

Restaurant Queue Management System

Oleh Kelompok DAN :

1. Johannes Felix Rimbun 13217006
2. Ryan Dharma Chandra 13217018
3. Wilfrid Azariah 13217071

1 Latar Belakang dan Deskripsi Umum Sistem

1.1 Latar Belakang

Pandemi COVID19 menyebabkan dibuatnya kebijakan *social distancing* oleh pemerintah. Kontak fisik antara manusia dibatasi untuk menghambat perpindahan virus dari satu orang kepada yang lainnya. Dampak dari kebijakan ini sangat berpengaruh pada ekonomi Indonesia sebab berbagai kegiatan ekonomi masih mengandalkan kontak fisik antara manusia secara langsung. Salah satu sektor yang paling terpengaruh adalah sektor industri makanan. Pemesanan makanan secara langsung di restoran sangat dibatasi karena banyaknya kontak fisik yang terjadi. Kontak fisik antar manusia paling banyak terjadi pada antri/pemesanan makanan. Untuk itu, kontak fisik pada proses antrian/pemesanan harus dapat direduksi. Salah satu solusi untuk menanganinya adalah proses antrian/pemesanan dapat dipindahkan ke media virtual berbasis aplikasi.

1.2 Deskripsi Umum

Berdasarkan permasalahan yang telah dijelaskan pada paragraf di atas, perlu dikembangkan suatu sistem yang dapat mengakomodasi proses antrian pada restoran. Sistem yang akan dikembangkan ini adalah aplikasi pemesanan makanan di restoran yang berbasis ponsel pintar. Pelanggan mengunduh aplikasi terlebih dahulu untuk memesan makanan. Selanjutnya, ponsel pelanggan harus terhubung ke jaringan WiFi restoran dan melakukan registrasi akun agar profil terdaftar pada sistem restoran. Sebelum melakukan pemesanan, perlu dipastikan bahwa saldo akun pelanggan cukup untuk melakukan pembayaran. Apabila saldo tidak cukup, pelanggan harus melakukan *top-up* saldo melalui kasir. Selanjutnya, pelanggan dapat membuka aplikasi dan memilih menu makanan yang hendak dipesan. Dengan melakukan pemesanan ini, saldo pelanggan akan secara otomatis berkurang. Pemantauan terhadap status pemesanan dapat ditampilkan di ponsel pelanggan. Apabila pesanan makanan telah selesai disiapkan, aplikasi pada ponsel akan memberikan notifikasi untuk mengambil makanan di kasir. Selain itu, pemesan juga dapat melihat riwayat pesanan.

Pada aplikasi di kasir restoran, order makanan yang dipesan oleh pelanggan dapat ditampilkan pada komputer kasir. Kasir dapat melihat daftar pesanan dan memberitahu koki untuk menyiapkannya. Ketika makanan telah selesai disiapkan, kasir dapat melakukan *update* terhadap status pemesanan sehingga aplikasi pelanggan akan langsung mendapatkan

notifikasi untuk melakukan pengambilan makanan. Ketika pelanggan datang untuk mengambil makanan, pelanggan menunjukkan nomor antrian pada kasir. Kasir lalu melakukan *update* status pemesanan tersebut sehingga pesanan selesai. Kasir juga dapat mencari order-order dengan nomor spesifik atau order berdasarkan tanggal pemesanan.

Pembayaran makanan terjadi secara otomatis pada sistem. Sistem pembayaran didasarkan pada saldo virtual pelanggan. Pelanggan perlu mengisi saldo secara non tunai ke kasir. Kasir akan mengupdate saldo dari profil pelanggan sehingga selanjutnya transaksi dapat bersifat non tunai hingga saldo pelanggan habis. Karena sistem yang dibuat cukup mewakili proses pemesanan makanan yang terjadi di restoran, pengembangan selanjutnya dari sistem ini bisa dikembangkan menjadi ERP yang dimiliki seluruh restoran.

1.3 Fitur Utama

Sistem yang Restaurant Queue Management System ini memiliki dua fitur utama, yaitu sebagai berikut.

