4. IMPLEMENTASI SISTEM

4.1.Perangkat Lunak yang Digunakan

Aplikasi mobile pada penelitian ini dibuat menggunakan Framework Flutter SDK versi 3.22.1 dengan bahasa pemrograman Dart versi 3.4.1. Website di buat dengan menggunakan framework Laravel versi 4.5.0 dan Visual Studio Code sebagai source code editor. Aplikasi dan website menggunakan laravel sebagai platform yang memberikan layanan backend services.

4.2. Langkah-langkah Implementasi

Berikut merupakan langkah-langkah yang diperlukan untuk melakukan implementasi program:

1. Melakukan instalasi terhadap perangkat lunak yang diperlukan yaitu Framework Flutter, bahasa pemrograman Dart, Bahasa pemograman python, dan framework Laravel. serta melakukan setup server dan domain.
2. Membuat tampilan user interface website menggunakan Laravel dan menggunakan Laravel sebagai backend
3. Membuat tampilan user interface aplikasi menggunakan Flutter.
4. Melakukan koneksi aplikasi Flutter ke Laravel untuk dapat mengakses database sama-sama.

4.3. Instalasi Perangkat Lunak

Pada sub bab ini akan membahas mengenai persiapan perangkat lunak yang digunakan untuk membuat program dan mengimplementasikan sistem yaitu instalasi Framework Flutter dan bahasa pemrograman Dart serta melakukan setup dan koneksi dengan laravel.

4.3.1. Instalasi Framework Flutter

Berikut merupakan langkah-langkah yang diperlukan untuk melakukan instalasi Framework Flutter:

1. Mengunduh Flutter SDK melalui website https://docs.flutter.dev/get-started/install/windows.

2. Menambahkan path Flutter SDK pada system environment > environment variables > path pada Windows.
3. Melakukan instalasi plugin Flutter pada Visual Studio Code.

4.3.2. Instalasi Framework Laravel

Berikut merupakan langkah-langkah yang diperlukan untuk melakukan instalasi Framework Laravel:

1. Mengunduh aplikasi XAMPP melalui website https://www.apachefriends.org/index.html.
2. Menginstall Composer kemudian lakukan instalasi setup melalui website https://getcomposer.org/.
3. Masuk ke dalam command prompt lalu mengetik composer global require laravel/installer.

4.4. Implementasi Program

Pada sub bab ini akan dibahas mengenai implementasi program dalam bentuk source code berdasarkan pada pembahasan dan activity diagram yang ada pada bab 3. Hubungan antara segmen program dan pembahasan activity diagram dapat dilihat pada tabel 4.1.

Nama Proses	Segmen Program	Activity Diagram
Login	Segmen Program 4.1. Segmen Program 4.2.	Gambar 3.2
Sign-up	Segmen Program 4.3.	Gambar 3.3
Edit Profile	Segmen Program 4.4. Segmen Program 4.5.	Gambar 3.4
Manage data Brand	Segmen Program 4.6. Segmen Program 4.7. Segmen Program 4.8.	Gambar 3.5
Manage data Kategori	Segmen Program 4.9. Segmen Program 4.10. Segmen Program 4.11.	Gambar 3.6
Mengatur data user	Segmen Program 4.12. Segmen Program 4.13. Segmen Program 4.14.	Gambar 3.7
Manage data supplier	Segmen Program 4.5. Segmen Program 4.5. Segmen Program 4.16. Segmen Program 4.17.	Gambar 3.8
Manage data customer		Gambar 3.8
Manage Barang		Gambar 3.9
Manage Stok Barang		Gambar 3.10

Manage Penjualan		Gambar 3.11
Manage Pembelian		Gambar 3.12
Manage Restock barang		Gambar 3.13

4.4.1. Login

Dalam fitur yang digunakan untuk login, pengguna dapat melakukan login ke aplikasi dan website dengan menggunakan email dan password yang sudah didaftarkan. Ketika fitur login dengan email dan password dijalankan, sistem akan memanggil fungsi store akan mengecek dengan email dan password yang di berikan sudah terdaftar sebelumnya. Apabila, memasukkan email atau password yang salah akan memberikan fungsi error. Fungsi login dapat dilihat pada segmen program 4.1.

Segmen Program 4.1. Fungsi login pada website

```
public function create()
    {
        return view('session.login-session');
    }

    public function store()
    {
        $attributes = request()->validate([
            'email'=>'required|email',
            'password'=>'required'
        ]);

        if(Auth::attempt($attributes))
        {
            session()->regenerate();
            if (Auth::user()->role === 'Owner') {
                return redirect('dashboard')->with(['success' => 'You are logged in.']);
            } else {
                Auth::logout();
                return redirect('/login')->withErrors(['email' => 'You do not have permission to access this
page.']);
            }
        }
        else{

            return back()->withErrors(['email'=>'Email or password invalid.']);
        }
    }

    public function destroy()
    {

        Auth::logout();
```

```
        return redirect('/login')->with(['success'=>'You\'ve been logged out.']);
    }
```

Ada juga pengecekan untuk login di aplikasi dengan menggunakan fungsi submit. Sama hal nya dengan login yang ada di website dengan apabila ada kesalahan dalam menginput email maupun password akan terjadi error. Begitu pula dengan api yang tidak tersambung dengan benar akan memunculkan error. Fungsi login aplikasi dapat dilihat pada segmen program 4.2.

Segmen Program 4.2. Fungsi login pada aplikasi

```
void _submit() async {
   final form = _formKey.currentState;
   if (form != null && form.validate()) {
    form.save();
    setState(() {
     isLoading = true;
    });
    var result = await checkLogin(_username, _password);
    print("result user " + result.toString());
    if (result != null) {
     if (result["result"] == "ok") {
      userlogin = result["user"];
      Navigator.of(context, rootNavigator: true).push(
       MaterialPageRoute(builder: (context) => MenuPage()),
      );
     } else {
      showMessageDialog(context, "Informasi", "Login gagal");
     }
    } else {
     showMessageDialog(context, "Informasi", "API gagal dihubungi");
    }
    setState(() {
     isLoading = false;
    });
   }
  }
```

4.4.2. Sign-up

Dalam fitur yang digunakan untuk sign-up, pengguna dapat melakukan sign-up di dalam website dengan memasukkan nama, email, nomor telepon, role pekerjaan dan password. Untuk fungsi sign-up ini hanya bisa dilakukan dengan email-email yang berbeda. Fungsi signup function store dapat dilihat pada segmen program 4.4.

Segmen Program 4.3. Fungsi Sign-up

```
public function store()
  {
    $attributes = request()->validate([
      'name' => ['required', 'max:50'],
      'email' => ['required', 'email', 'max:50', Rule::unique('users', 'email')],
      'phone' => ['required', 'max:50'],
      'password' => ['required', 'min:5', 'max:20'],

      'role' => ['required', Rule::in(['owner', 'karyawan'])],
    ]);
    $attributes['password'] = bcrypt($attributes['password'] );

    if (!in_array($attributes['role'], ['owner', 'karyawan'])) {
      $attributes['role'] = 'karyawan';
    }


    session()->flash('success', 'Your account has been created.');
    $user = User::create($attributes);
    Auth::login($user);
    return redirect('/dashboard');
  }
```

4.4.3. Edit Profile

Dalam fitur yang digunakan untuk edit profile, pengguna dapat melakukan edit profile ke website. Hal-hal yang bisa diubah hanya nama, email, nomor telepon, dan alamat dari user tersebut. Ketika fitur login dengan email dan password dijalankan, sistem akan memanggil fungsi store. Fungsi ini akan. Fungsi signIn dapat dilihat pada segmen program 4.4.

Segmen Program 4.4 Fungsi edit profile pada website

```
public function update(Request $request)
  {
    $user = Auth::user();

    $attributes = $request->validate([
      'name' => ['required', 'max:50'],
      'email' => [
        'required',
        'email',
        'max:50',
        Rule::unique('users')->ignore($user->id), // Gunakan id pengguna yang sedang masuk
      ],
      'phone' => ['required', 'max:50'],
      'location' => ['max:50'],
    ]);
```

```
      if ($request->email != $user->email) {
         if (env('IS_DEMO') && $user->id == 1) {
            return redirect()->back()->withErrors(['msg2' => 'You are in a demo version, you can\'t
change the email address.']);
         }
      }


      $user->name = $attributes['name'];
      $user->email = $attributes['email'];
      $user->phone = $attributes['phone'];
      $user->location = $attributes['location'];

      $user->save();

      return redirect('/profile')->with('success', 'Profile updated successfully');
   }
```

Pengguna dapat melakukan edit profile ke didalam aplikasi. Hal-hal yang bisa diubah hanya nama, email, nomor telepon, dan password. Fitur loadDataUser untuk mengambil data yang sebelumnya sudah ada dan fitur saveDataUser untuk dalam proses untuk mengubah data. Fitur GantiPassword untuk mengubah password dengan melakukan pengecekan password sama dengan sebelumnya. Dan harus memberikan password yang baru dan melakukan konfirmasi dengan password yang sama dengan password baru. Fungsi loadDataUser, saveDataUSer, dan GantiPassword dapat dilihat pada segmen program 4.5.

