

BAB IV PEMBAHASAN

4.1 Gambaran Umum Obyek Penelitian

Objek penelitian yang digunakan oleh penulis adalah gudang logistik kantor BPBD Kabupaten Malang, yang berlokasi di Jalan Trunojoyo, Kedungpedaringan, Kecamatan Kepanjen, Kabupaten Malang, Jawa Timur. Kondisi peletakan rak dan seluruh item logistik di dalam gudang saat penelitian berlangsung tampak cukup rapi, meskipun banyak rak terlihat kosong karena persediaan logistik yang telah berkurang untuk didistribusikan sebagai bantuan bencana, serta beberapa item logistik yang sudah tidak layak atau kadaluarsa. Berikut adalah denah tata letak rak di gudang logistik, sekaligus gambar kondisi gudang disaat penelitian ini berlangsung.



Gambar 4.1 Denah Tata Letak Rak Gudang Logistik



Gambar 4.2 Kondisi Gudang Logistik

4.2 Implementasi

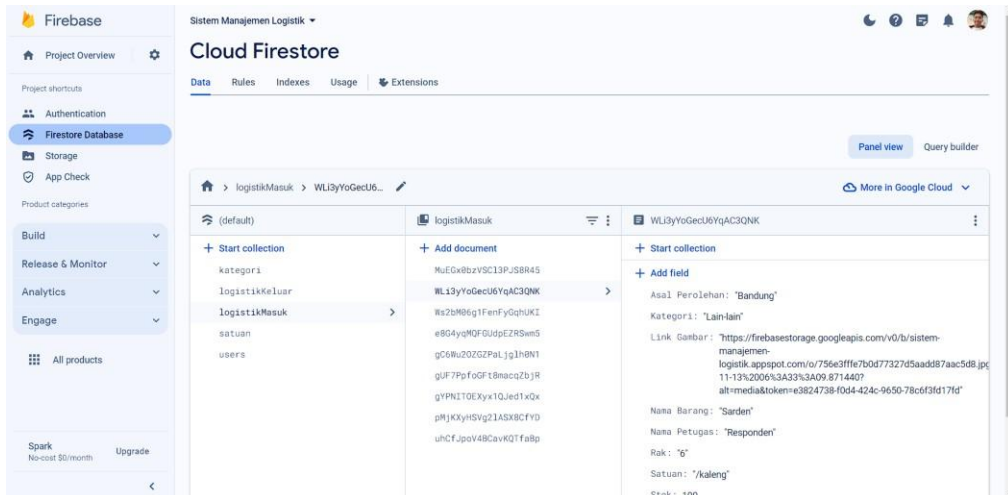
4.2.1 Spesifikasi Produk

Aplikasi LogPal BPBD Kab.Malang memiliki spesifikasi sebagai berikut:

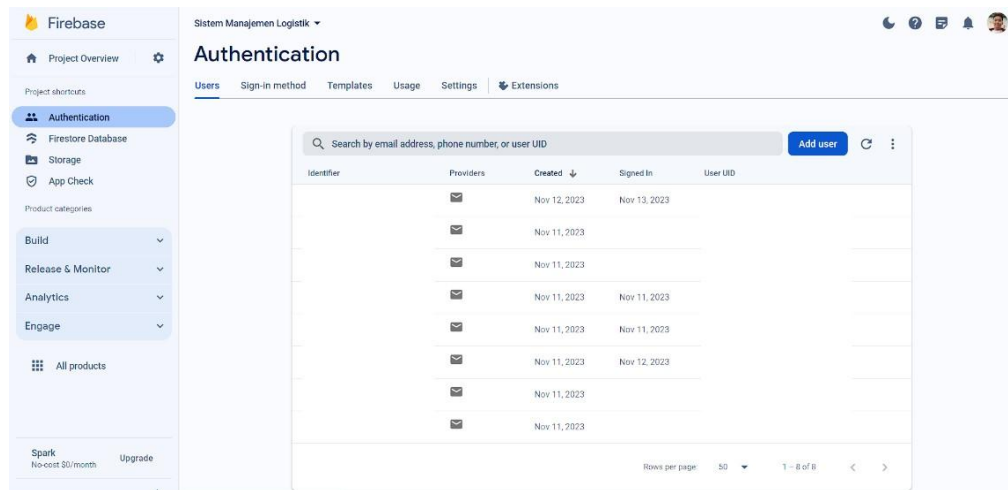
- a. Bahasa Pemrograman : Dart
- b. *Framework* : Flutter
- c. Versi Android : Android 12

4.2.2 Implementasi *Database*

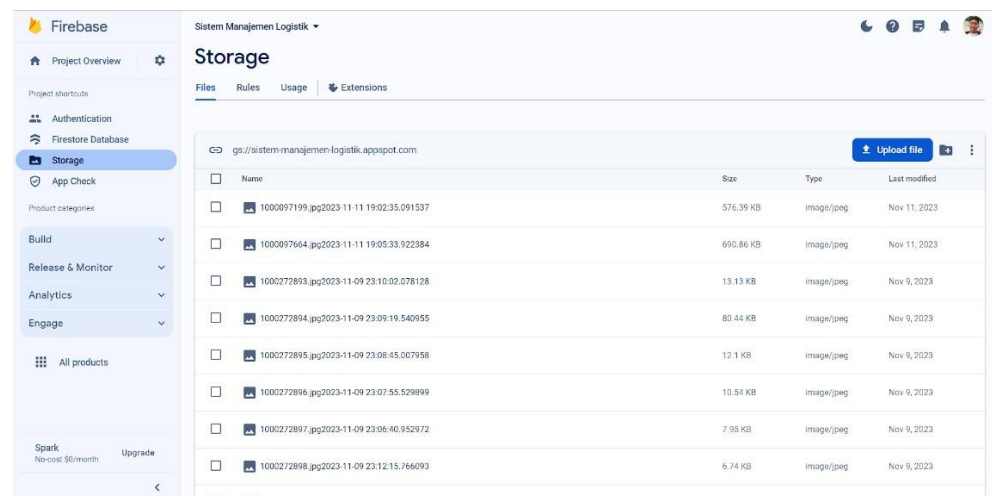
Implementasi database yang digunakan untuk perancangan aplikasi sistem manajemen logistik berbasis Android adalah Firebase. Penggunaan Firebase dalam aplikasi ini memudahkan petugas dalam mengelola data logistik, karena data yang tersimpan bersifat real-time, yang berarti dapat diakses di perangkat mana pun dan kapan pun, asalkan perangkat tersebut telah menginstal aplikasi sistem manajemen logistik. Fitur Firebase yang digunakan untuk menyimpan data logistik selama proses perancangan terdiri dari Authentication, Firestore Database, dan Storage. Penggunaan Firebase beserta fitur-fiturnya dapat dilihat pada Gambar 4.3 hingga 4.5 berikut ini.



Gambar 4.3 Implementasi Firestore Database



Gambar 4.4 Implementasi Authentication



Gambar 4.5 Implementasi Storage

4.2.3 Implementasi Program

Source code lengkap dari implementasi program LogPal BPBD Kab.Malang dapat diakses melalui tautan: <http://bit.ly/sistem-manajemen-logistik>.

4.2.3.1 Main Function

Segmen Program 4.1 dibawah merupakan fungsi utama dari aplikasi LogPal Kab.Malang.

```
8 >> void main() async {
9     WidgetsFlutterBinding.ensureInitialized();
10
11     await Firebase.initializeApp().then((value) {
12         Get.put(AuthenticationRepository());
13     });
14
15     await initializeDateFormatting('id_ID', null);
16
17     runApp(const MyApp());
18 }
```

Segmen Program 4.1 *Main Function*

Baris kode apapun yang terdapa didalam *function* main() akan dijalankan pertama kali ketika aplikasi dijalankan. Pada segmen ini, *method* initializeApp() akan menjalankan baris yang ada didalamnya, dimana baris tersebut merupakan *method call* yang memanggil *method* lain, dimana *method* yang dipanggil merupakan *method* yang memiliki fungsi mengecek kondisi autentikasi pengguna yang terdaftar didalam Authentication pada Firebase.

4.2.3.2 Halaman Login & Authentication Repository

Berikut merupakan halaman *login* dari aplikasi LogPal Kab.Malang. Pada bagian bawah, terdapat *form* yang digunakan untuk masuk kedalam aplikasi. Pada

bagian kanan atas *form*, terdapat juga ikon bantuan yang dapat menunjukkan teks singkat terkait bantuan ketika terjadi kesalahan pada proses masuk kedalam aplikasi.



The image shows a mobile application login screen. At the top, there is an orange header containing the logo of BPBD (Badan Penganggulangan Bencana Daerah) Kabupaten Malang and the text 'LOGPAL (LOGISTIK & PERALATAN)'. Below the header is a white login form with the title 'Login' and a help icon. The form contains two input fields: 'Username' and 'Password', followed by a blue 'Masuk' button.