1. Aplikasi berbasis Android pada pelanggan dapat melakukan pendaftaran akun, menyimpan login pengguna, melakukan pemesanan makanan, mendapatkan notifikasi terhadap *update* status pemesanan, dan melihat *history* pemesanan.
2. Aplikasi berbasis komputer yang dimiliki oleh kasir dapat menunjukkan daftar pesanan yang masuk, melakukan perubahan terhadap status pemesanan, mencari pemesanan, dan melakukan *top up* saldo pelanggan.

2 Spesifikasi Teknis

2.1 Physical Network dan Transport

Sistem yang dirancang akan menggunakan physical network yaitu WiFi yang disediakan di restoran. Diasumsikan restaurant telah menyediakan WiFi yang dapat digunakan oleh pelanggan dan juga pegawai dari restaurant. Network addressing yang akan dipakai adalah IPv4. Transport yang akan dipakai untuk sistem TCP karena dibutuhkan koneksi yang reliable antara server dan client. Pengiriman message antara server dan client akan dilakukan secara LAN WiFi restaurant, tidak melalui internet.

2.2 Opsi Server Client dan Bahasa Pemrograman

Sistem yang dirancang terdiri atas 1 jenis server dan 2 jenis client. Server akan berperan untuk memberikan jawaban atas request dari client. Client akan berperan untuk melakukan request kepada server. Terdapat 2 jenis client, yaitu client kasir dan client pelanggan. Client pelanggan adalah aplikasi yang akan memberikan request berdasarkan fitur yang akan digunakan oleh pelanggan. Client kasir adalah aplikasi yang akan memberikan request berdasarkan fitur yang akan digunakan oleh kasir. Seperti yang telah disebutkan, server dan client dapat saling berkomunikasi menggunakan jaringan WiFi yang ada pada restaurant.

Server akan diimplementasikan pada suatu komputer yang terhubung pada jaringan WiFi restaurant. Terdapat satu buah server. Fungsi utama server adalah menyimpan data-data client. Server juga berfungsi untuk memberikan data-data yang dibutuhkan oleh client sesuai request yang diberikan. Server bersifat pasif dan hanya akan memberikan data pada client jika diminta. Server akan diimplementasikan pada suatu aplikasi dengan menggunakan bahasa pemrograman Python. Untuk keperluan penyimpanan data, akan digunakan MySQL database pada server yang tersambung dengan program socket server.

Client pelanggan akan diimplementasikan dengan suatu aplikasi mobile native android. Pertimbangan dari pembuatan aplikasi mobile adalah pelanggan biasanya membawa *smartphone* ke restoran. Platform Android dipilih karena market share *smartphone* di Indonesia masih didominasi oleh pengguna Android. Fungsi utama aplikasi Android ini adalah pelanggan dapat melakukan pemesanan makanan dan *tracking* terhadap status pemesanan. Aplikasi client pelanggan akan diimplementasikan dengan menggunakan bahasa pemrograman Java. Lingkungan pengembangan yang akan digunakan adalah Android Studio. Berikut adalah request yang akan dilakukan oleh client pelanggan.

1. Order makanan (POST)
2. Melakukan pembayaran (POST)
3. Melihat status order sendiri (GET)
4. Melihat nomor order (GET)
5. Melihat Saldo (GET)

Fungsi utama client kasir adalah menerima dan melakukan update terhadap status pemesanan. Client kasir akan diimplementasikan dengan suatu program komputer. Pertimbangan pemilihan program komputer ini adalah kasir perlu melihat daftar pesanan yang panjang. Kasir dapat lebih mudah melakukan *tracking* pesanan dengan layar lebar yang tersedia pada komputer konvensional dibandingkan dengan menggunakan *smartphone* atau tablet. Aplikasi client kasir akan diimplementasikan dengan menggunakan bahasa Python. Untuk memudahkan interaksi antara personnel kasir dan program, akan digunakan GUI untuk aplikasi client kasir tersebut. GUI akan dibuat dengan API tkinter yang tersedia pada bahasa pemrograman Python.

1. Melihat semua order makanan (GET)
2. Melakukan perubahan pada status order (POST)
3. Menghapus order (DELETE)
4. Melihat nomor order (GET)
5. Melakukan Top-Up pada saldo pelanggan (POST)

2.3 Security

Sesuai dengan pembahasan pada bagian sebelumnya, komunikasi antara server dan client dilakukan dengan socket. Untuk mengamankan komunikasi socket ini, digunakan SSL

(Secure Socket Layer) dengan pembangkitan sertifikat dan kunci server dan client dengan format openssl X509.

