Bab IV Pembahasan Sistem Informasi Dan Aplikasi Android

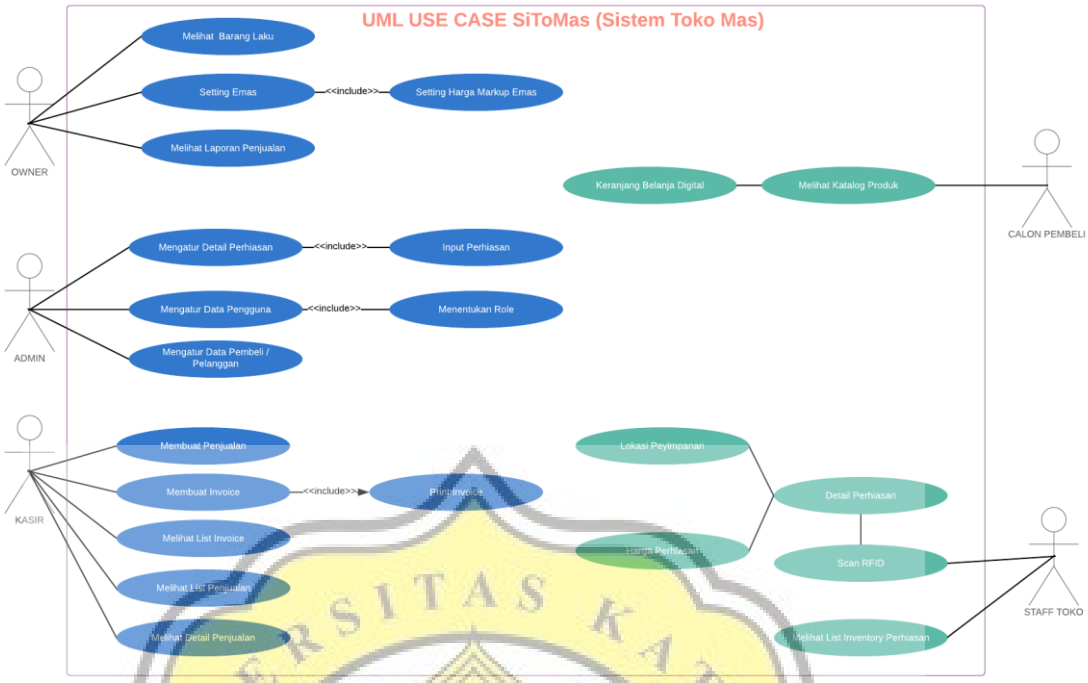
4.1 Perancangan Sistem Informasi & Aplikasi Android

Sistem Informasi & Aplikasi Android untuk toko perhiasan memiliki tujuan untuk membantu dan mengoptimalkan proses bisnis yang berjalan pada toko perhiasan. Keunggulan Sistem Informasi berbasis web adalah dapat diakses dimana saja dan kapanpun asalkan memiliki akses internet.

Dengan pemberian RFID (*Radio Frequency Identification)* pada tiap perhiasan, proses pencarian dan penyimpanan menjadi lebih mudah. Dan juga aplikasi Android untuk *staff* memudahkan dalam melakukan kegiatan sehari – hari di toko. Dari sisi calon konsumen juga bisa melihat katalog produk yang telah disediakan oleh toko perhiasan dan mengetahui harga secara langsung.

4.1.1. Perancangan Use Case Diagram

Langkah pertama dalam pembuatan Sistem Informasi & Aplikasi Android dimulai dengan merancang *use case diagram*. Hal ini dilakukan supaya memiliki gambaran keseluruhan dan cakupan apa saja yang ada di dalam sistem informasi & aplikasi nantinya. *Use case diagram* bisa dilihat pada gambar 4.1.

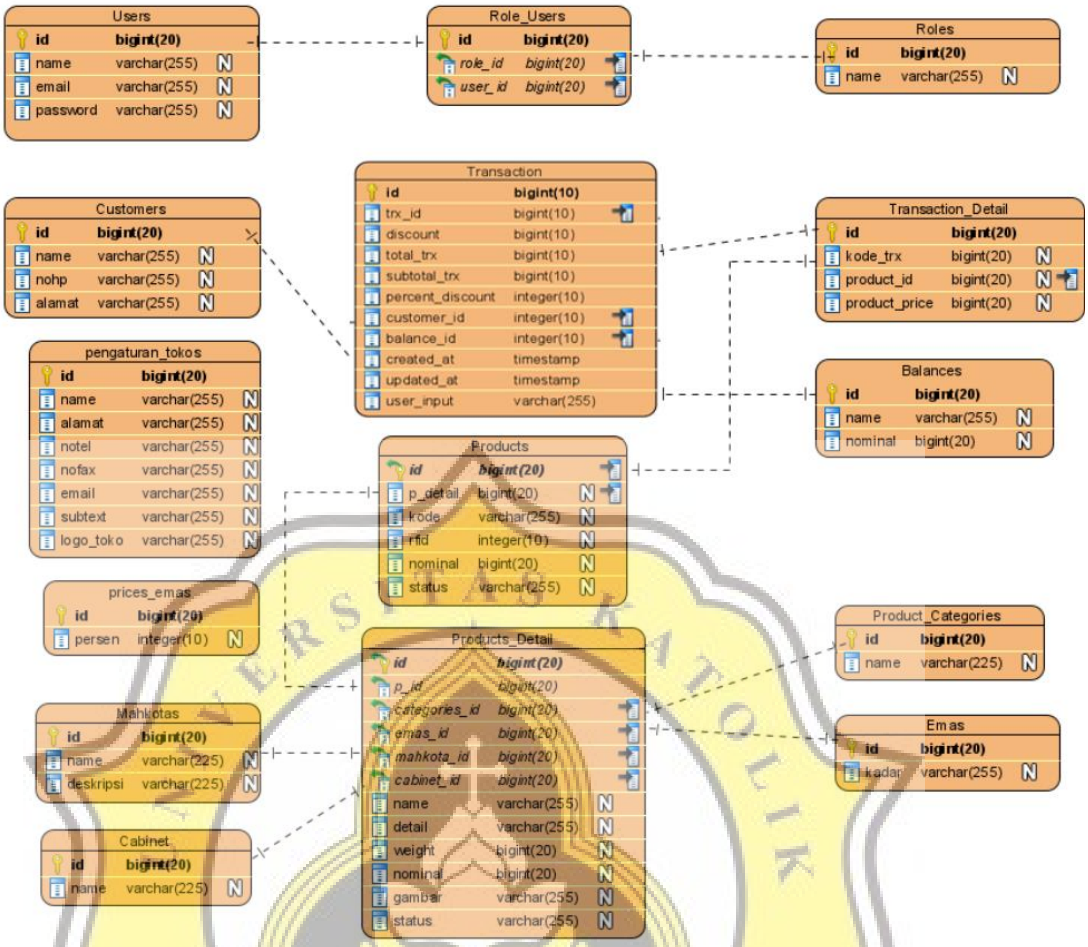


Gambar 4.1 UML Use Case Diagram

Use case diagram diatas kita dapat melihat ada 4 peran toko dan 1 peran calon pembeli. Tiap peran memiliki fitur – fitur yang berbeda dan memiliki hak akses masing – masing. Hal ini dilakukan agar adanya pembagian wewenang yang jelas dan terstruktur. Warna biru menggambarkan kegiatan yang dilakukan pada Sistem Informasi, sedangkan warna hijau kegiatan dilakukan pada aplikasi Android.

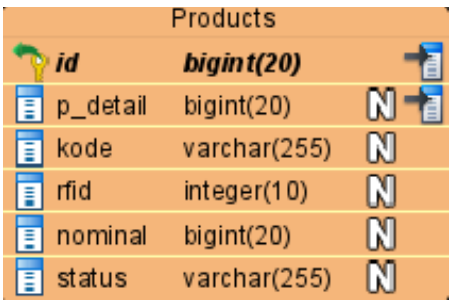
4.1.2. Perancangan Entity Relationship Diagram

ERD (*Entity Relationship Diagram*) merupakan tahapan lanjutan dalam proses perancangan. ERD memiliki tujuan menggambarkan relasi antar entitas data. Entitas digambarkan sebagai tabel data dalam *database*.



Gambar 4.2 Entity Relationship Diagram

Gambar 4.2 adalah ERD (Entity Relationship Diagram) dari Sistem Informasi toko perhiasan, bisa dilihat ada tabel yang memiliki relasi antar tabel lain, ada juga tabel yang dependen seperti tabel *price_emas* dan *pengaturan_tokos*.



Gambar 4.3 Detail Tabel Products

Gambar 4.3 diatas merupakan dari tabel "*Products",* tabel ini berisikan data perhiasan. Tabel ini menyimpan informasi utama produk perhiasan seperti kode perhiasan, rfid yang tercantum pada perhiasan, nominal dan status perhiasan. Kolom "*p_detail*" merupakan index yang merujuk pada tabel *Products_Detail.* Maka dari itu tabel ini memiliki relasi *one to one* dengan tabel *Products_Detail,* karena setiap perhiasan pasti memiliki 1 detail perhiasan.

Gambar 4.4 Detail Tabel Products_Detail

Gambar 4.4 diatas merupakan dari tabel *Products_Detail,* tabel ini mempunyai fungsi untuk menyimpan detail produk perhiasan yang di index oleh kolom "*p_detail*" pada tabel "*Products".* Kolom "*categories_id*", "*emas_id*", "*mahkota_id*", dan "*cabinet_id*" merupakan *index* dari masing – masing tabel dibawah ini.

Gambar 4.5 Detail Tabel Products

Gambar 4.4 diatas merupakan detail dari tabel "*Products_Categories"*, tabel ini berisikan kategori perhiasan. Tabel ini memiliki relasi *one to one*

dengan tabel "*Products_Detail"*. Yang berarti tiap perhiasan hanya mempunyai 1 kategori.

Gambar 4.6 Detail Tabel Emas

Gambar 4.5 diatas merupakan detail dari tabel "*Emas"*, tabel ini berisikan jenis emas atau kadar emas dalam *kr* (karat). Tabel ini memiliki relasi *one to one* dengan tabel "*Products_Detail"*. Yang berarti tiap perhiasan hanya mempunyai 1 jenis emas.

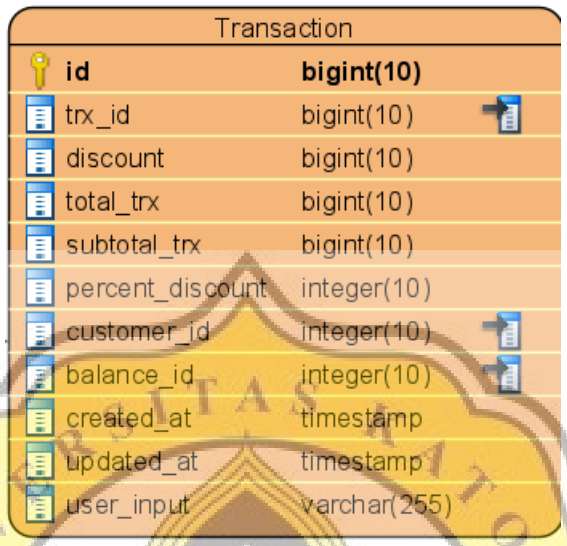
Gambar 4.7 Detail Tabel Mahkotas

Gambar 4.6 diatas merupakan detail dari tabel *Mahkotas*, tabel ini berisikan hiasan atau mahkota yang terdapat pada suatu perhiasan. Tabel ini memiliki relasi *one to one* dengan tabel "*Products_Detail"*. Yang berarti tiap perhiasan hanya mempunyai 1 jenis mahkota.

Gambar 4.8 Detail Tabel Cabinet

Gambar 4.7 diatas merupakan detail dari tabel "*Cabinet"*, tabel ini tempat penyimpanan atau lokasi yang nanti akan diberikan ke tiap perhiasan. Tabel

ini memiliki relasi *one to one* dengan tabel "*Products_Detail"*. Yang berarti tiap perhiasan hanya mempunyai 1 tempat peyimpanan.

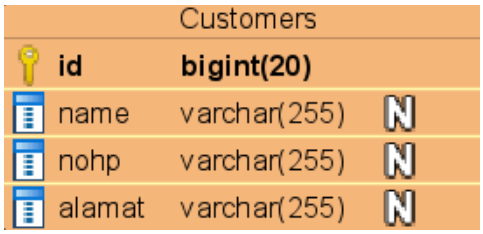


Gambar 4.9 Detail Tabel Transactions

Gambar 4.8 diatas merupakan detail dari tabel *Transactions*, tabel ini berisikan data detail transaksi. Tabel ini memiliki relasi one to one ke tabel *transaksi, customers, dan balance. Dan relasi one to many* ke tabel "*Products"*. Jadi transaksi lebih dari 1 perhiasan, bisa dilakukan dengan 1 nota / *invoice*, pada 1 customer, dan menggunakan 1 metode pembayaran.

Gambar 4.10 Detail Tabel Transaction_Details

Gambar 4.9 diatas merupakan detail dari tabel "*Transaction_Detail",* memiliki relasi one to one dengan tabel *transaction*.

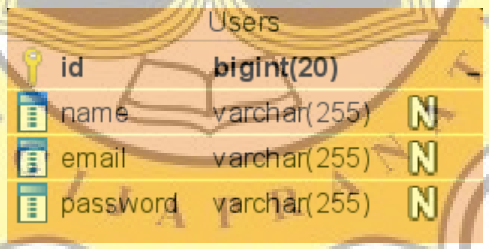


Gambar 4.11 Detail Tabel Customers

Gambar 4.10 diatas merupakan detail dari tabel "*Customers",* memiliki relasi one to one dengan tabel "*transaction"*. Dalam 1 invoice hanya bisa menamakan 1 pelanggan saja.

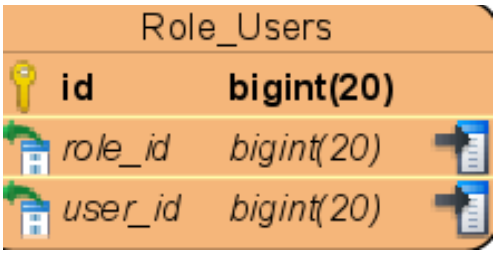
Gambar 4.12 Detail Tabel Balances

Gambar 4.11 diatas merupakan detail dari tabel "*Balances",* memiliki relasi one to one dengan tabel "*transaction"*. Dalam 1 invoice hanya bisa memakai 1 metode pembayaran.



Gambar 4.12 Detail Tabel Users

Gambar 4.12 diatas merupakan detail dari tabel "*Users",* memiliki relasi *one to many* dengan tabel "*roles_user"*. 1 *user* bisa memiliki role lebih dari 1, sebagai contoh pemilik bisa juga memegang *role* sebagai kasir dan auditor.



Gambar 4.13 Detail Tabel Role_Users

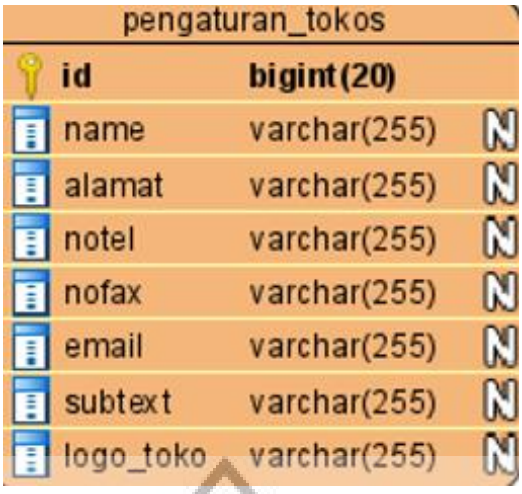
Gambar 4.14 diatas merupakan detail dari tabel "*Role_Users"*, memiliki relasi one to many dengan tabel "*Users"*. Tabel ini menampung role user yang sudah di set oleh admin, dan 1 *user* bisa memiliki 2 *roles* atau lebih.