Versi 1.8.10

Gambar 4.6 Halaman *Login*

```

class AuthenticationRepository extends GetxController{
    static AuthenticationRepository get instance => Get.find();

    final auth = FirebaseAuth.instance;
    late Rx<User?> firebaseUser;

    @override
    void onReady(){
        super.onReady();

        firebaseUser = Rx<User?>(auth.currentUser);

        firebaseUser.bindStream(auth.userChanges());
        ever(firebaseUser, _setInitialScreen);
    }

    _setInitialScreen(User? user){
        user == null
            ? Get.offAll(() => const LoginPage())
            : Get.offAll(() => const LogisticMain());
    }

    Future<String?> login(String email, String password) async{
        try{
            await auth.signInWithEmailAndPassword(email: email, password: password);
            return null;
        } on FirebaseAuthException catch (e){
            String errorMessage = "Terjadi kesalahan. Silahkan coba lagi!.";

            if (e.code == 'invalid-email') {
                errorMessage = 'Email salah, mohon cek kembali.';
            } else if (e.code == 'wrong-password') {
                errorMessage = 'Password salah, mohon cek kembali.';
            } else if (e.code == 'user-not-found') {
                errorMessage = 'Akun tidak ditemukan, mohon periksa kembali.';
            }

            return errorMessage; // Return an error message on login failure
        }
    }

    void logout() async => await auth.signOut();
}

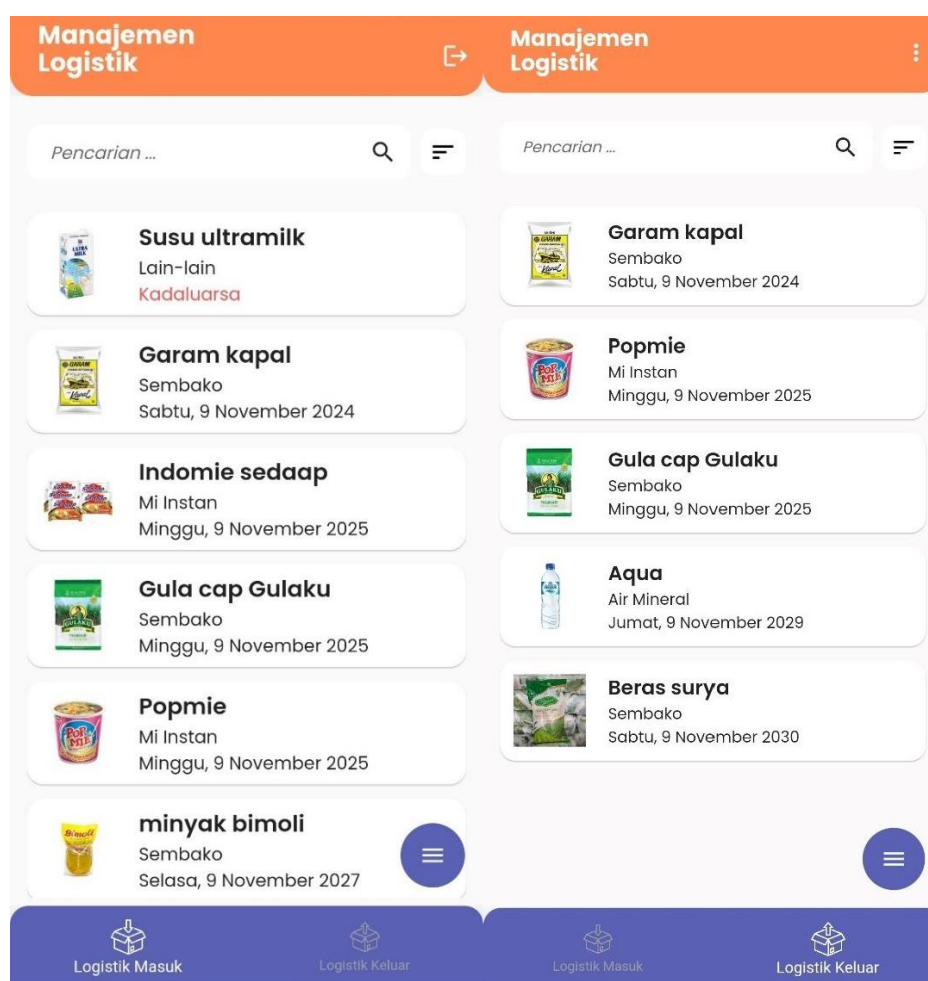
```

Segmen Program 4.2 Kelas AuthenticationRepository

Pada Segmen Program 4.2 diatas, baris kode yang terdapat pada kelas AuthenticationRepository berfungsi dalam mengatur status autentikasi menggunakan Authentication Firebase dan menavigasi pengguna ke layar yang berbeda berdasarkan apakah mereka telah diautentikasi atau tidak. Kelas ini juga menyediakan metode untuk *login* dan *logout*. Sifat reaktif GetX memastikan bahwa UI diperbarui secara otomatis berdasarkan perubahan status autentikasi.

4.2.3.3 Navigasi Halaman Utama

Berikut merupakan halaman utama dari aplikasi LogPal Kab.Malang. Ketika masuk kedalam aplikasi, halaman utama akan secara otomatis memilih menu Logistik Masuk, dan menampilkan daftar item logistik masuk yang telah tersedia didalam sistem. Dalam daftar logistik masuk maupun keluar, setiap item logistik ditampilkan dalam sebuah *widget* bernama *card*, dimana terdapat beberapa detail singkat dari item tersebut, yaitu nama barang, kategori barang, dan tanggal kadaluarsa barang. Kedua daftar ini juga telah terurut berdasarkan tanggal kadaluarsa paling dekat dengan tanggal saat ini.



Gambar 4.7 Halaman Utama


```

int _selectedIndex = 0;

class LogisticMain extends StatefulWidget {
  const LogisticMain({super.key});

  @override
  State<LogisticMain> createState() => _LogisticMainState();
}

class _LogisticMainState extends State<LogisticMain> {

  final List<Widget> _pages = [
    const LogisticIn(),
    const LogisticOut(),
  ];
}

class CustomBottomNavigationBar extends StatelessWidget {
  final Function(int) onTabSelected;

  const CustomBottomNavigationBar({Key? key, required this.onTabSelected})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
    var screenHeight = MediaQuery.of(context).size.height;
    return Container(
      height: screenHeight * 0.08,
      decoration: const BoxDecoration(...), // BoxDecoration
      child: BottomNavigationBar(
        items: [
          BottomNavigationBarItem(...), // BottomNavigationBarItem
          BottomNavigationBarItem(...), // BottomNavigationBarItem
        ],
        selectedItemColor: Colors.white, // Customize selected item color
        unselectedItemColor: Colors.grey, // Customize unselected item color
        backgroundColor: Colors.transparent, // Make background transparent
        elevation: 0,
        currentIndex: _selectedIndex,
        onTap: onTabSelected,
      ), // BottomNavigationBar
    ); // Container
  }
}

```

Segmen Program 4.3 Proses Navigasi Halaman pada *Class* Logistic Main

Pada Segmen Program 4.3 kode diatas merupakan kelas utama yang mengelola halaman yang berisi konten item logistik dengan bilah navigasi bawah yang dibuat khusus untuk menavigasi antara 2 *method*, yaitu LogisticIn() sebagai halaman logistik masuk dan LogistikOut() sebagai halaman logistik keluar. Pengguna bisa beralih antara halaman logistik masuk dan keluar menggunakan

bilah navigasi bawah tersebut, dan ada fungsionalitas *logout* yang akan memicu *dialog* konfirmasi *logout*. Setiap halaman *method* berisi konten spesifik untuk manajemen logistik yang berkaitan.

4.2.3.4 Fungsi daftar item logistik masuk dan logistik keluar.

Pada Segmen Program 4.4 dibawah, fungsi `getRecords()` berfungsi untuk menangani pengambilan dan penyaringan data logistik masuk yang didapat dari Firestore Database, yang hasilnya akan ditampilkan pada menu daftar item Logistik Masuk. Fungsi yang berjalan pada segmen program ini, dikhususkan untuk menangani data yang diambil dari sebuah tabel, atau umumnya dalam Firestore dikenal dengan *collection*, dengan nama `logistikMasuk`, yaitu kumpulan *records* yang berisi data data item logistik masuk.

```

getRecords()async{
  setState() {
    _isLoading = true;
  });

  Query<Map<String, dynamic>> query = FirebaseFirestore.instance.collection('logistikMasuk');
  DateTime defaultEndDate = DateTime(...);

  // Check if there are selected filter options
  if (_selectedFilterOption.isNotEmpty) {...}

  if (_selectedFilterOption2.isNotEmpty) {
    // If only one filter option is selected, use normal where
    if(_selectedFilterOption2.length == 1){...}
    if(_selectedFilterOption2.length == 2){...}
  }

  var logisticData = await query.get();

  List<Map<String, dynamic>> logisticResults = [];
  for (var doc in logisticData.docs) {
    Map<String, dynamic> resultWithId = doc.data();
    resultWithId['id'] = doc.id; // Adding the document ID to the result map
    resultWithId['Tanggal Masuk'] = Timestamp(resultWithId['Tanggal Masuk'].seconds, 0,);
    logisticResults.add(resultWithId);
  }
}