Segmen Program 4.5 Fungsi edit profile pada aplikasi

```
loadDataUser() async {
   var user = await loadUser();
   if (user != null) {
     userProfile = user['user'];
     _nameController.text = userProfile['name'];
     _emailController.text = userProfile['email'];
     _phoneController.text = userProfile['phone'];

     setState(() {});
   }
  }
saveDataUser() async {
   var name = _nameController.text; // = userProfile['name'];
   var email = _emailController.text; // = userProfile['email'];
   var phone = _phoneController.text; // = userProfile['phone'];
   var result = await saveUser(userProfile['id'], name, email, phone);
   if (result["result"] == "ok") {
     showMessageDialog(context, "Informasi", "Profile berhasil diupdate");
   } else {
     showMessageDialog(
        context, "Informasi", "Email yang anda pakai sudah digunakan");
   }
```

```
  }
class _GantiPasswordState extends State<GantiPassword> {
  final _formKey = GlobalKey<FormState>();
  late TextEditingController _passwordController;
  late TextEditingController _passwordController1;
  late TextEditingController _passwordController2;

  //var userProfile = null;
  @override
  void initState() {
    _passwordController = TextEditingController();
    _passwordController1 = TextEditingController();
    _passwordController2 = TextEditingController();
    super.initState();
  }

  @override
  void dispose() {
    super.dispose();
  }

  updatePassword() async {
    //userProfile = user['user'];
    var passwordlama = _passwordController.text; // = userProfile['name'];
    var password1 = _passwordController1.text; // = userProfile['email'];
    var password2 = _passwordController2.text; // = userProfile['phone'];
    var result = await gantiPassword(userlogin['id'], passwordlama, password1);
    print("result " + result["result"]);
    if (result["result"] == "ok") {
      showMessageDialog(context, "Informasi", "Password berhasil diupdate", () {
        Navigator.pop(context);
      });
      //Navigator.pop(context);
    } else {
      showMessageDialog(
          context, "Informasi", "Password lama yang anda masukkan salah");
    }

    //print(user.toString());
  }
```

4.4.4. Manage Data Brand

Dalam fitur yang digunakan untuk manage data brand, pengguna dapat melihat, melakukan menambahkan dan menghapus nama brand di dalam website. Fungsi index ini untuk menampilkan seluruh data brand yang sudah terdaftar di dalam database. Fungsi melihat seluruh data brand dapat dilihat pada segmen program 4.6.

Segmen Program 4.6. Fungsi melihat seluruh data brand

```
public function index(Request $request)
  {
    $search = $request->input('search');

    $brand = Brand::when($search, function ($query) use ($search) {
      return $query->where('name_brand', 'like', "%{$search}%");
    })->paginate(10);

    return view('brand.index', compact('brand'));
  }
```

Sedangkan untuk menambahkan data brand memerlukan fungsi store. Fungsi store disini untuk menambahkan data untuk nama brand yang baru. Penambahan disini tidak bisa dilakukan dengan nama yang sama walaupun hanya beda huruf besar dan kecil. Fungsi tambah data brand dapat dilihat pada segmen program 4.7.

Segmen Program 4.7. Fungsi penambahan data brand

```
public function create()
  {
    return view('brand.create');
  }
public function store(Request $request)
  {
    $validatedData = $request->validate([
      'name_brand' => [
        'required',
        'max:50',
        Rule::unique('brand')->where(function ($query) {
          return $query->whereRaw('LOWER(name_brand) = LOWER(?)', [request('name_brand')]);
        })
      ]
    ]);

    brand::create($validatedData);

    return redirect('/brand');
  }
```

Sedangkan untuk menghapus data brand memerlukan fungsi destroy. Tetapi, tidak bisa menghapus apabila brand sudah digunakan dari tabel barang. Fungsi hapus data brand dapat dilihat pada segmen program 4.8.

Segmen Program 4.8. Fungsi hapus data brand

```
public function destroy(brand $brand)
   {
      try {
         brand::destroy($brand->id_brand);
         return redirect('/brand')->with('success', 'Brand has been deleted!');
      } catch (\Illuminate\Database\QueryException $e) {
         return redirect('/brand')->withErrors('Tidak dapat menghapus brand karena masih terkait
dengan tabel barang.');
      }
   }
```

4.4.5. Manage Data Kategori

Dalam fitur yang digunakan untuk manage data kategori, hampir sama dengan manage data brand. Pengguna dapat melihat, melakukan menambahkan dan menghapus nama kategori di dalam website. Fungsi index ini untuk menampilkan seluruh data kategori yang sudah terdaftar di dalam database. Dan fungsi terakhir untuk destroy untuk menghapus data kategori yang ada. Tetapi, tidak bisa menghapus apabila kategori sudah digunakan dari tabel barang. Fungsi manage data kategori dapat dilihat pada segmen program 4.9.

Segmen Program 4.9. Fungsi melihat seluruh data kategori

```
public function index(Request $request)
   {
      $search = $request->input('search');

      $kategori = kategori::when($search, function ($query) use ($search) {
         return $query->where('name_kategori', 'like', "%{$search}%");
      })->paginate(10);

      return view('kategori.index', compact('kategori'));
   }
```

Sedangkan fungsi untuk menambahkan data kategori memerlukan fungsi store. Hampir sama dengan manage data brand. Fungsi store disini untuk menambahkan data untuk nama kategori tidak bisa menambahkan data kategori yang sama walaupun dengan menggunakan huruf besar dan kecil yang berbeda. Fungsi penambahan kategori dapat dilihat dari segmen program 4.10.

Segmen Program 4.10. Fungsi penambahan data kategori

```
public function create()
   {
      return view('kategori.create');
   }
public function store(Request $request)
   {
      // Validasi input dari form
      $validatedData = $request->validate([
         'name_kategori' => [
            'required',
            'max:50',
            Rule::unique('kategori')->where(function ($query) {
               return $query->whereRaw('LOWER(name_kategori) = LOWER(?)',
[request('name_kategori')]);
            })
         ]
      ]);
      kategori::create($validatedData);
      // Buat instance baru dari model Brand dan simpan data


      // Redirect ke halaman yang sesuai setelah menyimpan data
      return redirect('/category');
   }
```

Sedangkan untuk menghapus data kategori memerlukan fungsi destroy. Tetapi, tidak bisa menghapus apabila kategori sudah digunakan dari tabel barang. Fungsi hapus data kategori dapat dilihat pada segmen program 4.11.

Segmen Program 4.11. Fungsi penghapusan data kategori

```
public function destroy(kategori $category)
   {

      try {
         kategori::destroy($category->id_kategori);
         return redirect('/category')->with('success', 'category has been deleted!');
      } catch (\Illuminate\Database\QueryException $e) {
         return redirect('/category')->withErrors('Tidak dapat menghapus kategori karena masih terkait
dengan tabel barang.');
      }
   }
```

4.4.6. Manage Data User

Dalam fitur yang digunakan untuk manage data user, pengguna dapat melihat seluruh data user, menambahkan data user, dan menghapus data user. Fungsi index untuk mengambil data supplier yang sudah terdaftar. Fungsi melihat seluruh data user dapat dilihat pada segmen program 4.12.

Segmen Program 4.112. Fungsi untuk melihat seluruh data user

```
public function index()
   {
      $users = User::with('toko')->get(); // 'toko' sesuai dengan nama method relasi di model User

      return view('user.data-user', compact('users'));
   }
```

Sedangkan fungsi untuk menambahkan data user memerlukan fungsi store. Fungsi store hampir sama dengan registrasi. Memerlukan beberapa hal untuk menambahkan data seperti nama user, email user, alamat user,nomor telepon user, role di toko, nama toko, dan juga password. Fungsi penambahan data user dapat dilihat dari segmen program 4.13.

Segmen Program 4.13. Fungsi untuk menambahkan data user

```
   public function create()
   {
      $toko = Toko::all(); // Mengambil semua data toko
      return view('user.create', compact('toko'));
   }

   /**
    * Store a newly created resource in storage.
    */
   public function store(Request $request)
   {
       $attributes = request()->validate([
         'name' => ['required', 'max:50'],
         'email' => ['required', 'email', 'max:50', Rule::unique('users')->ignore(Auth::user()->id)],
         'location' => ['max:255'],
         'phone'    => ['required', 'max:50'],
         'password' => ['required', 'min:5', 'max:20'],
         'role' => ['required', Rule::in(['Owner', 'Karyawan'])],
         'id_toko' => ['required'],
      ]);
      if($request->get('email') != Auth::user()->email)
      {
         if(env('IS_DEMO') && Auth::user()->id == 1)
         {
```

```
            return redirect()->back()->withErrors(['msg2' => 'You are in a demo version, you can\'t
change the email address.']);
        }

    }
    else{
        $attributes = request()->validate([
            'email' => ['required', 'email', 'max:50', Rule::unique('users')->ignore(Auth::user()->id)],
        ]);
    }

    $attributes['password'] = bcrypt($attributes['password'] );

    if (!in_array($attributes['role'], ['Owner', 'Karyawan'])) {
        $attributes['role'] = 'Karyawan';
    }

    session()->flash('success', 'Account has been created.');
    User::create($attributes);
    return redirect('/data-user')->with('toko', Toko::all());
  }
```

Sedangkan untuk menghapus data user memerlukan fungsi destroy. Tetapi, tidak bisa menghapus apabila user sudah digunakan dari tabel transaksi. Fungsi hapus data user dapat dilihat pada segmen program 4.14.