Untuk keamanan dari data pada database, database MySQL diberikan password sehingga hanya pihak tertentu yang bisa mengakses database tersebut. Pada database itu sendiri, khusus penyimpanan password dihash dengan metode SHA-224 sehingga admin database tidak bisa melihat password pengguna.

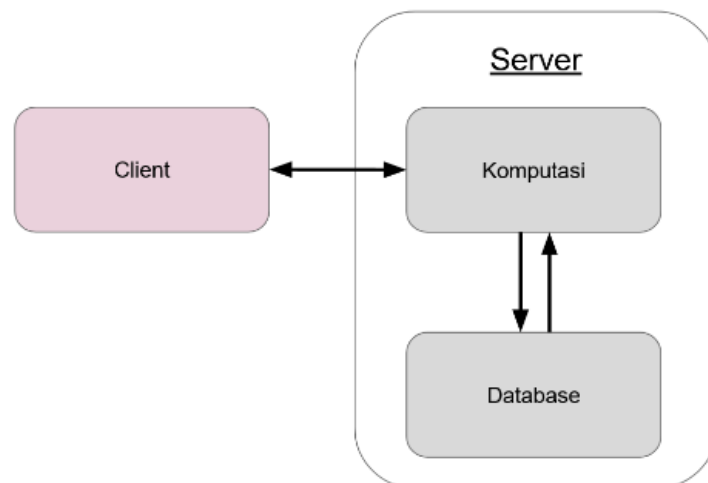
Untuk permission penggunaan dari client kasir dan client pelanggan, seluruh fitur hanya dapat digunakan setelah login. Dalam hal ini, kasir harus login terlebih dahulu sebelum melihat order makanan, mengatur order makanan, melakukan top up, dan lain-lain. Di lain pihak, pelanggan harus login terlebih dahulu sebelum melakukan pemesanan makanan, melakukan pembayaran, melihat history, dan lain-lain.

3 Perancangan Sistem

Sistem yang dirancang terdiri atas dua bagian utama, yaitu Server dan Client. Penjelasan masing-masing terhadap rancangan sistem tersebut terdapat pada subbab berikut ini.

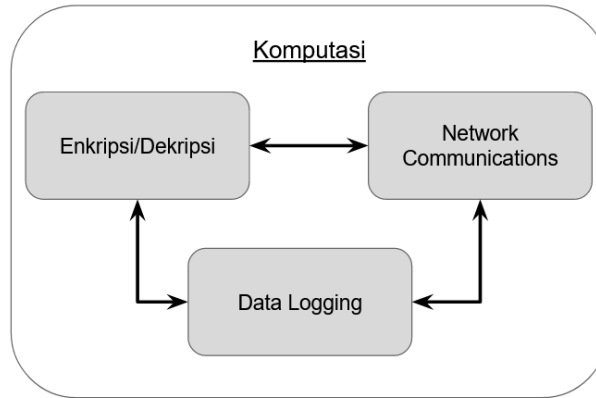
3.1 Rancangan Sistem Server

Server bekerja dengan menerima suatu permintaan informasi dari client dan mengirimkan informasi yang dibutuhkan tersebut ke client. Secara umum Sistem Server dapat dibagi menjadi dua buah subsistem, yaitu subsistem komputasi dan subsistem database. Kedua subsistem ini digambarkan melalui diagram berikut ini.



Gambar 1 Blok Diagram Sistem Server

Pada gambar di atas, subsistem database hanya memiliki fungsi penyimpanan data file dengan bantuan MySQL. Informasi ini disimpan di komputer server dan hanya dapat diakses oleh subsistem komputasi server. Sedangkan, subsistem komputasi memiliki beberapa fungsi utama, yaitu *network communication*, *data logging*, dan *enkripsi/dekripsi*. Ketiga fungsi ini dapat direpresentasikan sebagai tiga buah modul seperti pada gambar berikut ini.