```

```

var categoryData = await FirebaseFirestore
    .instance
    .collection('kategori')
    .get();

List<Map<String, dynamic>> categoryResults = [];
for (var doc in categoryData.docs) {
    categoryResults.add(doc.data());
}

// Populate filterOptions with unique values from 'Kategori' field
Set<String> uniqueCategories = categoryResults.map((result) => result['nama'] as String).toSet();
_filterOptions = uniqueCategories.toList();

setState(() {
    _allResults = logisticResults;
    _isLoading = false;
});
_allResults.sort((a, b) {
    // Extract and parse 'Tanggal Kadaluarsa' as DateTime
    DateTime tanggalKadaluarsaA = a['Tanggal Kadaluarsa'].toDate();
    DateTime tanggalKadaluarsaB = b['Tanggal Kadaluarsa'].toDate();

    // Compare 'Tanggal Kadaluarsa'
    return tanggalKadaluarsaA.compareTo(tanggalKadaluarsaB);
});
_searchResultList();
}

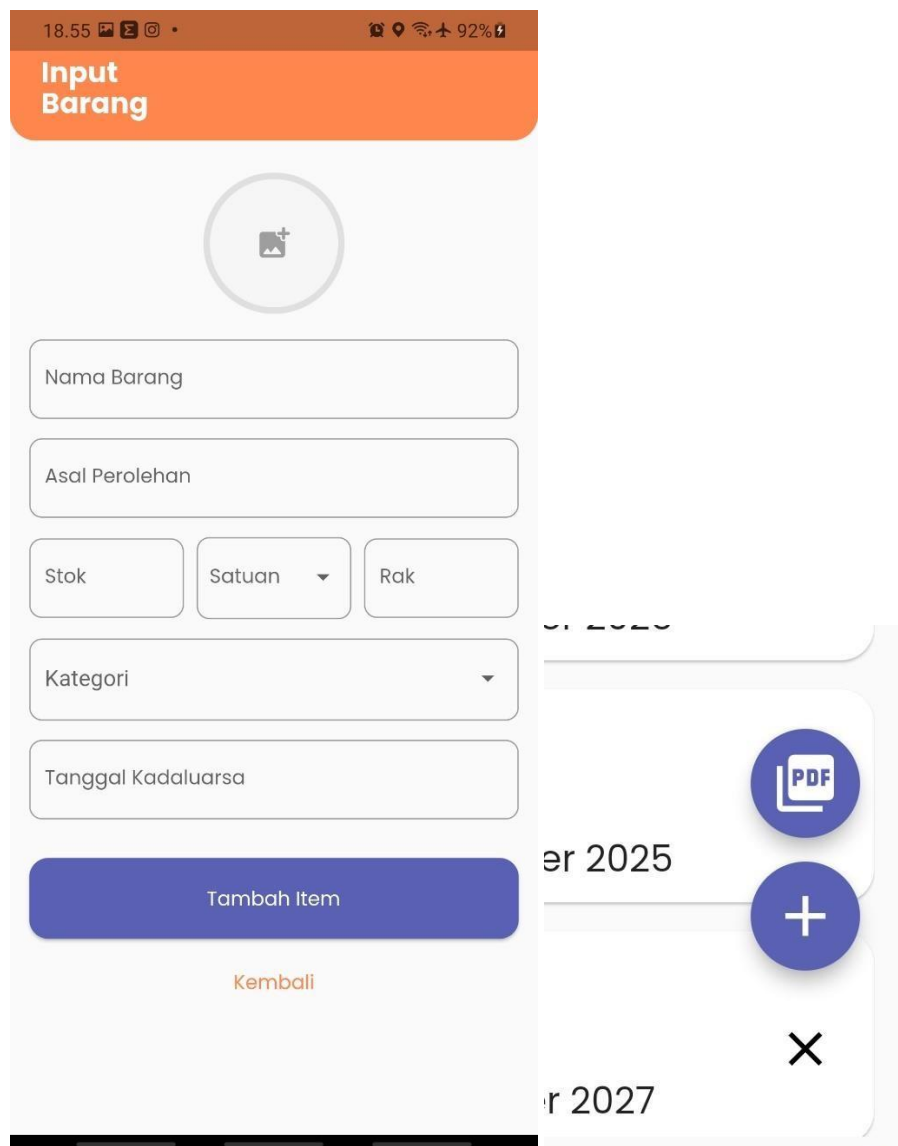
```

Segmen Program 4.4 *Method* getRecords()

Menu logistik masuk dan logistik keluar memiliki kode *method* getRecords() yang identik, namun tidak sama. Salah satu faktor pembeda yang dapat diketahui terdapat pada variable *query*, dimana kode yang ada didalam variable tersebut berfungsi untuk memilih *collection* mana yang akan dipilih untuk diambil datanya dari dalam Firestore yang berisi data-data dari item logistik masuk atau logistik keluar, dimana field-field yang terdapat pada tiap *collection* dapat memiliki nama *field* yang berbeda-bed. *Method* ini juga memiliki fungsi untuk menangani *filter* data item logistik dengan beberapa pilihan seperti daftar kategori baran kategori dan rentang tanggal masuk atau keluar sebuah item logistik.

4.2.3.5 *Input* Barang Masuk

Berikut merupakan halaman *input* barang dari aplikasi LogPal Kab.Malang. Untuk dapat masuk ke halaman berikut, petugas harus mengklik tombol '+' yang ada pada tombol mengambang, atau lebih sering dikenal sebagai *floating action button* yang ada pada halaman menu logistik masuk. Terdapat sebuah *form* dengan kolom-kolom *input*-an untuk memasukkan informasi terkait item logistik yang akan dimasukkan ke dalam sistem.



The screenshot shows a mobile application interface for 'Input Barang'. The top bar is orange with the title 'Input Barang'. Below the title is a circular icon with a plus sign and a document symbol. The form consists of several input fields: 'Nama Barang', 'Asal Perolehan', 'Stok', 'Satuan' (with a dropdown arrow), 'Rak', 'Kategori' (with a dropdown arrow), and 'Tanggal Kadaluarsa'. At the bottom of the form is a blue button labeled 'Tambah Item' and a link labeled 'Kembali'. On the right side of the screen, there is a floating action button (FAB) with a plus sign, and a close button (X) is visible below it. The background shows a list of items with dates like 'er 2025' and 'r 2027'.

Gambar 4.8 Tombol '+' yang mengarah ke halaman *input* barang

```

class LogisticsInModel{
    final String? id;
    final String name;
    final String source;
    final String storageId;
    final String units;
    final double stock;
    final String category;
    Timestamp dateEnd;
    Timestamp insertDate;
    final String imgPath;
    final String officer;

    LogisticsInModel({
        this.id,
        required this.name,
        required this.source,
        required this.storageId,
        required this.units,
        required this.stock,
        required this.category,
        required this.dateEnd,
        required this.insertDate,
        required this.imgPath,
        required this.officer,
    });

    toJson(){
        return{
            "Nama Barang": name,
            "Asal Perolehan": source,
            "Rak": storageId,
            "Satuan": units,
            "Stok": stock,
            "Kategori": category,
            "Tanggal Kadaluarasa":dateEnd,
            "Tanggal Masuk":insertDate,
            "Link Gambar": imgPath,
            "Nama Petugas": officer,
        };
    }

    factory LogisticsInModel.fromSnapshot(DocumentSnapsh
        final data = document.data()!;
        return LogisticsInModel(
            id: document.id,
            name: data['Nama Barang'] ?? '',
            source: data['Asal Perolehan'] ?? '',
            storageId: data['Rak'] ?? '',
            units: data['Satuan'] ?? '',
            stock: double.parse(data['Stok'].toString()),
            category: data['Kategori'] ?? '',
            dateEnd: data['Tanggal Kadaluarasa'],
            insertDate: data['Tanggal Masuk'],
            imgPath: data['Link Gambar'] ?? '',
            officer: data['Nama Petugas'] ?? '',
        );
    }
}

```

Segmen Program 4.5 Logistik Model

Segmen Program 4.5 di atas berisi kode yang berfungsi untuk digunakan untuk merepresentasikan dan mengelola data item logistik dalam aplikasi. *Instance* dari kelas ini dapat dibuat, dikonversi ke JSON untuk penyimpanan di Firestore, dan direkonstruksi dari snapshot Firestore.

```

class LogisticDb extends GetxController{
  static LogisticDb get instance => Get.find();
  final _dbLogistikMasuk = FirebaseFirestore.instance.collection('logistikMasuk');
  final _dbLogistikKeluar = FirebaseFirestore.instance.collection('logistikKeluar');

  insertLogisticAlert(LogisticsInModel logistics) async {
    try {
      await _dbLogistikMasuk.add(logistics.toJson());
      Get.snackbar(...);

      // Return a placeholder DocumentReference to match the expected type
      return Future.value(_dbLogistikMasuk.doc()); // Replace with the correct metho

    } catch (error) {
      Get.snackbar(...);

      // Return a FutureOr<DocumentReference<Map<String, dynamic>>>
      return Future.value(null);
    }
  }
}

```

Segmen Program 4.6 Class LogisticDb

Segmen Program 4.6 berisi kode yang berfungsi untuk menangani *insert* kedalam *collection* *logistikMasuk* di Firestore. Terdapat 2 variabel *final* yang digunakan untuk mendeklarasikan *collection* mana yang akan digunakan, lalu *method* *insertLogisticAlert()*, akan melakukan proses *insert* kedalam Firestore dengan *collection* yang telah dipilih dengan menggunakan variable yang telah di inialisasi sebelumnya, lalu menampilkan peringatan bahwa item telah berhasil di *input* atau gagal.

```

class LogisticInputController extends GetxController{
  static LogisticInputController get instance => Get.find();

  //Textfield Controllers to get data from Textfields
  final name = TextEditingController();
  final source = TextEditingController();
  final storageID = TextEditingController();
  final units = TextEditingController();
  final stock = TextEditingController();
  final category = TextEditingController();
  final dateEnd = TextEditingController();

  final logisticDb = Get.put(LogisticDb());

  Future<void> insertItem(LogisticsInModel logistics) async {
    await logisticDb.insertLogisticAlert(logistics);
  }
}

