

PROPOSAL
PEMBUATAN APLIKASI HAN YING
Mata Kuliah IS534-C – Mobile Application Development
Dosen Pengampu: Nofriyadi Nurdam



Disusun oleh Kelompok :

Kevin Aditya Hartono - (NIM: 00000069875)
Ray Anthony Pranoto - (NIM: 00000066655)
Nathan Vilbert Kosasih - (NIM: 00000069903)
Juanito Arvin William - (NIM: 00000069483)
Julius Calvin Saputra - (NIM: 00000068626)
Christopher Abie Diaz Doviano - (NIM: 00000067692)

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK DAN INFORMATIKA
UNIVERSITAS MULTIMEDIA NUSANTARA
TANGERANG 2023

HALAMAN PENGESAHAN

Dokumen ini telah disetujui sebagai proposal pengembangan aplikasi untuk penggerjaan proyek akhir program studi Sistem Informasi, khususnya dalam mata kuliah Mobile Application Development. Aplikasi ini berjudul "Han Ying Security App" yang bertujuan untuk meningkatkan keamanan dan pengawasan di lingkungan perusahaan Han Ying.

NIM	NAMA	TANDA TANGAN
69875	Kevin Aditya Hartono	
66655	Ray Anthony Pranoto	
69903	Nathan Vilbert Kosasih	
69483	Juanito Arvin William	
68626	Julius Calvin Saputra	

67692	Christopher Abie Diaz Doviano	
-------	-------------------------------	---

Dibimbing oleh:

Nofriyadi

Dosen Mata Kuliah Mobile Application Development

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dimsum merupakan hidangan tradisional Cina yang terdiri dari berbagai jenis makanan kecil seperti dumpling, bakpao, dan hidangan laut lainnya yang biasanya disajikan dalam keranjang bambu. Belakangan ini, dimsum telah menjadi tren makanan populer di seluruh dunia, dengan banyak restoran yang khusus menyajikan hidangan ini. Restoran-restoran ini tidak hanya menawarkan kelezatan kuliner, tetapi juga memberikan pengalaman bersantap yang unik dan menyenangkan bagi para pelanggan.

Namun, meskipun dimsum dan restoran-restoran yang menyajikannya sedang populer, masih terdapat permasalahan yang dihadapi oleh para pemilik restoran terkait dengan pemesanan makanan. Beberapa dari mereka mengalami kesulitan dalam mengelola pesanan secara efisien, terutama pada jam-jam sibuk. Masalah-masalah ini termasuk kesalahan dalam mencatat pesanan, keterlambatan dalam pengantaran makanan, dan meningkatnya pelanggan yang kecewa karena pesanan mereka tidak tepat waktu.

Untuk mengatasi permasalahan tersebut, kelompok kami telah merancang dan mengembangkan sebuah aplikasi pemesanan makan bernama "Han Ying". Aplikasi ini memungkinkan pelanggan untuk dengan mudah melihat menu restoran, memesan hidangan dimsum favorit mereka, dan melakukan pembayaran secara online. Di sisi lain,

restoran dapat mengelola pesanan dengan lebih efisien, melacak inventaris, dan memastikan bahwa pesanan pelanggan disiapkan dan diantar tepat waktu. Dengan adanya aplikasi ini, diharapkan restoran-restoran yang menyajikan dimsum dapat meningkatkan kualitas layanan mereka, meningkatkan kepuasan pelanggan, dan mengoptimalkan proses bisnis mereka.

1.2. Rumusan Masalah

Rumusan masalah berdasarkan latar belakang tersebut adalah sebagai berikut:

1. Bagaimana aplikasi pemesanan makan "Han Ying" dapat membantu pelanggan dalam memesan hidangan dimsum dengan mudah dan melakukan pembayaran secara online?
2. Bagaimana sistem basis data dalam aplikasi "Han Ying" dapat membantu perusahaan restoran dimsum untuk mencatat pesanan dengan akurat, melacak inventaris, dan memastikan ketersediaan stok hidangan?
3. Bagaimana UI/UX yang diterapkan dalam aplikasi "Han Ying" dapat membantu restoran dalam mengelola pesanan secara efisien, meminimalkan kesalahan dalam pengantaran, dan memastikan bahwa pesanan pelanggan disiapkan dan diantar tepat waktu?

1.3. Tujuan

Terdapat pula beberapa tujuan berdasarkan rumusan masalah di atas, antara lain:

1. Menciptakan aplikasi pemesanan makan "Han Ying" yang user-friendly dan responsif, sehingga memudahkan pelanggan dalam memesan hidangan dimsum favorit mereka dan melakukan pembayaran secara online dengan cepat dan efisien.
2. Mengembangkan sistem basis data yang akurat dan efisien dalam aplikasi "Han Ying" untuk membantu perusahaan restoran dimsum mencatat pesanan pelanggan dengan detail yang lengkap, melacak inventaris dengan tepat, serta memastikan ketersediaan stok hidangan yang cukup untuk memenuhi permintaan pelanggan.
3. Merancang UI/UX yang dapat mengoptimalkan pengelolaan pesanan dalam aplikasi "Han Ying", termasuk otomatisasi pengaturan pengantaran, pelacakan real-time terhadap status pesanan, dan memberikan notifikasi kepada koki atau staf dapur mengenai pesanan yang perlu disiapkan. Hal ini bertujuan untuk meminimalkan

kesalahan dalam pengantaran dan memastikan bahwa pesanan pelanggan disajikan dengan cepat dan sesuai dengan harapan.

1.4. Profil Perusahaan

Restoran Han Ying, yang terletak di Alam Sutera, Allogio dan BSD City, Tangerang Selatan, adalah sebuah perusahaan UMKM yang membanggakan diri dalam menyajikan hidangan dimsum otentik dan berkualitas tinggi. Dengan tekad untuk memberikan pengalaman kuliner yang tak terlupakan kepada pelanggan, Han Ying telah meluncurkan aplikasi pemesanan makan "Han Ying". Aplikasi ini memungkinkan pelanggan untuk menjelajahi menu dimsum yang beragam, memesan hidangan favorit mereka, dan melakukan pembayaran secara online dengan cepat dan efisien. Dengan fokus pada rasa autentik dan kenyamanan pelanggan, Restoran Han Ying bertujuan menjadi destinasi kuliner terbaik bagi pecinta masakan Cina tradisional di Alam Sutera, Allogio dan BSD City dan sekitarnya, memberikan pengalaman bersantap yang memuaskan dan praktis.

1.5. Proses Bisnis Aplikasi

Dalam proses bisnis aplikasi "Han Ying," pelanggan pertama kali diminta untuk mendaftar akun. Mereka mengisi data diri seperti nama, alamat email, nomor telepon, dan membuat kata sandi. Setelah pendaftaran berhasil, aplikasi "Han Ying" akan menampilkan menu lengkap dengan beragam hidangan dimsum yang tersedia di restoran. Pelanggan dapat dengan mudah memilih hidangan dimsum yang mereka inginkan dari menu yang ditampilkan.

Setelah memilih makanan, pelanggan dapat menambahkannya ke dalam keranjang belanja di dalam aplikasi. Di sini, mereka memiliki fleksibilitas untuk mengubah jumlah pesanan atau menghapus item jika diperlukan. Setelah selesai memilih hidangan, pelanggan melanjutkan untuk menyelesaikan pesanan mereka dengan memilih metode pembayaran yang nyaman, termasuk kartu kredit, dompet digital, atau pembayaran tunai jika tersedia. Pelanggan juga dapat memilih apakah makanan akan diantar ke alamat mereka atau apakah mereka akan mengambilnya langsung di lokasi restoran "Han Ying."

Setelah pesanan selesai dipesan, aplikasi "Han Ying" mengirimkan konfirmasi pesanan kepada pelanggan. Konfirmasi ini mencakup detail pesanan, perkiraan waktu persiapan, dan total biaya. Pelanggan dapat melacak status pesanan mereka secara real-time, dari persiapan makanan hingga pengiriman atau pengambilan. Ketika pesanan tiba di alamat pelanggan atau siap diambil, pelanggan mengkonfirmasi penerimaan pesanan sebagai tanda bahwa pesanan telah diterima.

Selain itu, ada juga tampilan ADMIN dalam aplikasi "Han Ying." Admin memiliki akses untuk mengelola stok produk, termasuk penghapusan, penambahan, dan pembaruan data produk. Admin juga dapat melakukan pembelian stok jika diperlukan. Selain itu, admin memiliki kemampuan untuk mengedit data karyawan dalam sistem, memastikan bahwa data karyawan tetap terkini dan akurat. Dengan fitur ini, admin dapat mengoptimalkan manajemen inventaris dan informasi perusahaan.

Fitur-fitur dalam aplikasi Han Ying

1. Pengaturan Profil Pengguna:

- Pengguna dapat mengatur profil mereka dengan mengubah informasi pribadi seperti alamat, nomor telepon, dan metode pembayaran.

2. Pencarian dan Filter Menu:

- Fitur pencarian memungkinkan pengguna mencari menu atau item dimsum tertentu berdasarkan kategori menu.
- Filter dapat digunakan untuk mengurutkan atau memilah menu berdasarkan jenis dimsum, harga, atau popularitas.

3. Detail Menu dan Foto:

- Setiap item menu dimsum memiliki deskripsi lengkap dan gambar yang memungkinkan pengguna melihat dengan jelas makanan yang mereka pesan.

4. Rekomendasi Makanan:

- Aplikasi dapat memberikan rekomendasi makanan atau best seller pada homepage.

5. Keranjang Belanja:

- Pengguna dapat dengan mudah mengubah jumlah pesanan atau menghapus item dari keranjang belanja mereka.

6. Konfirmasi Pesanan:

- Konfirmasi pesanan mencakup semua detail penting termasuk rincian pesanan, biaya total, dan perkiraan waktu pengiriman atau pengambilan.
7. Pilihan Pengiriman atau Pengambilan Sendiri:
 - Pengguna dapat memilih apakah mereka ingin makanan diantar ke alamat mereka atau mengambilnya di restoran Han Ying.
 8. Metode Pembayaran:
 - Aplikasi menyediakan metode pembayaran yang aman seperti kartu kredit, dompet digital, atau pembayaran tunai.
 9. Info Lokasi Cabang restaurant:
 - Pengguna dapat melihat berbagai lokasi cabang restoran hanying yang berada di indonesia dan melihat lokasi akuratnya lewat google maps.

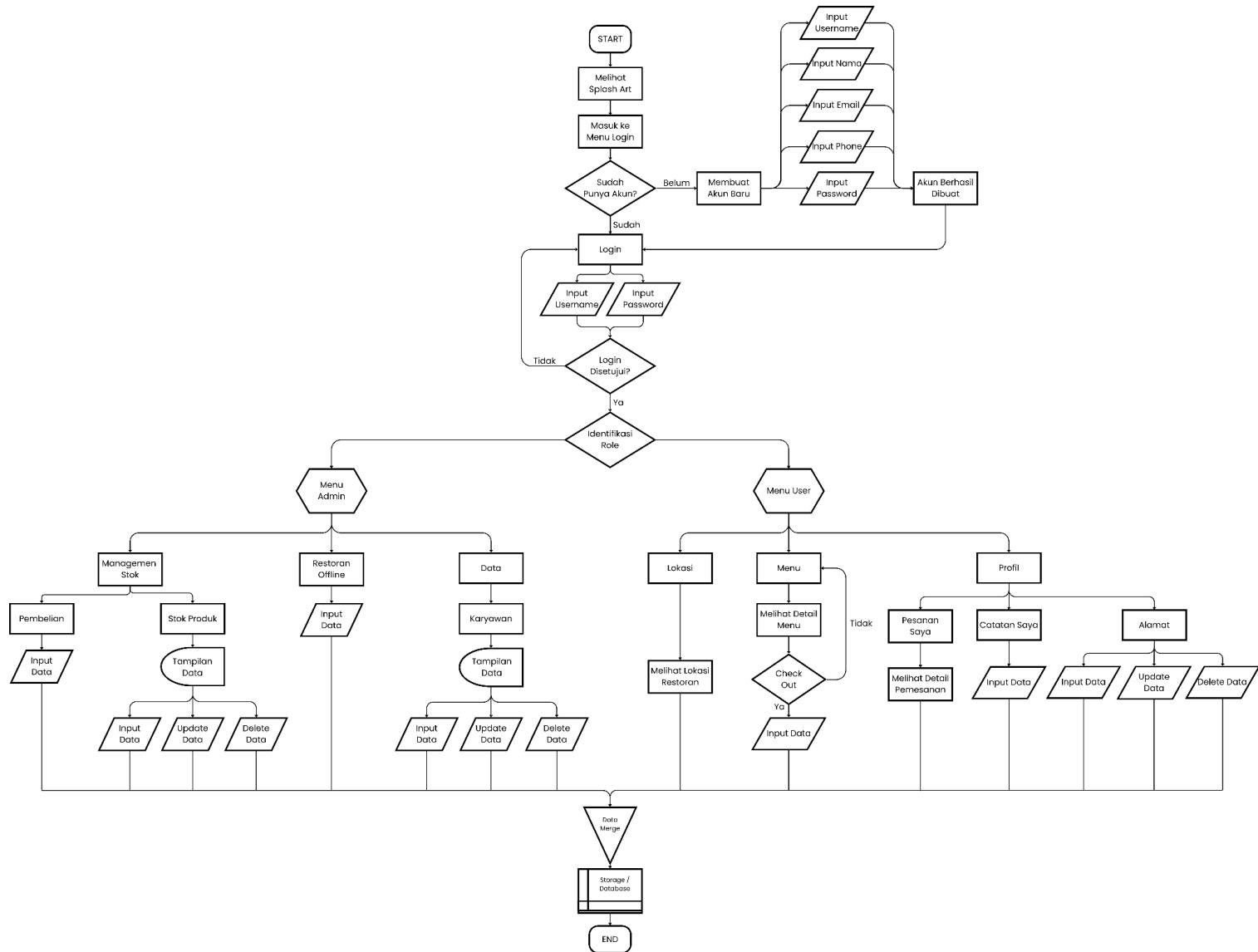
BAB II

ANALISIS MASALAH

2.1. Masalah Bisnis

Dalam industri pemesanan makanan yang semakin berkembang, terdapat sejumlah masalah bisnis yang harus diatasi. Salah satunya adalah kesulitan dalam mengelola pesanan, terutama pada jam-jam sibuk, yang dapat mengakibatkan kesalahan dalam pencatatan pesanan dan penurunan efisiensi dalam pelayanan pelanggan. Keterlambatan dalam pengantaran makanan juga menjadi masalah serius yang dapat mempengaruhi kepuasan pelanggan. Risiko kesalahan dalam pesanan seperti pesanan yang tidak sesuai dengan yang diinginkan dapat menyebabkan pelanggan kecewa. Selain itu, ada ketidakjelasan dalam proses pembayaran, yang mungkin membuat beberapa pelanggan mengalami kesulitan. Pengalaman pelanggan juga perlu diperbaiki, terutama dalam hal antarmuka pengguna yang kurang responsif atau tidak intuitif. Sistem basis data yang belum efisien dalam mencatat pesanan dan mengelola inventaris juga menjadi tantangan. Dalam usaha mengatasi masalah-masalah ini, aplikasi "Han Ying" akan memberikan solusi yang inovatif dan *user-friendly* untuk memastikan pengalaman pemesanan makanan yang lebih baik bagi pelanggan dan efisiensi yang lebih tinggi bagi restoran dimsum.

2.2. Flowchart



2.3. System Requirements

2.3.1 Functional Requirements

2.3.2 User

1. Melakukan login menggunakan username dan password yang telah mereka daftarkan sebelumnya.
2. Menyimpan informasi seperti nama, nomor telepon.
3. Menyimpan, mengubah atau menghapus alamat pengiriman yang sudah ada.
4. Memilih makanan dari beberapa menu yang tersedia
5. Melakukan filter makanan, minuman, dan hidangan penutup
6. Menambahkan makanan yang telah dipilih ke keranjang belanja.
7. Mengubah jumlah pesanan atau menghapus item dari keranjang belanja mereka.
8. Melakukan pemesanan makanan.
9. Menghitung total biaya pesanan berdasarkan item di keranjang belanja.
10. Melakukan konfirmasi penerimaan pesanan yang mencakup rincian pesanan, biaya total, dan perkiraan waktu pengiriman atau pengambilan.
11. Mendukung metode pembayaran seperti kartu kredit, dompet digital, atau pembayaran tunai.
12. Menyediakan informasi yang jelas tentang menu dimsum, termasuk deskripsi, harga, dan gambar.
13. Melihat berbagai lokasi cabang restoran Hanying yang berada di Indonesia dan melihat lokasi akuratnya lewat Google Maps.

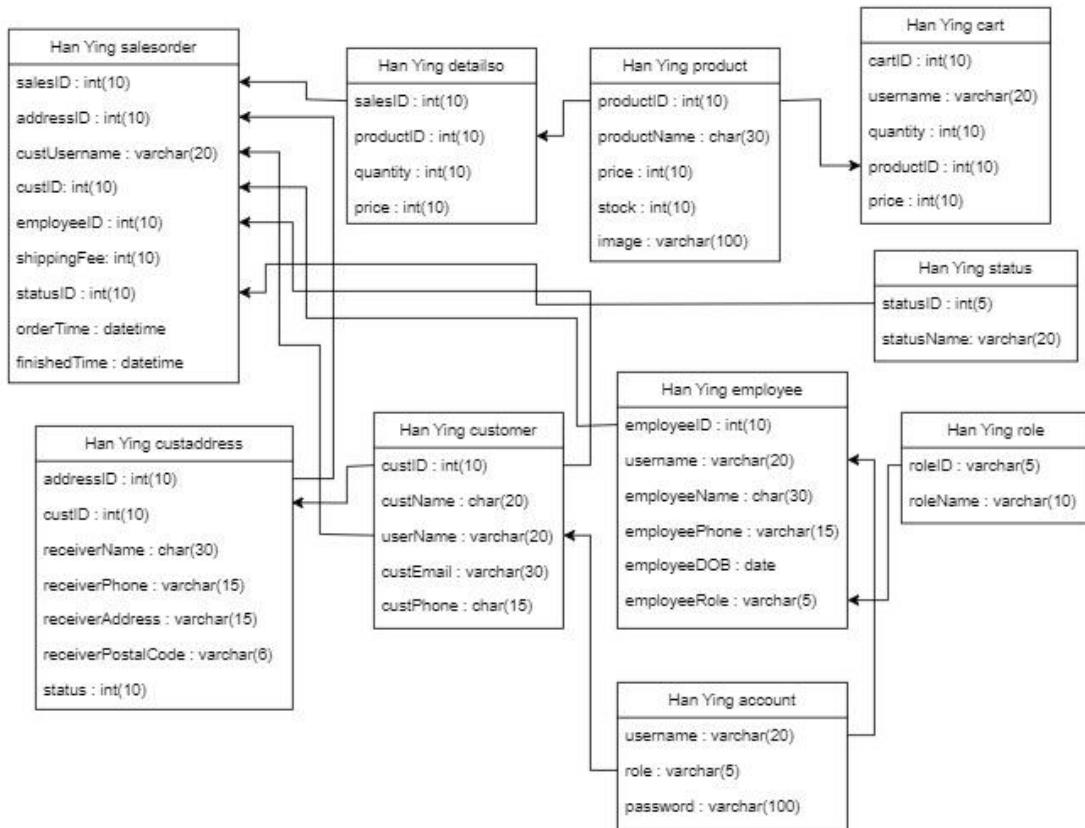
2.3.3 Admin

1. Mengubah data user seperti menambahkan, mengedit, dan menghapus user aplikasi.
2. Mengubah data produk seperti menambahkan, mengedit, dan menghapus produk.

2.3.4 Use Case Diagram



2.4. Entity Relationship Diagram (ERD)



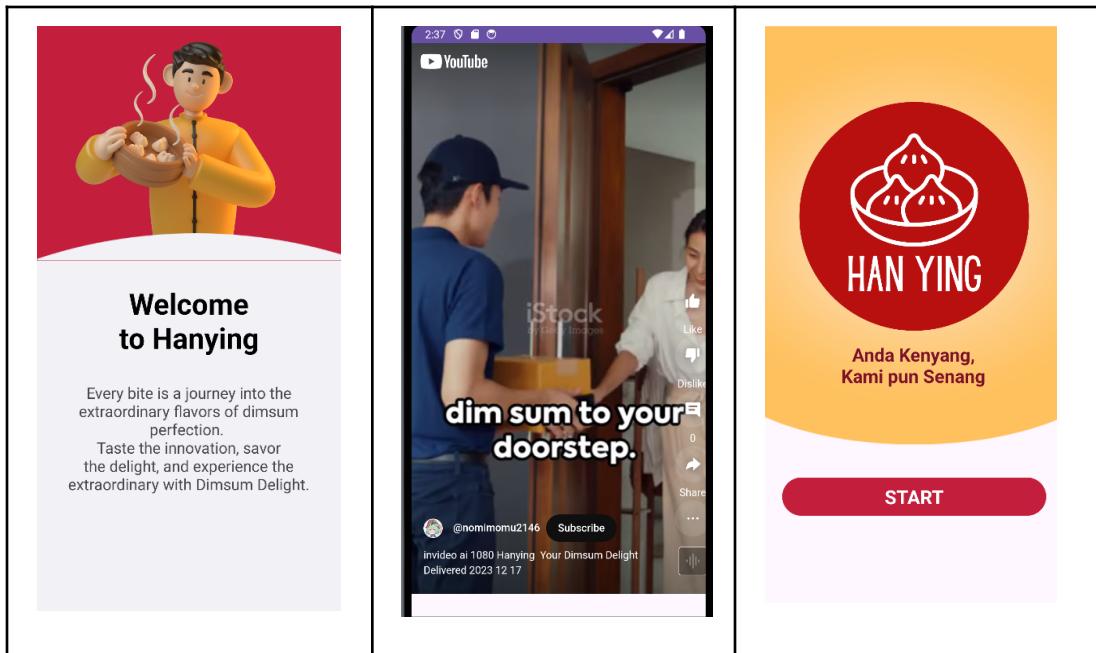
BAB III

HASIL DAN PEMBAHASAN

3.1 User (Client)

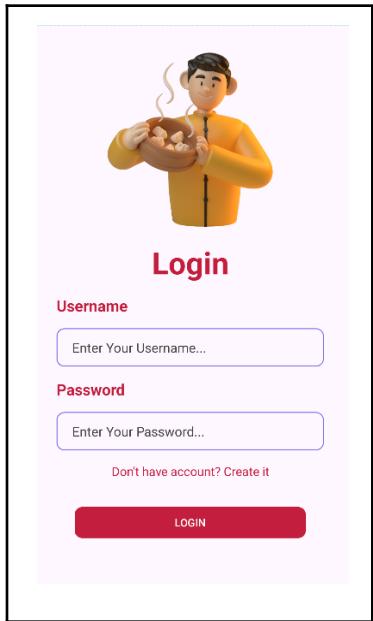
3.1.1 Interface

3.1.1.1 Splash Screen



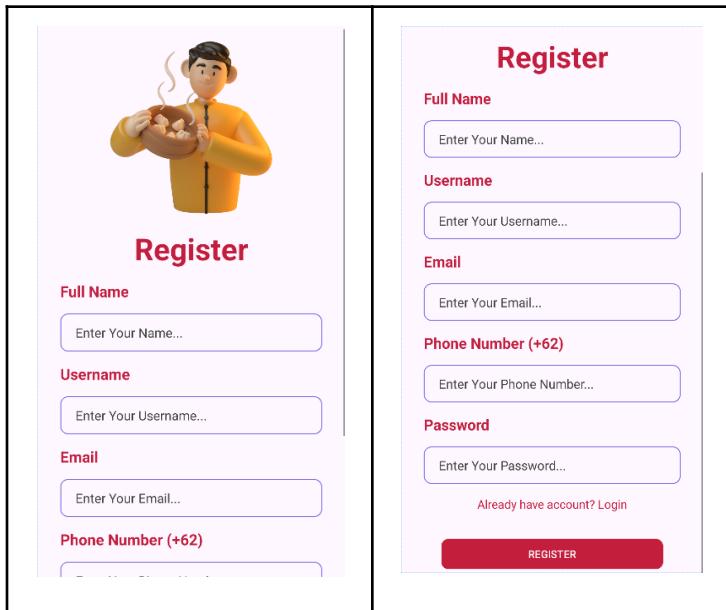
Pada gambar di atas, terdapat *Splash Screen* dalam aplikasi Hanying yang menyapa pengguna ketika mereka pertama kali membuka aplikasi. Tiga tampilan awal telah diterapkan menggunakan pager, di mana pengguna dapat melihat berbagai halaman yang memperlihatkan logo dari aplikasi Hanying, tampilan pertama adalah kata *Welcome*, pada tampilan kedua terdapat *webview* video promosi Hanying yang ada di *youtube* , dan pada tampilan terakhir terdapat tombol *start* yang akan mengirim *user* ke tampilan *login*.

