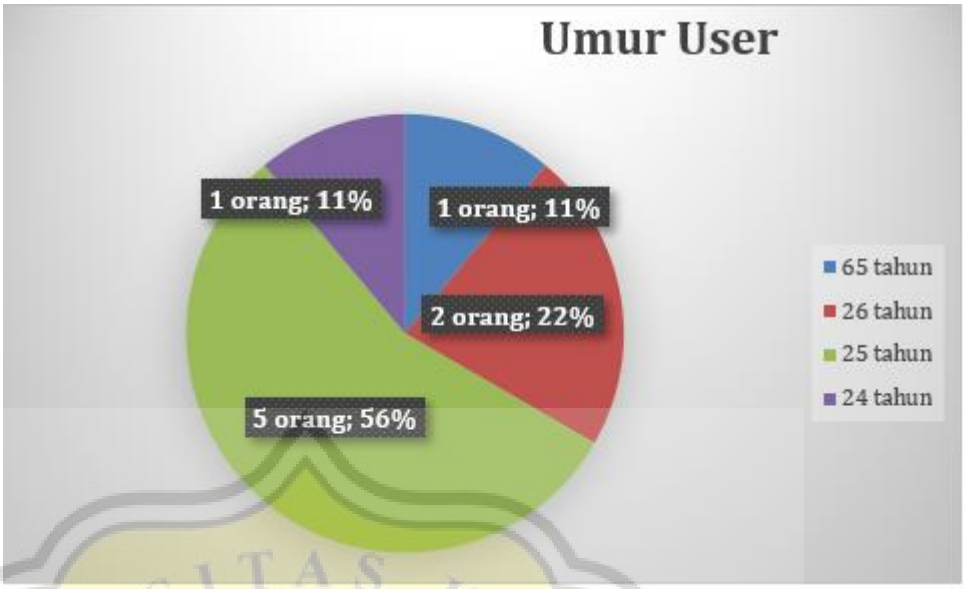
Gambar 4.62 merupakan script dimana table yang menampilkan artikel yang sudah pernah diupload oleh admin akan ditampilkan. Dan didalam table juga terdapat tombol add, edit, dan delete article untuk memunculkan popup edit dan add.

```
</div>
<table className="table mt-3">
  <thead>
    <tr>
      <th>Image</th>
      <th>Name</th>
      <th>Desc</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    {articles.map((item) => {
      return (
        <tr>
          <td>
            <img src={item.imageURL} height="80" width="80" />
          </td>

          <td>{item.name}</td>
          <td>{item.description}</td>
          <td>

            <FaEdit
              onClick={() => editArt(item)}
              color="blue"
              size={20}
            />
            <FaTrash
              color="red"
              size={20}
              onClick={() => {
                deleteArticle(item);
              }}
            />
```

Gambar 4.62 Script table Article Dashboard Admin

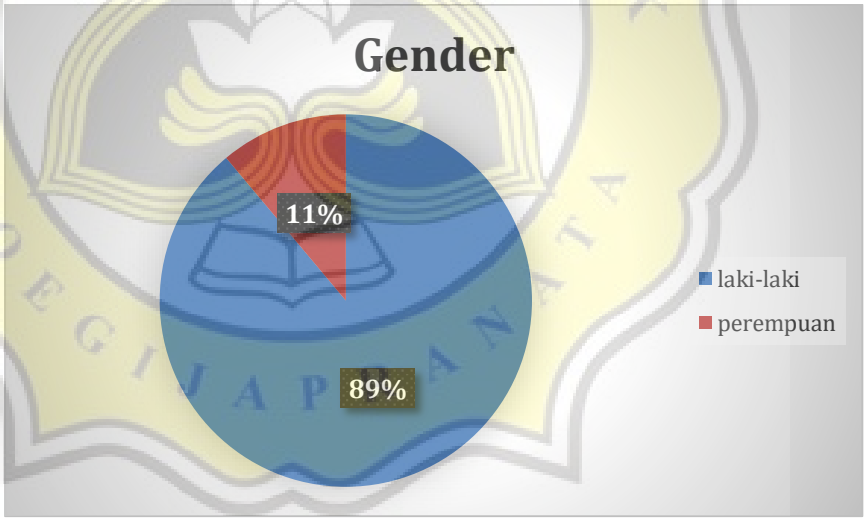
4.4 Hasil wawancara

Dari hasil wawancara yang dilakukan kepada beberapa user konsumen dan pemilik yang berumur rentan dari 24 tahun sampai 65 tahun dengan total 9 orang yang diperlihatkan dengan persentase oleh gambar dibawah ini



Gambar 4.63 Chart persentase umur user

Berdasarkan persentase gender dari user yang telah diwawancarai adalah sebagai gambar dibawah ini meliputi 8 orang laki-laki dan 1 orang perempuan.



Gambar 4.64 Chart persentase gender

Dari kedua gambar diatas semua user yang telah diwawancarai semua menyatakan respon positif terhadap 10 pertanyaan yang bersumber dari rumusan masalah. Hal yang mendapat respon positif mulai dari tampilan website dan fungsi fitur website, termasuk pemilik menyatakan website dapat