```

Segmen Program 4.7 *Class Controller* untuk *input*

Segmen Program 4.7 berisi kode yang berfungsi untuk menyimpan variabel untuk setiap *controller* dari text field pada halaman *input*. Terdapat juga *method* `insertItem()` yang digunakan sebagai fungsi perantara antara *class view input* barang dengan *method* `insetLogisticAlert` pada *class* `logisticDb()`

```

setState() {
  final logistics = LogisticsInModel(
    name: logisticController.name.text.trim(),
    source: logisticController.source.text.trim(),
    storageId: logisticController.storageID.text.trim(),
    units: units,
    stock: double.parse(logisticController.stock.text.trim()),
    category: category,
    dateEnd: Timestamp.fromDate(selectedDate),
    insertDate: Timestamp.fromDate(dateNow),
    imagePath: _urlItemImage?? 'Tidak ada gambar',
    officer: userData.name,
  ); // LogisticsInModel
  LogisticInputController.instance.insertItem(logistics);
  Get.offAll(() => const LogisticMain());
});

```

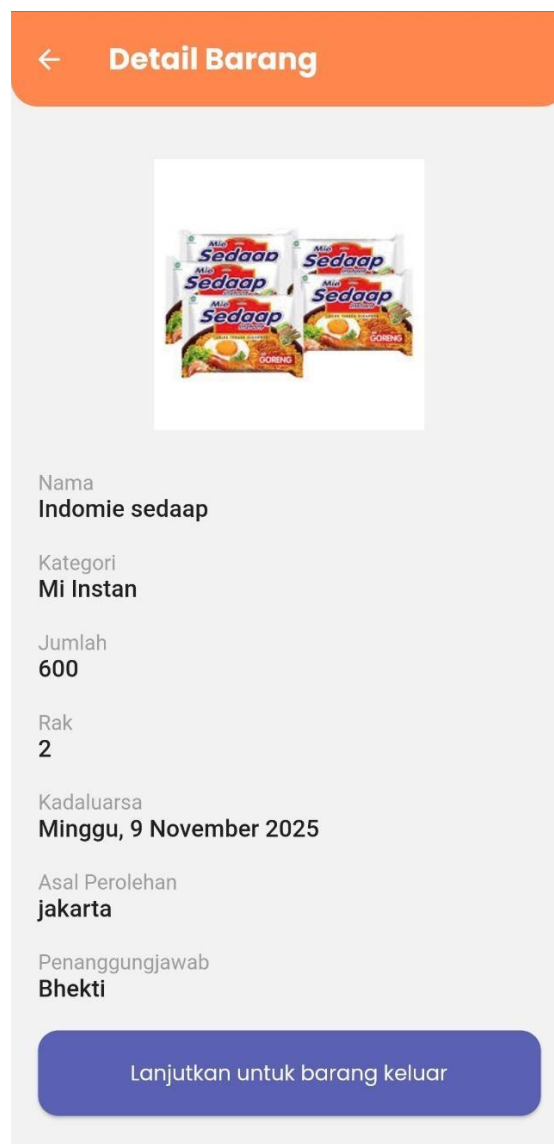
Segmen Program 4.8 Fungsi *OnClick* tombol *input* pada halaman *input*

Segmen Program 4.8 berisi kode yang berfungsi untuk mengirimkan nilai yang ada pada *text field* kedalam *model*, lalu akan di-pass melalui *class controller*

pada segmen sebelumnya kepada *class* `logisticDb()` untuk proses *input* kedalam Firestore.

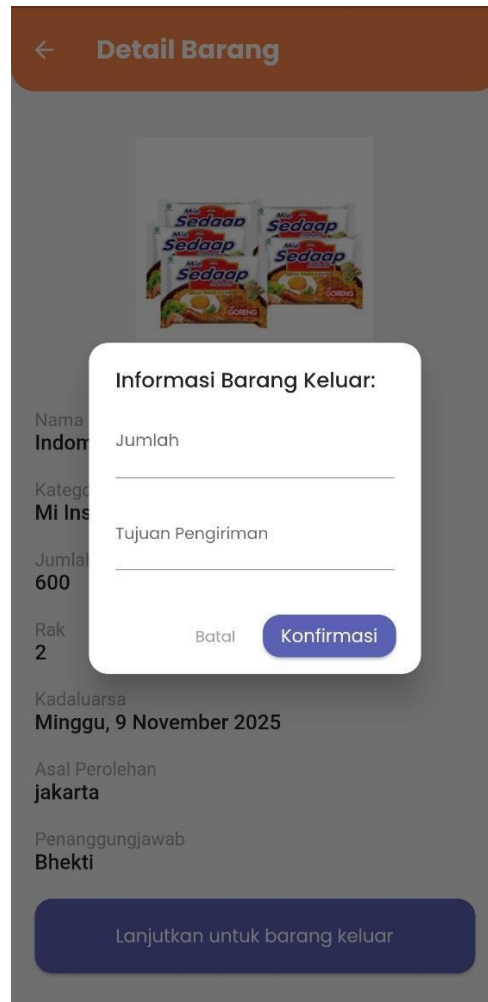
4.2.3.6 *Input* Barang Keluar

Berikut merupakan halaman detail barang item logistik masuk dari aplikasi LogPal Kab.Malang. Agar dapat mengeluarkan barang, petugas harus mengklik tombol ‘Lanjutkan untuk barang keluar’ yang terdapat pada halaman ini.



Gambar 4.9 Halaman detail barang item logistik masuk

Setelah petugas mengklik tombol tersebut, aplikasi akan menampilkan *popup* yang memiliki tampilan sebagai berikut.



The image shows a mobile application interface. At the top, there is a brown header with a back arrow and the text "Detail Barang". Below the header, there is a product image of "Mie Sedaap Goreng". A white popup form is overlaid on the screen, titled "Informasi Barang Keluar:". The form contains two input fields: "Jumlah" and "Tujuan Pengiriman". Below the input fields are two buttons: "Batal" and "Konfirmasi". At the bottom of the screen, there is a dark blue button with the text "Lanjutkan untuk barang keluar".

Gambar 4.10 *Popup form input* barang keluar

Petugas harus melengkapi kedua kolom yang tersedia pada *form* tersebut, dimana nilai yang akan berada pada kedua kolom ini akan dimasukkan bersamaan dengan data item logistik keluar lain yang akan di-*input* ke dalam *collection* logistikKeluar, seperti Nama Barang, Kategori, dan lain-lain, yang didapat dari data item logistik masuk. Setelah petugas selesai mengisi *form*, dan mengklik tombol konfirmasi, selanjutnya akan diarahkan kembali ke halaman utama.

```

class LogisticDetailController extends GetxController{

    final quantity = TextEditingController();
    final destination = TextEditingController();

    static LogisticDetailController get instance => Get.find();
    final logisticDb = Get.put(LogisticDb());

    Future<void> distributeItemController(
        LogisticsInModel logistics,
        String id,
        double quantity,
        String destination
    ) async {
        await logisticDb.distributeItem(logistics, id, quantity, destination);
    }
}

```

Segmen Program 4.9 *Class controller* LogisticDetailController()

Segmen Program 4.9 berisi kode yang berfungsi untuk menyimpan variabel untuk *controller* dari *text field* pada *form input* barang keluar yang ada pada halaman detail barang masuk. Terdapat juga *method* `distributeItemController()` yang digunakan sebagai fungsi perantara antara *form input* barang keluar pada halaman detail barang masuk dengan *method* `distributeItem()` yang terletak pada *class* `logisticDb()`.

```

Future<void> distributeItem(LogisticsInModel logistics, String id, double quantity, String destination) async {
  try {
    if (logistics.stock >= quantity) {

      // Update stock in logistikMasuk
      await _dbLogistikMasuk.doc(id).update({"Stok": logistics.stock - quantity});

      // Create an instance of LogisticsOutModel for the record in logistikKeluar
      LogisticsOutModel distributedItem = LogisticsOutModel(
        name: logistics.name,
        destination: destination,
        storageId: logistics.storageId,
        units: logistics.units,
        stock: quantity,
        remainingStock: logistics.stock - quantity,
        category: logistics.category,
        dateEnd: logistics.dateEnd,
        distributeDate: logistics.insertDate,
        imagePath: logistics.imgPath,
        officer: logistics.officer,
      );

      // Add record to logistikKeluar
      await _dbLogistikKeluar.add(distributedItem.toJson());

      Get.snackbar(...);
    }
    else {
      Get.snackbar(...);
    }
  } catch (e) {
    // Log the error
    debugPrint('Error distributing item: $e');
  }
}