Gambar 4.14 Detail Tabel Roles

Gambar 4.14 diatas merupakan detail dari tabel "*roles"*, memiliki relasi one to one dengan tabel "*roles_user"*. Tabel ini memiliki fungsi memberikan nama kepada kode role di tabel "*role_users"*.

Gambar 4.15 Detail Tabel prices_emas

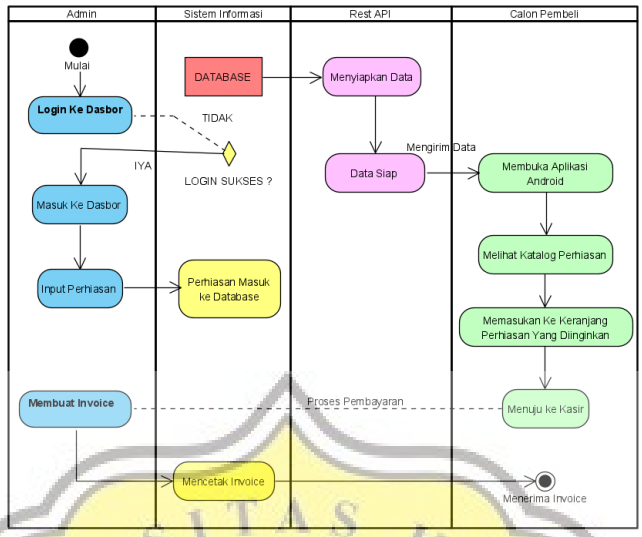
Gambar 4.15 diatas merupakan detail dari tabel "*prices_emas"*, fungsi tabel ini adalah menyimpan persentase harga *markup* emas yang disetting oleh owner atau pemilik toko.



Gambar 4.16 Detail Tabel pengaturan_tokos

Gambar 4.16 diatas merupakan detail dari tabel "*pengaturan_tokos"*, tabel ini sebagai peyimpanan konfigurasi tampilan, detail seperti, nama toko, alamat, email , dan logo akan diambil untuk kostumisasi pada pembuatan invoice & tampilan pada dasbor.

4.1.3. Perancangan Swimlane Flowchart Proses Bisnis

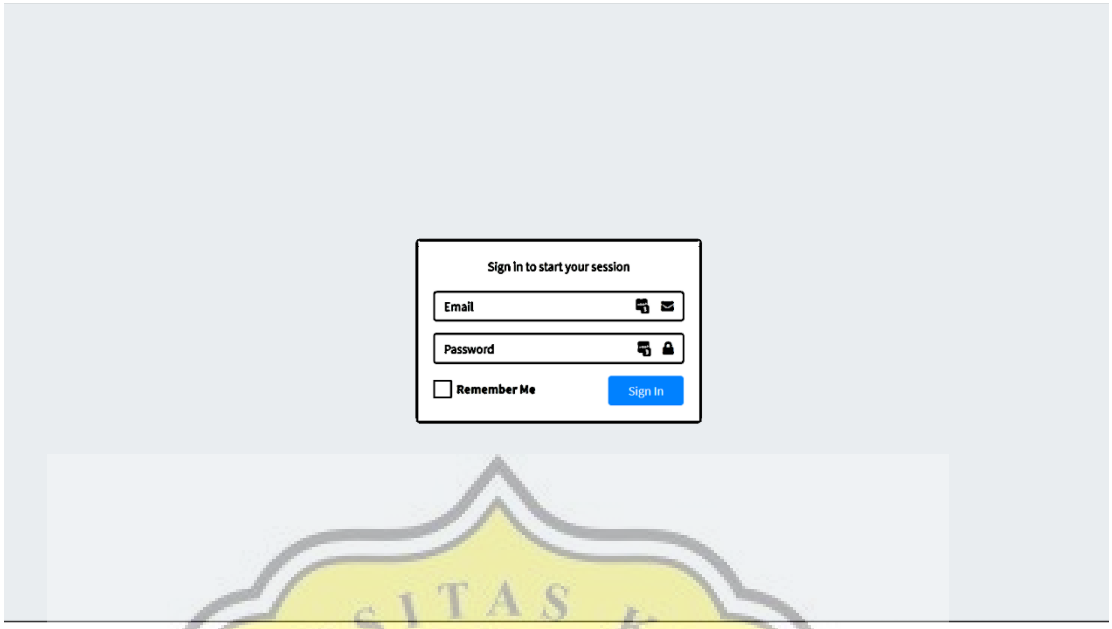


Gambar 4.17 Swimlane Proses Bisnis

Tahapan setelah mendesain *use case diagram & ERD (Entity Relationship Diagram)* dilanjutkan dengan pembuatan diagram *swimlane* proses bisnis. Dengan adanya *swimlane* proses bisnis dapat memberikan gambaran secara sederhana alur kerja sistem informasi & aplikasi Android.

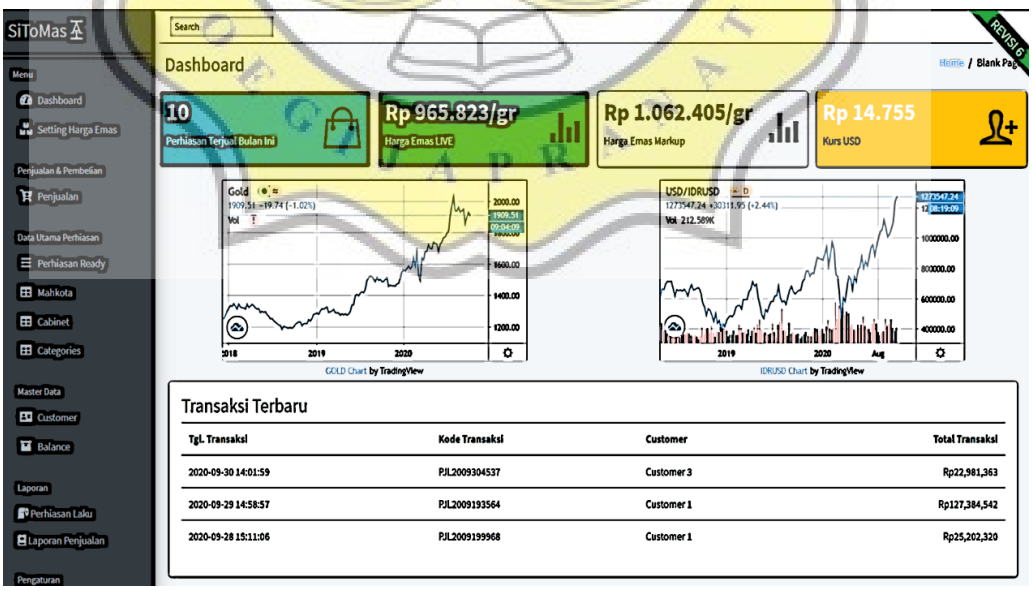
4.1.4. Perancangan Desain Sistem Informasi

Tahap selanjutnya adalah menrancang tampilan antarmuka untuk sistem informasi dan aplikasi Android. Tampilan awal saat membuka sistem informasi bisa dilihat pada Gambar 4.18. Pertama kali pengguna akan diminta *login* menggunakan akun yang sudah disediakan oleh admin / pemilik toko perhiasan.



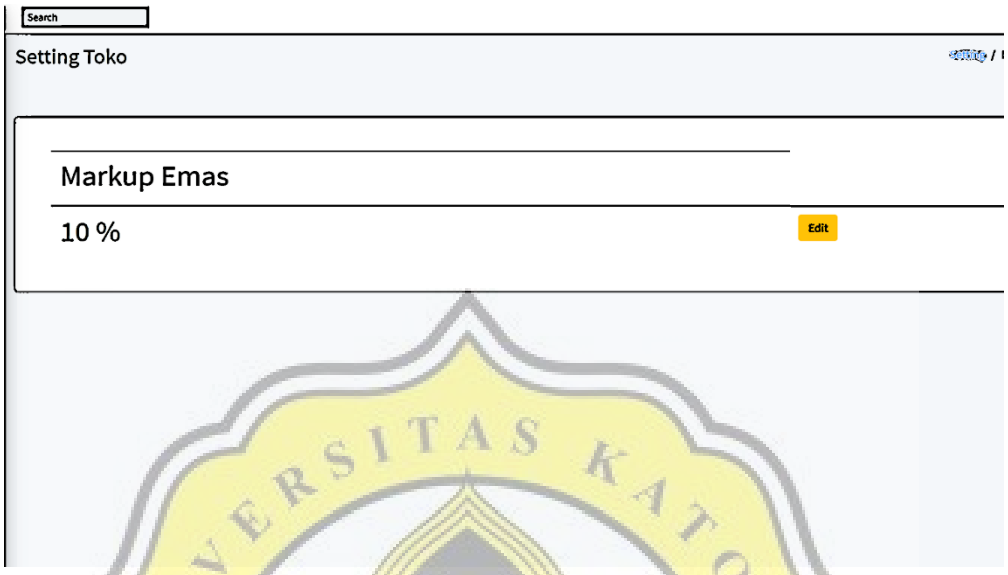
Gambar 4.19 Desain Halaman Login

Setelah berhasil melakukan *login*, pengguna otomatis diarahkan ke dasbor atau menu utama, jika tidak berhasil melakukan *login* pengguna akan dikembalikan ke halaman yang sama. Desain tampilan dasbor bisa dilihat pada Gambar 4.20 dibawah.



Gambar 4.20 Desain Halaman Dasbor

Gambar 4.21 merupakan desain dimana pengguna yang mempunyai *role* yang sesuai bisa menyetel harga emas *mark up*.

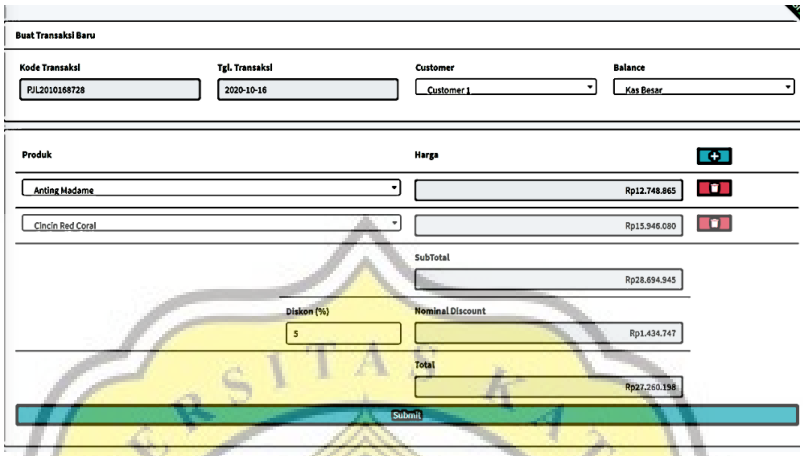


Gambar 4.21 Desain Halaman Setting Mark up Emas

Pengguna bisa melihat daftar transaksi yang telah dilakukan pada halaman penjualan, tampilan awal halaman penjualan adalah tabel data yang berisi list transaksi. Desain halaman penjualan bisa dilihat pada Gambar 4.22

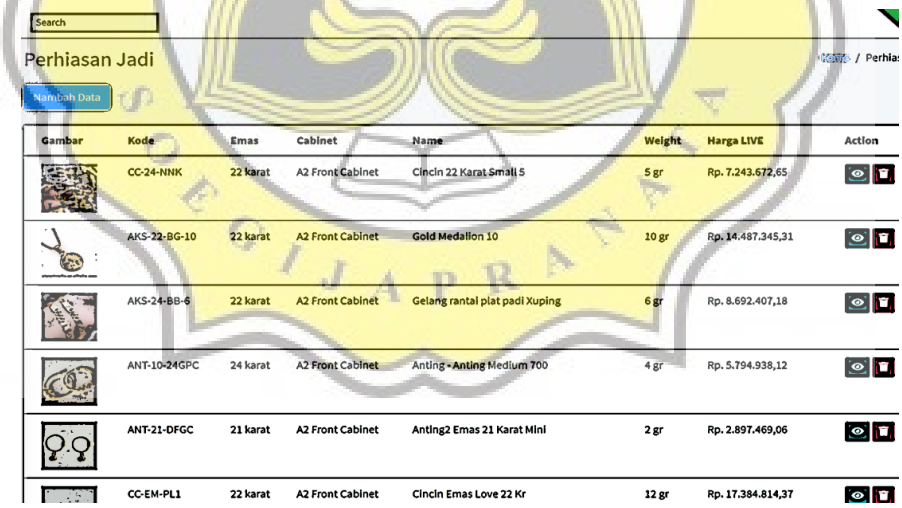
Gambar 4.22 Desain Halaman List Transaksi

Jika ingin membuat transaksi bisa menekan tombol buat transaksi Setelah pengguna menekan tombol tambah data, maka diarahkan ke halaman pembuatan nota atau *invoice*. Desain halaman pembuatan *invoice* bisa dilihat pada Gambar 4.23



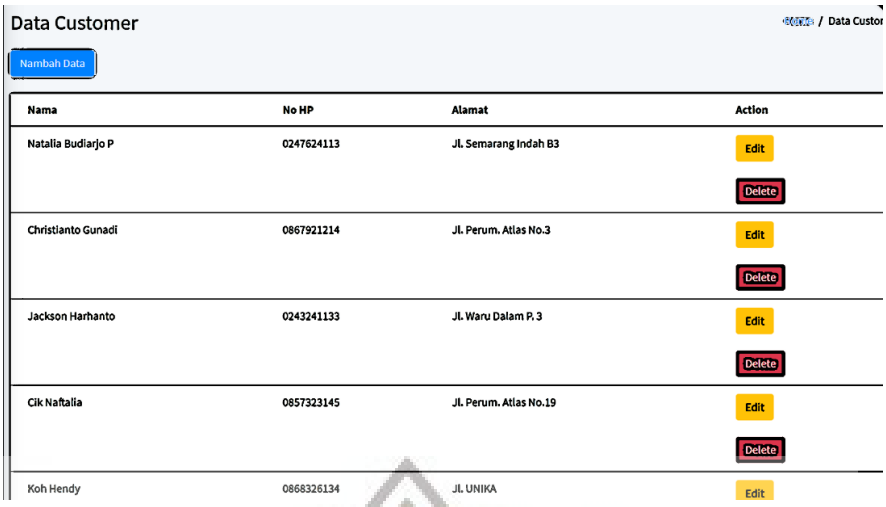
Gambar 4.23 Desain Halaman Pembuatan Invoice

Gambar 4.24 adalah desain dimana pengguna bisa melihat data perhiasan yang sudah diinputkan oleh user lain atau pengguna itu sendiri.



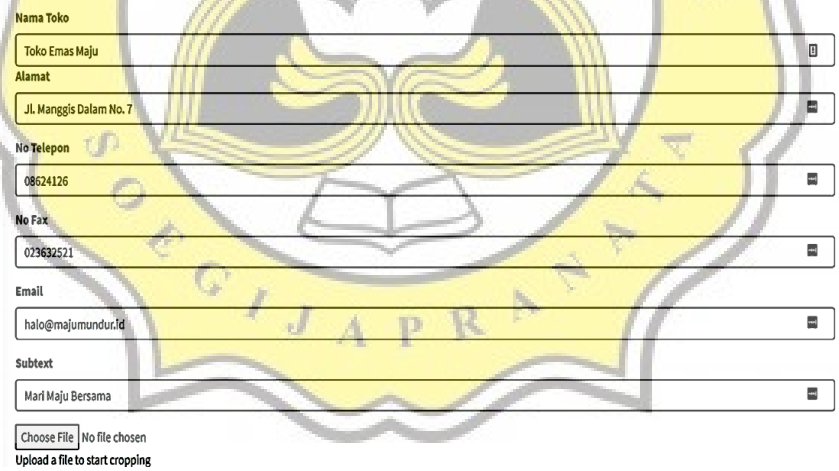
Gambar 4.25 Desain Halaman Perhiasan

Desain halaman customer bisa dilihat pada Gambar 4.26 menampilkan data customer yang ada, data customer ini akan dipakai untuk melakukan proses pembuatan nota.