Segmen Program 4.14. Fungsi untuk menghapus data user

```
public function destroy($id)
  {
    // Find the user by ID
    $user = User::find($id);

    if ($user) { // If user is found
        // Check if user is linked to any transactions
        $relatedSalesTransactions = transaksi_jual::where('id', $user->id)->count();
        $relatedSupplierTransactions = transaksi_supplier::where('id', $user->id)->count();

        if ($relatedSalesTransactions > 0 || $relatedSupplierTransactions > 0) {
            return redirect('/data-user')->withErrors('User cannot be deleted because it is still related to
sales or supplier transactions.');
        }

        try {
            // Attempt to delete the user
            $user->delete();

            // Redirect with success message
            return redirect('/data-user')->with('success', 'User has been deleted successfully!');
        } catch (\Illuminate\Database\QueryException $e) {
            // Handle error, for instance, due to a foreign key constraint
            return redirect('/data-user')->withErrors('Failed to delete user due to database relations.');
```

```
        }
    } else {
        // If user is not found
        return redirect('/data-user')->withErrors('User not found.');
    }
}
```

4.4.7. Manage Data Supplier

Dalam fitur yang digunakan untuk manage data supplier, pengguna dapat melihat seluruh data supplier, menambahkan data supplier, mengedit data supplier, dan menghapus data supplier. Fungsi index untuk mengambil data supplier yang sudah terdaftar. Fungsi melihat seluruh data supplier dapat dilihat pada segmen program 4.15.

Segmen Program 4.15. Fungsi melihat seluruh data supplier

```
public function index()
  {
    return view('supplier.index',[
      'supplier' => supplier::all()
    ]);
  }
```

Fungsi store untuk menambahkan data dengan memasukkan nama, email, nomor telepon, dan alamat dari supplier, dalam membuat data supplier hanya bisa satu email dan satu data supplier. Fungsi menambahkan data supplier dapat dilihat pada segmen program 4.16.

Segmen Program 4.16 Fungsi menambahkan data supplier

```
public function create()
  {
    return view('supplier.create');
  }
public function store(Request $request)
  {
    $ValidatedData = $request->validate([
      'name_supplier' => ['required', 'max:50'],
      'supplier_email' => ['required', 'email', 'max:50',Rule::unique('supplier')],
      'supplier_phone'    => ['required', 'max:50'],
      'supplier_address' => ['max:50'],
    ]);
    supplier::create($ValidatedData);
    // Buat instance baru dari model supplier dan simpan data
```

```
    // Redirect ke halaman yang sesuai setelah menyimpan data
    return redirect('/supplier');
  }
```

Fungsi update untuk mengubah data supplier apabila ada perubahan dari data supplier dari nama, email, nomor telepon, dan alamat dari supplier. Fungsi mengedit data supplier dapat dilihat pada segmen program 4.17.

Segmen Program 4.17. Fungsi mengedit data supplier

```
public function edit(supplier $supplier)
  {
    {
      return view('supplier.edit',[
        'supplier' => $supplier,
      ]);
    }
  }
public function update(Request $request, Supplier $supplier)
  {
    $validatedData = $request->validate([
      'name_supplier' => ['required', 'max:50'],
      'supplier_email' => [
        'required',
        'email',
        'max:50',
        Rule::unique('supplier')->ignore($supplier->id_supplier, 'id_supplier'),
      ],
      'supplier_phone' => ['required', 'max:50'],
      'supplier_address' => ['max:255'], // Atau sesuaikan maksimal karakter sesuai kebutuhan
    ]);

    // Ambil data asli dari database
    $originalData = $supplier->toArray();

    // Menggabungkan data asli dan data baru untuk perbandingan
    $newData = array_merge($originalData, $validatedData);

    // Periksa apakah ada perubahan antara data asli dan data baru
    $changes = array_diff_assoc($newData, $originalData);

    // Jika tidak ada perubahan, kembalikan dengan pesan tidak ada perubahan
    if (empty($changes)) {
      return redirect('/supplier')->with('info', 'No changes were made to the supplier.');
    }

    // Update supplier dengan data yang valid
    $supplier->update($validatedData);
```

```
    // Redirect setelah update
    return redirect('/supplier')->with('success', 'Supplier updated successfully.');
  }
```

Fungsi destroy untuk menghapus data supplier apabila ada kesalahan input. Menghapus data supplier tidak bisa dilakukan apabila data supplier sudah pernah digunakan untuk transaksi pembelian. Fungsi menghapus data supplier dapat dilihat dari segmen program 4.18.

Segmen Program 4.18 Fungsi menghapus data supplier

```
public function destroy($id_supplier)
  {
    $supplier = supplier::find($id_supplier);
    if ($supplier) {
      $supplier->delete();
      return redirect('supplier')->with('success', 'supplier has been deleted!');
    } else {
      // Handle case when user not found
      return redirect('supplier')->withErrors('supplier not found.');
    }
  }
```

4.4.8. Manage Data Customer

Dalam fitur yang digunakan untuk manage data customer, fitur yang dimiliki hampir sama dengan fitur-fitur yang ada untuk manage supplier. Pengguna dapat melihat seluruh data customer, menambahkan data customer, mengedit data customer, dan menghapus data customer. Fungsi index untuk mengambil data customer yang sudah terdaftar. Fungsi melihat seluruh data customer dapat dilihat pada segmen program 4.19.

Segmen Program 4.19. Fungsi melihat seluruh data customer

```
public function index()
  {
    return view('customer.index',[
      'customer' => customer::all()
    ]);
  }
```

Fungsi store untuk menambahkan data dengan memasukkan nama, email, nomor telepon, dan alamat dari customer, dalam membuat data customer hanya bisa satu email dan satu data customer. Fungsi menambahkan data supplier dapat dilihat pada segmen program 4.20.

Segmen Program 4.20. Fungsi menambahkan data customer

```
public function create()
   {
      return view('customer.create');
   }
public function store(Request $request)
   {
      $ValidatedData = $request->validate([
         'name_customer' => ['required', 'max:50'],
         'customer_email' => ['required', 'email', 'max:50',Rule::unique('customer')],
         'customer_phone'     => ['required', 'max:50'],
         'customer_address' => ['max:50'],
      ]);
      customer::create($ValidatedData);
      // Buat instance baru dari model Brand dan simpan data


      // Redirect ke halaman yang sesuai setelah menyimpan data
      return redirect('/customer');
   }
```

Fungsi update untuk mengubah data supplier apabila ada perubahan dari data customer dari nama, email, nomor telepon, dan alamat dari customer. Fungsi mengedit data supplier dapat dilihat pada segmen program 4.21.

Segmen Program 4.21. Fungsi mengedit data customer

```
public function edit(customer $customer)
   {
      return view('customer.edit',[
         'customer' => $customer,
      ]);

   }
public function update(Request $request, customer $customer)
   {
      // Validasi input form
      $validatedData = $request->validate([
         'name_customer' => ['required', 'max:50'],
         'customer_email' => [
            'required',
```

```
            'email',
            'max:50',
            // Pastikan email unik di tabel customer kecuali untuk customer yang sedang diupdate
            Rule::unique('customer')->ignore($customer->id_customer, 'id_customer'),
        ],
        'customer_phone' => ['required', 'max:50'],
        'customer_address' => ['max:255'], // Atau sesuaikan maksimal karakter sesuai kebutuhan
    ]);

    // Ambil data asli dari database
    $originalData = $customer->toArray();

    // Menggabungkan data asli dan data baru untuk perbandingan
    $newData = array_merge($originalData, $validatedData);

    // Periksa apakah ada perubahan antara data asli dan data baru
    $changes = array_diff_assoc($newData, $originalData);

    // Jika tidak ada perubahan, kembalikan dengan pesan tidak ada perubahan
    if (empty($changes)) {
        return redirect('/customer')->with('info', 'No changes were made to the customer.');
    }

    // Update customer dengan data yang valid
    $customer->update($validatedData);

    // Redirect setelah update
    return redirect('/customer')->with('success', 'Customer updated successfully.');
}
```

Fungsi destroy untuk menghapus data customer apabila ada kesalahan input. Menghapus data customer tidak bisa dilakukan apabila data customer sudah pernah digunakan untuk transaksi penjualan. Fungsi menghapus data supplier dapat dilihat pada segmen program 4.22.

Segmen Program 4.22. Fungsi menghapus data customer

```
public function destroy($id_customer)
    {
        $customer = customer::find($id_customer);
        if ($customer) {
            $customer->delete();
            return redirect('customer')->with('success', 'customer has been deleted!');
        } else {
            // Handle case when user not found
            return redirect('customer')->withErrors('customer not found.');
        }
    }
```

4.4.9. Manage Data Barang

Dalam fitur yang digunakan untuk manage data barang, pengguna dapat melihat data barang, melihat seluruh data barang, menambahkan data barang, mengedit data barang, menghapus data barang. Fungsi index untuk melihat seluruh data barang yang sudah terdaftar. Fungsi melihat seluruh barang dapat dilihat pada segmen program 4.23.

