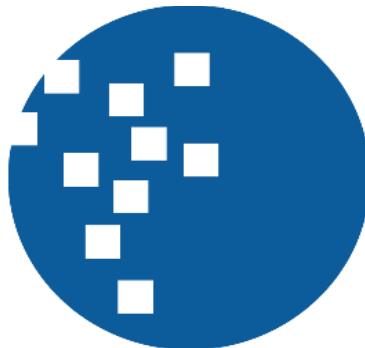


**LAPORAN FINAL PROJECT
MOBILE APPLICATION DEVELOPMENT
IS 534-E**



RANCANG BANGUN APLIKASI LAUNDRY-IN

Nama Anggota:

1. Rhauma Syira Anggina (00000079236)
2. Haniva Yuniar Praja P (00000082522)
3. Lola Naomi Enzelin M (00000081913)
4. Kayla Abigail Gunawan (00000081824)

**INFORMATION SYSTEM STUDY PROGRAM
FACULTY OF ENGINEERING AND INFORMATION
UNIVERSITAS MULTIMEDIA NUSANTARA
BANTEN
2023**

DAFTAR ISI

DAFTAR ISI.....	2
BAB I PENDAHULUAN.....	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan dan Manfaat Aplikasi.....	4
1.4 Proses Bisnis.....	4
BAB II ANALISIS DAN HASIL PERANCANGAN.....	6
2.1 Perancangan	6
2.2 Analisis Perancangan Sistem	9
BAB III SET-UP APLIKASI.....	14
3.1 Set-Up Aplikasi.....	14
3.2 Implementasi Basis Data	16
3.3 Penggunaan Firebase - Database.....	20
3.4 Log-In	30
3.5 Kelas CustomerModel.....	33
BAB IV IMPLEMENTASI APLIKASI.....	35
4.1 Implementasi Aplikasi	35
4.1.1 Spinner Activity	35
4.1.2 Log-In	36
4.1.3 Register	40
ROLE OF PROJECT	98

BAB I

PENDAHULUAN

1.1 Latar Belakang

Usaha Laundry berkembang pesat dalam beberapa tahun terakhir. Strategi bisnis yang tepat adalah krusial dalam menghadapi persaingan yang semakin sengit di era digital ini. Mengadopsi teknologi menjadi suatu keharusan. Beberapa startup telah mencoba menggunakan sistem pemesanan jasa laundry berbasis Android. Ini menunjukkan bahwa inovasi terus berkembang seiring kemajuan teknologi.

Untuk merancang aplikasi laundry berbasis Android, penelitian sangat penting. Tujuannya adalah agar aplikasi mampu menampilkan informasi dari penyedia jasa laundry serta memberikan kemudahan dalam proses pelayanan bagi konsumen dan penyedia jasa laundry. Aplikasi ini diharapkan bisa meningkatkan efisiensi, kenyamanan, dan kualitas layanan secara keseluruhan.

Rancangan aplikasi harus mempertimbangkan berbagai aspek, seperti antarmuka pengguna yang intuitif, sistem manajemen pesanan yang efisien, integrasi pembayaran yang aman, serta fitur yang memudahkan komunikasi antara konsumen dan penyedia jasa laundry. Keamanan data juga menjadi hal yang sangat penting, karena aplikasi akan mencakup informasi pribadi pengguna.

Selain itu, strategi pemasaran juga menjadi kunci. Memperkenalkan aplikasi ini kepada konsumen potensial dan memastikan penyedia jasa laundry terlibat secara aktif dalam platform ini dapat menjadi langkah penting dalam memastikan keberhasilan aplikasi.

Dengan melibatkan teknologi dalam bisnis laundry, diharapkan dapat menciptakan pengalaman yang lebih baik bagi konsumen sambil membantu para penyedia jasa laundry untuk meningkatkan operasional mereka secara keseluruhan.

1.2 Rumusan Masalah

1. Melihat bagaimana cara untuk meningkatkan akses konsumen terhadap informasi penyedia jasa laundry melalui aplikasi?
2. Mengintegrasikan sistem pembayaran yang aman dalam aplikasi untuk memudahkan proses transaksi antara konsumen dan penyedia jasa laundry?
3. Bagaimana merancang strategi pemasaran yang efektif?

1.3 Tujuan dan Manfaat Aplikasi

1.3.1 Tujuan

1. Membuat aplikasi laundry berbasis android yang dapat memudahkan konsumen serta penyedia jasa laundry.
2. Membuat aplikasi yang dapat memberikan laporan transaksi dengan mudah dan akurat.
3. Membuat sistem yang mampu merekam semua proses pengelolaan laundry.

1.3.2 Manfaat

1. Memudahkan konsumen dalam mengakses informasi laundry.
2. Memudahkan penyedia jasa laundry dalam melihat proses pengelolaan laundry.
3. Memudahkan konsumen menerima laporan transaksi.

1.4 Proses Bisnis

1.4.1 Customer

Tabel 1.4.1. Proses Bisnis *Customer*

NO	PROSES BISNIS
1.	Customer [user non-admin] akan melakukan Sign-Up dengan mengisi data pribadi.
2.	Apabila telah melakukan Sign-Up, maka yang akan ditampilkan adalah Log-In dan akan masuk ke tampilan Home.
3.	Di Halaman homepage, Customer dapat memilih kategori, kategori berisi <i>Complete</i> (Cuci - Setrika), <i>Dry-Clean</i> (Cuci Kering), Setrika (Hanya Setrika), ketika kategori dipilih, akan menampilkan daftar harga per-item.
4.	Customer dapat memasukkan pesanan ke keranjang terlebih dahulu.
5.	Keranjang berfungsi apabila Customer ingin merubah (<i>Update</i>) pesanan.

6.	Apabila <i>Customer</i> ingin langsung <i>checkout</i> , <i>Customer</i> langsung diarahkan ke halaman pembayaran lalu ke halaman <i>invoice</i> , begitu pula dengan <i>Customer</i> yang langsung ke halaman keranjang.
----	---

1.4.2 Seller

Tabel 1.4.2. Proses Bisnis *Seller*

NO	PROSES BISNIS
1.	Seller [user admin] melakukan Log-in, lalu akan diarahkan ke page <i>Seller Home page</i> .
2.	Page tersebut akan muncul seluruh Transaksi Customer dan Request Cancel Order.
3.	Seller dapat melihat laporan seluruh transaksi yang telah berhasil oleh Customer, agar dapat melihat perkembangan profit bisnis.

BAB II

ANALISIS DAN HASIL PERANCANGAN

2.1 Perancangan

2.1.1 Tema dan Ikon

a) Nama Aplikasi: Laundry-In

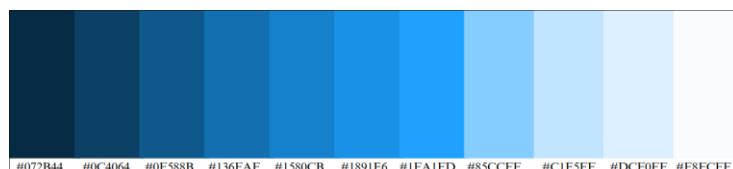
Aplikasi “Laundry-In” diibaratkan seperti pelanggan yang ‘mengundang’ atau ‘mengantar’ cucian mereka ke dalam layanan laundry secara praktis dan efisien. kata “In” dalam nama aplikasi ini menggambarkan adanya proses pemasukan atau penerimaan pakaian oleh pemilik usaha laundry, sehingga pelanggan dapat dengan mudah dan cepat mencuci pakaian mereka tanpa harus repot. Hal ini menunjukkan kesederhanaan dan kemudahan yang ditawarkan oleh aplikasi dalam mengelola layanan laundry.

b) Logo



Gambar 2.1.1.1 Logo

c) Warna



Gambar 2.1.1.2 Color Palette

Seperti pada gambar 2.1.2 dapat terlihat pemilihan warna biru menunjukkan ketenangan, profesionalisme, dan keterbukaan akan inovasi. Warna biru sangat sering dikaitkan dengan stabilitas dan kepercayaan yang dimana dapat mencerminkan kepercayaan yang dimiliki oleh pengguna aplikasi “Laundry-In” terhadap layanan ini. Hal ini dapat memberikan kesan profesional dan dapat diandalkan dengan memberikan pelayanan pelanggan yang baik dan hasil cucian yang berkualitas tinggi.

d) Font

Fonts

UBUNTU

Poppins

Laundry-In

Laundry-In

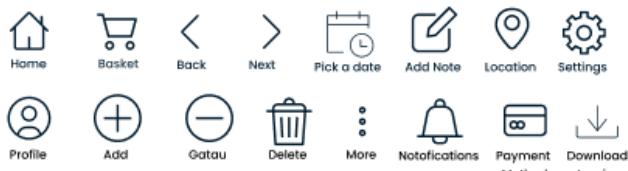
Gambar 2.1.1.3 Color Palette

Seperti pada gambar 2.1.3 fonts Ubuntu dipilih untuk logo aplikasi Laundry-in karena desainnya yang modern, jelas, dan mudah dibaca. Fonts ini memberikan kesan teknologi yang sesuai dengan identitas perusahaan, mencerminkan kekinian yang ingin dihadirkan oleh aplikasi Laundry-in. Desain simpel dan profesionalnya membuatnya ideal untuk mencerminkan citra perusahaan dengan baik.

Fonts Poppins dipilih sebagai fonts utama pada desain aplikasi Laundry-in karena mudah dibaca, ramah pengguna, dan tetap terlihat baik bahkan pada layar kecil. Fonts ini memiliki desain elegan yang memberikan kesan profesional pada tampilan teks aplikasi. Kelebihan lainnya adalah variasi berat huruf yang dimiliki Poppins, memungkinkan kita untuk menekankan bagian penting seperti judul dengan tebal dan mempertahankan kejelasan pada teks isi dengan versi reguler. Dengan Poppins, tata letak teks dalam aplikasi terlihat menarik dan teratur, meningkatkan pengalaman pengguna secara keseluruhan.

e) Ikon

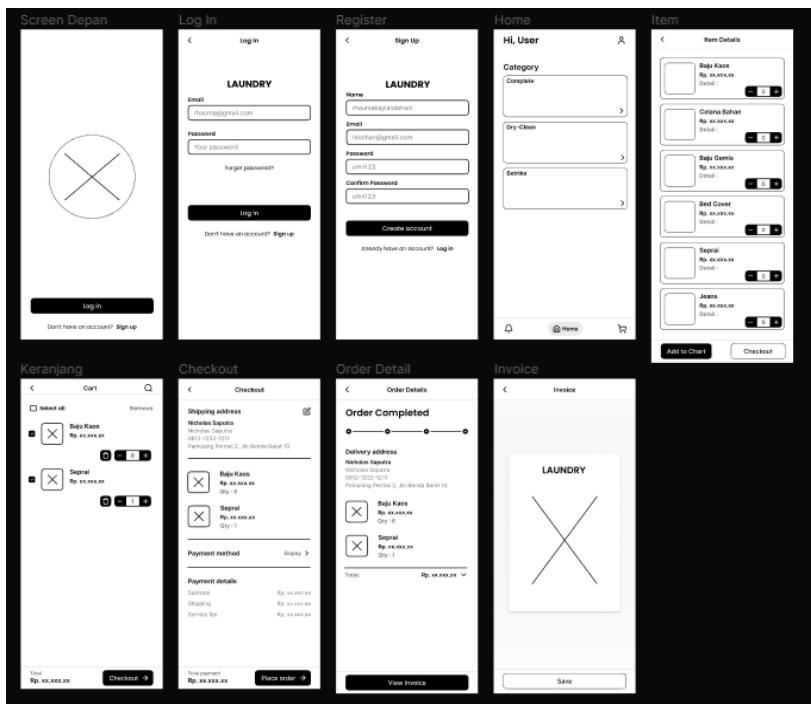
Icon



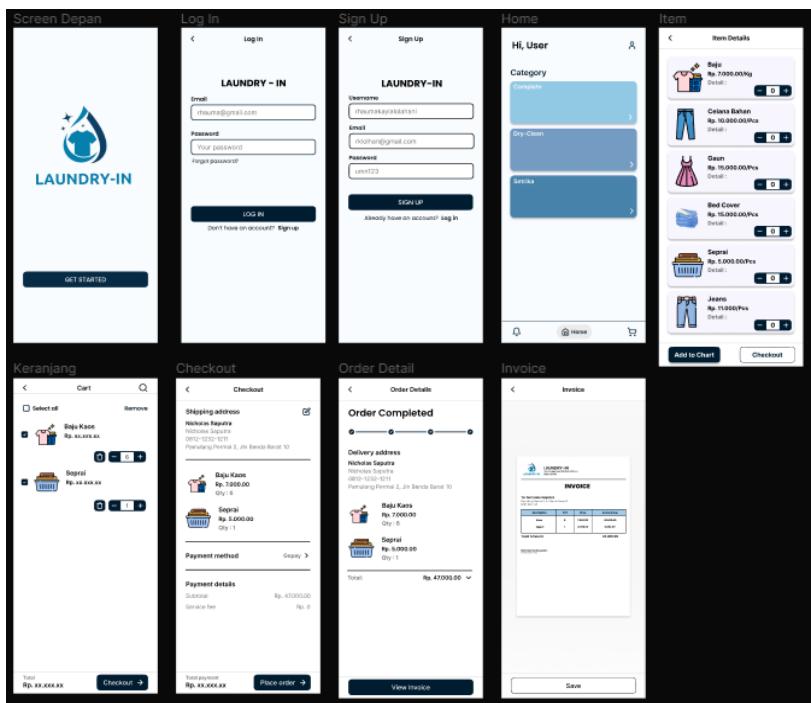
Gambar 2.1.1.4 Ikon

Ikon yang terlampir merupakan ikon yang digunakan untuk kebutuhan aplikasi Laundry-In. Ikon tersebut tidak hanya mempercantik tampilan, namun juga sebagai tanda arahan, sehingga user saat menggunakan Aplikasi tahu tujuan ikon tersebut.

2.1.2 Wireframe dan Prototyping



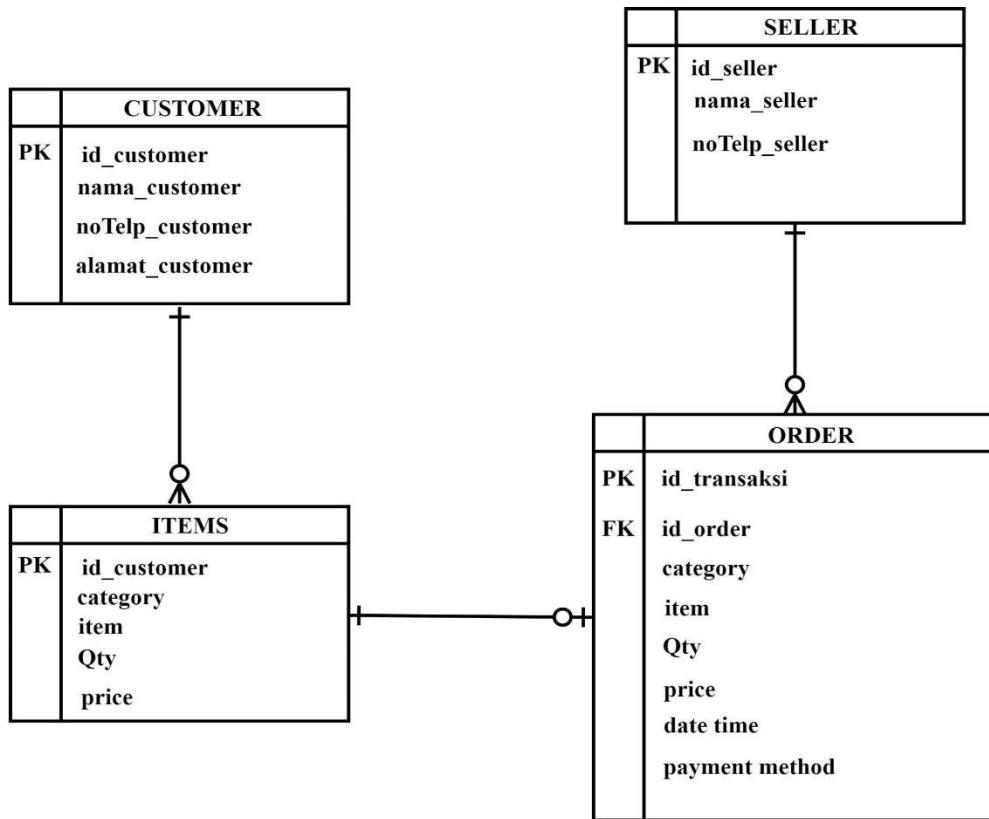
Gambar 2.1.2.1 Wireframe Laundry-In



Gambar 2.1.2.2 Prototype Laundry-In

2.2 Analisis Perancangan Sistem

2.2.1 Diagram UML



Gambar 2.2.1.1 Diagram UML

Terlampir diagram UML pada gambar 2.2.1.1 untuk aplikasi Laundry-In. Berikut merupakan detail mengenai tabel tersebut:

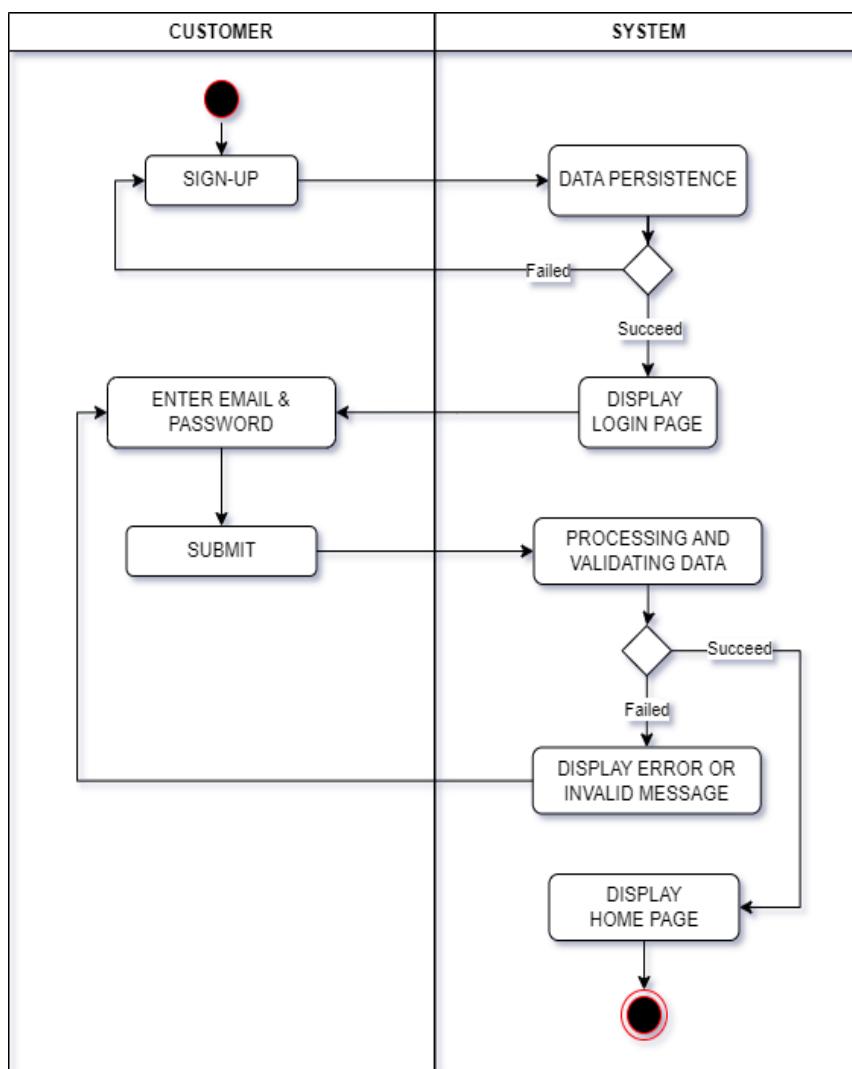
- **Tabel Customer** yang merupakan database Customer. Entity berisikan id_customer (Primary Key), nama_customer, noTelp_customer, serta alamat_customer.
- **Tabel Item** yang berisikan id_customer (Primary Key), category (Kategori), item (Barang), Qty (Jumlah), dan price (Harga), yang merupakan database pesanan customer. Setiap Order dapat dipastikan dilakukan oleh satu akun Customer dan setiap akun Customer dapat melakukan Order banyak atau berkali-kali [**One to Zero or Many**].
- **Tabel Order** yang akan berguna untuk data *invoice*. Terdapat id_transaksi (Primary Key), id_order (Reference from Table Customer), category, item, Qty, price, datetime (Waktu Transaksi), dan payment method (Cash/Debit). Setiap transaksi dapat dipastikan berasal dari satu Order, namun Order bisa saja tidak dilakukan proses *checkout* atau transaksi [**One to Zero or One**].

- **Tabel Seller** yang merupakan database akun Seller. Berisikan id_seller (Primary Key), nama_seller, dan noTelp_seller. Sudah dapat dipastikan Transaksi yang di Approve/Disapprove terlibat satu akun Seller, namun Transaksi bisa saja tidak dilakukan Approve/Disapprove karena tidak dilakukan *request cancel* oleh Customer [**One to Zero or Many**].

2.2.2 Diagram Aktivitas

Pada Diagram Aktivitas berikut merupakan gambaran alur proses bisnis pada Aplikasi Laundry-In.

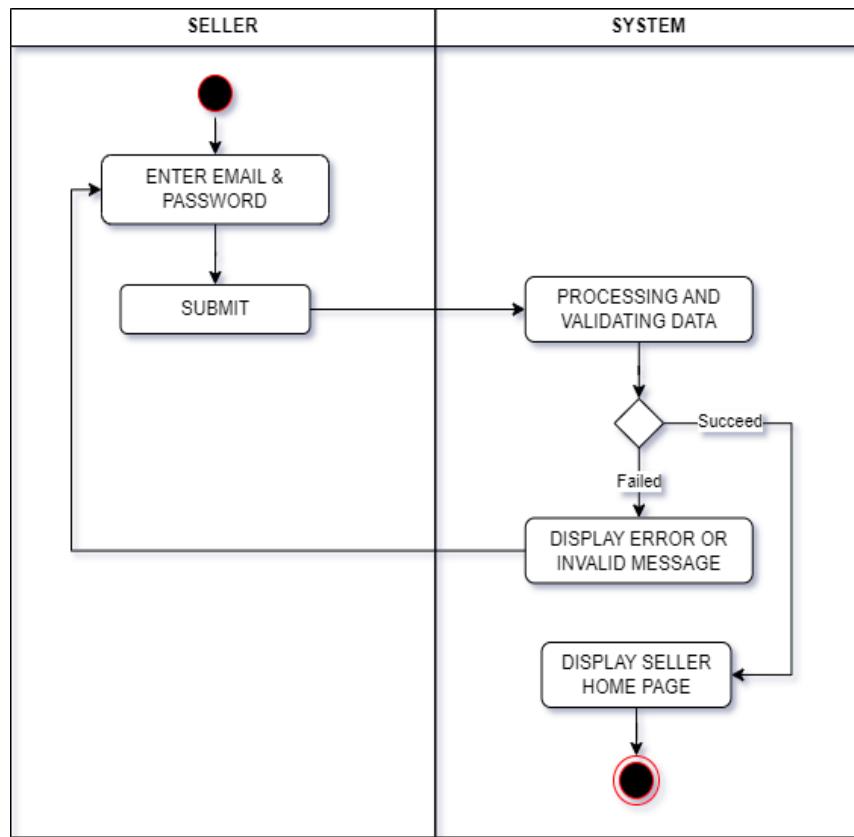
a) *Customer Sign-Up and Log-In*



Gambar 2.2.1.2 Diagram Aktivitas *Customer Sign-Up* dan *Log-in*

Pada gambar 2.2.1.2 terlihat bahwa *customer* merepresentasikan *customer* (user non-admin) aplikasi Laundry-In dan System merupakan sistem aplikasi Laundry-In. Diagram diatas menjelaskan *flow* aktivitas *customer* mulai dari kegiatan **Sign-Up** terlebih dahulu dengan mengisi identitas pribadi, lalu System akan mengambil data identitas ke dalam Database. Apabila berhasil maka *customer* diarahkan ke **Login** page untuk melakukan Login dengan mengisi email dan password yang telah terdaftar. System akan memproses dan memvalidasi data yang diinput. Apabila berhasil, System akan mengarahkan *customer* ke **Home Page**, yang menjadi halaman utama aplikasi dalam segala proses bisnis. Namun, jika gagal *customer* diminta untuk mengisi ulang email dan password sesuai yang terdaftar.

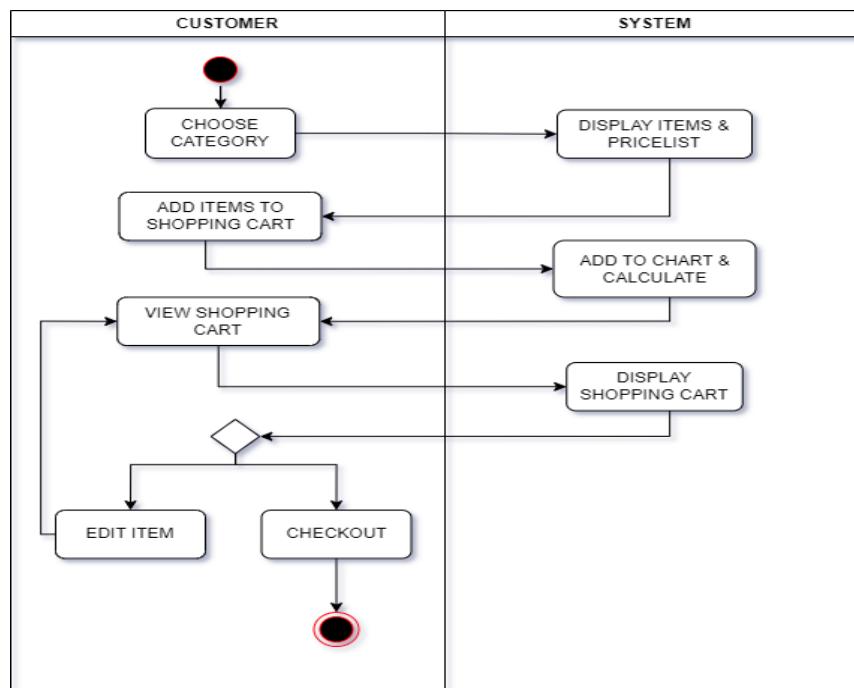
b) *Seller Sign-Up and Log-In*



Gambar 2.2.1.3 Diagram Aktivitas Seller Sign-Up dan Log-in

Pada gambar 2.2.1.3 terdapat Seller yang merupakan *seller* (user admin) aplikasi Laundry-In. *Seller* cukup melakukan Login untuk masuk ke dalam **Seller Home Page**. Dengan memasukkan username dan password yang terdaftar, System akan melakukan proses validasi data, apabila sukses maka *seller* diarahkan ke Seller Home Page. Namun, apabila gagal *seller* diminta untuk mengisi ulang username dan password sesuai yang terdaftar.

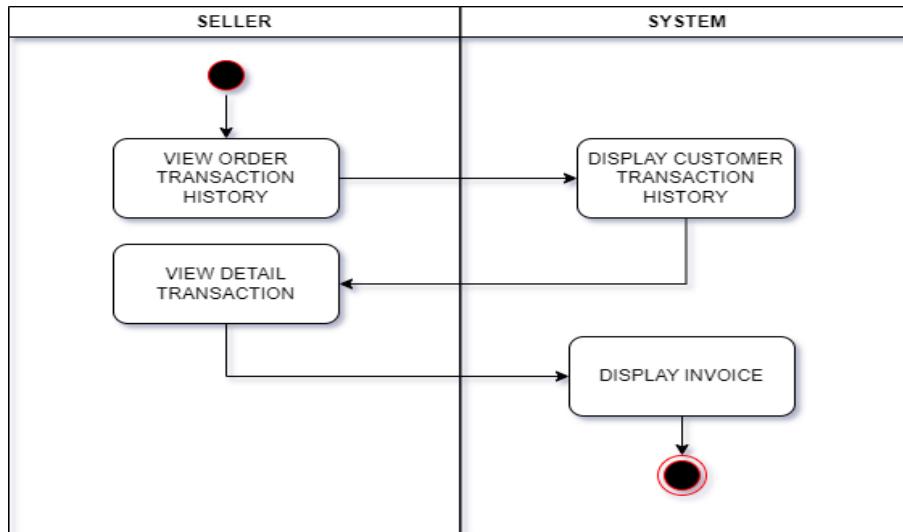
c) *Checkout Process*



Gambar 2.2.1.4 Diagram Aktivitas *Checkout Process*

Pada gambar 2.2.1.4 *Customer* dapat melakukan Order dengan memilih kategori barang terlebih dahulu, kemudian System akan menampilkan Items and Price List yang tersedia. *Customer* akan menambah pilihan item untuk di-laundry, lalu System akan memasukkannya ke dalam shopping cart. *Customer* lalu melihat isi shopping cart untuk melakukan **Edit/Update** items dan/atau langsung melakukan **Checkout** transaksi.

d) ***Order Transaction History***

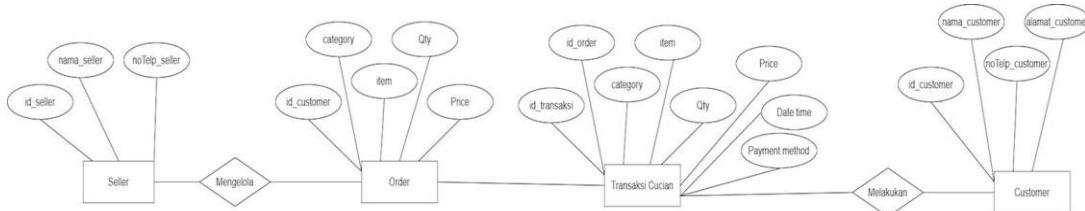


Gambar 2.2.1.6 Diagram Aktivitas Transaksi Order

Seperti yang terlihat pada gambar 2.2.1.6, *Seller* dapat melihat rekap riwayat seluruh transaksi customer. System akan menampilkan tiap transaksi customer yang apabila seller melihat detailnya, Sistem akan menampilkan *invoice*.

2.2.3 Entity Relationship Diagram

Berikut merupakan Entity Relationship Diagram yang disesuaikan oleh database yang diperlukan untuk Aplikasi Laundry-In.



Gambar 2.2.3.1 Entity Relationship Diagram

BAB III

SET-UP APLIKASI

3.1 Set-Up Aplikasi

3.1.1 Android Manifest

a) Set-up Permission

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Gambar 3.1.1.1 Set-Up Permission

Kode pada gambar 3.1.1.1, menentukan untuk izin penting untuk aplikasi Android yang di rancang, seperti akses internet, lokasi perangkat, dan status koneksi jaringan. Izin ini diperlukan agar aplikasi dapat berfungsi dengan baik sesuai dengan persyaratan fungsionalnya dan ketergantungan akses yang diperlukan.