Gambar 4.26 Desain Halaman Perhiasan

Desain halaman *setting* toko bisa dilihat pada Gambar 4.27. Menu ini menampilkan detail informasi toko perhiasan. Data ini akan dipakai untuk menampilkan identitas toko pada nota penjualan dan laporan penjualan.



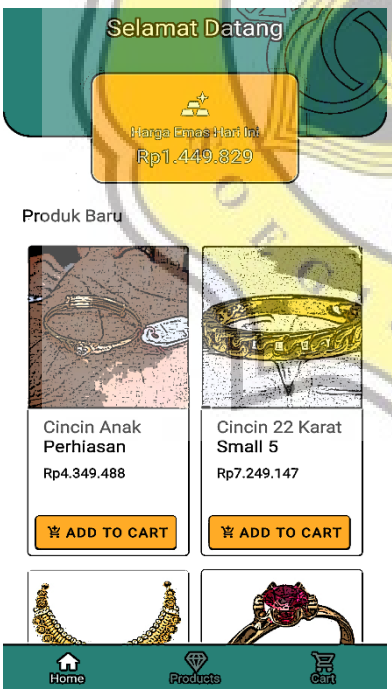
Gambar 4.27 Desain Halaman Setting Toko

Gambar 4.28 merupakan desain dari halaman *setting user role*, halaman ini memiliki fungsi memberikan *role* kepada pengguna.

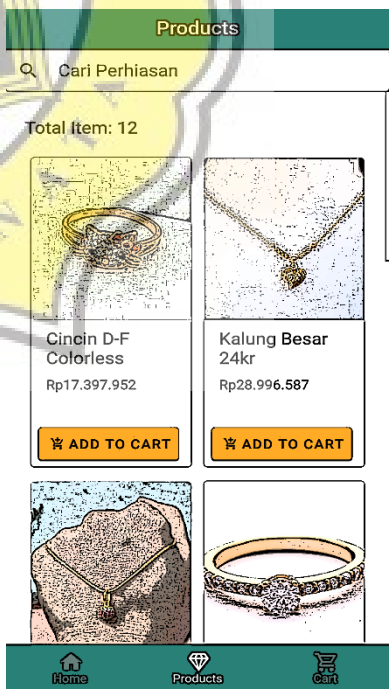
Gambar 4.28 Desain Halaman Setting Toko

4.1.5. Perancangan Desain Aplikasi Android

Berikut adalah desain aplikasi Android untuk toko perhiasan, pada Gambar 4.29 merupakan *home*, menu pertama kali yang akan muncul saat membuka aplikasi Android. Data yang dimunculkan pada halaman ini adalah harga emas terkini dan 4 perhiasan terbaru dengan format 2 x 2 *grid*.



Gambar 4.29 Desain Home

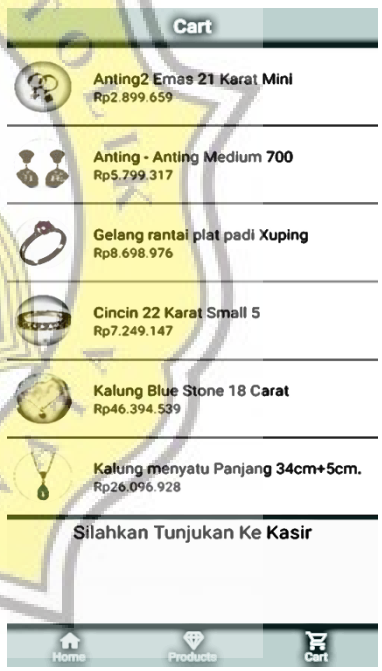


Gambar 4.30 Desain Products

Desain halaman *products* bisa dilihat pada Gambar 4.30, halaman ini menampilan semua perhiasan yang ada dengan format tampilan *grid.* Jika pengguna ingin melihat lebih detail perhiasan, pengguna bisa menyentuh pada gambar perhiasan yang diinginkan.

Setelah pengguna menyentuh gambar perhiasan, desain tampilan pada Gambar 4.30 akan muncul. Pengguna bisa melihat detail perhiasan dengan gambar yang lebih besar. Jika pengguna menyentuh tombol *add to cart* maka perhiasan itu akan muncul di halaman *cart*. Desain halaman *cart* bisa dilihat pada Gambar 4.31.

Gambar 4.30 Desain Products



Gambar 4.31 Desain Cart

4.2 Pembuatan Sistem Informasi

Sistem Informasi yang dibuat menggunakan PHP Framework Laravel versi 7 dengan Bootstrap 4 & Admin LTE Dashboard. Aplikasi Android dibuat menggunakan react-native. Untuk mendapatkan data harga emas secara live menggunakan *scripting language* Python.

4.2.1 Script Python

Script Python dipakai untuk scraping data dari yahoo finance. Ada 2 data yang diambil menggunakan script ini. Data harga emas per *troy ons* dalam US Dollar dan data kurs US Dollar (USD) terhadap Indonesia Rupiah (IDR).

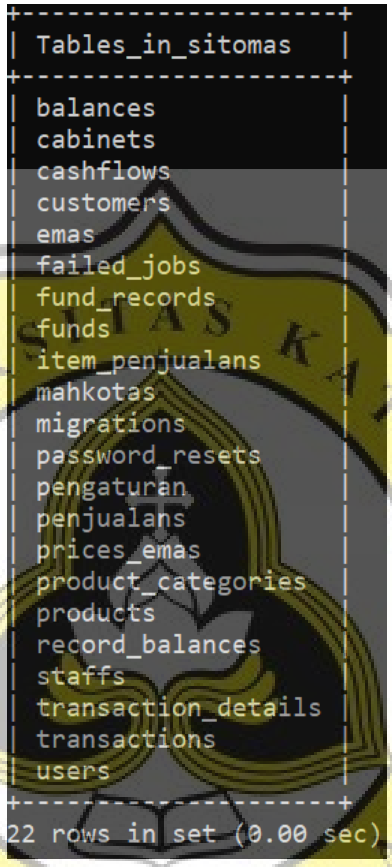
```
weekend = 1,6,7

    data = yfinance.Ticker("GC=F" , “USDIDR=X")

    now = datetime.datetime.now()
    if now == weekend:
        data_rcv = data.history(period="3d")
    else:
        data_rcv = data.history(period="0d")


@api.route('/gold', methods=['GET'])
def get_gold():

    print(data_rcv)
    json_parsed = json.loads(data_rcv.to_json(index=True, orient='table',date_format='iso')
)['data']

    open_value = {
        "Open":json_parsed[0]['High']
    }

    return json.dumps(open_value)
```

Cuplikan kode diatas memiliki output berupa json dengan data harga emas *per troy ons*. Hal yang penting lainnya adalah mendefinisikan jadwal dimana *trading* tidak buka, supaya hasil tidak membalikan hasil *NULL*. Cuplikan kode di bawah sama seperti diatas hanya memberikan output kurs US Dollar terhadap Indonesia Rupiah.

```
@api.route('/currency', methods=['GET'])
def get_currency():

    data = yfinance.Ticker("USDIDR=X")
    data_rcv = data.history(period="1d")
    print(data_rcv)
    json_parsed = json.loads(data_rcv.to_json(index=True, orient='table',date_format='iso'
))['data']

    open_value = {
        "Open":json_parsed[0]['High']
    }

    return json.dumps(open_value)
```

4.2.2 Arsitektur Database

Arsitektur database yang dipakai menggunakan *single database model*, dengan *engine* MariaDB. Database ini akan diakses oleh Sistem Informasi untuk menyimpan data. Daftar tabel yang dipakai bisa dilihat pada Gambar 4. 32.



```
+---------------------+
| Tables_in_sitomas   |
+---------------------+
| balances            |
| cabinets            |
| cashflows           |
| customers           |
| emas                |
| failed_jobs         |
| fund_records        |
| funds               |
| item_penjualans     |
| mahkotas            |
| migrations          |
| password_resets     |
| pengaturan          |
| penjualans          |
| prices_emas         |
| product_categories  |
| products            |
| record_balances     |
| staffs              |
| transaction_details |
| transactions        |
| users               |
+---------------------+
22 rows in set (0.00 sec)
```

Gambar 4.32 Desain Tabel Database

4.2.3 Halaman Login Sistem Informasi

Cuplikan kode berikut berfungsi untuk melakukan *login* bagi pengguna yang belum *login*, cuplikan kode bisa dilihat pada Gambar. Untuk menampilkan antarmuka untuk halaman *login*, cuplikan kode antarmuka halaman *login* bisa dilihat pada Gambar 4.34.

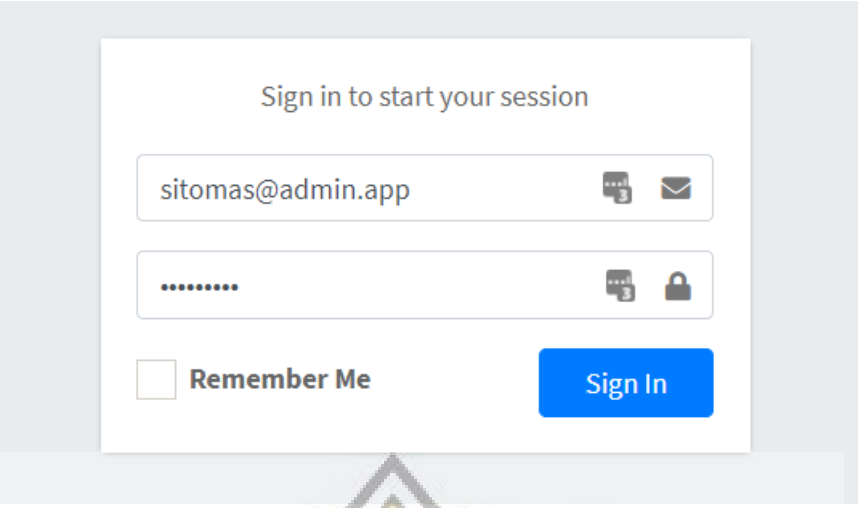
```
class LoginController extends Controller
{

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
```

```
     */
    protected $redirectTo = RouteServiceProvider::HOME;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Gambar 4.33 Kode Login

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Log in</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Font Awesome -->
  <link rel="stylesheet" href="{{asset('adminlte/plugins/fontawesome-
free/css/all.min.css')}}">
  <!-- Ionicons -->
  <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons
.min.css">
  <!-- icheck bootstrap -->
  <link rel="stylesheet" href="{{asset('adminlte/plugins/icheck-bootstrap/icheck-
bootstrap.min.css')}}">
  <!-- Theme style -->
  <link rel="stylesheet" href="{{asset('adminlte/dist/css/adminlte.min.css')}}">
  <!-- Google Font: Source Sans Pro -->
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" re
l="stylesheet">
</head>
<body class="hold-transition login-page">
    @yield('content')

<!-- jQuery -->
<script src="{{asset('adminlte/plugins/jquery/jquery.min.js')}}"></script>
<!-- Bootstrap 4 -->
<script src="{{asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js')}}"></script>
<!-- AdminLTE App -->
<script src="{{asset('adminlte/dist/js/adminlte.min.js')}}"></script>

</body>
</html>
```

Gambar 4.34 Kode UI Login



Gambar 4.34 Hasil Tampilan Login

4.2.4 Halaman Utama Dasbor

Setelah pengguna berhasil menggunakan *login*, makan akan diarahkan ke halaman utama dashboard atau dalam kode ini disebut sebagai *HOME*. Halaman utama memiliki tampilan informasi sederhana yang menyeluruh. Cuplikan kode untuk halaman utama dasbor bisa dilihat pada Gambar 4.36.

```
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {

        $client = new \GuzzleHttp\Client();

        // Create a request
        $requestEmas = $client->get('https://api.sitomas.my.id/gold');
        $requestUSD = $client->get('https://api.sitomas.my.id/currency');

        // Get the actual response without headers
        $responseEmas = $requestEmas->getBody();
        $responseEmas = json_decode($responseEmas, true);
```

```
        $responseUSD = $requestUSD->getBody();
        $responseUSD = json_decode($responseUSD, true);

        $hargaemas  = $responseEmas['Open'];
        $usd        = $responseUSD['Open'];

        $data = Transaction::orderBy('id')->get();
        $countpenjualan = $data->count();

        function convert_to_rupiah($angka)
        {
            return 'Rp' . strrev(implode('.', str_split(strrev(strval($angka)), 3)));
        }

        $items = Transaction::with('customer', 'customer.transactions')-
>orderBy('created_at', 'DESC')->limit(3)->get();

        $persen = PricesEmas::first()->value('persen');

        $goldprice = $hargaemas * $usd / 29.3;
        $markupprice = $goldprice * $persen / 100;

        return view('home')-
>with(compact('persen','countpenjualan','goldprice','usd','markupprice','items'));
    }
}
```