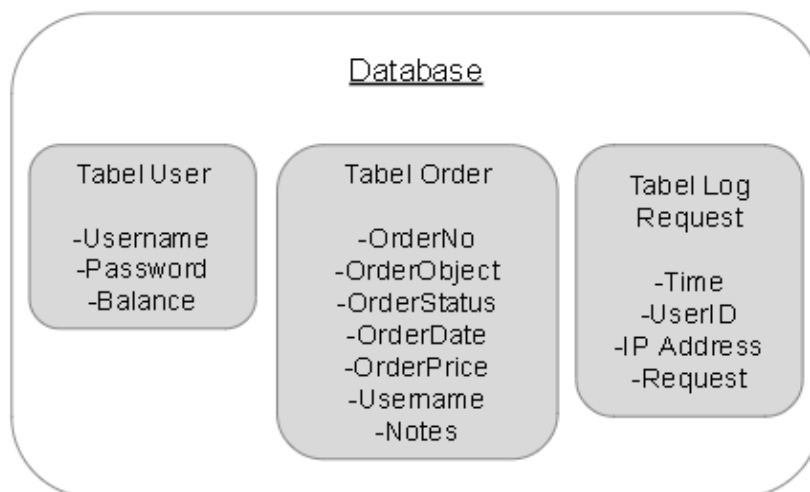


Gambar 2 Blok Diagram Sub Sistem Komputasi Server

Berdasarkan gambar di atas, ketiga modul sub sistem komputasi server adalah sebagai berikut.

1. Enkripsi/Dekripsi : Modul ini berfungsi untuk melakukan enkripsi dan dekripsi pesan yang dikirimkan dan diterima oleh server. Modul ini diimplementasikan menggunakan SSL. Dengan demikian, dapat dipastikan bahwa informasi yang dikirimkan antara server dan client aman.
2. Network Communications : Modul ini bertanggung jawab untuk melakukan komunikasi melalui socket programming, yaitu menerima dan mengirimkan data. Modul ini perlu berkomunikasi dengan modul enkripsi/dekripsi untuk aspek keamanan terhadap data yang diterima dan dikirimkan. Setelah data yang diterima berhasil didekripsi, modul ini melakukan analisis terhadap informasi yang dibutuhkan sehingga modul data logging dapat memberikan informasi yang dibutuhkan.
3. Data Logging : modul ini berfungsi untuk melakukan akses pembacaan dan penulisan informasi pada sub sistem database.

Selain itu, terdapat subsistem database dan penyimpanan file pada server. Database atau file inilah yang dapat diakses oleh modul data logging pada sub sistem komputasi Server. Adapun mekanisme konfigurasi ini dapat dijelaskan melalui diagram berikut.



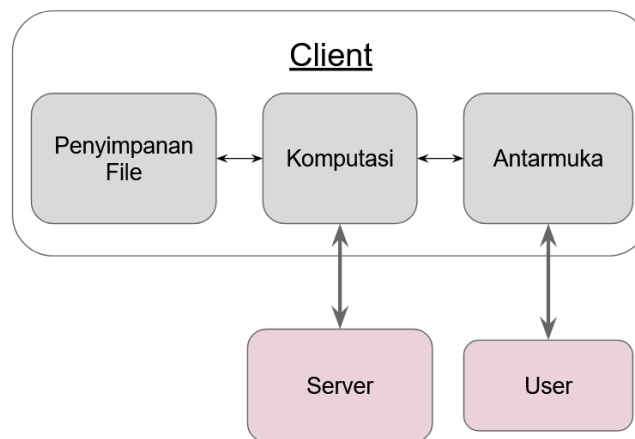
Gambar 3 Konfigurasi Database pada server

Pada struktur database di atas, terdapat tiga buah tabel, yaitu

- Tabel user yang berisi username, password, dan balance setiap akun.
- Tabel Order, yang berisi informasi setiap kali order dilakukan, yaitu nomor order, list pesanan order, status order, tanggal order, username yang melakukan pemesanan, dan catatan tambahan pada order.
- Tabel Log Request merupakan tabel yang berisi history komunikasi server dan client. Pada setiap kali request, terdapat informasi waktu yang request, username yang melakukan request, IP address, dan kalimat request.