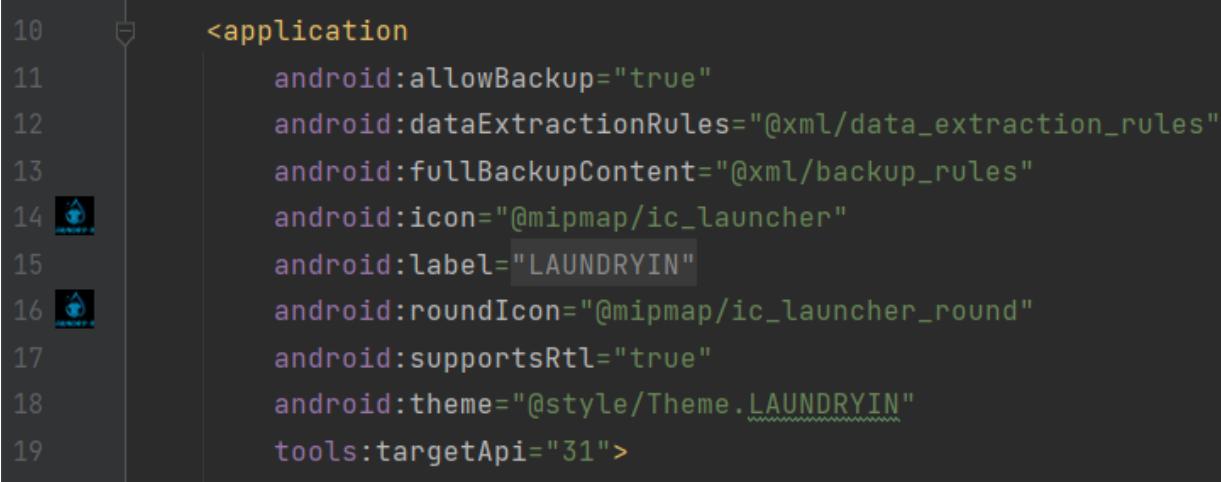
b) API Key

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyBq8Nr_gLDvm14-Usojs0hfJvYo7lkqog8" />
```

Gambar 3.1.1.2 API Key

Kode diatas ini merupakan kode untuk menyediakan kunci API Google untuk layanan geografis dalam konfigurasi aplikasi android yang memungkinkan aplikasi untuk mengakses layanan peta atau lokasi dari Google.

c) Laundry-In Set-Up

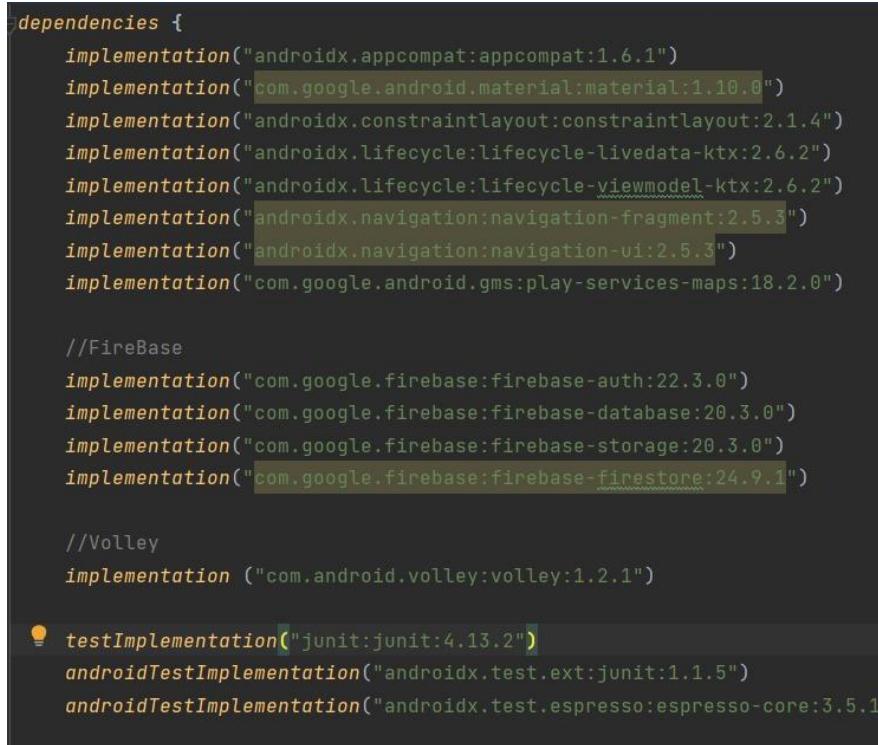


```
10     <application
11         android:allowBackup="true"
12         android:dataExtractionRules="@xml/data_extraction_rules"
13         android:fullBackupContent="@xml/backup_rules"
14         android:icon="@mipmap/ic_launcher"
15         android:label="LAUNDRYIN"
16         android:roundIcon="@mipmap/ic_launcher_round"
17         android:supportsRtl="true"
18         android:theme="@style/Theme.LAUNDRYIN"
19         tools:targetApi="31">
```

Gambar 3.1.1.2 API Key

Kode diatas digunakan untuk mengonfigurasi aplikasi android dengan memperbolehkan pencadangan data (`allowBackup="true"`), menentukan aturan ekstraksi, konten pencadangan, mengatur ikon dan label aplikasi ("LAUNDRYIN"), mendukung tata letak dari kanan ke kiri, dan menetapkan tema aplikasi.

3.1.2 GradleBuild



```
dependencies {
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("androidx.lifecycle:lifecycle-livedata-ktx:2.6.2")
    implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.2")
    implementation("androidx.navigation:navigation-fragment:2.5.3")
    implementation("androidx.navigation:navigation-ui:2.5.3")
    implementation("com.google.android.gms:play-services-maps:18.2.0")

    //Firebase
    implementation("com.google.firebaseio:firebase-auth:22.3.0")
    implementation("com.google.firebaseio:firebase-database:20.3.0")
    implementation("com.google.firebaseio:firebase-storage:20.3.0")
    implementation("com.google.firebaseio:firebase-firebase:24.9.1")

    //Volley
    implementation ("com.android.volley:volley:1.2.1")

    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
```

Gambar 3.1.2.1 GradleBuild

3.2 Implementasi Basis Data

3.2.1 Penggunaan SQLite

Pada aplikasi Laundry-In, penggunaan SQLite yaitu ketika seller membuat akun, data seller seperti Username dan Password akan tersimpan pada database SQLite.

a) Tampilan pada SQLite untuk Database Admin/Seller

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - F:\adminLaundry.db". The menu bar includes File, Edit, View, Tools, and Help. Below the menu is a toolbar with New Database, Open Database, Write, Database Structure, Browse Data, and Edit Pragmas. A dropdown menu "Table:" is set to "user". The main area displays a table with three columns: id, username, and password. There are two rows of data:

1	1	admin	laundry...
2	2	adminlaundry	laundryin

Gambar 3.2.1.1 Tampilan pada SQLite

b) Kode DBHelper untuk Admin

```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) { super(context, "adminLaundry.db", null, 1); }

    @Override
    public void onCreate(SQLiteDatabase myDB) {
        // Buat tabel untuk menyimpan data sesi
        myDB.execSQL("CREATE TABLE session(id INTEGER PRIMARY KEY, login TEXT NOT NULL);

        // Buat tabel untuk menyimpan data pengguna
        myDB.execSQL("CREATE TABLE user(id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT NOT NULL, password TEXT NOT NULL");

        // Masukkan data sesi awal
        myDB.execSQL("INSERT INTO session(id, login) VALUES(1, 'kosong')");
    }
}
```

```

@Override
public void onUpgrade(SQLiteDatabase myDB, int oldVersion, int newVersion) {
    // Hapus tabel sesi dan tabel pengguna jika ada
    myDB.execSQL("DROP TABLE IF EXISTS session");
    myDB.execSQL("DROP TABLE IF EXISTS user");

    // Buat kembali tabel sesi dan tabel pengguna
    onCreate(myDB);
}

// Periksa apakah sesi masih aktif
public Boolean checkSession(String sessionValues) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM session WHERE login = ?", new String[]{sessionValues});

    // Periksa apakah ada data sesi yang ditemukan
    if (cursor.getCount() > 0) {
        return true;
    } else {
        return false;
    }
}

// Perbarui sesi
public Boolean upgradeSession(String sessionValues, int id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("login", sessionValues);

    // Perbarui data sesi
    long update = myDB.update("session", contentValues, "id=" + id, null);

    // Periksa apakah data sesi berhasil diperbarui
    if (update == -1) {
        return false;
    } else {
        return true;
    }
}

//insert user
public Boolean insertUser(String username, String password) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("username", username);
    contentValues.put("password", password);
    long insert = myDB.insert("user", null, contentValues);
    if (insert == -1) {
        return false;
    } else {
        return true;
    }
}

```

```
// Periksa apakah pengguna dapat masuk
public Boolean checkLogin(String username, String password) {
    SQLiteDatabase myDB = this.getReadableDatabase();
    Cursor cursor = myDB.rawQuery("SELECT * FROM user WHERE username = ? AND password = ?", new String[]{username, password});

    // Periksa apakah ada data pengguna yang ditemukan
    if (cursor.moveToFirst()) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 3.2.1.2 Kode DBHelper untuk Admin

c) Kode Sign-Up Admin

INPUT:

```
public class SSignup extends AppCompatActivity {

    DBHelper myDB;
    Button signup;
    EditText username, password, repassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ssignup);

        myDB = new DBHelper( context: this);

        username = (EditText) findViewById(R.id.edtUsername);
        password = (EditText) findViewById(R.id.edtPass);
        repassword = (EditText) findViewById(R.id.edtConf);
        signup = (Button)findViewById(R.id.btnSigUpSeller);

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String strUsername = username.getText().toString();
                String strPassword = password.getText().toString();
                String strPasswordConf = repassword.getText().toString();
            }
        });
    }
}
```

```

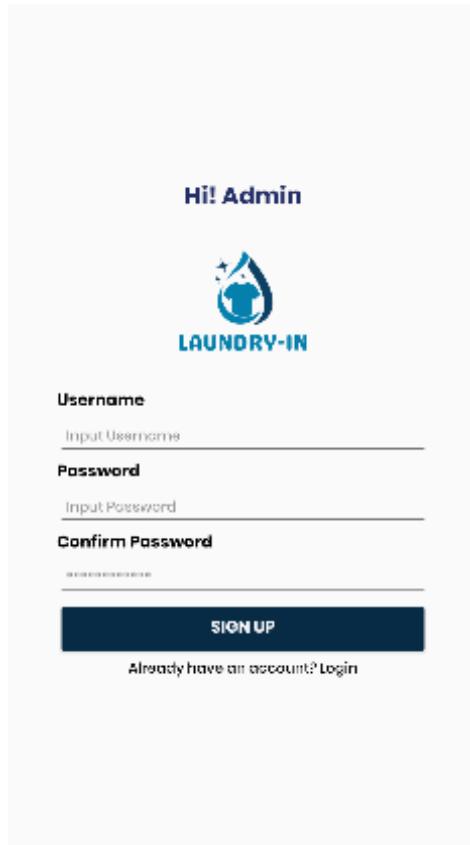
signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String strUsername = username.getText().toString();
        String strPassword = password.getText().toString();
        String strPasswordConf = repassword.getText().toString();

        // Periksa apakah password dan password konfirmasi cocok
        if (strPassword.equals(strPasswordConf)) {
            // Daftarkan pengguna baru
            Boolean daftar = myDB.insertUser(strUsername, strPassword);
            if (daftar == true) {
                Toast.makeText(getApplicationContext(), text: "Daftar Berhasil", Toast.LENGTH_SHORT).show();
                Intent loginIntent = new Intent(packageContext: SSignup.this, SLogin.class);
                startActivity(loginIntent);
                finish();
            } else {
                Toast.makeText(getApplicationContext(), text: "Daftar Gagal", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(), text: "Password Tidak Cocok", Toast.LENGTH_SHORT).show();
        }
    }
});

```

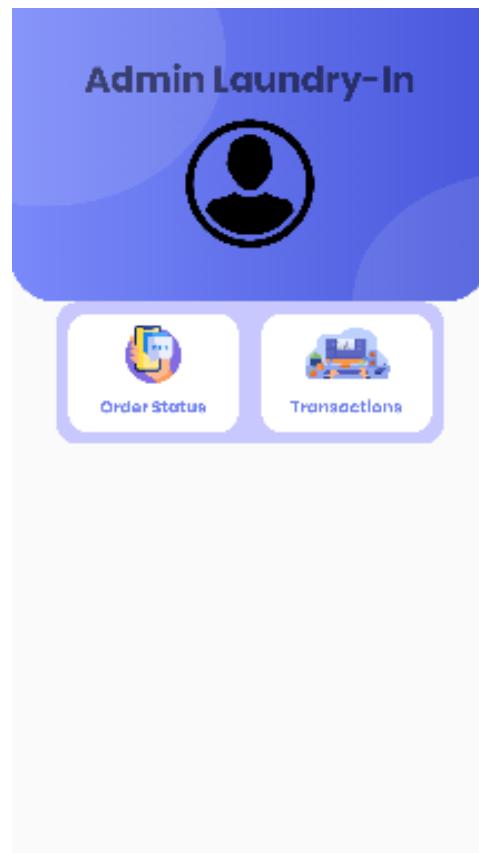
Gambar 3.2.1.3 Kode Sign-Up Admin

OUPUT:



Gambar 3.2.1.4 Sign-Up Page Admin

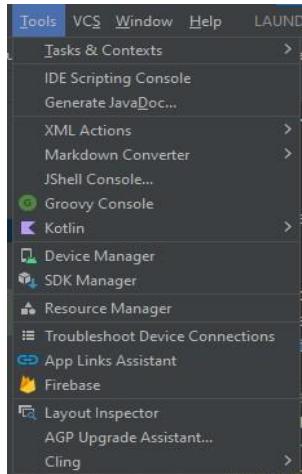
d) Home Page Admin



Gambar 3.2.1.5 Home Page Admin

3.3 Penggunaan Firebase - Database

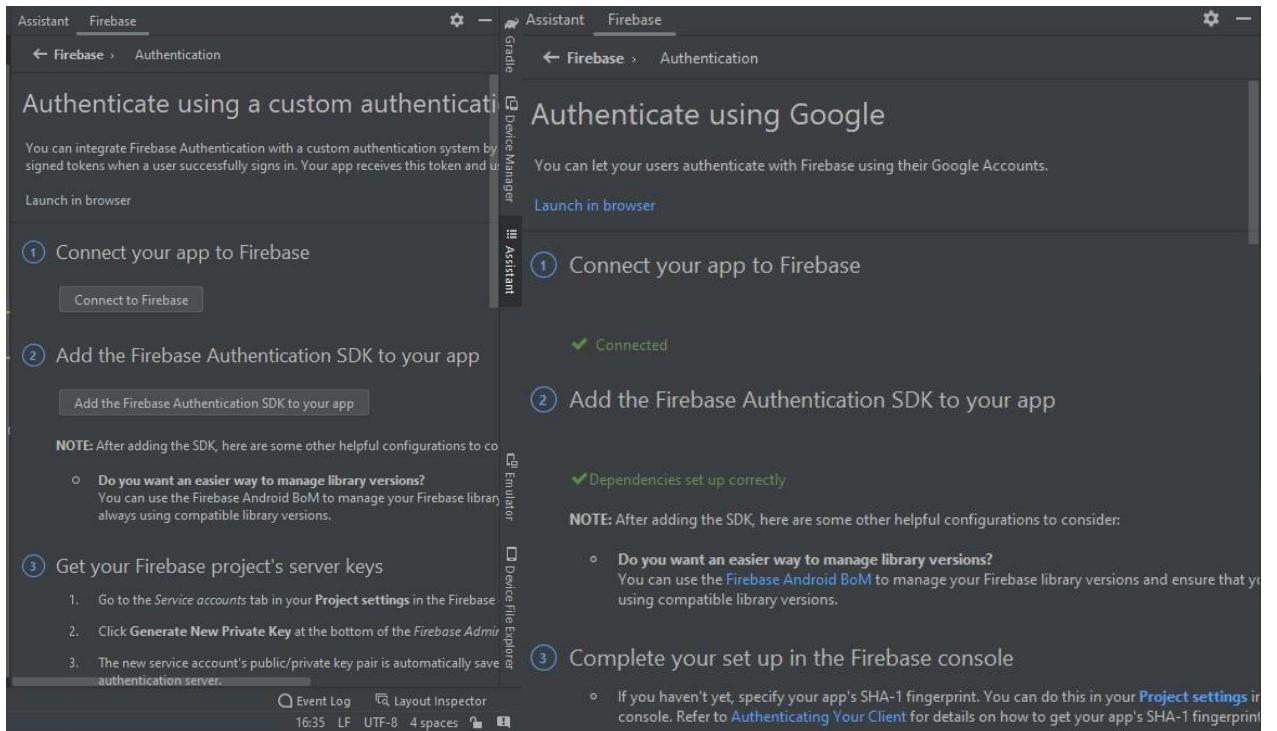
3.3.1 Firebase Connection



Gambar 3.3.1.1 Firebase Connection

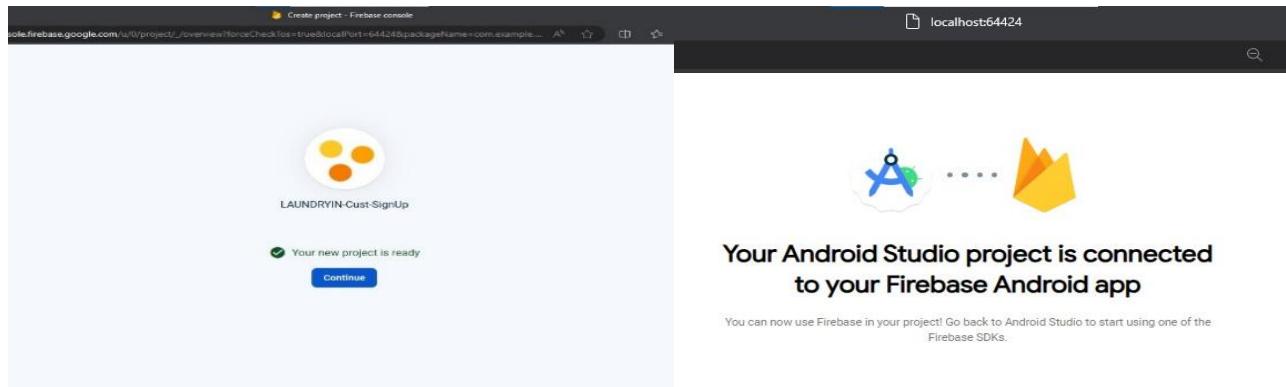
Aplikasi Laundryin akan menggunakan Firebase, sebuah platform pengembangan aplikasi, sebagai basis data. Firebase menyediakan fitur seperti otentikasi pengguna, penyimpanan cloud, dan basis data real-time, memungkinkan aplikasi untuk menyimpan dan menyinkronkan data pengguna di cloud secara real-time. Selain itu, Firebase juga menawarkan fitur keamanan yang kuat untuk melindungi data pengguna. Dengan demikian, aplikasi Laundryin dapat memberikan pengalaman yang lebih baik dan lebih aman bagi penggunanya.

Before & After connect:



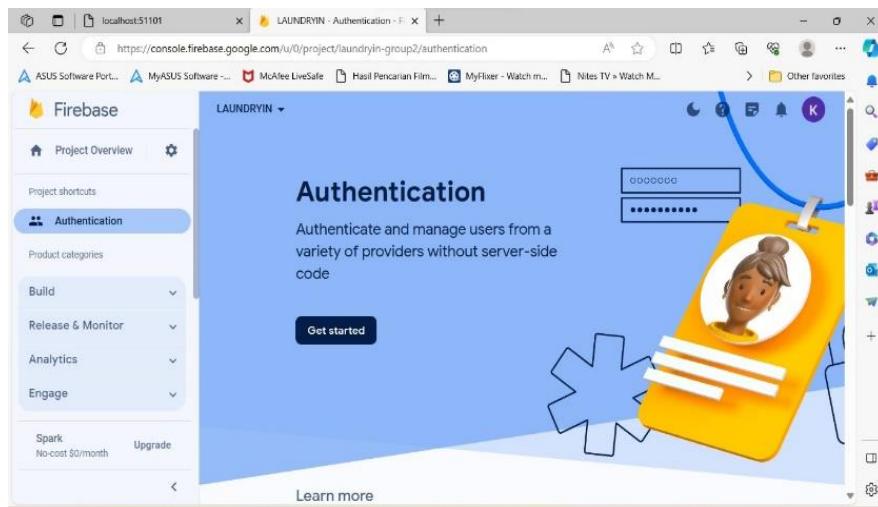
Gambar 3.3.1.2 Firebase Connection

a) Buat Project baru dalam firebase untuk aplikasi Laundryin



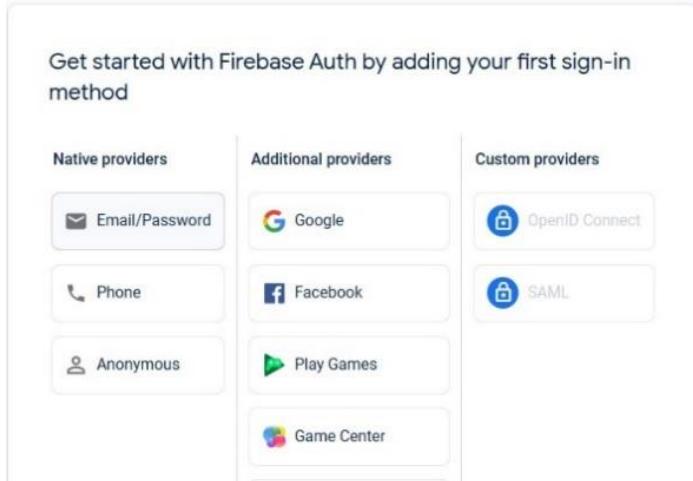
Gambar 3.3.1.3 Firebase Connection Process

b) Start Authentication untuk Database Akun User saat Register

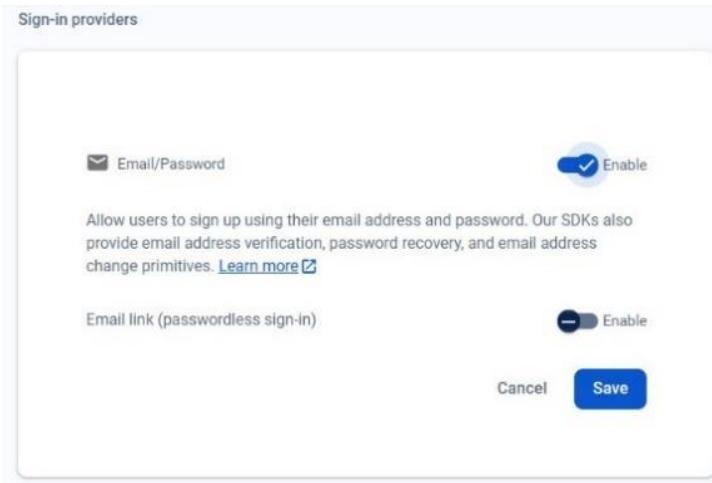


Gambar 3.3.1.4 Firebase Connection Process

c) Set-Up Authentication



Gambar 3.3.1.5 Pilih Email/Password



Gambar 3.3.1.6 Enable Akses

Sign-in providers	
Provider	Status
Email/Password	Enabled

Gambar 3.3.1.7 Akses telah Enable

Firebase Authentication memberikan solusi cepat, aman, dan mudah diintegrasikan untuk manajemen keamanan aplikasi Anda. Autentikasi Firebase:

1. Integrasi Mudah

Terintegrasi dengan Firebase dan layanan Cloud, meningkatkan efisiensi manajemen keamanan.

2. Metode Login Beragam

Mendukung login melalui email, Google, Facebook, Twitter, dll., memberikan fleksibilitas kepada pengguna.

3. Keamanan Tinggi

Otentikasi dua faktor, verifikasi email, dan opsi keamanan tambahan melindungi data pengguna.

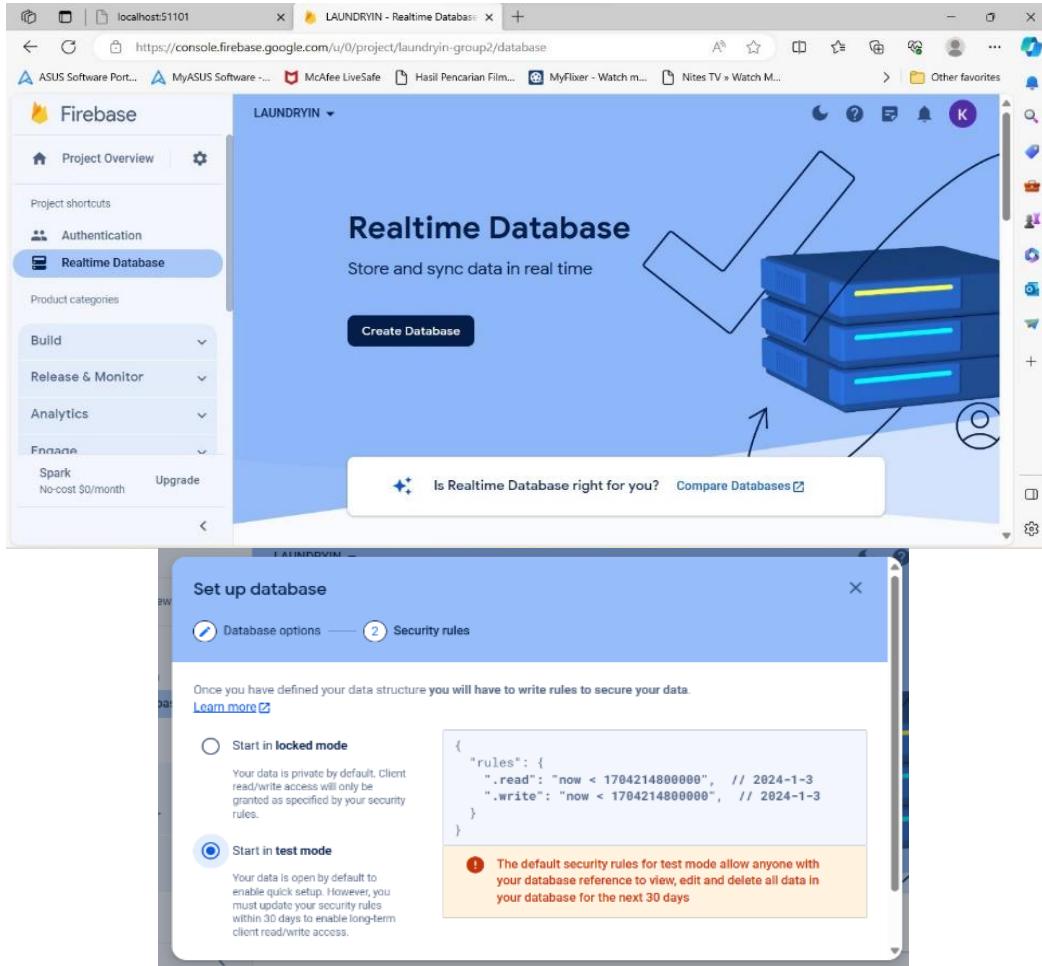
4. Manajemen Sesi Mudah

Menyederhanakan pengelolaan status login/logout.

5. Skalabilitas Tinggi

Mendukung jutaan pengguna secara bersamaan, cocok untuk basis pengguna besar.

d) Start Realtime Database



Gambar 3.3.1.8 Realtime Database

Firebase Realtime Database adalah solusi kuat dan fleksibel untuk basis data aplikasi Laundryin. Realtime Database Firebase memiliki keunggulan utama:

1. Format JSON

Menyimpan data dalam format JSON, memudahkan manipulasi dan penggunaan data dengan efisiensi.

2. Realtime

Sinkronisasi data secara real-time, perubahan data langsung diterima oleh semua perangkat terhubung dalam hitungan milidetik.

3. Informasi Akun Customer

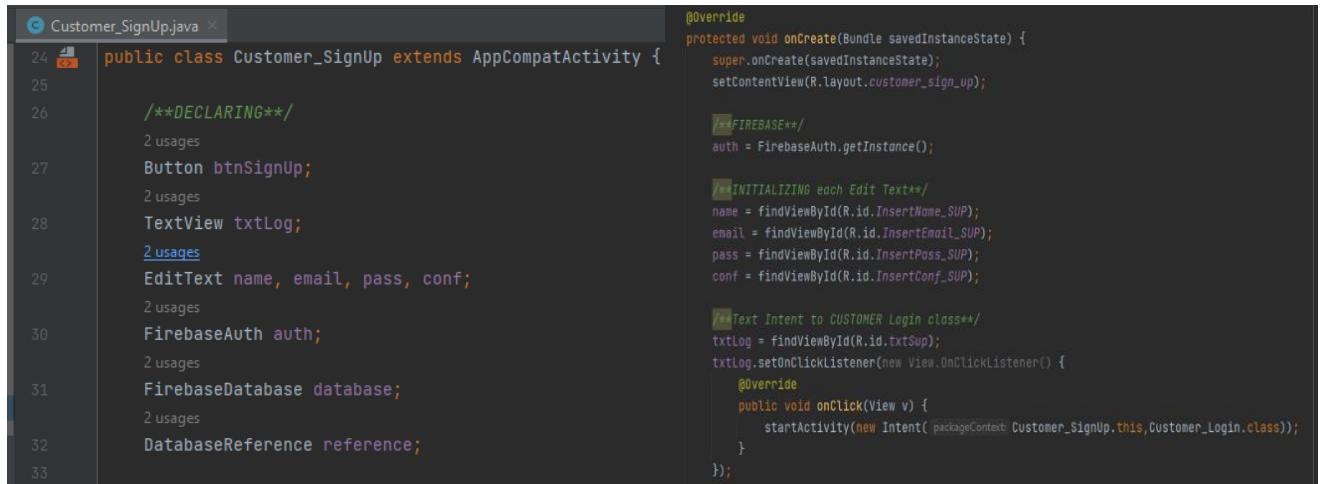
Firebase Authentication memungkinkan penyimpanan dan manajemen informasi akun pengguna untuk pengalaman yang lebih personal dan aman.

4. CRUD Profile

Mendukung operasi Create, Read, Update, dan Delete (CRUD) untuk manipulasi data secara lengkap.

3.3.2 Sign-Up Customer

INPUT:



```
Customer_SignUp.java
public class Customer_SignUp extends AppCompatActivity {
    Button btnSignUp;
    TextView txtLog;
    EditText name, email, pass, conf;
    FirebaseAuth auth;
    FirebaseDatabase database;
    DatabaseReference reference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.customer_sign_up);
        auth = FirebaseAuth.getInstance();
        name = findViewById(R.id.InsertName_SUP);
        email = findViewById(R.id.InsertEmail_SUP);
        pass = findViewById(R.id.InsertPass_SUP);
        conf = findViewById(R.id.InsertConf_SUP);
        txtLog = findViewById(R.id.txtSup);
        txtLog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), Customer_Login.class));
            }
        });
    }
}
```

Gambar 3.3.2.1 Sign-Up Customer

Pada potongan kode ini, terdapat deklarasi dan inisialisasi elemen-elemen antarmuka pengguna dan objek Firebase yang diperlukan untuk aktivitas pendaftaran pelanggan (`Customer_SignUp`). Deklarasi mencakup tombol, teks tampilan, dan kolom teks untuk menerima input pengguna seperti nama, email, kata sandi, dan konfirmasi kata sandi. Objek Firebase seperti `FirebaseAuth` untuk otentikasi dan `FirebaseDatabase` beserta `DatabaseReference` untuk berinteraksi dengan database diinisialisasi. Selain itu, elemen antarmuka pengguna dihubungkan dengan kode Java menggunakan `findViewById`, dan layout aktivitas diatur menggunakan `setContentView`. Keseluruhan kode membentuk dasar dari implementasi pendaftaran pelanggan dalam aplikasi laundry.

Dalam potongan kode ini, teks tampilan (`txtLog`) diinisialisasi untuk mengakses dan memanipulasi propertiannya. Ketika teks tersebut ditekan, sebuah `Intent` digunakan untuk memulai aktivitas masuk pelanggan (`Customer_Login`). Ini memberikan pengalaman pengguna yang lancar, memungkinkan mereka dengan mudah beralih dari layar pendaftaran pelanggan ke layar masuk pelanggan hanya dengan mengklik teks tampilan yang sesuai.

```
/**BUTTON SIGN UP*/
btnSignUp = findViewById(R.id.btnSignUpCustomer);
btnSignUp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { createUser(); }
});
}
```

Gambar 3.3.2.2 Button Sign-Up

Pada gambar 3.3.1.10, tombol pendaftaran (`btnSignUpCustomer`) diinisialisasi untuk merespons klik pengguna. Ketika tombol tersebut ditekan, metode `createUser()` akan dipanggil untuk memproses logika pendaftaran pengguna baru. Dengan penanganan klik tombol ini, pengguna dapat dengan mudah memulai proses pendaftaran hanya dengan satu kali klik.