3.1.1.2 Login Screen



Pada halaman *Login* aplikasi Hanying, terdapat dua kolom isian yang dapat diisi oleh pengguna, yaitu kolom untuk *username* dan *password*. Selain itu, terdapat opsi untuk membuat akun jika pengguna belum memiliki akun sebelumnya.

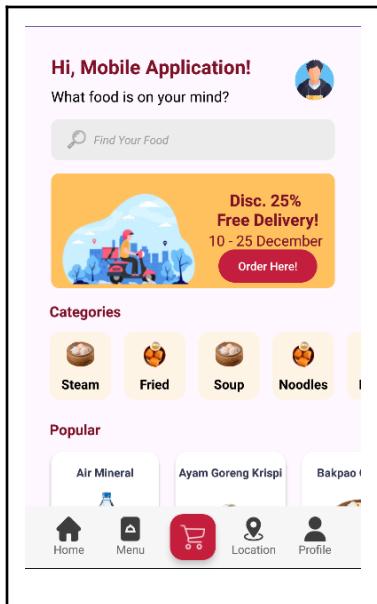
3.1.1.3 Register



Gambar di atas merupakan halaman *register* yang muncul ketika pengguna menekan tombol 'Register' pada halaman *login* ketika pengguna belum pernah mendaftarkan diri di dalam aplikasi. Pada halaman *Register*, pengguna baru diminta

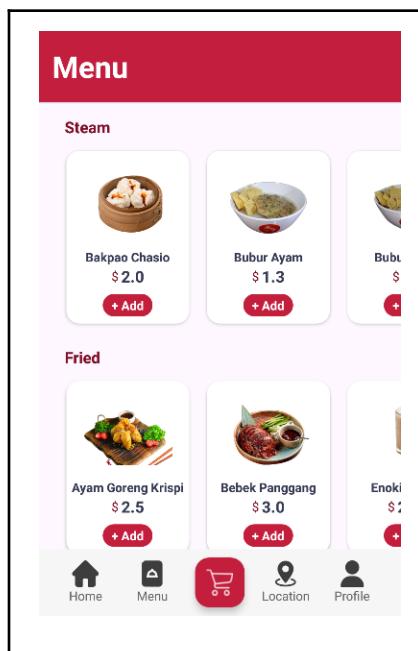
untuk mengisi informasi pribadi seperti *username*, nama, *email*, nomor telepon, dan kata sandi yang akan digunakan secara berkelanjutan dalam aplikasi Hanying. Setelah semua informasi pribadi diisi, pengguna dapat menekan tombol 'Register', dan selanjutnya, data pengguna akan disimpan dalam *database* Hanying untuk penggunaan aplikasi yang lebih lanjut

3.1.1.4 Main Menu



Gambar di atas merupakan *Main Page* dari aplikasi Hanying setelah berhasil melakukan *login user*, disini *user* dapat melakukan berbagai interaksi dengan interface termasuk mencari makanan, memilih kategori makanan dan memilih dimsum yang sedang populer. Pada bagian bawah *Main Page* juga terdapat *navigation bar home* untuk ke halaman main menu, *menu* untuk melihat semua menu makanan hanying, *cart/order list* untuk melihat makanan yang sudah dimasukan ke keranjang, *location* untuk melihat restoran offline hanying dan *profile* untuk melihat detail profil pengguna

3.1.1.5 Menu



Gambar di atas merupakan tampilan halaman *Menu* dimana user dapat melihat secara keseluruhan dan memilih makanan dan minuman dari berbagai kategori seperti *Steam*, *Fried*, *Soup*, *Drinks*, dan lainnya.

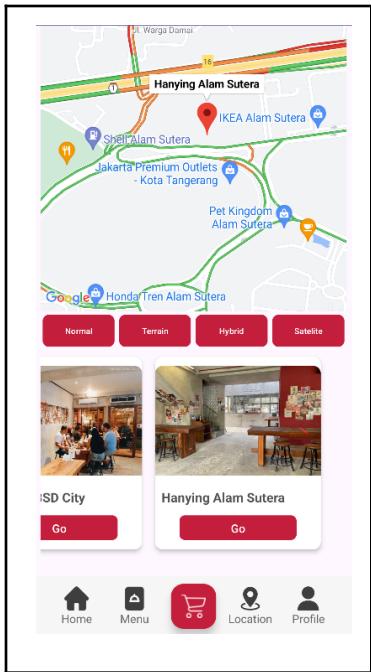
3.1.1.6 Detail Menu



Gambar di atas menampilkan halaman *Detail Menu* yang aktif saat pengguna memilih suatu produk dari halaman utama. Pada halaman ini, pengguna dapat

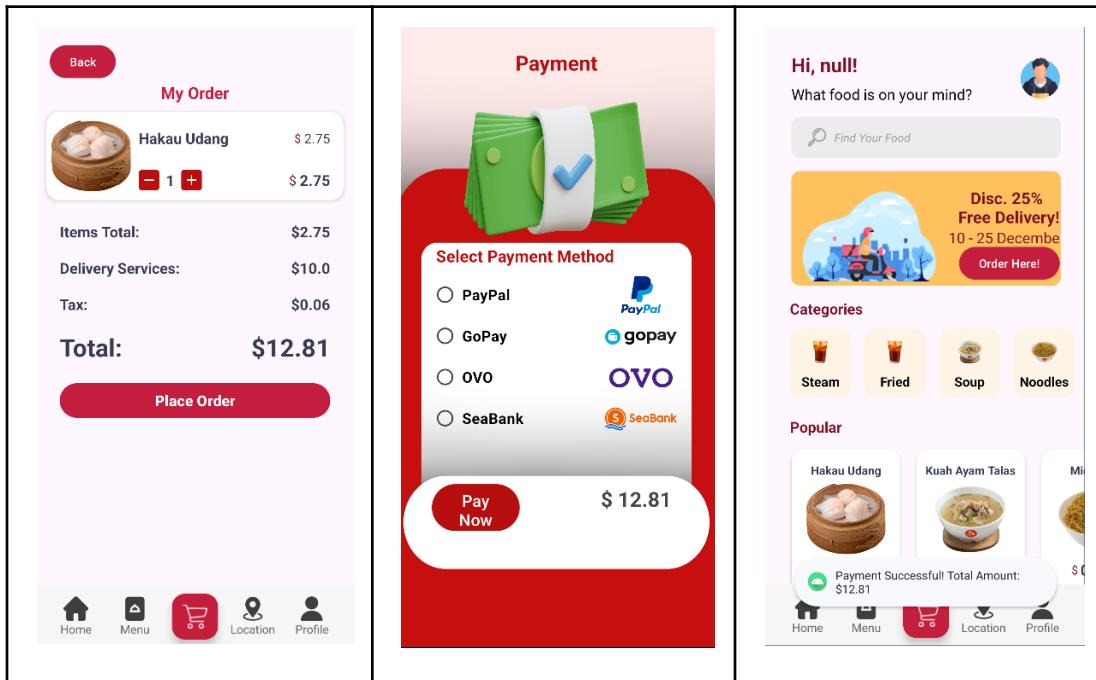
menelusuri informasi terkait dengan nama produk, harga, foto produk, dan informasi produk yang sudah terkirim melalui halaman *Menu* ataupun *Main Page* pada bagian menu yang populer. Selain itu, di sini pengguna dapat menambah dan mengurangi jumlah dimsum yang ingin mereka pesan.

3.1.1.7 Map Location



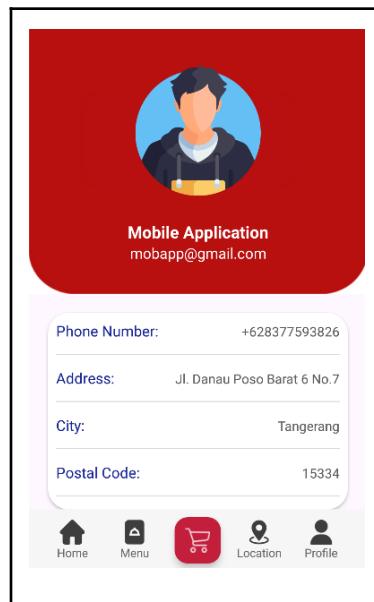
Pada gambar diatas adalah tampilan *Map Location*, disini *user* dapat melihat cabang-cabang restoran offline aplikasi Hanying, Saat *user* menekan tombol 'go' di lokasi Hanying yang ingin mereka kunjungi, peta akan menyesuaikan kamera dengan lokasi restoran tersebut. Terdapat juga opsi untuk mengubah tampilan peta sesuai keinginan *user*, seperti *Normal*, *Terrain*, *Hybrid*, dan *Satelite*, memberikan fleksibilitas untuk menyesuaikan tampilan peta sesuai preferensi mereka."

3.1.1.8 Payment



Gambar di atas merupakan halaman *Payment* dari aplikasi Hanying, setelah *user* memilih makanan yang ingin dipesan maka mereka dapat menekan tombol atau tombol keranjang dimana *user* dapat melakukan proses pembayaran. Setelah *user* sudah memastikan bahwa yang ingin dipesan sudah sesuai, maka mereka dapat menekan tombol “Place Order”, dimana mereka dapat memilih dari beberapa metode pembayaran seperti Paypal, GoPay, Ovo ataupun Seabank, apabila pembayaran berhasil maka akan keluar pesan *Toast* yang memunculkan *pop-up* “Payment Successful! Total Amount: \$...”.

3.1.1.9 Profile



Gambar diatas merupakan tampilan *Profile* yang terdiri dari gambar profil pengguna, nama lengkap, alamat email, nomor telepon, alamat, dan kode pos. Desain menggunakan latar belakang yang estetis, disertai *Navigation Bar* yang memberikan akses cepat ke menu, lokasi, dan profil, sementara tombol *cart* memfasilitasi navigasi ke daftar pesanan

3.1.2 Code

3.1.2.1 Splash Screen (SplashActivity.java)

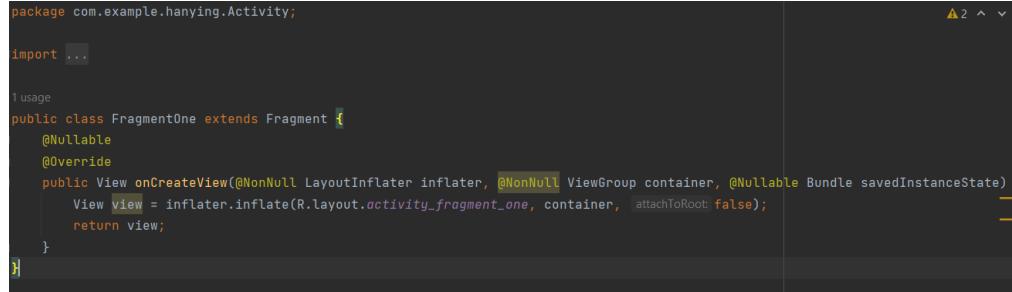
```
public class SplashActivity extends AppCompatActivity {  
    2 usages  
    ViewPager2 mViewPager;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
  
        mViewPager = findViewById(R.id.pager);  
        mViewPager.setAdapter(new ViewPagerAdapter(activity: this));  
  
    }  
    1 usage  
    public class ViewPagerAdapter extends FragmentStateAdapter {  
        1 usage  
        public ViewPagerAdapter(FragmentActivity activity) { super(activity); }  
    }
```

```
public class ViewPagerAdapter extends FragmentStateAdapter {  
    1 usage  
    public ViewPagerAdapter(FragmentActivity activity) { super(activity); }  
  
    1 usage  
    @Override  
    public Fragment createFragment(int position) {  
        if (position == 0) {  
            return new FragmentOne();  
        } else if (position == 1) {  
            return new FragmentTwo();  
        } else if (position == 2) {  
            return new FragmentThree();  
        }  
  
        return null;  
    }  
  
    @Override  
    public int getItemCount() { return 3; }  
}
```

Kode `SplashActivity.java` digunakan untuk menampilkan tampilan awal atau *splash screen* pada aplikasi Hanying. Pada saat metode `onCreate` dijalankan, tata letak aktivitas diatur dengan menggunakan `setContentView`, sementara `ViewPager2` diinisialisasi untuk menangani tiga fragment, yaitu `FragmentOne`, `FragmentTwo`, dan `FragmentThree`. Fungsi `createFragment` menentukan *fragment* yang akan muncul berdasarkan posisi halaman di

ViewPager, dan fungsi getItemCount mengembalikan total jumlah halaman, yaitu tiga. Dengan adanya ViewPager2, dengan menggunakan *pager*, halaman *splash screen* dapat dilakukan *slide* sebanyak 3 kali.

3.1.2.2 Introduction Fragment One (FragmentOne.java)



```
package com.example.hanying.Activity;

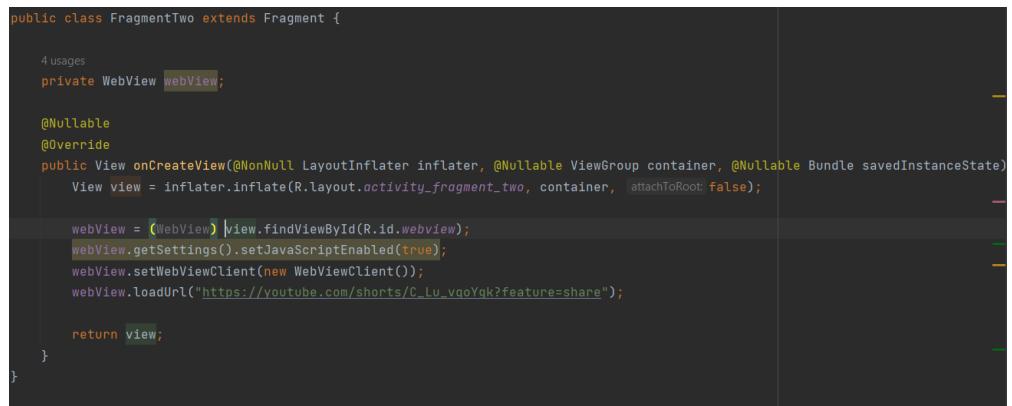
import ...

1 usage
public class FragmentOne extends Fragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @NonNull ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.activity_fragment_one, container, attachToRoot: false);
        return view;
    }
}
```

A screenshot of the Android Studio code editor showing the implementation of the FragmentOne class. The code defines a onCreateView method that inflates a layout from R.layout.activity_fragment_one. The code editor shows syntax highlighting for Java and XML, and there are some status indicators at the top right.

Kode di atas merupakan implementasi kelas FragmentOne dalam aplikasi Hanying. Kelas ini merupakan bagian dari struktur fragment pada ViewPager2 yang digunakan untuk menampilkan konten pada tampilan pertama setelah splash screen. Metode onCreateView pada kelas ini digunakan untuk menginflate atau memuat layout yang terkait dengan fragment ini, dalam hal ini diwakili oleh layout yang didefinisikan dalam berkas file activity_fragment_one.xml.

3.1.2.3 Introduction Fragment Two (WebView, FragmentTwo.java)



```
public class FragmentTwo extends Fragment {

    4 usages
    private WebView webView;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.activity_fragment_two, container, attachToRoot: false);

        webView = (WebView) view.findViewById(R.id.webview);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.setWebViewClient(new WebViewClient());
        webView.loadUrl("https://youtube.com/shorts/C_Lu_vgoYgk?feature=share");

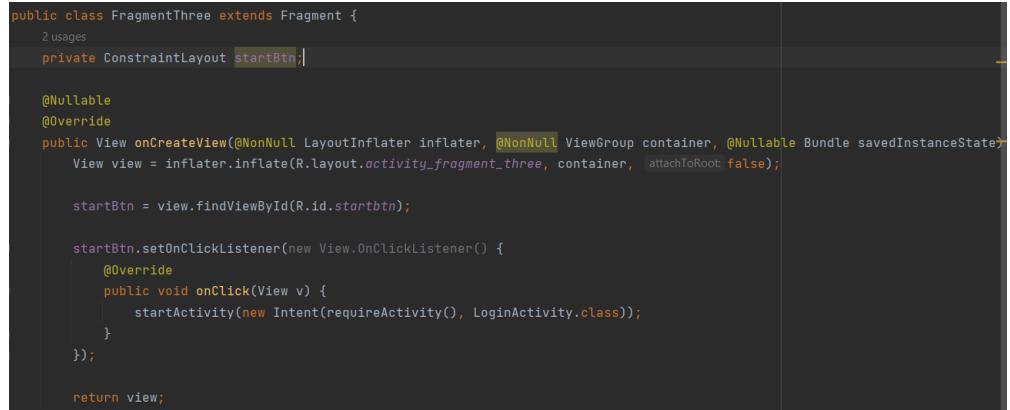
        return view;
    }
}
```

A screenshot of the Android Studio code editor showing the implementation of the FragmentTwo class. The code defines a onCreateView method that creates a WebView and loads a URL from YouTube. The code editor shows syntax highlighting for Java and XML, and there are some status indicators at the top right.

Kode di atas merupakan implementasi kelas FragmentTwo untuk menjadi tampilan kedua pada *pager* di *splash screen*. Kode tersebut mengimplementasikan *WebView* di mana kami menampilkan *video YouTube* sebagai teaser dari video promosi Hanying. *WebView* ini memungkinkan *user*

untuk mengakses konten situs web tanpa meninggalkan aplikasi. Implementasi ini memberikan pengalaman pengguna yang mulus dalam menavigasi dan menikmati *teaser video* promosi Hanying langsung dari *emulator android* di Android Studio."

3.1.2.4 Introduction Fragment Three (Start, FragmentThree.java)



```
public class FragmentThree extends Fragment {
    2 usages
    private ConstraintLayout startBtn;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @NonNull ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.activity_fragment_three, container, false);

        startBtn = view.findViewById(R.id.startbtn);

        startBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(requireActivity(), LoginActivity.class));
            }
        });
    }

    return view;
}
```

Kode di atas merupakan implementasi kelas FragmentThree dalam aplikasi Hanying. Kelas ini adalah bagian dari struktur fragment pada *ViewPager2* dan berguna untuk menampilkan tombol "Start" pada tampilan. Lalu, sebuah tombol dengan ID startbtn diinisialisasi, dan *listener* onClick ditetapkan untuk membuka aktivitas *login* (LoginActivity) ketika tombol tersebut ditekan. Pengguna dapat memulai atau masuk ke dalam aplikasi setelah melihat tampilan awal atau *splash screen*.

3.1.2.5 Login (LoginActivity.java)

```
public class LoginActivity extends AppCompatActivity {
    2 usages
    private EditText usernameEditText, passwordEditText;
    2 usages
    private Button loginButton;
    2 usages
    private TextView textViewCreateAccount;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        textViewCreateAccount = findViewById(R.id.createTxt);
        textViewCreateAccount.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { goToRegisterActivity(); }
        });

        usernameEditText = findViewById(R.id.edtUsername2);
        passwordEditText = findViewById(R.id.edtPassword2);
        passwordEditText.setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        loginButton = findViewById(R.id.btnLogin);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = usernameEditText.getText().toString();
                String password = passwordEditText.getText().toString();

                new LoginTask().execute(username, password);
            }
        });
    }
}
```

Kode tersebut merupakan bagian dari aktivitas login dalam aplikasi. Pada metode `onCreate`, *layout* utama diinisialisasi, dan elemen-elemen antarmuka seperti `TextView` untuk pendaftaran akun, `EditText` untuk `username` dan `password`, serta tombol `login` diinisialisasi. Terdapat juga tombol pendaftaran bagi para *user* yang belum terdaftar agar *user* diarahkan ke halaman pendaftaran, dan persiapan dilakukan pada elemen `password`. Saat tombol `login` ditekan, nilai `username` dan `password` diambil, dan proses login dilakukan secara asinkron melalui `AsyncTask` `LoginTask`.

```

private class LoginTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... params) {
        String urlString = "http://192.168.1.20/hanying/login.php";
        String urlString = "http://192.168.0.107/hanying/login.php";

        try {
            URL url = new URL(urlString);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setDoOutput(true);
            String postData = "username=" + params[0] + "&password=" + params[1];
            try (OutputStream os = connection.getOutputStream()) {
                os.write(postData.getBytes());
            }
            try (BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
                StringBuilder response = new StringBuilder();
                String line;

                while ((line = br.readLine()) != null) {
                    response.append(line);
                }
                return response.toString();
            }
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }
    }
}

```

Pada kelas LoginTask, bertindak sebagai background task untuk melakukan proses *login* ke *server*. Pada metode *doInBackground*, URL server ditentukan dan koneksi dibuat menggunakan protokol HTTP. Pengaturan metode *request POST* dilakukan untuk mengirim data *login* (*username* dan *password*) ke server. Selanjutnya, kode ini membaca respons dari server, mengumpulkan data dalam bentuk *string*, dan mengembalikannya.

```

@Override
protected void onPostExecute(String result) {
    try {
        JSONObject jsonResult = new JSONObject(result);

        if (jsonResult.getString("status").equals("success")) {
            // Login success
            Toast.makeText(getApplicationContext(), jsonResult.getString("message"), Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(getApplicationContext(), LoginActivity.this, MainActivity.class);
            startActivity(intent);
            finish(); // Menutup aktivitas login agar pengguna tidak dapat kembali ke sini
        } else {
            // Login failed
            Toast.makeText(getApplicationContext(), jsonResult.getString("message"), Toast.LENGTH_SHORT).show();
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

1 usage
public void goToRegisterActivity() {
    Intent intent = new Intent(getApplicationContext(), LoginActivity.this, RegisterActivity.class);
    startActivity(intent);
}

```

Dalam metode onPostExecute, hasil respons dari operasi jaringan (dalam format *string*) diubah menjadi objek `JSONObject` untuk diproses lebih lanjut. Pada contoh ini, JSON *response* diharapkan memiliki atribut "status" yang menunjukkan keberhasilan atau kegagalan *login*. Jika login berhasil ("status" adalah "success"), maka pesan sukses ditampilkan menggunakan `Toast`, dan aktivitas dialihkan ke `MainActivity`. Jika login gagal, pesan kesalahan ditampilkan. Selain itu, terdapat metode `goToRegisterActivity`, yang digunakan untuk membuat dan memulai *intent* ke aktivitas pendaftaran (`RegisterActivity`). *Intent* ini digunakan untuk berpindah antar aktivitas dalam aplikasi Android.

3.1.2.6 Login (login.php)

```
<?php
require_once 'config.php';

1 reference
function login($username, $password)
{
    global $conn;

    $username = mysqli_real_escape_string($conn, $username);
    $password = mysqli_real_escape_string($conn, $password);

    $query = "SELECT * FROM user WHERE username = '$username' AND password = '$password'";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        $user = $result->fetch_assoc();
        switch ($user['role']) {
            case 'admin':
                return array("status" => "success", "message" => "Welcome, Administrator!", "role" => "admin");
            case 'user':
                return array("status" => "success", "message" => "Welcome, ". $user['custname'],
                            "custname" => $user['custname'], "username" => $user['username'], "role" => "user");
            default:
                return array("status" => "error", "message" => "Undefined role");
        }
    } else {
        return array("status" => "error", "message" => "Invalid username or password");
    }
}

header('Content-Type: application/json');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $inputUsername = $_POST["username"];
    $inputPassword = $_POST["password"];

    $result = login($inputUsername, $inputPassword);

    echo json_encode($result);
} else {
    echo json_encode(array("error" => "Invalid Request"));
}
?>
```

Pada `login.php`, terdapat fungsi yang menerima dua parameter, yaitu *username* dan *password*. Di awal fungsi, ada sebuah file konfigurasi database yang di-include menggunakan `require_once`. Kemudian, *username* dan

password yang diterima oleh fungsi akan disanitasi menggunakan fungsi `mysqli_real_escape_string` untuk mencegah SQL Injection. Setelah itu, *query* SQL akan dieksekusi untuk mencari user dengan *username* dan *password* yang sesuai di database. Jika *user* ditemukan, maka kode akan memeriksa *role* dari *user* tersebut. Jika rolenya adalah ‘admin’, maka akan mengembalikan pesan sukses dengan pesan “Welcome, Administrator!”, jika rolenya adalah ‘user’, maka akan mengembalikan pesan sukses dengan pesan “Welcome” beserta nama *username*-nya. Jika role tidak didefinisikan, maka kode akan mengembalikan error “Undefined role”. Jika *username* atau *password* tidak valid, maka kode juga akan mengembalikan error “Invalid username or password”. Bagian bawah kode menangani *request POST* dan memanggil fungsi login dengan parameter *username* dan *password* yang diterima dari *request* tersebut.