```

Segmen Program 4.10 Method `distributeItem()` pada Class `LogisticDb`

Segmen Program 4.10 berisi kode yang berfungsi untuk menangani *insert* kedalam *collection* `logistikKeluar` di `Firestore`. Langkah pertama pada *method* `distributeItem()`, yaitu akan meng-*update* stok pada *collection* `logistikMasuk` untuk item logistik masuk yang akan dikeluarkan berdasarkan jumlah *input*-an yang dimasukkan pada halaman *input* item keluar, setelah itu akan melakukan proses *insert* kedalam *collection* `logistikKeluar` dengan nilai yang didapat dari *model* logistik, lalu menampilkan peringatan bahwa item telah berhasil di *input* atau gagal.

```

ElevatedButton()
onPressed: () {
  if(_formKey.currentState!.validate()){
    final quantityText = logisticDetailController.quantity.text.trim();
    // Check if the input contains only digits
    if (!RegExp(r'^\d+$').hasMatch(quantityText)) {...}
    final quantity = double.tryParse(quantityText);
    if (quantity == null) {...}
    else if(quantity <= 0){...}
    else {
      final availableStock = widget.data['Stok'].toInt();

      if (quantity > availableStock) {
        // Show an error message or handle the case where the entered quantity exceeds available stock
        Get.snackbar(
          "PERINGATAN!",
          "Input nilai melebihi stok yang tersedia. Mohon cek kembali!",
          snackPosition: SnackPosition.BOTTOM,
          backgroundColor: Colors.redAccent,
          colorText: Colors.white,
        );
        return;
      }
      final destination = logisticDetailController.destination.text.trim();
      setState() {
        logisticDetailController.distributeItemController(
          logistics,
          widget.data['id'],
          quantity,
          destination
        );
      };
      Get.offAll(() => const LogisticMain());
    };
  }
}

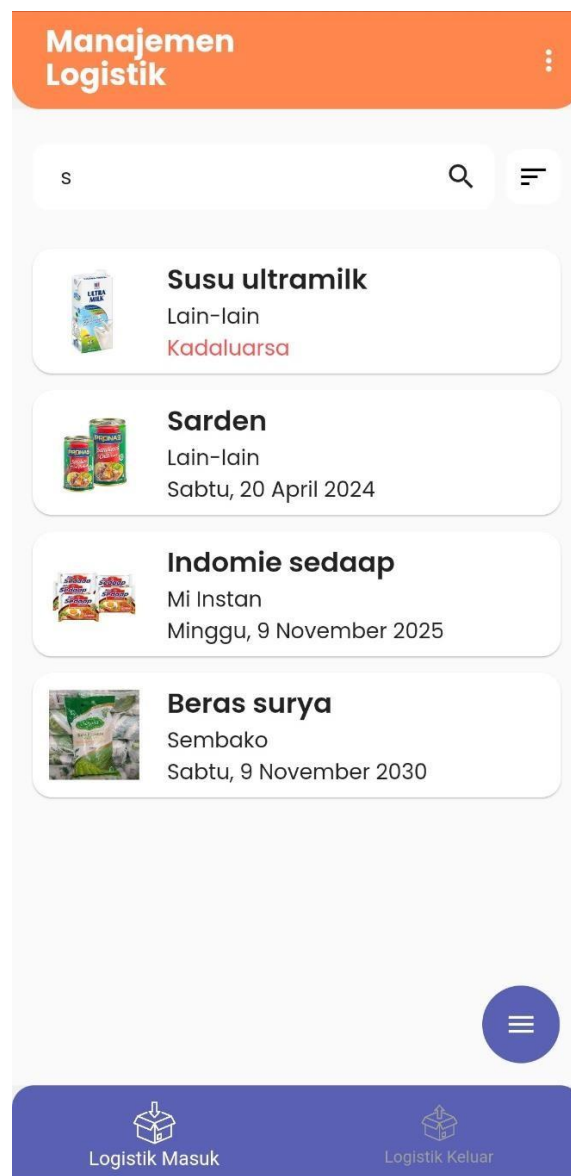
```

Segmen Program 4.11 Fungsi onPressed() *button input* barang keluar

Segmen Program 4.11 berisi kode yang berfungsi untuk menangani perintah yang akan dijalankan ketika tombol di-*input* barang keluar di klik. Terdapat beberapa kondisi yang digunakan untuk validasi *input*-an dan peringatan yang akan keluar. Apabila tidak ada masalah, akan menjalankan *method* distributeItemController yang terhubung dengan *method* distributeItem() yang ada pada *class* logisticDb() dengan mengirimkan beberapa parameter yang akan diproses didalam *method* distributeItem(). Setelah proses selesai, akan langsung dikembalikan ke halaman utama.

4.2.3.7 Pencarian Item Logistik

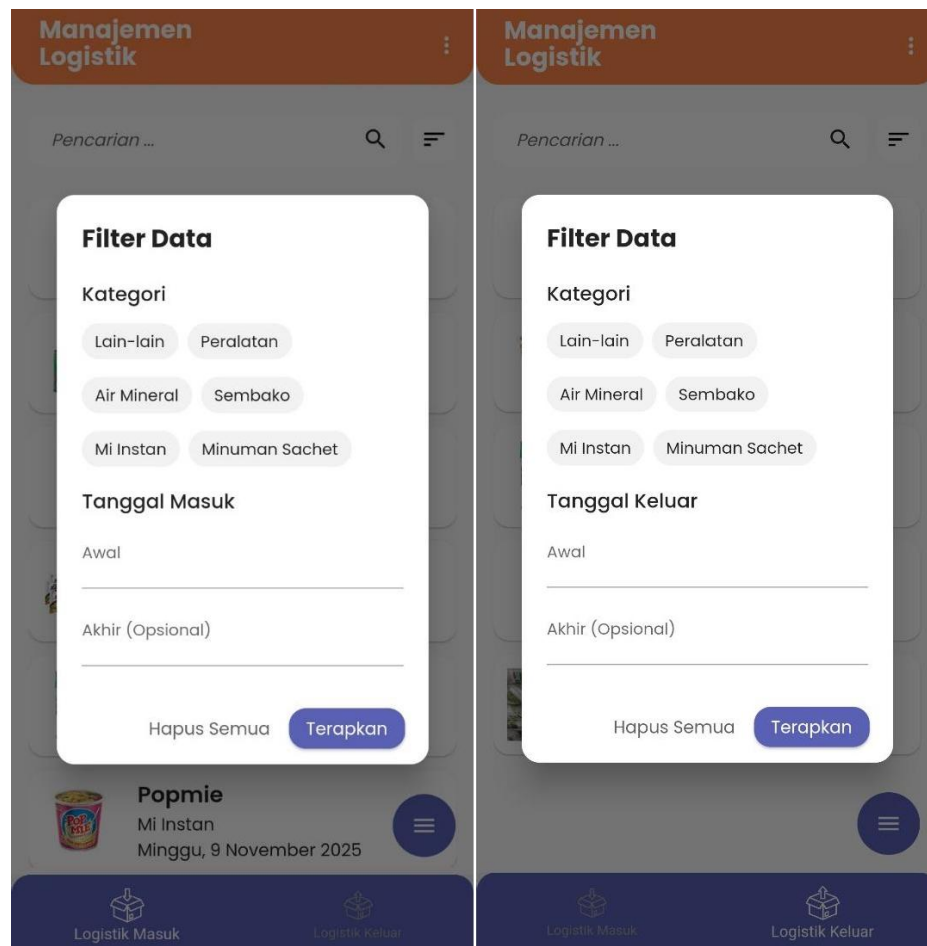
Berikut merupakan implementasi dari kode yang terdapat pada fungsi `getRecords()` yang menangani fitur pencarian yang ada pada aplikasi LogPal BPBD Kab.Malang. Petugas yang ingin mencari item dengan nama tertentu, dapat langsung mengetikkan kata kunci yang diinginkan melalui kotak pencarian, dan daftar item logistik akan langsung terbaru secara otomatis



Gambar 4.11 Daftar item logistik yang telah diperbarui

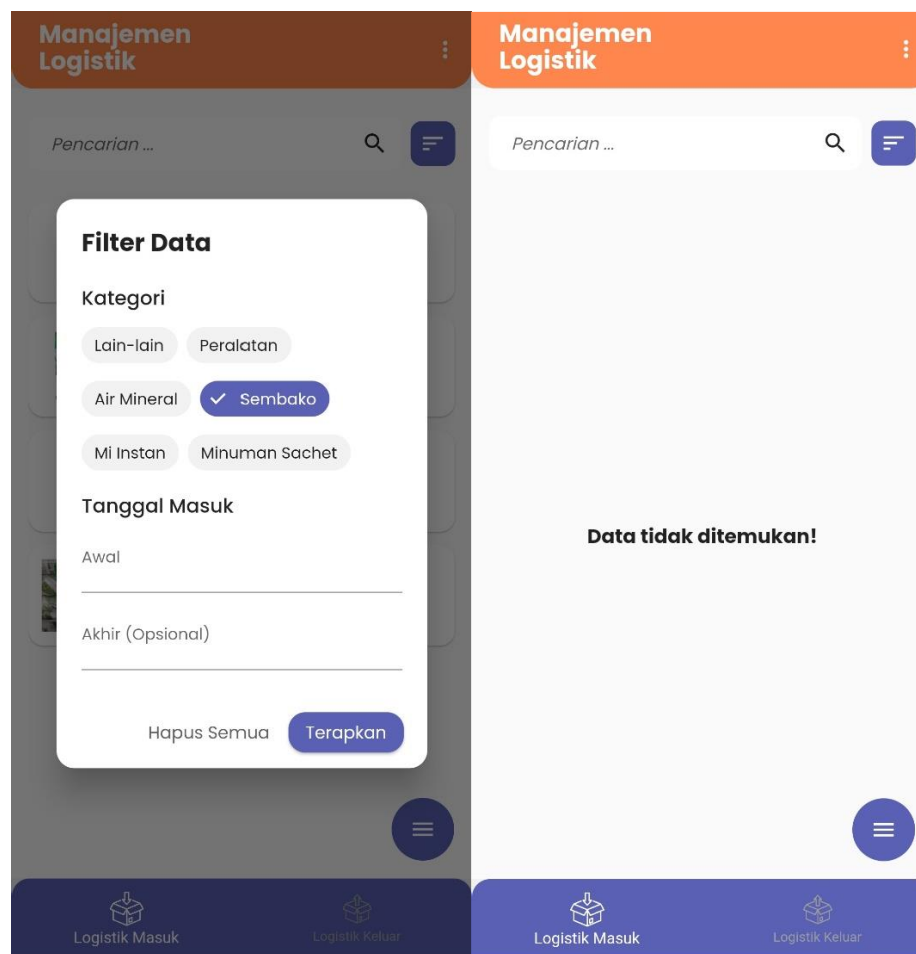
4.2.3.8 Filter Item Logistik

Berikut merupakan implementasi dari kode yang terdapat pada fungsi `getRecords()` yang menangani fungsi *filter* data pada aplikasi LogPal BPBD Kab.Malang. Petugas yang ingin menyaring item dengan opsi *filter* yang telah tersedia, dapat langsung memilih salah satu atau lebih opsi, berdasarkan Kategori maupun Tanggal Masuk atau Tanggal Keluar.