Pendeklarasian dan penanganan klik tombol dilakukan dengan menghubungkan tombol "Sign Up" dengan objek `btnSignUp` dan menetapkannya dengan penanganan klik menggunakan `setOnClickListener`. Saat tombol ditekan, metode `createUser()` dipanggil untuk menginisiasi proses pendaftaran.

```
/**Create User Method*/
1 usage
private void createUser() {

    /**Parse**/
    String userName = name.getText().toString();
    String userEmail = email.getText().toString();
    String userPass = pass.getText().toString();
    String userConf = conf.getText().toString();
}
```

Gambar 3.3.2.3 Metode "createUser"

Dalam langkah pertama seperti gambar 3.3.1.11, data dari kolom teks yang mencakup nama (`userName`), email (`userEmail`), kata sandi (`userPass`), dan konfirmasi kata sandi (`userConf`) diambil menggunakan metode `getText().toString()`.

```

/**Alert**/
if (TextUtils.isEmpty(userName)) {
    Toast.makeText(context: this, text: "Your Name is Empty!", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userEmail)) {
    Toast.makeText(context: this, text: "Your Email is Empty!", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userPass)) {
    Toast.makeText(context: this, text: "Your Password is Empty!", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userConf)) {
    Toast.makeText(context: this, text: "Please confirm your Password!", Toast.LENGTH_SHORT).show();
    return;
}

```

Gambar 3.3.2.4 Toast Method

Selanjutnya, dalam langkah kedua seperti gambar 3.3.1.12, dilakukan serangkaian pemeriksaan untuk memvalidasi bahwa semua kolom teks telah diisi dengan benar. Jika terdapat kolom yang masih kosong, pengguna akan diberikan pesan peringatan melalui objek `Toast`, dan eksekusi metode akan dihentikan dengan pernyataan `return`. Langkah ini bertujuan untuk memastikan bahwa data yang diberikan oleh pengguna lengkap dan sesuai sebelum melanjutkan proses pendaftaran.

```

/**Create User connect to Firebase*/
auth.createUserWithEmailAndPassword(userEmail, userPass)
    .addOnCompleteListener(activity: this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // User creation successful
                String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
                database = FirebaseDatabase.getInstance();
                reference = database.getReference(path: "Users").child(userId);

                /**For Realtime Customer Information*/
                CustomerModel customerModel = new CustomerModel(userName, userEmail, userPass);
                reference.setValue(customerModel);

                /**Successful Alert*/
                Toast.makeText(context: Customer_SignUp.this, text: "Registration Successful!",
                    Toast.LENGTH_SHORT).show();
            } else {
                // If sign in fails, display a message to the user.
                Toast.makeText(context: Customer_SignUp.this, text: "Error: " + task.getException(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    });

```

Gambar 3.3.2.5 Toast Registration Successful

1. Pendaftaran Pengguna di Firebase Authentication

Menggunakan `auth.createUserWithEmailAndPassword`, pengguna baru dicoba didaftarkan dengan email dan kata sandi yang diberikan. Proses ini dilakukan secara asinkron, dan metode `addOnCompleteListener` digunakan untuk menangani hasilnya.

2. Penanganan Hasil Pendaftaran

Jika pendaftaran berhasil (`task.isSuccessful()`), ID pengguna (`userId`) diperoleh dari Firebase Authentication, dan objek `DatabaseReference` dibuat untuk menyimpan informasi pelanggan di Firebase Realtime Database.

3. Penyimpanan Informasi Pelanggan di Firebase Realtime Database

Objek `CustomerModel` dibuat dengan menggunakan data nama, email, dan kata sandi yang diambil sebelumnya. Kemudian, informasi ini disimpan di Firebase Realtime Database dengan memanggil `setValue(customerModel)` pada `reference`.

4. Pesan Toast Sukses

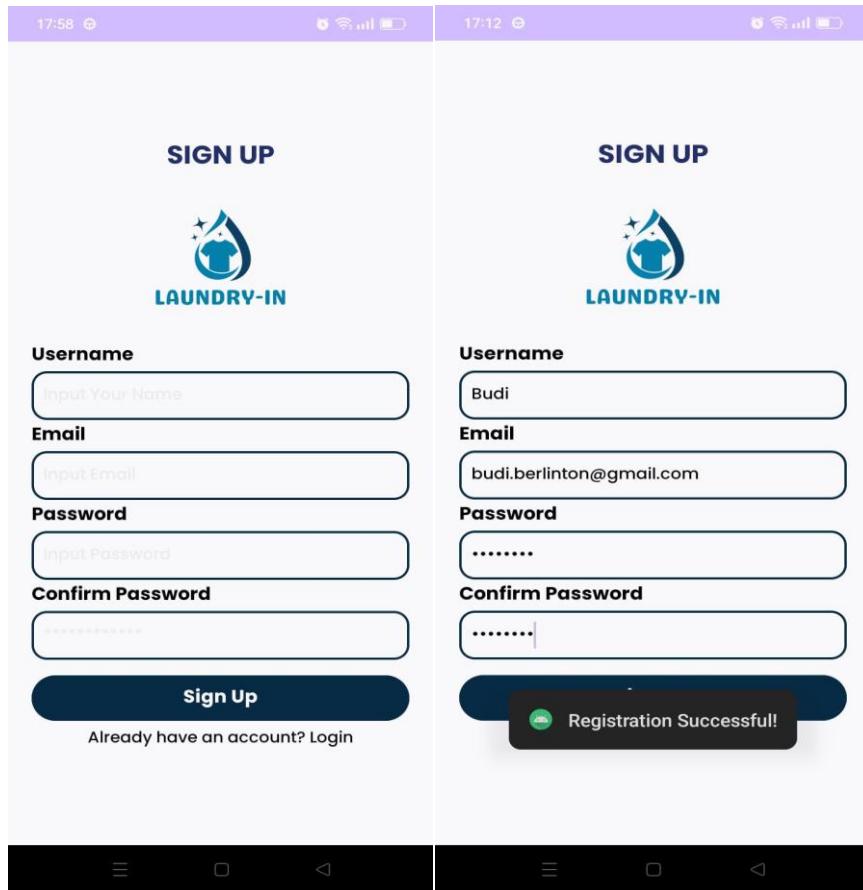
Jika seluruh proses berjalan lancar, pesan toast ditampilkan memberi tahu pengguna bahwa pendaftaran berhasil.

5. Penanganan Kesalahan

Jika terjadi kesalahan selama pendaftaran (misalnya, email sudah digunakan), pesan kesalahan ditampilkan kepada pengguna melalui `Toast`.

Dengan demikian, metode `createUser()` mengelola seluruh proses pendaftaran pengguna baru, mulai dari validasi input hingga penyimpanan informasi pelanggan di Firebase Authentication dan Realtime Database.

OUTPUT:



Gambar 3.3.2.6 Output Toast Method

a) Authentication

Identifier	Providers	Created	Signed In	User UID
budi.berlinton@gmail.com	✉	Dec 5, 2023	Dec 5, 2023	3JPSeHHND9ShpfGwQQUIWbRBk...
bebekkaleyo@gmail.com	✉	Dec 5, 2023	Dec 5, 2023	jjPN7E15aCNGGxxfsu7LgkNe5on1
josejose@gmail.com	✉	Dec 4, 2023	Dec 4, 2023	1POBJ1jOCsffjjJeEqJ0wD5GA33
lolipop@gmail.com	✉	Dec 4, 2023	Dec 4, 2023	y8hkjmgUIGMz9RlxNcnTrE99a0h2
chikk@gmail.com	✉	Dec 4, 2023	Dec 5, 2023	WRL8qm795CebehFFLOJ7K1u1...
kira@gmail.com	✉	Dec 4, 2023	Dec 4, 2023	CnBZrUwblnTpRlvKr7Zv9mBiQ9H3
chiko@gmail.com	✉	Dec 4, 2023	Dec 4, 2023	77W74Finv1Ysv0kgmoZApzgzxK12...
jakihaki@gmail.com	✉	Dec 4, 2023	Dec 4, 2023	yJRRqOC8LMSA0sqtg9U8NJNztB...

Gambar 3.3.2.7 Authentication

b) Realtime Database

The screenshot shows the Firebase Realtime Database interface for a project named 'LAUNDRYIN'. The left sidebar has 'Realtime Database' selected. The main area shows a tree view of database nodes. One node under '3JPSeHHND9ShpfGwQQUlWbRBkL33' is highlighted with a red box. This node contains three user profiles:

- 1POBj1j0CsffjjJeQEqJ0wD5GA33: email: "josejose@gmail.com", name: "Jose Maukar", password: "josejose"
- 3JPSeHHND9ShpfGwQQUlWbRBkL33 (highlighted): email: "budi.berlinton@gmail.com", name: "Budi", password: "budi1234"
- WRL8qm795CebhkehFFL0J7K1u102: email: "chikk@gmail.com"

At the bottom, it says 'Database location: United States (us-central1)'.

Gambar 3.3.2.8 Realtime Database

3.4 Log-In

INPUT:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.customer_login);

    /**FIREBASE*/
    auth = FirebaseAuth.getInstance();

    /**INITIALIZING each Edit Text*/
    email = findViewById(R.id.InsertEmail_LOGIN);
    password = findViewById(R.id.InsertPass_LOGIN);

    /**BUTTON LOGIN*/
    btnLogin = findViewById(R.id.btnLogCustomer);    /**INITIALIZING*/
    btnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            loginUser();
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
        }
    });
}

```

Gambar 3.4.1. Kode Log-In

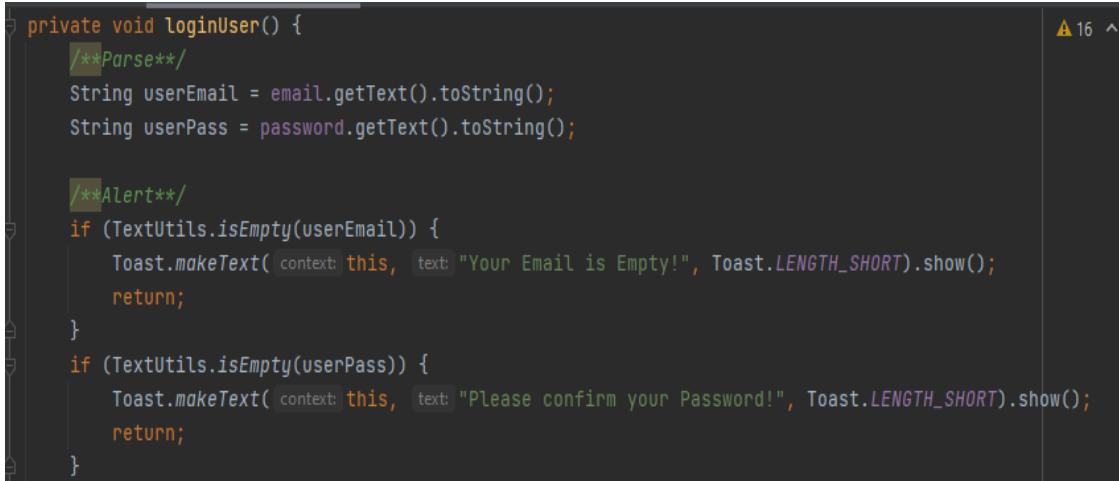
Kode ini digunakan untuk inisialisasi aktivitas login dalam aplikasi. Ini mencakup pengaturan tampilan user interface, inisialisasi autentikasi Firebase, dan penanganan interaksi pengguna untuk proses login.

```
/**Text Intent to CUSTOMER SignUp class*/
txtLog = findViewById(R.id.txtLog);    /**INITIALIZING*/
txtLog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), Customer_SignUp.class);
        startActivity(intent);
    }
});

/**Text Intent to SELLER Login class*/
txtSeller = findViewById(R.id.txtLogSeller);  /**INITIALIZING*/
txtSeller.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), Seller_Login.class);
        startActivity(intent);
    }
});
```

Gambar 3.4.2. Kode Log-In

Tombol tersebut berfungsi sebagai pemicu untuk berpindah ke aktivitas lain dalam aplikasi. Saat tombol diklik, kode akan membuat sebuah Intent yang mengarah ke kelas Customer_Signup_Class atau SELLER_Login_Class, tergantung tombol mana yang diklik. Kemudian, metode startActivity(intent) dipanggil dengan Intent tersebut sebagai argumen, yang akan memulai aktivitas baru sesuai dengan kelas yang ditentukan dalam Intent. Ini memungkinkan pengguna untuk berpindah dari aktivitas saat ini ke aktivitas pendaftaran pelanggan atau login penjual.

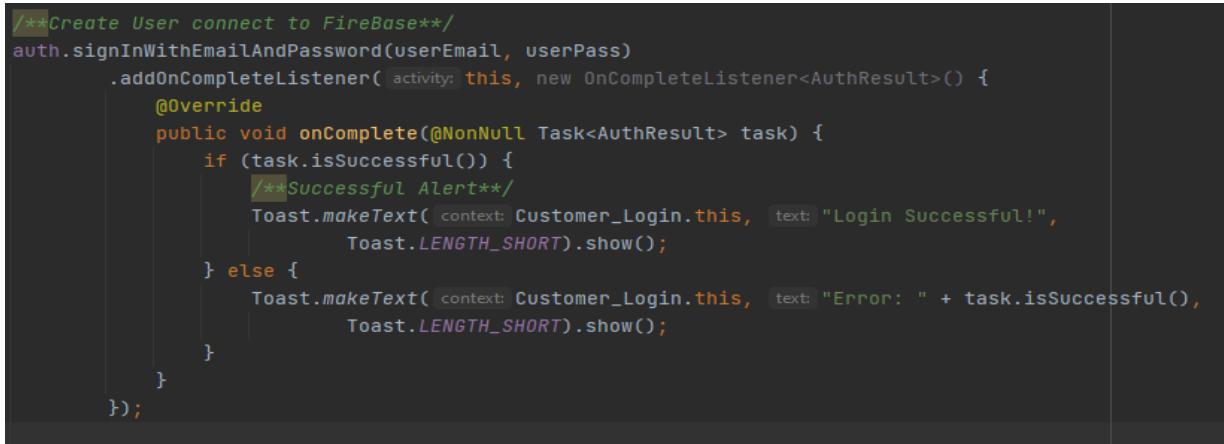


```
private void loginUser() {
    /**Parse**/
    String userEmail = email.getText().toString();
    String userPass = password.getText().toString();

    /**Alert**/
    if (TextUtils.isEmpty(userEmail)) {
        Toast.makeText(context: this, text: "Your Email is Empty!", Toast.LENGTH_SHORT).show();
        return;
    }
    if (TextUtils.isEmpty(userPass)) {
        Toast.makeText(context: this, text: "Please confirm your Password!", Toast.LENGTH_SHORT).show();
        return;
    }
}
```

Gambar 3.4.3. Kode Log-In

Kode ini berfungsi sebagai “Alert” atau peringatan dalam aplikasi. Jika bidang email atau password kosong, kode ini akan menampilkan pesan “Your Email is Empty” kepada pengguna. Ini dilakukan dengan menggunakan metode `Toast.makeText()`, yang membuat dan menampilkan pesan pop-up singkat (dikenal sebagai `Toast`) pada layar.



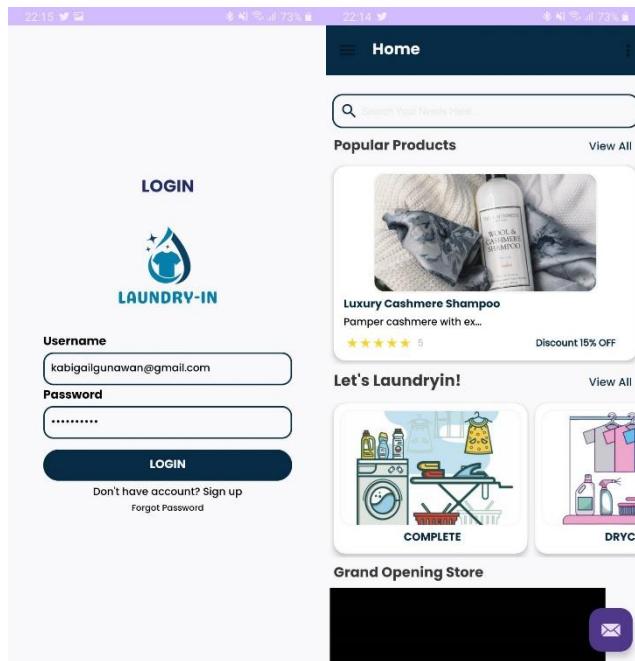
```
/**Create User connect to FireBase**/
auth.signInWithEmailAndPassword(userEmail, userPass)
    .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                /**Successful Alert**/
                Toast.makeText(context: Customer_Login.this, text: "Login Successful!",
                    Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context: Customer_Login.this, text: "Error: " + task.isSuccessful(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Gambar 3.4.4. Kode Log-In

Kode pada gambar 3.4.4 adalah fungsi login menggunakan otentikasi Firebase dalam bahasa pemrograman Java. Fungsi `onComplete` dipanggil setelah proses otentikasi selesai. Parameter `task` mengandung hasil dari proses otentikasi.

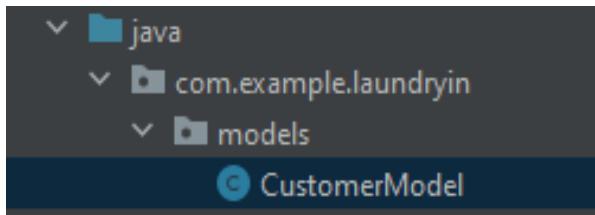
- Jika `task.isSuccessful()` mengembalikan `true`, berarti proses login berhasil. Sebuah pesan toast “Login Successful.” akan ditampilkan.
- Jika `task.isSuccessful()` mengembalikan `false`, berarti proses login gagal. Sebuah pesan toast yang berisi “Error: ” diikuti dengan hasil dari `task.isSuccessful()` akan ditampilkan. Ini akan membantu dalam debugging untuk mengetahui jenis error yang terjadi.

OUTPUT:



Gambar 3.4.5. Output Kode Log-In

3.5 Kelas CustomerModel



Gambar 3.5.1. Kelas CustomerModel

Kelas `CustomerModel` merupakan representasi model atau struktur data untuk pelanggan dalam konteks aplikasi laundry.

```
CustomerModel.java
1 package com.example.laundryin.models;
2
3 public class CustomerModel {
4     String name, email, password;
5
6     public String getName() {return name;}
7     public void setName(String name) {this.name = name;}
8     public String getEmail() {return email;}
9     public void setEmail(String email) {this.email = email;}
10    public String getPassword() {return password;}
11    public void setPassword(String password) {this.password = password;}
12
13    public CustomerModel(String name, String email, String password) {
14        this.name = name;
15        this.email = email;
16        this.password = password;
17    }
18
19    public CustomerModel() {}
20}
```

Gambar 3.5.1. Kode Kelas Customer Model

Kelas ini menggabungkan informasi tentang seorang pelanggan, termasuk nama, email, dan kata sandi. Fungsinya adalah untuk mengorganisir dan mengelola data terkait pelanggan dalam aplikasi, memberikan cara yang nyaman untuk mengakses dan memanipulasi informasi pelanggan. Kelas ini menyertakan metode **getter** dan **setter** untuk setiap atribut, memungkinkan komponen lain dari aplikasi untuk mengambil dan mengubah detail pelanggan. Konstruktor berparameter memfasilitasi pembuatan instansi kelas `CustomerModel` dengan nilai khusus untuk nama, email, dan kata sandi, sementara konstruktor default memungkinkan pembuatan instansi tanpa mengatur nilai-nilai ini secara awal. Secara keseluruhan, kelas `CustomerModel` berfungsi sebagai panduan untuk menyimpan dan mengelola data pelanggan dalam aplikasi laundry.

BAB IV

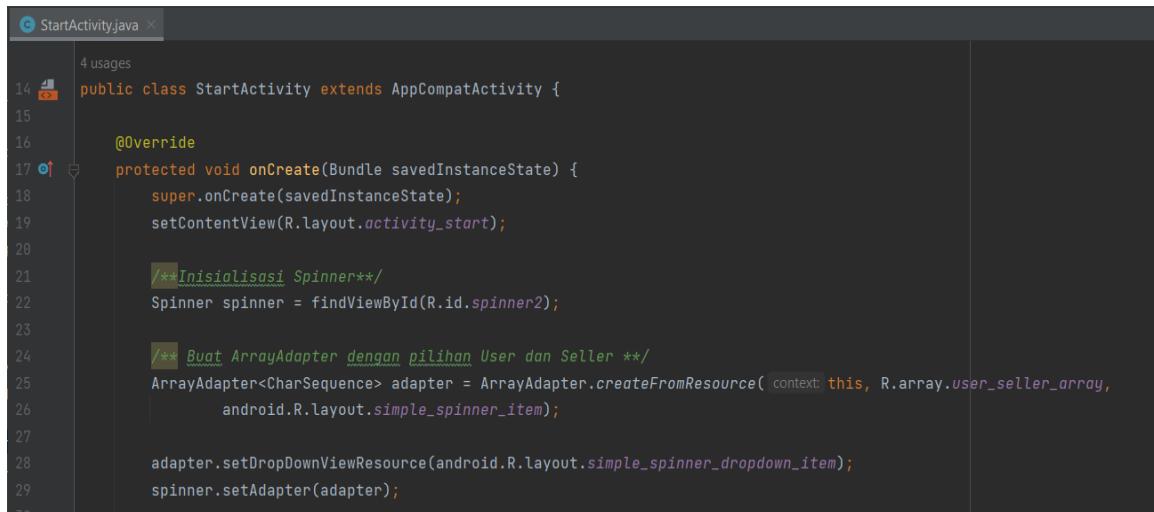
IMPLEMENTASI APLIKASI

4.1 Implementasi Aplikasi

- Download Aplikasi Laundry-In pada Google Play Store di ponsel/tablet.
- Buka Aplikasi Laundry-In: Klik button ‘Get Started’ untuk memulai aplikasi Laundry-In. Untuk pelanggan dapat memilih ‘Customer’, dan untuk kasir dapat memilih ‘Seller’.

4.1.1 Spinner Activity

1) Spinner

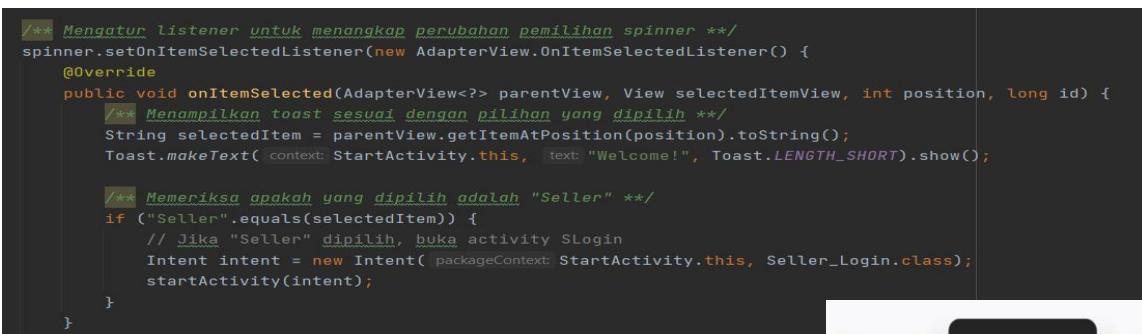


```
StartActivity.java
14 public class StartActivity extends AppCompatActivity {
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_start);
20
21         /** Inisialisasi Spinner */
22         Spinner spinner = findViewById(R.id.spinner2);
23
24         /** Buat ArrayAdapter dengan pilihan User dan Seller */
25         ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(context, R.array.user_seller_array,
26             android.R.layout.simple_spinner_item);
27
28         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
29         spinner.setAdapter(adapter);
```

Gambar 4.1.1.1 Kode Spinner

Kode di atas adalah implementasi kelas `StartActivity` dalam bahasa Java untuk aplikasi Android. Melalui metode `onCreate`, layout ditetapkan, dan spinner diinisialisasi dengan pilihan dari array sumber daya. Sebuah `ArrayAdapter` dibuat untuk menangani pilihan pengguna dan penjual pada spinner.

2) Spinner Toast dan Intent



```
StartActivity.java
14 /**
15  * Mengatur listener untuk menangkap perubahan pemilihan spinner
16 */
17 spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
18     @Override
19     public void onItemSelected(AdapterView<?> parentView, View selected.itemView, int position, long id) {
20
21         /**
22          * Menampilkan toast sesuai dengan pilihan yang dipilih
23          */
24         String selectedItem = parentView.getItemAtPosition(position).toString();
25         Toast.makeText(context, StartActivity.this, "Welcome!", Toast.LENGTH_SHORT).show();
26
27         /**
28          * Memeriksa apakah yang dipilih adalah "Seller"
29          */
30         if ("Seller".equals(selectedItem)) {
31             // Jika "Seller" dipilih, buka activity SLogin
32             Intent intent = new Intent(packageContext, Seller_Login.class);
33             startActivity(intent);
34         }
35     }
36 }
```

Gambar 4.1.1.2 Kode Spinner Toast dan Intent



Welcome!

Kode dia atas digunakan untuk menetapkan pendengar (listener) untuk menangkap perubahan pemilihan pada spinner. Ketika pengguna memilih "Seller", aktivitas `Seller_Login` dibuka melalui intent.

3) Layout (start_activity.xml)



```
/** Button Start untuk memulai */
Button btnGStart = (Button) findViewById(R.id.btnGStart);
btnGStart.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getApplicationContext(), Customer_Login.class);
        startActivity(i);
    }
});
```

Gambar 4.1.1.3 Tombol Intent untuk Log-In

Kode diatas digunakan untuk membuat tombol "Start" (`btnGStart`) pada tampilan, menetapkan pendengar klik, dan membuka aktivitas `Customer_Login` ketika tombol tersebut ditekan melalui intent.

4.1.2 Log-In

1) Declaring and Initializing

```
8 usages
public class Customer_Login extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.customer_login);

        /**FIREBASE**/
        auth = FirebaseAuth.getInstance();

        /**INITIALIZING each Edit Text*/
        email = findViewById(R.id.InsertEmail_LOGIN);
        password = findViewById(R.id.InsertPass_LOGIN);
        progressBar = findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);
    }

    /**DECLARING**/
    Button btnLogin;
    ProgressBar progressBar;
    TextView txtLog, txtSeller;
    EditText email, password;
    FirebaseAuth auth;
```

Gambar 4.1.2.1 Declaring and Initializing

Class `Customer_Login` merupakan aktivitas untuk masuk pelanggan dengan mendeklarasikan elemen UI seperti tombol, indikator kemajuan, teks, dan kolom masukan, serta objek `FirebaseAuth` untuk otentikasi. Metode `onCreate`, layout `customer_Login` diatur. Objek `FirebaseAuth` diinisialisasi. Elemen UI seperti kolom email, password, dan indikator kemajuan diinisialisasi dengan referensi ke tata letak XML. Indikator kemajuan diatur sebagai tidak terlihat.

2) Tombol Log-In



Gambar 4.1.2.2 Tombol Log-in

Pada tombol login, saat di klik maka akan memanggil method `loginUser()`, yang mana jika berhasil, maka akan berhasil login dan intent button ke `MainActivity.class`.

3) Metode Log-In User



Gambar 4.1.2.3 Metode Log-in

- Teks jika tidak memiliki akun, textview intent ke Customer Sign Up.class

```
/**Text Intent to CUSTOMER SignUp class*/
txtLog = findViewById(R.id.txtLog); /*INITIALIZING*/
txtLog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), Customer_SignUp.class);
        startActivity(intent);
    }
});
```

Don't have account? Sign up

Gambar 4.1.2.4 textview intent

- Jika lupa password

```
/**Text Intent to SELLER Login class*/
forgotPassword = findViewById(R.id.txtForgotPass); /*INITIALIZING*/
forgotPassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: Customer_Login.this);
        View dialogView = getLayoutInflater().inflate(R.layout.dialog_forgot, root: null);
        EditText emailBox = dialogView.findViewById(R.id.emailBox);
        builder.setView(dialogView);
        AlertDialog dialog = builder.create();
        /*Dialog Box to fill your real email to reset password*/
        dialogView.findViewById(R.id.btnReset).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String userEmail = emailBox.getText().toString();

                if(TextUtils.isEmpty(userEmail) && !Patterns.EMAIL_ADDRESS.matcher(userEmail).matches()){
                    Toast.makeText( context: Customer_Login.this, text: "Enter your register email id", Toast.LENGTH_SHORT).show();
                    return;
                }
                auth.sendPasswordResetEmail(userEmail).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if(task.isSuccessful()){
                            Toast.makeText( context: Customer_Login.this, text: "Check your Email!", Toast.LENGTH_SHORT).show();
                            dialog.dismiss();
                        }else {
                            Toast.makeText( context: Customer_Login.this, text: "Unable to send, failed", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
        });
    }
});
```

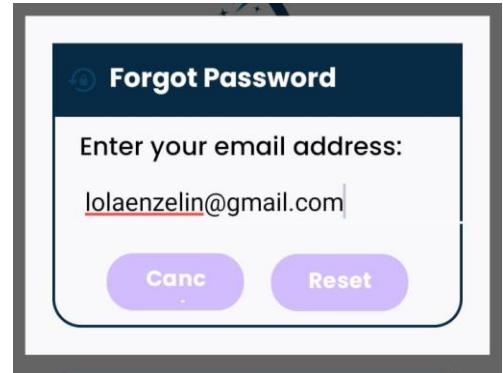
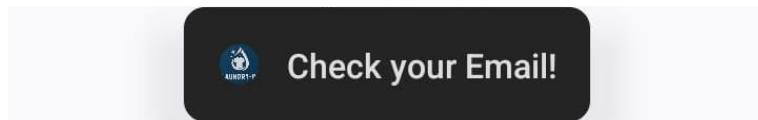
Gambar 4.1.2.5 Lupa Password

Ini akan menampilkan kotak dialog untuk memasukkan alamat email Anda, sehingga akan terkirim link ke email pribadi Customer untuk reset password.