3.1.2.7 Register (RegisterActivity.java)

```

1 usage
private class RegisterTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... params) {...}

    @Override
    protected void onPostExecute(String result) {
        try {
            JSONObject jsonResult = new JSONObject(result);

            if (jsonResult.getString("status").equals("success")) {
                // Registration success
                Toast.makeText(getApplicationContext(), jsonResult.getString("message"), Toast.LENGTH_SHORT).show();

                // Handle registration success action
                // Pindah ke menu login setelah registrasi berhasil
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
                finish(); // Menutup aktivitas registrasi agar pengguna tidak dapat kembali ke sini
            } else {
                // Registration failed
                Toast.makeText(getApplicationContext(), jsonResult.getString("message"), Toast.LENGTH_SHORT).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

public void goToLoginActivity() {
    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
    startActivity(intent);
}
}

```

Dalam metode `onPostExecute` kelas `RegisterTask`, hasil registrasi diubah menjadi objek JSON. Jika "status" adalah "success", tampilkan pesan sukses

dengan Toast dan beralih ke MainActivity. Jika gagal, tampilkan pesan kesalahan. Metode goToLoginActivity membuat *intent* untuk LoginActivity, memungkinkan berpindah antar aktivitas di aplikasi Android setelah registrasi berhasil atau gagal dalam membuat akun.

3.1.2.8 Register (register.php)

```
<?php
require_once 'config.php';

1 reference
function register($custname, $username, $custemail, $custphone, $password, $role = 'user')
{
    global $conn;

    $username = mysqli_real_escape_string($conn, $username);
    $custname = mysqli_real_escape_string($conn, $custname);
    $custemail = mysqli_real_escape_string($conn, $custemail);
    $custphone = mysqli_real_escape_string($conn, $custphone);
    $password = mysqli_real_escape_string($conn, $password);
    $role = mysqli_real_escape_string($conn, $role);

    $query = "INSERT INTO user (username, custname, custemail, custphone, password, role)
              VALUES ('$username', '$custname', '$custemail', '$custphone', '$password', '$role')";

    if ($conn->query($query)) {
        return array("status" => "success", "message" => "Account created successfully!");
    } else {
        return array("status" => "error", "message" => "Failed to create account");
    }
}

header('Content-Type: application/json');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $inputCustname = $_POST["custname"];
    $inputUsername = $_POST["username"];
    $inputCustemail = $_POST["custemail"];
    $inputCustphone = $_POST["custphone"];
    $inputPassword = $_POST["password"];

    $result = register($inputCustname, $inputUsername, $inputCustemail, $inputCustphone, $inputPassword);

    echo json_encode($result);
} else {
    echo json_encode(array("error" => "Invalid Request"));
}
?>
```

Kode yang terdapat pada register.php adalah sebuah fungsi register yang dapat menerima beberapa parameter, yaitu custname, username, custemail, custphone, password, dan role. Fungsi ini bertujuan untuk mendaftarkan user baru ke dalam database. Di awal fungsi, ada sebuah file konfigurasi database yang *di-include* menggunakan require_once. Kemudian, username, nama customer, email customer, dan password yang diterima oleh fungsi akan disanitasi menggunakan fungsi mysqli_real_escape_string untuk mencegah SQL Injection. Setelah itu, query SQL akan dieksekusi untuk memasukkan data user baru ke dalam database. Jika akun berhasil dibuat, maka kode akan mengembalikan pesan sukses; jika tidak, maka kode akan mengembalikan

pesan error. Bagian bawah kode menangani request POST dan memanggil fungsi register dengan parameter yang diterima dari request tersebut.

3.1.2.9 Homepage (MainActivity.java)

```
public class MainActivity extends AppCompatActivity {

    2 usages
    private LinearLayout homeBtn, menuBtn, locationBtn, profileBtn;
    2 usages
    private RecyclerView.Adapter adapter, adapter2;
    3 usages
    private RecyclerView recyclerViewCategoryList, recyclerViewPopularList;

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

        TextView welcomeTxt = findViewById(R.id.welcomeTxt);
        String fullName = getIntent().getStringExtra("custName");
        welcomeTxt.setText("Hi, " + fullName + "!");

        bottomNavigation();

        recyclerViewCategoryList = findViewById(R.id.recyclerView);
        new CategoryList().execute();

        recyclerViewPopularList = findViewById(R.id.recyclerView2);
        new PopularList().execute();
    }
}
```

Halaman *homepage* dirancang untuk menampilkan *interface* yang dapat berinteraksi dengan pengguna secara langsung. Dalam kelas MenuActivity, kita akan menginisialisasi elemen-elemen yang ada pada *homepage*, seperti LinearLayout untuk *navigation bar*, RecyclerView untuk menampilkan kategori produk dan menu yang populer, dan RecyclerView.Adapter agar dapat melakukan penampilan produk dengan lebih baik menggunakan *adapter*. Lalu, pada metode onCreate terdapat TextView yang berfungsi untuk menyapa *user* dengan menampilkan “Hi, User!” pada bagian awal *homepage*. bottomNavigation berfungsi untuk memanggil metode tersebut yang merupakan sebuah *navigation bar* yang memudahkan pengguna dalam berselancar pada aplikasi Hanying.

```

private class CategoryList extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... voids) {
        // ...
        String urlString = "http://192.168.1.2/hanying/category_item.php";
        String urlString = "http://192.168.0.107/hanying/category_item.php";

        try {...} catch (Exception e) {
            e.printStackTrace();
            return "error";
        }
    }

    @Override
    protected void onPostExecute(String result) {
        try {
            JSONArray jsonArray = new JSONArray(result);

            // Proses data JSON ke dalam ArrayList<CategoryDomain>
            ArrayList<CategoryDomain> categoryList = new ArrayList<>();
            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = jsonArray.getJSONObject(i);
                String catName = jsonObject.getString( name: "catname");
                String catPict = jsonObject.getString( name: "catpict");

                // Tambahkan data ke ArrayList
                categoryList.add(new CategoryDomain(catName, catPict));
            }
        }
    }
}

private class PopularList extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... voids) {
        // ...
        String urlString = "http://192.168.1.2/hanying/popular_item.php";
        String urlString = "http://192.168.0.107/hanying/popular_item.php";

        try {...} catch (Exception e) {...}
    }

    @Override
    protected void onPostExecute(String result) {
        try {
            JSONArray jsonArray = new JSONArray(result);

            // Proses data JSON ke dalam ArrayList<FoodDomain>
            ArrayList<FoodDomain> foodList = new ArrayList<>();
            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = jsonArray.getJSONObject(i);
                String foodName = jsonObject.getString( name: "foodname");
                String foodPic = jsonObject.getString( name: "foodpic");
                String foodDescription = jsonObject.getString( name: "foode desc");
                double foodPrice = jsonObject.getDouble( name: "foodprice");

                // Tambahkan data ke ArrayList
                foodList.add(new FoodDomain(foodName, foodPic, foodDescription, foodPrice));
            }
        }
    }
}

```

Kedua metode diatas, CategoryTask dan PopularTask, berfungsi untuk melakukan koneksi dengan database melalui API PHP yang terdapat pada urlString ‘<http://192.168.1.2/hanying/...php>’ dan melakukan fungsi try catch untuk mengirimkan *request method* kepada server dan menerima respons sesuai dengan input yang dikirimkan ke dalam server. Jika hasil respons sesuai

pada metode `onPostExecute`, maka akan dilakukan *parsing* menggunakan `ArrayList` untuk mengirimkan informasi ke dalam berbagai tipe data yang nantinya akan ditampilkan pada elemen UI.

3.1.2.10 Category Menu (category_item.php)

```
<?php
include 'config.php';

1 reference
function getCategoryList()
{
    global $conn;

    $query = "SELECT catName, catImage FROM product_category";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        $response = array();
        while ($row = $result->fetch_assoc()) {
            $response[] = $row;
        }
        return $response;
    } else {
        return array("status" => "error", "message" => "No categories found");
    }
}

header('Content-Type: application/json');

if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $result = getCategoryList();

    echo json_encode($result);
} else {
    echo json_encode(array("error" => "Invalid Request"));
}

?>
```

File PHP `category_item` memiliki sebuah fungsi utama yang dinamakan `getCategoryList` dan digunakan untuk mengambil daftar kategori dari database dan mengembalikannya dalam format JSON. Di awal fungsi, ada sebuah file konfigurasi database yang di-include menggunakan `include`. Kemudian, query SQL akan dieksekusi untuk mencari kategori dari tabel “`product_category`”. Jika kategori ditemukan, maka kode akan menambahkan setiap kategori ke dalam sebuah array `$response`. Jika tidak ada kategori yang ditemukan, maka kode akan mengembalikan pesan error “`No categories found`”. Bagian bawah kode menangani request GET dan memanggil fungsi `getCategoryList`. Hasilnya akan di-encode dalam format JSON dan dikirimkan sebagai

response. Jika request method bukan GET, maka kode akan mengembalikan pesan error “Invalid Request”.

3.1.2.11 Popular Menu (popular_item.php)

```
<?php
include 'config.php';

function getPopularList()
{
    global $conn;

    $query = "SELECT productName, image, productDesc, price FROM product";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        $response = array();
        while ($row = $result->fetch_assoc()) {
            $response[] = $row;
        }
        return $response;
    } else {
        return array("status" => "error", "message" => "No popular items found");
    }
}

header('Content-Type: application/json');

if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $result = getPopularList();

    echo json_encode($result);
} else {
    echo json_encode(array("error" => "Invalid Request"));
}

?>
```

Kode diatas adalah sebuah fungsi getPopularList yang bertujuan untuk mengambil daftar produk populer dari database dan mengembalikannya dalam format JSON. Di awal fungsi, ada sebuah file konfigurasi database yang *di-include* menggunakan config.php. Kemudian, *query* SQL akan dieksekusi untuk mencari produk populer dari tabel “product”. Jika produk ditemukan, maka kode akan menambahkan setiap produk ke dalam sebuah array dan mengembalikan array tersebut; jika tidak, maka kode akan mengembalikan pesan error “No popular items found”. Bagian bawah kode menangani request GET dan memanggil fungsi getPopularList. Hasilnya akan di-encode dalam format JSON dan dikirimkan sebagai respons. Jika *request method* bukan GET, maka kode akan mengembalikan pesan error “Invalid Request”.

3.1.2.12 Menu (MenuActivity.java)

```
15 usages
public class MenuActivity extends AppCompatActivity {

    2 usages
    private LinearLayout homeBtn, menuBtn, locationBtn, profileBtn;

    2 usages
    private RecyclerView
        recyclerViewMenuList,
        2 usages
        recyclerViewMenuList2,
        2 usages
        recyclerViewMenuList3,
        2 usages
        recyclerViewMenuList4,
        2 usages
        recyclerViewMenuList5;
    2 usages
    private RecyclerView.Adapter adapter;
```

Kelas *MenuActivity* di atas merupakan kelas yang akan menampilkan produk-produk yang ada pada aplikasi Hanying. Pada bagian awal kelas ini dilakukan inisialisasi beberapa elemen UI seperti *LinearLayout*, *RecyclerView*, dan *Adapter*.

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

    bottomNavigation();

    recyclerViewMenuList = findViewById(R.id.menuview);
    recyclerViewMenuList2 = findViewById(R.id.menuview2);
    recyclerViewMenuList3 = findViewById(R.id.menuview3);
    recyclerViewMenuList4 = findViewById(R.id.menuview4);
    recyclerViewMenuList5 = findViewById(R.id.menuview5);

    recyclerViewMenu(category: "Steam", recyclerViewMenuList);
    recyclerViewMenu(category: "Fried", recyclerViewMenuList2);
    recyclerViewMenu(category: "Soup", recyclerViewMenuList3);
    recyclerViewMenu(category: "Noodles", recyclerViewMenuList4);
    recyclerViewMenu(category: "Drinks", recyclerViewMenuList5);
}
```

Selanjutnya, akan dilakukan penyesuaian elemen UI dengan inisialisasi yang sudah dilakukan sebelumnya pada metode *onCreate*. *recyclerViewMenu*

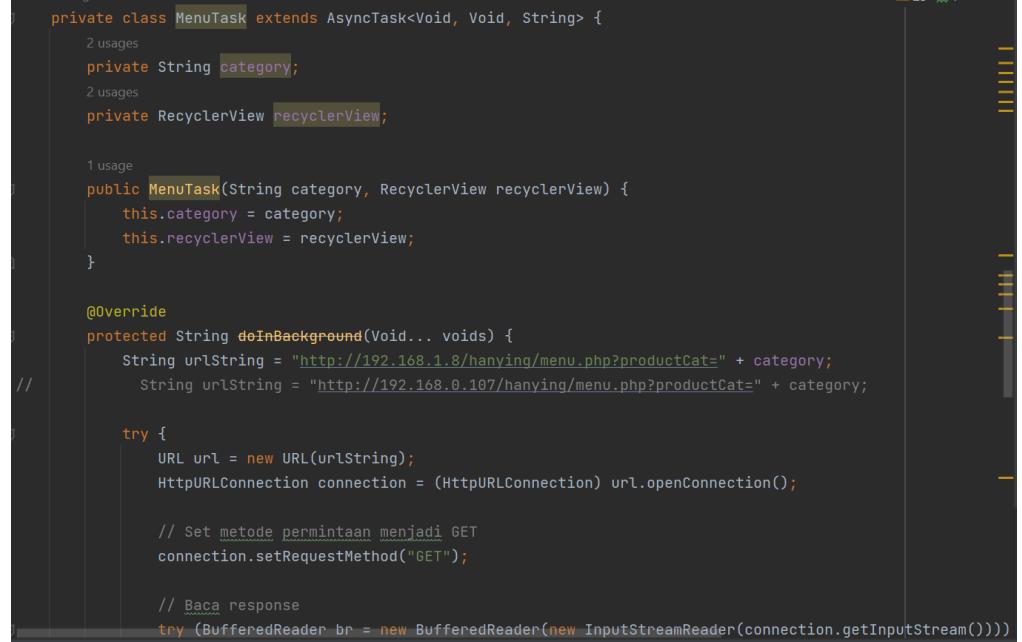
adalah salah satu cara untuk menampilkan menu berdasarkan kategori produk secara dinamis.



```
private void bottomNavigation() {...}

5 usages
private void recyclerViewMenu(String category, RecyclerView recyclerView) {
    LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: MenuActivity.this, LinearLayoutManager.VERTICAL, false );
    recyclerView.setLayoutManager(linearLayoutManager);
    recyclerView.setAdapter(null);
    new MenuTask(category, recyclerView).execute();
}
```

recyclerViewMenu akan bekerja sesuai input yang diberikan pada metode *onCreate* dan akan mengirimkan perintah kepada *MenuTask* untuk melakukan komunikasi dengan server.



```
private class MenuTask extends AsyncTask<Void, Void, String> {
    2 usages
    private String category;
    2 usages
    private RecyclerView recyclerView;

    1 usage
    public MenuTask(String category, RecyclerView recyclerView) {
        this.category = category;
        this.recyclerView = recyclerView;
    }

    @Override
    protected String doInBackground(Void... voids) {
        String urlString = "http://192.168.1.8/hanying/menu.php?productCat=" + category;
        String urlString = "http://192.168.0.107/hanying/menu.php?productCat=" + category;

        try {
            URL url = new URL(urlString);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Set metode permintaan menjadi GET
            connection.setRequestMethod("GET");

            // Baca response
            try (BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream())))
}
```

MenuTask berperan penting sebagai penghubung antara aplikasi dengan server database yang terdapat pada metode *doInBackground* dan mengirimkan *request method* “GET”.

```
// Baca response
try (BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
    StringBuilder response = new StringBuilder();
    String line;

    while ((line = br.readLine()) != null) {
        response.append(line);
    }

    return response.toString();
}
} catch (Exception e) {
    e.printStackTrace();
    return "error";
}
}
```

```
@Override
protected void onPostExecute(String result) {
    try {
        JSONArray jsonArray = new JSONArray(result);

        // Proses data JSON ke dalam ArrayList<FoodDomain>
        ArrayList<FoodDomain> foodList = new ArrayList<>();
        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            String prodName = jsonObject.getString("productName");
            String prodImage = jsonObject.getString("image");
            String prodDesc = jsonObject.getString("productDesc");
            double prodPrice = jsonObject.getDouble("price");

            // Tambahkan data ke ArrayList
            foodList.add(new FoodDomain(prodName, prodImage, prodDesc, prodPrice));
        }

        // Inisialisasi RecyclerView dan adapter
        adapter = new MenuAdapter(foodList);
        recyclerView.setAdapter(adapter);

    } catch (JSONException e) {
        e.printStackTrace();
        Toast.makeText(context, "Error processing JSON (Menu)", Toast.LENGTH_SHORT).show();
    }
}
```

Hasil respons dari server akan disimpan pada JSONArray, dan akan dilakukan parsing ke adapter FoodList agar dapat tampil dinamis menyesuaikan dengan banyaknya data pada server.

3.1.2.13 Menu (menu.php)

```
<?php
require_once 'config.php';

1 reference
function getMenuList($category)
{
    global $conn;

    $query = "SELECT productName, image, productDesc, price FROM product WHERE productCat = '$category'";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        $response = array();
        while ($row = $result->fetch_assoc()) {
            $response[] = $row;
        }
        return $response;
    } else {
        return array("status" => "error", "message" => "No menu items found for this category");
    }
}

header('Content-Type: application/json');

if ($_SERVER["REQUEST_METHOD"] == "GET") {
    if (isset($_GET["productCat"])) {
        $category = $_GET["productCat"];
        $result = getMenuList($category);
        echo json_encode($result);
    } else {
        echo json_encode(array("error" => "Invalid Request (No Matches)"));
    }
} else {
    echo json_encode(array("error" => "Invalid Request"));
}
?>
```

Fungsi getMenuList menerima parameter \$category dan mengambil informasi produk dari database berdasarkan kategori tersebut. Fungsi ini menjalankan query SQL yang mengambil nama produk, gambar, deskripsi produk, dan harga dari tabel produk di mana kategori produk sesuai dengan \$category. Jika ada item yang ditemukan, fungsi ini mengembalikan array dari semua baris hasil. Jika tidak ada item yang ditemukan, fungsi ini mengembalikan array dengan pesan error. Bagian kode lainnya menangani jenis permintaan HTTP dan memberikan respons error jika metode permintaan tidak valid atau jika parameter productCat tidak disetel dalam permintaan GET. Jika semuanya berjalan lancar, fungsi getMenuList dipanggil dan hasilnya dikodekan ke dalam format JSON dan dikirim kembali sebagai respons.

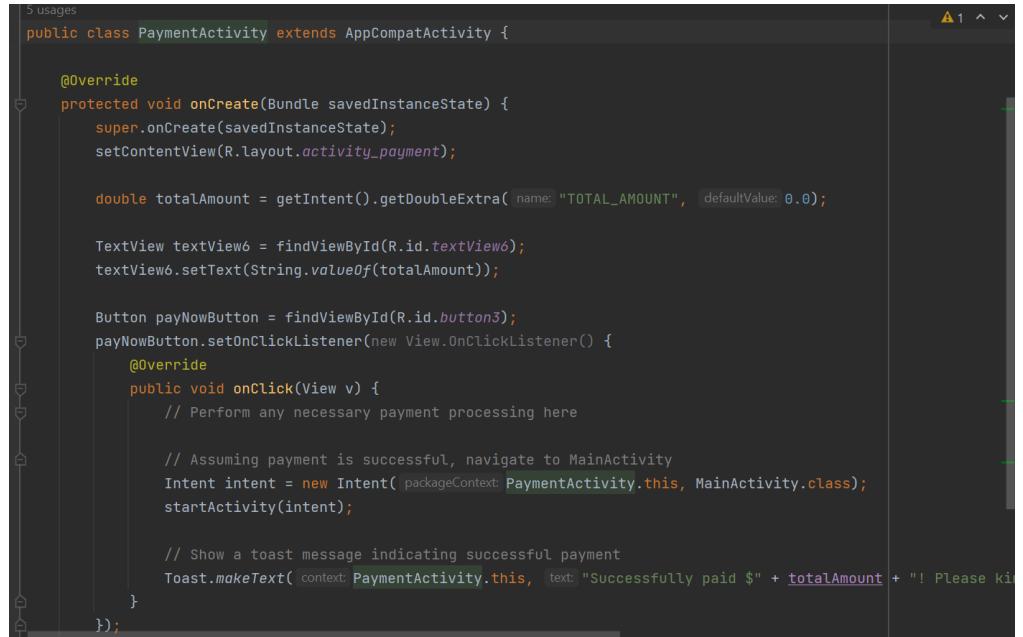
3.1.2.14 Order List (OrderListActivity.java)

```
usage:  
private void initList() {  
    LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL, reverseLayout: false);  
    recyclerViewList.setLayoutManager(linearLayoutManager);  
    adapter = new CartListAdapter(managementCart.getListCard(), context: this, new ChangeNumberItemsListener() {  
        4 usages  
        @Override  
        public void changed() { calculateCard(); }  
    });  
  
    recyclerViewList.setAdapter(adapter);  
    if (managementCart.getListCard().isEmpty()) {  
        emptyTxt.setVisibility(View.VISIBLE);  
        scrollView.setVisibility(View.GONE);  
    } else {  
        emptyTxt.setVisibility(View.GONE);  
        scrollView.setVisibility(View.VISIBLE);  
    }  
}
```

```
private void calculateCard() {  
    double percentTax = 0.02;  
    double delivery = 10;  
  
    tax = Math.round((managementCart.getTotalFee() * percentTax) * 100.0) / 100.0;  
    double total = Math.round((managementCart.getTotalFee() + tax + delivery) * 100.0) / 100.0;  
    double itemTotal = Math.round(managementCart.getTotalFee() * 100.0) / 100.0;  
  
    totalFeeTxt.setText("$" + itemTotal);  
    taxTxt.setText("$" + tax);  
    deliveryTxt.setText("$" + delivery);  
    totalTxt.setText("$" + total);  
  
    placeOrderBtn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            // Pass the total value to the Payment activity  
            Intent intent = new Intent( packageContext: OrderListActivity.this, Payment.class);  
            intent.putExtra( name: "TOTAL_AMOUNT", total);  
            startActivity(intent);  
        }  
    });  
  
    backBtn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            Intent intent = new Intent( packageContext: OrderListActivity.this, MainActivity.class);  
            startActivity(intent);  
        }  
    });  
}
```

Metode `initList` menginisialisasi tampilan daftar belanja dan menyesuaikannya berdasarkan ketersediaan barang. Metode `calculateCard` menghitung total biaya belanja, termasuk pajak dan biaya pengiriman, serta menampilkan informasi ini pada antarmuka pengguna. Tombol "Place Order" mengarahkan ke aktivitas pembayaran dengan menyertakan total biaya, sementara tombol "Back" kembali ke halaman utama.

3.1.2.15 Payment (PaymentActivity.java)



```
5 usages
public class PaymentActivity extends AppCompatActivity {

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

        double totalAmount = getIntent().getDoubleExtra("TOTAL_AMOUNT", 0.0);

        TextView textView6 = findViewById(R.id.textView6);
        textView6.setText(String.valueOf(totalAmount));

        Button payNowButton = findViewById(R.id.button3);
        payNowButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Perform any necessary payment processing here

                // Assuming payment is successful, navigate to MainActivity
                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                startActivity(intent);

                // Show a toast message indicating successful payment
                Toast.makeText(getApplicationContext(), "Successfully paid $" + totalAmount + "! Please ki
            }
        });
    }
}
```

Pada kelas PaymentActivity terdapat beberapa metode yang berfungsi untuk melakukan pembayaran. totalAmount merupakan biaya keseluruhan yang harus dibayarkan oleh pengguna, yang diambil melalui intent dari kelas OrderList. Pengguna dapat melakukan pembayaran dengan mengklik tombol payNowButton. Jika pembayaran berhasil maka akan memunculkan Toast dan akan dialihkan ke MainActivity dengan asumsi transaksi pembayaran sudah selesai

3.1.2.16 Detail (ShowDetailActivity.java)

```
private void getBundle() {
    object = (FoodDomain) getIntent().getSerializableExtra("object");

    int drawableResourceId = this.getResources().getIdentifier(object.getPic(), "drawable", this.getPackageName());

    Glide.with(activity).RequestManager
        .load(drawableResourceId).RequestBuilder<Drawable>
        .into(picFood);

    titleTxt.setText(object.getName());
    priceTxt.setText("$" + object.getPrice());
    descriptionTxt.setText(object.getDescription());
    numberOrderTxt.setText(String.valueOf(numberOrder));

    plusBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            numberOrder = numberOrder + 1;
            numberOrderTxt.setText(String.valueOf(numberOrder));
        }
    });

    minusBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (numberOrder > 1) {
                numberOrder = numberOrder - 1;
            }
            numberOrderTxt.setText(String.valueOf(numberOrder));
        }
    });
}
```