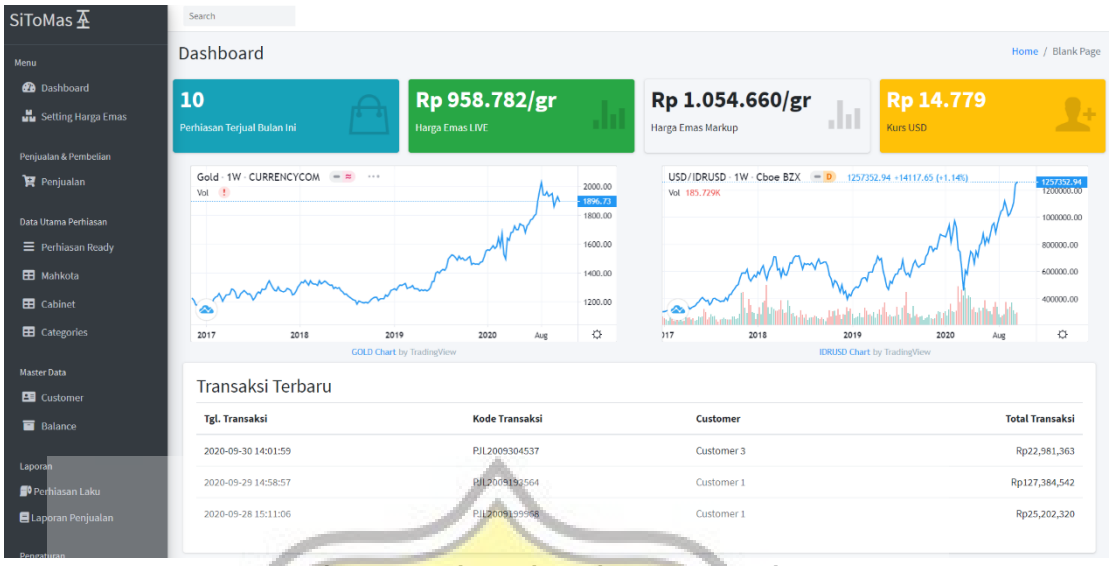
Gambar 4.36 Kode Halaman Utama Dasbor

```
@section('content')
<div class="row">
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-info">
              <div class="inner">
                <h3>{{$countpenjualan}}</h3>
                <p>Perhiasan Terjual Bulan Ini</p>
              </div>
              <div class="icon">
                <i class="ion ion-bag"></i>
              </div>
            </div>
          </div>
          <!-- ./col -->
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-success">
              <div class="inner">
                <h3>Rp {{number_format($goldprice, 0, ',', '.')}}/gr<sup style="font-
size: 10px"></sup></h3>
                <p>Harga Emas LIVE</p>
              </div>
              <div class="icon">
                <i class="ion ion-stats-bars"></i>
              </div>
            </div>
          </div>

          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-fail">
              <div class="inner">
```

```
                <h3>Rp {{number_format($goldprice + $markupprice, 0, ',', '.')}}/gr<sup st
yle="font-size: 10px"></sup></h3>
                <p>Harga Emas Markup</p>
              </div>
              <div class="icon">
                <i class="ion ion-stats-bars"></i>
              </div>
            </div>
          </div>

          <!-- ./col -->
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-warning">
              <div class="inner">
                <h3 class="text-white">Rp {{number_format($usd, 0, ',', '.')}}</h3>
                <p>Kurs USD</p>
              </div>
              <div class="icon">
                <i class="ion ion-person-add"></i>
              </div>
            </div>
          </div>
  </div>
<div class="row">
<!-- TradingView Widget BEGIN -->
  <div class="col">
    <div class="tradingview-widget-container">
          <div id="tradingview_cdf91"></div>
          <div class="tradingview-widget-
copyright"><a href="https://www.tradingview.com/symbols/CURRENCYCOM-
GOLD/" rel="noopener" target="_blank"><span class="blue-
text">GOLD Chart</span></a> by TradingView</div>
          <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>
          <script type="text/javascript">
          new TradingView.widget(
          {
          "width": 650,
          "height": 300,
          "symbol": "CURRENCYCOM:GOLD",
          "interval": "W",
          "timezone": "Asia/Bangkok",
          "theme": "light",
          "style": "2",
          "locale": "en",
          "toolbar_bg": "#f1f3f6",
          "enable_publishing": true,
          "hide_top_toolbar": true,
          "save_image": false,
          "container_id": "tradingview_cdf91"
        }
          );
          </script>
      </div>
  </div>
<!-- TradingView Widget END -->

<!-- TradingView Widget BEGIN -->
  <div class="col">
    <div class="tradingview-widget-container">
        <div id="tradingview_de783"></div>
        <div class="tradingview-widget-
copyright"><a href="https://www.tradingview.com/" rel="noopener" target="_blank"><span cla
ss="blue-text">IDRUSD Chart</span></a> by TradingView</div>
        <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>
        <script type="text/javascript">
```

```
        new TradingView.widget(
        {
        "width": 650,
        "height": 300,
        "symbol": "USD/FX_IDC:IDRUSD",
        "interval": "W",
        "timezone": "Asia/Bangkok",
        "theme": "light",
        "style": "2",
        "locale": "en",
        "toolbar_bg": "#f1f3f6",
        "enable_publishing": true,
        "hide_top_toolbar": true,
        "save_image": false,
        "container_id": "tradingview_de783"
      }
        );
        </script>
    </div>
  </div>
</div>
<!-- TradingView Widget END -->
  <div class="container-fluid">
    <div class="orders">
          <div class="col-12">
              <div class="card">
                  <div class="card-body">
                    <span><h3>Transaksi Terbaru</h3></span>
                      <div class="table-responsive order-table ov-h">
                          <table class="table">
                              <thead>
                                  <tr>
                                      <th>Tgl. Transaksi</th>
                                      <th>Kode Transaksi</th>
                                      <th>Customer</th>
                                      <th class="text-right">Total Transaksi</th>
                                  </tr>
                              </thead>
                              <tbody>
                                  @forelse ($items as $item)
                                  <tr>
                                      <td>{{$item->created_at}}</td>
                                      <td>{{$item->kode_trx}}</td>
                                      <td>{{$item->customer->name}}</td>
                                      <td class="text-right">Rp{{ number_format($item-
>total_trx) }}</td>
                                  </tr>
                                  @empty
                                  @endforelse
                              </tbody>
                          </table>
                      </div>
                  </div>
              </div>
          </div>
    </div>
  </div>
@endsection
```

Gambar 4.37 Kode UI Halaman Utama Dasbor



Gambar 4.38 Hasil Tampilan Halaman Utama Dasbor

4.2.5 Halaman Setting Harga Emas

Cuplikan kode dibawah, mempunyai untuk mengkonfigurasi harga emas sesuai keinginan pengguna. Hanya pengguna tertentu yang memili akses ke halaman ini. Dua fungsi utama merupakan index & update. Cuplikan kode bisa dilihat pada Gambar 4.39 dan Gambar 4.40.

```
class PricesEmasController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

        /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = PricesEmas::orderBy('id')->get();

        return view('data.pricesemas.index')->with(compact('data'));
    }

        /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        try {
            $response = PricesEmas::insert($request->except(['_token']));
```

```
            return redirect()->back()->with(['messages'=>'Data telah ter input!']);

        } catch (Throwable $e) {
            return redirect()->back()->with(compact('e'));
        }

    }

        /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $pricesemas = PricesEmas::orderBy('id')->get();
        return view('data.pricesemas.form')->with(compact('pricesemas'));
    }

    public function edit($id)
    {

        $pricesemas = PricesEmas::find($id);
        return view('data.pricesemas.form')->with(compact('pricesemas'));
     }
}
```

Gambar 4.39 Kode Halaman Setting Emas

```
@section('content')
    <div class="container-fluid">
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Test Echo data</h5>

        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
      <br/>
      <br/>
        <div class="row">
          <div class="col-12">
            <div class="card">
              <div class="card-body table-responsive p-5" style="height: 250px;">
                <table class="table table-head-fixed text-nowrap">
                  <thead>
                    <tr>
                      <th><h1>Markup Emas</h1></th>
                    </tr>
                  </thead>
                  <tbody>
                    @foreach($data as $row)
                      <tr>
                        <td><h1>{{$row->persen}} %</h1></td>
                        <td>
                          <a href="{{route('pricesemas.edit',$row->id)}}" class="btn btn-
warning">Edit</a><br/><br/>
                            @csrf
                          </form>
                        </td>
```

```
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
    <!-- /.card-body -->
</div>
<!-- /.card -->
</div>
</div>
</div>
@endsection
```

Gambar 4.40 Kode UI Halaman Setting Emas

Gambar 4.41 Hasil Halaman Setting Emas

4.2.6 Halaman Daftar Penjualan

Halaman Penjualan memiliki 2 halaman, halaman daftar penjualan & halaman pembuatan nota / *invoice*. Saat pengguna pertama kali pergi ke halaman penjualan. Maka tampilan pertama adalah halaman daftar penjualan. Cuplikan kode bisa dilihat pada Gambar 4.42 dan Gambar 4.43.

```
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        function convert_to_rupiah($angka)
        {
            return 'Rp' . strrev(implode('.', str_split(strrev(strval($angka)), 3)));
        }
        $items = Transaction::with('customer', 'customer.transactions')-
>orderBy('created_at', 'DESC')->get();
```

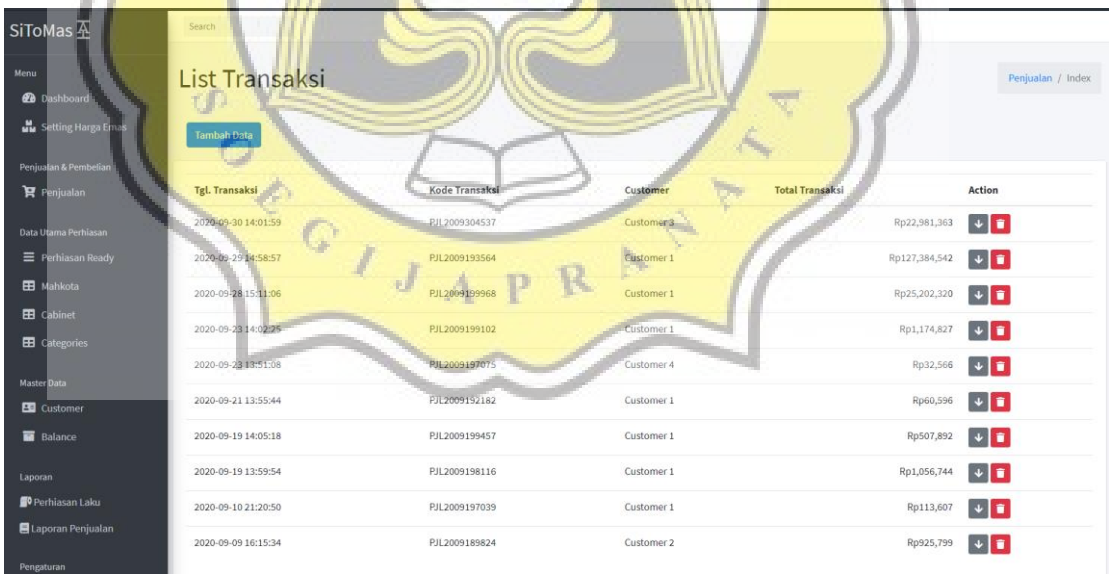
```
        // dd($items);
        return view('data.transaksi.index')->with([
            'items' => $items
        ]);
    }
```

Gambar 4.42 Kode Halaman Daftar Penjualan

```
@section('content')
<div class="container-fluid">
    <div class="row mb-2">
      <div class="col-sm-6">
        <h1>List Transaksi</h1>
      </div>
      <div class="col-sm-6">
        <ol class="breadcrumb float-sm-right">
          <li class="breadcrumb-item"><a href="#">Penjualan</a></li>
          <li class="breadcrumb-item active">Index</li>
        </ol>
      </div>
    </div>
</div>

  <div class="card-body">
    <a href="{{ route('transaksi.create') }}" class="btn btn-primary float-left">
        Tambah Data
    </a>
</div>

<div class="orders">
    <div class="row">
        <div class="col-12">
            <div class="card">
                <div class="card-body">
                    <div class="table-responsive order-table ov-h">
                        <table class="table">
                            <thead>
                                <tr>
                                    <th>Tgl. Transaksi</th>
                                    <th>Kode Transaksi</th>
                                    <th>Customer</th>
                                    <th>Total Transaksi</th>
                                    <th>Action</th>
                                </tr>
                            </thead>
                            <tbody>
                                @forelse ($items as $item)
                                <tr>
                                    <td>{{$item->created_at}}</td>
                                    <td>{{$item->kode_trx}}</td>
                                    <td>{{$item->customer->name}}</td>
                                    <td class="text-right">Rp{{ number_format($item-
>total_trx) }}</td>
                                    <td>
                                        <a target="_blank" href="{{ route('transaksi.print
', $item->id) }}"
                                            class="btn btn-secondary btn-sm">
                                            <i class="fa fa-arrow-down"></i>
                                        </a>
                                        {{-
- <a href="#" value="{{ route('transaksi.show',$item->id) }}"
                                            class="btn btn-sm btn-info modalMd" data-
toggle="modal"
                                            data-target="#modalMd"><i class="fa fa-
eye"></i>
```

```
                                    </a> --}}
                                    <form action="{{ route('transaksi.destroy',$item-
>id) }}" method="POST"
                                        class="d-inline">
                                        @csrf
                                        @method('delete')
                                        <button class="btn btn-danger btn-sm">
                                            <i class="fa fa-trash"></i>
                                        </button>
                                    </form>
                                </td>
                            </tr>
                            @empty
                            <tr>
                                <td colspan="7">
                                    <div class="alert alert-secondary" role="alert">
                                        Data Tidak Ditemukan!
                                    </div>
                                </td>
                            </tr>
                            @endforelse
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>

@endsection
```

Gambar 4.43 Kode UI Halaman Daftar Penjualan



Gambar 4.44 Hasil Halaman Daftar Penjualan

4.2.7 Halaman Perhiasan

Pengguna bisa melihat perhiasan – perhiasan yang tersedia pada toko emas melalui halaman perhiasan. Halaman ini akan mendaftar perhiasan serta detailnya. Untuk menambahkan data / *input* perhiasan bisa menekan tombol "tambah data". Halaman perhiasan memilih *sub-menu* seperti : mahkota, cabinet, categories. Cuplikan kode halaman perhiasan beserta submenu bisa dilihat pada gambar dibawah.

```
class ProductController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $client = new \GuzzleHttp\Client();

        // Create a request
        $requestEmas = $client->get('https://api.sitomas.my.id/gold');
        $requestUSD = $client->get('https://api.sitomas.my.id/currency');

        // Get the actual response without headers
        $responseEmas = $requestEmas->getBody();
        $responseEmas = json_decode($responseEmas, true);

        $responseUSD = $requestUSD->getBody();
        $responseUSD = json_decode($responseUSD, true);

        $hargaemas  = $responseEmas['Open'];
        $usd        = $responseUSD['Open'];

        $data = Product::orderBy('id','desc')->get()->where('status', 1);

        $persen = PricesEmas::first()->value('persen');

        $goldprice = $hargaemas * $usd / 29.3;
        $markupprice = $goldprice * $persen / 100;
        $markupgoldprice = $goldprice + $markupprice;

        return view('data.product.index')->with(compact('data','markupgoldprice'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
```

```
    public function create()
    {
        $categories = Categories::orderBy('name')->get();
        $emas       = Emas::orderBy('name')->get();
        $mahkota    = Mahkota::orderBy('name')->get();
        $cabinet    = Cabinet::orderBy('name')->get();


        return view('data.product.form')-
>with(compact('categories','emas','mahkota','cabinet'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        try {
            if (preg_match('/^data:image\/(\w+);base64,/', $request->gambare)) {
                $data = substr($request->gambare, strpos($request->gambare, ',') + 1);

                $data = base64_decode($data);
                $name = date('mdYHis').".png";
                $path = Storage::disk('product')->put($name, $data);
            }

            $response = new Product;
            $response->kode = $request->kode;
            $response->categories_id = $request->categories_id;
            $response->emas_id = $request->emas_id;
            $response->mahkota_id = $request->mahkota_id;
            $response->cabinet_id = $request->cabinet_id;
            $response->name = $request->name;
            $response->detail = $request->detail;
            $response->weight = $request->weight;
            $response->nominal = $request->nominal;
            $response->rfid = $request->rfid;
            $response->gambar = $name;
            $response->save();
            return redirect()->back()->with(['messages'=>'Data telah ter input!']);

        } catch (Throwable $e) {
            return redirect()->back()->with(compact('e'));
        }

    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {

        $product    = Product::find($id);
        $categories = Categories::orderBy('name')->get();
        $emas       = Emas::orderBy('name')->get();
        $mahkota    = Mahkota::orderBy('name')->get();
        $cabinet    = Cabinet::orderBy('name')->get();
```

```
        return view('data.product.form')-
>with(compact('categories','emas','mahkota','cabinet','product'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $product = Product::find($id);
        $categories = Categories::orderBy('name')->get();
        $emas = Emas::orderBy('name')->get();
        $mahkota = Mahkota::orderBy('name')->get();
        $cabinet = Cabinet::orderBy('name')->get();

        return view('data.product.form')-
>with(compact('categories','emas','mahkota','cabinet','product'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        try {
            $response = Product::findOrFail($id);
            $name = $response->gambar;
            if (preg_match('/^data:image\/(\w+);base64,/', $request->gambare)) {
                Storage::disk('product')->delete($response->gambar);
                $data = substr($request->gambare, strpos($request->gambare, ',') + 1);

                $data = base64_decode($data);
                $name = date('mdYHis').".png";
                $path = Storage::disk('product')->put($name, $data);
            }

            $response->kode = $request->kode;
            $response->categories_id = $request->categories_id;
            $response->emas_id = $request->emas_id;
            $response->mahkota_id = $request->mahkota_id;
            $response->cabinet_id = $request->cabinet_id;
            $response->name = $request->name;
            $response->detail = $request->detail;
            $response->weight = $request->weight;
            $response->nominal = $request->nominal;
            $response->rfid = $request->rfid;
            $response->gambar = $name;
            $response->save();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

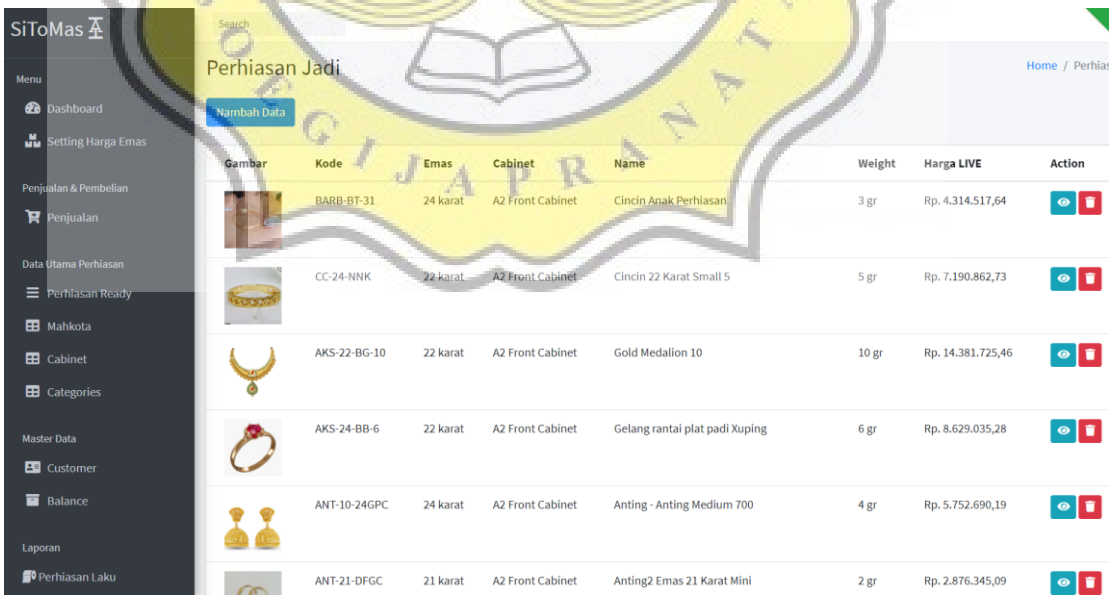
```
     */
    public function destroy($id)
    {
        try {
            $response = Product::findOrFail($id);
            Storage::disk('product')->delete($response->gambar);
            $response->delete();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }

    public function resale($id)
    {
        try {
            $response = Product::findOrFail($id);
            Storage::disk('product')->delete($response->gambar);
            $response->delete();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }
}
```

Gambar 4.45 Kode Halaman Perhiasan