4) Tombol Cancel pada Dialog Box

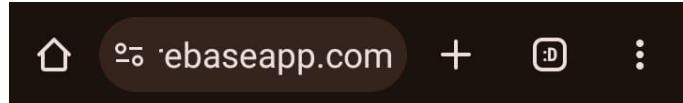
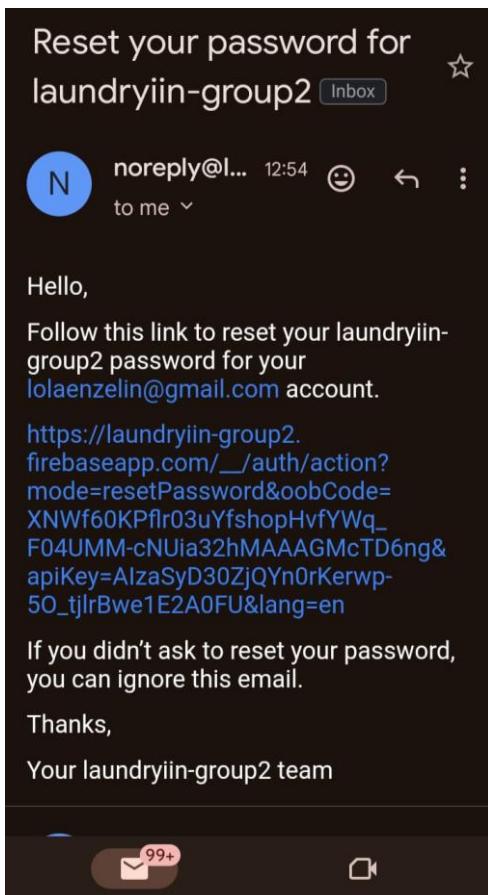
```
/*Dialog Box Cancel Button*/
dialogView.findViewById(R.id.btnCancel).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { dialog.dismiss(); }
});

if(dialog.getWindow() != null){
    dialog.getWindow().setBackgroundDrawable(new ColorDrawable(0));
}
dialog.show();
```



Gambar 4.1.2.6 Tombol cancel pada Dialog Box

- Email



Reset your password

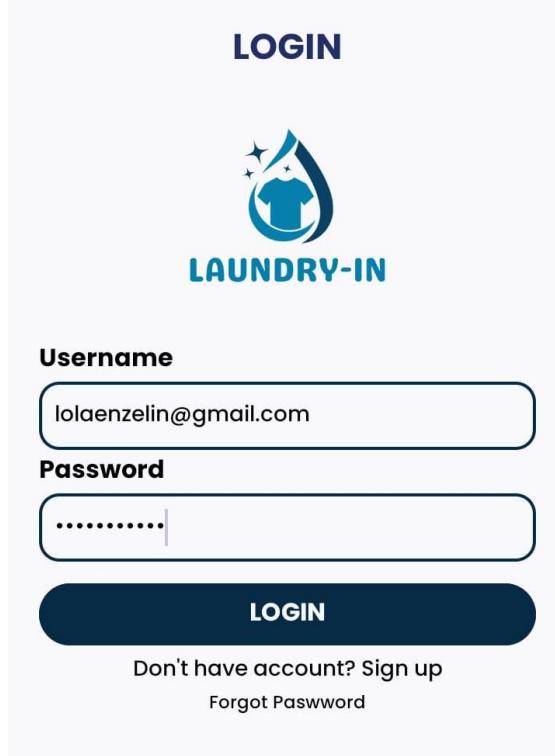
for **lolaenzelin@gmail.com**

New password
lola2004enz

SAVE

Gambar 4.1.2.7 Email

- Layout (Customer_login.xml)



Gambar 4.1.2.8 Layout Log-In customer

4.1.3 Register

1) Declaring dan Initializing

```
public class Customer_SignUp extends AppCompatActivity { @Override
    /**DECLARING*/
    2 usages
    Button btnSignUp;
    2 usages
    TextView txtLog;
    2 usages
    EditText name, email, pass, conf;
    5 usages
    ProgressBar progressBar;
    2 usages
    FirebaseAuth auth;
    2 usages
    FirebaseDatabase database;
    2 usages
    DatabaseReference reference;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.customer_sign_up);

        /**FIREBASE*/
        auth = FirebaseAuth.getInstance();

        /**INITIALIZING each Edit Text*/
        name = findViewById(R.id.InsertName_SUP);
        email = findViewById(R.id.edtMail);
        pass = findViewById(R.id.edtPass);
        conf = findViewById(R.id.edtConf);
        progressBar = findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);
    }
}
```

Gambar 4.1.3.1 Declaring dan Initializing

- Teks intent jika ingin ke halaman Log-In

```
/**Text Intent to CUSTOMER Login class*/
txtLog = findViewById(R.id.txtSup);
txtLog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), Customer_SignUp.this, Customer_Login.class));
    }
});
```

Gambar 4.1.3.2 Kode teks inten ke halaman Log-In

- Tombol Sign-Up

```
/**BUTTON SIGN UP*/
btnSignUp = findViewById(R.id.btnSignUpCustomer);
btnSignUp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        createUser();
        progressBar.setVisibility(View.VISIBLE);
    }
});
```

Gambar 4.1.3.3 Kode tombol Sign-Up

Memanggil method createUser() , dan menampilkan progressBar. Berikut, code method:

- Peringatan Persyaratan

```
/*Alert*/
if (TextUtils.isEmpty(userName)) {
    Toast.makeText(context, "Your Name is Empty!", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userEmail)) {
    Toast.makeText(context, "Your Email is Empty!", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userPass)) {
    Toast.makeText(context, "Your Password is Empty!", Toast.LENGTH_SHORT).show();
    return;
} //Password harus lebih dari 6 length
if (userPass.length() < 6){
    Toast.makeText(context, "Password Length must be greater than 6 letter", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(userConf)) {
    Toast.makeText(context, "Please confirm your Password!", Toast.LENGTH_SHORT).show();
    return;
} //Password Confirm harus sama persis dengan password yang dibuat
if (!userConf.equals(userPass)) {
    Toast.makeText(context, "Passwords do not match", Toast.LENGTH_SHORT).show();
    return;
}
```

Gambar 4.1.3.4 Kode Peringatan Persyaratan

2) Menyambungkan ke Firebase

```
/**Create User connect to FireBase*/
auth.createUserWithEmailAndPassword(userEmail, userPass)
    .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // User creation successful
                String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
                database = FirebaseDatabase.getInstance();
                reference = database.getReference( path: "Users").child(userId);

                /**For Realtime Customer Information*/
                CustomerModel customerModel = new CustomerModel(userName, userEmail, userPass);
                reference.setValue(customerModel);

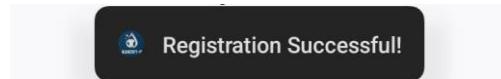
                /**ProgressBar*/
                progressBar.setVisibility(View.GONE);

                /**Successful Alert*/
                Toast.makeText( context: Customer_SignUp.this, text: "Registration Successful!",
                    Toast.LENGTH_SHORT).show();

                /**Intent*/
                startActivity(new Intent( packageContext: Customer_SignUp.this, Customer_Login.class));

            } else {
                // If sign in fails, display a message to the user.
                progressBar.setVisibility(View.GONE);
                Toast.makeText( context: Customer_SignUp.this, text: "Error: " + task.getException(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    })
}
```

Jika Berhasil:



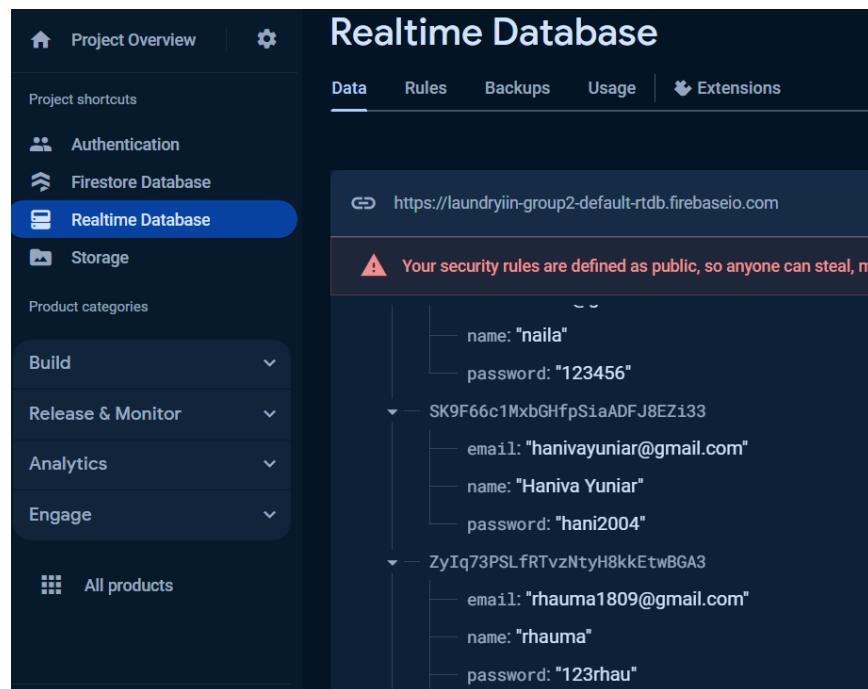
Gambar 4.1.3.5 Registrasi berhasil

- **Output
Authentification:**

A screenshot of the Firebase Authentication screen in the Firebase console. The left sidebar shows project details like 'Project Overview', 'Authentication' (which is selected), 'Firestore Database', 'Realtime Database', and 'Storage'. The main area is titled 'Authentication' and has tabs for 'Users', 'Sign-in method', 'Templates', 'Usage', 'Settings', and 'Extensions'. A search bar at the top allows searching by email address, phone number, or user UID. Below the search bar is a table with columns: Identifier, Providers, Created, Signed In, and User UID. The table lists several user accounts, each with their email address, provider (mostly email), creation date, sign-in date, and unique User UID. The first few rows include 'hanilivayuniar@gmail.com', 'oce@gmail.com', 'widiqwaskitoam05@gmail.com', 'lolaenzelin@gmail.com', 'nailia@gmail.com', 'rhuma1899@gmail.com', and 'hanilivayuniar@gmail.com' again.

Gambar 4.1.3.6 Firebase Authetification

Realtime Database:



The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with project navigation links like Project Overview, Authentication, Firestore Database, and Realtime Database (which is highlighted). Below that are sections for Product categories, Build, Release & Monitor, Analytics, and Engage, with a link to All products. The main area is titled "Realtime Database" and has tabs for Data, Rules, Backups, Usage, and Extensions. Under the Data tab, it shows a URL: <https://laundryin-group2-default-rtbd.firebaseio.com>. A warning message states: "⚠ Your security rules are defined as public, so anyone can steal, modify or delete your data." Below this, the database structure is displayed with three users: naila, SK9F66c1MxbGHfpSiaADFJ8EZi33, and ZyTq73PSLfRTvzNtyH8kkEtwBGA3. Each user has fields for name, password, email, and a unique identifier.

```
name: "naila"
password: "123456"
SK9F66c1MxbGHfpSiaADFJ8EZi33
email: "hanivayuniar@gmail.com"
name: "Haniva Yuniar"
password: "hani2004"
ZyTq73PSLfRTvzNtyH8kkEtwBGA3
email: "rhauma1809@gmail.com"
name: "rhauma"
password: "123rhau"
```

Gambar 4.1.3.7 Realtime Database

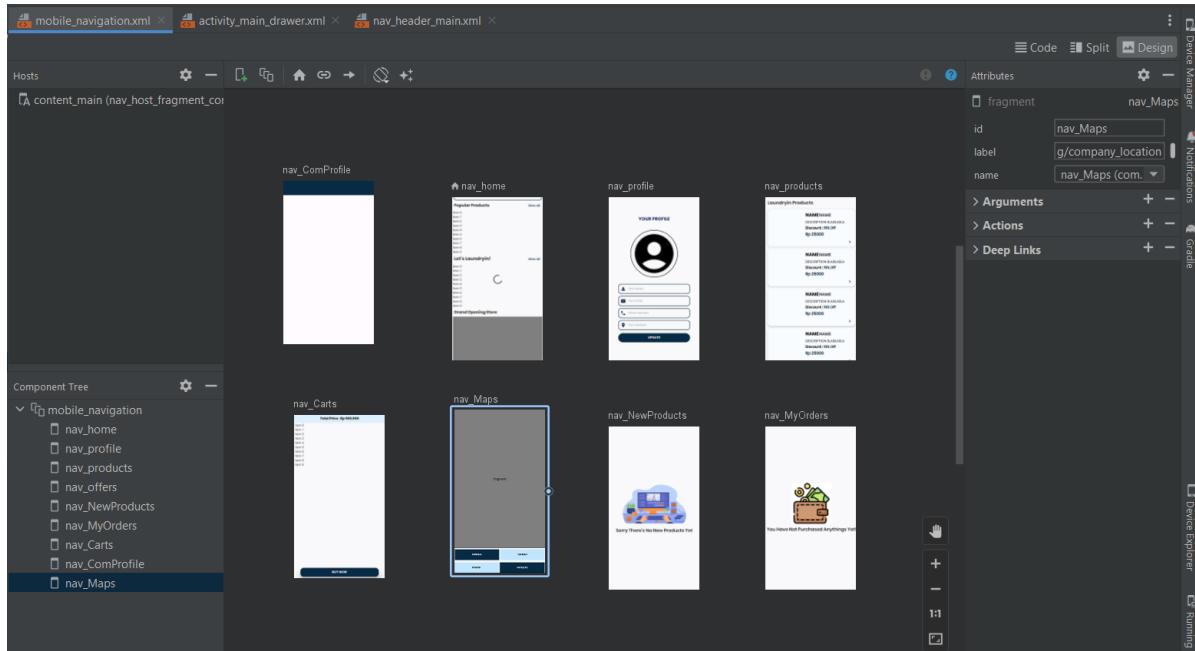
- Layout



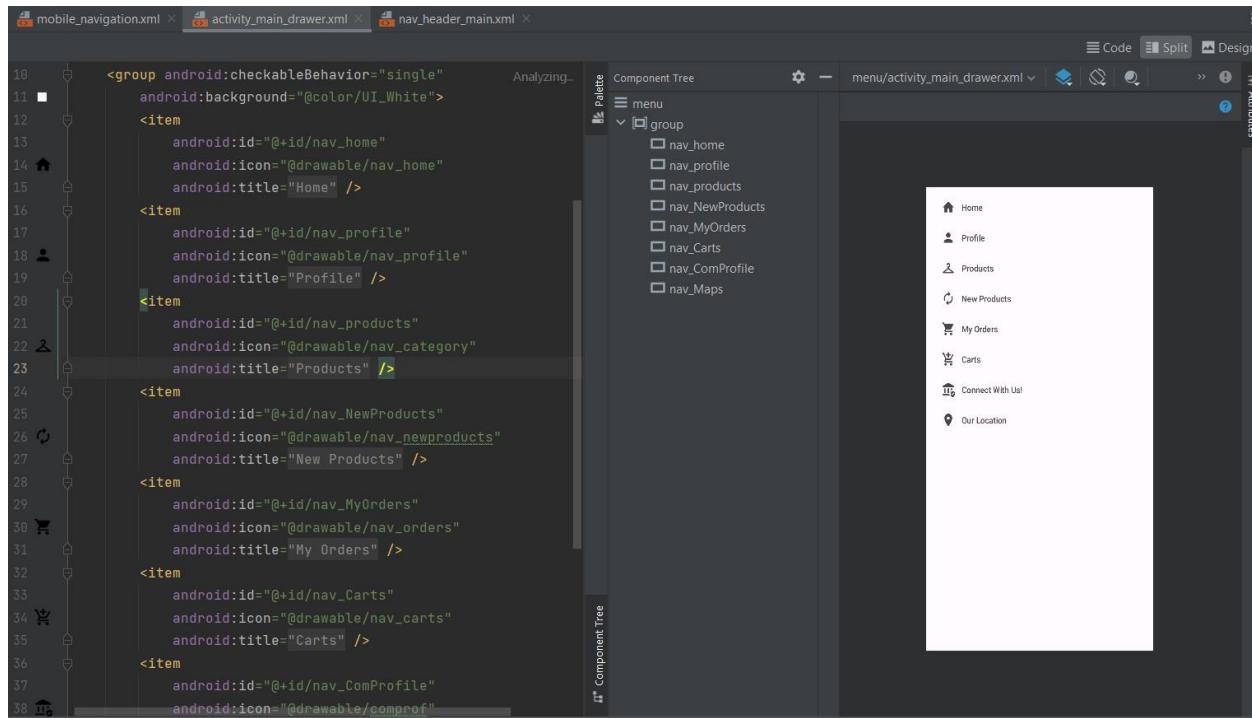
Gambar 4.1.3.7 Layout

3) Activity Main

- activity_main (Side Nav Bar) -> masuk banyak fragment



Gambar 4.1.3.8 Activity Main



Gambar 4.1.3.9 Activity Main Drawer

Kode XML ini mendefinisikan menu navigasi Android dengan grup yang dapat diselidiki. Ini mencakup item untuk Rumah, Profil, Produk, Produk Baru, Pesanan Saya, Kartu, dan Pengaturan

Profil. Setiap item memiliki ID unik, ikon terkait, dan judul. Kode ini mungkin melengkapi logika Java/Kotlin untuk menangani interaksi item menu dalam aplikasi Android.

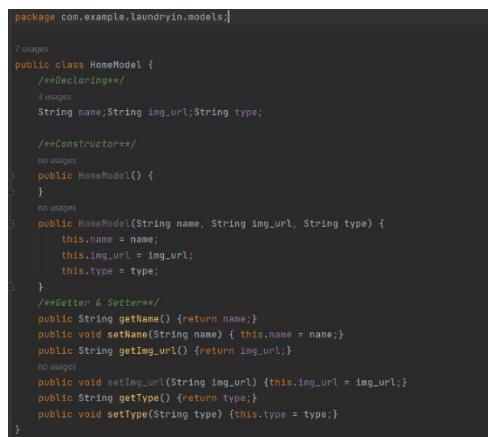
4) Home Page Fragment

fragment_nav_home (Home Page)

- EditText : Searching
- Popular Products : Products (home_popular_products.xml) (RecyclerView) -> Intent to (fragment_nav_products.xml)
- Let's Laundryin! : Categories (home_category_laundry.xml) (RecyclerView) -> Intent to (activity_views_all.xml)
- S-- Grand Opening : Multimedia Implementation (VideoView)

Models (HomeModel.java)

Model (HomeModel): mewakili struktur data atau entitas. Ia bertanggung jawab untuk menyimpan dan mengelola data.



```
package com.example.laundryin.models;

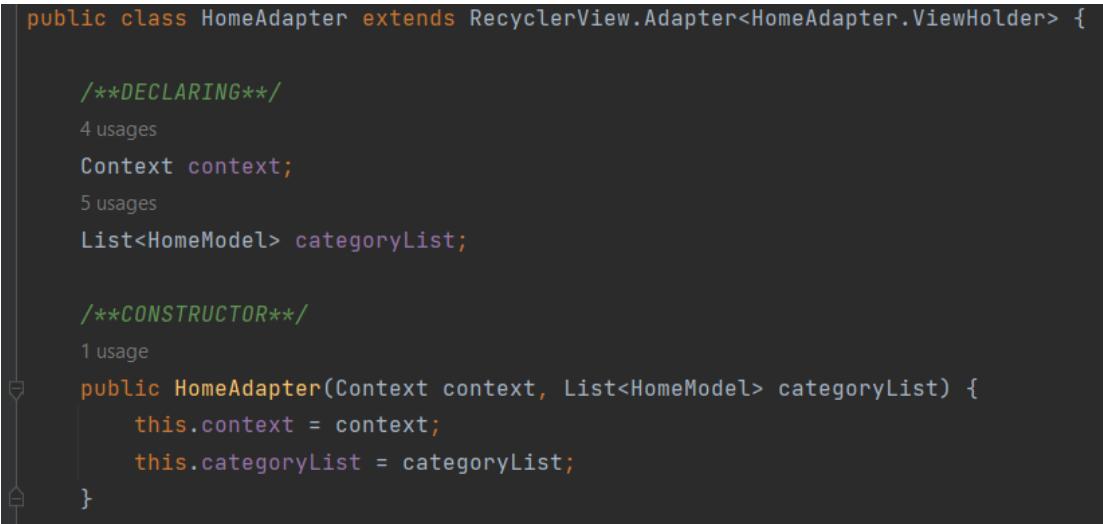
7 usages
public class HomeModel {
    /**Declarations*/
    3 usages
    String name;String img_url;String type;

    /**Constructors*/
    no usages
    public HomeModel() {
    }
    no usages
    public HomeModel(String name, String img_url, String type) {
        this.name = name;
        this.img_url = img_url;
        this.type = type;
    }
    /**Getter & Setter*/
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public String getImg_url() {return img_url;}
    public void setImg_url(String img_url) {this.img_url = img_url;}
    public String getType() {return type;}
    public void setType(String type) {this.type = type;}
}
```

Gambar 4.1.3.10 Home Model

Adapters (HomeAdapter.java)

Adapter (HomeAdapter): bertindak sebagai perantara antara data (model) dan UI (views). Ini membantu dalam membuat tampilan untuk setiap item dan mengikat data ke mereka, memungkinkan data untuk ditampilkan di UI secara efisien.



Gambar 4.1.3.10 Home Adapter

```
/** Called when the RecyclerView needs a new ViewHolder to represent an item.*/
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    return new ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.home_category_laundry,parent, attachToRoot: false));
}

/**Called by RecyclerView to display the data at the specified position.*/
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    // Load image using Glide library into the ImageView
    Glide.with(context).load(categoryList.get(position).getImg_url()).into(holder.catImg);
    // Set the name text in the TextView
    holder.name.setText(categoryList.get(position).getName());
    /**Set an OnClickListener to handle item click events*/
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            int adapterPosition = holder.getAdapterPosition();
            /**INTENT*/
            if (adapterPosition != RecyclerView.NO_POSITION) {
                Intent intent = new Intent(context, ViewsAllActivity.class);
                intent.putExtra("name", categoryList.get(adapterPosition).getType());
                context.startActivity(intent);
            }
        }
    });
}
```

Gambar 4.1.3.11 Home Adapter

```

/** Returns the total number of items in the data set held by the adapter. */
@Override
public int getItemCount() {
    return categoryList.size();
}

4 usages
public class ViewHolder extends RecyclerView.ViewHolder {
    /**Declaring & Initializing*/
2 usages
    ImageView catImg;
2 usages
    TextView name, type;
1 usage
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        catImg = itemView.findViewById(R.id.pop_img_cat);
        name = itemView.findViewById(R.id.pop_name_cat);
    }
}

```

Gambar 4.1.3.12 RecyclerView

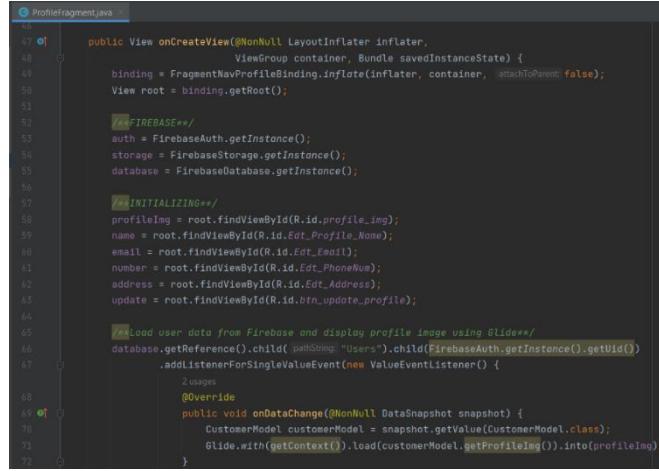
RecyclerView Adapter untuk Home Categories Adaptor ini bertanggung jawab untuk mengelola data dan membuat tampilan untuk menampilkan kategori rumah di RecyclerView. Ini mengikat data dari instansi HomeModel ke tampilan yang sesuai dan menangani peristiwa klik item untuk menavigasi ke ViewsAllActivity dengan rincian tambahan.

5) CRUD Profile Fragment

3. fragment_nav_profile.xml (CRUD Profile Picture)

-> Update profile picture sesuai foto profile, dan NAME + EMAIL

-> Will be update to the Side Bar navigation [nav_header_main.xml]



```

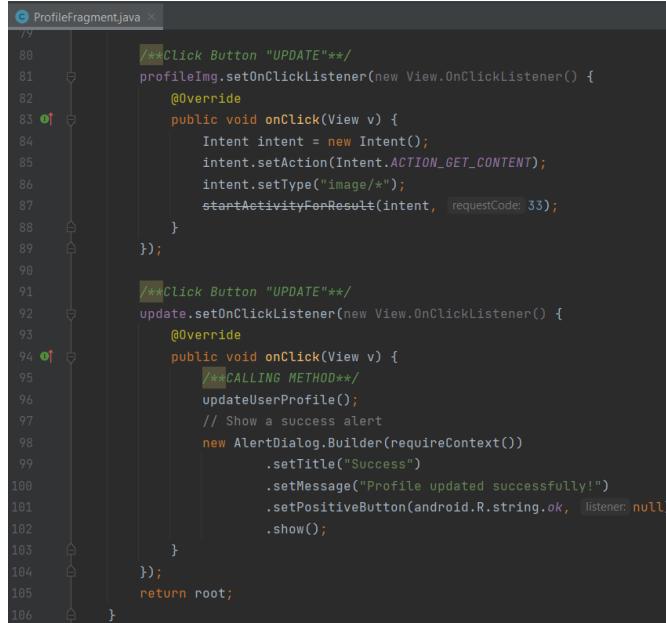
46
47     public View onCreateView(@NonNull LayoutInflater inflater,
48                             ViewGroup container, Bundle savedInstanceState) {
49         binding = FragmentNavProfileBinding.inflate(inflater, container, false);
50         View root = binding.getRoot();
51
52         //Initialize
53         auth = FirebaseAuth.getInstance();
54         storage = FirebaseStorage.getInstance();
55         database = FirebaseDatabase.getInstance();
56
57         //INITIALIZING
58         profileImg = root.findViewById(R.id.profile_img);
59         name = root.findViewById(R.id.edt_Profile_Name);
60         email = root.findViewById(R.id.edt_Email);
61         number = root.findViewById(R.id.edt_PhoneNum);
62         address = root.findViewById(R.id.edt_Address);
63         update = root.findViewById(R.id.btn_update_profile);
64
65         //Load user data from Firebase and display profile image using Glide
66         database.getReference().child("Users").child(FirebaseAuth.getInstance().getUid())
67             .addListenerForSingleValueEvent(new ValueEventListener() {
68                 2 usages
69                 @Override
70                 public void onDataChange(@NonNull DataSnapshot snapshot) {
71                     CustomerModel customerModel = snapshot.getValue(CustomerModel.class);
72                     Glide.with(getApplicationContext()).load(customerModel.getProfileImg()).into(profileImg);
73                 }
74             });

```

Gambar 4.1.3.13 CRUD

Metode ini menginisialisasi UI fragmen menggunakan view binding. Menginisialisasi Firebase Authentication, Storage, dan Database. Lalu, bagian “database.getReference” Mengambil data

pengguna dari Firebase, khususnya bidang profileImg, dan menggunakan perpustakaan Glide untuk memuat gambar profil.



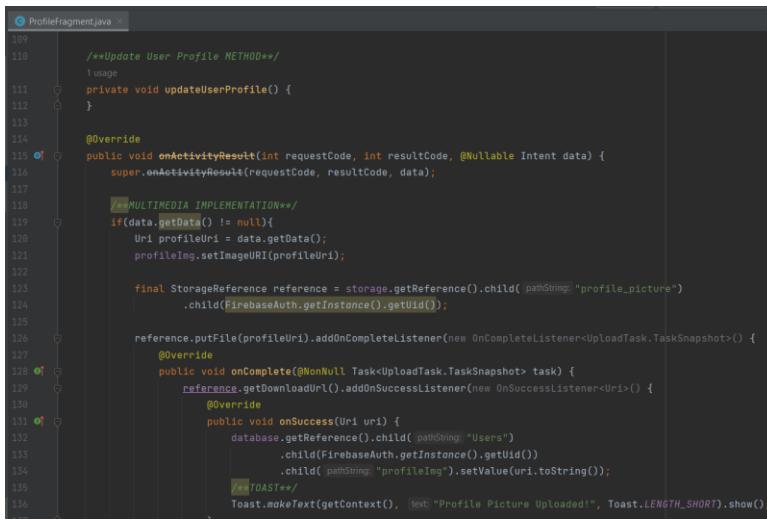
```

7
80     /**Click Button "UPDATE"*/
81     profileImg.setOnClickListener(new View.OnClickListener() {
82         @Override
83         public void onClick(View v) {
84             Intent intent = new Intent();
85             intent.setAction(Intent.ACTION_GET_CONTENT);
86             intent.setType("image/*");
87             startActivityForResult(intent, requestCode: 33);
88         }
89     });
90
91     /**Click Button "UPDATE"*/
92     update.setOnClickListener(new View.OnClickListener() {
93         @Override
94         public void onClick(View v) {
95             /**CALLING METHOD**/
96             updateUserProfile();
97             // Show a success alert
98             new AlertDialog.Builder(requireContext())
99                 .setTitle("Success")
100                .setMessage("Profile updated successfully!")
101                .setPositiveButton(android.R.string.ok, listener: null)
102                .show();
103         }
104     });
105     return root;
106 }

```

Gambar 4.1.3.14 Button Update

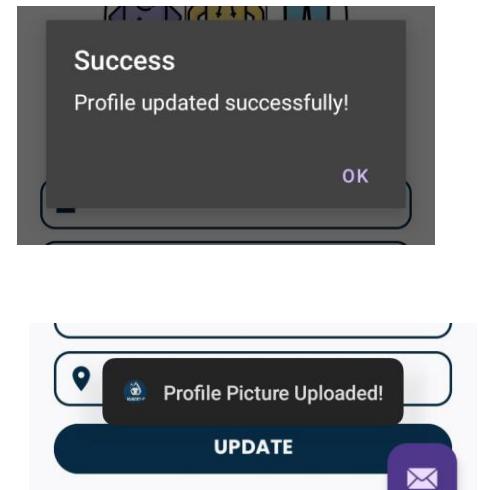
Update process apabila klik profileimg, lalu setelah pilih gambar, klik tombol button Update, sehingga terupdate di database.



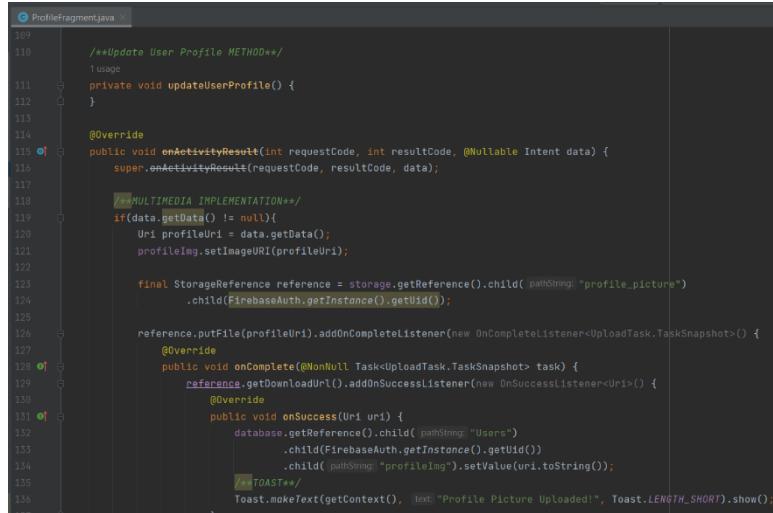
```

109
110     /**Update User Profile METHOD*/
111     private void updateUserProfile() {
112     }
113
114     @Override
115     public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
116         super.onActivityResult(requestCode, resultCode, data);
117
118         /*MULTIMEDIA IMPLEMENTATION*/
119         if(data.getData() != null){
120             Uri profileUri = data.getData();
121             profileImg.setImageURI(profileUri);
122
123             final StorageReference reference = storage.getReference().child( pathString: "profile_picture" )
124                         .child(FirebaseAuth.getInstance().getUid());
125
126             reference.putFile(profileUri).addOnCompleteListener(new OnCompleteListener<UploadTask.TaskSnapshot>() {
127                 @Override
128                 public void onComplete(@NonNull Task<UploadTask.TaskSnapshot> task) {
129                     reference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
130                         @Override
131                         public void onSuccess(Uri uri) {
132                             database.getReference().child( pathString: "Users" )
133                                 .child(FirebaseAuth.getInstance().getUid())
134                                 .child( pathString: "profileImg" ).setValue(uri.toString());
135
136                         /*TOAST*/
137                         Toast.makeText(getContext(), text: "Profile Picture Uploaded!", Toast.LENGTH_SHORT).show();
138                     }
139                 }
140             });
141         }
142     }

```



Gambar 4.1.3.15 Update Profile



```

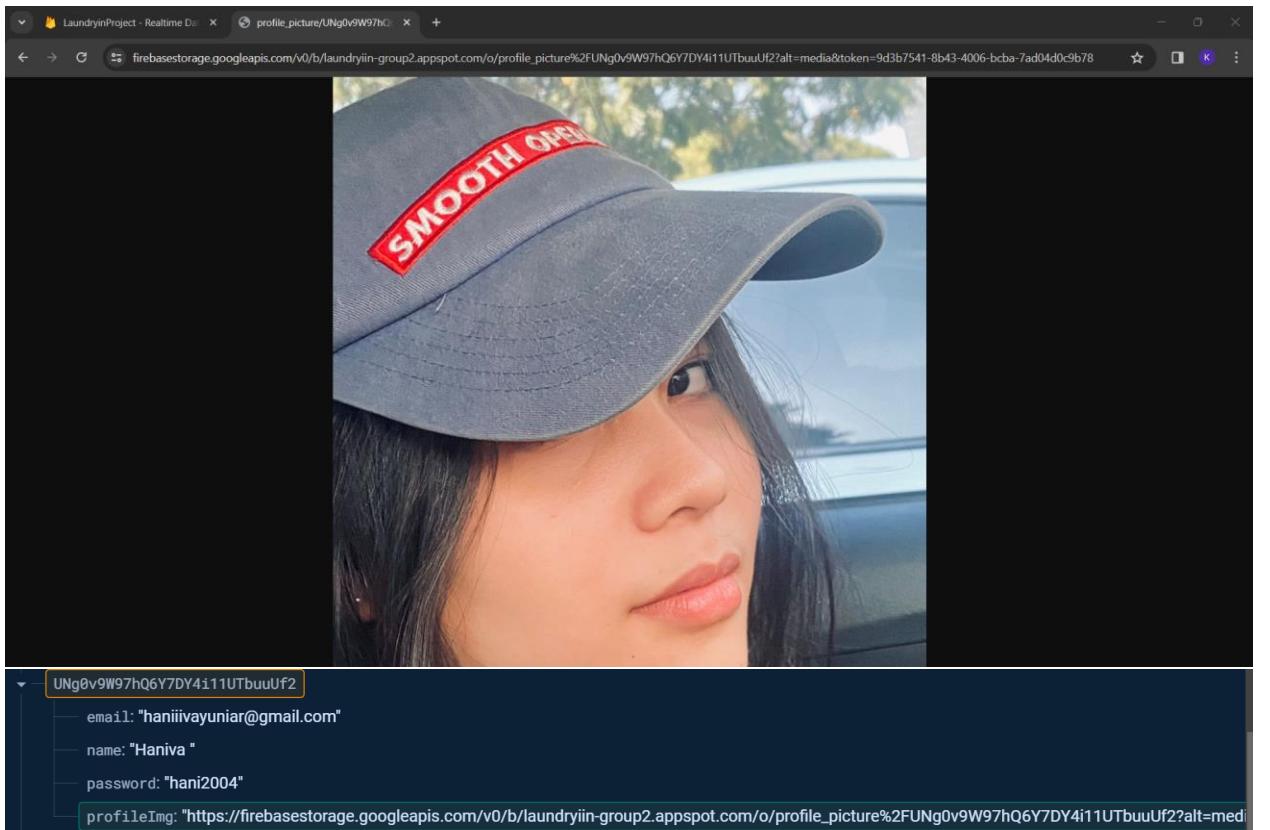
109     /**Update User Profile METHOD*/
110     1 usage
111     private void updateUserProfile() {
112     }
113
114     @Override
115     public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
116         super.onActivityResult(requestCode, resultCode, data);
117
118         /*MULTIMEDIA IMPLEMENTATION*/
119         if(data.getData() != null){
120             Uri profileUri = data.getData();
121             profileImg.setImageURI(profileUri);
122
123             final StorageReference reference = storage.getReference().child( pathString: "profile_picture" )
124                         .child(FirebaseAuth.getInstance().getUid());
125
126             reference.putFile(profileUri).addOnCompleteListener(new OnCompleteListener<UploadTask.TaskSnapshot>() {
127                 @Override
128                 public void onComplete(@NonNull Task<UploadTask.TaskSnapshot> task) {
129                     reference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
130                         @Override
131                         public void onSuccess(Uri uri) {
132                             database.getReference().child( pathString: "Users" )
133                                 .child(FirebaseAuth.getInstance().getUid())
134                                     .child( pathString: "profileImg").setValue(uri.toString());
135
136                         /*TOAST*/
137                         Toast.makeText(getApplicationContext(), text: "Profile Picture Uploaded!", Toast.LENGTH_SHORT).show();
138                     }
139                 }
140             });
141         }
142     }

```

Gambar 4.1.3.16 Update User Profile

Kode tersebut menangani pemilihan gambar dari galeri, mengunggahnya ke Firebase Storage, menyimpan URL gambar di Database Firebase, dan menampilkan pesan Toast kepada pengguna setelah berhasil mengunggah gambar profil.