```
addOrderBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        object.setNumberInCart(numberOrder);
        managementCart.insertFood(object);

        Intent intent = new Intent(getApplicationContext(), ShowDetailActivity.this, OrderListActivity.class);
        startActivity(intent);
    }
});
```

Kode `getBundle()` pada aktivitas ini berfungsi untuk menampilkan detail makanan. Pertama, objek makanan (FoodDomain) diambil dari intent sebagai objek serialisasi, dan gambar makanan ditampilkan menggunakan Glide. Informasi seperti nama, harga, deskripsi, dan jumlah pesanan ditetapkan ke elemen-elemen UI seperti TextView. Pengguna dapat menyesuaikan jumlah pesanan melalui tombol tambah dan kurang, yang menanggapi klik untuk

menambah atau mengurangi jumlah pesanan. Saat tombol "Tambahkan ke Keranjang" di klik, jumlah pesanan dan objek makanan disimpan, dan pengguna dialihkan ke aktivitas daftar pesanan (OrderListActivity). Dengan demikian, kode ini menyediakan fungsi lengkap untuk memilih, menyesuaikan jumlah, dan menambahkan makanan ke dalam keranjang belanja pada aplikasi tersebut.

3.1.2.17 Map Location (MapLocation.java)

```
@Override  
public void onMapReady(GoogleMap googleMap) {  
    map = googleMap;  
  
    home = new LatLng( latitude: -6.256201442141357, longitude: 106.61856120065639);  
    map.addMarker(new MarkerOptions().position(home).title("Welcome to UMN!")).showInfoWindow();  
    map.moveCamera(CameraUpdateFactory.newLatLng(home));  
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(home, zoom: 16));  
    map.setTrafficEnabled(true);  
}
```

Kode diatas terdapat metode onMapReady() pada Android menggunakan Google Maps API. Peta Google Maps disiapkan dengan objek GoogleMap, dan sebuah marker pada lokasi UMN, kemudian dengan map.moveCamera ,kamera awalnya dipindahkan ke lokasi tanpa efek zoom, dan kemudian dipindahkan kembali dengan efek zoom sebesar 16, memberikan tampilan yang lebih terperinci. Lapisan traffic diaktifkan untuk menampilkan informasi lalu lintas di sekitar lokasi tersebut.

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

    recyclerView = findViewById(R.id.view1);

    trendItemList = new ArrayList<>();
    trendItemList.add(new TrendsDomain(R.drawable.allo, title: "Hanying Allegio", buttonAction: 1));
    trendItemList.add(new TrendsDomain(R.drawable.bsd, title: "Hanying BSD City", buttonAction: 2));
    trendItemList.add(new TrendsDomain(R.drawable.alsut, title: "Hanying Alam Sutera", buttonAction: 3));

    TrendsAdapter adapterTrends = new TrendsAdapter( context: this, trendItemList, buttonClickListener: this);
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this, LinearLayoutManager.HORIZONTAL, reverseLayout: false));
    recyclerView.setAdapter(adapterTrends);

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
            MY_PERMISSIONS_REQUEST_LOCATION);
    } else {
        initializeLocation();
    }
    try {
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.mapFragment);
        mapFragment.getMapAsync( callback: this);
    } catch (Exception e) {
        Toast.makeText(getApplicationContext(), e.toString(), Toast.LENGTH_LONG).show();
    }
}
```

Pada kode diatas, RecyclerView diidentifikasi dengan menggunakan metode findViewById untuk menampilkan tren lokasi pada layar. Objek-objek TrendsDomain yang merepresentasikan tren lokasi ditambahkan ke dalam ArrayList trendItemList. Sebuah adapter kustom, TrendsAdapter, diinisialisasi dengan menggunakan data tren lokasi tersebut. RecyclerView kemudian dikonfigurasi dengan setLayoutManager untuk menampilkan item secara horizontal, dan dihubungkan dengan adapter menggunakan setAdapter. Selanjutnya, kita melakukan pengecekan izin untuk akses lokasi map dan jika belum disetujui, permintaan izin diajukan. Jika izin sudah diberikan sebelumnya, metode initializeLocation() dipanggil untuk memulai pengaturan lokasi. Terakhir, SupportMapFragment digunakan untuk mendapatkan referensi peta dari fragment di layout dengan getMapAsync(this), yang akan memicu pemanggilan metode onMapReady() untuk menginisialisasi peta Google Maps.

```
Button terrain = findViewById(R.id.btnTerrainMode);
Button hybrid = findViewById(R.id.btnHybrid);
Button satellite = findViewById(R.id.btnSatelliteMode);
Button normal = findViewById(R.id.btnNormalMode);

terrain.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { map.setMapType(GoogleMap.MAP_TYPE_TERRAIN); }
});

hybrid.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { map.setMapType(GoogleMap.MAP_TYPE_HYBRID); }
});

satellite.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { map.setMapType(GoogleMap.MAP_TYPE_SATELLITE); }
});

normal.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { map.setMapType(GoogleMap.MAP_TYPE_NORMAL); }
});

bottomNavigation();
```

Lanjutan dari kode sebelumnya, kita menginisialisasi empat tombol (terrain, hybrid, satellite, dan normal) dalam tampilan menggunakan metode findViewById. Setiap tombol memiliki clicklistener yang ditetapkan dengan menggunakan antarmuka View.OnClickListener. Ketika tombol ditekan, peta Google yang diasosiasikan dengan aktivitas akan mengubah jenisnya sesuai dengan mode yang dipilih: MAP_TYPE_TERRAIN untuk tombol terrain, MAP_TYPE_HYBRID untuk tombol hybrid, MAP_TYPE_SATELLITE untuk tombol satellite, dan MAP_TYPE_NORMAL untuk tombol normal. Ada juga fungsi bottomNavigation() untuk tampilan navigasi di bagian bawah layar.

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PLACE_AUTO) {
        if (resultCode == RESULT_OK) {
            Place place = PlaceAutoComplete.getPlace(context: this, data);
            LatLng search = place.getLatLng();
            double x = place.getLatLng().latitude;
            double y = place.getLatLng().longitude;
            String address = place.getAddress().toString();
            String phone = place.getPhoneNumber().toString();
            String name = place.getName().toString();
            String snippet = address + System.getProperty("line.separator") + phone;
            map.addMarker(new MarkerOptions().position(search).title(name).snippet(snippet)).showInfoWindow();
            map.animateCamera(CameraUpdateFactory.newLatLng(search));
            map.animateCamera(CameraUpdateFactory.newLatLngZoom(search, zoom: 18));
        }
    }
}

```

Metode `onActivityResult` di atas digunakan untuk menanggapi hasil dari aktivitas pemilihan tempat (PLACE_AUTO). Jika hasilnya berhasil (RESULT_OK), maka objek Place diperoleh dari data intent. Informasi seperti koordinat lintang dan bujur, alamat, nomor telepon, nama tempat, dan deskripsi tambahan diambil dari objek Place. Selanjutnya, sebuah marker ditambahkan ke peta Google dengan menggunakan informasi tersebut, dan kamera peta diarahkan untuk fokus pada lokasi baru.

```

@Override
public void onButtonClick(int position) {
    TrendsDomain trendItem = trendItemList.get(position);
    int buttonAction = trendItem.getButtonAction();

    switch (buttonAction) {
        case 1:
            LatLng location1 = new LatLng(latitude: -6.280770, longitude: 106.663774);
            map.addMarker(new MarkerOptions().position(location1).title("Hanying Allegio")).showInfoWindow();
            map.animateCamera(CameraUpdateFactory.newLatLng(location1));
            break;
        case 2:
            LatLng location2 = new LatLng(latitude: -6.256201442141357, longitude: 106.61856120065639);
            map.addMarker(new MarkerOptions().position(location2).title("Hanying BSD City")).showInfoWindow();
            map.animateCamera(CameraUpdateFactory.newLatLng(location2));
            break;
        case 3:
            LatLng location3 = new LatLng(latitude: -6.220716, longitude: 106.659409);
            map.addMarker(new MarkerOptions().position(location3).title("Hanying Alam Sutera")).showInfoWindow();
            map.animateCamera(CameraUpdateFactory.newLatLng(location3));
            break;
    }
}

```

Kode terakhir pada `maplocation.java` adalah metode `onButtonClick` yang menanggapi klik tombol pada elemen tren di RecyclerView. Melalui objek TrendsDomain, kode menentukan tindakan tombol yang sesuai dengan item tren yang diklik. Setiap tindakan tombol menghasilkan penambahan marker ke peta Google pada lokasi yang ditentukan, dengan menyesuaikan kamera untuk

fokus pada lokasi tersebut. Dengan menggunakan switch statement, tindakan tombol terkait dengan masing-masing lokasi tertentu, seperti Allogio, BSD City, atau Alam Sutera.

3.1.2.18 Profile (ProfileActivity.java)

```
private LinearLayout homeBtn, menuBtn, locationBtn, profileBtn;

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

    bottomNavigation();
}

1 usage
private void bottomNavigation() {
    FloatingActionButton floatingActionButton = findViewById(R.id.order_btn);
    homeBtn = findViewById(R.id.homeBtn);
    menuBtn = findViewById(R.id.menuBtn);
    locationBtn = findViewById(R.id.locationBtn);
    profileBtn = findViewById(R.id.profileBtn);

    floatingActionButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(packageContext: Profile.this, OrderListActivity.class));
        }
    });
};

homeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(packageContext: Profile.this, MainActivity.class));
    }
});

menuBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(packageContext: Profile.this, MenuActivity.class));
    }
});

locationBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(packageContext: Profile.this, MapLocation.class));
    }
});

profileBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(packageContext: Profile.this, Profile.class));
    }
});
```

Gambar diatas adalah kode dari profileActivity, pada profil activity hanya terdapat kode untuk bottomNavigation. Saat aktivitas profil dimulai, fungsi

bottomNavigation dijalankan untuk menginisialisasi dan menangani berbagai tombol navigasi, termasuk tombol floating action button untuk mengakses daftar pesanan. Setiap tombol navigasi memiliki implementasi OnClickListener yang mengarahkan pengguna ke aktivitas terkait, seperti beranda, menu, lokasi pada peta, atau halaman profil.

3.1.2.19 Profile (profile.php)

```
<?php
include 'config.php';

1 reference
function getProfileData($username)
{
    global $conn;

    $query = "SELECT custname, custemail, custphone FROM user WHERE username = '$username'";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        return array("status" => "success", "data" => $row);
    } else {
        return array("status" => "error", "message" => "Profile data not found");
    }
}

header('Content-Type: application/json');

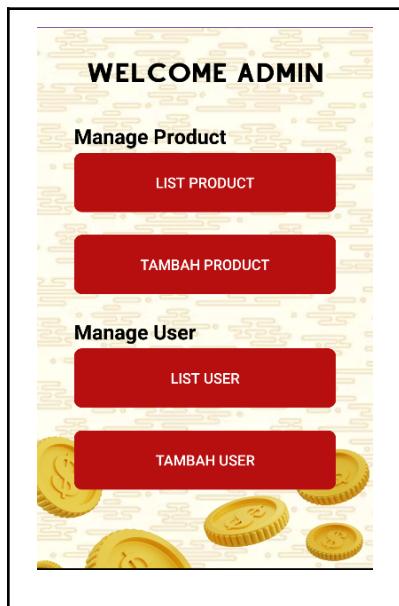
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $username = $_GET['username'];
    $result = getProfileData($username);

    echo json_encode($result);
} else {
    echo json_encode(array("error" => "Invalid Request"));
}
?>
```

3.2 Admin (Server)

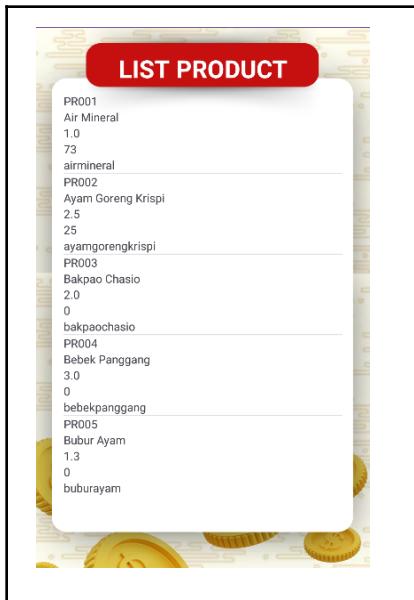
3.2.1 Interface

3.2.1.1 Main Menu



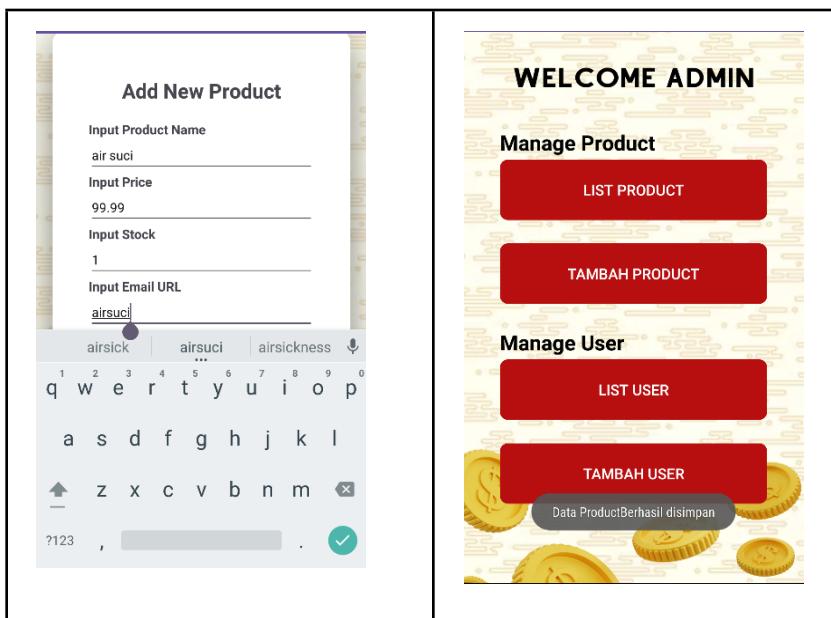
Gambar diatas merupakan tampilan untuk login. Tampilan admin akan muncul apabila user menginput username dengan role Admin pada database. Setiap admin di Hanying memiliki username yang unique dan password yang hanya diketahui oleh masing-masing Admin. Pada tampilan admin dibagi menjadi 2 bagian, yaitu Manage Product dan Manage User

3.2.1.2 Manage Product



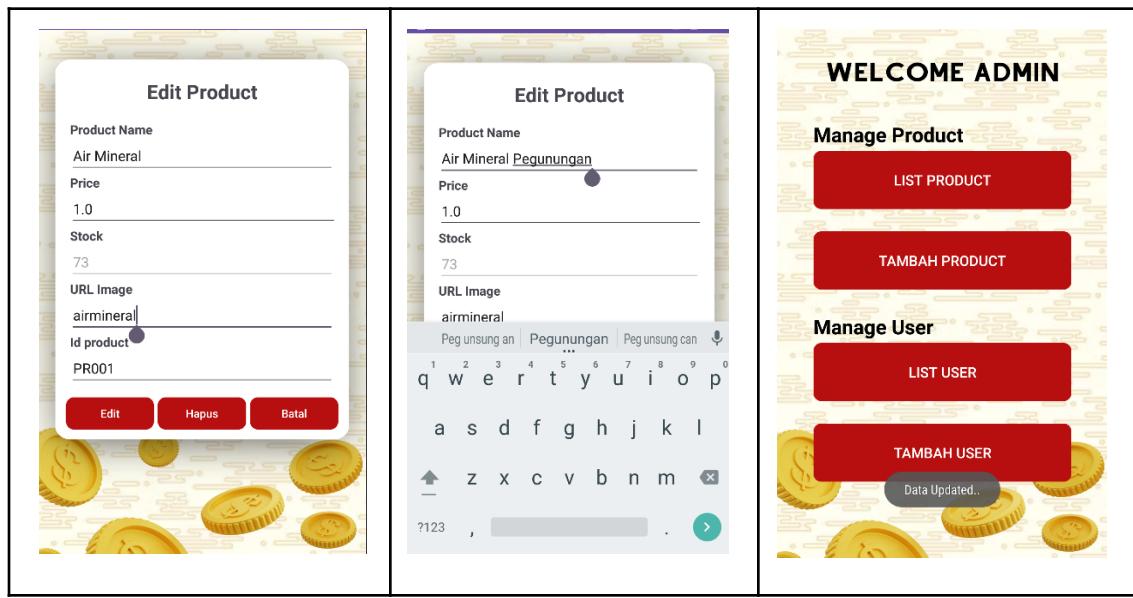
Gambar diatas adalah tampilan Manage Product, pada halaman ini admin dapat melihat semua produk yang ada pada database Hanying, data produk yang ada pada halaman ini akan muncul pada halaman menu di tampilan *user*

3.2.1.3 Add Product



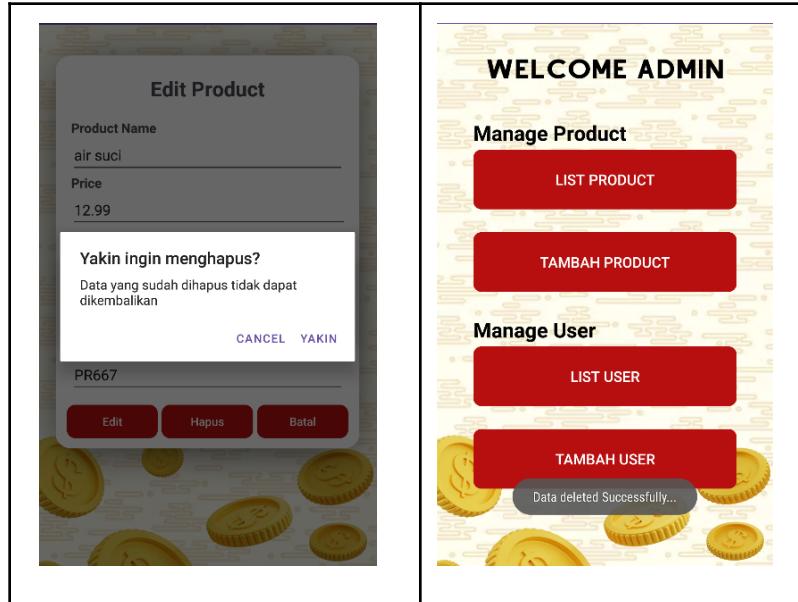
Gambar diatas merupakan tampilan Add Product, dimana admin dapat menambahkan produk baru dengan menginput nama produk, harga, stock dan URL. Jika admin sudah tersimpan maka system akan memunculkan toast “Berhasil Disimpan”

3.2.1.4 Edit Product



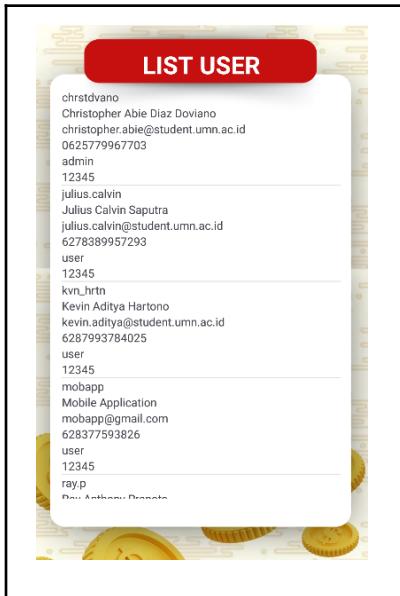
Gambar diatas merupakan tampilan Edit product, dimana admin dapat merubah Product Name, Price, URL Image dan ID Product. Setelah menekan tombol “Edit” maka perubahan akan tersimpan dan memunculkan toast “Data Terupload”

3.2.1.5 Delete Product



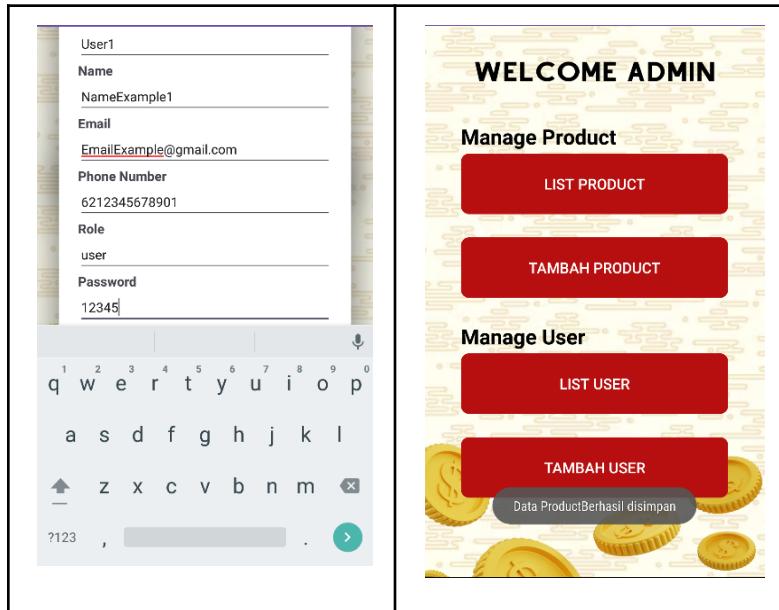
Gambar diatas merupakan jika admin ingin menghapus sebuah produk, jika menekan tombol “Hapus” maka akan memunculkan pop-up “Yakin Ingin Menghapus?” dan jika menekan tombol cancel maka akan balik, jika menekan tombol yakin maka akan menghapus produk tersebut dan memunculkan toast “Data deleted Successfully”.

3.2.1.6 Manage User



Gambar diatas merupakan tampilan Manage User, dimana admin dapat melihat list dari para pengguna aplikasi dan melihat data dari akun para pengguna.

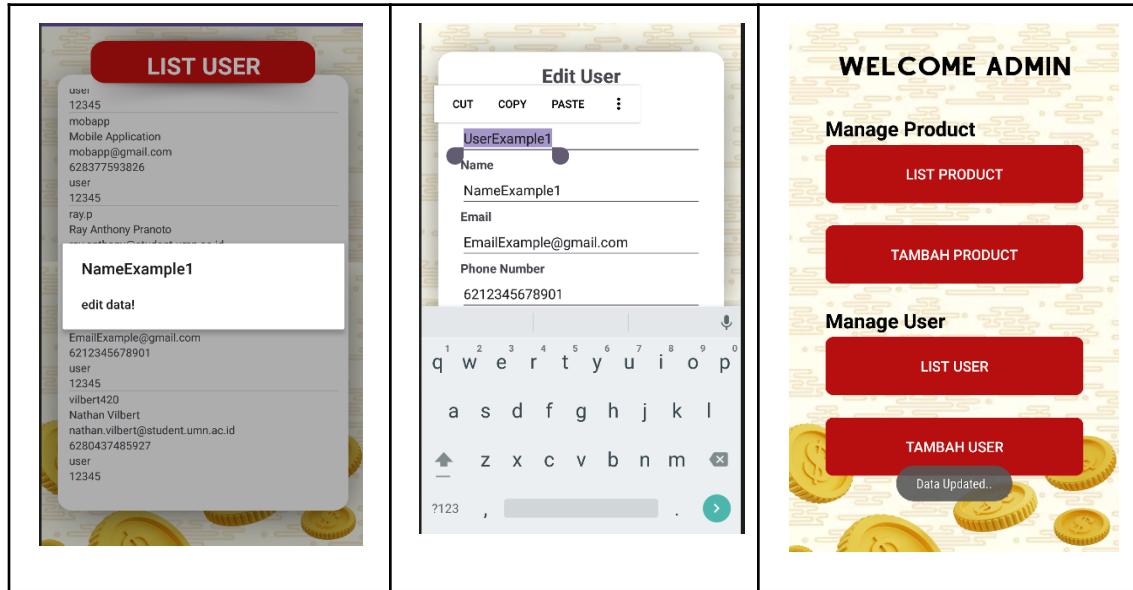
3.2.1.7 Add User



Gambar diatas merupakan tampilan Add User, dimana admin dapat melakukan penambahan pengguna dengan memasukan data seperti Nama,

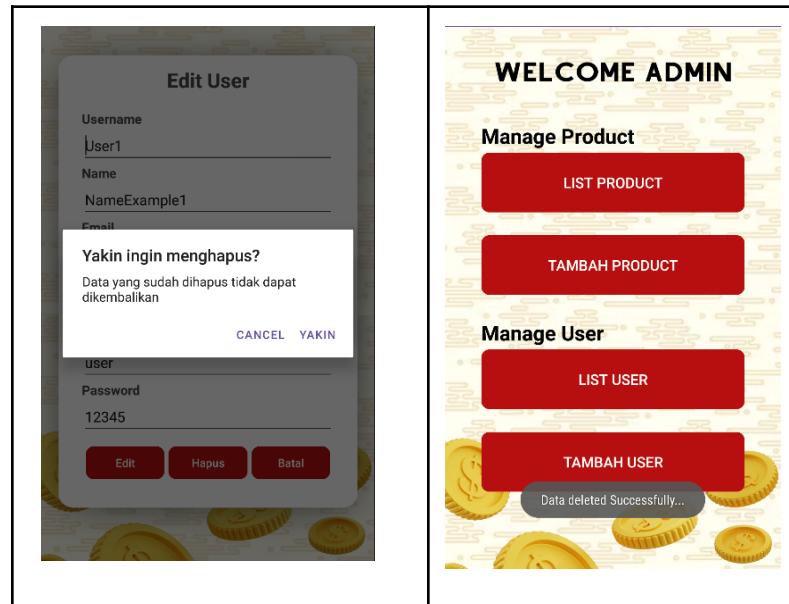
Email, Nomor Telepon, Role, serta password dari akun yang ingin dibuat. Setelah berhasil terbuat maka akan menunjukkan toast “Berhasil disimpan”.

3.2.1.8 Edit User



Gambar diatas merupakan tampilan Edit User, admin dapat merubah data dari akun para pengguna dan jika disimpan maka akan menunjukkan toast “Data Updated”.

3.2.1.9 Delete Product

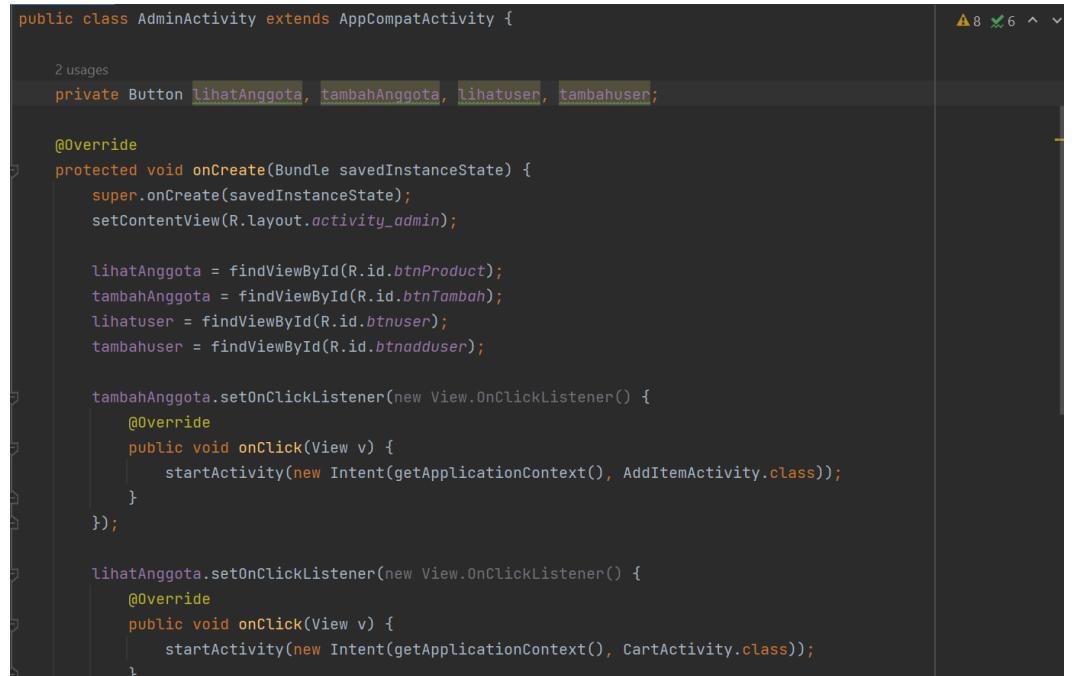


Gambar diatas merupakan jika admin ingin menghapus akun dari pengguna, jika menekan tombol “Hapus” maka akan memunculkan pop-up

“Yakin Ingin Menghapus?” dan jika menekan tombol cancel maka akan balik, jika menekan tombol yakin maka akan menghapus produk tersebut dan memunculkan toast “Data deleted Successfully”.

3.2.2 Code

3.2.2.1 Admin Dashboard (AdminActivity.java)