Segmen Program 4.23. Fungsi melihat seluruh barang

```
public function index(Request $request)
  {
    try {
      // Ambil semua barang dengan relasi kategori dan brand
      $query = Barang::query();

      // Filter berdasarkan kategori
      if ($request->filled('kategori')) {
        $query->where('id_kategori', $request->kategori);
      }

      // Filter berdasarkan brand
      if ($request->filled('brand')) {
        $query->where('id_brand', $request->brand);
      }

      // Lakukan pencarian berdasarkan nama atau kode barang
      if ($request->filled('search')) {
        $search = $request->search;
        $query->where(function ($q) use ($search) {
          $q->where('kode_barang', 'like', "%$search%")
            ->orWhere('name_barang', 'like', "%$search%");
        });
      }

      // Ambil semua kategori dan brand
      $kategori = Kategori::all();
      $brand = Brand::all();

      // Ambil hasil pencarian
      $barang = $query->with('kategori', 'brand')->get();

      // Kembalikan view dengan data barang, kategori, dan brand
      return view('barang.index', compact('barang', 'kategori', 'brand'));
    } catch (\Exception $e) {
      // Tangani exception jika terjadi kesalahan
      return redirect('/product')->withErrors('Failed to fetch data: ' . $e->getMessage());
    }
  }
```

Fungsi store untuk menambahkan data barang dengan memasukkan kode barang, nama barang, nama kategori yang sudah terdaftar, nama brand yang sudah terdaftar, harga barang,

Gambar barang dan deskripsi barang. Dalam membuat data barang hanya bisa satu kode barang untuk satu barang tidak bisa lebih dari satu. Fungsi menambahkan data supplier dapat dilihat pada segmen program 4.24.

Segmen Program 4.24. Fungsi penambahan data barang

```
public function create()
    {
        $categories = Kategori::all();
        $brands = Brand::all();

        return view('barang.create', compact('categories', 'brands'));
    }

    public function store(Request $request)
    {

        // Validasi input form
        $validatedData = $request->validate([
            'kode_barang' => ['required', 'max:20', Rule::unique('barang')],
            'name_barang' => ['required', 'max:50'],
            'id_kategori' => ['required'],
            'id_brand'=> ['required'],
            'harga_jual' => ['numeric', 'integer', 'min:0'],
            'image' => ['image', 'max:1024'],
            'barang_description' => ['max:10000'],
        ]);

        if ($request->file('image')) {
            // Menyimpan file yang diupload ke dalam variabel $image
            $validatedData['image'] = $request->file('image')->store('product-images');
        }



        $validatedData["stok"]=10;
        $validatedData["nilai"]=0;

        barang::create($validatedData);

        return redirect('/product');
    }
```

Fungsi detail untuk melihat data barang lebih spesifik. Atribut yang di tampilkan adalah kode barang, nama barang, nama kategori, nama brand, harga barang, Gambar barang dan deskripsi barang. Fungsi melihat detail data barang dapat dilihat pada segmen program 4.25.

Segmen Program 4.25. Fungsi melihat detail data barang

```
public function detail($kode_barang)
   {
      try {
         // Ambil barang berdasarkan kode_barang dengan relasi kategori dan brand
         $barang = Barang::with('kategori', 'brand')->where('kode_barang', $kode_barang)-
>firstOrFail();

         // Kembalikan view dengan data barang
         return view('barang.show', compact('barang'));
      } catch (ModelNotFoundException $e) {
         // Tangani exception jika barang tidak ditemukan
         return redirect('/product')->withErrors('Product not found.');
      } catch (\Exception $e) {
         // Tangani exception lainnya jika terjadi kesalahan
         return redirect('/product')->withErrors('Failed to fetch data: ' . $e->getMessage());
      }
   }
```

Fungsi update untuk mengedit data barang dengan memunculkan data barang yang sudah terdaftar. Dalam fungsi ini pengguna dapat mengubah harga jual, harga supplier, Gambar barang, dan deskripsi barang. Fungsi mengedit data barang dapat dilihat pada segmen program 4.26.

Segmen Program 4.26. Fungsi mengedit data barang

```
public function edit(Barang $barang,$kode_barang)
   {
      // Mengambil semua kategori dan brand
      $kategori = kategori::all();
      $brand = brand::all();
      $barang = barang::find($kode_barang);

      // Mengembalikan view edit dengan data barang, kategori, dan brand
      return view('barang.edit', compact('barang', 'kategori', 'brand'));
   }

   public function update(Request $request, Barang $barang, $kode_barang)
   {
      // Ambil data asli dari database
      $barang = Barang::findOrFail($kode_barang);

      // Validasi input form
      $validatedData = $request->validate([
         'harga_jual' => ['numeric', 'integer', 'min:0'],
         'harga_supplier' => ['numeric', 'integer', 'min:0'],
         'image' => ['image', 'max:1024'],
         'barang_description' => ['max:10000'],
      ]);
```

```
    // Menyimpan Gambar hanya jika ada file yang diunggah
    if ($request->file('image')) {
        $validatedData['image'] = $request->file('image')->store('product-images');
    }

    // Menggabungkan data asli dan data baru untuk perbandingan
    $originalData = $barang->toArray();
    $newData = array_merge($originalData, $validatedData);

    // Periksa apakah ada perubahan antara data asli dan data baru
    $changes = array_diff_assoc($newData, $originalData);

    // Jika tidak ada perubahan, kembalikan dengan pesan tidak ada perubahan
    if (empty($changes)) {
        return redirect('/product')->with('info', 'No changes were made to the product.');
    }

    // Mengisi model barang dengan data yang valid
    $barang->fill($validatedData);

    // Simpan data barang ke dalam database
    $barang->save();

    return redirect('/product')->with('success', 'Product has been updated.');
  }
```

Fungsi destroy untuk menghapus data barang apabila ada kesalahan input. Menghapus data barang tidak bisa dilakukan apabila data barang sudah pernah digunakan untuk tabel stok. Fungsi menghapus data supplier dapat dilihat pada segmen program 4.27.

Segmen Program 4.27. Fungsi menghapus data barang

```
public function destroy($kode_barang)
  {
    // Cari barang berdasarkan kode_barang
    $barang = barang::find($kode_barang);

    if ($barang) { // Jika barang ditemukan
        try {
            // Hapus barang
            $barang->delete();

            // Redirect dengan pesan sukses
            return redirect('/product')->with('success', 'Product has been deleted!');
        } catch (\Illuminate\Database\QueryException $e) {
            // Tangani error, misal karena constraint foreign key
            return redirect('/product')->withErrors('Tidak dapat menghapus product karena masih
terkait dengan tabel stok');
        }
```

```
    } else {
      // Barang tidak ditemukan
      return redirect('/product')->withErrors('Product not found.');
    }
```

Dalam fitur yang digunakan untuk manage data barang untuk aplikasi. Fitur yang tersedia di aplikasi adalah melihat seluruh data barang, menambahkan ,dan mengedit data barang. Fungsi _BarangState untuk melihat seluruh data barang yang sudah terdaftar. Fungsi Barang untuk menampilkan seluruh data barang di aplikasi dapat dilihat pada segmen program 4.28.

Segmen Program 4.28. Fungsi menampilkan seluruh data barang di aplikasi

```
class _BarangState extends State<BarangPage> {
  bool isLoading = false;

  var daftarbarang = [];
  void callloadbarang() async {
    isLoading = true;
    setState(() {});
    var result = await loadbarang();
    if (result != null) {
      daftarbarang = result["barang"];
      print("daftarbarang");
      print(daftarbarang);
    }
    isLoading = false;
    setState(() {});
  }

  @override
  void initState() {
    super.initState();
    callloadbarang();
  }
```

Fungsi _TambahBarangState untuk menambahkan data barang. Atribut yang bisa ditambahkan nama kategori, nama brand, nama barang, kode barang, dan Gambar barang. Fungsi Barang untuk menambahkan data barang di aplikasi dapat dilihat pada segmen program 4.29.