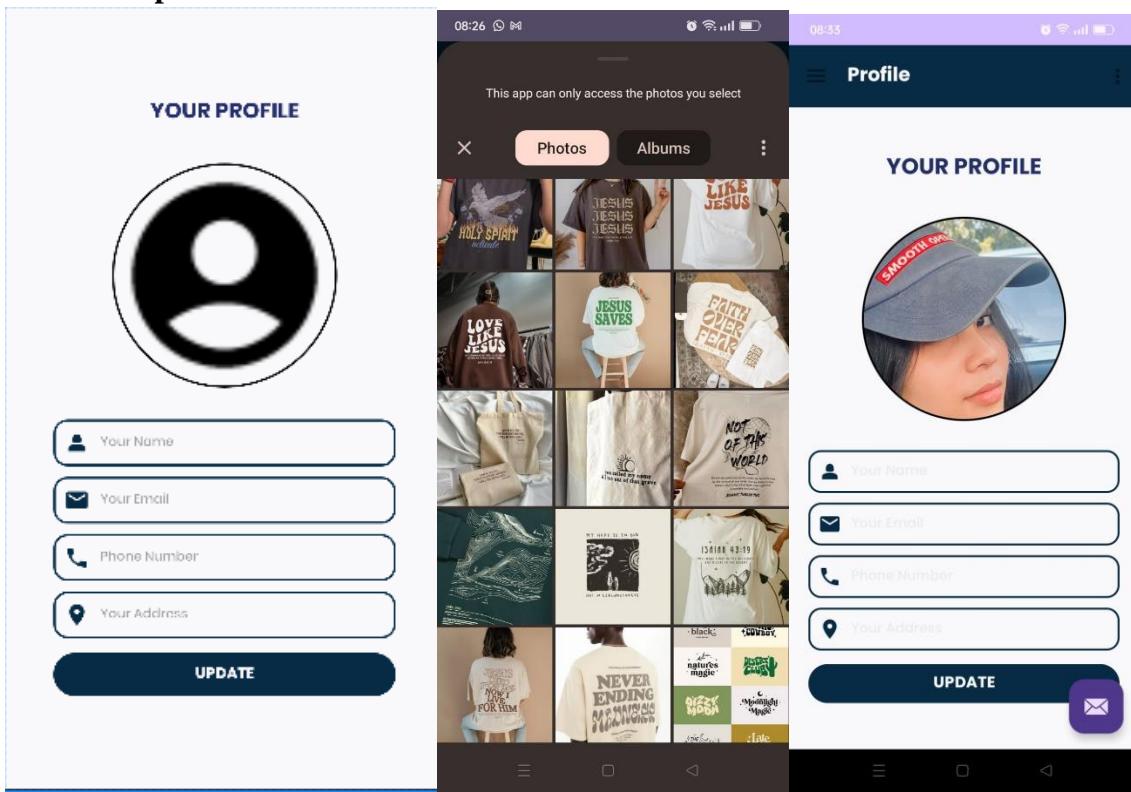
- **Database Realtime:**



Gambar 4.1.3.16 Database Realtime

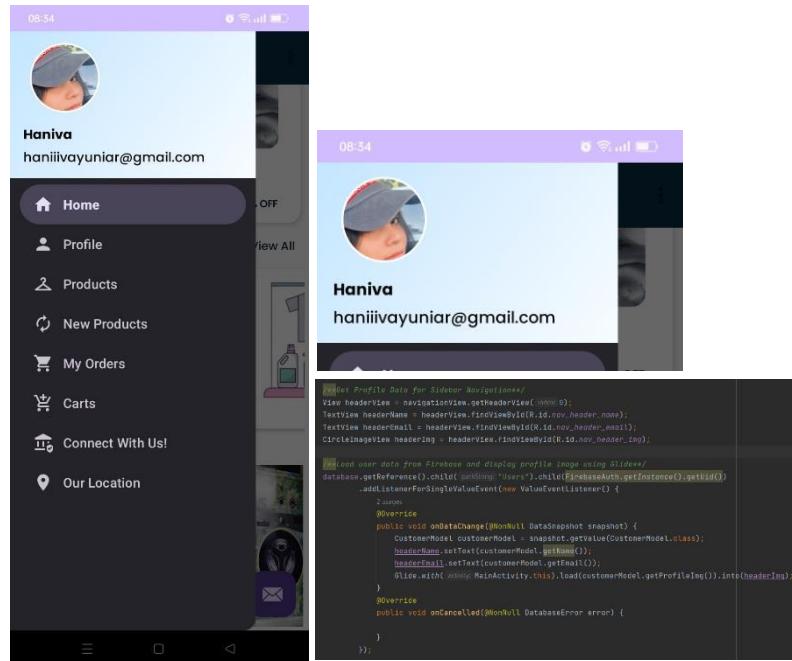
Apa yang User upload untuk foto profile akan ter update secara realtime

- **Output**



Gambar 4.1.3.17 Output Database Realtime

[MainActivity <-> nav_header_main.xml]



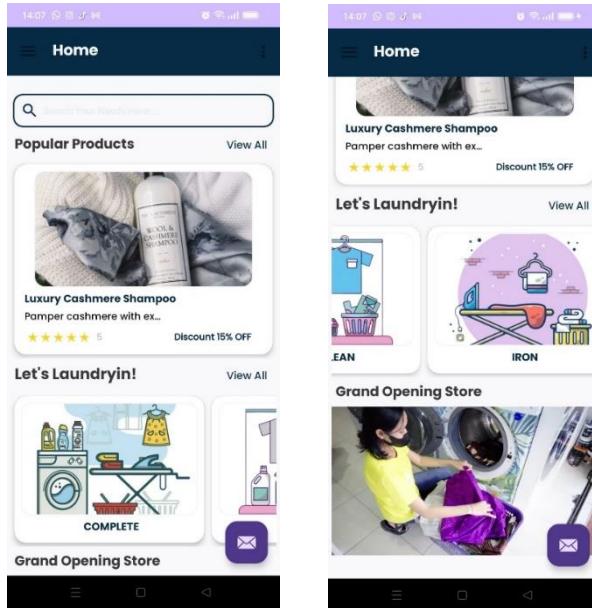
Gambar 4.1.3.18 Output Database Realtime

Kode ini menampilkan informasi produk dari database yang disebut "produk" di dalam koleksi yang bernama "laundry". Ini menggunakan ID produk tertentu untuk menarik detail seperti nama, harga, dan kategori. Kode kemudian memformat informasi ini untuk ditampilkan di antarmuka pengguna, kemungkinan toko online.

6) Home Fragment

HomeFragment.java

- Layout:



Gambar 4.1.3.19 Layout Home Fragment

Initializing

```

59
60     public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
61         binding = FragmentNavHomeBinding.inflate(inflater, container, false);
62         View root = binding.getRoot();
63
64
65         /*INITIALIZING*/
66         db = FirebaseFirestore.getInstance();
67         popularRec = root.findViewById(R.id.pop_rec);
68         homeCatRec = root.findViewById(R.id.explore_cat);
69         scrollView = root.findViewById(R.id.scroll_view);
70         progressBar = root.findViewById(R.id.progressbar);
71
72         /*VISIBLE & GONE*/
73         progressBar.setVisibility(View.VISIBLE);
74         scrollView.setVisibility(View.GONE);
75
76         //Popular Products
77         popularRec.setLayoutManager(new LinearLayoutManager(getActivity(), RecyclerView.HORIZONTAL, false));
78         popularModelList = new ArrayList<>();
79         popularAdapters = new PopularAdapters(getActivity(), popularModelList);
80         popularRec.setAdapter(popularAdapters);

```

Gambar 4.1.3.20 Initializing Home Fragment

Kode ini digunakan untuk memulai elemen UI dari fragmen rumah. Ini mengatur Firebase Firestore untuk interaksi database dan mengelola Recycler View untuk tampilan produk populer.

Kode ini menangani visibilitas UI, memulai adaptor, dan mengkonfigurasi manajer tata letak untuk scroll horizontal.

- **Popular Products**



```

/*POPULAR PRODUCTS - READ*/
db.collection( collectionPath: "PopularProducts" ) CollectionReference
    .get() Task<QuerySnapshot>
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult()) {

                        PopularModel popularModel = document.toObject(PopularModel.class);
                        popularModelList.add(popularModel);
                        popularAdapters.notifyDataSetChanged();

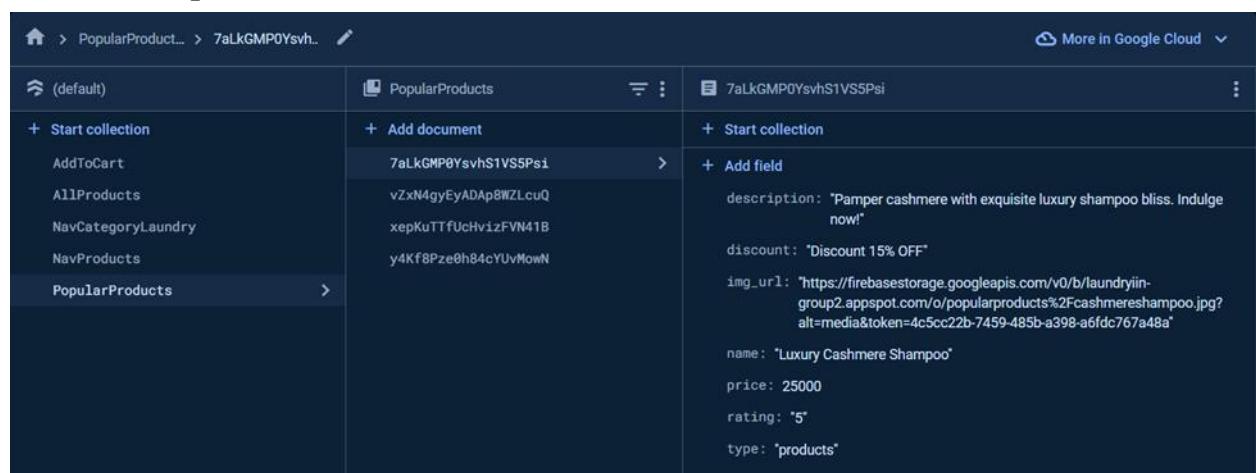
                        /*VISIBLE & GONE*/
                        progressBar.setVisibility(View.GONE);
                        scrollView.setVisibility(View.VISIBLE);

                    }
                } else {
                    /*TOAST*/
                    Toast.makeText(getApplicationContext(), "Error" + task.getException(), Toast.LENGTH_SHORT).show();
                }
            }
        });
    
```

Gambar 4.1.3.21 Popular Products

Kode ini menggunakan Firestore untuk mengambil data dari koleksi "PopularProducts" secara asynchronous. 'addOnCompleteListener' menangani hasilnya, memperbarui 'popularModelList' dengan objek 'Popular Model' yang mewakili produk populer. Daftar ini kemudian ditampilkan dalam RecyclerView melalui adaptor 'popularAdapters', dengan elemen UI seperti ProgressBar dan ScrollView mengelola keadaan visual. Pesan toast digunakan untuk umpan balik kesalahan, meningkatkan pengalaman pengguna. Kode ini mematuhi standar pengembangan Android untuk kueri Firestore dan pembaruan RecyclerView, berkontribusi pada UI yang responsif dan informatif dalam aplikasi Laundry-Home.

- **Database : (Product-Home)**
 - **Popular Products:**



	PopularProducts	7aLkGMP0YsvhS1VS5Psi
+ Start collection	+ Add document	+ Start collection
AddToCart	7aLkGMP0YsvhS1VS5Psi >	+ Add field
AllProducts	vZxN4gyEyADAp8WLcuQ	description: "Pamper cashmere with exquisite luxury shampoo bliss. Indulge now!"
NavCategoryLaundry	xepKuTTfUchHvi2FVN41B	discount: "Discount 15% OFF"
NavProducts	y4Kf8Pze0h84cYUvMowN	img_url: "https://firebasestorage.googleapis.com/v0/b/laundryiiin-group2.appspot.com/o/popularproducts%2fcashmereshampoo.jpg?alt=media&token=4c5cc22b-7459-485b-a398-a6fdc767a48a"
PopularProducts >		name: "Luxury Cashmere Shampoo"
		price: 25000
		rating: "5"
		type: "products"

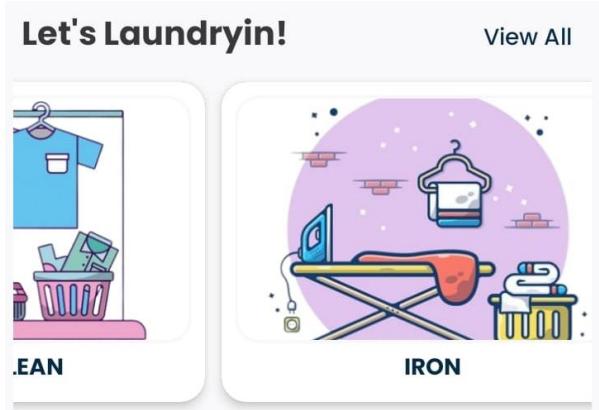
Gambar 4.1.3.22 Database Popular Products

Proyek Firebase ini tampaknya untuk mengelola toko online yang disebut "Laundryin." Dashboard menampilkan bagian untuk mengelola produk, kategori, dan mungkin pesanan dan pelanggan.

- Laundry Categories

```
//Explore Laundry Categories
homeCatRec.setLayoutManager(new LinearLayoutManager(getActivity(), RecyclerView.HORIZONTAL, false));
homeModelList = new ArrayList<>();
homeAdapter = new HomeAdapter(getActivity(), homeModelList);
homeCatRec.setAdapter(homeAdapter);

/*Explore Laundry Categories - READ*/
db.collection("NavCategoryLaundry").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                HomeModel homeModel = document.toObject(HomeModel.class);
                homeModelList.add(homeModel);
                homeAdapter.notifyDataSetChanged();
            }
        } else {
            /*TOAST*/
            Toast.makeText(getActivity(), "Error" + task.getException(), Toast.LENGTH_SHORT).show();
        }
    }
});
```



Gambar 4.1.3.23 Laundry Categories

Kode ini untuk mengatur RecyclerView (`homeCatRec`) untuk menampilkan kategori pakaian secara horizontal. Ini memulai daftar kosong (`homeModelList`) dan adaptor (`homeAdapter`) untuk pengolahan data RecyclerView. Sebuah kueri Firestore asynchronous mengambil data dari koleksi "NavCategoryLaundry". Dalam cabang keberhasilan, setiap dokumen dalam QuerySnapshot menjadi objek 'HomeModel' yang ditambahkan ke 'homeModelList', memicu pemberitahuan perubahan dataset dengan 'notifyDataSetChanged()' pada adaptor. Dalam kasus kesalahan kueri Firestore, pesan Toast memperingatkan pengguna. Secara keseluruhan, kode menciptakan UI responsif dengan secara dinamis memperbarui RecyclerView dengan kategori pakaian dari Firestore, memasukkan umpan balik kesalahan untuk pengalaman pengguna yang lebih baik.

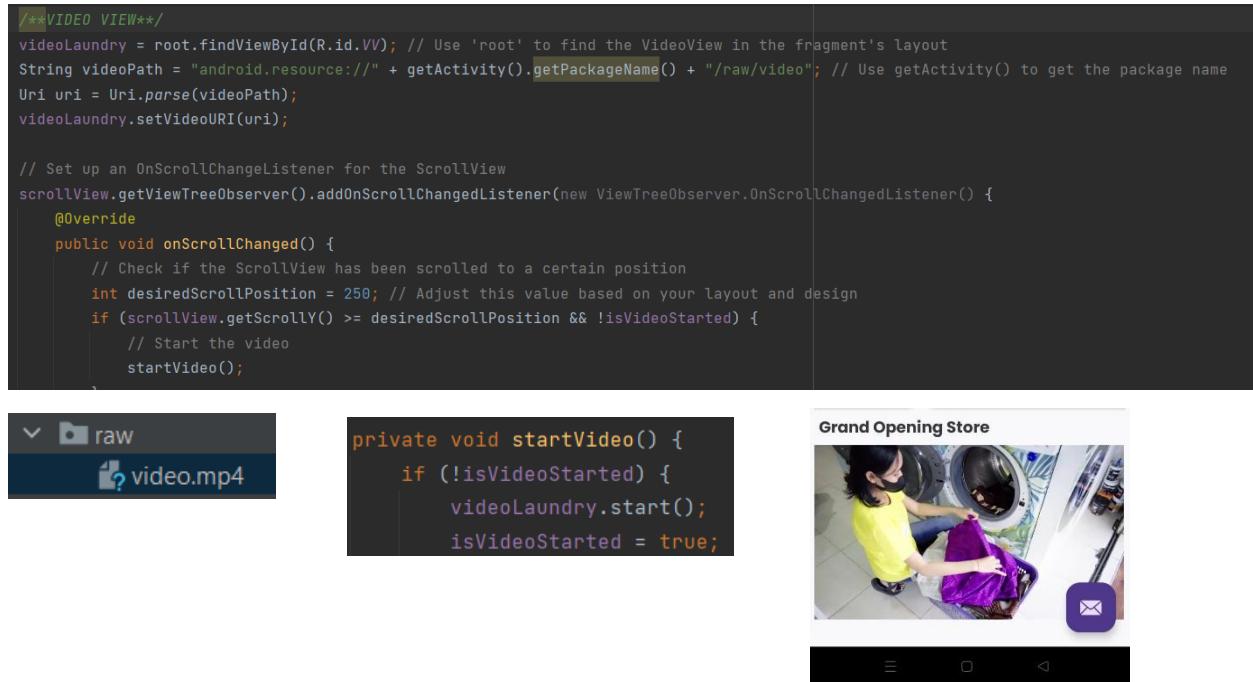
- Database: (Laundry-Home)
- NavCategoryLaundyr (3)-> All Products

(default)	NavCategoryLaundry	5MiFWkcPVAcx7yPdFtiY	
+ Start collection	+ Add document	+ Start collection	
AddToCart	5MiFWkcPVAcx7yPdFtiY >	+ Add field	
AllProducts	VHgDVfW7nNOFeaoKPM3H	img_url: "https://firebasestorage.googleapis.com/v0/b/laundryin-group2.appspot.com/o/categorylaundrylist%2Fcomplete.jpg?alt=media&token=ee06c136-0764-4b9a-9a95-a3d4ce38a860"	
NavCategoryLaundry >	ZA3zYULW2vSmwHiWvSFJ	name: "COMPLETE"	
NavProducts		type: "complete"	
PopularProducts			

Gambar 4.1.3.24 Database Laundry Categories

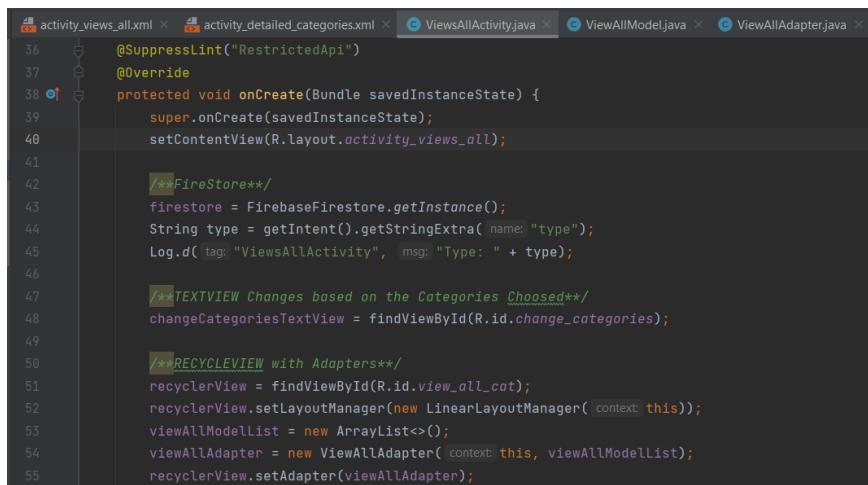
Firebase digunakan sebagai database NoSQL real-time, di mana rincian produk, seperti nama, harga, peringkat, dan URL gambar, disimpan.

- Grand Opening
- ## Multimedia Implementation



Gambar 4.1.3.25 Multimedia

- Kode ini memulai VideoView (`videoLaundry`) dalam fragment, mengatur sumber video ke sumber dari folder "raw", dan menggunakan ScrollView untuk memicu pemutaran video ketika pengguna meluncur ke posisi tertentu. Metode 'startVideo' bertanggung jawab untuk memulai pemutaran video. Kode meningkatkan pengalaman pengguna dengan memulai video secara dinamis berdasarkan posisi scroll.
- **Click Categories Laundry**
 - **(activity_views_all.xml)**



Gambar 4.1.3.26 Activity Views

Dalam snippet kode Java Android ini, dalam metode 'onCreate', aktivitas ini dimulai dengan tata letak yang ditentukan. Ini mengambil instansi Firestore, mencatat parameter tipe, dan mengatur RecyclerView untuk menampilkan item berdasarkan kategori yang dipilih. Kode secara dinamis memperbarui TextView dan mengkonfigurasi adaptor untuk menampilkan data.

```
/**Subtitle below the Toolbar will change based on what categories the customer choose*/
// Check if the "type" is not null before accessing its value
if (type != null) {
    // Set the TextView based on the "type"
    switch (type.toLowerCase()) {
        case "complete":
            changeCategoriesTextView.setText("Complete");
            break;
        case "iron":
            changeCategoriesTextView.setText("Iron");
            break;
        case "dryclean":
            changeCategoriesTextView.setText("Dryclean");
            break;
        default:
            // Handle other cases if needed
            break;
    }
}
```

Gambar 4.1.3.27 Categories

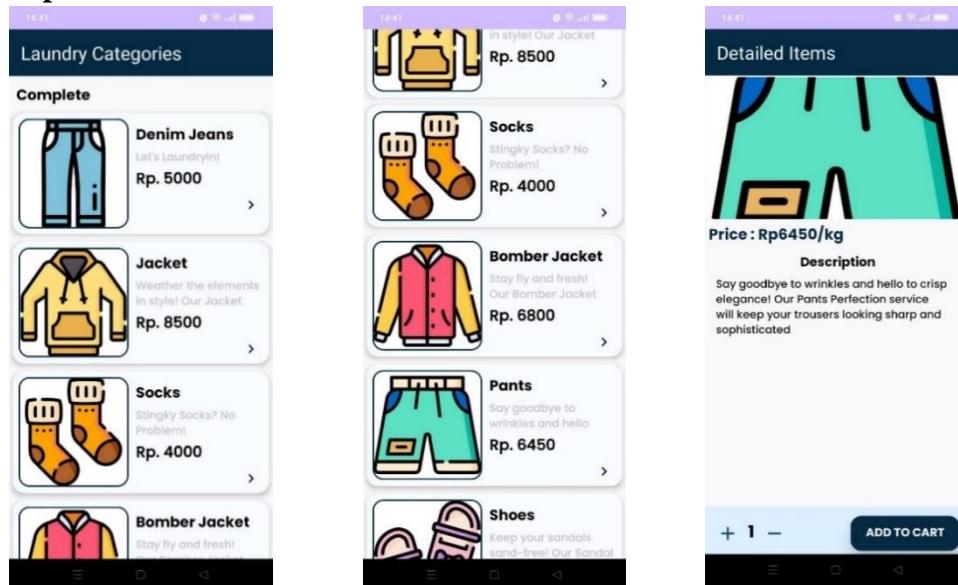
Kode Java ini digunakan untuk memeriksa apakah jenis kategori ("komplit," "iron," atau "dryclean") tidak null sebelum secara dinamis memperbarui TextView di bawah Toolbar. Berdasarkan tipe, itu mengatur TextView untuk menampilkan nama kategori yang sesuai, menangani kasus yang berbeda dengan sopan.

design UI RecycleView : [design_view_all_categories.xml]

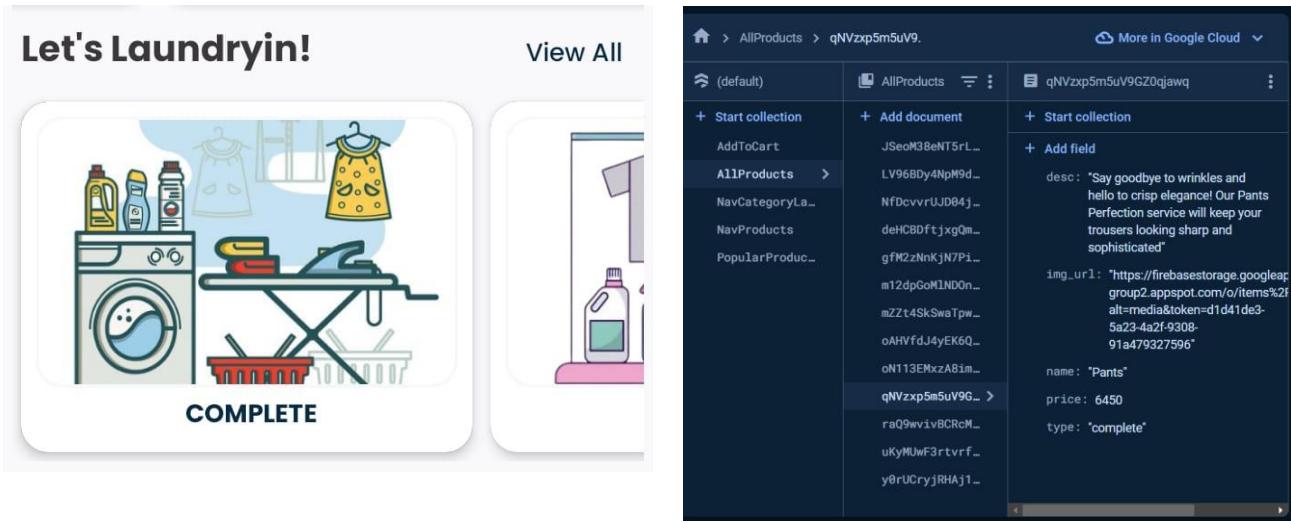
Each RecycleView clicked, it will Intent to Activity for detail items

[activity_detailed_categories.xml; DetailedCategoriesActivity.java]

- **Complete**



Gambar 4.1.3.28 Layout Categories



Gambar 4.1.3.29 Database Categories

```

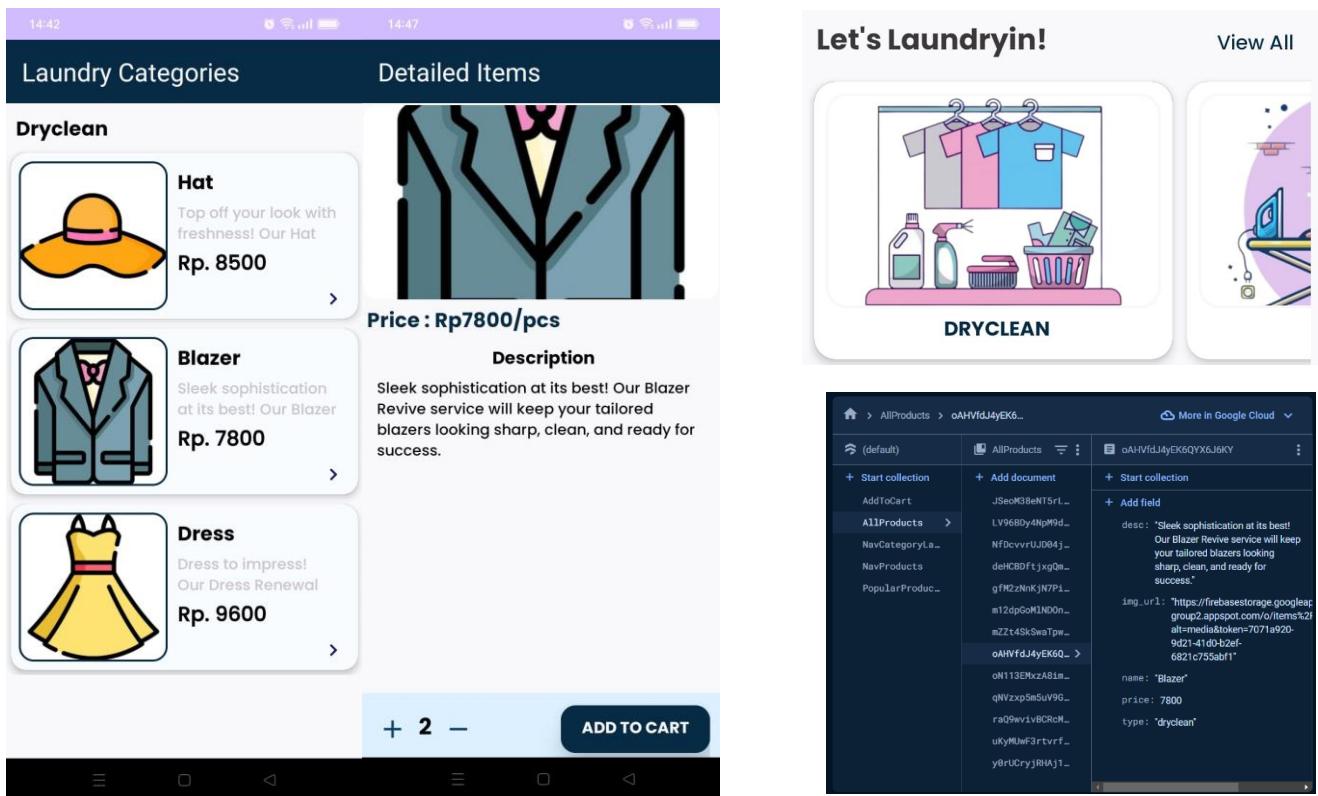
// Check if the "type" is not null before accessing its value
/** Getting IRON */
if (type != null) {
    // Getting Complete
    if (type.equalsIgnoreCase("complete")) {
        firestore.collection(collectionPath: "AllProducts").whereEqualTo(field: "type", value: "complete")
            .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    for (DocumentSnapshot documentSnapshot : task.getResult().getDocuments()) {
                        ViewAllModel viewAllModel = documentSnapshot.toObject(ViewAllModel.class);
                        viewAllModelList.add(viewAllModel);
                    }
                    viewAllAdapter.notifyDataSetChanged(); // Move this line outside the loop
                }
            })
    }
}

```

Gambar 4.1.3.30 Kode Database Categories

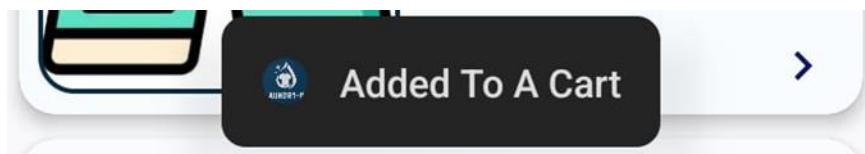
Kode Java ini untuk memeriksa jika tipe kategori ("komplit") tidak null. Jika benar, itu menginterogasi Firestore untuk produk jenis itu, mengambil data secara asynchronous, dan menampung daftar objek ViewAllModel. Adaptor RecyclerView kemudian diberitahu tentang perubahan data untuk pembaruan UI dinamis.