```
public class AdminActivity extends AppCompatActivity {

    2 usages
    private Button lihatAnggota, tambahAnggota, lihatuser, tambahuser;

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

        lihatAnggota = findViewById(R.id.btnProduct);
        tambahAnggota = findViewById(R.id.btnAdd);
        lihatuser = findViewById(R.id.btnuser);
        tambahuser = findViewById(R.id.btnadduser);

        tambahAnggota.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), AddItemActivity.class));
            }
        });

        lihatAnggota.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), CartActivity.class));
            }
        });
    }
}
```

Pada code diatas terdapat 4 button yang akan digunakan. Button pertama adalah button untuk lihatAnggota, button kedua adalah untuk tambah Anggota, button ketiga adalah untuk lihat user, dan button terakhir adalah untuk tambah User.

```
tambahAnggota.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), addItemActivity.class));
    }
});

lihatAnggota.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), CartActivity.class));
    }
});

lihatuser.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), userActivity.class));
    }
});

tambahuser.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), addUserActivity.class));
    }
});
```

Kemudian button lihat anggota begitu di click akan mengarahkan admin ke menu CartActivity yaitu berisi list list dari product aplikasi Hanying. Kemudian pada button Tambah anggota begitu di click maka akan mengarahkan admin ke menu addItemActivity yang berisi form untuk menambahkan product pada aplikasi Hanying. Kemudian ketika admin click pada button lihat user maka admin akan diarahkan ke menu userActivity yang berisi list list dari user yang telah membuat akun pada aplikasi Hanying dan terakhir ketika admin click button tambah user maka admin akan diarahkan ke menu addUserActivity yang berisi form untuk menambahkan akun dari user pada aplikasi Hanying.

3.2.2.2 Sistem Get & Set Product Admin (Product.java)

```
8 usages
public class product {
    3 usages
    private String productID, productName, image;
    3 usages
    private double price;
    3 usages
    private int stock;
```

Untuk langkah pertamanya adalah kita perlu untuk mendeklarasi kolom - kolom yang akan digunakan berdasarkan dengan tipe dan juga jenisnya. Berdasarkan gambar diatas dimana kolom “productID, productName, image” bertipe String, “price” bertipe double, dan “stock” bertipe int.

```
1 usage
public product(String productID, String productName, double price, int stock, String image) {
    this.productID = productID;
    this.productName = productName;
    this.price = price;
    this.stock = stock;
    this.image = image;
}
```

Dari code diatas memiliki arti adalah untuk menginisialisasi variable instan dengan nilai tertentu ketika object ‘product’ dibuat.

```
4 usages
public String getProductID() { return productID; }

no usages
public void setProductID(String productID) {this.productID = productID;}

5 usages
public String getProductName() { return productName; }

1 usage
public void setProductName(String productName) {this.productName = productName;}

3 usages
public double getPrice() { return price; }

1 usage
public void setPrice(double price) {this.price = price; }

3 usages
public int getStock() {return stock; }

1 usage
public void setStock(int stock) {this.stock = stock; }

3 usages
public String getImage() {return image; }

1 usage
public void setImage (String image) {this.image = image;}
```

Diatas merupakan sistem Getter & Setter. Berdasarkan gambar diatas Getter berfungsi untuk mengambil nilai variabel yang telah diinput admin kemudian Setter menentukan nilai variabel tersebut.

Secara keseluruhan product.java ini bertujuan untuk memodelkan data produk dengan menyimpan informasi seperti productID, productName, price, stock, dan image. Hal ini memungkinkan penggunaan kelas ini untuk merepresentasikan dan mengelola data produk dalam suatu program.

3.2.2.3 Konfigurasi Admin (konfigurasi.java)

```
+ usages
public class konfigurasi extends ArrayAdapter<product> {

    1 usage
    Context context;
    3 usages
    List<product> arraylistproduct;
```

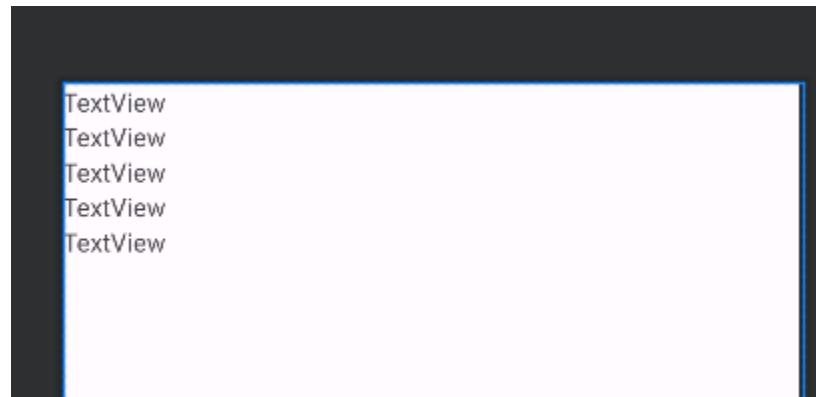
Pertama tama kita perlu untuk mendeklarasi beberapa yang diperlukan. Pada gambar diatas kita mendeklarasi context dan arraylistproduct untuk menampung listproduct pada bagian admin HanYing.

```
2 usages
public Konfigurasi(@NonNull Context context, List<Product> arrayListproduct) {
    super(context, R.layout.list_item, arrayListproduct);

    this.context = context;
    this.arrayListproduct = arrayListproduct;
}
```

Konstruktor kelas ini menerima konteks aplikasi Android dan daftar produk sebagai parameter. Konstruktor kelas induk ArrayAdapter (super) diinisialisasi dengan tata letak list_item dan daftar produk.

Berikut merupakan bentuk dari “list_item.xml”



Setiap TextView berisi tabel tabel yang akan kita gunakan yaitu table productID, productName, price, stock, dan image.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item, null, true);

    TextView tmpTulisanID = view.findViewById(R.id.edtproductID);
    TextView tmpTulisanProduct = view.findViewById(R.id.edtproductName);

    tmpTulisanID.setText(arrayListproduct.get(position).getProductID());
    tmpTulisanProduct.setText(arrayListproduct.get(position).getProductName());

    return view;
}
```

Pada gambar diatas digunakan untuk menginisialisasi tampilan setiap elemen dalam daftar produk. Setiap elemen direpresentasikan oleh objek View. Inflate dari kelas LayoutInflater digunakan untuk mengonversi tata letak XML (list_item.xml) menjadi objek View. Selanjutnya, dua teks view (TextView) yang sesuai dengan ID yang didefinisikan dalam tata letak (edtproductID dan edtproductName) diambil dan diisi dengan informasi produk sesuai dengan posisi dalam daftar.

Secara keseluruhan konfigurasi.java bertujuan untuk mengatur koneksi antara data produk (product) dan UI admin melalui tata letak list_item. Hal ini memudahkan penampilan data produk dalam suatu daftar (list) di dalam aplikasi Android.

3.2.2.4 Koneksi Database (conn.php)

```
 conn.php
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "hanying";
6
7 $conn = new mysqli($servername, $username, $password, $dbname);
8
9 if ($conn->connect_error) {
10 | die(json_encode(array("error" => "Connection failed: " . $conn->connect_error)));
11 }
12 ?>
```

Fungsi diatas adalah melakukan koneksi kedalam database yang kita gunakan. Pada gambar diatas kita menggunakan localhost sebagai servername, kemudian username nya adalah root, dan nama dari database yang kita miliki adalah “hanying”

Kemudian pada bagian \$conn berfungsi untuk membuat objek koneksi baru menggunakan kelas mysqli untuk berinteraksi dengan server MySQL. Jika koneksi berhasil, objek koneksi (\$conn) akan dibuat. Jika tidak, program akan berhenti dan mengirim pesan kesalahan.

Kemudian pada bagian if berfungsi ketika database tidak berhasil connect maka akan menampilkan kesalahan dan mengirimkan pesan bahwa database tidak berhasil connect.

3.2.2.5 Add Product (add_to_cart.php)

```
<?php  
header('Content-type: application/json; charset=utf-8');  
include "conn.php";
```

Code diatas berfungsi untuk menetapkan tipe konten sebagai JSON dan mengimpor file conn.php yang berisi konfigurasi koneksi ke database. Agar fungsi fungsi pada aplikasi HanYing dapat dibuat.

```
4  
5     if ((  
6         isset($_POST['productName']) &&  
7         isset($_POST['price']) &&  
8         isset($_POST['stock']) &&  
9         isset($_POST['image']))  
0     ) {  
1         // Kode untuk insert data  
2     } else {  
3         $response["success"] = -1;  
4         $response["message"] = "Data kosong";  
5         echo json_encode($response);  
6     }
```

Code diatas berfungsi untuk melakukan pemeriksaan apakah data yang dibutuhkan (productName, price, stock, dan image) ada dalam permintaan POST. Jika tidak, maka akan mengembalikan pesan JSON yang menunjukkan bahwa data kosong.

```

$productName = $_POST['productName'];
$price = $_POST['price'];
$stock = $_POST['stock'];
$image = $_POST['image'];

// Generate a unique productID (e.g., PR123)
$productID = generateProductID();

// Create a new MySQLi connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$stmt = $conn->prepare("INSERT INTO product(productID, productName, price, stock, image) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param("sssss", $productID, $productName, $price, $stock, $image);

```

Code diatas adalah kode yang berada ditengah tengah code if diatas. Jika data masukan valid, Maka system kemudian membuat koneksi baru ke database dan menyisipkan data produk ke dalam tabel "product" yang berisi kolom kolom seperti gambar diatas.. Sebuah prepared statement digunakan untuk menghindari serangan SQL injection.

```

$response = array();

if ($stmt->execute()) {
    $response["success"] = 1;
    $response["message"] = "Data berhasil ditambah";
    echo json_encode($response);
} else {
    $response["success"] = 0;
    $response["message"] = "Data gagal ditambahkan";
    echo json_encode($response);
}

```

Code diatas berfungsi untuk memberikan respon dalam bentuk JSON berdasarkan hasil dari operasi dari fungsi PHP diatas.

```

function generateProductID() {
    $prefix = "PR";
    $uniqueNumber = mt_rand(100, 999);
    return $prefix . $uniqueNumber;
}
?>

```

Kemudian code diatas berfungsi untuk melakukan generateProductID. Ketika admin menambahkan product di aplikasi HanYing maka system akan melakukan generate ID dengan 5 digit. 2 digit pertama bertulisan “PR” dan 3 digit terakhir angka random 1 hingga 9.

3.2.2.6 Menambahkan Produk/Item Baru (AddItemActivity.java)

```
5 usages
public class AddItemActivity extends AppCompatActivity {
    5 usages
    ProgressBar progressBar;
    2 usages
    EditText edtproductName, edtPrice, edtStock, edtImage;

    2 usages
    Button btnTambah;
    no usages
    String productID, productName, image;
    2 usages
    double price;
    2 usages
    int stock;

    1 usage
    String url_tambah_makanan = "http://192.168.1.8/hanying/add_to_cart.php";
```

Langkah awal kita perlu untuk mendeklarasi bagian bagian apa saja yang akan kita gunakan. Pada code diatas saya menggunakan progressBar, editText, button, dan mendeklarasi kolom kolom pada table “product” yang akan digunakan.

```
1 usage
String url_tambah_makanan = "http://192.168.30.202/hanying/add_to_cart.php";
```

String url diatas merupakan link pada PHP yang telah kita buat sebelumnya. Didalam berisikan fungsi fungsi agar sistem menambahkan product dapat berjalan dengan lancar. Kemudian ditengah link tersebut terdapat ipAddress. Ip Address tersebut merupakan Ip Address pada koneksi kita sehingga disesuaikan koneksi kita tersambung ke saluran koneksi internet yang mana.

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

    edtproductName = findViewById(R.id.edtProductName);
    edtPrice = findViewById(R.id.edtPrice);
    edtStock = findViewById(R.id.edtStock);
    btnTambah = findViewById(R.id.btnAddToCart);
    edtImage = findViewById(R.id.edtImage);
    progressBar = findViewById(R.id.progressBar);
```

Pada code diatas kita melakukan inisialisasi variabel untuk (UI) yang yang terdapat pada layout XML activity_add_item.xml. Dimana fungsi dari findViewById digunakan untuk mendapatkan referensi dari setiap komponen berdasarkan ID-nya.

```
btnTambah.setOnClickListener(v -> {
    productName = edtproductName.getText().toString();
    price = Double.parseDouble(edtPrice.getText().toString());
    stock = Integer.parseInt(edtStock.getText().toString());
    image = edtImage.getText().toString();
```

Pada code di atas Data yang diinputkan olehadmin diambil dari komponen EditText dan disimpan dalam variable yang sesuai dengan nama kolom - kolomnya. Kemudian editText Mengharuskan data berbentuk String oleh sebab itu beberapa data yang tidak String dapat kita convert kedalam bentuk String seperti code diatas.

```

RequestQueue queue = Volley.newRequestQueue(context: addItemActivity.this);
StringRequest stringRequest = new StringRequest(Request.Method.POST,
    url_tambah_makanan, response -> {
    try {
        JSONObject jObj = new JSONObject(response);
        int sukses = jObj.getInt(name: "success");
        if (sukses == 1) {
            Toast.makeText(context: addItemActivity.this, text: "Data Product" +
                "Berhasil disimpan", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
            finish();
        }
        progressBar.setVisibility(View.GONE);
    } catch (Exception ex) {
        Log.e(tag: "Error", ex.toString());
        progressBar.setVisibility(View.GONE);
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e(tag: "Error: ", error.getMessage());
        Toast.makeText(context: addItemActivity.this, text: "Silahkan cek koneksi" +
            "internet Anda!", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.GONE);
    }
}

```

Kode ini menggunakan library Volley untuk membuat permintaan POST ke server berdasarkan alamat link pada add_to_cart.php. Sebuah StringRequest dibuat dengan mendefinisikan URL tujuan, menangani respon sukses dan respon error.

Kemudian pada code try dan catch berfungsi setelah menerima respon dari server, kode mencoba mengurai response JSON. Jika respon sukses (success == 1), pesan toast akan ditampilkan, dan aplikasi akan beralih kembali ke halaman utama (MainActivity). Jika terjadi kesalahan, pesan kesalahan akan dicatat di log yang bisa kita lihat pada logcat.

Kemudian pada bagian OnErrorResponse, jika terjadi kesalahan saat melakukan permintaan, metode onResponse akan dipanggil. Kesalahan tersebut dicatat di log, dan pesan kesalahan ditampilkan kepada pengguna melalui toast .

```

        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<>();
            params.put("productName", productName);
            params.put("price", String.valueOf(price));
            params.put("stock", String.valueOf(stock));
            params.put("image", image);

            return params;
        }

        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("Content-type", "application/x-www-form-urlencoded");
            return params;
        }
    };
    queue.getCache().clear();
    queue.add(stringRequest);

}

```

Code diatas pada bagian getParams digunakan untuk menentukan parameter yang akan dikirimkan dalam permintaan, dan getHeaders mengatur header permintaan. Kedua metode ini digunakan untuk menentukan parameter yang akan dikirimkan dalam permintaan POST (getParams) dan mengonfigurasi header permintaan (getHeaders). Parameter yang diatur mencakup productName, price, stock, dan image.

Secara keseluruhan Kode AddItemActivity bertujuan untuk memungkinkan admin menambahkan produk baru ke dalam database dengan mengirimkan permintaan ke server menggunakan protokol HTTP POST. Respon dari server kemudian diolah untuk memberikan umpan balik kepada admin, baik itu berhasil atau tidak.

3.2.2.7 Fungsi Menampilkan list Product (get_cart_items.php)

```

<?php
header('Content-type: application/json; charset=utf-8');
include "conn.php";

```

Code diatas berfungsi untuk menetapkan tipe konten sebagai JSON dan mengimpor file conn.php yang berisi konfigurasi koneksi ke database MySQL.

```
$q = mysqli_query($conn, "SELECT * FROM product");
$response = array();
```

Code diatas berfungsi untuk menjalankan query SQL untuk mengambil semua data dari tabel "product" menggunakan koneksi database yang telah diatur sebelumnya.

```
if (mysqli_num_rows($q) > 0) {
    $response["data"] = array();
    while ($r = mysqli_fetch_array($q)) {
        $product = array();
        $product["productID"] = $r["productID"];
        $product["productName"] = $r["productName"];
        $product["price"] = $r["price"];
        $product["stock"] = $r["stock"];
        $product["image"] = $r["image"];
        array_push($response["data"], $product);
    }
    $response["Success"] = 1;
    $response["Message"] = "Data product berhasil dibaca";
    echo json_encode($response);
}
```

Code diatas berfungsi untuk memeriksa apakah hasil query mengembalikan lebih dari 0 baris data. Jika ya, maka setiap baris hasil query diproses dalam loop while. Informasi setiap produk, seperti productID, productName, price, stock, dan image, diambil dan dimasukkan ke dalam array \$product. Array \$product kemudian dimasukkan ke dalam array "data". Setelah loop selesai, respons JSON dibentuk dengan menetapkan "Success" menjadi 1, yang berarti operasi berhasil, dan memberikan pesan sukses.

```
        echo json_encode($response),
    } else {
        $response["Success"] = 0;
        $response["Message"] = "Tidak ada data";
        echo json_encode($response);
    }

    // Close the connection if needed
    mysqli_close($conn);
?>
```

Jika hasil query tidak mengembalikan baris data, maka respons JSON dibentuk dengan menetapkan "Success" menjadi 0, yang menunjukkan kegagalan atau tidak adanya data, dan memberikan pesan bahwa tidak ada data. Kemudian pada bagian mysqli_close berfungsi untuk menutup koneksi ke database MySQL. Menutup koneksi setelah selesai dengan operasi database dapat membantu mengelola sumber daya dengan lebih efisien.

Secara keseluruhan fungsi code pada PHP diatas berfungsi untuk memberikan respons JSON yang berisi data produk dari tabel "product" di database MySQL. Jika ada data, respons akan berisi daftar produk beserta informasinya dan pesan sukses. Jika tidak ada data, respons akan berisi pesan bahwa tidak ada data dan indikasi kegagalan. Respons ini dapat diambil oleh aplikasi klien untuk menampilkan informasi produk kepada pengguna.

3.2.2.8 Show list product (CartActivity.java)

```
4 usages
ProgressBar progressBar;
5 usages
konfigurasi adapter;

17 usages
public static ArrayList<product> arraylistproduct = new ArrayList<>();

2 usages
product product;

6 usages
ListView lv;

3 usages
ArrayList<HashMap<String, String>> list_makanan;
```

Pertama kita perlu untuk melakukan deklarasi fungsi fungsi apa saja yang ingin kita gunakan. Pada code diatas terdapat progressBar, konfigurasi, ArrayListproduct untuk menampung data dalam bentuk array, kemudian product yang berisi fungsi sistem Getter dan Setter, kemudian listview yang kita berikan nama “lv” yang berfungsi nama dari fungsi untuk menampilkan data dalam bentuk list, kemudian terdapat hashmap yang berfungsi untuk menyimpan dan mengakses data, pada code diatas haspmap diberikan ama list_makanan.

```
1 usage
String url_get_product = "http://192.168.30.202/hanying/get_cart_items.php";

1 usage
private static final String TAG_PRODUCT = "data";
2 usages
private static final String TAG_ID = "productID";
2 usages
private static final String TAG_NAME = "productName";
2 usages
private static final String TAG_PRICE = "price";
2 usages
private static final String TAG_STOCK = "stock";
2 usages
private static final String TAG_IMAGE = "image";
```

Url_get_product merupakan link fungsi PHP yang berguna untuk menampilkan data data dari product HanYing kedalam android studio.

Kemudian url tersebut menjadi kunci dalam permintaan HTTP untuk mengambil data produk dari server, dan konstanta-konstanta string (TAG_PRODUCT, TAG_ID, TAG_NAME, TAG_PRICE, TAG_STOCK, dan TAG_IMAGE) digunakan dalam ekstraksi informasi dari respons JSON.

```
1 usage
private void mendapatkanData() {
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();

    if (networkInfo == null || !networkInfo.isConnected()) {
        Toast.makeText(context: CartActivity.this, text: "No internet Connection!", Toast.LENGTH_SHORT).show();
        return;
    }
}
```

Code diatas menggunakan ConnectivityManager untuk memeriksa ketersediaan koneksi internet. Jika tidak terhubung, pesan "No internet Connection!" akan ditampilkan melalui Toast dan metode berhenti.