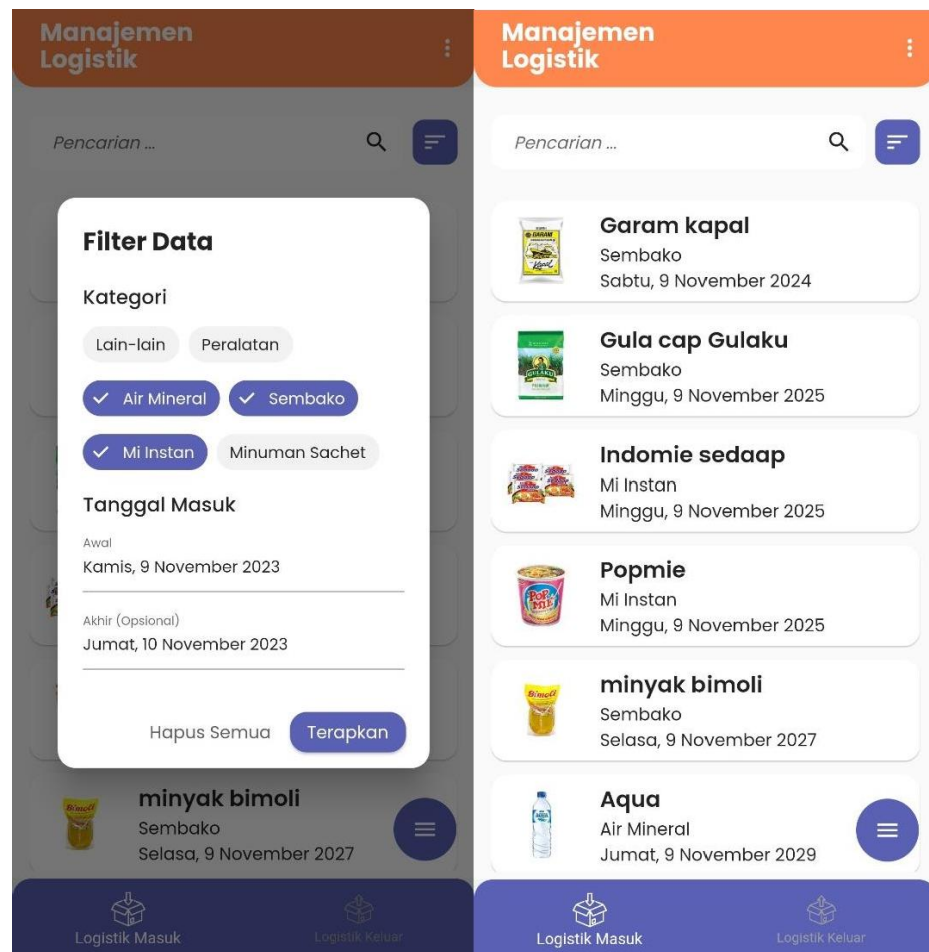
Gambar 4.12 Popup Filter Data

Berikut merupakan salah satu kondisi yang dapat terjadi saat melakukan *filter* data. Ketika petugas memilih opsi *filter* yang ada, namun sistem tidak memiliki atau tidak dapat mengirim hasil karena ketidaksediaan data yang tersimpan dalam *database*, maka daftar item logistik akan memunculkan *error* seperti gambar dibawah ini, yaitu ‘Data tidak dapat ditemukan’



Gambar 4.13 Penerapan *Filter* dengan hasil data yang kosong

Selain kondisi *error*, apabila opsi *filter* yang diterapkan memiliki hasil, *database* akan selalu menampilkan hasil sesuai dengan kondisi *filter* yang telah ditentukan, meskipun opsi *filter* digunakan secara menyeluruh seperti yang ada pada contoh gambar dibawah ini.



Gambar 4.14 Penerapan *Filter* dengan hasil data yang valid

4.2.3.9 Cetak laporan

Berikut merupakan *popup* cetak laporan logistik dari aplikasi LogPal BPBD Kab.Malang. Petugas dapat langsung mengklik tombol dengan ikon 'Pdf' yang berada pada tombol mengambang di menu logistik masuk maupun logistik keluar untuk mencetak laporan. Petugas juga dapat menerapkan *filter* terhadap daftar logistik yang akan dicetak, seperti gambar di sisi kanan yang berada dibawah.

The screenshot shows the 'Manajemen Logistik' interface with a grid of items. Two floating popups are visible, each displaying a report for a specific item. The left popup is for 'Susu ultramilk' and the right is for 'Garam kapal'. Both reports are from the 'PEMERINTAH KABUPATEN MALANG' and include a table with columns for No, Nama, Jumlah, Satuan, Jenis Barang, Tanggal Masuk, Kadaluarsa, and Asal Perolehan.

Laporan Logistik Masuk
Kategori: 10 November 2022

No	Nama	Jumlah	Satuan	Jenis Barang	Tanggal Masuk	Kadaluarsa	Asal Perolehan
1	Aqua	45	liter	Air Mineral	09/11/2023	09/11/2023	cv anjapa
2	Sarden	100	kaleng	Lain-lain	13/11/2023	29/04/2024	Bandung
3	Susu ultramilk	0	liter	Lain-lain	09/11/2023	09/11/2023	pt opna
4	kecap manis	400	liter	Ml kecap	09/11/2023	09/11/2023	pt sri
5	Pismania	25	liter	Ml Pismania	09/11/2023	09/11/2023	pt sri
6	Beras surya	90	kg	Berbekas	09/11/2023	09/11/2030	cv anjapa
7	Garam kapal	1	kg	Sembako	09/11/2023	09/11/2024	cv anjapa

Laporan Logistik Keluar
Kategori: 09 November 2022

No	Nama	Jumlah	Satuan	Jenis Barang	Tanggal Keluar	Detail Pengiriman	Stok Awal
1	Aqua	10	liter	Air Mineral	09/11/2023	cv	45
2	Pismania	10	liter	Ml Pismania	09/11/2023	bandung	0
3	Beras surya	11	kg	Berbekas	09/11/2023	bandung	90
4	Garam kapal	2	kg	Berbekas	09/11/2023	bandung	0
5	Gula cap Gulaku	2	kg	Berbekas	09/11/2023	cv	12

Gambar 4.15 *Pop-up* cetak laporan dengan data logistik masuk

```

Future<void> generatePDF() async {
    // Initialize the localization
    await initializeDateFormatting('id_ID', null);

    // Generate PDF and fetch a Logo
    final pdf = pw.Document();
    final ByteData image = await rootBundle.load(taSplashImage);
    Uint8List imageData = (image).buffer.asUint8List();

    // Specify the fields you want to include in the PDF
    List<String> fieldName = [
        'No', 'Nama', 'Jumlah',
        'Satuan', 'Jenis Barang', 'Tanggal Masuk',
        'Kadaluarsa', 'Asal Perolehan'
    ];
    List<String> fieldContentsFrom = [
        'Nama Barang', 'Stok',
        'Satuan', 'Kategori',
        'Tanggal Masuk', 'Tanggal Kadaluarsa',
        'Asal Perolehan'];

    DateTime dateNow = DateTime.now();
    String formattedDateNow = DateFormat('EEEE, d MMMM yyyy', 'id_ID').format(dateNow);

    // Set the number of rows you want to display on each page
    const int firstPageRow = 7;
    const int otherPagesRow = 8;
    // Calculate the number of pages needed for subsequent pages
    final int otherPagesCount = ((_allResults.length - firstPageRow) / otherPagesRow).ceil();

    // Table contents column widths
    Map<int, pw.TableColumnWidth> customColumnWidths = {...};

    //re-Align the ordering of all the items
    _allResults.sort((a, b) {...});

    // First Page
    pdf.addPage(
        pw.MultiPage(
            pageTheme: pw.PageTheme(
                orientation: pw.PageOrientation.landscape,
                pageFormat: PdfPageFormat.a4.landscape,
            ), // pw.PageTheme
            build: (context) {...},
        ), // pw.MultiPage
    );

    // Subsequent pages
    for (int currentPage = 1; currentPage <= otherPagesCount; currentPage++) {...}
}