- DRYCLEAN



Gambar 4.1.3.31 Database Categories

sesudah pencet tombol button "Add To Cart", maka akan masuk Cart (keranjang belanja)[fragment_nav_carts.xml]. Akan ada Toast untuk memberitahu customer bahwa item telah masuk kedalam keranjang sesuai quantity yang dimasukkan dengan total harga.



Gambar 4.1.3.32 Alert Add to Cart

Models:

Antarmuka 'Serializable' diimplementasikan dalam kelas 'ViewAllModel' untuk memungkinkan instansi dari kelas ini untuk dikonversi menjadi aliran byte. Ini memungkinkan transmisi mudah dari objek 'ViewAllModel' antara aktivitas atau fragmen di Android menggunakan Intent atau Bundle, memfasilitasi transfer data di berbagai komponen aplikasi.

```

16 usages
5   public class ViewAllModel implements Serializable {
6     /*Declaring*/
7     String name; String desc;String img_url;String type;int price;
8   }
9   no usages
10  public ViewAllModel() {}
11  no usages
12  public ViewAllModel(String name, String desc, String img_url, String type, int price) {
13    this.name = name;
14    this.desc = desc;
15    this.img_url = img_url;
16    this.type = type;
17    this.price = price;
18  }
19  /**
20   * Getter & Setter*/
21  public String getName() { return name; }
22  public void setName(String name) { this.name = name; }
23  2 usages
24  public String getDesc() { return desc; }
25  no usages
26  public void setDesc(String desc) { this.desc = desc; }
27  public String getImg_url() { return img_url; }
28  no usages
29  public void setImg_url(String img_url) { this.img_url = img_url; }
30  public String getType() { return type; }
31  public void setType(String type) { this.type = type; }
32  public int getPrice() { return price; }
33  public void setPrice(int price) { this.price = price; }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

```

Gambar 4.1.3.33 Kode Alert Add to Cart

`ViewAllModel.java` bertindak sebagai model data yang dirancang untuk mewakili struktur setiap kategori. Dengan menyediakan variabel seperti nama, deskripsi, URL gambar, jenis, dan harga, kelas ini berfungsi sebagai entitas penyimpanan informasi terkait kategori.

- **Adapters:**

```

/**CONSTRUCTOR*/
1 usage
public ViewAllAdapter(Context context, List<ViewAllModel> list) {
    this.context = context;
    this.list = list;
}
/** Called when the RecyclerView needs a new ViewHolder to represent an item.*/
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    return new ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.design_view_all_categories, parent, false));
}

/**Called by RecyclerView to display the data at the specified position.*/
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    // Load image using Glide library into the ImageView
    Glide.with(context).load(list.get(position).getImg_url()).into(holder.imageView);
    // Set the name, description, and price text in their respective TextViews
    holder.name.setText(list.get(position).getName());
    holder.desc.setText(list.get(position).getDesc());
    holder.price.setText(String.valueOf(list.get(position).getPrice())); // Convert int to String

    /**
     * Set an OnClickListener to handle item click events*/
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(context, DetailedCategoriesActivity.class);
            intent.putExtra("detail", list.get(position));
            context.startActivity(intent);
        }
    });
}

```

Gambar 4.1.3.34 Adapters

RecyclerView Adapter untuk Tampilkan Semua Kategori Adaptor ini mengelola data dan membuat tampilan untuk menampilkan informasi rinci tentang berbagai kategori dalam RecyclerView. Mengikat data dari instansi ViewAllModel ke tampilan yang sesuai dan menangani item Klik acara untuk menavigasi ke DetailCategoriesActivity untuk informasi lebih lanjut.

```
/** Returns the total number of items in the data set held by the adapter. */
@Override
public int getItemCount() { return list.size(); }
4 usages
public class ViewHolder extends RecyclerView.ViewHolder {
    /**Declaring & Initializing*/
    2 usages
    ImageView imageView; TextView name, desc, price;

    1 usage
    public ViewHolder(@NotNull View itemView) {
        super(itemView);

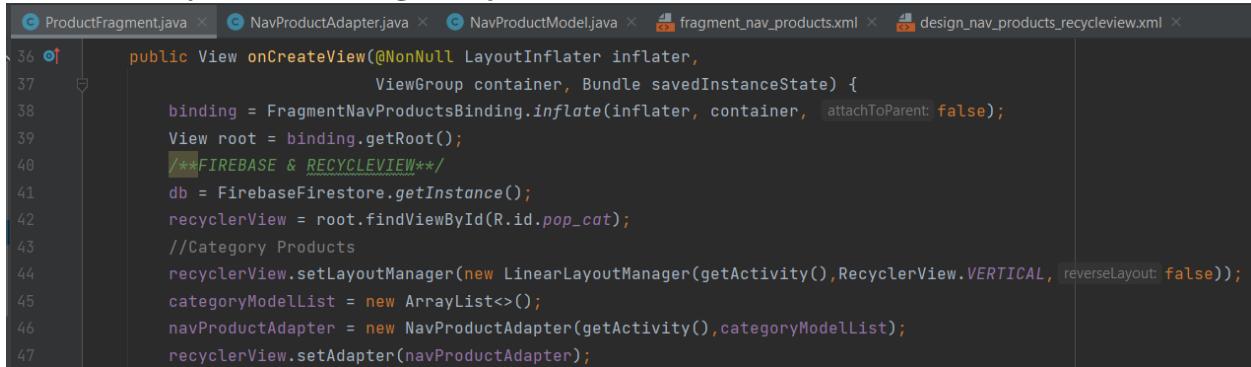
        imageView = itemView.findViewById(R.id.view_img);
        name = itemView.findViewById(R.id.view_cat_name);
        desc = itemView.findViewById(R.id.view_cat_desc);
        price = itemView.findViewById(R.id.view_cat_price);
```

Gambar 4.1.3.35 Item View

Di sisi lain, `ViewAllAdapter.java` berperan sebagai penghubung antara data (diwakili oleh model `ViewAllModel`) dan antarmuka pengguna (UI). Melalui implementasi RecyclerView, adapter ini menciptakan tampilan untuk menampilkan informasi rinci tentang berbagai kategori. Selain itu, adapter ini menanggapi interaksi pengguna, khususnya klik pada setiap item, dan membuka `DetailedCategoriesActivity` untuk menampilkan informasi lebih lanjut tentang kategori yang dipilih. Dengan demikian, `ViewAllAdapter` memfasilitasi pengelolaan data dan presentasi yang efisien dalam konteks tampilan kategori yang lebih lengkap.

7) Product

- activity: ProductsFragment.java



```
ProductFragment.java × NavProductAdapter.java × NavProductModel.java × fragment_nav_products.xml × design_nav_products_recyclerview.xml ×
36     public View onCreateView(@NonNull LayoutInflater inflater,
37             ViewGroup container, Bundle savedInstanceState) {
38         binding = FragmentNavProductsBinding.inflate(inflater, container, attachToParent: false);
39         View root = binding.getRoot();
40         /**FIREBASE & RECYCLEVIEW*/
41         db = FirebaseFirestore.getInstance();
42         recyclerView = root.findViewById(R.id.pop_cat);
43         //Category Products
44         recyclerView.setLayoutManager(new LinearLayoutManager(getActivity(), RecyclerView.VERTICAL, reverseLayout: false));
45         categoryModelList = new ArrayList<>();
46         navProductAdapter = new NavProductAdapter(getActivity(), categoryModelList);
47         recyclerView.setAdapter(navProductAdapter);
```

Gambar 4.1.3.36 Product Frgament

Dalam kode ini, dalam fragmen, tata letak ditebang menggunakan pengikat data. Firebase Firestore dimulai, dan RecyclerView dikonfigurasi untuk menampilkan produk kategori secara vertikal. Kode mengatur adaptor, mengasosiasikannya dengan RecyclerView, dan mempersiapkan untuk mengisi data dari database Firestore untuk render UI dinamis.

```
/**POPULAR PRODUCTS - READ*/
db.collection( collectionPath: "NavProducts" ) CollectionReference
    .get() Task<QuerySnapshot>
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {

                    NavProductModel navProductModel = document.toObject(NavProductModel.class);
                    categoryModelList.add(navProductModel);
                    navProductAdapter.notifyDataSetChanged();
                } } else {
                    /**TOAST**/
                    Toast.makeText(getActivity(), text: "Error"+task.getException(), Toast.LENGTH_SHORT).show();
                }}});return root;
    }
    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

Gambar 4.1.3.37 Nav Product

Kode ini untuk mengambil produk populer dari koleksi Firestore ("NavProducts"). Ini menggunakan 'addOnCompleteListener' untuk menangani respons asynchronous, menampung daftar objek NavProductModel, dan memberi tahu adaptor RecyclerView tentang perubahan data. Dalam kasus kesalahan, ia menampilkan pesan toast pendek. Kode mengelola siklus hidup tampilan dengan tepat.

- **FireStore:**

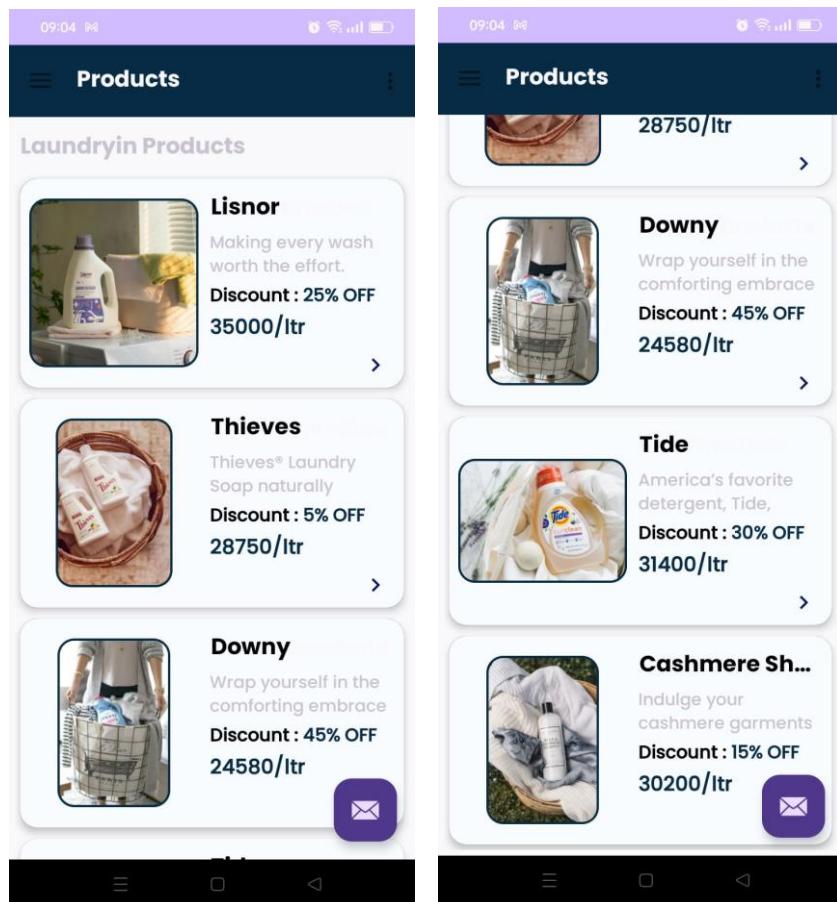
Collection	Document ID	Preview	Details
AllProducts	55JJ58SmZrX4bqDYj7ax	Thumbnail	
CurrentUser	I9WesOZYxpRaxowSU7Du	Thumbnail	
NavCategoryLaundry	S2qpWwN15hWmIwCJhosY	Thumbnail	
NavProducts	ZzFs905E7yIq7R0fc5CW	Thumbnail	
PopularProducts	k1kzoE7XztAC004uTk8	Thumbnail	

Selected Document Details:

- name:** "Thieves"
- price:** 28750
- type:** "products"
- img_url:** https://firebasestorage.googleapis.com/v0/b/laundryiiingroup2.appspot.com/o/popularproducts%2Fthieves.jpg?alt=media&token=047e9493-e177-4cca-9e56-7f105e82fe2b
- description:** "Thieves® Laundry Soap naturally washes your clothes, cleaning them without any chemical or synthetic residue and the combined power of essential oils."
- discount:** "5% OFF"

Gambar 4.1.3.38 Firestore

- Layout: fragment_nay_products



Gambar 4.1.3.39 Layout: fragment_nay_products

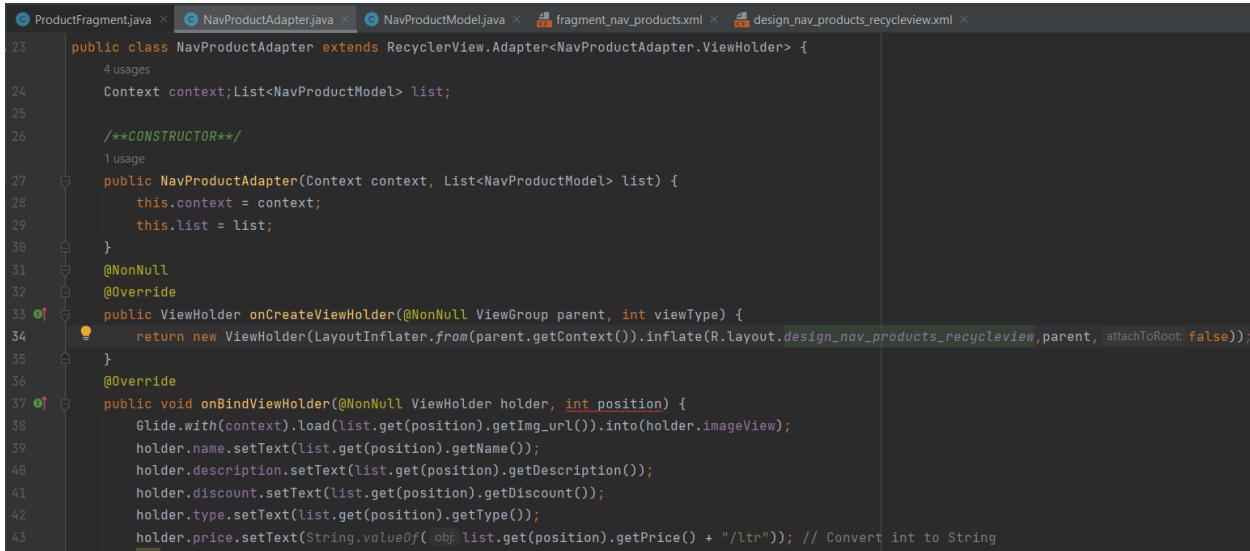
- RecycleViewer design : design_nav_products_recyclerview.xml



Gambar 4.1.3.40 Recycle Viewer Design

Data diambil dengan adanya adapters dan model yang menciptakan constructor & Getter and Setter, sehingga data yang diambil dari FireStore (get), di tampilkan ke RecycleView sesuai data yang ada (set).

- Adapters : NavProductAdapter



The screenshot shows the Android Studio code editor with the NavProductAdapter.java file open. The code defines a RecyclerView.Adapter for displaying products. It includes a constructor that takes a Context and a List of NavProductModel. The onCreateViewHolder method inflates a View from a layout XML file named design_nav_products_recyclerview.xml. The onBindViewHolder method binds data to a ViewHolder, using Glide to load an image from a URL and setting text for product name, description, discount, type, and price.

```
23 public class NavProductAdapter extends RecyclerView.Adapter<NavProductAdapter.ViewHolder> {
24     4 usages
25     Context context;List<NavProductModel> list;
26
27     /**CONSTRUCTOR*/
28     1 usage
29     public NavProductAdapter(Context context, List<NavProductModel> list) {
30         this.context = context;
31         this.list = list;
32     }
33     @NonNull
34     @Override
35     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
36         return new ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.design_nav_products_recyclerview,parent, attachToRoot: false));
37     }
38     @Override
39     public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
40         Glide.with(context).load(list.get(position).getImg_url()).into(holder.imageView);
41         holder.name.setText(list.get(position).getName());
42         holder.description.setText(list.get(position).getDescription());
43         holder.discount.setText(list.get(position).getDiscount());
44         holder.type.setText(list.get(position).getType());
45         holder.price.setText(String.valueOf( obj: list.get(position).getPrice() + "/ltr")); // Convert int to String
46     }
47 }
```

Gambar 4.1.3.41 Adapter Nav Product

Kode ini untuk mendefinisikan adaptor RecyclerView (NavProductAdapter) untuk menampilkan item NavProduktModel. Adaptor memiliki konstruktor untuk memulai konteks dan daftar produk. Ini membengkak tata letak item, mengikat data ke tampilan dalam metode 'onBindViewHolder', menggunakan Glide untuk muat gambar. Kode ini menangani penciptaan tampilan dan pengikat data dalam RecyclerView.

```

/**RecycleView Click*/
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Open DetailedProductsActivity when the item is clicked
        Intent intent = new Intent(context, DetailedProductsActivity.class);
        intent.putExtra("name", list.get(position));
        context.startActivity(intent);
    }
});
/**Metode getItemCount() memberikan informasi tentang jumlah item dalam daftar yang akan ditampilkan.*/
@Override
public int getItemCount() { return list.size(); }
4 usages
public class ViewHolder extends RecyclerView.ViewHolder {
    /**DECLARE & INITIALIZING*/
2 usages
ImageView imageView; TextView name, description, discount, price, rating, type;
1 usage
public ViewHolder(@NonNull View itemView) {
    super(itemView);
    imageView = itemView.findViewById(R.id.cat_nav_img);
    name = itemView.findViewById(R.id.nav_cat_name);
    description = itemView.findViewById(R.id.nav_cat_desc);
    discount = itemView.findViewById(R.id.nav_cat_disc);
    price = itemView.findViewById(R.id.nav_cat_price);
    type = itemView.findViewById(R.id.nav_cat_type);
}

```

Gambar 4.1.3.42 Detailed Product Activity

Kode Java ini memperluas adaptor RecyclerView untuk menangani klik item. Ketika sebuah item diklik, itu memicu niat untuk membuka DetailProductsActivity, melewati rincian produk. Kelas ViewHolder menyatakan dan memulai tampilan untuk informasi produk. Metode getItemCount mengembalikan ukuran daftar produk untuk ditampilkan di RecyclerView.

- **Model: NavProductModel**

```

/**Getter & Setter*/
public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }
1 usage
public String getDiscount() { return discount; }
no usages
public void setDiscount(String discount) { this.discount = discount; }
public String getImg_url() { return img_url; }
no usages
public void setImg_url(String img_url) { this.img_url = img_url; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public int getPrice() { return price; }
public void setPrice(int price) { this.price = price; }
no usages
public String getRating() { return rating; }
no usages
public void setRating(String rating) { this.rating = rating; }
public String getType() { return type; }
public void setType(String type) { this.type = type; }

```

Gambar 4.1.3.43 Nav Product Model

Kode ini untuk mendefinisikan metode getter dan setter untuk kelas, yang mungkin mewakili rincian produk. Setiap metode sesuai dengan atribut tertentu seperti nama, deskripsi, diskon, URL gambar, harga, peringkat, dan jenis. Metode ini memfasilitasi akses dan modifikasi atribut instansi kelas.

```
11 usages
public class NavProductModel implements Serializable {
    /**Declaring**/
    3 usages
    String description;String discount;String img_url;String name;int price;String rating;String type;
    /**Constructor**/
    no usages
    public NavProductModel() {}
    no usages
    public NavProductModel(String description, String discount, String img_url, String name, int price, String rating, String type) {
        this.description = description;
        this.discount = discount;
        this.img_url = img_url;
        this.name = name;
        this.price = price;
        this.rating = rating;
        this.type = type;
    }
}
```

Gambar 4.1.3.44 Nav Product Model

Kode ini digunakan untuk mendefinisikan kelas 'NavProductModel' yang menerapkan Serializable, yang mewakili model untuk rincian produk navigasi. Ini termasuk atribut seperti deskripsi, diskon, URL gambar, nama, harga, peringkat, dan jenis. Kelas ini memiliki konstruktor untuk instantiasi, dan metode getter dan setter untuk mengakses dan memodifikasi atributnya.

- Intent to Checkout Process:

```
/**RecycleView Click**/
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Open DetailedProductsActivity when the item is clicked
        Intent intent = new Intent(context, DetailedProductsActivity.class);
        intent.putExtra("name", "detail", list.get(position));
        context.startActivity(intent);
    }
});
```

Gambar 4.1.3.45 Intent Checkout Process

Kode ini adalah bagian dari adaptor RecyclerView. Ketika sebuah item di RecyclerView diklik (`itemView.setOnClickListener`), itu memicu peristiwa `onClick`. Kode menciptakan niat untuk membuka DetailProductsActivity, melewati rincian produk melalui niat tambahan.

Akhirnya, ia memulai aktivitas untuk menampilkan informasi rinci tentang produk yang dipilih.

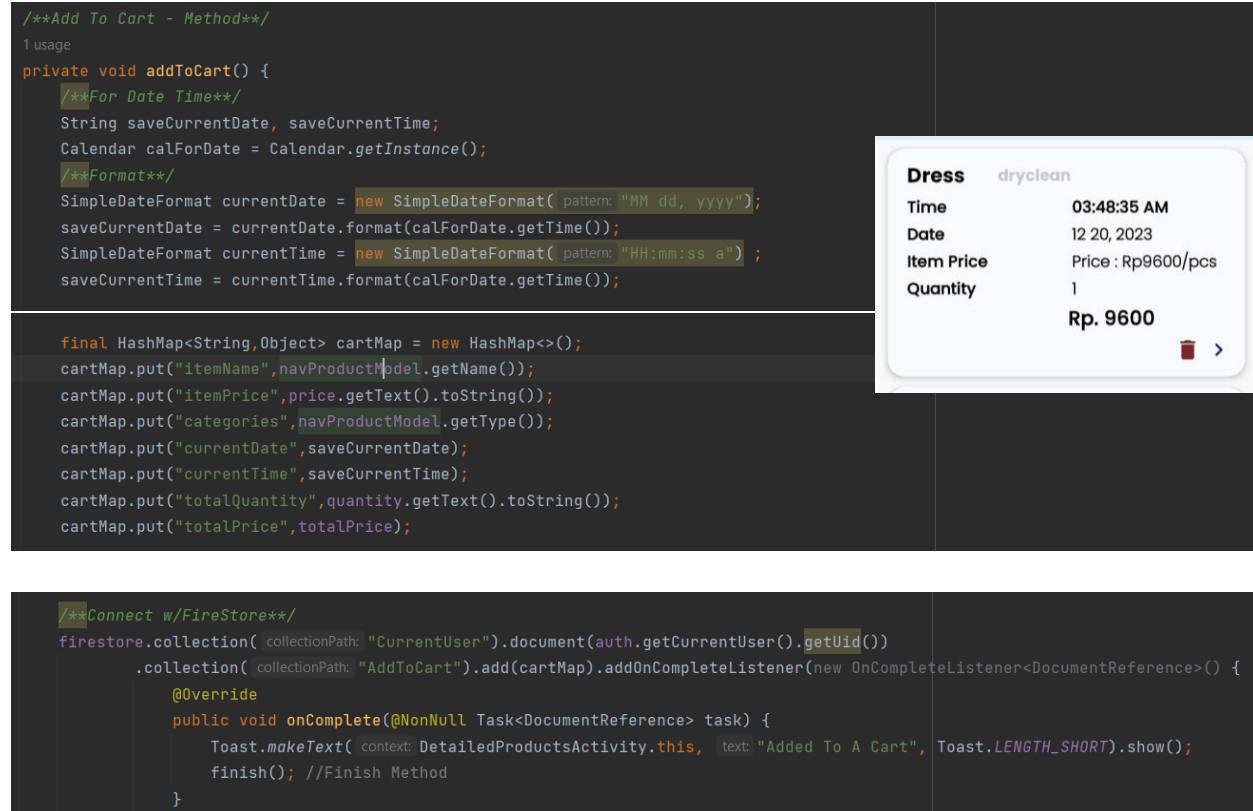
8) Fragment Carts

-> activity : nav_Carts.java
-> adapters : MyCartAdapter
-> model : MyCartModel

RecycleViewer design : design_nav_cart_items.xml

FROM Add to Cart Method on Catgeoreis Laundry and Products :

- DetailedProductsActivity.java



```

/**Add To Cart - Method*/
1 usage
private void addToCart() {
    /**For Date Time**/
    String saveCurrentDate, savecurrentTime;
    Calendar calForDate = Calendar.getInstance();
    /**Format**/
    SimpleDateFormat currentDate = new SimpleDateFormat( pattern: "MM dd, yyyy");
    saveCurrentDate = currentDate.format(calForDate.getTime());
    SimpleDateFormat currentTime = new SimpleDateFormat( pattern: "HH:mm:ss a");
    savecurrentTime = currentTime.format(calForDate.getTime());

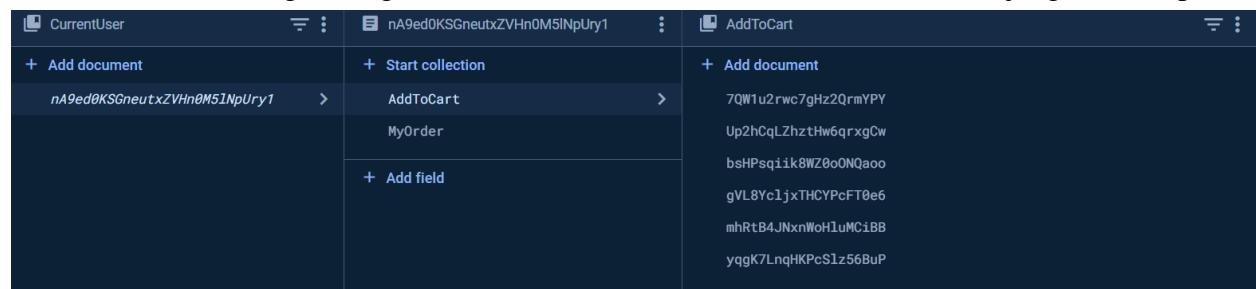
    final HashMap<String, Object> cartMap = new HashMap<>();
    cartMap.put("itemName", navProductModel.getName());
    cartMap.put("itemPrice", price.getText().toString());
    cartMap.put("categories", navProductModel.getType());
    cartMap.put("currentDate", saveCurrentDate);
    cartMap.put("currentTime", savecurrentTime);
    cartMap.put("totalQuantity", quantity.getText().toString());
    cartMap.put("totalPrice", totalPrice);
}

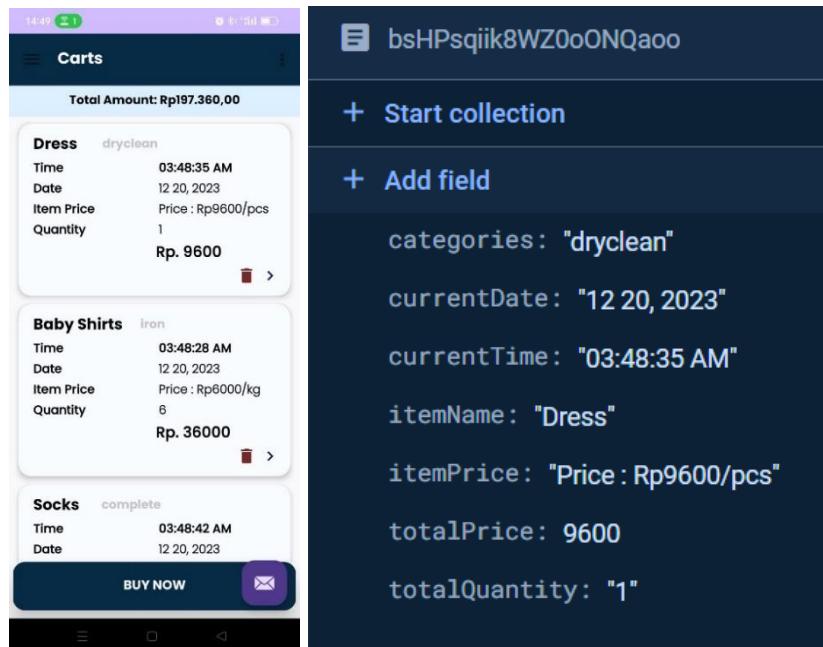
/**Connect w/FireStore**/
firestore.collection( collectionPath: "CurrentUser").document(auth.getCurrentUser().getUid())
    .collection( collectionPath: "AddToCart").add(cartMap).addOnCompleteListener<DocumentReference>() {
    @Override
    public void onComplete(@NonNull Task<DocumentReference> task) {
        Toast.makeText( context: DetailedProductsActivity.this, text: "Added To A Cart", Toast.LENGTH_SHORT).show();
        finish(); //Finish Method
    }
}


```

Gambar 4.1.3.46 Fragment Carts

Kode Java ini mendefinisikan metode "addToCart" untuk aplikasi e-commerce. Ini menangkap tanggal dan waktu saat ini, membuat HashMap dengan rincian produk, dan menyimpannya di Firestore. Kode menghubungkan ke Firestore, menambahkan rincian keranjang, menampilkan





Gambar 4.1.3.47 Database Carts

- **DetailedCategoriesActivity.java**

```

/**Add To Cart - Method*/
1 usage
private void addToCart() {
    /**For Date Time*/
    String saveCurrentDate, savecurrentTime;
    Calendar calForDate = Calendar.getInstance();
    /**Format*/
    SimpleDateFormat currentDate = new SimpleDateFormat( pattern: "MM dd, yyyy")
    saveCurrentDate = currentDate.format(calForDate.getTime());
    SimpleDateFormat currentTime = new SimpleDateFormat( pattern: "HH:mm:ss a");
    savecurrentTime = currentTime.format(calForDate.getTime());

    final HashMap<String, Object> cartMap = new HashMap<>();
    cartMap.put("itemName", viewAllModel.getName());
    cartMap.put("itemPrice", price.getText().toString());
    cartMap.put("categories", viewAllModel.getType());
    cartMap.put("currentDate", saveCurrentDate);
    cartMap.put("currentTime", savecurrentTime);
    cartMap.put("totalQuantity", quantity.getText().toString());
    cartMap.put("totalPrice", totalPrice);

    /**Connect w/FireStore*/
    firestore.collection( collectionPath: "CurrentUser").document(auth.getCurrentUser().getUid())
        .collection( collectionPath: "AddToCart").add(cartMap).addOnCompleteListener<DocumentReference>() {
            @Override
            public void onComplete(@NotNull Task<DocumentReference> task) {
                Toast.makeText( context: DetailedCategoriesActivity.this, text: "Added To A Cart", Toast.LENGTH_SHORT).show();
                finish(); //Finish Method
            }
        });
}


```

The screenshot shows a mobile application interface for a shopping cart. The cart contains three items: Downy products. Each item has details like time, date, price, and quantity. To the right, a database interface shows the same data as documents in a collection named 'AddToCart'.

Gambar 4.1.3.48 Detailed Categories

This Java code defines an "addToCart" method for an e-commerce app. It captures the current date and time, creates a HashMap with product details, and stores it in Firestore. The code connects to Firestore, adds the cart details, displays a toast message confirming the addition, and finishes the activity upon successful completion.

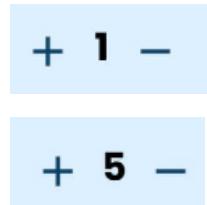
+ Start collection	+ Add document	+ Start collection
AddToCart > MyOrder + Add field	7QW1u2rwc7gHz2QrmYPY > Up2hCqLZhztHw6qrwgCw bsHPsqiik8WZ0oONQao0 gVL8YcljxTHCYPcFT0e6 mhRtB4JNxnWoHluMCiBB yqgK7LnqHKPcSlz56BuP	+ Add field categories: "products" currentDate: "12 20, 2023" currentTime: "03:48:15 AM" itemName: "Cashmere Shampoo" itemPrice: "Price : Rp. 30200" totalPrice: 30200 totalQuantity: "1"

Gambar 4.1.3.49 Database Detailed Categories

- Plus minus