```
}
StringRequest stringRequest = new StringRequest(Request.Method.POST, url_get_product,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {

            arraylistproduct.clear();
            try {
                JSONObject jObj = new JSONObject(response);
                JSONArray member = jObj.getJSONArray(TAG_PRODUCT);

                for (int i = 0; i < member.length(); i++) {
                    JSONObject a = member.getJSONObject(i);

                    String productID = a.getString(TAG_ID);
                    String productName = a.getString(TAG_NAME);
                    double price = a.getDouble(TAG_PRICE);
                    int stock = a.getInt(TAG_STOCK);
                    String image = a.getString(TAG_IMAGE);

                    HashMap<String, String> map = new HashMap<>();
                    map.put("productID", productID);
                    map.put("productName", productName);
                    map.put("price", String.valueOf(price));
                    map.put("stock", String.valueOf(stock));
                    map.put("image", image);
                    list_makanan.add(map);

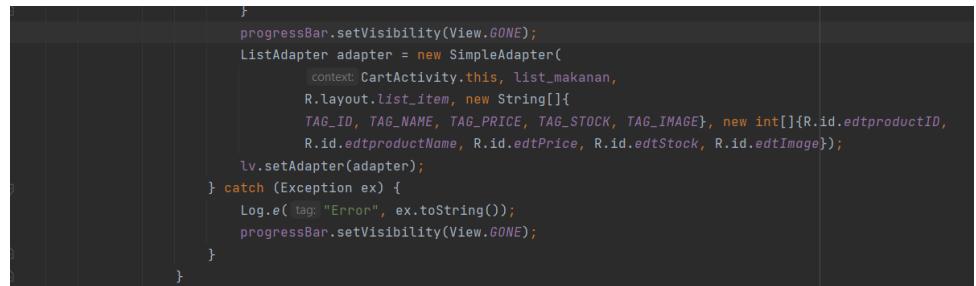
                    product = new product(productID, productName, price, stock, image);
                    arraylistproduct.add(product);
                    adapter.notifyDataSetChanged();
                }
            }
        }
    }
}
```

Pada code diatas merupakan implementasi permintaan data dari server menggunakan StringRequest dalam metode onResponse() dari Volley. Ketika respons dari server diterima, data produk yang dihasilkan dalam format JSON

dipecah menjadi objek-objek individual. Pertama-tama, ArrayList arraylistproduct dibersihkan dengan menggunakan metode clear() untuk memastikan bahwa data yang baru akan ditambahkan.

Selanjutnya, dilakukan iterasi melalui JSONArray member yang mengandung informasi produk. Setiap objek JSON di dalam array diambil dan nilai-nilai atribut seperti productID, productName, price, stock, dan image diekstrak. Objek HashMap baru dibuat, dan nilai-nilai ini dimasukkan ke dalamnya menggunakan metode put(). Selanjutnya, objek HashMap ini ditambahkan ke dalam ArrayList list_makanan.

Selain itu, objek product juga dibuat menggunakan nilai-nilai yang diekstrak dan ditambahkan ke dalam ArrayList arraylistproduct. Setelah setiap iterasi, metode adapter.notifyDataSetChanged() dipanggil untuk memberitahu adapter bahwa data telah berubah dan tampilan harus diperbarui.



```
        }
        progressBar.setVisibility(View.GONE);
       ListAdapter adapter = new SimpleAdapter(
            context: CartActivity.this, list_makanan,
            R.layout.list_item, new String[]{TAG_ID, TAG_NAME, TAG_PRICE, TAG_STOCK, TAG_IMAGE}, new int[]{R.id.edtproductID, R.id.edtproductName, R.id.edtPrice, R.id.edtStock, R.id.edtImage});
        lv.setAdapter(adapter);
    } catch (Exception ex) {
        Log.e( tag: "Error", ex.toString());
        progressBar.setVisibility(View.GONE);
    }
}
```

Pada Kode diatas berfungsi untuk menangani bagian akhir dari permintaan data ke server dalam metode onResponse(). Setelah selesai mendapatkan dan mengolah data produk dari server, ProgressBar (progressBar) diatur menjadi tidak terlihat (View.GONE) untuk menunjukkan bahwa operasi pengambilan data sudah selesai.

Selanjutnya, sebuah objek ListAdapter baru dibuat menggunakan kelas SimpleAdapter. Adapter ini berfungsi sebagai perantara antara data (list_makanan) dan tampilan (R.layout.list_item) yang akan ditampilkan di ListView (lv). Setiap elemen dalam data akan dihubungkan dengan elemen-elemen tampilan menggunakan kunci-kunci yang telah ditentukan,

yaitu TAG_ID, TAG_NAME, TAG_PRICE, TAG_STOCK, dan TAG_IMAGE.

Kemudian, adapter baru tersebut diatur untuk digunakan oleh ListView dengan memanggil lv.setAdapter(adapter). Hal ini mengakibatkan pembaruan tampilan dengan data yang baru. Jika ada kesalahan selama proses ini, blok catch akan menangkap dan mencatat pesan kesalahan menggunakan Log.e(), dan ProgressBar diatur kembali menjadi tidak terlihat untuk menyelesaikan proses.

```
    @Override
    public void onErrorResponse(VolleyError error) {
        if (error.getMessage() != null) {
            Log.e(tag, "Error: ", error.getMessage());
        } else {
            Log.e(tag, "Error: ", msg: "Unknown error occurred");
        }
        Toast.makeText(context: CartActivity.this, text: "Silahkan cek koneksi Internet Anda!!", Toast.LENGTH_SHORT).show();
        finish();
    });
}
```

Code diatas berfungsi untuk mendeklarasi metode `onErrorResponse()`, jika terdapat pesan kesalahan (`error.getMessage()` tidak null), pesan tersebut dicatat menggunakan `Log.e()`. Jika tidak ada pesan kesalahan yang dapat diambil, pesan "Unknown error occurred" dicatat. Selanjutnya, pengguna diberitahu melalui `Toast` bahwa terjadi masalah koneksi internet, dan proses aplikasi diakhiri (`finish()`) untuk menghindari operasi yang tidak akurat tanpa koneksi yang stabil.

```
lv.setAdapter(lv.getAdapter());
lv.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        AlertDialog.Bulder builder = new AlertDialog.Bulder(view.getContext());
        ProgressDialog progressDialog = new ProgressDialog(view.getContext());

        adapter.notifyDataSetChanged();

        CharSequence[] dialogitem = {"Edit data!"};
        builder.setTitle(arraylistproduct.get(position).getProductName());
        builder.setItems(dialogitem, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                startActivity(new Intent(getApplicationContext(), editItemActivity.class));
                putExtra( name: "position", position);
                adapter.notifyDataSetChanged();
            }
        });
        builder.create().show();
        return false;
    }
});
RequestQueue queue = Volley.newRequestQueue( context: this);
queue.add(stringRequest);
}
}
```

Pada Code diatas lv.setAdapter(lv.getAdapter()) digunakan untuk merefresh tampilan ListView, dan lv.setOnItemLongClickListener() menangani long click pada item. Saat terjadi long click, dialog konfirmasi ditampilkan dengan satu opsi "Edit data!". Jika opsi tersebut dipilih, aktivitas editItemActivity dimulai dengan menyertakan posisi item. Adapter juga diperbarui setelah pengeditan. Kode ini memberikan fungsionalitas pengeditan data produk melalui long click dengan antarmuka yang responsif.

3.2.2.9 Fungsi Edit Product (edit_cart_items.php)

```
<?php
header('Content-type: application/json; charset=utf-8');
include "conn.php";
```

Pada kode diatas pertama menetapkan header response sebagai JSON dan melakukan inklusi file conn.php yang berisi informasi koneksi ke database.

```
if (isset($_POST['productID']) && isset($_POST['productName']) && isset($_POST['price'])&& isset($_POST['stock'])&& isset($_POST['image'])) {
    $productID = $_POST['productID'];
    $productName = $_POST['productName'];
    $price = $_POST['price'];
    $stock = $_POST['stock'];
    $image = $_POST['image'];

    // Use mysqli instead of mysql, and prepared statements to prevent SQL injection
    $stmt = mysqli_prepare($conn, "UPDATE product SET productName=?, price=?, stock=?, image=? WHERE productID=?");
    mysqli_stmt_bind_param($stmt, "sdiss", $productName, $price, $stock, $image, $productID);

    $response = array();

    if (mysqli_stmt_execute($stmt)) {
        $response["Success"] = 1;
        $response["Message"] = "Data berhasil diupdate";
        echo json_encode($response);
    } else {
        $response["Success"] = 0;
        $response["Message"] = "Data gagal diupdate";
        echo json_encode($response);
    }

    // Close the statement
    mysqli_stmt_close($stmt);
}
```

Kode PHP di atas berfungsi untuk menangani permintaan update data produk dari aplikasi Android secara ringkas dan aman. Pengecekan awal dilakukan terhadap keberadaan seluruh data yang dibutuhkan dalam permintaan POST. Jika data tersedia, nilai-nilai tersebut diambil untuk digunakan dalam prepared statement.

Prepared statement digunakan untuk menjalankan operasi update data pada tabel produk di database MySQL dengan memperhatikan keamanan terhadap serangan SQL injection. Setelah eksekusi, dilakukan pengecekan keberhasilan operasi update. Respons JSON dibuat sesuai dengan hasilnya, mencakup indikator keberhasilan dan pesan yang menjelaskan status operasi.

```
    mysqli_stmt_close($stmt);
} else [
    $response["productID"] = isset($_POST['productID']);
    $response["productName"] = isset($_POST['productName']);
    $response["price"] = isset($_POST['price']);
    $response["stock"] = isset($_POST['stock']);
    $response["image"] = isset($_POST['image']);
    $response["productID"] = $_POST['productID'];
    $response["productName"] = $_POST['productName'];
    $response["price"] = $_POST['price'];
    $response["stock"] = $_POST['stock'];
    $response["image"] = $_POST['image'];
    $response["Success"] = -1;
    $response["Message"] = "Data Kosong";
    echo json_encode($response);
]

// Close the connection if needed
mysqli_close($conn);
?>
```

Pada kode diatas berfungsi untuk menangani kondisi ketika data yang dibutuhkan untuk operasi update tidak lengkap dalam permintaan POST. Jika salah satu data, seperti `productID`, `productName`, `price`, `stock`, atau `image`, tidak tersedia, respons JSON dibuat dengan menyertakan informasi data yang kurang. Indikator kegagalan diatur menjadi -1, dan pesan menyatakan "Data Kosong". Selanjutnya, koneksi ke database MySQL ditutup. Ini memastikan bahwa sumber daya database dibebaskan setelah selesai digunakan.

3.2.2.10 Fungsi Delete Product (delete_item.php)

```
delete_item.php
1 <?php
2 header('Content-type: application/json; charset=utf-8');
3 include "conn.php";
4
5 if (isset($_POST['productID'])) {
6     $productID = $_POST['productID'];
7
8     // Use mysqli instead of mysql
9     $stmt = mysqli_prepare($conn, "DELETE FROM product WHERE productID=?");
10    mysqli_stmt_bind_param($stmt, "s", $productID);
11
12    $response = array();
13
14    if (mysqli_stmt_execute($stmt)) {
15        $response["Success"] = 1;
16        $response["Message"] = "Data berhasil dihapus";
17        echo json_encode($response);
18    } else {
19        $response["Success"] = 0;
20        $response["Message"] = "Data gagal dihapus";
21        echo json_encode($response);
22    }
23
24    // Close the statement
25    mysqli_stmt_close($stmt);
26 } else {
27     $response["Success"] = -1;
28     $response["Message"] = "Data Kosong";
29     echo json_encode($response);
30 }
31
32 // Close the connection if needed
33 mysqli_close($conn);
34 ?>
```

Kode PHP di atas digunakan untuk mengelola penghapusan data produk dari sebuah database. Pertama, skrip menetapkan header respons sebagai JSON dan mengimpor file `conn.php` yang berisi detail koneksi database.

Selanjutnya, skrip melakukan pengecekan apakah data produk, yaitu `productID`, telah disertakan dalam permintaan POST. Jika ya, skrip menggunakan prepared statement untuk menghapus data produk dari tabel `product` berdasarkan `productID`.

Setelah penghapusan, skrip memberikan respons dalam format JSON, mencakup status keberhasilan operasi dan pesan yang sesuai. Jika data produk tidak disertakan, skrip memberikan respons bahwa data kosong.

Terakhir, skrip menutup prepared statement dan koneksi database untuk menjaga efisiensi. Kode ini dirancang untuk diintegrasikan sebagai bagian dari layanan web yang menyediakan antarmuka untuk mengelola data produk dalam konteks aplikasi atau situs web tertentu.

3.2.2.11 Edit Product (editItemActivity.java)

```
20 usages
public class editItemActivity extends AppCompatActivity {

    6 usages
    konfigurasi adapter;
    15 usages
    public static ArrayList<product> arraylistProduct = new ArrayList<>();
    8 usages
    EditText edtproductName, edtPrice, edtStock, edtImage, editTextId;
    2 usages
    Button btnUpdate, btnHapus, btnKembali;
    1 usage
    String url_update_makanan = "http://192.168.30.202/hanying/edit_cart_item.php";
    1 usage
    String url_delete_makanan = "http://192.168.30.202/hanying/delete_item.php";
    25 usages
    private int position;
    no usages
    private AlertDialog.Builder alertDialogBuilder;
    no usages
    String productID, productName, image;
    no usages
    double price;
    no usages
    int stock;
    no usages
    product product;
```

Pada code diatas berisi mengenai Deklarasi variabel adapter, ArrayList arraylistProduct, dan beberapa komponen UI dilakukan, termasuk EditText (edtproductName, edtPrice, edtStock, edtImage, dan editTextId) dan Button (btnUpdate, btnHapus, dan btnKembali). Dua URL didefinisikan untuk mengupdate (url_update_makanan) dan menghapus (url_delete_makanan) data produk.

Variabel position menyimpan posisi produk yang diolah, dan alertDialogBuilder digunakan untuk membuat dialog konfirmasi. Terdapat pula variabel-variabel untuk menyimpan informasi produk seperti productID, productName, image, price, dan stock.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Intent intent = getIntent();  
    if (intent != null && intent.getExtras() != null) {  
        position = intent.getExtras().getInt("position", defaultValue: -1);  
    } else {  
        finish();  
        return;  
    }  
    setContentView(R.layout.activity_edit_item);  
    btnUpdate = findViewById(R.id.btnEdit);  
    btnHapus = findViewById(R.id.btnDelete);  
    btnKembali = findViewById(R.id.btnCancel);  
    edtproductName = findViewById(R.id.edtNama);  
    edtPrice = findViewById(R.id.edtPrice);  
    edtStock = findViewById(R.id.edtStock);  
    edtImage = findViewById(R.id.edtImage);  
    editTextId = findViewById(R.id.id_product);  
  
    edtproductName.setText(CartActivity.arraylistproduct.get(position).getProductName());  
    edtPrice.setText(String.valueOf(CartActivity.arraylistproduct.get(position).getPrice()));  
    edtStock.setText(String.valueOf(CartActivity.arraylistproduct.get(position).getStock()));  
    edtImage.setText(CartActivity.arraylistproduct.get(position).getImage());  
    editTextId.setText(CartActivity.arraylistproduct.get(position).getProductID());
```

Pada code diatas pertama-tama, aktivitas mengambil data ekstra dari intent yang memulai aktivitas ini. Jika data ekstra ditemukan, nilai dari kunci "position" diambil; jika tidak, aktivitas ditutup.

Selanjutnya, tata letak aktivitas diatur menggunakan metode `setContentView` dengan merujuk ke layout XML "activity_edit_item". UI seperti tombol ('btnUpdate', 'btnHapus', 'btnKembali') dan EditText ('edtproductName', 'edtPrice', 'edtStock', 'edtImage', 'editTextId') diinisialisasi dengan merujuk ke elemen yang sesuai pada tata letak.

Nilai-nilai EditText diisi dengan data produk yang akan diedit, yang diambil dari ArrayList `CartActivity.arraylistproduct` menggunakan indeks (`position`). Ini memungkinkan admin melihat dan mengedit informasi produk sebelum menyimpan perubahan atau menghapus produk dari list produk.

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (edtproductName.getText().toString().isEmpty()) {
            Toast.makeText(context, editItemActivity.this,
                text: "please enter your productName...", Toast.LENGTH_SHORT).show();
            return;
        } else if (edtPrice.getText().toString().isEmpty()) {
            Toast.makeText(context, editItemActivity.this,
                text: "please input price...", Toast.LENGTH_SHORT).show();
        } else if (edtStock.getText().toString().isEmpty()) {
            Toast.makeText(context, editItemActivity.this,
                text: "please input stock", Toast.LENGTH_SHORT).show();
        } else if (edtImage.getText().toString().isEmpty()) {
            Toast.makeText(context, editItemActivity.this,
                text: "please enter your URL Image...", Toast.LENGTH_SHORT).show();
        } else
            callPUTDataMethod(edtproductName.getText().toString(),
                Double.parseDouble(edtPrice.getText().toString()),
                Integer.parseInt(edtStock.getText().toString()),
                edtImage.getText().toString(),
                editTextId.getText().toString());
    }
});
```

Pada code diatas berfungsi untuk membuat fungsi pada tombol btnUpdate. Ketika tombol tersebut diklik, metode onClick akan dijalankan. Pada metode ini, dilakukan pemeriksaan apakah input dari pengguna valid. Jika ada input yang kosong, akan ditampilkan pesan kesalahan menggunakan Toast. Jika semua input valid, maka metode callPUTDataMethod akan dipanggil dengan mengirimkan informasi produk yang baru, seperti nama produk, harga, stok, dan URL gambar. Dengan melakukan ini, data produk dapat diperbarui sesuai dengan nilai yang dimasukkan oleh pengguna.

```
    | usage
    private void callPUTDataMethod(String productName, double price, int stock, String image, String productID) {
        final ProgressDialog progressDialog = new ProgressDialog(context: this);

        StringRequest request = new StringRequest(Request.Method.POST, url_update_makanan,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Log.d(tag: "DEBUG", msg: "response" + response);
                    adapter.notifyDataSetChanged();
                    try {
                        JSONObject jsonObject = new JSONObject(response);
                        edtproductName.setText("");
                        edtPrice.setText("");
                        edtStock.setText("");
                        edtImage.setText("");
                        editTextId.setText("");

                        Toast.makeText(context: editItemActivity.this, text: "Data Updated..", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(), MainActivity.class));

                        // Update local data after the update
                        if (!arraylistProduct.isEmpty() && position < arraylistProduct.size()) {
                            arraylistProduct.get(position).setProductName(productName);
                            arraylistProduct.get(position).setPrice(price);
                            arraylistProduct.get(position).setStock(stock);
                            arraylistProduct.get(position).setImage(image);

                            // Update EditText widgets with new data
                            populateData();
                        }
                    } catch (JSONException e) {
                        Log.e(tag: "DEBUG", msg: "Invalid position or empty arraylistProduct");
                    }
                }
            }
        );
    }
}
```

```
    } else {
        Log.e(tag: "DEBUG", msg: "Invalid position or empty arraylistProduct");
    }

    finish();
} catch (JSONException jsonObject) {
    jsonObject.printStackTrace();
}
}
```

Pada code diatas metode callPUTDataMethod dalam kelas editItemActivity digunakan untuk mengirim permintaan HTTP POST ke server dengan tujuan memperbarui data makanan berdasarkan input pengguna. Pada awalnya, sebuah objek ProgressDialog dibuat untuk menampilkan indikator selama proses pengiriman data. Selanjutnya, dibuat objek StringRequest yang melakukan permintaan POST ke URL yang telah ditentukan.

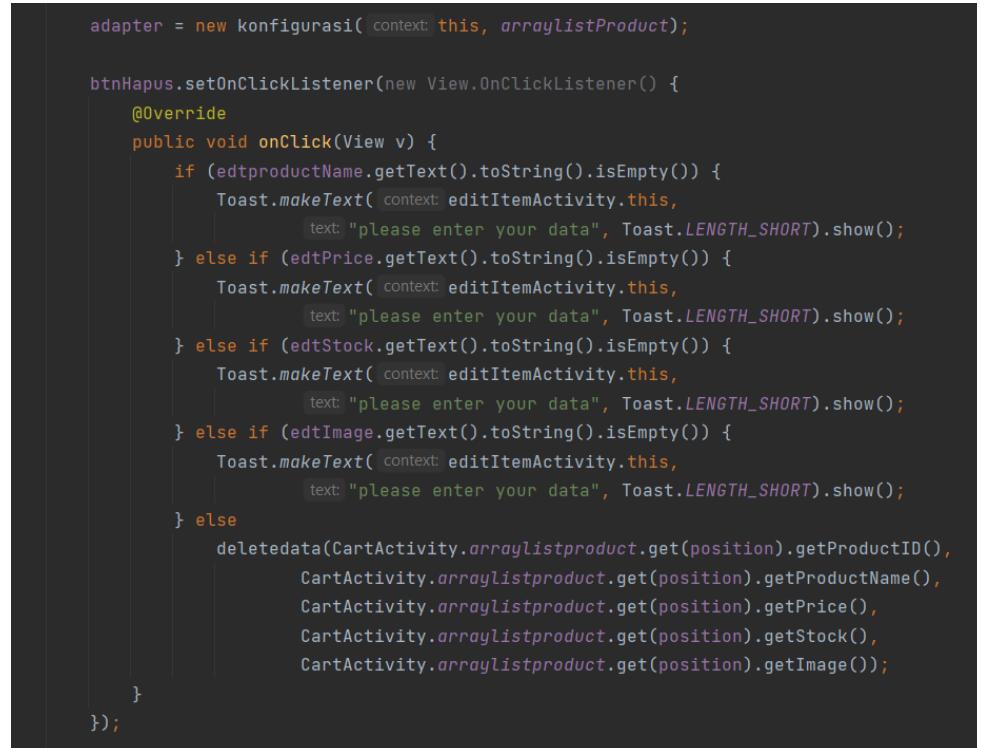
Metode onResponse akan dijalankan ketika respon berhasil diterima dari server. Pada tahap ini, dilakukan pemrosesan JSON untuk menangani hasil dari permintaan update. Nilai-nilai di beberapa elemen antarmuka pengguna, seperti nama produk (edtproductName), harga (edtPrice), stok (edtStock), dan URL gambar (edtImage), dibersihkan setelah pembaruan berhasil. Pesan toast

juga ditampilkan untuk memberi tahu pengguna bahwa data telah berhasil diperbarui.

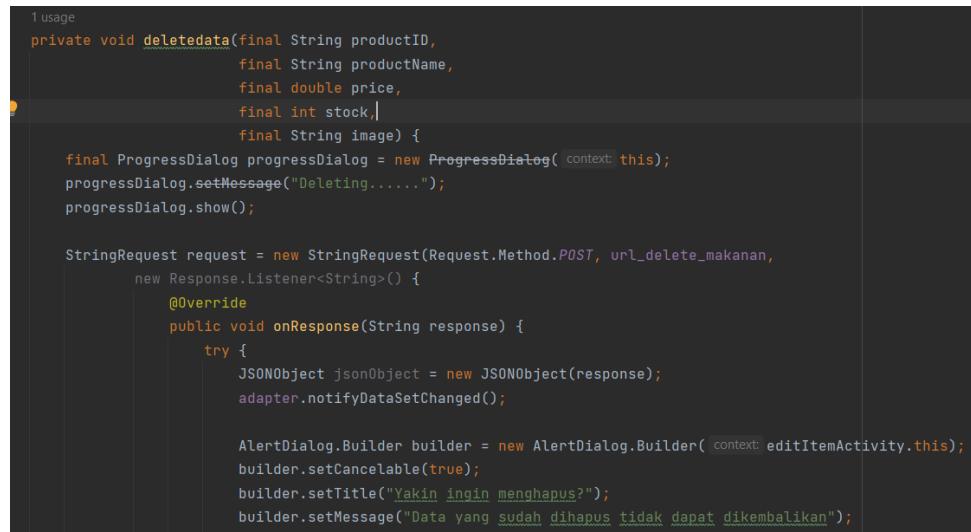
Selanjutnya, pengguna diarahkan kembali ke MainActivity, dan data lokal dalam arraylistProduct diperbarui berdasarkan input yang baru. Metode populateData dipanggil untuk memperbarui widget EditText dengan nilai yang baru, dan aktivitas saat ini ditutup.



Pada code diatas Blok kode `onErrorResponse` menangani kesalahan saat memperbarui data. Jika terjadi kesalahan, pesan kesalahan ditampilkan menggunakan Toast, ProgressDialog dihentikan, dan tampilan diupdate melalui adapter antarmuka pengguna untuk mencerminkan hasil operasi. Keseluruhannya, respons ini memberikan umpan balik yang jelas kepada pengguna ketika terjadi kesalahan, meningkatkan pengalaman pengguna.



Pada code di atas membuat objek adapter baru dari kelas `konfigurasi` dengan menggunakan `ArrayList` `arraylistProduct` sebagai data, dan mengaitkannya dengan aktivitas saat ini (`this`). Selanjutnya, pada pengaturan tombol hapus (`btnHapus`), kode melakukan pemeriksaan apakah beberapa field input kosong, dan jika tidak, memanggil metode `deletedata` untuk menghapus data sesuai dengan indeks posisi di dalam `CartActivity.arraylistproduct`. Kode ini bertanggung jawab untuk mengelola aksi penghapusan data dengan memberikan umpan balik kepada pengguna melalui pesan Toast.