```
@section('content')
    <div class="container-fluid">
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Test Echo data</h5>
        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
      <a href="{{route('product.create')}}" class="btn btn-primary">Nambah Data</a>
      <br/>
      <br/>
        <div class="row">
          <div class="col-12">
            <div class="card">
              <div class="card-body table-responsive p-0" style="height: 700px;">
                <table class="table table-head-fixed text-nowrap">
                  <thead>
                    <tr>
                      <th>Gambar</th>
                      {{-- <th>Kategori</th> --}}
                      <th>Kode</th>
                      <th>Emas</th>
                      {{-- <th>Gemstone</th> --}}
                      <th>Cabinet</th>
                      <th>Name</th>
                      <th>Weight</th>
                      <th>Harga LIVE</th>
                      <th>Action</th>
                    </tr>
                  </thead>
                  <tbody>
                    @foreach($data as $row)
                      <tr>
                        <td>@if($row-
>gambar) <img width="75px" src="{{asset('/storage/product/'.$row->gambar)}}"> @endif</td>
```

```
                        {{-- <td>{{$row->categories->name}}</td> --}}
                        <td>{{$row->kode}}</td>
                        <td>{{$row->emas->name}}</td>
                        {{-- <td>{{$row->mahkota->name}}</td> --}}
                        <td>{{$row->cabinet->name}}</td>
                        <td>{{$row->name}}</td>
                        <td>{{$row->weight}} gr</td> <!-- Weight e string -->
                        <td>Rp. {{number_format($row-
>weight * $markupgoldprice, 2, ',', '.')}}</td> <!-- $row->weight * $harga emas -->
                        <td>
                          <a href="{{route('product.edit',$row->id)}}" class="btn btn-
sm btn-info modalMd">
                            <i class="fa fa-eye"></i>
                          </a>
                          <form action="{{route('product.destroy',$row->id)}}" class="d-
inline" method="post">
                            @csrf
                              <input name="_method" type="hidden" value="DELETE">
                              <button type="submit" class="btn btn-danger btn-
sm"><i class="fa fa-trash"></i></button>
                          </form>
                        </td>
                      </tr>
                    @endforeach
                  </tbody>
                </table>
              </div>
              <!-- /.card-body -->
            </div>
            <!-- /.card -->
          </div>
        </div>
    </div>

@endsection
```

Gambar 4.46 Kode UI Halaman Perhiasan



Gambar 4.46 Hasil Halaman Perhiasan

4.2.8 Halaman Data Customer

Pengguna yang memiliki *role* tertentu yang bisa mengakses halaman data customer. Halaman ini menyimpan data – data pembeli, untuk menambahkan customer baru bisa menekan tombol “tambah data”. Cuplikan kode halaman data customer bisa dilihat pada gambar dibawah.

```
class CustomerController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

        /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = Customer::orderBy('id')->get();

        return view('data.customer.index')->with(compact('data'));
    }

    public function create()
    {

        $customer = Customer::orderBy('id')->first();
        return view('data.customer.form');
    }

        /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        try {
            $response = Customer::insert($request->except(['_token']));
            return redirect()->back()->with(['messages'=>'Data telah ter input!']);

        } catch (Throwable $e) {
            return redirect()->back()->with(compact('e'));
        }

    }

        /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
```

```
        $customer = Customer::orderBy('id')->get();
        return view('data.customer.form')->with(compact('customer'));
    }

    public function edit($id)
    {

        $customer = Customer::find($id);
        return view('data.customer.form')->with(compact('customer'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        try {
            $response = Customer::findOrFail($id);
            $input = $request->except(['_token']);
            $response->fill($input)->save();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        try {
            $response = Customer::findOrFail($id)->delete();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }
}
```

Gambar 4.47 Kode Halaman Customer

```
@section('content')
    <div class="container-fluid">
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Test Echo data</h5>

        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
      <a href="{{route('customer.create')}}" class="btn btn-primary">Nambah Data</a>
      <br/>
      <br/>
        <div class="row">
```

```
          <div class="col-12">
            <div class="card">
              <div class="card-body table-responsive p-0" style="height: 700px;">
                <table class="table table-head-fixed text-nowrap">
                  <thead>
                    <tr>
                      <th>Nama</th>
                      <th>No HP</th>
                      <th>Alamat</th>
                      <th>Action</th>
                    </tr>
                  </thead>
                  <tbody>
                    @foreach($data as $row)
                      <tr>
                        <td>{{$row->name}}</td>
                        <td>{{$row->nohp}}</td>
                        <td>{{$row->alamat}}</td>
                        <td>
                          <a href="{{route('customer.edit',$row->id)}}" class="btn btn-
warning">Edit</a><br/><br/>
                          <form action="{{route('customer.destroy',$row-
>id)}}" method="post">
                            @csrf
                            <input name="_method" type="hidden" value="DELETE">
                            <button type="submit" class="btn btn-danger">Delete</button>
                          </form>
                        </td>
                      </tr>
                    @endforeach
                  </tbody>
                </table>
              </div>
              <!-- /.card-body -->
            </div>
            <!-- /.card -->
          </div>
        </div>
    </div>
@endsection
```

Gambar 4.48 Kode UI Halaman Customer

4.2.9 Halaman Balance

Halaman balance hanya bisa diakses oleh pengguna yang memiliki *role* pemilik. Halaman ini menampilkan akun atau *balance* yang dipakai pada toko perhiasan. Cuplikan kode halaman balance bisa dilihat pada gambar dibawah.

```
class FundController extends Controller
{
    protected $data;

    public function __construct()
    {
        $this->data = array();
    }
```

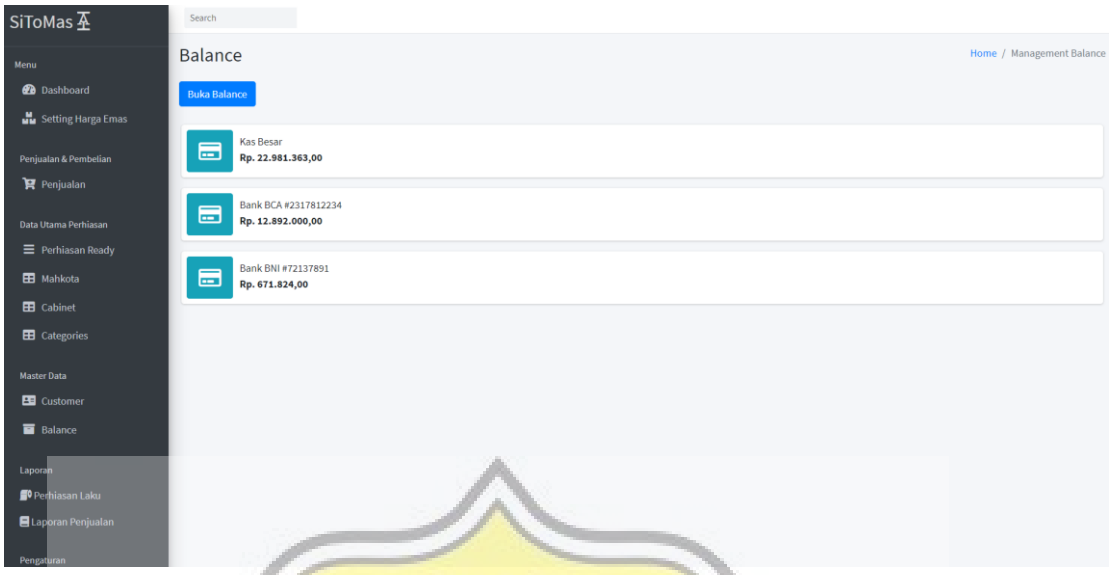
```php
public function index()
{
    $this->data['balance'] = Fund::get();

    return view('data.balance.index')->with($this->data);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    $categories = Fund::orderBy('id')->first();
    return view('data.balance.form');
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    try {
        $response = Fund::insert($request->except(['_token']));
        return redirect()->back()->with(['messages'=>'Data telah ter input!']);

    } catch (Throwable $e) {
        return redirect()->back()->with(compact('e'));
    }

}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
```

```
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

Gambar 4.49 Kode Halaman Balance

```
@section('content')
    <div class="container-fluid">
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Test Echo data</h5>

        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
      <a href="{{route('balance.create')}}" class="btn btn-primary">Buka Balance</a>
      <br/>
      <br/>
        <div class="card-body table-responsive p-1" style="height: 700px;">
        @foreach($balance as $row)
            <div class="info-box">
                <span class="info-box-icon bg-info"><i class="far fa-credit-
card"></i></span>
                    <div class="info-box-content">
                        <span class="info-box-text">{{$row->name}}</span>
                        <span class="info-box-number">Rp. {{number_format($row-
>nominal, 2, ',', '.')}}</span>
                    </div>
            </div>
        @endforeach
          </div>
        </div>
    </div>

@endsection
```

Gambar 4.50 Kode UI Halaman Balance



Gambar 4.51 Hasil Halaman Balance

4.2.10 Halaman Setting Toko

Identitas toko perhiasan bisa dikostumisasi secara bebas, yang bisa membuka menu ini adalah pengguna yang memiliki *role* admin & pemilik. Data ini akan dipakai untuk menampilkan identitas toko pada nota atau *invoice* penjualan dan laporan penjualan. Cuplikan kode halaman balance bisa dilihat pada gambar dibawah.

```
class PengaturanController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }

        /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = Pengaturan::orderBy('id')->get();

        return view('data.pengaturan.index')->with(compact('data'));
    }

    public function create()
    {

        $pengaturan = Pengaturan::orderBy('id')->first();
        return view('data.pengaturan.form');
```

```
}

    /**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    try {
        $response = Pengaturan::insert($request->except(['_token']));
        return redirect()->back()->with(['messages'=>'Data telah ter input!']);

    } catch (Throwable $e) {
        return redirect()->back()->with(compact('e'));
    }

}

    /**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{

    $pengaturan = Pengaturan::orderBy('id')->first();
    return view('data.pengaturan.form')->with(compact('pengaturan'));
}

public function edit($id)
{

    $pengaturan = Pengaturan::orderBy('id')->first();
    return view('data.pengaturan.form')->with(compact('pengaturan'));
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    try {
        $response = Pengaturan::findOrFail($id);
        $name = $response->gambar;
        if (preg_match('/^data:image\/(\w+);base64,/', $request->gambare)) {
            Storage::disk('product')->delete($response->gambar);
            $data = substr($request->gambare, strpos($request->gambare, ',') + 1);

            $data = base64_decode($data);
            $name = date('mdYHis').".png";
            $path = Storage::disk('product')->put($name, $data);
        }

        $response->name = $request->name;
        $response->alamat = $request->alamat;
        $response->notel = $request->notel;
        $response->nofax = $request->nofax;
        $response->email = $request->email;
```

```
            $response->subtext = $request->subtext;
            $response->gambar = $name;
            $response->save();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        try {
            $response = Pengaturan::findOrFail($id)->delete();
        } catch (Throwable $e) {
            return back()->withError($exception->getMessage())->withInput();
        }
        return back()->withError($response)->withInput();
    }


}
```

Gambar 4.51 Kode Halaman Setting Toko

```
@section('content')
    <div class="container-fluid">
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Test Echo data</h5>

        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
      <br/>
      <br/>
        <div class="row">
          <div class="col-12">
            <div class="card">
              <div class="card-body table-responsive p-0" style="height: 150px;">
                <table class="table table-head-fixed text-nowrap">
                  <thead>
                    <tr>
                      <th>Nama</th>
                      <th>Alamat</th>
                      <th>No Telepon</th>
                      <th>Logo</th>
                    </tr>
                  </thead>
                  <tbody>
                    @foreach($data as $row)
                      <tr>
                        <td>{{$row->name}}</td>
                        <td>{{$row->alamat}}</td>
                        <td>{{$row->notel}}</td>
```

```
                        <td>@if($row->gambar) <img width="75px" src="{{asset('/storage/product/'.$row->gambar)}}"> @endif</td>
                        <td>
                          <a href="{{route('pengaturan.edit',$row->id)}}" class="btn btn-warning">Edit</a><br/><br/>
                            @csrf
                          </form>
                        </td>
                      </tr>
                    @endforeach
                  </tbody>
                </table>
              </div>
              <!-- /.card-body -->
            </div>
            <!-- /.card -->
          </div>
        </div>
      </div>

@endsection
```

Gambar 4.52 Kode UI Halaman Setting Toko

Gambar 4.52 Hasil Halaman Setting Toko

4.2.11 Halaman Pengaturan Role

Halaman pengaturan memungkinkan pengguna yang memiliki hak akses mengatur *role* tiap pengguna. Tampilan halaman ini berupa daftar pengguna

yang sudah dibuat dan *role* yang sudah diberikan. Cuplikan kode halaman balance bisa dilihat pada gambar dibawah.

```
class UserController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        $data = User::all();

        return view('data.user.index')->with('data',$data);
    }

    public function edit(User $user)
    {
        //CheckROLe
        if(Gate::denies('edit-users')){
            return redirect(route('user.index'))-
>with('alert', 'Anda Tidak Memiliki Hak Akses');
        }
        //CheckROLe

        $role = Role::all();

        return view('data.user.form')->with([
            'user' => $user,
            'roles' => $role
        ]);
    }

    public function update(Request $request, User $user)
    {
        $user->role()->sync($request -> roles);

        return redirect()->route('user.index');

    }

    public function destroy(Request $request, User $user)
    {

    }
}
```