```

if(navProductModel != null){
    Glide.with(getApplicationContext()).load(navProductModel.getImg_url()).into(detailedImg);
    desc.setText(navProductModel.getDescription());
    price.setText("Price : Rp. "+navProductModel.getPrice());
    /*COUNT TOTAL PRICE*/
    totalPrice = navProductModel.getPrice() * totalQuantity;
}
/**Add Qty**/
addItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (totalQuantity < 10) {
            totalQuantity++;
            quantity.setText(String.valueOf(totalQuantity));
            totalPrice = navProductModel.getPrice() * totalQuantity;
        }});
}
/**Remove Qty**/
removeItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (totalQuantity > 1) {
            totalQuantity--;
            quantity.setText(String.valueOf(totalQuantity));
            totalPrice = navProductModel.getPrice() * totalQuantity;
        }});
}
  
```



Gambar 4.1.3.50 Tombol Plus Minus

Kode Java ini menangani tampilan rincian produk menggunakan Glide untuk muat gambar. Ini memperbarui kuantitas secara dinamis, menyesuaikan harga total sesuai, dengan batas yang ditetapkan antara 1 dan 10 item. Tombol “addItem” dan “removeItem” mengubah jumlah, dan total harga dihitung ulang berdasarkan jumlah yang dipilih dan harga produk

- activity : nav_Carts.java

```


/**Total**/
overTotalAmount = root.findViewById(R.id.textview_totalprice);
/**Button Buy**/
buyNow = root.findViewById(R.id.buy_now);
buyNow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getContext(), PlacedOrderActivity.class);
        intent.putExtra("name", (Serializable) cartModelList);
        startActivity(intent);
    }
});
/**Connecting the Recycleview with database**/
db.collection("CurrentUser").document(auth.getCurrentUser().getUid())
    .collection("AddToCart").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
@Override
public void onComplete(@NonNull Task<QuerySnapshot> task) {
    if (task.isSuccessful()){
        for (DocumentSnapshot documentSnapshot : task.getResult().getDocuments()){

            String documentId = documentSnapshot.getId();
            MyCartModel cartModel = documentSnapshot.toObject(MyCartModel.class);
            cartModel.setDocumentId(documentId);
            cartModelList.add(cartModel);
            cartAdapter.notifyDataSetChanged();
            recyclerView.setVisibility(View.VISIBLE);
        }
        /**CALLING METHOD**/
        calculateTotalAmount(cartModelList);
    }
}
/**METHOD**/
1 usage
private void calculateTotalAmount(List<MyCartModel> cartModelList) {
    double totalAmount = 0.0;
    for (MyCartModel myCartModel : cartModelList) {
        totalAmount += myCartModel.getTotalPrice();
    }
    // Format totalAmount to rupiah format
    NumberFormat formatter = NumberFormat.getCurrencyInstance(new Locale("id", "ID"));
    String formattedTotalAmount = formatter.format(totalAmount);
    overTotalAmount.setText("Total Amount: " + formattedTotalAmount);
}


```



Downy products	
Time	03:47:44 AM
Date	12 20, 2023
Item Price	Price : Rp. 24580
Quantity	2
Rp. 49160	



Gambar 4.1.3.51 Nav Carts

Kode ini digunakan untuk menghitung jumlah total item dalam keranjang belanja, mendapatkan data keranjang dari Firestore, dan memperbarui RecyclerView sesuai. Ini juga memformat jumlah total menjadi Rupiah Indonesia dan menampilkan itu. Dengan mengklik tombol "Beli Sekarang", ia memulai sebuah niat ke PlacedOrderActivity, melewati data keranjang serialisasi.

- adapters : MyCartAdapter

```

@NonNull
@Override
public MyCartAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    return new ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.design_nav_cart_items, parent, attachToRoot: false));
}
@Override
public void onBindViewHolder(@NonNull MyCartAdapter.ViewHolder holder, int position) {
    /**GET SET**/
    holder.name.setText(cartModelList.get(position).getItemName());
    holder.category.setText(cartModelList.get(position).getCategories());
    holder.price.setText(cartModelList.get(position).getItemPrice());
    holder.date.setText(cartModelList.get(position).getCurrentDate());
    holder.time.setText(cartModelList.get(position).getCurrentTime());
    holder.quantity.setText(cartModelList.get(position).getTotalQuantity());
    holder.totalPrice.setText(String.valueOf(cartModelList.get(position).getTotalPrice()));
}

/**DELETE BUTTON**/
holder.deleteItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        firestore.collection("CurrentUser").document(auth.getCurrentUser().getUid())
            .collection("AddToCart")
            .document(cartModelList.get(position).getDocumentId())
            .delete()
            .addOnCompleteListener(new OnCompleteListener() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful()){
                        cartModelList.remove(cartModelList.get(position));
                        notifyDataSetChanged();
                        Toast.makeText(context, "Item Deleted", Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(context, "Error"+task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                }
            });
    }
}

/**Metode getItemCount() memberikan informasi tentang jumlah item dalam daftar yang akan ditampilkan.*/
@Override
public int getItemCount() { return cartModelList.size(); }

```

```

2 usages
public class ViewHolder extends RecyclerView.ViewHolder {
    /**DECLARING**/
    2 usages
    TextView name, category, price, date, time, quantity, totalPrice; ImageView deleteItem;
    1 usage
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        /**INITIALIZING**/
        name = itemView.findViewById(R.id.item_name);
        category = itemView.findViewById(R.id.item_category);
        price = itemView.findViewById(R.id.item_Price);
        date = itemView.findViewById(R.id.current_Date);
        time = itemView.findViewById(R.id.current_Time);
        quantity = itemView.findViewById(R.id.item_QTY);
        totalPrice = itemView.findViewById(R.id.Total_Price);
        deleteItem = itemView.findViewById(R.id.delete);
    }
}

```

Gambar 4.1.3.52 My Carts Adapter

Kode Java ini mendefinisikan adaptor RecyclerView untuk keranjang belanja dalam aplikasi Android. Ini membebani tampilan item, mengikat data dari daftar item keranjang, dan menangani penghapusan item. Kelas ViewHolder memulai dan mengumumkan elemen UI. Tombol Hapus menghapus item dari Firestore dan memperbarui RecyclerView sesuai.

- Model : MyCartModel

```

public class MyCartModel implements Serializable {
    /**Declaring**/
    3 usages
    String itemName;String itemPrice;String currentDate;String currentTime;String totalQuantity;String categories;int totalPrice;String documentId;
    /**Constructor**/
    no usages
    public MyCartModel() {}
    no usages
    public MyCartModel(String documentId) { this.documentId = documentId; }
    no usages
    public MyCartModel(String itemName, String itemPrice, String currentDate, String currentTime, String totalQuantity, String categories, int totalPrice) {
        this.itemName = itemName;
        this.itemPrice = itemPrice;
        this.currentDate = currentDate;
        this.currentTime = currentTime;
        this.totalQuantity = totalQuantity;
        this.categories = categories;
        this.totalPrice = totalPrice;
    }
    /**
     * Getter & Setter*/
    1 usage
    public String getDocumentId() { return documentId; }
    1 usage
    public void setDocumentId(String documentId) { this.documentId = documentId; }
    2 usages
    public String getItemName() { return itemName; }
    no usages
    public void setItemName(String itemName) { this.itemName = itemName; }
    2 usages
    public String getItemPrice() { return itemPrice; }
    no usages
    public void setItemPrice(String itemPrice) {this.itemPrice = itemPrice;}
    2 usages
    public String getCurrentDate() {return currentDate;}
    no usages
    public void setCurrentDate(String currentDate) {this.currentDate = currentDate;}
    2 usages
    public String getCurrentTime() {return currentTime;}
    no usages
    public void setCurrentTime(String currentTime) {this.currentTime = currentTime;}
    2 usages
    public String getTotalQuantity() {return totalQuantity;}
    no usages
    public void setTotalQuantity(String totalQuantity) {this.totalQuantity = totalQuantity;}
    2 usages
    public String getCategories() {return categories;}
    no usages
    public void setCategories(String categories) {this.categories = categories;}
    3 usages
    public int getTotalPrice() {return totalPrice;}
}

```

Gambar 4.1.3.53 My Carts Model

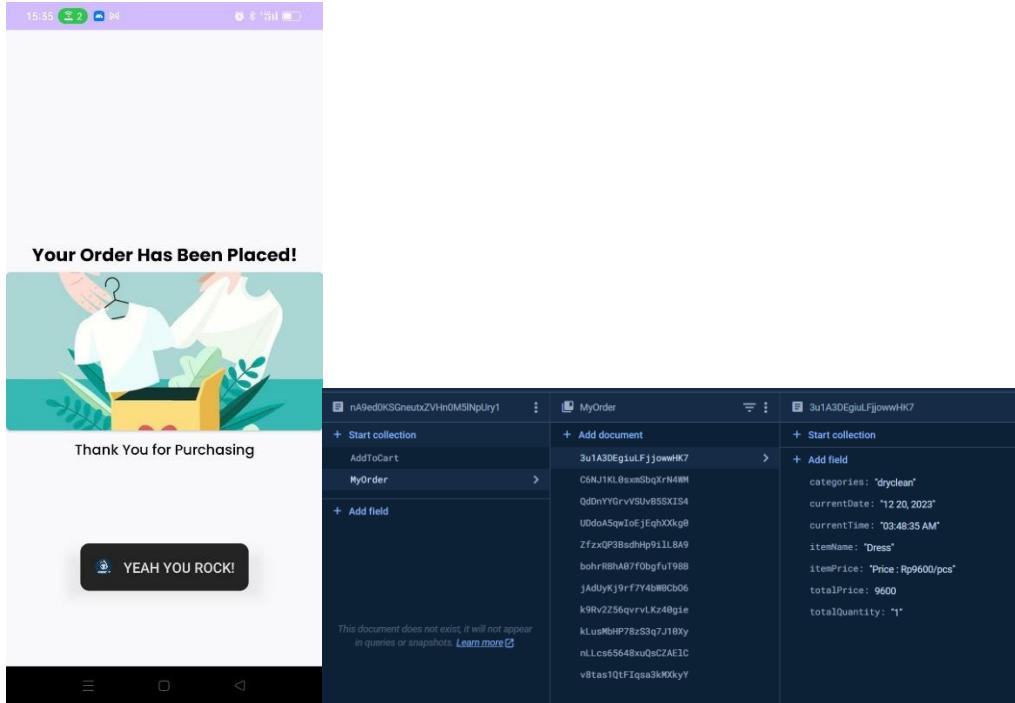
Kode Java ini mendefinisikan kelas model, 'MyCartModel', untuk mewakili item dalam keranjang belanja. Ini mengimplementasikan antarmuka Serializable untuk transfer data. Kelas ini memiliki atribut seperti itemName, itemPrice, currentDate, dan metode untuk mengatur dan mendapatkan attribut ini. Ini juga mencakup konstruktor untuk instansiasi dan manipulasi item keranjang.

- **Checkout**

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_placed_order);  
    /***FIREBASE*/  
    auth = FirebaseAuth.getInstance();  
    firestore = FirebaseFirestore.getInstance();  
    /**ARRAY RECYCLEVIEW**/  
    List<MyCartModel> list = (ArrayList<MyCartModel>) getIntent().getSerializableExtra("itemList");  
    if ((list != null && list.size() > 0)) {  
        for (MyCartModel model : list){  
            final HashMap<String, Object> cartMap = new HashMap<>();  
            cartMap.put("itemName",model.getItemName());  
            cartMap.put("itemPrice",model.getItemPrice());  
            cartMap.put("categories",model.getCategories());  
            cartMap.put("currentDate",model.getCurrentDate());  
            cartMap.put("currentTime",model.getCurrentTime());  
            cartMap.put("totalQuantity",model.getTotalQuantity());  
            cartMap.put("totalPrice",model.getTotalPrice());  
            /**Connect w/FireStore ADD TO My Order database**/  
            firestore.collection("CurrentUser").document(auth.getCurrentUser().getUid())  
                .collection("MyOrder").add(cartMap).addOnCompleteListener(new OnCompleteListener<DocumentReference>() {  
                    @Override  
                    public void onComplete(@NonNull Task<DocumentReference> task) {  
                        Toast.makeText(context, PlacedOrderActivity.this, "YEAH YOU ROCK!", Toast.LENGTH_SHORT).show();  
                    }  
                });  
        }  
    }  
}
```

Gambar 4.1.3.54 Checkout

Kode ini digunakan untuk menangani penciptaan "PlacedOrderActivity", mentransfer data keranjang yang diterima melalui Intent, dan menambahkan ke database Firestore sebagai pesanan. Ini berterusan melalui item keranjang, membuat HashMap, dan terhubung dengan Firestore untuk menyimpan rincian pesanan. Sebuah pesan toast mengkonfirmasi penempatan pesanan yang sukses.



Gambar 4.1.3.55 Alert Checkout

9) View pager (about company) fragment

Viewpager terhubung dengan 3 layout fragment yaitu:

- Activity: nav_ComProfile.java

```

fragment_nav_com_profile.xml x nav_ComProfile.java x MyPagerAdapter.java x
2 usages
19 public class nav_ComProfile extends Fragment {
20     private ViewPager viewPager;
21     private MyPagerAdapter adapter;
22
23     @Override
24     public View onCreateView(LayoutInflater inflater, ViewGroup container,
25                             Bundle savedInstanceState) {
26         // Inflate the layout for this fragment
27         View view = inflater.inflate(R.layout.fragment_nav_com_profile, container, false);
28
29         /**
30          * Fragment yang menampilkan halaman profil perusahaan dengan menggunakan ViewPager.
31          * Setiap halaman diwakili oleh fragmen yang berbeda (fragment_frag_pro_com1, fragment_frag_pro_com2, fragment_frag_pro_com3).
32          */
33
34         viewPager = view.findViewById(R.id.viewPager);
35         List<Fragment> fragments = new ArrayList<>();
36         fragments.add(new fragment_frag_pro_com1()); /*ABOUT LAUNDRYIN*/
37         fragments.add(new fragment_frag_pro_com2()); /*DESCRIPTION*/
38         fragments.add(new fragment_frag_pro_com3()); /*LAUNDRYIN INSTAGRAM ACCOUNT*/
39
40         /**Menginisialisasi ViewPager dan Adapter.*/
41         adapter = new MyPagerAdapter getChildFragmentManager(), fragments);
42         viewPager.setAdapter(adapter);
43
44         return view;

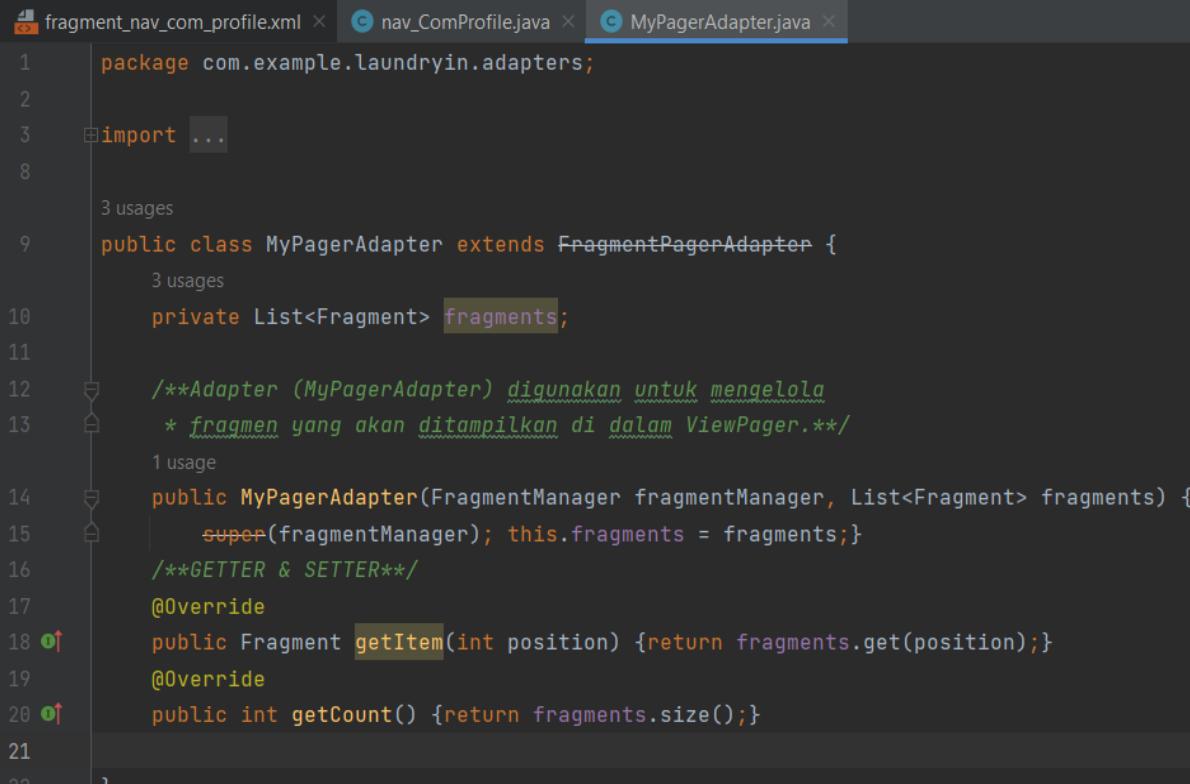
```

Gambar 4.1.3.56 View Pager (About Company)

Penjelasan:

- ✓ fragments.add(new fragment_frag_pro_com1()); -> "About Us"
- ✓ fragments.add(new fragment_frag_pro_com2()); -> "Description about LaundryIn"
- ✓ fragments.add(new fragment_frag_pro_com3()); -> "Website akun instagram LaundryIn"

• Adapter : MyPagerAdapter.java

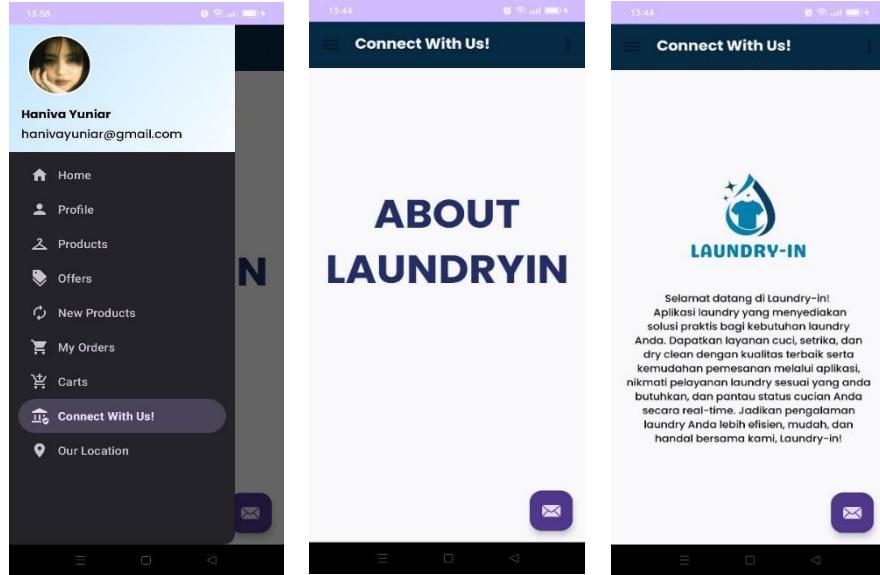


```
1 package com.example.laundryin.adapters;
2
3 import ...
4
5 3 usages
6
7 public class MyPagerAdapter extends FragmentPagerAdapter {
8     3 usages
9     private List<Fragment> fragments;
10
11
12     /**
13      * Adapter (MyPagerAdapter) digunakan untuk mengelola
14      * fragmen yang akan ditampilkan di dalam ViewPager.
15      */
16
17     public MyPagerAdapter(FragmentManager fragmentManager, List<Fragment> fragments) {
18         super(fragmentManager); this.fragments = fragments;
19     }
20
21     /**
22      * GETTER & SETTER
23      */
24
25     @Override
26     public Fragment getItem(int position) {return fragments.get(position);}
27
28     @Override
29     public int getCount() {return fragments.size();}
30
31 }
```

Gambar 4.1.3.57 View Pager (About Company)

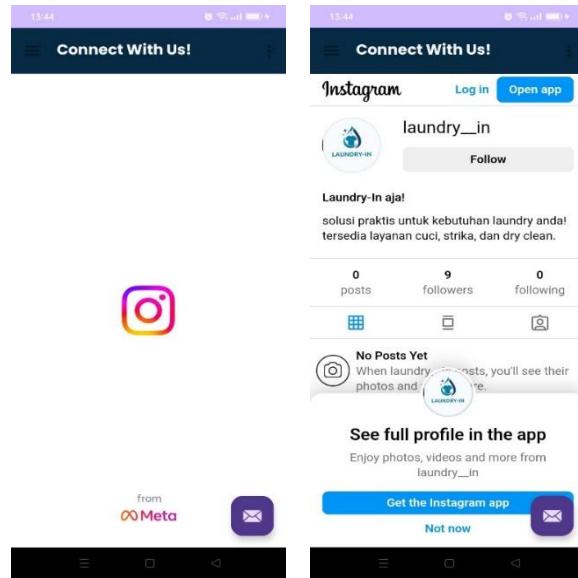
- **Output**

- ✓ fragments.add(new fragment_frag_pro_com1()); -> "About Us"
- ✓ fragments.add(new fragment_frag_pro_com2()); -> "Description about LaundryIn"



Gambar 4.1.3.58 Layout View Pager (About Company)

- fragments.add(new fragment_frag_pro_com3()); -> "Website akun instagram LaundryIn"
- Akan loading terlebih dahulu, kemudian ditampilkan:



Gambar 4.1.3.59 Instagram Laundryin

10) Layout Maps Location (Fragment Maps)

- API Key pada Manifest

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="AIzaSyBq8Nr_gLDvm14-Usojs0hfJvYo7lkqog8" />
```

Gambar 4.1.3.60 API Key Maps

- Import

```
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.Marker;  
import com.google.android.gms.maps.model.MarkerOptions;
```

Gambar 4.1.3.61 Import Maps

- Inflate Layout, Declaring dan Initializing

```
/**Fragment to display a map with LaundryIn location and map type buttons.**/  
3 usages  
public class nav_Maps extends Fragment {  
  
    /**DECLARE & INITIALIZING**/  
    7 usages  
    private GoogleMap gMap;  
    /**LANDRYIN LOCATION**/  
    2 usages  
    private final LatLng laundryInLocation = new LatLng( latitude: -6.2947360, longitude: 106.6841978);  
    1 usage  
    private static final float DEFAULT_ZOOM = 18;  
  
    @Nullable  
    @Override  
    public View onCreateView(@NonNull LayoutInflater inflater,  
                            @Nullable ViewGroup container,  
                            @Nullable Bundle savedInstanceState) {  
        /**Inflate the layout for this fragment**/  
        return inflater.inflate(R.layout.fragment_nav_maps, container, attachToRoot: false);  
    }  
}
```

Gambar 4.1.3.62 Declaring

- Initialize Google Maps when OnReady

```

@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    /**Fragment Map Support*/
    SupportMapFragment mapFragment =
        (SupportMapFragment) getChildFragmentManager().findFragmentById(R.id.map);
    if (mapFragment != null) {
        /**Asynchronously initializes the GoogleMap and calls onMapReady when ready*/
        mapFragment.getMapAsync(new OnMapReadyCallback() {
            no usages
            @Override
            public void onMapReady(GoogleMap googleMap) {
                gMap = googleMap;
                setupMap();
            }
        });
    }
}

```

Gambar 4.1.3.63 Initializing

- Tombol Declare, Initializing, dan Set on Click

```

/** Find buttons by ID and set click listeners */
Button btnNormalMode = view.findViewById(R.id.btnNormalMode);
Button btnTerrainMode = view.findViewById(R.id.btnTerrainMode);
Button btnHybrid = view.findViewById(R.id.btnHybrid);
Button btnSatelliteMode = view.findViewById(R.id.btnSatelliteMode);

```

Gambar 4.1.3.64 Tombol Declare Initializing dan Set on Click

- **Normal dan Terrain**

```

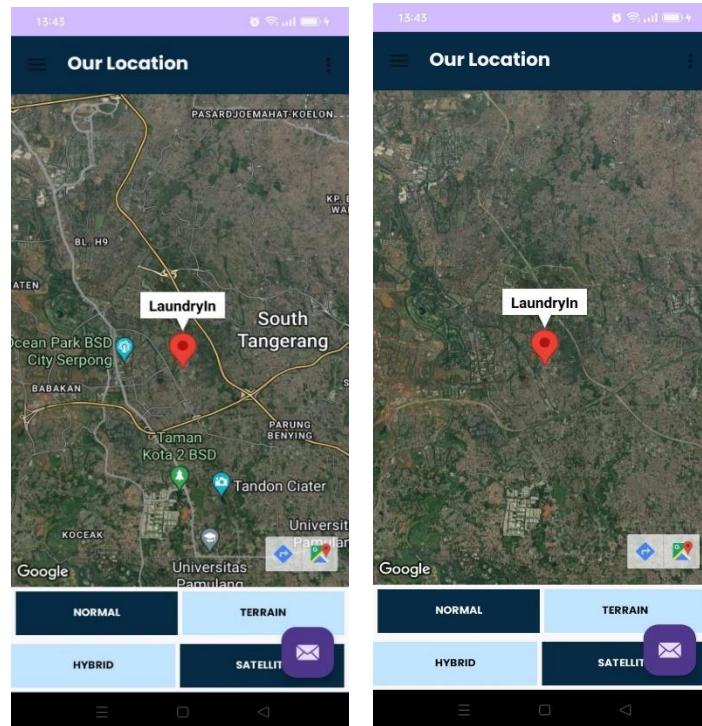
/**NORMAL BUTTON**/
btnNormalMode.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        changeMapType(GoogleMap.MAP_TYPE_NORMAL);
    }
});

/**TERRAIN BUTTON**/
btnTerrainMode.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        changeMapType(GoogleMap.MAP_TYPE_TERRAIN);
    }
});
}

```

Gambar 4.1.3.65 Normal dan Terrain

- **Output**



Gambar 4.1.3.66 Layout Maps

- **Marker Setup dan Directory ke Maps App**

```
/**Set up the GoogleMap with a marker at LaundryIn location and default map type.*/
1 usage
private void setupMap() {
    // Add a marker with a title to the specified location and move the camera
    Marker laundryInMarker = gMap.addMarker(new MarkerOptions()
        .position(laundryInLocation)
        .title("LaundryIn")
        .draggable(true)); // Allow the marker to be dragged
    gMap.moveCamera(CameraUpdateFactory.newLatLngZoom(laundryInLocation, DEFAULT_ZOOM));

    /** Set up a marker click listener to show additional information*/
    gMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
        no usages
        @Override
        public boolean onMarkerClick(Marker marker) {return false;}
    });
    /**Set default map type*/
    gMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
}
```

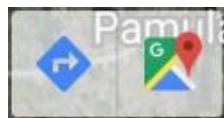
Gambar 4.1.3.67 Marker Setup dan Directory ke Maps

- **Tipe Metode Map**

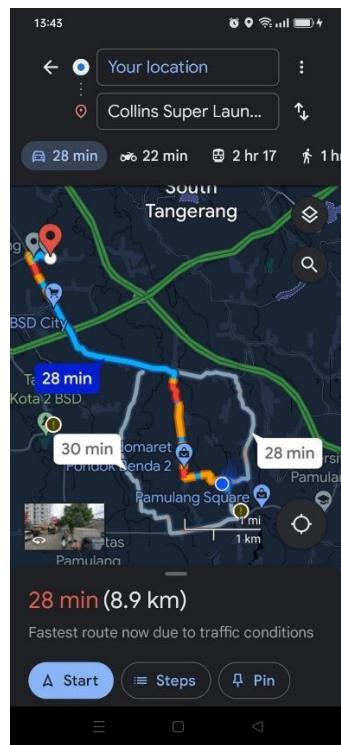
```
/**Change the map type based on the provided mapType constant.
 * @param mapType The desired map type (e.g., GoogleMap.MAP_TYPE_NORMAL).*/
4 usages
private void changeMapType(int mapType) {
    if (gMap != null) {
        gMap.setMapType(mapType);
    }
}
```

Gambar 4.1.3.68 Tipe Metode Maps

➔ Apabila di Klik:

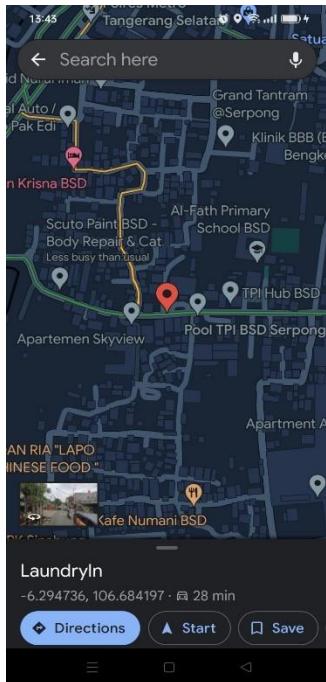


a) Route Direct from Customer device location



Gambar 4.1.3.69 Customer Device Location

b) Open Location on Google Mapsadm



Gambar 4.1.3.70 Open Location

4.2. Admin

- Tampilan Pada SQLite untuk Database Admin/Seller

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - F:\adminLaundry.db". The menu bar includes File, Edit, View, Tools, and Help. Below the menu is a toolbar with New Database, Open Database, and Write buttons. A tab bar at the top has "Database Structure", "Browse Data", and "Edit Pragmas", with "Browse Data" selected. A dropdown menu labeled "Table: user" is open, showing filter options. The main area displays a table with three columns: id, username, and password. The data is as follows:

	id	username	password
	Filter	Filter	Filter
1	1	admin	laundry...
2	2	adminlaundry	laundryin

Gambar 4.2.1 Tampilan SQLite untuk Admin

- DBHelper

```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) { super(context, "adminLaundry.db", null, 1); }

    @Override
    public void onCreate(SQLiteDatabase myDB) {
        // Buat tabel untuk menyimpan data sesi
        myDB.execSQL("CREATE TABLE session(id INTEGER PRIMARY KEY, login TEXT NOT NULL)");

        // Buat tabel untuk menyimpan data pengguna
        myDB.execSQL("CREATE TABLE user(id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT NOT NULL, password TEXT NOT NULL)");

        // Masukkan data sesi awal
        myDB.execSQL("INSERT INTO session(id, login) VALUES(1, 'kosong')");

    }
    @Override
    public void onUpgrade(SQLiteDatabase myDB, int oldVersion, int newVersion) {
        // Hapus tabel sesi dan tabel pengguna jika ada
        myDB.execSQL("DROP TABLE IF EXISTS session");
        myDB.execSQL("DROP TABLE IF EXISTS user");

        // Buat kembali tabel sesi dan tabel pengguna
        onCreate(myDB);
    }
    // Periksa apakah sesi masih aktif
    public Boolean checkSession(String sessionValues) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM session WHERE login = ?", new String[]{sessionValues});

        // Periksa apakah ada data sesi yang ditemukan
        if (cursor.getCount() > 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

Gambar 4.2.2 DBHelper

```

// Perbarui sesi
public Boolean upgradeSession(String sessionValues, int id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("login", sessionValues);

    // Perbarui data sesi
    long update = myDB.update( table: "session", contentValues, whereClause: "id="+id, whereArgs: null);

    // Periksa apakah data sesi berhasil diperbarui
    if (update == -1) {
        return false;
    } else {
        return true;
    }
}

//insert user
public Boolean insertUser(String username, String password) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("username", username);
    contentValues.put("password", password);
    long insert = myDB.insert( table: "user", nullColumnHack: null, contentValues);
    if (insert == -1) {
        return false;
    }
    else {
        return true;
    }
}

// Periksa apakah pengguna dapat masuk
public Boolean checkLogin(String username, String password) {
    SQLiteDatabase myDB = this.getReadableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "SELECT * FROM user WHERE username = ? AND password = ?", new String[]{username, password});

    // Periksa apakah ada data pengguna yang ditemukan
    if (cursor.moveToFirst()) {
        return true;
    } else {
        return false;
    }
}

```

Gambar 4.2.3 DBHelper SQLite

Berikut merupakan code dari DBHelper SQLite pada proses login dan sign up admin, DBHelper digunakan karena memberikan jalur ke database SQLite. Proses login menggunakan fungsi get_admin_by_username() untuk mendapatkan objek admin berdasarkan username yang dimasukkan oleh pengguna. Proses sign up menggunakan fungsi insert_admin() untuk memasukkan data admin baru ke tabel admins.