Segmen Program 4.29. Fungsi barang untuk menambahkan data barang di aplikasi

```
class _TambahBarangState extends State<TambahBarang> {
  final _formKey = GlobalKey<FormState>();
```

```
dynamic? _selectedCategory;
dynamic? _selectedBrand;
String _itemName = '';
String _itemKode = "";

BuildContext? tcontext;

List<dynamic> _categories = [];
List<dynamic> _brands = [];

var isLoading = false;
@override
void initState() {
 super.initState();

 loadDataUser();
}

@override
void dispose() {
 super.dispose();
}

loadDataUser() async {
 var data = await getKategoriBrand();
 _categories = data["kategori"];
 _brands = data["brand"];

 if (_categories.length > 0) {
  _selectedCategory = _categories[0];
 }
 if (_brands.length > 0) {
  _selectedBrand = _brands[0];
 }
 setState(() {});
 //print(user.toString());
}

Future<void> _uploadImage(File image) async {
 final url = Uri.parse(baseURL + "tambahbarang");

 var request = http.MultipartRequest('POST', url)
  ..fields['name'] = _itemName
  ..fields['category'] = _selectedCategory['id_kategori'].toString()
  ..fields['brand'] = _selectedBrand['id_brand'].toString()
  ..fields['kode'] = _itemKode
  ..files.add(await http.MultipartFile.fromPath(
   'image',
   image.path,
   contentType: MediaType('image', lookupMimeType(image.path)!),
  ));
 try {
  var response = await request.send();
  String responseBody = await response.stream.bytesToString();
```

```
    print("result upload " + responseBody);
    if (response.statusCode == 200) {
      ScaffoldMessenger.of(tcontext!).showSnackBar(
        SnackBar(content: Text('Barang berhasil ditambah')),
      );

      if (widget.callback != null) {
        widget.callback!();
      }
      Navigator.pop(tcontext!);
    } else {
      ScaffoldMessenger.of(tcontext!).showSnackBar(
        SnackBar(content: Text('Barang gagal ditambah')),
      );
    }
  } on http.ClientException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('ClientException: $e')),
    );
  } on SocketException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('SocketException: $e')),
    );
  } on TimeoutException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('TimeoutException: $e')),
    );
  } catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('An unexpected error occurred: $e')),
    );
  }
}

File? _selectedImage;

Future<void> _pickImage() async {
  DeviceInfoPlugin deviceInfo = DeviceInfoPlugin();
  AndroidDeviceInfo androidInfo = await deviceInfo.androidInfo;

  bool isStoragePermission = true;
  //bool isVideosPermission = true;
  bool isPhotosPermission = true;
  if (androidInfo.version.sdkInt >= 33) {
    //isVideosPermission = await Permission.videos.status.isGranted;
    isPhotosPermission = await Permission.photos.status.isGranted;
  } else {
    isStoragePermission = await Permission.storage.status.isGranted;
  }

  if (isStoragePermission && isPhotosPermission) {
    FilePickerResult? result =
        await FilePicker.platform.pickFiles(type: FileType.image);
```

```
    if (result != null) {
      setState(() {
        _selectedImage = File(result.files.single.path!);
      });
    }
  } else {
    // Open app settings so the user can manually grant the permission
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(
        content: Text(
            'Media image access permission is permanently denied. Please grant permission in settings.'),
        action: SnackBarAction(
          label: 'Settings',
          onPressed: () {
            openAppSettings();
          },
        ),
      ),
    );
  }
}
```

Fungsi _EditBarangState untuk menambahkan data barang. Atribut yang bisa di ubah adalah nama kategori, nama brand, nama barang, dan Gambar barang. Fungsi Barang untuk mengedit data barang di aplikasi dapat dilihat pada segmen program 4.30.

Segmen Program 4.30. Fungsi barang untuk mengedit data barang di aplikasi

```
class _EditBarangState extends State<EditBarang> {
  final _formKey = GlobalKey<FormState>();
  dynamic? _selectedCategory;
  dynamic? _selectedBrand;
  String _itemName = ""; //= widget.item['name_barang'];
  String _itemKode = ""; // = widget.item['kode_barang'];
  late TextEditingController _nameController;
  late TextEditingController _kodeController;

  BuildContext? tcontext;

  List<dynamic> _categories = [];
  List<dynamic> _brands = [];

  var isLoading = false;
  @override
  void initState() {
    super.initState();

    loadDataUser();

    _nameController = TextEditingController(text: widget.item['name_barang']);
    _kodeController = TextEditingController(text: widget.item['kode_barang']);
```

```
}

@override
void dispose() {
 super.dispose();
}

loadDataUser() async {
 var data = await getKategoriBrand();
 _categories = data["kategori"];
 _brands = data["brand"];

 if (_categories.length > 0) {
  _selectedCategory = _categories[0];
 }
 if (_brands.length > 0) {
  _selectedBrand = _brands[0];
 }

 _itemName = widget.item["name_barang"]; //= widget.item['name_barang'];
 _itemKode = widget.item["kode_barang"]; // = widget.item['kode_barang'];

 for (var i = 0; i < _categories.length; i++) {
  if (_categories[i]["id_kategori"] == widget.item["id_kategori"]) {
   _selectedCategory = _categories[i];
  }
 }

 for (var i = 0; i < _brands.length; i++) {
  if (_brands[i]["id_brand"] == widget.item["id_brand"]) {
   _selectedBrand = _brands[i];
  }
 }

 setState(() {});
 //print(user.toString());
}

Future<void> _uploadImage(File? image) async {
 print("step1");
 final url = Uri.parse(baseURL + "editbarang");

 var request = http.MultipartRequest('POST', url)
  ..fields['name'] = _nameController.text
  ..fields['category'] = _selectedCategory['id_kategori'].toString()
  ..fields['brand'] = _selectedBrand['id_brand'].toString()
  ..fields['kode'] = _itemKode
  ..fields["id"] = widget.item["kode_barang"];

 if (image != null) {
  request.files.add(await http.MultipartFile.fromPath(
   'image',
   image.path,
   contentType: MediaType('image', lookupMimeType(image.path)!),
```

```
    ));
  }

  try {
    var response = await request.send();
    String responseBody = await response.stream.bytesToString();

    print("result upload " + responseBody);
    if (response.statusCode == 200) {
      ScaffoldMessenger.of(tcontext!).showSnackBar(
        SnackBar(content: Text('Barang berhasil diedit')),
      );

      if (widget.callback != null) {
        widget.callback!();
      }
      Navigator.pop(tcontext!);
    } else {
      ScaffoldMessenger.of(tcontext!).showSnackBar(
        SnackBar(content: Text('Barang gagal ditambah')),
      );
    }
  } on http.ClientException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('ClientException: $e')),
    );
  } on SocketException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('SocketException: $e')),
    );
  } on TimeoutException catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('TimeoutException: $e')),
    );
  } catch (e) {
    ScaffoldMessenger.of(tcontext!).showSnackBar(
      SnackBar(content: Text('An unexpected error occurred: $e')),
    );
  }
}

File? _selectedImage;

Future<void> _pickImage() async {
  DeviceInfoPlugin deviceInfo = DeviceInfoPlugin();
  AndroidDeviceInfo androidInfo = await deviceInfo.androidInfo;

  bool isStoragePermission = true;
  //bool isVideosPermission = true;
  bool isPhotosPermission = true;
  if (androidInfo.version.sdkInt >= 33) {
    //isVideosPermission = await Permission.videos.status.isGranted;
    isPhotosPermission = await Permission.photos.status.isGranted;
  } else {
```

```
    isStoragePermission = await Permission.storage.status.isGranted;
   }

   if (isStoragePermission && isPhotosPermission) {
    FilePickerResult? result =
        await FilePicker.platform.pickFiles(type: FileType.image);
    if (result != null) {
     setState(() {
      _selectedImage = File(result.files.single.path!);
     });
    }
   } else {
    // Open app settings so the user can manually grant the permission
    ScaffoldMessenger.of(tcontext!).showSnackBar(
     SnackBar(
      content: Text(
          'Media image access permission is permanently denied. Please grant permission in settings.'),
      action: SnackBarAction(
       label: 'Settings',
       onPressed: () {
        openAppSettings();
       },
      ),
     ),
    );
   }
  }
```

4.4.10. Manage Data Stok Barang

Dalam fitur yang digunakan untuk manage data stok barang, pengguna dapat melihat data stok barang, melihat seluruh data stok barang, menambahkan data stok barang, mengedit data stok barang, menghapus datastok barang. Fungsi index untuk melihat seluruh data stok barang yang sudah terdaftar. Fungsi melihat seluruh stok barang dapat dilihat pada segmen program 4.31.

Segmen Program 4.31. Fungsi Melihat seluruh data stok barang

```
public function index(Request $request)
   {
      $query = Stok::query()->with('barang', 'toko');

      if ($request->filled('toko')) {
         $query->where('id_toko', $request->toko);
      }

      if ($request->filled('search')) {
```

```
        $search = $request->search;
        $query->whereHas('barang', function ($q) use ($search) {
            $q->where('name_barang', 'like', "%{$search}%")
              ->orWhere('kode_barang', 'like', "%{$search}%");
        });
    }

    $stok = $query->get();
    $toko = Toko::all();
    $barang = barang::all();

    return view('stok.index', compact('stok', 'toko','barang'));
}
```

Fungsi store untuk menambahkan data stok barang dengan memilih kode barang, toko yang dituju, dan jumlah stok. Jumlah stok disini harus lebih dari 0 atau akan muncul error. Dan kalau kode barang dan toko yang dipilih sudah terdaftar dan mau membuat lagi akan muncul error. Fungsi menambahkan data stok barang dapat dilihat pada segmen program 4.32.

Segmen Program 4.32. Fungsi menambahkan data stok barang

```
public function create()
    {
        $toko = toko::all();
        $barang = barang::all();

        return view('stok.create', compact('toko', 'barang'));
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        // Validate input
        $validatedData = $request->validate([
            'kode_barang' => 'required',
            'id_toko' => 'required',
            'jumlah_stok' => 'required|integer|min:0',
            'nilai' => 'integer|nullable'
        ]);

        // Set default value for 'nilai' if not provided
        $validatedData['nilai'] = $validatedData['nilai'] ?? 0;

        // Check if stock already exists
        $existingStock = Stok::where('id_toko', $validatedData['id_toko'])
                        ->where('kode_barang', $validatedData['kode_barang'])
                        ->first();
```

```
        if ($existingStock) {
            return redirect()->back()->with('error', 'Data stok untuk toko dan barang tersebut sudah ada.');
        }

        // Create new stock
        Stok::create($validatedData);

        return redirect('/stok')->with('success', 'Data stok berhasil disimpan.');
    }
```

Fungsi detail untuk melihat data stok barang lebih spesifik. Atribut yang di tampilkan adalah kode barang, nama barang, Gambar barang, nama toko, harga barang, dan jumlah stok. Fungsi melihat detail data barang dapat dilihat pada segmen program 4.33.