Gambar 4.53 Kode Halaman Pengaturan Role

```
@section('content')
    <div class="container-fluid">
      <a href="{{route('user.index')}}" class="btn btn-primary">Back</a>
      <br/>
      <br/>
      @if(session('error'))
      <div class="callout callout-primary">
        <h5>Data Send!</h5>

        <p>{{session('error')}}</p>
    </div><br/>
      <br/>
    @endif
```

```
        <div class="row">
          <div class="col-12">
          <div class="card card-primary">
              <div>
              <h4 class="card-header">Edit Role Untuk User : {{$user -> email}}</h3>
              </div>
              <!-- /.card-header -->
              <!-- form start -->
              <form role="form" method="post" @if(@$user)  action="{{route('user.update',@
$user->id)}}"  @else action="{{route('user.store')}}" @endif>
                @csrf
                {{method_field('PUT')}}
                @foreach ($roles as $role)
                    <div class="form-check">
                      <input type="checkbox" name="roles[]" value="{{$role ->id}}">
                      <label>{{$role -> name}}</label>
                    </div>
                @endforeach
                    <button type="submit" class="btn btn-warning col-12">
                      Update Role
                    </button>
              </form>
            </div>
          </div>
        </div>
    </div>

@endsection
```

Gambar 4.54 Kode UI Halaman Pengaturan Role

Gambar 4.55 Hasil Halaman Pengaturan Role

4.2.12 RESTful API

API diperlukan untuk menyediakan data yang nantinya ditampilkan pada aplikasi Android. API menggunakan konsep *RESTful* (Representational state transfer). Data yang akan dikirim dipecah menjadi 2 bagian. Bagian pertama API untuk mendapatkan perhiasan yang tersedia, yang ke-dua menyediakan data harga emas yang sudah di markup. Cuplikan kode *RESTful* API perhiasan dan harga *markup* emas bisa dilihat pada gambar dibawah.

```
class APIController extends Controller
{
    public function responseSuccess($data){
        return $response = [
            'status' => true,
            'message' => 'success',
            'data' => $data,
        ];
    }

    public function responseFailed($data){
        return $response = [
            'status' => false,
            'message' => 'failed',
            'data' => $data,
        ];
    }

    public function getRawPrice(){
        $client = new \GuzzleHttp\Client();

        // Create a request
        $requestEmas = $client->get('https://api.sitomas.my.id/gold');
        $requestUSD = $client->get('https://api.sitomas.my.id/currency');

        // Get the actual response without headers
        $responseEmas = $requestEmas->getBody();
        $responseEmas = json_decode($responseEmas, true);

        $responseUSD = $requestUSD->getBody();
        $responseUSD = json_decode($responseUSD, true);

        $hargaemas  = $responseEmas['Open'];
        $usd        = $responseUSD['Open'];

        $hargaemas  = 1920;
        $usd        = 14750;

        $price = Product::orderBy('name')->get()->where('status', 1);

        $persen = PricesEmas::first()->value('persen');

        $goldprice = $hargaemas * $usd / 29.3;
        $markupprice = $goldprice * $persen / 100;
        return $markupgoldprice = $goldprice + $markupprice;
    }

    public function getProductAPI(){

        $markupgoldprice = $this->getRawPrice();
```

```
        $data = Product::select('id','kode','name','detail','rfid','weight','gambar')-
>get()->where('status', 0);

        foreach ($data as $value) {
            $price = number_format($markupgoldprice * $value['weight'], 0, ',', '.');
            $value['harga'] = $price;
        }
        return response()->json($this->responseSuccess($data));
    }

    public function getProductDetailAPI($id){
        $data = Product::select('id','kode','name','detail','rfid','weight','gambar')-
>get()->where('id', $id)->where('status', 0)->first();
        if ($data) {
            $markupgoldprice = $this->getRawPrice();
            $price = number_format($markupgoldprice * $data['weight'], 0, ',', '.');
            $data['harga'] = $price;
            return response()->json($this->responseSuccess($data));
        } else {
            return response()->json($this->responseFailed($data));
        }

    }

    public function getMarkPricesAPI(){
        $markupgoldprice = $this->getRawPrice();
        $data = number_format($markupgoldprice, 0, ',', '.');

        return response()->json($this->responseSuccess($data));
    }
}
```

Gambar 4.56 Hasil Halaman Pengaturan Role

4.3 Pembuatan Aplikasi Android

Aplikasi Android yang dibuat menggunakan bahasa pemrograman react-native. Peran utama aplikasi ini adalah mengambil data yang sudah disiapkan oleh RESTful API dari Sistem Informasi dan menyajikannya kedalam sebuah aplikasi.

4.3.1 Halaman Home

Halaman *home* adalah tampilan pertama saat pengguna membuka aplikasi. Terdapat 2 komponen utama pada halaman *home*. Komponen pertama adalah menampilkan harga emas yang sudah diautur oleh toko atau harga *markup*. Komponen kedua adalah tampilan *grid-view* dari 4 produk terbaru dengan format 2 x 2. Kode untuk memunculkan halaman *home* bisa dilihat pada Gambar 4.57

```
export default function HomeScreen(props) {
    const [goldPrice, setGoldPrice] = useState(0);
    const [newProduct, setNewProduct] = useState(null);
```

```
    const [refreshing, setRefreshing] = useState(false)

    useEffect(() => {
        getGoldPrice();
        getNewProduct();
    }, [])

    const onRefresh = () => {
        setRefreshing(true);
        getGoldPrice();
        getNewProduct();
    }

    const getGoldPrice = async () => {
        try {
            let res = await Axios.get(`${API_config.api}/getmarkprice`);
            if (res.data.status) {
                setRefreshing(false)
                setGoldPrice(res.data.data);
            }
        } catch (error) {
            console.log(error)
        }
    }

    const getNewProduct = async () => {
        try {
            let res = await Axios.get(`${API_config.api}/getproduct`);
            if (res.data.status) {
                setRefreshing(false)
                let value = res.data.data;
                value.sort((a,b) => {
                    return b.id - a.id;
                })
                value = value.splice(0, 4);
                setNewProduct(value);
            }
        } catch (error) {
            console.log(error)
        }
    }
    return (
        <SafeAreaView>
            <ScrollView
            contentInsetAdjustmentBehavior="automatic"
            refreshControl={<RefreshControl refreshing={refreshing} onRefresh={onRefresh}
/>}>
                <View>
                    <View style={styles.jumboTron}>
                        <Image source={jewelryStore} style={styles.jewelryStore}>
                        </Image>
                        <Text style={styles.welcomeText}>Selamat Datang</Text>
                        <View style={styles.statusPriceBox}>
                            <MaterialCommunityIcons style={styles.statusTitleText} name="g
old" color="#fff" size={32} />
                            <Text style={styles.statusTitleText}>Harga Emas Hari Ini</Text
>

                            <Text style={styles.statusPriceText}>Rp{goldPrice}</Text>
                        </View>
                    </View>
                    <View style={styles.parentProducts}>
                        <Text style={styles.headingProducts}>Produk Baru</Text>
                        <View style={styles.products}>
                            {
                                newProduct
                                ?
```

```
                            newProduct.length
                            ?
                            newProduct.map(item => {
                                return <Products key={item.id} data={item} {...props}
/>
                            })
                            :
                            <Text>Product Tidak Ada</Text>
                            :
                            <ActivityIndicator />
                        }
                    </View>
                </View>
            </View>
        </ScrollView>
    </SafeAreaView>
  )
}
```

Gambar 4.57 Kode Aplikasi Android Halaman Home

Gambar 4.58 Hasil Aplikasi Android Halaman Home

4.3.2 Halaman Perhiasan

Halaman produk menampilkan semua perhiasan yang ada, dengan format *grid-view*. Untuk melihat detail dari perhiasan, pengguna cukup menyentuh

gambar perhiasan yang ingin dilihat. Kode untuk memunculkan halaman perhiasan bisa dilihat pada Gambar 4.59.

```
export default function ProductsScreen(props) {
    const [product, setProduct] = useState([]);
    const [searchResult, setSearchResult] = useState(null);
    const [searchInput, setSearchInput] = useState('');
    const [refreshing, setRefreshing] = useState(false)

    useEffect(() => {
        getProduct();
    }, [])

    useEffect(() => {
        searchProduct(searchInput)
    }, [searchInput])

    const onRefresh = () => {
        setRefreshing(true);
        getProduct();
        setSearchInput('');
    }

    const getProduct = async () => {
        try {
            let res = await Axios.get(`${API_config.api}/getproduct`);
            if (res.data.status) {
                setRefreshing(false)
                let value = res.data.data;
                setProduct(value);
                setSearchResult(value);
            }
        } catch (error) {
            console.log(error)
        }
    }
    const searchProduct = (value) => {
        if (value.length > 1) {
            let temp = product.filter(e => {
                return e.name.toLowerCase().includes(value.toLowerCase()) || e.rfid.toStri
ng().includes(value);
            })
            setSearchResult(temp)
        }else{
            setSearchResult(product)
        }
    }
    return (
        <SafeAreaView>
            <Searchbar placeholder="Cari Perhiasan" value={searchInput} onChangeText={(e)
=> setSearchInput(e)} />
            <ScrollView
            contentInsetAdjustmentBehavior="automatic"
            refreshControl={<RefreshControl refreshing={refreshing} onRefresh={onRefresh}
/>}>
                <View>
                    <View style={styles.parentProducts}>
                        <Text style={styles.headingProducts}>Total Item: {searchResult &&
searchResult.length}</Text>
                        <View style={styles.products}>
                        {
                            searchResult && product.length
                            ?
                            searchResult.length
```

```
                            ?
                            searchResult.map(item => {
                                return <Products key={item.id} data={item} {...props} />
                            })
                            :
                            <Text>Product Tidak Ada</Text>
                            :
                            <ActivityIndicator />
                        }
                        </View>
                    </View>
                </View>
            </ScrollView>
        </SafeAreaView>
    )
}
```

Gambar 4.59 Kode Aplikasi Android Halaman Perhiasan

4.3.3 Halaman Detail Perhiasan

Setelah user menekan gambar perhiasan yang diinginkan, maka muncul *pop-up* halaman detail perhiasan. Halaman ini memunculkan detail perhiasan seperti, gambar yang lebih jelas, nama perhiasan, dan berat perhiasan. Kode untuk memunculkan halaman detail perhiasan bisa dilihat pada Gambar 4.60.

```
export default function ProductDetail(props){
    const [data, setData] = useState({});
    const [zoomImage, setZoomImage] = useState(false)
    const [readMore, setReadMore] = useState(false)

    useEffect(() => {
        getProductDetail();
    }, [])

    const getProductDetail = async () => {
        try {
            let id = props.route.params.id;
            let res = await Axios.get(`${API_config.api}/getproduct/${id}`);
            if (res.data.status) {
                let value = res.data.data;
                setData(value);
            }
        } catch (error) {
            console.log(error)
        }
    }

    const storeData = async (key, value) => {
        AsyncStorage.getItem(key).then(item => {
            let temp = item ? JSON.parse(item) : [];
```

```
            if (temp.find(e => e.id === value.id)){
                temp.forEach((e, i) => {
                    if (e.id === value.id) {
                        temp[i] = value;
                    }
                })
            }else{
                temp.push(value);
            }
            const setObjectValue = JSON.stringify(temp);
            AsyncStorage.setItem(key, setObjectValue);
        })
        alert('Sukses')
    }

    const handleAddToCart = () => {
        storeData('cart', data)
    }

    const handleZoomImage = () => {
        setZoomImage(!zoomImage)
    }
    return(
        <SafeAreaView>
            <ScrollView
            contentInsetAdjustmentBehavior="automatic">

                <View style={styles.container}>
                    <Modal visible={zoomImage} transparent={true}>
                        <ImageViewer enableSwipeDown={true} onSwipeDown={handleZoomImage}
imageUrls={[{url: `${API_config.img}/${data.gambar}`}]} />
                    </Modal>
                    <Card>
                        {
                            data.gambar
                            ?
                            <TouchableOpacity onPress={handleZoomImage}>
                                <Card.Cover height={20} source={{ uri: `${API_config.img}/
${data.gambar}` }} />
                            </TouchableOpacity>
                            :
                            <Card.Cover height={20} source={defaultPhoto} />
                        }
                        <Card.Content>
                            <Title>{data.name && data.name}</Title>
                            <View style={styles.field}>
                                <Text style={styles.label}>Kode :</Text>
                                <Text>{data.kode && data.kode}</Text>
                            </View>
                            <View style={styles.field}>
                                <Text style={styles.label}>Deskripsi :</Text>
                                {
                                    !readMore
                                    ?
                                    <View>
                                        <Text style={{textAlign: "justify"}}>{`${data.deta
il && data.detail.substring(0, 50)}${data.detail && data.detail.length > 50 ? '...' : ''}`
}</Text>
                                        {
                                            data.detail && data.detail.length > 50 &&
                                            <TouchableOpacity onPress={() => setReadMore(!
readMore)}>

                                                <Text style={{color: "blue"}}>read more</T
ext>

                                            </TouchableOpacity>
                                        }
```

```
                                    </View>
                                    :
                                    <View>
                                        <Text style={{textAlign: "justify"}}>{data.detail
&& data.detail}</Text>

                                        <TouchableOpacity onPress={() => setReadMore(!read
More)}>

                                            <Text style={{color: "blue"}}>read less</Text>
                                        </TouchableOpacity>
                                    </View>

                                }
                            </View>
                            {/* <View style={styles.field}>
                                <Text style={styles.label}>RFID :</Text>
                                <Text>{data.rfid && data.rfid}</Text>
                            </View> */}
                            <View style={styles.field}>
                                <Text style={styles.label}>Weight :</Text>
                                <Text>{data.weight && data.weight} gram</Text>
                            </View>
                            <View style={styles.field}>
                                <Text style={styles.label}>Harga :</Text>
                                <Text>Rp{data.harga && data.harga}</Text>
                            </View>
                        </Card.Content>
                        <Card.Actions>
                            <Button style={{marginVertical: 12}} icon="cart-
plus" mode="contained" color="#f8aa27" onPress={handleAddToCart}>
                                Add to Cart
                            </Button>
                        </Card.Actions>
                    </Card>
                </View>
            </ScrollView>
        </SafeAreaView>
    );
}
```

Gambar 4.60 Kode Aplikasi Android Halaman Detail Perhiasan

Gambar 4.61 Hasil Aplikasi Android Halaman Detail Perhiasan

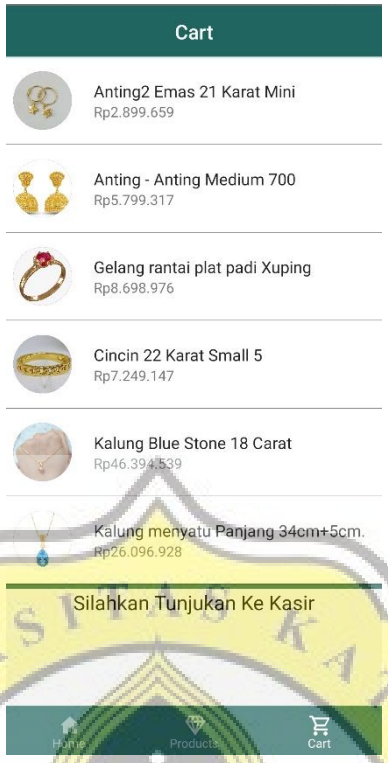
4.3.4 Halaman Keranjang Belanja

Setelah pengguna menemukan perhiasan yang cocok, pengguna bisa menekan tombol "add to cart" untuk memasukan perhiasan tersebut kedalam keranjang belanja digital. Di Dalam halaman ini memberikan daftar barang yang sudah masuk ke keranjang belanja digital. Kode untuk memunculkan halaman keranjang belanja bisa dilihat pada Gambar 4.62.

```
export default function CartScreen(props) {
    const [data, setData] = useState([])
    const isFocused = useIsFocused()
    const [refreshing, setRefreshing] = useState(false)

    useEffect(() => {
        getCart()
    }, [])

    useEffect(() => {
        getCart()
    }, [isFocused])

    const onRefresh = () => {
        setRefreshing(true);
        getCart();
```

```
        setRefreshing(false)
    }

    const getCart = () => {
        AsyncStorage.getItem('cart').then(item => {
            let temp = item ? JSON.parse(item) : [];
            setData(temp);
        })
    }

    const handleChanged = () => {
        getCart()
    }

    return (
        <SafeAreaView>
            <ScrollView
            contentInsetAdjustmentBehavior="automatic"
            refreshControl={<RefreshControl refreshing={refreshing} onRefresh={onRefresh}
/>}>
                <View>
                    {
                    data.length
                    ?
                    data.map(item => {
                        return <ProductList key={item.id} changed={handleChanged} data
={item} {...props} />
                    })
                    :
                    <Text style = {styles.warningText}>Keranjang Kosong, Silahkan Masu
kan Produk</Text>
                }
                <View
                    style={{
                        borderBottomColor: '#20655f',
                        borderBottomWidth: 5,
                    }}
                    />
                    <Text style = {styles.baseText}>Silahkan Tunjukan Ke Kasir</Text>
                </View>
            </ScrollView>
        </SafeAreaView>
    )
}
```