```

Segmen Program 4.12 *Method generatePdf()* untuk membuat tabel

Segmen Program 4.12 berisi kode yang berfungsi untuk membuat tabel yang datanya berasal dari list item logistik yang telah didapatkan dari Firestore Database untuk ditampilkan pada dokumen berformat Pdf. Terdapat beberapa proses seperti *import* logo instansi, pendefinisian nama *field* pada tabel yang disimpan pada *array* *fieldName*, mendapatkan beberapa data dari Firestore yang dipilih dari pendefinisian nama *field* pada yang disimpan pada *array* *fieldContentsFrom* dan akan ditampilkan pada *record* dari tabel yang nantinya akan dicetak.

```
// Save the PDF to Documents directory
final output = await getApplicationDocumentsDirectory();
final file = File("${output.path}/Laporan Logistik Masuk.pdf");

final bytes = await pdf.save(); // Await here to get the actual bytes
await file.writeAsBytes(bytes);

final pdfPinchController = pdfx_show.PdfControllerPinch(
  document: pdfx_show.PdfDocument.openFile(file.path),
);

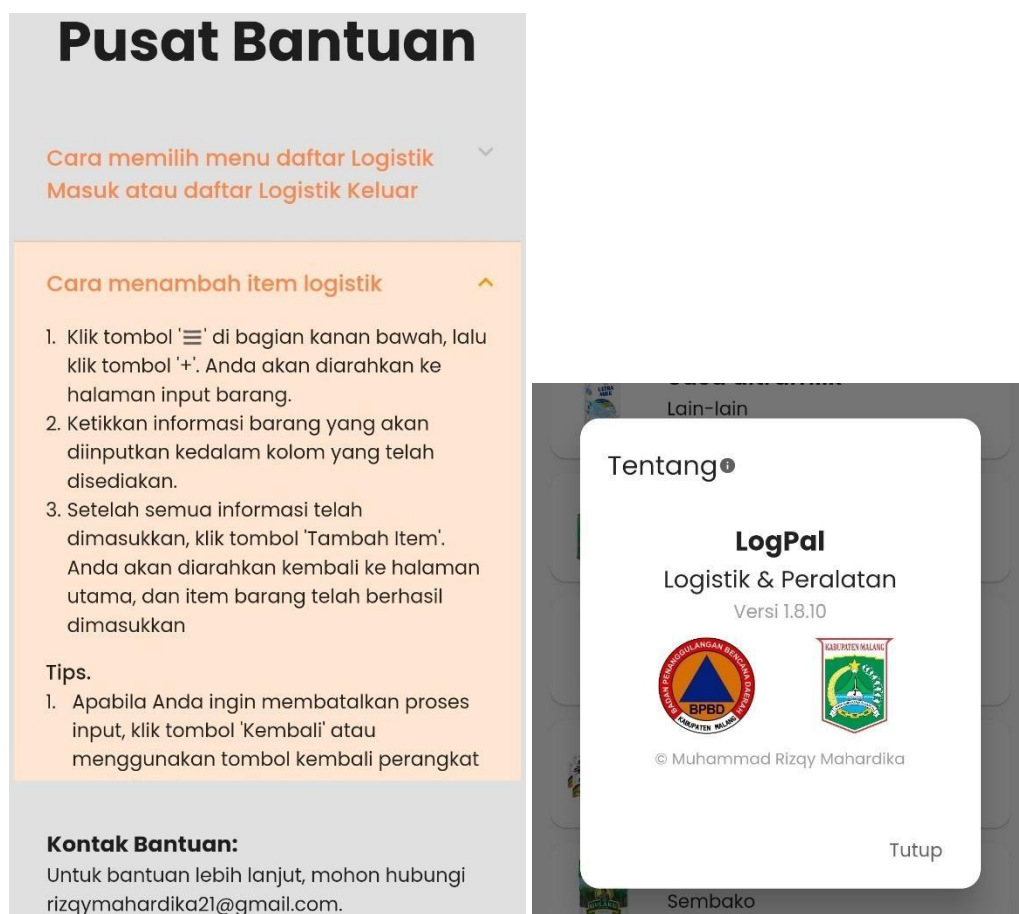
if(context.mounted){
  await showDialog(
    context: context,
    builder: (context) {...},
  );
}
```

Segmen Program 4.13 Fungsi menampilkan pratinjau

Segmen Program 4.13 berisi kode yang berfungsi untuk *generate* fail pdf yang dapat ditampilkan melalui *dialog* pratinjau sebelum dapat dibagikan. Dokumen akan disimpan dengan nama sedemikian rupa, lalu dokumen tersebut akan ditampilkan kedalam *popup dialog*.

4.2.3.10 Halaman Pusat Bantuan dan Tentang

Berikut merupakan halaman Pusat Bantuan dan Tentang aplikasi LogPal BPBD Kab.Malang. Pada halaman pusat bantuan, terdapat beberapa informasi terkait cara penggunaan aplikasi, beserta tips yang dapat membantu dalam menangani kesalahan saat penggunaan aplikasi.



Gambar 4.16 Halaman Bantuan & Tentang

4.3 Uji Coba

4.3.1 Blackbox Testing

Berikut ini merupakan uji coba menggunakan metode *blackbox testing* pada sistem manajemen logistik LogPal BPBD Kab.Malang

Tabel 4.1 Pengujian Blackbox

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
Buka Aplikasi	Validasi aplikasi dapat dijalankan	Buka aplikasi LogPal sebelum login	Menampilkan halaman "Log In"	Valid
		Buka aplikasi LogPal sesudah login	Menampilkan halaman utama aplikasi LogPal	Valid
Log In	Validasi fungsi Login	Buka aplikasi LogPal	Pengguna dapat melihat halaman "Log In"	Valid
		Isi formulir login dengan data akun yang telah terdaftar (email dan password yang sesuai)	Pengguna berhasil masuk ke akun	Valid
		Klik tombol "Masuk"	Masuk kedalam halaman utama aplikasi	Valid
	Validasi data tidak valid saat Log In	Isi formulir login dengan data akun yang tidak valid (email atau password salah)	Sistem memberikan pesan error terkait data yang tidak valid	Valid
		Klik tombol "Masuk"	Pengguna tetap berada di halaman Login	Valid
Manajemen Logistik	Validasi tampil data item logistik masuk	Halaman utama secara default akan memilih menu Logistik Masuk	Pengguna dapat melihat daftar item logistik keluar yang berbentuk daftar	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
			yang dapat di scroll	
	Validasi fungsi menambahkan item logistik masuk	Klik <i>floating action button</i> pada pojok kanan bawah halaman	Pengguna dapat melihat pilihan tambahan	Valid
		Klik tombol "+"	Pengguna diarahkan ke halaman "Input Barang"	Valid
		Isi formulir <i>input</i> barang dengan data valid	Informasi item berhasil ditampilkan kedalam formulir	Valid
		Klik tombol "Tambah Item"	Muncul peringatan bahwa informasi item berhasil ditambahkan, lalu pengguna diarahkan kembali ke halaman utama. Daftar logistik di halaman utama menu logistik masuk juga akan otomatis terbaru	Valid
		Klik tombol "Kembali"	Muncul popup konfirmasi	Valid
		Klik "Ya" pada <i>Popup</i> konfirmasi batal <i>input</i> item	Informasi item tidak jadi ditambahkan, lalu pengguna diarahkan kembali ke halaman utama	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
		Klik ” Tidak” pada <i>Popup</i> konfirmasi batal <i>input</i> item	<i>Popup</i> tertutup	Valid
	Validasi data tidak valid saat menambahkan informasi item barang	Isi formulir <i>input</i> barang dengan data tidak valid lalu klik tombol "Tambah Item"	Formulir akan memberikan peringatan error sesuai ketidakvalidan data dan pengguna tetap berada di halaman <i>input</i> barang	Valid
	Validasi fungsi detail halaman item logistik masuk	Klik salah satu item yang tersedia dalam daftar logistik masuk	Pengguna akan diarahkan ke halaman detail dari item logistik tersebut	Valid
		Klik tombol “Lanjutkan untuk barang keluar”	<i>Dialog popup</i> untuk mengisi informasi tambahan akan muncul	Valid
		Mengisi data yang valid pada <i>popup</i> informasi tambahan, lalu klik “Konfirmasi”	<i>Dialog popup</i> akan tertutup, dan pengguna akan diarahkan kembali ke halaman utama. Daftar item logistik keluar yang ada pada halaman Logistik Keluar juga akan otomatis terbaru	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
		Mengisi data yang tidak valid pada <i>popup</i> informasi tambahan Klik tombol “Konfirmasi”	Muncul peringatan <i>error</i> , dan pengguna tidak melanjutkan proses sebelum data valid	Valid
		Klik tombol “Batal” pada <i>popup</i> informasi tambahan	<i>Dialog popup</i> akan tertutup, kembali ke halaman detail	Valid
	Validasi tampil data item logistik keluar	Memilih menu Logistik Keluar	Pengguna dapat melihat daftar item logistik keluar yang berbentuk daftar yang dapat di scroll	Valid
	Validasi fungsi detail halaman item logistik keluar	Klik salah satu item yang tersedia dalam daftar logistik keluar	Pengguna akan diarahkan ke halaman detail dari item logistik tersebut. Halaman detail item logistik keluar tidak memiliki tombol “Lanjutkan untuk barang keluar”	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
Cetak Laporan	Validasi fungsi cetak laporan	Klik <i>floating action button</i> pada pojok kanan bawah halaman, lalu, klik tombol icon pdf	Pengguna dapat melihat <i>dialog popup</i> yang muncul	Valid
		Melakukan gestur gerakan memperbesar layar (<i>pinch to zoom</i>) ke layar pratinjau	Pengguna dapat memperbesar ukuran layar pratinjau	Valid
		Klik tombol “Bagikan”	Pengguna dapat melihat berbagai opsi pilihan untuk membagikan dokumen	Valid
		Klik salah satu opsi pilihan tujuan bagikan dokumen	Pengguna akan diarahkan ke proses membagikan yang selanjutnya	Valid
		Klik tombol “Tutup Pratinjau”	<i>Dialog popup</i> layar pratinjau akan segera tertutup	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
Pencarian & Filter data logistik	Validasi fungsi pencarian	Mengetik kata kunci di bilah pencarian	Pengguna dapat melihat hasil dari pencarian yang dilakukan. Apabila data yang dicari tidak ada atau valid, daftar item akan berganti menjadi peringatan "Data tidak ditemukan"	Valid
	Validasi fungsi <i>filter</i> data logistik	Menekan tombol <i>filter</i> disebelah kanan bilah pencarian	<i>Dialog popup</i> Filter Data akan muncul dengan berbagai macam pilihan untuk <i>filter</i> data	Valid
		Memilih satu atau lebih <i>chip</i> pilihan kategori yang tersedia	Warna dan tulisan <i>chip</i> akan berubah menjadi aktif/terpilih	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
		Memilih kembali satu atau lebih <i>chip</i> kategori yang telah dipilih sebelumnya	Warna dan tulisan <i>chip</i> akan berubah menjadi nonaktif/tidak terpilih	Valid
		Klik <i>text field</i> tanggal masuk “Awal”	Tampil <i>date picker</i> yang dapat dipilih sesuai keinginan	Valid
		Memilih tanggal melalui <i>Date picker</i> tanggal masuk awal	<i>Text field</i> tanggal masuk “Awal” terbaru sesuai masukan <i>date picker</i> tanggal masuk “Awal”	Valid
		Klik <i>text field</i> tanggal masuk “Akhir”	Tampil <i>date picker</i> yang dapat dipilih sesuai keinginan	Valid
		Memilih tanggal melalui <i>Date picker</i> tanggal masuk akhir	<i>Text field</i> tanggal masuk “Akhir” terbaru sesuai masukan <i>date picker</i> tanggal masuk “Akhir”	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
		Memilih opsi <i>filter</i> yang valid lalu klik tombol “Terapkan”	<i>Dialog popup</i> Filter Data akan tertutup, dan akan kembali ke halaman menu logistik masuk/keluar yang telah memiliki daftar yang telah ter <i>filter</i> yang sesuai	Valid
		Memilih opsi <i>filter</i> yang tidak valid lalu klik tombol “Terapkan” (Tanggal Akhir lebih dahulu daripada Tanggal Awal atau Tanggal Awal dan Akhir sama)	Muncul peringatan <i>error</i> sesuai dengan kesalahan <i>input</i> atau pilihan dan <i>Dialog popup</i> Filter Data akan tetap terbuka	Valid
		Klik tombol “Hapus Semua”	Semua pilihan opsi <i>filter</i> akan terhapus	Valid
Pusat Bantuan & Tentang	Validasi halaman Pusat Bantuan	Klik tombol opsi pada pojok kanan atas halaman utama, lalu klik tombol ‘Bantuan’	Tampil halaman Pusat Bantuan	Valid