Segmen Program 4.33. Fungsi melihat detail data stok barang

```
public function detail($id_stok)
    {
        try {
            // Ambil barang berdasarkan kode_barang dengan relasi kategori dan brand
            $stok = stok::with('barang', 'toko')->where('id_stok', $id_stok)->firstOrFail();

            // Kembalikan view dengan data barang
            return view('stok.show', compact('stok'));
        } catch (ModelNotFoundException $e) {
            // Tangani exception jika barang tidak ditemukan
            return redirect('/stok')->withErrors('Stok not found.');
        } catch (\Exception $e) {
            // Tangani exception lainnya jika terjadi kesalahan
            return redirect('/stok')->withErrors('Failed to fetch data: ' . $e->getMessage());
        }
    }
```

Fungsi update untuk mengedit data barang dengan memunculkan data stok barang yang sudah terdaftar. Dalam fungsi ini pengguna hanya dapat mengubah jumlah stok. Fungsi mengedit data barang dapat dilihat pada segmen program 4.34.

Segmen Program 4.34. Fungsi mengedit data stok barang

```
public function edit($id_stok)
    {
        $stok = Stok::with('toko', 'barang')->findOrFail($id_stok);
        return view('stok.edit', compact('stok'));
    }

    public function update(Request $request, $id_stok)
    {
```

```
        $stok = Stok::findOrFail($id_stok);

        $validatedData = $request->validate([
            'jumlah_stok' => 'required|integer|min:1',
            'nilai' => 'integer|nullable'
        ]);

        // Set default value for 'nilai' if not provided
        $validatedData['nilai'] = $validatedData['nilai'] ?? $stok->nilai;

        // Ambil data asli dari database
        $originalData = $stok->toArray();

        // Menggabungkan data asli dan data baru untuk perbandingan
        $newData = array_merge($originalData, $validatedData);

        // Periksa apakah ada perubahan antara data asli dan data baru
        $changes = array_diff_assoc($newData, $originalData);

        // Jika tidak ada perubahan, kembalikan dengan pesan tidak ada perubahan
        if (empty($changes)) {
            return redirect('/stok')->with('info', 'No changes were made to the stock.');
        }

        $stok->update($validatedData);

        return redirect('/stok')->with('success', 'Stock has been updated.');
    }
```

Fungsi destroy untuk menghapus data barang apabila ada kesalahan input. Menghapus data barang tidak bisa dilakukan apabila data barang sudah pernah digunakan untuk tabel stok. Fungsi menghapus data supplier dapat dilihat pada segmen program 4.35.

Segmen Program 4.35. Fungsi menghapus data stok barang

```
public function destroy($id_stok)
    {
        // Cari barang berdasarkan kode_barang
        $stok = stok::find($id_stok);

        // Debug untuk melihat apakah barang ditemukan
        // dd($barang);

        if ($stok) { // Jika barang ditemukan

            // Hapus barang
            $stok->delete();

            // Redirect dengan pesan sukses
            return redirect('/stok')->with('success', 'Stok has been deleted!');

        } else {
```

```
        // Barang tidak ditemukan
        return redirect('/stok')->withErrors('Stok not found.');
      }
  }
```

4.4.11. Manage Penjualan

Dalam fitur yang digunakan untuk penjualan. pengguna dapat melihat semua data penjualan terdaftar, melihat detail dari transaksi penjualan dan menambahkan penjualan. Atribut-atribut yang di tampilkan adalah nomor transaksi, nama customer, nama karyawan/owner yang membuat transaksi penjualan, tanggal pembuatan, dan total transaksi. Fungsi melihat seluruh daftar penjualan dapat dilihat dari segmen program 4.36.

Segmen Program 4.36. Fungsi melihat seluruh daftar penjualan di website

```
public function index()
  {
    $transaksi=DB::table("transaksi_jual")
            ->join("customer","customer.id_customer","transaksi_jual.id_customer")
            ->join("toko","toko.id_toko","transaksi_jual.id_toko")
            ->select("customer.name_customer","toko.name_toko","transaksi_jual.*")
            ->get();
    return view('transaksijual.index', compact('transaksi'));
  }
```

Fungsi detail untuk melihat data transaksi penjualan lebih spesifik. Atribut yang di tampilkan adalah tanggal pembuatan transaksi, nama customer, nama barang yang di beli, jumlah barang, subtotal dari harga barang dikalikan dengan jumlah, dan total harga dari semua subtotal. Fungsi melihat detail data penjualan dapat dilihat pada segmen program 4.37.

Segmen Program 4.37. Fungsi melihat detail data penjualan di website

```
public function detail(Request $request)
  {
    $id=$request->input("id");
    $barang=DB::table("barang")->get();
    $kategori=DB::table("kategori")->get();
    $customer=DB::table("customer")->get();
    $transaksijual=DB::table("transaksi_jual")->where("nomor_transaksi",$id)->first();
    $transaksidetailjual=DB::table("transaksi_detail_jual")
                ->join("barang","transaksi_detail_jual.kode_barang","barang.kode_barang")
                ->where("nomor_transaksi",$id)
                ->select("barang.name_barang","transaksi_detail_jual.*")
```

```
                    ->get();
        return view('transaksijual.detail',
compact('barang','kategori','customer','transaksijual','transaksidetailjual'));
    }
```

Fungsi store untuk menambahkan transaksi data penjualan barang. Atribut yang perlu diisikan adalah memilih nama customer yang membeli, memilih metode pembayaran, memilih toko, memilih nama barang yang dibeli, jumlah stok yang masih tersedia, jumlah barang yang dibeli, harga barang, subtotal dari harga barang dikalikan dengan jumlah barang yang dibeli, dan total keseluruhan dari subtotal. Untuk menambahkan barang yang akan dibeli bisa menggunakan tambah detail, apabila stok tidak tersedia berarti tidak bisa ditambahkan. Membeli barang tidak bisa melebihi dari jumlah stok yang ada. Apabila, ada kesalahan input masih bisa dihapus. Sebelum akhirnya menyimpan transaksi. Fungsi menambahkan transaksi data penjualan di website dapat dilihat pada segmen program 4.38.

Segmen Program 4.38. Fungsi menambahkan transaksi data penjualan di website

```
public function create()
    {
        $barang=DB::table("barang")->get();
        $kategori=DB::table("kategori")->get();
        $customer=DB::table("customer")->get();
        $toko=DB::table("toko")->get();
        $stok=DB::table("stok")->get();
        return view('transaksijual.create', compact('barang','kategori','customer','toko','stok'));
        //
    }
public function store(Request $request)
    {
        //echo "in here";
        $jumlah=$request->input("jumlah");
        $barang=$request->input("barang");
        $customer=$request->input("customer");

        print_r($jumlah);
        echo "<br/>";
        print_r($barang);
        $idpenjualan=Str::uuid()->toString();
        //
        echo "<br/>";

        $user=Auth::user();
        print_r($user->id);

        $toko=DB::table("toko")->where("id_toko",$request->input("toko"))->first();
        $dinsert=[];
```

```
    $dinsert["id_customer"]=$customer;
    $dinsert["nomor_transaksi"]=$idpenjualan;
    $dinsert["id_user"]=$user->id;
    $dinsert["id_toko"]=$toko->id_toko;
    $dinsert["harga_jual"]=0;
    $dinsert["status_pembayaran"]=$request->input("status_pembayaran");
    $dinsert["tanggal_jual"]=date("y-m-d H:i:s");
    $dinsert["created_at"]=date("y-m-d H:i:s");
    $dinsert["updated_at"]=date("y-m-d H:i:s");
    DB::table("transaksi_jual")->insert($dinsert);

    $total=0;
    for ($i=0;$i<count($barang);$i++)
    {
      $totalStok=0;
      $nilai=0;
      $hpp=0;
      $stokSebelum=DB::table("stok")->where("id_toko",$toko->id_toko)-
>where("kode_barang",$barang[$i])->first();
      if ($stokSebelum!=null)
      {
        $totalStok=$stokSebelum->jumlah_stok;
        $nilai=$stokSebelum->nilai;
        $hpp=$nilai/$totalStok;
      }


      $detail=DB::table("barang")->where("kode_barang",$barang[$i])->first();
      $jml=$jumlah[$i];
      $dinsertdetail=[];
      $dinsertdetail["harga"]=$detail->harga_jual;
      $dinsertdetail["kode_barang"]=$barang[$i];
      $dinsertdetail["quantity"]=$jml;
      $dinsertdetail["harga"]=$detail->harga_jual;
      $dinsertdetail["subtotal"]=$jml*$detail->harga_jual;
      $dinsertdetail["nomor_transaksi"]=$idpenjualan;
      $dinsertdetail["hpp"]=$hpp;
      $total+=$dinsertdetail["subtotal"];

      DB::table("transaksi_detail_jual")->insert($dinsertdetail);







      $stokBaru=$totalStok-$jml;
      $nilaiBaru=$nilai-($hpp*$jml);



      if ($stokSebelum==null)
      {
        $dtl=[];
```

```
            $dtl["kode_barang"]=$barang[$i];
            $dtl["id_toko"]=$toko->id_toko;
            $dtl["jumlah_stok"]=$stokBaru;
            $dtl["min_stok"]=0;
            $dtl["nilai"]=$nilaiBaru;
            DB::table("stok")->insert($dtl);
          }
          else {
            $dtl=[];
            $dtl["jumlah_stok"]=$stokBaru;
            $dtl["nilai"]=$nilaiBaru;
            DB::table("stok")->where("id_stok",$stokSebelum->id_stok)->update($dtl);
          }
        }

        DB::table("transaksi_jual")->where("nomor_transaksi",$idpenjualan)-
>update(["harga_jual"=>$total]);
        return redirect("penjualan");
    }
```

Dalam fitur yang digunakan untuk penjualan untuk aplikasi. Fitur yang tersedia di aplikasi adalah melihat seluruh data penjualan dan menambahkan data penjualan. Fungsi _DashboardState untuk melihat seluruh data penjualan yang sudah terdaftar. Fungsi penjualan untuk menampilkan seluruh data barang di aplikasi dapat dilihat pada segmen program 4.39.