```
1 usage
private void deletedata(final String productID,
                      final String productName,
                      final double price,
                      final int stock,
                      final String image) {
    final ProgressDialog progressDialog = new ProgressDialog(context: this);
    progressDialog.setMessage("Deleting.....");
    progressDialog.show();

    StringRequest request = new StringRequest(Request.Method.POST, url_delete_makanan,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    adapter.notifyDataSetChanged();

                    AlertDialog.Builder builder = new AlertDialog.Builder(context: editItemActivity.this);
                    builder.setCancelable(true);
                    builder.setTitle("Yakin ingin menghapus?");
                    builder.setMessage("Data yang sudah dihapus tidak dapat dikembalikan");

```

```
builder.setPositiveButton("YAKIN",
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            edtproductName.setText("");
            edtPrice.setText("");
            edtStock.setText("");
            edtImage.setText("");
            editTextId.setText("");

            Toast.makeText(context, editItemActivity.this,
                text: "Data deleted Successfully...", Toast.LENGTH_SHORT).show();

            startActivity(new Intent(getApplicationContext(), MainActivity.class));

            finish();
            progressDialog.dismiss();

            // Perbarui data lokal setelah penghapusan
            if (!arraylistProduct.isEmpty() && position < arraylistProduct.size()) {
                arraylistProduct.remove(position);
                if (!CartActivity.arraylistproduct.isEmpty() &&
                    position < CartActivity.arraylistproduct.size()) {
                    CartActivity.arraylistproduct.remove(position);
                } else {
                    Log.e(tag: "DEBUG", msg: "Invalid position or empty arraylistProduct");
                }
                adapter.notifyDataSetChanged();
            } else {
                Log.e(tag: "DEBUG", msg: "Invalid position or empty arraylistProduct");
            }
        }
    });
});
```

```
});
builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(context: editItemActivity.this,
            text: "Kamu menekan cancel, +
                " perhatikan data yang akan anda hapus kembali !~!",
            Toast.LENGTH_SHORT).show();
        progressDialog.dismiss();
        adapter.notifyDataSetChanged();
    }
});
AlertDialog dialog = builder.create();
dialog.show();

} catch (JSONException jsonObject) {
    jsonObject.printStackTrace();
}
}
```

Pada code-code diatas adalah metode deletedata pada kode di atas bertanggung jawab untuk menghapus data produk dari server melalui permintaan POST. Pertama, sebuah ProgressDialog ditampilkan untuk memberi tahu pengguna bahwa proses penghapusan sedang berlangsung dengan pesan "Deleting...".

Selanjutnya, dibuat objek StringRequest untuk melakukan permintaan POST ke URL url_delete_makanan. Dalam metode onResponse, respons dari server diubah menjadi objek JSONObject, dan adapter diberitahu untuk memperbarui dirinya dengan memanggil adapter.notifyDataSetChanged().

Sebuah AlertDialog juga dibuat untuk memastikan bahwa pengguna benar-benar ingin menghapus data. Jika pengguna menekan tombol "YAKIN", data dihapus. Setelah penghapusan berhasil, data lokal diperbarui dengan menghapus item pada posisi tertentu dalam arrayListProduct. Jika CartActivity.arraylistproduct juga berisi data, item dihapus dari posisi yang sesuai. Terakhir, adapter diupdate dengan memanggil adapter.notifyDataSetChanged().

Metode ini dirancang untuk memberikan respons yang baik terhadap penghapusan data, memberikan pengguna umpan balik visual selama proses, dan memastikan bahwa data lokal diupdate dengan benar sesuai dengan hasil penghapusan.

```
        }, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(context: editItemActivity.this,
                      error.getMessage(), Toast.LENGTH_SHORT).show();
        Toast.makeText(context: editItemActivity.this,
                      text: "Fail to delete data..", Toast.LENGTH_SHORT).show();
        progressDialog.dismiss();
    }
}
```

Pada code di atas berfungsi untuk menangani respons kesalahan dari server saat melakukan permintaan penghapusan data. Jika terjadi kesalahan, dua pesan kesalahan akan ditampilkan menggunakan 'Toast'. Pertama, pesan kesalahan yang diperoleh dari objek 'VolleyError' disajikan dengan pesan "error.getMessage()". Kedua, pesan umum "Fail to delete data.." juga ditampilkan. Selanjutnya, 'progressDialog' ditutup untuk mengakhiri tampilan ProgressDialog setelah menanggapi kesalahan.

```
    btnKembali.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { kembali(); }
    });
}

1 usage
private void kembali() {
    final ProgressDialog progressDialog = new ProgressDialog(context: editItemActivity.this);
    progressDialog.show();

    Toast.makeText(context: editItemActivity.this,
        text: "kembali ke menu utama!!!.....",
        Toast.LENGTH_SHORT).show();

    startActivity(new Intent(getApplicationContext(), MainActivity.class));
    finish();
    progressDialog.dismiss();
}
```

Pada code di atas menunjukkan implementasi dari `OnClickListener` pada tombol "btnKembali". Saat tombol tersebut diklik, metode `kembali()` akan dipanggil. Pada metode `kembali()`, sebuah ProgressDialog ditampilkan, pesan "kembali ke menu utama!!!....." ditampilkan menggunakan `Toast`, kemudian aktivitas saat ini diakhiri dan aktivitas MainActivity dimulai. Akhirnya, ProgressDialog ditutup.

```
@Override
protected Map<String, String> getParams() throws AuthFailureError {
    Map<String, String> params = new HashMap<>();
    params.put("productID", productID);
    params.put("productName", productName);
    params.put("price", String.valueOf(price));
    params.put("stock", String.valueOf(stock));
    params.put("image", image);
    return params;
}
};

RequestQueue queue = Volley.newRequestQueue(context: editItemActivity.this);
queue.add(request);
```

Code diatas berfungsi untuk menunjukkan penggunaan StringRequest dari Volley untuk mengirim permintaan ke server. Dalam blok kode ini, metode `getParams()` digunakan untuk menyusun parameter yang akan dikirim ke server dalam bentuk peta (Map). Parameter yang disertakan melibatkan informasi produk seperti productID, productName, price, stock, dan image. Setelah itu, objek StringRequest dibuat dengan menggunakan URL dan

metode permintaan POST, serta listener untuk menangani respons dari server. permintaan dimasukkan ke dalam antrian RequestQueue dari Volley untuk dieksekusi.

```
    ↑ usage
    private void populateData() {
        edtProductName.setText(arraylistProduct.get(position).getProductName());
        edtPrice.setText(String.valueOf(arraylistProduct.get(position).getPrice()));
        edtStock.setText(String.valueOf(arraylistProduct.get(position).getStock()));
        edtImage.setText(arraylistProduct.get(position).getImage());
        editTextId.setText(arraylistProduct.get(position).getProductID());
    }
}
```

Fungsi code diatas adalah untuk mendeklarasi `populateData()`. `populateData()` pada kode di atas bertujuan untuk mengisi elemen-elemen antarmuka pengguna (UI) dengan data dari objek `product`. Pada dasarnya, fungsi ini mengambil nilai-nilai seperti nama produk, harga, stok, URL gambar, dan ID produk dari objek `product` pada posisi tertentu dalam `arraylistProduct`, lalu menetapkannya ke elemen-elemen UI seperti `EditText`. Dengan cara ini, data produk dapat ditampilkan pada layar sehingga pengguna dapat melihat dan mengedit informasi produk secara langsung.

3.2.2.12 Getter & Setter User(user.php)

```
package com.example.hanying;

public class user {
    3 usages
    private String username, custname, custemail, custphone, role, password;
}

public user()
```

Untuk langkah pertamanya adalah kita perlu untuk mendeklarasi kolom - kolom yang akan digunakan berdasarkan dengan tipe dan juga jenisnya. Berdasarkan gambar diatas dimana kolom ‘username, custname, custemail, custphone, role, dan password’ semua bertipe String,

3.2.2.13 Konfigurasi user (konfigurasiUser.java)

```
no usages
public konfigurasiUser(@NonNull Context context, List<user> arraylistuser) {
    super(context, R.layout.list_user, arraylistuser);

    this.context = context;
    this.arraylistuser = arraylistuser;
}

no usages
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_user, null, true);

    TextView tmptTulisanID = view.findViewById(R.id.text);
    TextView tmptTulisanuser = view.findViewById(R.id.edtcustname);

    tmptTulisanID.setText(arraylistuser.get(position).getUsername());
    tmptTulisanuser.setText(arraylistuser.get(position).getCustname());

    return view;
}
```

Kode diatas merupakan implementasi kelas adapter dalam pengembangan aplikasi Android, yang diberi nama konfigurasiUser. Melalui konstruktor, kelas ini menerima konteks aplikasi dan daftar data pengguna. Metode getView digunakan untuk mengambil dan mengatur data pada posisi tertentu dalam daftar ke tampilan yang sesuai. Dalam proses ini, tampilan diinisialisasi dengan layout yang telah ditentukan, elemen-elemen tampilan dicari berdasarkan ID, dan data pengguna diatur ke dalam elemen-elemen tersebut. Dengan demikian, kelas adapter ini memungkinkan integrasi data pengguna ke dalam tampilan daftar, memudahkan tugas manajemen tampilan, dan meningkatkan efisiensi interaksi dengan data dalam konteks antarmuka pengguna aplikasi Android.

3.2.2.14 Fungsi show list user (get_user.php)

```
if (
    isset($_POST['username']) &&
    isset($_POST['custname']) &&
    isset($_POST['custemail']) &&
    isset($_POST['custphone']) &&
    isset($_POST['role']) &&
    isset($_POST['password'])
```

Pada code diatas berfungsi untuk memeriksa apakah semua data yang diperlukan (username, custname, custemail, custphone, role, password) disertakan dalam permintaan POST.

```
$username = $_POST['username'];
$custname = $_POST['custname'];
$custemail = $_POST['custemail'];
$custphone = $_POST['custphone'];
$role = $_POST['role'];
$password = $_POST['password'];

// Use mysqli instead of mysql, and prepared statements to prevent SQL injection
$stmt = mysqli_prepare($conn, "UPDATE user SET custname=?, custemail=?, custphone=?, role=?, password=? WHERE username=?");
mysqli_stmt_bind_param($stmt, "ssssss", $custname, $custemail, $custphone, $role, $password, $username);
```

Pada code diatas menggunakan prepared statement untuk mencegah serangan SQL injection dan melakukan pembaruan data pengguna.

```
$response = array();

if (mysqli_stmt_execute($stmt)) {
    $response["Success"] = 1;
    $response["Message"] = "Data berhasil diupdate";
    echo json_encode($response);
} else {
    $response["Success"] = 0;
    $response["Message"] = "Data gagal diupdate";
    echo json_encode($response);
}
```

Pada code diatas berfungsi untuk memberikan respons dalam format JSON, menyertakan status keberhasilan dan pesan yang sesuai setelah mencoba memperbarui data pengguna.

```
// Close the statement
mysqli_stmt_close($stmt);
} else {
    $response["Success"] = -1;
    $response["Message"] = "Data Kosong";
    echo json_encode($response);
}

// Close the connection if needed
mysqli_close($conn);
?>
```

Pada code diatas merupakan code yang berfungsi untuk memberikan sebuah informasi Jika data yang diperlukan tidak disertakan, memberikan respons bahwa data kosong. Kemudian pada bagian skrips terakhir berfungsi untuk menutup prepared statement dan koneksi database untuk efisiensi.

3.2.2.15 List User (userActivity.java)

```
21 usages
public class userActivity extends AppCompatActivity {

    4 usages
    ProgressBar progressBar;
    5 usages
    konfigurasiUser adapter;

    19 usages
    public static ArrayList<user> arraylistuser = new ArrayList<>();
    2 usages
    user user;
    6 usages
    ListView lv;
    3 usages
    ArrayList<HashMap<String, String>> list_user;
```

Code diatas merupakan variabel-variabel yang digunakan untuk mengelola UI antar pengguna, data pengguna, dan URL untuk mengambil data pengguna dari server PHP.

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

    list_user = new ArrayList<>();

    progressBar = findViewById(R.id.progressBar2);
    progressBar.setVisibility(View.VISIBLE);

    lv = findViewById(R.id.listView);
    adapter = new konfigurasiUser( context: this, arraylistuser);

    lv.setAdapter(adapter);
    mendapatkanData2();
}
```

Code diatas merupakan code yang ada pada onCreate(). Melibatkan inisialisasi elemen UI, pengaturan adapter, dan memanggil metode mendapatkanData2 untuk mendapatkan data pengguna.

```
1 usage
private void mendapatkanData2() {
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();

    if(networkInfo == null || !networkInfo.isConnected()) {
        Toast.makeText( context, userActivity.this, text: "No internet Connection!", Toast.LENGTH_SHORT).show();
        return;
    }
}
```

Pada code diatas berfungsi untuk memeriksa ketersediaan koneksi internet sebelum mencoba mengambil data. Jika tidak ada koneksi internet, pesan "No internet Connection!" ditampilkan dan metode dihentikan.

```
new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        arraylistUser.clear();
        try {
            JSONObject jObj = new JSONObject(response);
            JSONArray member = jObj.getJSONArray(TAG_USER);

            tv.setAdapter(adapter);
        } catch (Exception ex) {
            Log.e( tag: "Error", ex.toString());
            progressBar.setVisibility(View.GONE);
        }
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
```

Code diatas berfungsi untuk membuat permintaan StringRequest menggunakan Volley untuk mengambil data dari server dengan metode POST. Respons dari server diolah dalam blok onResponse.

```

        for (int i = 0; i < member.length(); i++) {
            JSONObject a = member.getJSONObject(i);

            String username = a.getString(TAG_USERNAME);
            String custname = a.getString(TAG_CUSTNAME);
            String custemail = a.getString(TAG_CUSTEMAIL);
            String custphone = a.getString(TAG_CUSTPHONE);
            String role = a.getString(TAG_ROLE);
            String password = a.getString(TAG_PASSWORD);

            HashMap<String, String> map = new HashMap<>();
            map.put("username", username);
            map.put("custname", custname);
            map.put("custemail", custemail);
            map.put("custphone", custphone);
            map.put("role", role);
            map.put("password", password);
            list_user.add(map);

            user = new user(username, custname, custemail, custphone, role, password);
            arraylistuser.add(user);
            adapter.notifyDataSetChanged();
        }
    }
}

```

Code diatas berfungsi untuk memproses respons JSON dari server. Data pengguna diambil dari objek JSON dan ditambahkan ke list dan arraylistuser. Setelah itu, adapter diberitahu tentang perubahan data dengan memanggil adapter.notifyDataSetChanged().

```

    }
    progressBar.setVisibility(View.GONE);
   ListAdapter adapter = new SimpleAdapter( context: userActivity.this, list_user,
        R.layout.list_user, new String[]{ TAG_USERNAME, TAG_CUSTNAME, TAG_CUSTEMAIL, TAG_CUSTPHONE, TAG_ROLE, TAG_PASSWORD },
        new int[]{R.id.text,
            R.id.edtcustname,
            R.id.edtcustemail,
            R.id.edtcustphone,
            R.id.edtrole,
            R.id.edtpassword});
    lv.setAdapter(adapter);
}

```

Pada code diatas Setelah memproses data, ProgressBar diatur menjadi tidak terlihat, dan data pengguna ditampilkan dalam ListView menggunakan adapter SimpleAdapter. Adapter ini mengaitkan data dengan tata letak elemen

antarmuka pengguna yang telah didefinisikan dalam R.layout.list_user.

```
@Override
public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
    AlertDialog.Builder builder = new AlertDialog.Builder(view.getContext());
    ProgressDialog progressDialog = new ProgressDialog(view.getContext());

    adapter.notifyDataSetChanged();

    CharSequence[] dialoguser = {"edit data!"};
    builder.setTitle(arraylistuser.get(position).getCustname());
    builder.setItems(dialoguser, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            startActivity(new Intent(getApplicationContext(), editUserActivity.class));
            putExtra( name: "position", position);
            adapter.notifyDataSetChanged();
        }
    });
    builder.create().show();
    return false;
}
RequestQueue queue = Volley.newRequestQueue( context: this);
queue.add(stringRequest);
```

Code diatas lv.setOnItemLongClickListener digunakan untuk menanggapi kejadian long-click pada item ListView. Ketika pengguna melakukan long-click pada item, sebuah AlertDialog.Builder dibuat untuk menampilkan opsi pengguna, dalam hal ini, hanya satu opsi yaitu "edit data". Jika pengguna memilih opsi ini, aktivitas editUserActivity dimulai dengan menyertakan posisi item yang dipilih.

Selanjutnya, adapter.notifyDataSetChanged() digunakan untuk memberi tahu adapter bahwa data telah berubah, sehingga tampilan dapat diperbarui sesuai. Akhirnya, objek RequestQueue dibuat dari Volley dan permintaan string ditambahkan ke antrian. Ini memulai permintaan data ke server setelah konfigurasi pengaturan item-click selesai.

3.2.2.16 Fungsi Add User (add_to_user.php)

```
if (
    isset($_POST['username']) &&
    isset($_POST['custname']) &&
    isset($_POST['custemail']) &&
    isset($_POST['custphone']) &&
    isset($_POST['role']) &&
    isset($_POST['password']))
```

Sama seperti pada fungsi get yang lain, kode php diatas berfungsi untuk memeriksa apakah semua data yang diperlukan (username, custname, custemail, custphone, role, password) disertakan dalam permintaan POST.

```
$username = $_POST['username'];
$custname = $_POST['custname'];
$custemail = $_POST['custemail'];
$custphone = $_POST['custphone'];
$role = $_POST['role'];
$password = ($_POST['password']);
//$password = password_hash($_POST['password'], PASSWORD_DEFAULT);

// Periksa apakah username sudah ada di database
$checkUsernameQuery = "SELECT * FROM user WHERE username = ?";
$checkUsernameStmt = $conn->prepare($checkUsernameQuery);
$checkUsernameStmt->bind_param("s", $username);
$checkUsernameStmt->execute();
$checkUsernameResult = $checkUsernameStmt->get_result();
```

Kode diatas digunakan untuk memeriksa apakah username yang diinputkan oleh pengguna sudah ada di database atau belum. Jika username sudah ada di database, maka kode program akan mengembalikan hasil berupa data dari tabel user yang memiliki username yang sama dengan username yang diinputkan. Sebaliknya, jika username belum ada di database, maka kode program tidak akan mengembalikan hasil apapun

```

if ($checkUsernameResult->num_rows > 0) {
    // Jika username sudah ada, kirimkan respons
    $response["success"] = 0;
    $response["message"] = "Failed to add data, username has been used.";
    echo json_encode($response);
} else {
    // Jika username belum ada, lakukan operasi insert
    $stmt = $conn->prepare("INSERT INTO user(username, custname, custemail, custphone, role, password) VALUES (?, ?, ?, ?, ?, ?)");
    $stmt->bind_param("ssssss", $username, $custname, $custemail, $custphone, $role, $password);

    $response = array();

    if ($stmt->execute()) {
        $response["success"] = 1;
        $response["message"] = "User added.";
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "Failed to add user.";
        echo json_encode($response);
    }
    // Tutup statement
    $stmt->close();
}
} else {
    $response["success"] = -1;
    $response["message"] = "User data is empty.";
    echo json_encode($response);
}

```

Selanjutnya kode tersebut digunakan untuk memeriksa apakah username yang diinputkan oleh pengguna sudah ada di database atau belum. Jika username sudah ada di database, maka kode program akan mengembalikan hasil berupa data dari tabel user yang memiliki username yang sama dengan username yang diinputkan. Sebaliknya, jika username belum ada di database, maka kode program akan melakukan operasi insert untuk memasukkan data pengguna ke dalam tabel user.

3.2.2.17 Add User (addUserActivity.java)

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_add_user);  
  
    edtusername = findViewById(R.id.edtusername);  
    edtcustname = findViewById(R.id.edtcustname);  
    edtcustemail = findViewById(R.id.edtcustemail);  
    edtcustphone = findViewById(R.id.edtcustphone);  
    edtrole = findViewById(R.id.edtrole);  
    edtpassword = findViewById(R.id.edtpassword);  
    progressBar = findViewById(R.id.progressBar);  
    btnAdd = findViewById(R.id.btnAdd);  
  
    btnAdd.setOnClickListener(v -> {  
        username = edtusername.getText().toString();  
        custname = edtcustname.getText().toString();  
        custemail = edtcustemail.getText().toString();  
        custphone = edtcustphone.getText().toString();  
        role = edtrole.getText().toString();  
        password = edtpassword.getText().toString();  
        progressBar.setVisibility(View.VISIBLE);  
        RequestQueue queue = Volley.newRequestQueue(context: addUserActivity.this);  
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
```

Kode di atas bertujuan untuk menangani penambahan data pengguna baru ke server melalui protokol HTTP POST. Saat tombol "btnAdd" ditekan, nilai dari elemen antarmuka pengguna seperti username, nama pelanggan, email pelanggan, dan sebagainya diambil dan disimpan dalam variabel terkait. Selanjutnya, ditampilkan indikator progres untuk menandakan bahwa proses penambahan sedang berlangsung. Menggunakan library Volley, dilakukan permintaan POST ke URL tertentu, dan respons dari server akan ditangani dalam blok yang menangani respons.

```
url_tambah_user, response -> {
    try {
        if (response.startsWith("{")) {
            JSONObject jObj = new JSONObject(response);
            int sukses = jObj.getInt( name: "success");
            if (sukses == 1) {
                Toast.makeText( context: AddUserActivity.this, text: "Data User" +
                    "Berhasil disimpan", Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext(), AdminActivity.class));
                finish();
            } else {
                Toast.makeText( context: AddUserActivity.this, text: "Gagal menyimpan data. Username sudah digunakan.", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText( context: AddUserActivity.this, text: "Non-JSON Response: " + response, Toast.LENGTH_SHORT).show();
        }
        progressBar.setVisibility(View.GONE);
    } catch (Exception ex) {
        Log.e( tag: "Error", ex.toString());
        ex.printStackTrace();
        Toast.makeText( context: AddUserActivity.this, text: "Error converting response to JSON", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.GONE);
    }
}
```

Lanjutan kode add user tersebut berfungsi untuk melakukan penanganan respons dari server setelah permintaan penambahan pengguna baru. Saat respons diterima, dilakukan pemeriksaan apakah respons tersebut berformat JSON atau tidak. Jika respons dimulai dengan karakter "{", itu dianggap sebagai respons JSON. Selanjutnya, dilakukan penguraian JSON untuk mendapatkan nilai "success" yang menunjukkan keberhasilan atau kegagalan operasi penambahan data pengguna. Jika nilai "success" sama dengan 1, itu berarti penambahan data berhasil, dan pengguna diberi informasi dengan pesan Toast yang menyatakan berhasil. Selanjutnya, aktivitas diarahkan ke halaman "AdminActivity" dan aktivitas saat ini diakhiri. Jika nilai "success" tidak sama dengan 1, pesan Toast memberi tahu pengguna bahwa gagal menyimpan data karena username sudah digunakan. Jika respons tidak berformat JSON, pesan Toast memberitahu pengguna bahwa respons tidak berformat JSON.