Gambar 4.62 Kode Aplikasi Android Halaman Keranjang Belanja



Gambar 4.63 Hasil Aplikasi Android Halaman Keranjang Belanja

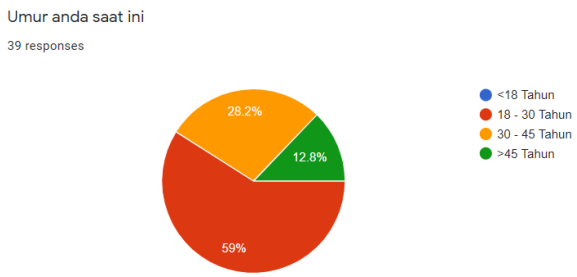
4.4 Pengujian Aplikasi

Pengumpulan data menggunakan metode pembagian kuesioner serta mencoba aplikasi Android. Terkumpul sebanyak 39 responden dengan rentang usia 18 – 45 tahun hasil data sebagai berikut

4.5.1 Profil Responden

4.5.1.1 Usia

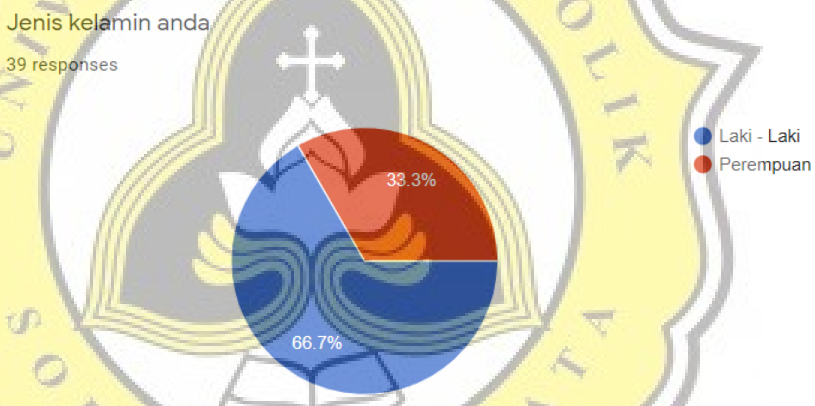
Responden terbanyak memiliki kelompok usia 18 – 30 tahun dengan persentase terbesar sebanyak 65.6%. Kemudian disusul dengan kelompok usia 30 – 45 dengan persentase sebanyak 18.8%. Responden dengan kelompok usia lebih dari 45 merupakan yang tersedikit dengan persentase sebanyak 15.6%. Gambar 4.64 dibawah menggambarkan diagram kelompok umur responden.



Gambar 4.64 Hasil Usia Responden

4.5.1.1 Jenis Kelamin

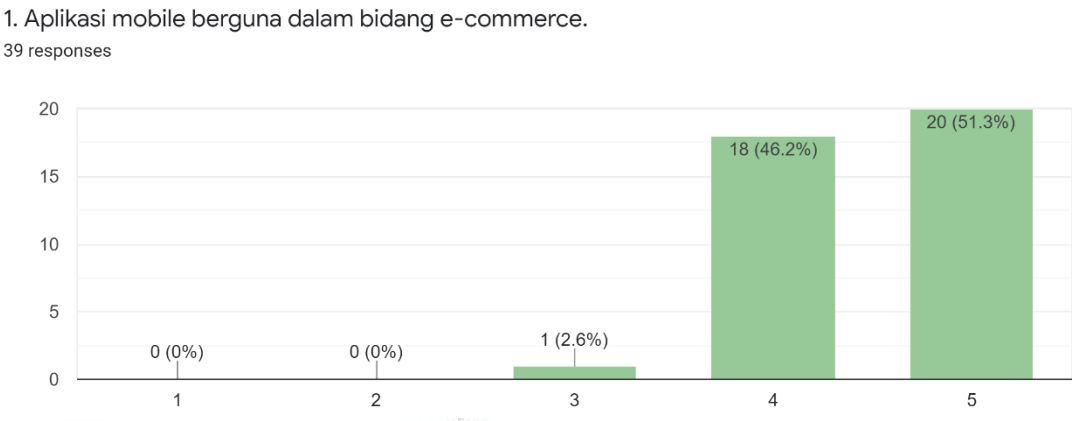
Responden terbanyak memiliki jenis kelamin laki - laki dengan persentase seanyakak 62.5% dan perempuan sebanyak 37.5%. Gambar 4.65 dibawah menggambarkan diagram jenis kelamin responden.



Gambar 4.65 Hasil Jenis Kelamin Responden

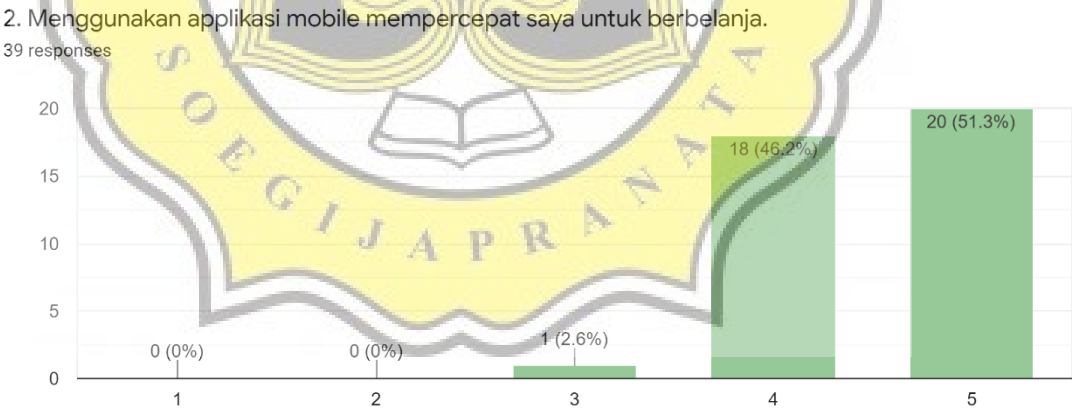
4.5.2 Pembahasan Data

Didapatkan data sebanyak 39 responden, data sebanyak 39 responden tadi digunakan untuk menguji atau menilai aplikasi. Pernyataan pertama menunjukan bahwa 51% sangat setuju, 46% setuju, dan 2.6% netral bahwa aplikasi mobile berguna dalam bidang e-commerce. Hasil diagram responden bisa dilihat pada Gambar 4.66 dibawah.



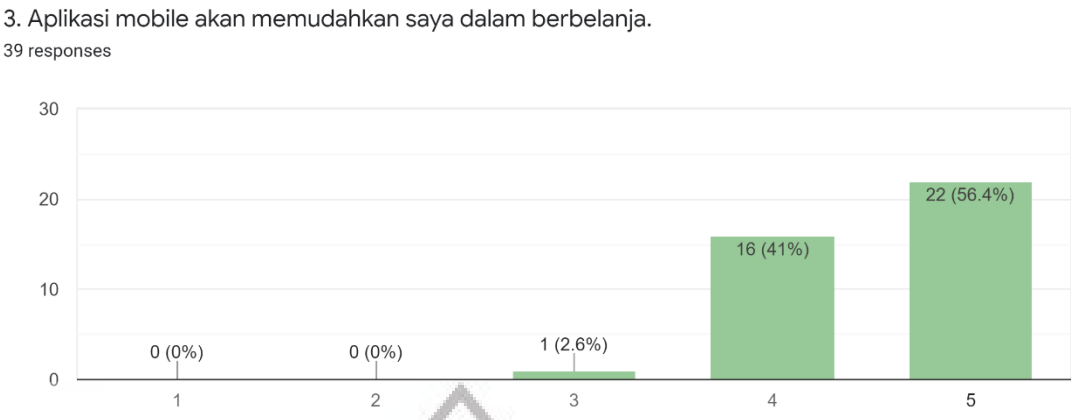
Gambar 4.66 Hasil Aplikasi Mobile Berguna Dalam E-commerce

Pernyataan kedua menunjukan bahwa 51.3% sangat setuju, 46.2% setuju, dan 2.6% netral bahwa aplikasi mobile mempercepat untuk berbelanja. Hasil diagram responden bisa dilihat pada Gambar 4.67dibawah.



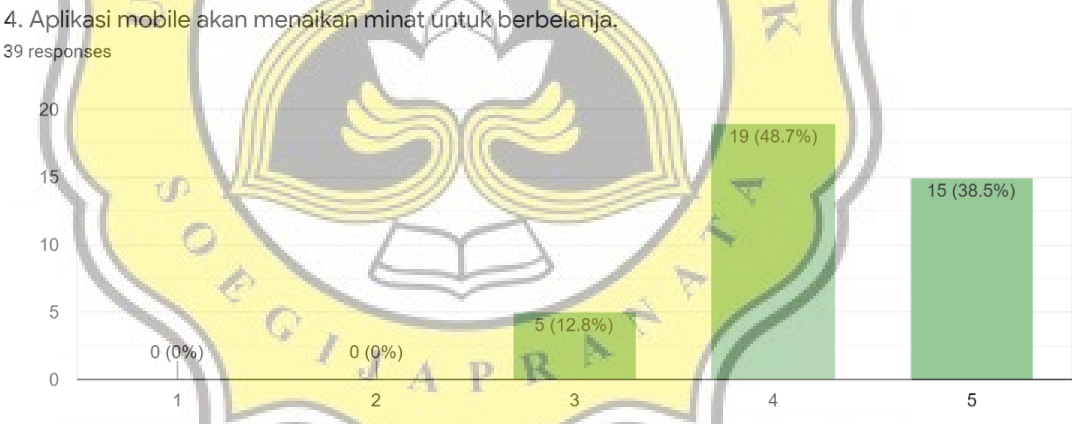
Gambar 4.67 Hasil Aplikasi Mobile Mempercepat Belanja

Pernyataan ketiga menunjukan bahwa 56.4% sangat setuju, 41% setuju, dan 2.6% netral bahwa aplikasi mobile memudahkan dalam berbelanja. Hasil diagram responden bisa dilihat pada Gambar 4.68 dibawah.



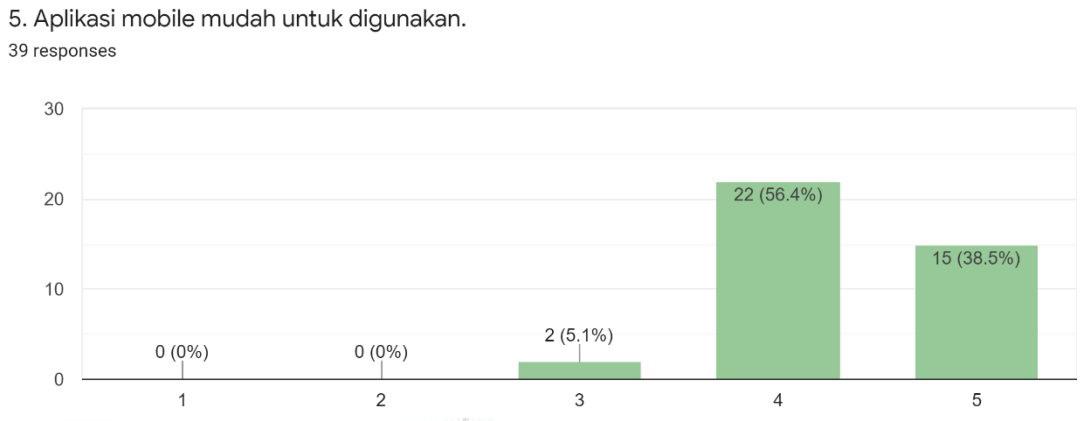
Gambar 4.67 Hasil Aplikasi Mobile Mempermudah Belanja

Pernyataan keempat menunjukan bahwa 58.5% sangat setuju, 48.7% setuju, dan 12.8% netral bahwa aplikasi mobile menaikan minat dalam berbelanja.



Gambar 4.68 Hasil Aplikasi Mobile Menaikan Minat Belanja

Pernyataan kelima menunjukan bahwa 38.5% sangat setuju, 56.4% setuju, dan 5.1% netral bahwa aplikasi mobile mudah untuk digunakan. Hasil diagram responden bisa dilihat pada Gambar 4.69 dibawah.

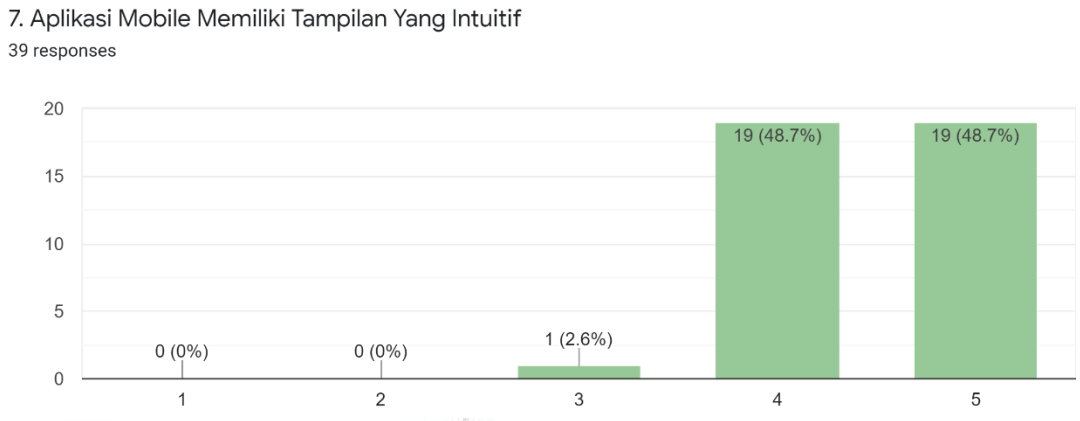


Gambar 4.69 Hasil Aplikasi Mobile Mudah Untuk Digunakan

Pernyataan keenam menunjukan bahwa 48.7% sangat setuju, 46.2% setuju, dan 5.1% netral bahwa aplikasi menggunakan fitur pada aplikasi mobile itu mudah. Hasil diagram responden bisa dilihat pada Gambar 4.70 dibawah.