Segmen Program 4.39. Fungsi penjualan seluruh data penjualan di aplikasi

```
class _PenjualanState extends State<PenjualanPage> {
  String active = "";
  List<dynamic> salesData = [];

  @override
  void initState() {
    super.initState();
    callloadpenjualan();
  }

  Future<void> callloadpenjualan() async {
    final response = await http.get(Uri.parse(baseURL + "getpenjualan"));
    if (response.statusCode == 200) {
      setState(() {
        salesData = json.decode(response.body)['result'];
      });
    } else {
      throw Exception('Failed to load sales data');
    }
  }
```

Fungsi _TambahPenjualanState untuk menambahkan data barang. Atribut yang bisa ditambahkan nama customer yang membeli, toko tempat menjual, metode pembayaran, nama barang dan jumlah barang yang dijualkan. Fungsi menambahkan data penjualan di aplikasi dapat dilihat pada segmen program 4.40.

Segmen Program 4.40. Fungsi menambahkan data penjualan di aplikasi

```
class _TambahPenjualanState extends State<TambahPenjualan> {
  final _formKey = GlobalKey<FormState>();
  late TextEditingController _jumlahController;
  dynamic? _selectedToko;
  dynamic? _selectedCustomer;
  dynamic? _selectedBarang;
  String _itemName = '';
  String _itemKode = "";
  String _selectedOption = "Cash";

  BuildContext? tcontext;

  List<dynamic> _toko = [];
  List<dynamic> _customer = [];
  List<dynamic> _barang = [];
  List<dynamic> _stok = [];

  var isLoading = false;
  @override
  void initState() {
    super.initState();
    _jumlahController = TextEditingController(text: "0");
    loadDataUser();
  }

  @override
  void dispose() {
    super.dispose();
  }

  loadDataUser() async {
    var data = await getTokoCustomer();
    _toko = data["toko"];
    _customer = data["customer"];
    _barang = data["barang"];
    _stok = data["stok"];

    if (_toko.length > 0) {
      _selectedToko = _toko[0];
    }

    if (_customer.length > 0) {
      _selectedCustomer = _customer[0];
```

```
  }

  if (_barang.length > 0) {
    _selectedBarang = _barang[0];
  }
  setState(() {});
  //print(user.toString());
}

Future<void> _uploadImage(File image) async {
  final url = Uri.parse(baseURL + "tambahbarang");
}

Widget radioMetode() {
  return Column(
    children: <Widget>[
      ListTile(
        title: const Text('Cash'),
        leading: Radio<String>(
          value: 'Cash',
          groupValue: _selectedOption,
          onChanged: (String? value) {
            setState(() {
              _selectedOption = value!;
            });
          },
        ),
      ),
      ListTile(
        title: const Text('Kredit'),
        leading: Radio<String>(
          value: 'Kredit',
          groupValue: _selectedOption,
          onChanged: (String? value) {
            setState(() {
              _selectedOption = value!;
            });
          },
        ),
      ),
    ],
  );
}

getSubtotal() {
  var jumlah = 0;
  var harga_jual = 0;
  try {
    harga_jual = int.parse(_selectedBarang["harga_jual"].toString());
    jumlah = int.parse(_jumlahController.text);
  } catch (ex) {}

  print("jumlah " + jumlah.toString());
  print("harga jual " + harga_jual.toString());
```

```
  print(_selectedBarang);
  if (jumlah == 0 || _selectedBarang == null) {
    return "";
  } else {
    return formatNumber(jumlah * harga_jual);
  }
  //if (_jumlahController.text=="")
 }

 getStok() {
  for (var i = 0; i < _stok.length; i++) {
    if (_stok[i]['id_toko'] == _selectedToko['id_toko'] &&
        _stok[i]['id_barang'] == _selectedBarang['id_barang']) {
      return _stok[i]['jumlah_stok'].toString();
    }
  }
  return "Tidak ada stok";
 }
```

4.4.12. Manage Pembelian

Dalam fitur yang digunakan untuk pembelian. pengguna dapat melihat semua data pembelian terdaftar, melihat detail dari transaksi pembelian dan menambahkan transaksi pembelian. Atribut-atribut yang di tampilkan adalah nomor transaksi, nama supplier, tanggal pembuatan, dan total transaksi. Fungsi melihat seluruh daftar pembelian dapat dilihat dari segmen program 4.41.

Segmen Program 4.41. Fungsi melihat seluruh daftar pembelian

```
public function index()
   {
      //
      $transaksi=DB::table("transaksi_supplier")
      ->join("supplier","supplier.id_supplier","transaksi_supplier.id_supplier")
      ->select("supplier.name_supplier","transaksi_supplier.*")
      ->get();
      return view('transaksibeli.index', compact('transaksi'));
   }
```

Fungsi detail untuk melihat data transaksi pembelian lebih spesifik. Atribut yang di tampilkan adalah tanggal pembuatan transaksi, nama supplier, nama barang yang di beli, jumlah barang, subtotal dari harga barang dikalikan dengan jumlah, dan total harga dari semua subtotal. Fungsi melihat detail data pembelian dapat dilihat pada segmen program 4.42.

Segmen Program 4.42. Fungsi melihat detail transaksi data pembelian

```
public function detail(Request $request)
  {
    $id=$request->input("id");
    $barang=DB::table("barang")->get();
    $kategori=DB::table("kategori")->get();
    $supplier=DB::table("supplier")->get();

    $transaksibeli=DB::table("transaksi_supplier")->where("nomor_nota",$id)->first();
    $transaksidetailbeli=DB::table("transaksi_detail_supplier")
                ->join("barang","transaksi_detail_supplier.kode_barang","barang.kode_barang")
                ->where("nomor_nota",$id)
                ->select("barang.name_barang","transaksi_detail_supplier.*")
                ->get();
    return view('transaksibeli.detail',
compact('barang','kategori','supplier','transaksibeli','transaksidetailbeli'));
  }
```

Fungsi store untuk menambahkan transaksi data pembelian barang. Atribut yang perlu diisikan adalah memilih nama supplier, memilih toko, memilih nama barang yang dibeli, jumlah stok, jumlah barang yang dibeli, harga barang, subtotal dari harga barang dikalikan dengan jumlah barang yang dibeli, dan total keseluruhan dari subtotal. Untuk menambahkan barang yang akan dibeli bisa menggunakan tambah detail, apabila stok tidak tersedia berarti tidak bisa ditambahkan. Membeli barang tidak bisa melebihi dari jumlah stok yang ada. Apabila, ada kesalahan input masih bisa dihapus. Sebelum akhirnya menyimpan transaksi. Fungsi menambahkan transaksi data pembelian dapat dilihat pada segmen program 4.43.