```

    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Error: ", error.getMessage());
        Toast.makeText( context: AddUserActivity.this, text: "Silahkan cek koneksi " +
            "internet Anda!", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.GONE);
    }
}) {
    @Override
    protected Map<String, String> getParams() {
        Map<String, String> params = new HashMap<>();
        params.put("username", username);
        params.put("custname", custname);
        params.put("custemail", custemail);
        params.put("custphone", custphone);
        params.put("role", role);
        params.put("password", password);
        return params;
    }

    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("Content-type", "application/x-www-form-urlencoded");
        return params;
    }
};

queue.getCache().clear();
queue.add(stringRequest);

```

Bagian kode terakhir diatas berfungsi saat pengguna menekan tombol "btnAdd" setelah mengisi informasi pengguna baru, data tersebut dikumpulkan dari elemen antarmuka pengguna seperti username, nama pelanggan, email, nomor telepon, peran, dan kata sandi. Selanjutnya, permintaan POST dibuat dengan memasukkan data ke parameter, dan header yang sesuai ditetapkan. Jika permintaan berhasil dikirim, respons dari server dianalisis untuk menentukan keberhasilan operasi. Jika sukses, pesan toast memberitahu pengguna bahwa data pengguna baru telah berhasil disimpan, dan aplikasi mengarahkan pengguna kembali ke aktivitas Admin. Jika ada kesalahan, pesan toast menyatakan bahwa pengguna dengan username yang sama sudah ada

3.2.2.18 Fungsi Edit User (edit_user.php)

```
<?php  
header('Content-type: application/json; charset=utf-8');  
include "conn.php";
```

Code diatas berfungsi untuk mengatur header HTTP untuk menunjukkan bahwa konten yang dikirim berformat JSON dan melakukan koneksi database kedalam conn.php

```
if (  
    isset($_POST['username']) &&  
    isset($_POST['custname']) &&  
    isset($_POST['custemail']) &&  
    isset($_POST['custphone']) &&  
    isset($_POST['role']) &&  
    isset($_POST['password'])  
) {
```

Pada code diatas berfungsi untuk memeriksa apakah parameter POST yang diperlukan (username, custname, custemail, custphone, role, dan password) sudah diatur atau belum.

```
$username = $_POST['username'];  
$custname = $_POST['custname'];  
$custemail = $_POST['custemail'];  
$custphone = $_POST['custphone'];  
$role = $_POST['role'];  
$password = $_POST['password'];
```

Pada code diatas memberikan nilai parameter POST ke variabel yang sesuai.

```
// Use mysqli instead of mysql, and prepared statements to prevent SQL injection
$stmt = mysqli_prepare($conn, "UPDATE user SET custname=?, custemail=?, custphone=?, role=?, password=? WHERE username=?");
mysqli_stmt_bind_param($stmt, "ssssss", $custname, $custemail, $custphone, $role, $password, $username);
```

Pada code diatas Ini menyiapkan pernyataan SQL untuk memperbarui seorang pengguna di database. Penggunaan mysqli_prepare dan mysqli_stmt_bind_param membantu mencegah SQL injection dengan menggunakan pertanyaan yang telah diparameterkan.

```
if (mysqli_stmt_execute($stmt)) {
```

Code diatas berfungsi untuk menjalankan pernyataan yang telah disiapkan. Jika berhasil, akan mengembalikan respons sukses; jika tidak, akan mengembalikan respons gagal.

```
if (mysqli_stmt_execute($stmt)) {
    $response[ "Success" ] = 1;
    $response[ "Message" ] = "Data berhasil diupdate";
    echo json_encode($response);
} else {
    $response[ "Success" ] = 0;
    $response[ "Message" ] = "Data gagal diupdate";
    echo json_encode($response);
}
```

Code diatas berfungsi untuk membuat sebuah array (\$response) untuk menyusun respons. Jika pembaruan berhasil, "Success" diatur menjadi 1; sebaliknya, diatur menjadi 0. Kunci "Message" berisi pesan yang sesuai. Respons ini kemudian dienkripsi dalam format JSON dan dicetak.

```
    } else {
        $response[ "Success" ] = -1;
        $response[ "Message" ] = "Data Kosong";
        echo json_encode($response);
    }
```

Jika parameter POST yang diperlukan tidak diatur, "Success" diatur menjadi -1 dan menyediakan pesan yang mengindikasikan bahwa data kosong.

```
// Close the statement
mysqli_stmt_close($stmt);
```

Pada code diatas berfungsi untuk menutup pernyataan.

```
// Close the connection if needed
mysqli_close($conn);
?>
```

Code diatas adalah untuk menutup database.

3.2.2.19 Fungsi Delete User (delete_user.php)

```
delete_user.php
1 <?php
2 header('Content-type: application/json; charset=utf-8');
3 include "conn.php";
4
5 if (isset($_POST['username'])) {
6     $username = $_POST['username'];
7
8     // Use mysqli instead of mysql
9     $stmt = mysqli_prepare($conn, "DELETE FROM user WHERE username=?");
10    mysqli_stmt_bind_param($stmt, "s", $username);
11
12    $response = array();
13
14    if (mysqli_stmt_execute($stmt)) {
15        $response["Success"] = 1;
16        $response["Message"] = "Data berhasil dihapus";
17        echo json_encode($response);
18    } else {
19        $response["Success"] = 0;
20        $response["Message"] = "Data gagal dihapus";
21        echo json_encode($response);
22    }
23
24    // Close the statement
25    mysqli_stmt_close($stmt);
26 } else {
27     $response["Success"] = -1;
28     $response["Message"] = "Data Kosong";
29     echo json_encode($response);
30 }
31
32 // Close the connection if needed
33 mysqli_close($conn);
34 ?>
```

Kode PHP di atas adalah skrip untuk menghapus data pengguna (user) dari tabel database menggunakan metode HTTP POST. Pertama, didefinisikan tipe konten sebagai JSON dengan mengatur header. Selanjutnya, koneksi ke database diinisialisasi dengan memasukkan file koneksi ("conn.php"). Jika parameter "username" diterima melalui metode POST, persiapan pernyataan SQL menggunakan mysqli dilakukan untuk menghapus entri dengan nama pengguna yang sesuai. Setelah eksekusi pernyataan, respons JSON dibangun berdasarkan keberhasilan atau kegagalan penghapusan, dan hasilnya dikirim kembali sebagai respons. Jika tidak ada parameter yang diterima, respons mengindikasikan data kosong.

3.2.2.20 Edit User (editUserActivity.java)

```
24 usages
public class editUserActivity extends AppCompatActivity {

    6 usages
    konfigurasiUser adapter;
    18 usages
    public static ArrayList<user> arraylistuser = new ArrayList<>();
    7 usages
    EditText edtusername, edtcustname, edtcustemail, edtcustphone, edtrole, edtpassword;
    2 usages
    Button btnUpdate, btnHapus, btnKembali;
    1 usage
    String url_update_user = "http://192.168.30.202/hanying/edit_user.php";
    1 usage
    String url_delete_user = "http://192.168.30.202/hanying/delete_user.php";
    30 usages
    private int position;
    no usages
    private AlertDialog.Builder alertDialogBuilder;
    no usages
    String username, custname, custemail, custphone, role, password;
    no usages
    user user;
```

Kode tersebut adalah implementasi kelas `editUserActivity` pada aplikasi Android. Adaptor `konfigurasiUser` dan ArrayList `arraylistuser` digunakan untuk manajemen data pengguna. UI seperti EditText dan Button digunakan untuk interaksi pengguna. Terdapat URL endpoint untuk operasi pembaruan dan penghapusan data pengguna. Variabel `position` melacak posisi data yang diubah atau dihapus, dan objek `AlertDialog.Builder` digunakan untuk dialog peringatan. Variabel String diinisialisasi untuk menyimpan nilai input. Keseluruhan, kelas ini bertugas melakukan manajemen data pengguna dan berkomunikasi dengan server sesuai dengan UI pengguna yang telah ditetapkan.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = getIntent();
    if (intent != null && intent.getExtras() != null) {
        position = intent.getExtras().getInt("position", defaultValue: -1);
    } else {
        finish();
        return;
    }
    //position = intent.getExtras().getInt("position");
    setContentView(R.layout.activity_edit_user);
    btnUpdate = findViewById(R.id.btnedit);
    btnHapus = findViewById(R.id.btnhapus);
    btnKembali = findViewById(R.id.btnbatal);
    edtusername = findViewById(R.id.edtusername);
    edtcustname = findViewById(R.id.edtcustname);
    edtcustemail = findViewById(R.id.edtcustemail);
    edtcustphone = findViewById(R.id.edtcustphone);
    edtrole = findViewById(R.id.edtrole);
    edtpassword = findViewById(R.id.edtpassword);

    edtusername.setText(userActivity.arraylistuser.get(position).getUsername());
    edtcustname.setText(userActivity.arraylistuser.get(position).getCustname());
    edtcustemail.setText(userActivity.arraylistuser.get(position).getCustemail());
    edtcustphone.setText(userActivity.arraylistuser.get(position).getCustphone());
    edtrole.setText(userActivity.arraylistuser.get(position).getRole());
    edtpassword.setText(userActivity.arraylistuser.get(position).getPassword());
```

Kode di atas merupakan bagian dari metode `onCreate` dalam kelas `editUserActivity` pada aplikasi Android. Pada saat aktivitas ini dibuat, ia menerima data dari intent yang memulainya, khususnya nilai posisi data pengguna yang akan diubah. Jika data yang diharapkan tidak ditemukan, aktivitas ditutup. Setelah itu, UI pengguna dari layout XML (`activity_edit_user`) diinisialisasi menggunakan metode `setContentView`. Komponen-komponen UI pengguna seperti tombol dan editText dihubungkan dengan variabel dalam kelas, dan nilai-nilai dari data pengguna yang akan diubah diisi ke dalam editText sesuai dengan posisi yang diterima dari aktivitas sebelumnya.

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (edtusername.getText().toString().isEmpty()) {
            // Your custom logic for handling empty username
            Toast.makeText(context: editUserActivity.this,
                           text: "Please input your username...", Toast.LENGTH_SHORT).show();
        } else if (edtcustname.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this,
                           text: "please enter your customer name...", Toast.LENGTH_SHORT).show();
        } else if (edtcustemail.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this,
                           text: "Please input your customer email...", Toast.LENGTH_SHORT).show();
        } else if (edtcustphone.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this,
                           text: "Please input your customer number phone...", Toast.LENGTH_SHORT).show();
        } else if (edtrole.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this,
                           text: "Please input your the role...", Toast.LENGTH_SHORT).show();
        } else if (edtpassword.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this,
                           text: "Please input your password...", Toast.LENGTH_SHORT).show();
        } else {
            callPUTDataMethod(edtusername.getText().toString(),
                               edtcustname.getText().toString(),
                               edtcustemail.getText().toString(),
                               edtcustphone.getText().toString(),
                               edtrole.getText().toString(),
                               edtpassword.getText().toString());
        }
    }
})
```

Kode di atas adalah bagian dari penanganan klik tombol (`btnUpdate`) pada kelas `editUserActivity`. Ketika tombol tersebut diklik, program akan memeriksa apakah beberapa editText yang berkaitan dengan data pengguna kosong atau tidak. Jika ada yang kosong, pesan kesalahan akan ditampilkan menggunakan `Toast`. Jika semua editText terisi, metode `callPUTDataMethod` akan dipanggil untuk melakukan pembaruan data pengguna dengan nilai-nilai yang diambil dari editText.

```

1 usage
private void callPUTDataMethod(String username,
                               String custname,
                               String custemail,
                               String custphone,
                               String role,
                               String password) {
    final ProgressDialog progressDialog = new ProgressDialog(context: this);

    StringRequest request = new StringRequest(Request.Method.POST, url_update_user,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Log.d(tag: "DEBUG", msg: "response" + response);
                adapter.notifyDataSetChanged();
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    edtusername.setText("");
                    edtcustname.setText("");
                    edtcustemail.setText("");
                    edtcustphone.setText("");
                    edtrole.setText("");
                    edtpassword.setText("");

                    Toast.makeText(context: editUserActivity.this, text: "Data updated..", Toast.LENGTH_SHORT).show();
                }
            }
        }
    );
}

```

Metode `callPUTDataMethod` pada kelas `editUserActivity` menggunakan `Volley` untuk mengirim permintaan POST ke server dengan URL yang telah ditentukan (`url_update_user`). Setelah menerima respons dari server, data pada UI admin dikosongkan, dan pesan singkat ditampilkan untuk memberi informasi kepada pengguna bahwa data telah berhasil diperbarui. Code diatas merupakan pembaruan pada adapter untuk memastikan tampilan yang terkait mendapatkan pembaruan data yang sesuai.

```

        startActivity(new Intent(getApplicationContext(), MainActivity.class));

        if (!arraylistuser.isEmpty() && position < arraylistuser.size()) {
            arraylistuser.get(position).setUsername(username);
            arraylistuser.get(position).setCustname(custname);
            arraylistuser.get(position).setCustemail(custemail);
            arraylistuser.get(position).setCustphone(custphone);
            arraylistuser.get(position).setRole(role);
            arraylistuser.get(position).setPassword(password);

            populateData();
        } else {
            Log.e(tag: "DEBUG", msg: "Invalid position or empty arraylistuser");
        }

        finish();
    } catch (JSONException jsonObject) {
        jsonObject.printStackTrace();
    }
}

```

Kode diatas merupakan lanjutan dari fungsi `callPUTDataMethod`. Setelah memperbarui data melalui metode `callPUTDataMethod`, admin diarahkan kembali ke `MainActivity`. Langkah ini diikuti dengan pengecekan

apakah array `arraylistuser` tidak kosong dan apakah posisi yang diinginkan berada dalam rentang ukuran array. Jika ya, data pada posisi tersebut dalam array diperbarui dengan nilai yang baru. Selanjutnya, metode `populateData` dipanggil untuk memastikan UI admin menampilkan data yang diperbarui. Jika kondisi tersebut tidak terpenuhi, pesan log kesalahan ("Invalid position or empty arraylistuser") dan proses ditutup.

```
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(context, editUserActivity.this, text: "Fail to update data..", Toast.LENGTH_SHORT).show();
            progressDialog.dismiss();
            adapter.notifyDataSetChanged();
        }
    });
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("username", username);
        params.put("custname", custname);
        params.put("custemail", custemail);
        params.put("custphone", custphone);
        params.put("role", role);
        params.put("password", password);

        return params;
    }
}
RequestQueue queue = Volley.newRequestQueue(context: editUserActivity.this);
queue.add(request);
```

Dalam bagian kode ini, ketika terjadi kesalahan dalam memperbarui data, respons kesalahan dari server Volley ditangani. Pesan kesalahan sesuai konteks ditampilkan menggunakan `Toast`. Selanjutnya, dialog progres (`progressDialog`) ditutup, dan adapter diinformasikan tentang perubahan data dengan memanggil `adapter.notifyDataSetChanged()`. Pada bagian ini, permintaan (`request`) untuk memperbarui data dikonfigurasi sebagai metode POST menggunakan Volley. Parameter seperti username, custname, custemail, custphone, role, dan password dikirimkan ke server sebagai bagian dari permintaan.

```

btnHapus.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (edtcustname.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this, text: "please enter your data", Toast.LENGTH_SHORT).show();
        } else if (edtcustemail.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this, text: "please enter your data", Toast.LENGTH_SHORT).show();
        } else if (edtcustphone.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this, text: "please enter your data", Toast.LENGTH_SHORT).show();
        } else if (edtrole.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this, text: "please enter your data", Toast.LENGTH_SHORT).show();
        } else if (edtpassword.getText().toString().isEmpty()) {
            Toast.makeText(context: editUserActivity.this, text: "please enter your data", Toast.LENGTH_SHORT).show();
        } else
            deletedata2(userActivity.arraylistuser.get(position).getUsername(),
                        userActivity.arraylistuser.get(position).getCustname(),
                        userActivity.arraylistuser.get(position).getCustomail(),
                        userActivity.arraylistuser.get(position).getCustphone(),
                        userActivity.arraylistuser.get(position).getRole(),
                        userActivity.arraylistuser.get(position).getPassword());
    }
});

```

Dalam Kode diatas, sebuah listener ('OnClickListener') ditetapkan pada tombol "Hapus" ('btnHapus'). Saat tombol ditekan, kode akan memeriksa apakah beberapa field (seperti `edtcustname`, `edtcustemail`, `edtcustphone`, `edtrole`, dan `edtpassword`) tidak kosong. Jika salah satu dari field tersebut kosong, maka akan ditampilkan pesan Toast untuk meminta pengguna mengisi data yang diperlukan. Jika semua field terisi, fungsi `deletedata2` dipanggil dengan menyediakan data pengguna yang dipilih, seperti username, custname, customail, custphone, role, dan password, dari arraylist yang sesuai pada posisi yang dipilih.

```

1 usage
private void deletedata2(final String username, final String custname, final String customail, final String custphone, final
    String custmail, final String custphone, final String role, final String password) {
    final ProgressDialog progressDialog = new ProgressDialog(context: this);
    progressDialog.setMessage("Deleting.....");
    progressDialog.show();

    StringRequest request = new StringRequest(Request.Method.POST, url_delete_user,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    adapter.notifyDataSetChanged();

                    AlertDialog.Builder builder = new AlertDialog.Builder(context: editUserActivity.this);
                    builder.setCancelable(true);
                    builder.setTitle("Yakin ingin menghapus?");
                    builder.setMessage("Data yang sudah dihapus tidak dapat dikembalikan");

                    builder.setPositiveButton(text: "YAKIN",
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                edtusername.setText("");
                                edtcustname.setText("");
                                edtcustemail.setText("");
                                edtcustphone.setText("");
                                edtrole.setText("");
                                edtpassword.setText("");

                                Toast.makeText(context: editUserActivity.this, text: "Data Deleted Successfully...", To

```

Pada code diatas menjelaskan metode pada `deletedata2`, setelah ProgressDialog ditampilkan, dibuat sebuah StringRequest dengan metode POST untuk mengirimkan permintaan penghapusan ke server menggunakan URL yang ditentukan (`url_delete_user`). Ketika respon diterima, JSON yang dikembalikan diuraikan, dan adapter diupdate untuk adanya perubahan. Selanjutnya, sebuah AlertDialog ditampilkan untuk mengonfirmasi pengguna apakah mereka yakin ingin menghapus data. Jika pengguna menekan tombol "YAKIN", semua field input diatur ke string kosong dan pesan Toast muncul memberi tahu pengguna bahwa data telah dihapus berhasil.

```
        startActivity(new Intent(getApplicationContext(), MainActivity.class));

        finish();
        progressDialog.dismiss();

        if (!arraylistuser.isEmpty() && position < arraylistuser.size()) {
            arraylistuser.remove(position);
            if (userActivity.arraylistuser.isEmpty() && position < userActivity.arraylistuser.size()) {
                userActivity.arraylistuser.remove(position);
            } else {
                Log.e("DEBUG", msg: "Invalid position or empty arraylistuser");
            }
            adapter.notifyDataSetChanged();
        } else {
            Log.e("DEBUG", msg: "Invalid position or empty arraylistuser");
        }
    });
}
```

Pada code diatas setelah berhasil menghapus data, dilakukan navigasi ke MainActivity dan menutup aktivitas editUserActivity. Selanjutnya, terdapat pengecekan dan penghapusan item pada arraylistuser, serta pembaruan adapter.

```
builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(context: editUserActivity.this,
                      text: "Kamu menekan cancel, " +
                           "perhatikan data yang akan anda hapus kembali !~!", Toast.LENGTH_SHORT).show();
        progressDialog.dismiss();
        adapter.notifyDataSetChanged();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
} catch (JSONException jsonException) {
    jsonException.printStackTrace();
}
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(context: editUserActivity.this, error.getMessage(), Toast.LENGTH_SHORT).show();
        Toast.makeText(context: editUserActivity.this, text: "Fail to delete data...", Toast.LENGTH_SHORT).show();
        progressDialog.dismiss();
    }
}
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(context: editUserActivity.this, error.getMessage(), Toast.LENGTH_SHORT).show();
        Toast.makeText(context: editUserActivity.this, text: "Fail to delete data...", Toast.LENGTH_SHORT).show();
        progressDialog.dismiss();
    }
}
};
```

Pada code diatas ketika admin click cancel maka proses penghapusan data akan gagal dan tampilan akan dikembalikan ke menu Admin. Kemudian pada code diatas juga menangani beberapa error jika proses tersebut gagal. Jika admin gagal melakukan delete data maka akan terdapat toast "Fail to delete data..." .

```
    ...
    btnKembali.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { kembali(); }
    });
}
1 usage
private void kembali() {
    final ProgressDialog progressDialog = new ProgressDialog(context: editUserActivity.this);
    progressDialog.show();

    Toast.makeText(context: editUserActivity.this, text: "Kembali ke menu utama!!!!....", Toast.LENGTH_SHORT).show();

    startActivity(new Intent(getApplicationContext(), MainActivity.class));
    finish();
    progressDialog.dismiss();
}
```

Ketika tombol "Kembali" ditekan, aplikasi menjalankan metode `kembali()`. Pada metode ini, ProgressDialog ditampilkan, pesan toast memberitahu pengguna bahwa mereka akan kembali ke menu utama, dan aktivitas saat ini (editUserActivity) diarahkan ke MainActivity. Setelah itu, ProgressDialog ditutup, dan aktivitas saat ini ditutup menggunakan metode `finish()`.

BAB IV

KESIMPULAN

3.1 Kesimpulan

Berdasarkan pencapaian pada berbagai aspek pengembangan aplikasi Hanying, dapat disimpulkan bahwa kami berhasil mencapai tujuan perancangan aplikasi ini. Beberapa poin kunci yang berhasil diimplementasikan adalah:

1. Database Design, WebView, dan ViewPager

Aplikasi Hanying berhasil mengimplementasikan desain database yang efisien untuk menyimpan dan mengelola data. Penggunaan WebView dan ViewPager juga terintegrasi dengan baik, meningkatkan pengalaman pengguna dalam menjelajahi dan berinteraksi dengan konten aplikasi.

2. SQLite

Implementasi SQLite pada aplikasi Hanying berhasil menciptakan sistem manajemen basis data lokal yang efektif, memungkinkan penyimpanan dan pengambilan data dengan efisien di perangkat pengguna.

3. PHP Script sebagai API

Aplikasi menggunakan skrip PHP sebagai API untuk menghubungkan ke database, memperoleh, menambah, memasukkan, mengedit, dan menghapus data. Hal ini menciptakan koneksi yang solid antara aplikasi mobile dan backend, memastikan keberlanjutan operasional aplikasi.

4. Parsing JSON Data ke Aplikasi Android

Proses parsing data JSON dari API PHP ke aplikasi Android berjalan dengan sukses. Ini berkontribusi pada keberhasilan aplikasi dalam mendapatkan, menambah, memasukkan, mengedit, dan menghapus data dari basis data secara dinamis.

5. Maps API

Integrasi Maps API pada aplikasi Hanying berhasil memberikan fungsionalitas peta yang kaya. Pengguna dapat dengan mudah menavigasi dan menemukan lokasi tertentu, memperkaya fitur dan kegunaan aplikasi secara keseluruhan.

Dengan demikian, kesimpulan yang dapat diambil oleh kelompok kami adalah bahwa aplikasi Hanying berhasil menerapkan konsep dan materi mata kuliah Mobile Application Development, seperti desain basis data, WebView, ViewPager, SQLite, Parsing JSON data, dan Maps API. Implementasi ini telah menghasilkan aplikasi yang efisien, fungsional, dan sesuai dengan tujuan yang ditetapkan pada awal pengembangan

4.2 Saran

Berdasarkan proses pengembangan yang kami lakukan pada aplikasi Hanying, banyak hal yang tidak sesuai dengan tujuan awal kami dan masih ada banyak hal yang dapat kami kembangkan seperti fitur review pada aplikasi, selain itu, pelacakan real-time terhadap status pesanan dan kemungkinan memberikan rekomendasi makanan populer dapat meningkatkan pengalaman pengguna. Untuk saran pada restoran dimsum Untuk meningkatkan kualitas layanan dan daya saing, restoran dimsum dapat mengambil langkah-langkah strategis. Pertama, pengembangan aplikasi mobile serupa dengan "Hanying" dapat mempermudah pelanggan dalam pemesanan, pelacakan pesanan, dan interaksi dengan restoran. Selanjutnya, implementasi sistem manajemen pesanan yang terintegrasi dapat meningkatkan efisiensi operasional dan akurasi layanan. Program loyalitas, penawaran spesial, dan pembaruan menu berkala dapat meningkatkan daya tarik pelanggan dan meningkatkan penjualan.

PERANAN ANGGOTA

Berikut merupakan pembagian peran dalam kelompok

1. Kevin Aditya Hartono - (NIM: 00000069875) → Proposal, Interface Aplikasi
2. Ray Anthony Pranoto - (NIM: 00000066655) → Admin Page, PHP Script, Database admin, SQLite
3. Nathan Vilbert Kosasih - (NIM: 00000069903) → Proposal, Interface Aplikasi
4. Juanito Arvin William - (NIM: 00000069483) → Maps (Fragment dan API's), WebView, ViewPager
5. Julius Calvin Saputra - (NIM: 00000068626) → Proposal, Interface Aplikasi
6. Christopher Abie Diaz Doviano - (NIM: 00000067692) → User Page, JSON, Database Design, PHP Script