Fitur	Skenario Pengujian	Langkah pengujian	Hasil Yang Diharapkan	Hasil Pengujian
	Validasi halaman Tentang	Klik tombol opsi pada pojok kanan atas halaman utama, lalu klik tombol 'Tentang'	Tampil <i>popup</i> Tentang	Valid
<i>Log Out</i>	Validasi fungsi <i>Logout</i>	Klik tombol ikon keluar/ <i>logout</i>	Pengguna dapat melihat <i>dialog popup</i> konfirmasi yang muncul	Valid
		Klik tombol "Ya"	Sesi pengguna akan berakhir, dan akan dikembalikan ke halaman <i>login</i>	Valid
		Klik tombol "Tidak"	<i>Dialog popup</i> konfirmasi akan tertutup, dan sesi pengguna akan terus berlanjut	Valid

Berdasarkan hasil pengujian *black box* yang telah dilakukan dengan beberapa poin uji, aplikasi yang telah dikembangkan telah memenuhi hasil yang diharapkan.

4.3.2 User Acceptance Testing

Tahap pengujian menggunakan *User Acceptance Testing* (UAT) dilakukan dengan memberikan pertanyaan kuisisioner menggunakan *platform* Google Form kepada responden terkait. Responden terdiri dari petugas logistik BPBD Kabupaten Malang sebagai pengguna aplikasi dan masyarakat umum. Sebelum melengkapinya kuisisioner, responden memiliki opsi untuk mengunduh aplikasi percobaan melalui tautan yang telah dilampirkan pada formulir untuk menguji aplikasi terlebih dahulu secara mandiri agar dapat memberikan tanggapan yang obyektif terhadap pertanyaan yang diberikan. Total hingga sekarang sejak formulir dibagi yaitu sejumlah 32 orang responden yang terlibat dalam mengisi kuisisioner. Berikut adalah hasil evaluasi kegunaan berdasarkan kuisisioner yang telah diberikan:

Tabel 4.2 Hasil Kuisisioner UAT

No.	Variabel	Pertanyaan	Nilai				
			1	2	3	4	5
1	Halaman Login Aplikasi "LogPal BPBD Kab.Malang"	Apakah fitur Login dalam Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	1	11	20
2		Apakah halaman Login dalam Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	4	18	10
3		Apakah fitur Login dalam Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	1	3	13	15
4	Halama Utama Aplikasi "LogPal BPBD Kab.Malang"	Apakah halaman utama Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	1	3	15	13
5		Apakah tampilan list item Logistik Masuk dan list item Keluar yang dipisah menjadi dua halaman yang berbeda pada Aplikasi "LogPal BPBD Kab.Malang" dapat memudahkan	0	1	2	13	16

		Anda dalam memahami informasi yang tersedia?					
6		Apakah data Logistik Masuk dan Keluar yang ditampilkan didalam Aplikasi "LogPal BPBD Kab.Malang" bersifat informatif?	0	0	1	14	17
7		Apakah halaman utama Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	12	18
8		Apakah halaman utama Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	12	17
9	Fitur Pencarian pada Aplikasi "LogPal BPBD Kab.Malang"	Apakah pencarian pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	2	17	13
10		Apakah fitur pencarian pada Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	2	15	15
11		Apakah fitur pencarian Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	12	18
12		Apakah fitur pencarian Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	4	10	18
13	Fitur Filter Data pada Aplikasi "LogPal BPBD Kab.Malang"	Apakah popup Filter Data pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	1	18	13
14		Apakah fitur filter data Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	16	14
15		Apakah fitur filter data Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	3	14	15
16		Apakah fitur filter data Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	10	19

17	Fitur Cetak Laporan Aplikasi "LogPal BPBD Kab.Malang"	Apakah popup pratinjau cetak laporan pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	3	14	15
18		Apakah fitur cetak laporan Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	1	15	16
19		Apakah fitur cetak laporan Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	3	12	17
20		Apakah fitur cetak laporan Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	10	19
21	Fitur Input Barang pada Aplikasi "LogPal BPBD Kab.Malang"	Apakah halaman <i>input</i> barang pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	1	19	12
22		Apakah <i>input</i> barang Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	4	10	18
23		Apakah fitur unggah gambar pada halaman <i>input</i> barang pada Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	3	11	18
24		Apakah fitur <i>input</i> barang Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	3	9	20
25		Apakah fitur <i>input</i> barang Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	9	20
26	Halaman Detail Barang Masuk Aplikasi "LogPal BPBD Kab.Malang"	Apakah halaman detail barang masuk pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	1	1	13	17
27		Apakah halaman detail barang masuk Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	12	18

28		Apakah data yang ditampilkan pada halaman detail masuk barang pada Aplikasi "LogPal BPBD Kab.Malang" bersifat informatif?	0	0	2	12	18
29		Apakah halaman detail barang masuk Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	2	11	19
30		Apakah halaman detail barang masuk Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	4	9	19
31		Apakah popup <i>input</i> barang keluar yang terletak didalam halaman detail barang masuk pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	3	15	14
32	Fitur Input Barang Keluar pada Aplikasi "LogPal BPBD Kab.Malang"	Apakah popup <i>input</i> barang keluar yang terletak didalam halaman detail barang masuk pada Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	10	20
33		Apakah popup <i>input</i> barang keluar yang terletak didalam halaman detail barang masuk pada Aplikasi "LogPal BPBD Kab.Malang" dapat berfungsi secara efektif?	0	0	3	9	20
34		Apakah popup <i>input</i> barang keluar yang terletak didalam halaman detail barang masuk pada Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	13	16
35		Apakah halaman detail barang keluar pada Aplikasi "LogPal BPBD Kab.Malang" memiliki tampilan yang menarik?	0	0	2	15	15
36	Halaman Detail Barang Keluar Aplikasi "LogPal BPBD Kab.Malang"	Apakah halaman detail barang keluar Aplikasi "LogPal BPBD Kab.Malang" mudah untuk dipahami dan digunakan?	0	0	2	11	19
37		Apakah data yang ditampilkan pada halaman detail keluar barang pada Aplikasi "LogPal BPBD Kab.Malang" bersifat informatif?	0	0	3	9	20

38		Apakah halaman detail barang keluar Aplikasi "LogPal BPBD Kab.Malang" bermanfaat bagi Anda?	0	0	3	10	19
----	--	---	---	---	---	----	----

Hasil kuisioner diatas dianalisis dengan menghitung rata-rata jawaban dari total 38 buah poin pertanyaan, berdasarkan skor yang diperoleh dari jawaban setiap responden.

Tabel 4.3 Analisis Hasil Kuisioner UAT

Angka	Skala Penilaian	Jumlah	Skor	Jumlah Skor
5	Sangat Setuju	640	5	3200
4	Setuju	478	4	1912
3	Ragu-ragu	94	3	282
2	Tidak Setuju	4	2	8
1	Sangat Tidak Setuju	0	1	0
Nilai Total				5402
Nilai Maksimal				6080

Untuk menghitung aspek *usability* digunakan rumus sebagai berikut:

$$Presentase = \frac{\text{nilai total}}{\text{nilai maksimal}} \times 100\%$$

Dengan menggunakan data dari tabel 4.4, maka akan didapatkan:

$$Presentase = \frac{5402}{6080} \times 100\% = 88.85\%$$

Hasil yang didapatkan apabila diubah menjadi presentase, akan mendapatkan hasil sebesar 88,84868421%, atau apabila dibulatkan menjadi 89%, sehingga dapat disimpulkan aspek *usability* dari aplikasi LogPal BPBD Kab.Malang mendapatkan skor sebesar 89%.