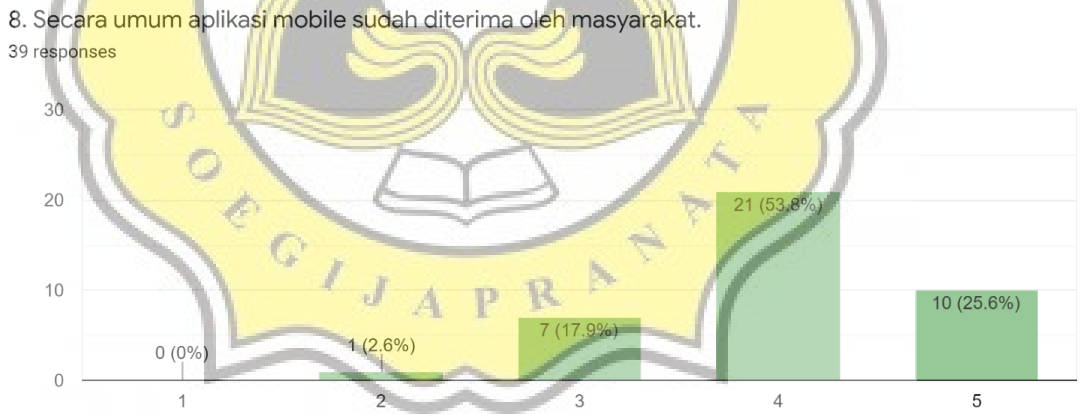
Gambar 4.70 Hasil Menemukan / Menggunakan Fitur Aplikasi Mobile

Pernyataan ketujuh menunjukan bahwa 48.7% sangat setuju, 48.7% setuju, dan 5.1% netral bahwa mobile memiliki tampilan yang intuitif. Hasil diagram responden bisa dilihat pada Gambar 4.71 dibawah.



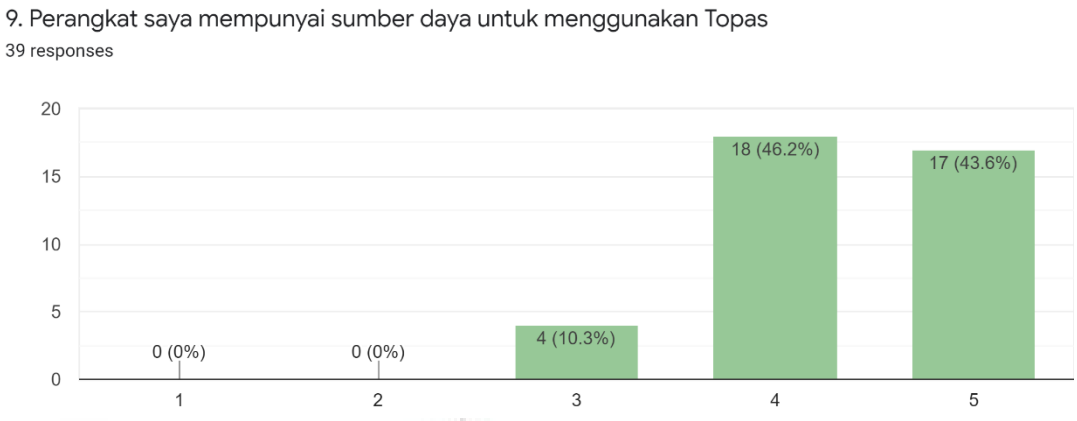
Gambar 4.71 Hasil Aplikasi Mobile Memiliki Tampilan Yang Intuitif

Pernyataan kedelapan menunjukan bahwa 26.6% sangat setuju, 53.8% setuju, 5.1% netral, dan 2.6% tidak setuju bahwa mobil sudah diterima oleh masyarakat. Hasil diagram responden bisa dilihat pada Gambar 4.72 dibawah.



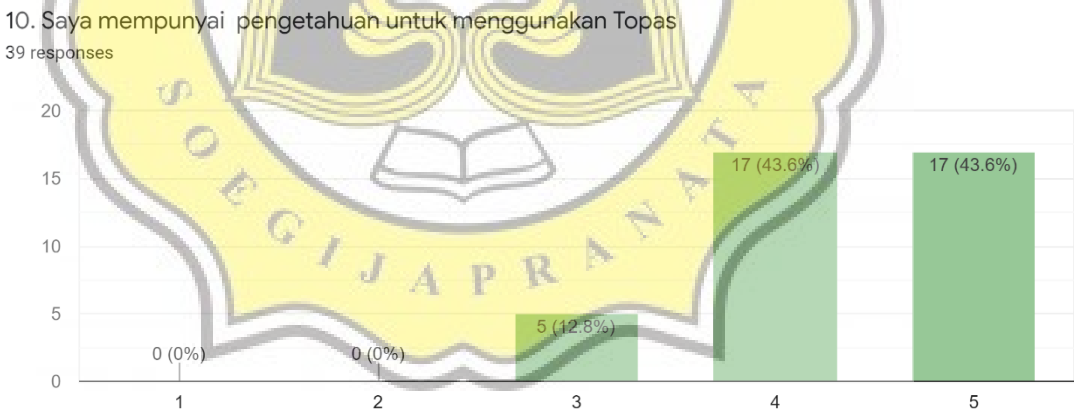
Gambar 4.72 Hasil Aplikasi Mobile Sudah Diterima Oleh Masyarakat

Pernyataan kesembilan menunjukan bahwa 43.6% sangat setuju, 46.2% setuju, dan 10.3% netral bahwa responden mempunyai sumber daya untuk menggunakan aplikasi topas. Hasil diagram responden bisa dilihat pada Gambar 4.73 dibawah.



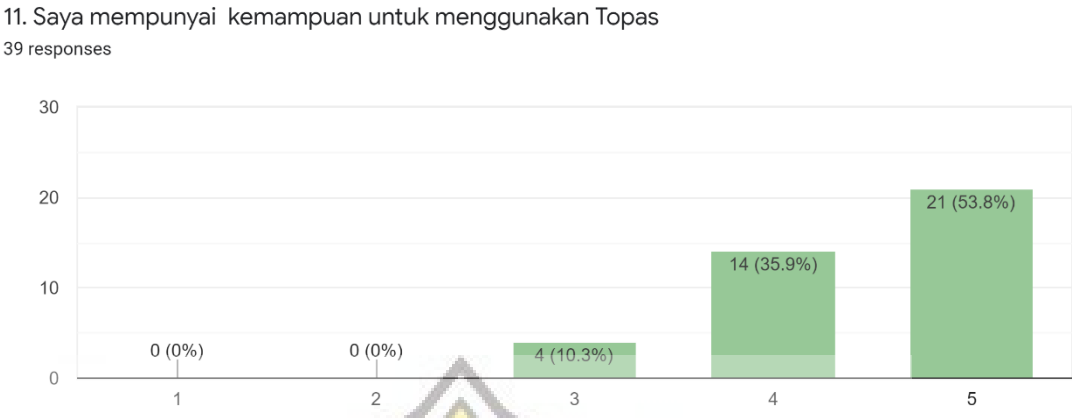
Gambar 4.73 Hasil Perangkat Mempunyai Sumber Daya Untuk Menggunakan Topas

Pernyataan kesepuluh menunjukan bahwa 43.6% sangat setuju, 43.6% setuju, dan 12.8% netral bahwa responden memiliki pengetahuan untuk menggunakan Topas. Hasil diagram responden bisa dilihat pada Gambar 4.73 dibawah.



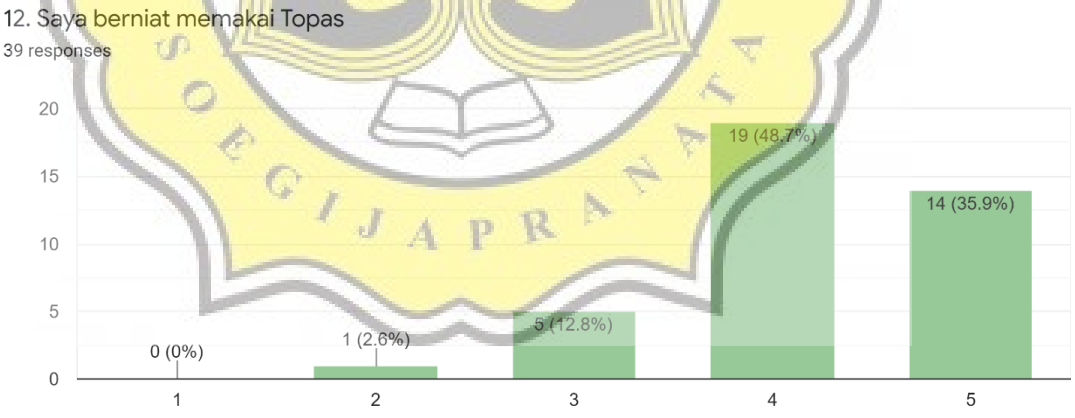
Gambar 4.73 Hasil Mempunyai Pengetahuan Untuk Menggunakan Topas

Pernyataan kesebelas menunjukan bahwa 53.8% sangat setuju, 35.9% setuju, dan 10.3% netral bahwa responden memiliki kemampuan untuk menggunakan Topas. Hasil diagram responden bisa dilihat pada Gambar 4.74 dibawah.



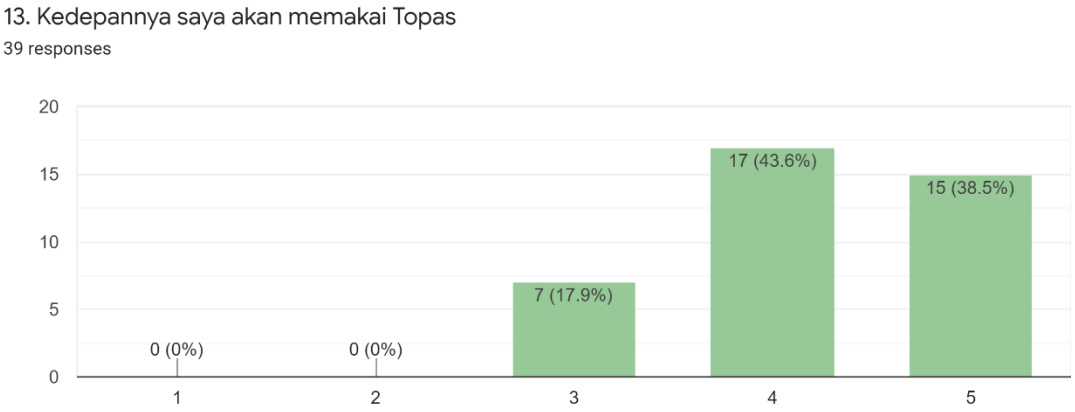
Gambar 4.74 Hasil Mempunyai Kemampuan Untuk Menggunakan Topas

Pernyataan kedua belas menunjukan bahwa 35.9% sangat setuju, 48.7% setuju, 12.3% netral, dan 2.6% tidak setuju bahwa responden memiliki niat memakai Topas. Hasil diagram responden bisa dilihat pada Gambar 4.75 dibawah.



Gambar 4.75 Hasil Berniat Memakai Topas

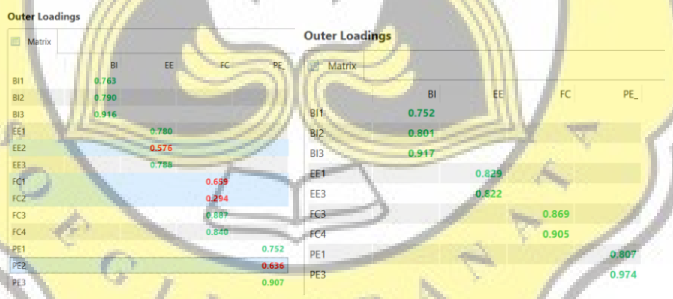
Pernyataan ketiga belas menunjukan bahwa 38.5% sangat setuju, 43.6% setuju, dan 17.9% netral bahwa kedepannya responden akan memakai Topas. Hasil diagram responden bisa dilihat pada Gambar 4.76 dibawah.



Gambar 4.76 Hasil Kedepannya Akan Memakai Topas

4.5.3 Uji Validitas

Proses uji validitas variabel dilakukan untuk mengeliminasi variabel yang tidak memiliki kepastian antar variabel yang kuat. Seperti gambar dibawah variable EE2, FC1, FC2, dan PE2 memiliki nilai dibawah 0.700. Maka dari itu variabel tersebut dihapus dari model.



Gambar 4.77 Eliminasi Variabel Tidak Valid

4.5.4 Uji Reabilitas

Cronbach's alpha digunakan untuk menguji realibilitas dari sebuah kuesioner. Tabel 4.1 dibawah merupakan penilaian cronbach's alpha.

Cronbach's alpha	*Internal consistency*
α ≥ 0.9	*Excellent*
0.9 > α ≥ 0.8	*Good*

0.8 > α ≥ 0.7	*Acceptable*
0.7 > α ≥ 0.6	*Questionable*
0.6 > α ≥ 0.5	*Poor*
0.5 > α	*Unacceptable*

Tabel 4.1 Tabel Cronbach's Alpha

Gambar 4.79 dibawah adalah hasil uji Cronbach's alpha dari 9 variabel. Hasil menunjukan angka 0.836, menurut tabel diatas hasil yang diperoleh menunjukan *Acceptable* atau dapat diterima.

Reliability Statistics

Cronbach's Alpha	N of Items
.836	9

Gambar 4.79 Cronbach's Alpha 9 Variabel

4.5.5 Uji Korelasi

Uji korelasi dilakukan untuk mencari tahu korelasi antar variabel. Hasil uji korelasi menggunakan metode *Pearson Correlation* bisa dilihat pada Tabel 4.2. Korelasi kuat ditandai dengan warna hijau dan korelasi lemah ditandai dengan warna kuning.

Hasil menunjukan variabel PE memiliki korelasi kuat terhadap variabel FC dan korelasi lemah terhadap variabel EE. Variabel EE memiliki korelasi lemah terhadap variabel PE dan FC. Variabel FC memiliki korelasi kuat dengan variabel PE dan BI. Dan Variabel BI memiliki korelasi kuat terhadap variabel FC. Adanya korelasi kuat antara variabel independen FC dan variabel dependen BI, menunjukan responden mempunyai keinginan menggunakan aplikasi topas dikarenakan adanya fasilitas yang dibutuhkan untuk mengoperasikan aplikasi.

Correlations

		TPE	TEE	TFC	TBI
TPE	Pearson Correlation	1	.339*	.556**	.312
	Sig. (2-tailed)		.050	.001	.073

	N	34	34	34	34
TEE	Pearson Correlation	.339*	1	.373*	.297
	Sig. (2-tailed)	.050		.030	.088
	N	34	34	34	34
TFC	Pearson Correlation	.556**	.373*	1	.651**
	Sig. (2-tailed)	.001	.030		.000
	N	34	34	34	34
TBI	Pearson Correlation	.312	.297	.651**	1
	Sig. (2-tailed)	.073	.088	.000	
	N	34	34	34	34

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

Tabel 4.2 Tabel Korelasi

4.4 Hasil Wawancara

Wawancara dilakukan pada 2 narasumber. Narasumber pertama merupakan toko perhiasan skala menengah dengan *staff* sebanyak 15 orang, subjek yang diwawancarai merupakan manajer toko. Dan toko perhiasan kedua dengan kecil dengan *staff* sebanyak 2 orang. subjek yang diwawancarai merupakan *co-owner*.

Respon dari narasumber pertama terhadap pertanyaan. Responden tidak mengalamai kesusahan dalam pengoprasian program dikarenakan sudah terbiasa dengan *layar sentuh*. Fitur yang dirasa paling berguna adalah penetapan harga secara *live* dan laporan penjualan. Responden memberi tanggapan ada fitur yang kurang pada program, akan lebih baik program juga menyertakan bagian kepegawaian seperti pembagian komisi penjualan. Untuk kedepannya narasumber pertama tertarik menggunakan program jika semua fitur sudah dipenuhi.

Respon dari narasumber kedua terhadap pertanyaan. Responden kedua tidak mengalami kesusahan dalam menavigasi program, setelah kembalikan ke mode *portrait*. Fitur yang dirasa paling berguna adalah setting harga emas, selain bisa melihat grafik emas dan pergerakan US Dollar dalam satu layar dan bisa mengambil keputusan dari grafik tersebut. Responden kedua tidak merasa ada fitur yang kurang dari program. Dan dalam waktu dekat belum membutuhkan program karena dirasa masih bisa menggunakan cara manual.