- **Sign Up Seller**

Code:

```

public class SSignup extends AppCompatActivity {

    DBHelper myDB;
    Button signup;
    EditText username, password, repassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ssignup);

        myDB = new DBHelper( context: this);

        username = (EditText) findViewById(R.id.edtUsername);
        password = (EditText) findViewById(R.id.edtPass);
        repassword = (EditText) findViewById(R.id.edtConf);
        signup = (Button) findViewById(R.id.btnSigUpSeller);

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String strUsername = username.getText().toString();
                String strPassword = password.getText().toString();
                String strPasswordConf = repassword.getText().toString();

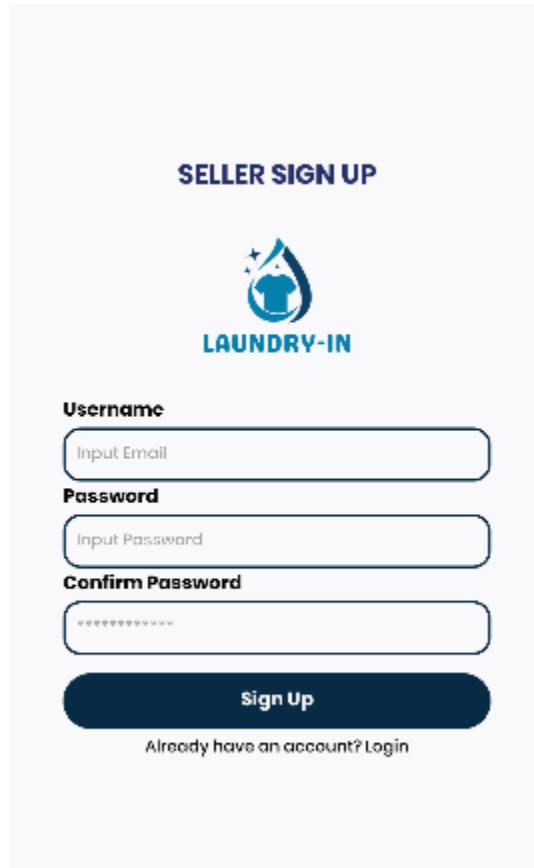
                signup.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        String strUsername = username.getText().toString();
                        String strPassword = password.getText().toString();
                        String strPasswordConf = repassword.getText().toString();

                        // Periksa apakah password dan password konfirmasi cocok
                        if (strPassword.equals(strPasswordConf)) {
                            // Daftarkan pengguna baru
                            Boolean daftar = myDB.insertUser(strUsername, strPassword);
                            if (daftar == true) {
                                Toast.makeText(getApplicationContext(), text: "Daftar Berhasil", Toast.LENGTH_SHORT).show();
                                Intent loginIntent = new Intent( packageContext SSignup.this, SLogin.class);
                                startActivity(loginIntent);
                                finish();
                            } else {
                                Toast.makeText(getApplicationContext(), text: "Daftar Gagal", Toast.LENGTH_SHORT).show();
                            }
                        } else {
                            Toast.makeText(getApplicationContext(), text: "Password Tidak Cocok", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
        });
    }
}

```

Gambar 4.2.4 Sign-Up Seller

Berikut merupakan codingan untuk sign up page, dimana seller akan membuat akun terlebih dahulu untuk bisa masuk ke main page admin, data yang diinput seller akan dikirim ke database, sehingga ketika seller ingin melakukan login sistem dapat melakukan verifikasi dan memberikan akses ke main page admin.



Gambar 4.2.5 Layout Sign-Up Seller

- **Login Admin**

Code:

```
public class SLogin extends AppCompatActivity {

    Button btnLogin;
    Button login;
    TextView txtLog;
    DBHelper myDB;
    EditText username, password;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_slogin);

        myDB = new DBHelper(context: this);

        username = (EditText) findViewById(R.id.edtMail);
        password = (EditText) findViewById(R.id.edtPass);
        login = (Button) findViewById(R.id.btnLogSeller);
    }
}
```

```

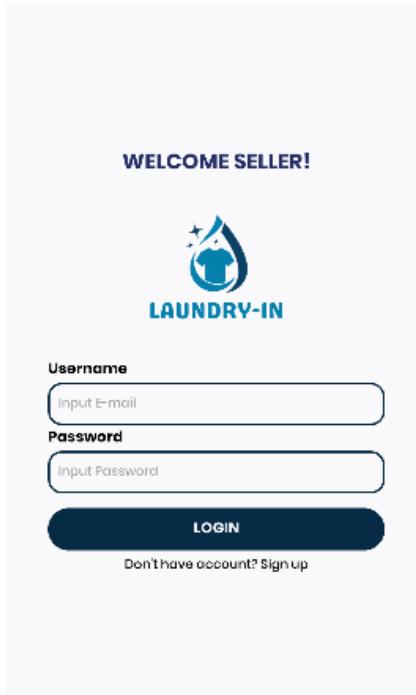
login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String strUsername = username.getText().toString();
        String strPassword = password.getText().toString();
        Boolean masuk = myDB.checkLogin(strUsername, strPassword);
        if (masuk == true) {
            Boolean updateSession = myDB.upgradeSession(sessionValues: "ada", id: 1);
            if (updateSession == true) {
                Toast.makeText(context: SLogin.this, text: "Berhasil Masuk", Toast.LENGTH_SHORT).show();
                Intent mainIntent = new Intent(packageContext: SLogin.this, AdminPage.class);
                startActivity(mainIntent);
                finish();
            }
        } else {
            Toast.makeText(getApplicationContext(), text: "Masuk Gagal", Toast.LENGTH_SHORT).show();
        }
    }
});

/**Text Intent to SELLER SignUp class*/
txtLog = findViewById(R.id.txtLog); /*INITIALIZING*/
txtLog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), SSignup.class);
        startActivity(intent);
    }
})

```

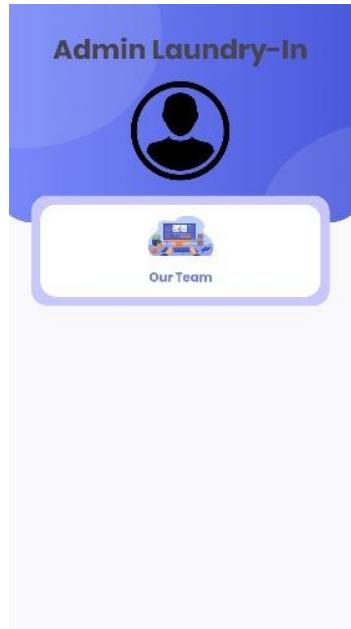
Gambar 4.2.6 Kode Log-In Seller

Berikut merupakan code dari login admin, ketika button di klik, data yang diinput akan dibaca oleh sistem dan mencocokan dengan data yang dibuat sebelumnya, apabila data cocok, seller akan berhasil masuk ke halaman utama pada aplikasi seller.



Gambar 4.2.7 Layout Log-In Seller

- **Home Page Admin**



Gambar 4.2.8 Home Page Admin

```
public class PageAdmin extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_page_admin);

        // Temukan TextView yang menunjukkan "Our Seller"
        TextView ourSellerTextView = findViewById(R.id.txtOurSeller);

        // Atur fungsi onClickListener untuk menangani klik pada teks "Our Seller"
        ourSellerTextView.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                // Arahan ke MainActivity ketika "Our Seller" diklik
                Intent intent = new Intent( packageContext PageAdmin.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

Gambar 4.2.9 Kode untuk Home Page Admin

- **Script PHP - CRUD**

- a. **Connection**

```
1 <?php
2     $servername = "localhost";
3     $database = "laundryin";
4     $username = "root";
5     $password = "";
6
7     // membuat koneksi
8     $conn = mysqli_connect($servername, $username, $password, $database);
9     mysqli_select_db($conn, $database) or die("Database belum siap");
10
11    // mengecek koneksi
12    if ($conn->connect_error) {
13        die("Koneksi gagal: " . mysqli_connect_error());
14    }
15 ?>
16
```

Gambar 4.2.10 Script PHP-CRUD

Code ini digunakan untuk melakukan koneksi ke database MySQL dengan menggunakan PHP. Pada akhir script, dapat diasumsikan bahwa jika eksekusi code ini berhasil, koneksi berarti sukses masuk ke database "laundryin" pada server MySQL yang berjalan di "localhost" dengan menggunakan username "root" dan tanpa password.

- b. **Insert Seller – Create**

```
1 <?php
2     include 'conn.php';
3
4     $Name = $_POST["Name"];
5     $Email = $_POST["Email"];
6     $NIM = $_POST["NIM"];
7     $Address = $_POST["Address"];
8
9     $sql = "INSERT INTO seller (Name, Email, NIM, Address) VALUES (?, ?, ?, ?)";
10    $stmt = $conn->prepare($sql);
11
12    if ($stmt) {
13        $stmt->bind_param("ssss", $Name, $Email, $NIM, $Address);
14
15        if ($stmt->execute()) {
16            echo "Data inserted successfully";
17        } else {
18            echo "Error: " . $sql . "<br>" . $conn->error;
19        }
20        $stmt->close();
21    } else {
22        echo "Error: Unable to prepare statement";
23    }
24    // Tutup koneksi
25    $conn->close();
26 ?>
```

Gambar 4.2.11 Insert Seller-Create

Code PHP di atas bertujuan untuk menangani proses penambahan data (insert) ke dalam tabel "seller" pada database. Dengan menggunakan code ini, data yang dikirim dari form akan ditambahkan ke dalam tabel "seller" pada database, dan pesan sukses atau pesan error akan ditampilkan tergantung pada hasil eksekusi query.

c. Retrieve.php – Read Data

```
1  <?php
2      include 'conn.php';
3
4      $result = $conn->query("SELECT * FROM seller");
5
6      $rows = array();
7      while ($r = mysqli_fetch_assoc($result)) {
8          $rows[] = $r;
9      }
10
11     if ($rows) {
12         echo json_encode(array("success" => 1, "data" => $rows));
13     } else {
14         echo json_encode(array("success" => 0, "data" => array()));
15     }
16
17     $conn->close();
18 ?>
19
```

Gambar 4.2.12 Read Data

Code PHP di atas bertujuan untuk mengambil data dari tabel "seller" dalam database dan mengembalikan hasilnya dalam format JSON. Dengan menggunakan code ini, data dari tabel "seller" akan dikonversi ke format JSON dan dikirimkan sebagai respons.

d. Delete Data

```
deleteSeller.php
1  <?php
2      include 'conn.php';
3
4      $Id = $_POST["Id"];
5
6      $sql = "DELETE FROM seller WHERE Id = '$Id'";
7
8      if ($conn->query($sql) === TRUE) {
9          echo "Data deleted successfully";
10     } else {
11         echo "Error: " . $sql . "<br>" . $conn->error;
12     }
13
14     $conn->close();
15 ?>
16
```

Gambar 4.2.13 Delete Data

Code PHP diatas merupakan implementasi operasi "DELETE" dalam konteks CRUD (Create, Read, Update, Delete). Pada tahap "DELETE", data dari tabel "seller" akan dihapus berdasarkan nilai ID tertentu yang diberikan.

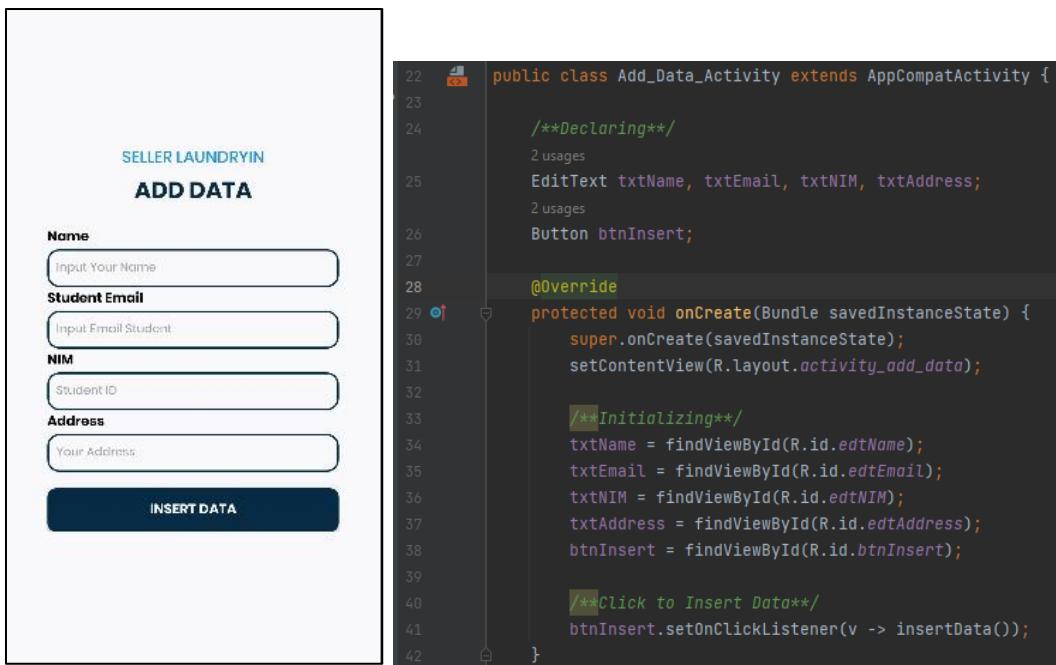
e. Update Data

```
1 ?php
2     include 'conn.php';
3
4     $Id = $_POST["Id"];
5     $Name = $_POST["Name"];
6     $Email = $_POST["Email"];
7     $NIM = $_POST["NIM"];
8     $Address = $_POST["Address"];
9
10    $sql = "UPDATE seller SET Name = '$Name', Email = '$Email', NIM = '$NIM',
11        Address = '$Address' WHERE Id = '$Id'";
12
13    if ($conn->query($sql) === TRUE) {
14        echo "Data updated successfully";
15    } else {
16        echo "Error: " . $sql . "<br>" . $conn->error;
17    }
18
19    $conn->close();
20 >
```

Gambar 4.2.14 Update Data

Code diatas berfungsi untuk memperbarui data ("seller") dalam database sesuai dengan nilai ID yang diberikan. Code mengambil parameter input berupa ID, Name, Email, Contact, dan Address melalui metode POST. Selanjutnya, query SQL digunakan untuk melakukan pembaruan data pada tabel "seller" berdasarkan ID tertentu. Setelah menjalankan query, dilakukan pengecekan keberhasilan operasi UPDATE, dan pesan yang sesuai dengan hasilnya akan ditampilkan.

- Add Seller – XML dan Java



```

68             }
69             Toast.makeText( context: this, response, Toast.LENGTH_SHORT).show();
70         }
71     }, error -> {
72         progressDialog.dismiss();
73         Toast.makeText( context: this, text: "Error: " + error.getMessage(), Toast.LENGTH_SHORT).show();
74     } {
75         @Override
76         @NotNull
77         protected Map<String, String> getParams() throws AuthFailureError {
78             Map<String, String> params = new HashMap<>();
79             params.put("Name", name);
80             params.put("Email", email);
81             params.put("NIM", nim);
82             params.put("Address", address);
83             return params;
84         }
85     };
86 
87     /**Volley Request**/
88     RequestQueue requestQueue = Volley.newRequestQueue( context: Add_Data_Activity.this);
89     requestQueue.add(request);
90 }

```

Gambar 4.2.15 Add Seller XML dan Java

Penjelasan: Code dalam Add_Data_Activity bertujuan untuk mengimplementasikan halaman pendaftaran pada aplikasi Android. Saat pengguna mengisi formulir dengan informasi yang diperlukan, seperti Nama, Email, Kontak, dan Alamat, dan menekan tombol "Insert", data tersebut dikirim ke server menggunakan metode HTTP POST.

- **EditActivity - XML dan Java**

The image shows two side-by-side screenshots. On the left is a code editor displaying the `EditActivity.java` file. The code is a standard Android activity setup with variable declarations and an `onCreate` method. On the right is a mobile application interface titled "SELLER PROFILE" and "EDIT DATA". It features five input fields labeled "ID Seller", "Name", "Email", "NIM", and "Address", each with an associated text input box. Below these fields is a large blue "UPDATE" button.

```

20 public class EditActivity extends AppCompatActivity {
21 
22     3 usages
23     EditText edt_id, edt_name, edt_email, edt_nim, edt_address;
24     2 usages
25     Button btnUpdate;
26     7 usages
27     SellerModels sellerModel;
28 
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_edit);
33 
34         edt_id = findViewById(R.id.edt_id);
35         edt_name = findViewById(R.id.edt_name);
36         edt_email = findViewById(R.id.edt_email);
37         edt_nim = findViewById(R.id.edt_nim);
38         edt_address = findViewById(R.id.edt_address);
39         btnUpdate = findViewById(R.id.btnUpdate);
40 
41         sellerModel = (SellerModels) getIntent().getSerializableExtra("sellerModel");
42     }
43 }

```

```

40     if (sellerModel != null) {
41         // Set the existing data to the EditText fields
42         edt_id.setText(sellerModel.getId());
43         edt_name.setText(sellerModel.getName());
44         edt_email.setText(sellerModel.getEmail());
45         edt_nim.setText(sellerModel.getNim());
46         edt_address.setText(sellerModel.getAddress());
47     }
48
49     btnUpdate.setOnClickListener(v -> updateData());
50 }
51
52 /**
53  * usage
54  * private void updateData() {
55     final String id = edt_id.getText().toString().trim();
56     final String name = edt_name.getText().toString().trim();
57     final String email = edt_email.getText().toString().trim();
58     final String nim = edt_nim.getText().toString().trim();
59     final String address = edt_address.getText().toString().trim();
60
61     if (name.isEmpty() || email.isEmpty() || nim.isEmpty() || address.isEmpty()) {
62         Toast.makeText(context, text: "All fields are required", Toast.LENGTH_SHORT).show();
63         return;
64     }
65     StringRequest request = new StringRequest(Request.Method.POST, url: "http://10.107.174.187/LaundryinPHP/updateSeller.php",
66         response -> {
67             Log.d(tag: "UPDATE_RESPONSE", response); // Log the response from the server
68
69             if (response.equalsIgnoreCase(anotherString: "Data updated successfully")) {
70                 Toast.makeText(context, text: "Data Updated", Toast.LENGTH_SHORT).show();
71                 // Optionally, you can go back to the previous activity or perform other actions
72                 finish();
73             } else {
74                 Toast.makeText(context, response, Toast.LENGTH_SHORT).show();
75             }
76         }, error -> {
77             Toast.makeText(context, text: "Error: " + error.getMessage(), Toast.LENGTH_SHORT).show();
78         }
79     }
80     @Override
81     protected Map<String, String> getParams() throws AuthFailureError {
82         Map<String, String> params = new HashMap<>();
83         params.put("Id", id);
84         params.put("Name", name);
85         params.put("Email", email);
86         params.put("NIM", nim);
87         params.put("Address", address);
88         return params;
89     }
90     RequestQueue requestQueue = Volley.newRequestQueue(context: EditActivity.this);
91     requestQueue.add(request);
92 }
93

```

Gambar 4.2.16 Edit Activity XML dan Java

Kode dalam kelas EditActivity bertujuan untuk menampilkan halaman pengeditan data seller pada aplikasi Android. Hal ini memungkinkan pengguna untuk mengubah informasi yang sudah ada.

- **List Seller - JSON Implementation**

```

90
91             }, new Response.ErrorListener() {
92
93     @Override
94     public void onErrorResponse(VolleyError error) {
95         Toast.makeText(context: MainSeller.this, text: "Error: " + error.getMessage(), Toast.LENGTH_SHORT).show();
96     }
97 }
98 @Override
99 protected Map<String, String> getParams() throws AuthFailureError {
100     Map<String, String> params = new HashMap<>();
101     params.put("Id", id);
102     return params;
103 }
104
105 RequestQueue requestQueue = Volley.newRequestQueue(context: this);
106 requestQueue.add(request);
107 }
108
109 @Override
110 private void handleDeleteResponse(String response) {
111     if (response.contains("Data Deleted")) {
112         Toast.makeText(context: MainSeller.this, text: "Data Deleted Successfully", Toast.LENGTH_SHORT).show();
113         sellerModelsArrayList.remove(position);
114         adapter.notifyDataSetChanged();
115         sellerModelsArrayList.remove(position);
116         adapter.notifyDataSetChanged();
117     } else {
118         Toast.makeText(context: MainSeller.this, text: "Data Not Deleted", Toast.LENGTH_SHORT).show();
119     }
120
121     StringRequest request = new StringRequest(Request.Method.POST, url,
122         new Response.Listener<String>() {
123             @Override
124             public void onResponse(String response) { handleRetrieveResponse(response); }
125         }, new Response.ErrorListener() {
126
127             @Override
128             public void onErrorResponse(VolleyError error) {
129                 Toast.makeText(context: MainSeller.this, text: "Error: " + error.getMessage(), Toast.LENGTH_SHORT).show();
130             }
131         });
132
133     RequestQueue requestQueue = Volley.newRequestQueue(context: this);
134     requestQueue.add(request);
135 }

```

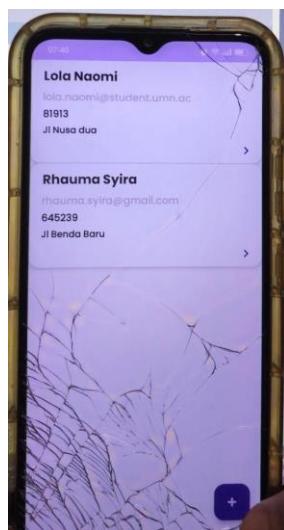
```

137     private void handleRetrieveResponse(String response) {
138         sellerModelsArrayList.clear();
139         try {
140             JSONObject jsonObject = new JSONObject(response);
141             String success = jsonObject.getString( name: "success");
142
143             if (success.equals("1")) {
144                 JSONArray jsonArray = jsonObject.getJSONArray( name: "data");
145
146                 for (int i = 0; i < jsonArray.length(); i++) {
147                     JSONObject object = jsonArray.getJSONObject(i);
148                     String id = object.getString( name: "Id");
149                     String name = object.getString( name: "Name");
150                     String email = object.getString( name: "Email");
151                     String nim = object.getString( name: "NIM");
152                     String address = object.getString( name: "Address");
153
154                     sellerModel = new SellerModels(id, name, email, nim, address);
155                     sellerModelsArrayList.add(sellerModel);
156                 }
157                 adapter.notifyDataSetChanged();
158             } else {
159                 Toast.makeText( context: MainSeller.this,  text: "No data available", Toast.LENGTH_SHORT).show();
160             }
161         } catch (JSONException e) {
162             e.printStackTrace();
163             Toast.makeText( context: MainSeller.this,  text: "JSON parsing error", Toast.LENGTH_SHORT).show();
164         }
165     }
166 }

```

Gambar 4.2.17 List Seller- JSON Implementation

- **Output**



Gambar 4.2.18 Layout

Penjelasan:

JSON (JavaScript Object Notation) pada kode tersebut digunakan sebagai format data untuk pertukaran informasi antara aplikasi Android dengan server.

- Saat aplikasi Android melakukan permintaan data ke server (melalui StringRequest), server merespons dengan data dalam format JSON. Response tersebut kemudian diolah di aplikasi Android, seperti pada fungsi onResponse di dalam metode retrieveData.
- Setelah menerima respons JSON dari server, data diolah dan diekstrak menggunakan objek JSONObject dan JSONArray. Pada kode tersebut, data seller diekstrak dari array JSON.
- Data yang telah diekstrak dari JSON kemudian dimanfaatkan untuk memperbarui tampilan aplikasi, seperti mengupdate sellerArrayList untuk kemudian diteruskan ke adapter dan ListView.

• Adapter – MyAdapter

```
public class MyAdapter extends ArrayAdapter<Seller> {

    Context context;
    List<Seller> arrayListSeller;

    public MyAdapter(@NotNull Context context, List<Seller> arrayListSeller) {
        super(context, R.layout.custom_list_item, arrayListSeller);

        this.context = context;
        this.arrayListSeller = arrayListSeller;
    }

    @NotNull
    @Override
    public View getView(int position, @Nullable View convertView, @NotNull ViewGroup parent) {

        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.custom_list_item, root: null, attachToRoot: true);

        TextView tvID = view.findViewById(R.id.txt_id);
        TextView tvName = view.findViewById(R.id.txt_name);

        tvID.setText(arrayListSeller.get(position).getId());
        tvName.setText(arrayListSeller.get(position).getName());

        return view;
    }
}
```

Activate Windows
Go to Settings to activate Windows

Gambar 4.2.19 My Adapter

Code tersebut digunakan untuk menyediakan adapter khusus yang akan mengaitkan data dari ArrayList Seller dengan tampilan yang akan ditampilkan dalam ListView. Adapter ini diperlukan agar data dari ArrayList dapat ditampilkan secara efisien dalam ListView.

- **Models – Seller**

```

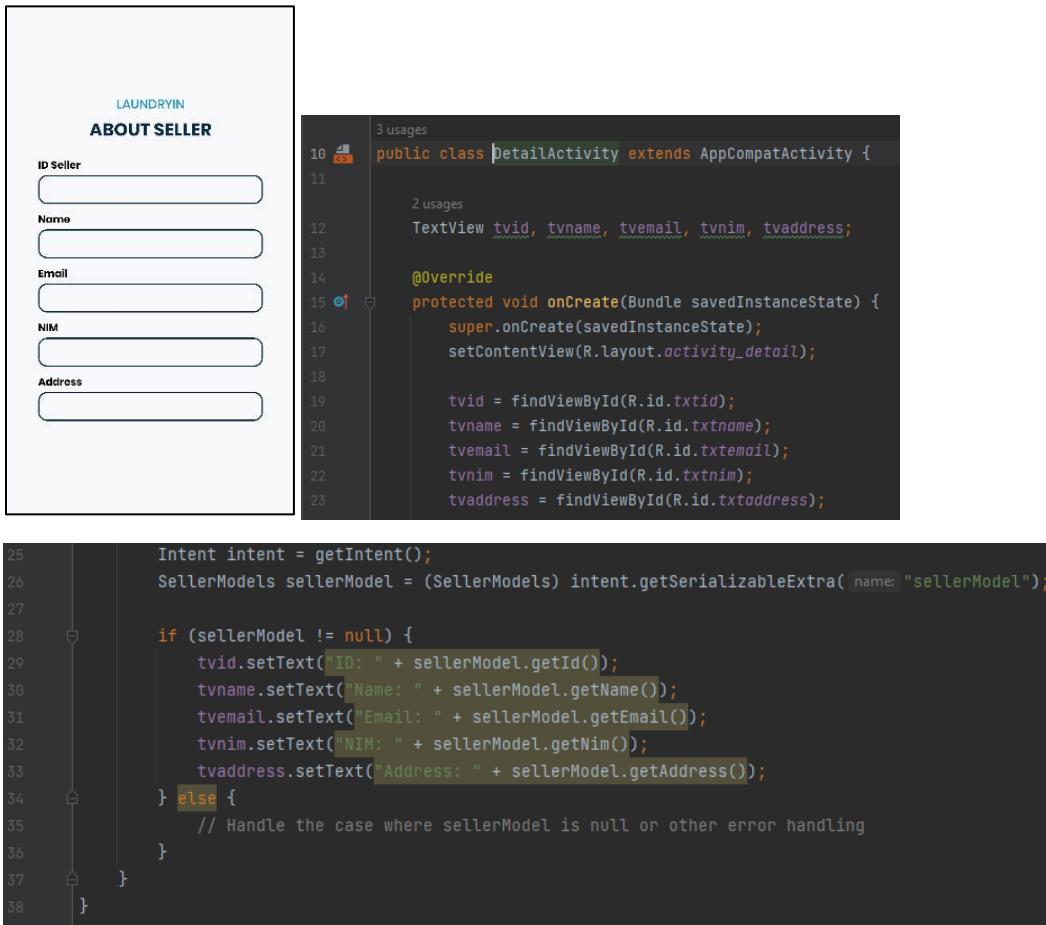
13 usages
5   public class SellerModels implements Serializable {
6     /**
7      * Declaring
8      */
9      private String id;
10     private String name;
11     private String email;
12     private String nim;
13     private String address;
14
15     /**
16      * Constructor
17      */
18     public SellerModels() {
19
20     }
21
22     /**
23      * Getter & Setter
24      */
25     public String getId() { return id; }
26
27     public void setId(String id) { this.id = id; }
28
29     public String getName() { return name; }
30
31     public void setName(String name) { this.name = name; }
32
33     public String getEmail() { return email; }
34
35     public void setEmail(String email) { this.email = email; }
36
37     public String getNim() { return nim; }
38
39     public void setNim(String nim) { this.nim = nim; }
40
41     public String getAddress() { return address; }
42
43     public void setAddress(String address) { this.address = address; }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

```

Gambar 4.2.20 Models-Seller

Class `Seller` dalam code diatas merupakan sebuah model yang mewakili entitas data dalam aplikasi Laundrin. Class ini memiliki variabel instance privat seperti `Id`, `Name`, `Email`, `Contact`, dan `Address` yang digunakan untuk menyimpan informasi. Metode getter dan setter digunakan untuk mendapatkan dan mengatur nilai variabel instance, sementara keamanan data diperkuat dengan menjadikan variabel instance bersifat privat. Class `Seller` berperan sebagai model data yang terstruktur, memfasilitasi proses CRUD pada aplikasi Laundrin dengan menyimpan informasi penting mengenai data admin seperti ID, nama, email, kontak, dan alamat secara efisien.

- **Detail Activity**



Gambar 4.2.21 Detail Activity

Class `DetailActivity` dalam kode diatas merupakan aktivitas yang digunakan untuk menampilkan detail lengkap dari entitas seller pada aplikasi CRUD Laundryin. Dalam metodenya, aktivitas ini menginisialisasi dan menetapkan nilai untuk objek `TextView` yang digunakan untuk menampilkan ID, nama, email, kontak, dan alamat. Nilai-nilai tersebut diambil dari objek `Seller` yang tersimpan dalam `sellerArrayList` yang telah diambil dari posisi tertentu melalui intent dari aktivitas sebelumnya. Dengan demikian, `DetailActivity` memberikan gambaran rinci tentang data yang dipilih, memungkinkan untuk melihat informasi secara terperinci terkait data seller.

ROLE OF PROJECT

NIM	NAMA	ROLE
00000079236	Rhauma Syira	<ul style="list-style-type: none"> - Membuat fitur SQLite untuk data pengguna aplikasi Laundry-In - Membantu rancangan UI Design pada aplikasi Laundry-In - Membantu membuat rancangan proses bisnis aplikasi Laundry-In. - Membantu membuat fitur Web view dan View Pager pada aplikasi Laundry-In. - Membuat tampilan widget dialog box pada aplikasi Laundry-In. - CRUD Json - SQLite - Seller Page - Design Layout - Merapihkan Laporan
00000082522	Haniva Yuniar	<ul style="list-style-type: none"> - Membantu membuat rancangan proses bisnis aplikasi Laundry-In. - Membantu rancangan UI Design pada aplikasi Laundry-In. - Membantu merancang UML beserta entitasnya. - Design Layout - Maps Page - Merapihkan Laporan
00000081824	Kayla Abigail	<ul style="list-style-type: none"> - Membantu membuat rancangan proses bisnis aplikasi Laundry-In - ERD - Design Layout - Membantu bagian Penelitian Terdahulu. - Merapihkan Laporan
00000081913	Lola Naomi	<ul style="list-style-type: none"> - Membantu membuat rancangan proses bisnis aplikasi Laundry-In - Membantu membuat rancangan UI Design pada aplikasi Laundry-In - Membantu bagian perancangan Activity Diagram

		<ul style="list-style-type: none">- Membantu merancang UML beserta entitasnya.- Membantu membuat fitur Web view dan View Pager pada aplikasi Laundry-In.- JSON- Maps- Design Layout- Customer Page- Checkout add to cart- Merapihkan Laporan
--	--	---