Segmen Program 4.43. Fungsi menambahkan transaksi data pembelian

```
public function create()
  {
    //
    $barang=DB::table("barang")->get();
    $kategori=DB::table("kategori")->get();
    $supplier=DB::table("supplier")->get();
    $toko=DB::table("toko")->get();
    return view('transaksibeli.create', compact('barang','kategori','supplier','toko'));
  }

  /**
   * Store a newly created resource in storage.
   */
  public function store(Request $request)
  {
    //
```

```
    $jumlah=$request->input("jumlah");
    $barang=$request->input("barang");
    $harga=$request->input("harga");

    //print_r($harga);
    $supplier=$request->input("supplier");
    $toko=$request->input("toko");

    //print_r($jumlah);
    //echo "<br/>";
    //print_r($barang);
    $idpenjualan=Str::uuid()->toString();
    //
    //echo "<br/>";

    $user=Auth::user();
    //print_r($user->id);

    //$toko=DB::table("toko")->first();
    $dinsert=[];
    $dinsert["id_supplier"]=$supplier;
    $dinsert["nomor_nota"]=$idpenjualan;
    $dinsert["id_user"]=$user->id;
    $dinsert["id_toko"]=$toko;
    //$dinsert["id_toko"]=$toko->id_toko;
    $dinsert["harga_supplier"]=0;
    //$dinsert["status_pembayaran"]="Cash";
    $dinsert["tanggal"]=date("y-m-d H:i:s");
    $dinsert["created_at"]=date("y-m-d H:i:s");
    $dinsert["updated_at"]=date("y-m-d H:i:s");
    DB::table("transaksi_supplier")->insert($dinsert);

    $total=0;
    for ($i=0;$i<count($barang);$i++)
    {
       $detail=DB::table("barang")->where("kode_barang",$barang[$i])->first();
       $jml=$jumlah[$i];
       $dinsertdetail=[];
       $dinsertdetail["harga"]=$harga[$i];
       $dinsertdetail["kode_barang"]=$barang[$i];
       $dinsertdetail["quantity"]=$jml;
       //$dinsertdetail["harga"]=$detail->harga_jual;
       $dinsertdetail["subtotal"]=$jml*$harga[$i];
       $dinsertdetail["nomor_nota"]=$idpenjualan;
       $total+=$dinsertdetail["subtotal"];

       DB::table("transaksi_detail_supplier")->insert($dinsertdetail);


       $totalStok=0;
       $nilai=0;

       $stokSebelum=DB::table("stok")->where("id_toko",$toko)-
>where("kode_barang",$barang[$i])->first();
```

```
        if ($stokSebelum!=null)
        {
           $totalStok=$stokSebelum->jumlah_stok;
           $nilai=$stokSebelum->nilai;
        }

        $nilaiBaru=$nilai+($jml*$harga[$i]);
        $stokBaru=$totalStok+$jml;

        if ($stokSebelum==null)
        {
           $dtl=[];
           $dtl["kode_barang"]=$barang[$i];
           $dtl["id_toko"]=$toko;
           $dtl["jumlah_stok"]=$stokBaru;
           $dtl["min_stok"]=0;
           $dtl["nilai"]=$nilaiBaru;
           DB::table("stok")->insert($dtl);
        }
        else {
           $dtl=[];
           $dtl["jumlah_stok"]=$stokBaru;
           $dtl["nilai"]=$nilaiBaru;
           DB::table("stok")->where("id_stok",$stokSebelum->id_stok)->update($dtl);
        }
     }

     DB::table("transaksi_supplier")->where("nomor_nota",$idpenjualan)-
>update(["harga_supplier"=>$total]);
     return redirect("pembelian");
  }
```

4.4.13. Manage Restock Barang dan Barang Tidak Laku

Dalam fitur yang digunakan untuk manage restock barang ini digabungkan dengan dashboard. Fitur restock ini mencari stok barang dari semua barang yang terdaftar yang memiliki nilai minimal stok lebih tinggi dari stok yang dimiliki sekarang. Akan muncul barang yang akan di restock di bawah tampilan dashboard. Tampilan dashboard disini menampilkan total penjualan, total pembelian, total customer, dan beberapa chart. Untuk Barang tidak laku ini didapatkan dari barang yang tidak terjual lebih dari 5 barang selama 1 bulan. Fungsi dashboard dan restock barang dapat dilihat pada segmen program 4.44.

Segmen Program 4.44. Fungsi dashboard dan restock barang

```
public function dashboard()
  {
```

```
    $data=[];

    $data["penjualan"]=DB::select("SELECT *
                        FROM
                        (
                        SELECT c.name_customer,SUM(tj.harga_jual) as
total_penjualan,COUNT(nomor_transaksi) as total_transaksi
                        FROM transaksi_jual tj
                        INNER JOIN customer c ON (tj.id_customer=c.id_customer)
                        GROUP BY tj.id_customer,c.name_customer
                        ) x
                        ORDER BY total_penjualan DESC
                        ");
    $data["penjualan2"]=DB::select("SELECT b.name_barang,SUM(tdj.quantity) as quantity
                          FROM transaksi_detail_jual tdj
                          INNER JOIN barang b ON (tdj.kode_barang=b.kode_barang)
                          GROUP BY tdj.kode_barang,b.name_barang
                          ORDER BY quantity DESC");

    $data["penjualan3"]=DB::select("SELECT DATE_FORMAT(tj.tanggal_jual,'%M-%Y') as periode,
                        SUM(tj.harga_jual) as total_penjualan
                        FROM transaksi_jual tj
                        GROUP BY DATE_FORMAT(tj.tanggal_jual,'%M-%Y')
                        ORDER BY tanggal_jual ASC");
    $pj=DB::select("SELECT SUM(harga_jual) as total FROM transaksi_jual");
    $pb=DB::select("SELECT SUM(harga_supplier) as total FROM transaksi_supplier");
    $cust=DB::select("SELECT COUNT(id_customer) as total FROM customer");


    $data["stok"]=DB::select("SELECT t.name_toko,b.name_barang,s.jumlah_stok,s.min_stok
    FROM stok s
    INNER JOIN barang b ON (s.kode_barang=b.kode_barang)
    INNER JOIN toko t ON (s.id_toko=t.id_toko)
    WHERE s.min_stok!=0 AND s.jumlah_stok<s.min_stok");
    $data["total_penjualan"]=$pj[0]->total;

    $data["total_pembelian"]=$pb[0]->total;
    $data["total_customer"]=$cust[0]->total;

    $t2=date("Y-m-d");
    $t1=date("Y-m-d",strtotime($t2." -30 days"));

    $arrT=[];
    $itlaku=[];
    $qstok="SELECT SUM(dtj.quantity) as ttl,dtj.kode_barang,b.name_barang
        FROM transaksi_detail_jual dtj
        INNER JOIN transaksi_jual tj ON (tj.nomor_transaksi=dtj.nomor_transaksi)
        INNER JOIN barang b ON (dtj.kode_barang=b.kode_barang)
        WHERE tj.tanggal_jual>='".$t1."'
        GROUP BY dtj.kode_barang,b.name_barang";
    $tlaku=DB::select($qstok);
    for ($i=0;$i<count($tlaku);$i++)
    {
      $itlaku[]=$tlaku[$i]->kode_barang;
```

```
        if ($tlaku[$i]->ttl<5)
        {
            $arrT[]=(array)$tlaku[$i];
        }

    }

    $tbl=DB::table("barang")->get();
    for ($i=0;$i<count($tbl);$i++)
    {
        if (!in_array($tbl[$i]->kode_barang,$itlaku))
        {
            $t=[];
            $t["ttl"]=0;
            $t["kode_barang"]=$tbl[$i]->kode_barang;
            $t["name_barang"]=$tbl[$i]->name_barang;
            $arrT[]=$t;
        }
    }
    $data["arrT"]=$arrT;

    return view('dashboard',$data);
}
```

4.4.14. Manage Laporan

Dalam fitur yang digunakan untuk manage laporan penjualan dan laporan analitik. Fungsi laporan penjualan akan menampilkan beberapa filter untuk laporan harian, bulanan, dan tahunan, untuk mencetak laporan penjualan perlu memilih waktu dari kapan laporan itu mau dibuat hingga waktu yang di tentukan oleh owner/admin. Harga barang per item dihitung menggunakan hpp menghitung HPP dengan mengambil rata-rata biaya barang yang tersedia untuk dijual selama periode tersebut. Dalam metode ini, setiap unit barang yang dijual dianggap memiliki biaya yang sama, yaitu rata-rata biaya barang yang tersedia. Sebagai contoh persedian awal suatu barang adalah 20 dengan harga 20.000. lalu membeli lagi barang tersebut 20 tetapi mendapatkan kenaikan barang menjadi 25.000. Jadi harga rata-rata nya menjadi 22.500 per barang dengan stok sekarang 40. Pada saat melakukan penjualan 10 barang dengan harga 40.000. Jadi pemilik memiliki keuntungan sebesar 175.000 dengan penjualan 10 barang. Fungsi melihat dan membuat laporan penjualan di website dapat dilihat pada segmen program 4.45.

Segmen Program 4.45. Fungsi melihat dan membuat laporan penjualan di website

```
public function lappenjualan(Request $request)
  {
```

```
    $data=[];
    return view('transaksijual.laporan',$data);
  }
public function cetaklaporanpenjualan(Request $request)
  {
    $data["tanggal_mulai"]=$request->input("tanggal_mulai");
    $data["tanggal_selesai"]=$request->input("tanggal_selesai");
    $data["periode"]=$request->input("periode");


    $t1=$data["tanggal_mulai"]." 00:00:00";
    $t2=$data["tanggal_selesai"]." 23:59:59";

    $format="DATE_FORMAT(tanggal_jual,'%d-%M-%Y')";
    if ($data["periode"]=="Bulanan")
    {
      $format="DATE_FORMAT(tanggal_jual,'%M-%Y')";
    }
    else if ($data["periode"]=="Tahunan")
    {
      $format="DATE_FORMAT(tanggal_jual,'%Y')";
    }

    $transaksi=DB::select("SELECT ".$format." as Hari,SUM(subtotal) as Total,SUM(quantity*(harga-
hpp)) as Keuntungan
    FROM transaksi_detail_jual dj
    INNER JOIN transaksi_jual tj ON (tj.nomor_transaksi=dj.nomor_transaksi)
    WHERE tj.tanggal_jual>=? AND tj.tanggal_jual<=?
    GROUP BY ".$format,[$t1,$t2]);
    $data["transaksi"]=$transaksi;
    return view('transaksijual.cetaklaporan',$data);
  }
```