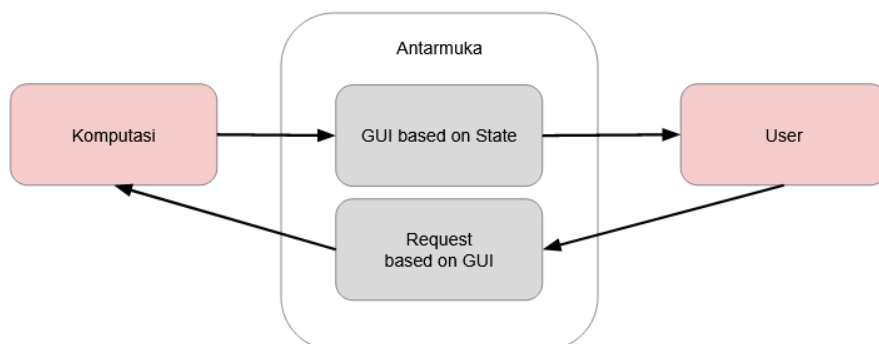
3.2 Rancangan Sistem Client

Sistem client merupakan penghubung antara server dengan user. Secara umum, sistem client ini dibagi menjadi tiga buah sub sistem, yaitu sub sistem penyimpanan file, komputasi, dan antarmuka. Hubungan ketiga sub sistem ini dapat dilihat pada gambar berikut ini.



Gambar 4 Blok Diagram Sistem Client

Seperti gambar di atas, subsistem antarmuka menjadi jalur keluar masuknya informasi oleh pengguna. Sub sistem antarmuka ini diimplementasikan pada dua jenis *platform*, yaitu pada aplikasi ponsel pintar Android untuk pelanggan restoran dan komputer untuk kasir restoran. Bagian-bagian pada subsistem antarmuka ini dapat dijelaskan melalui gambar berikut ini.

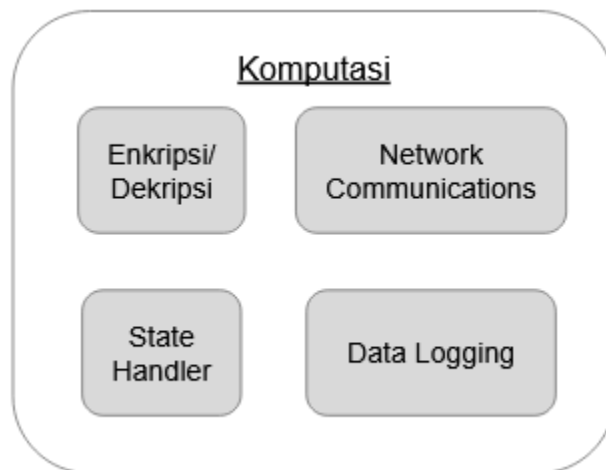


Gambar 5 Blok Diagram Subsistem Antarmuka Client

Sesuai dengan gambar di atas, subsistem antarmuka ini memiliki dua buah modul fungsional, yaitu sebagai berikut.

1. GUI based on State. Modul ini berfungsi untuk menghadirkan tampilan Graphic User Interface pada pengguna sehingga berbagai informasi keluaran sistem secara keseluruhan dapat ditunjukkan pada pengguna. Tampilan pada GUI ini pun dibagi menjadi beberapa state yang dioperasikan pada ponsel/komputer, misal : halaman *login*, pemesanan, pembayaran, *top up* saldo, dan lain sebagainya.
2. Request based on GUI. Modul ini berfungsi untuk menerima masukan dari pengguna. Ketika pengguna memberikan masukan, masukan ini diterjemahkan menjadi sebuah perintah *request* yang akan diolah sehingga dapat dipahami oleh subsistem komputasi.

Sedangkan, subsistem komputasi berfungsi untuk mengolah data yang diterima oleh GUI dan mengirimkan balasan terhadap permintaan tersebut. Untuk dapat menjawab permintaan tersebut, sub sistem komputasi ini juga perlu untuk melakukan mendapatkan informasi tambahan dari sistem server ataupun file yang disimpan secara lokal pada perangkat. Sub sistem komputasi Client ini dapat digambarkan oleh diagram berikut ini.



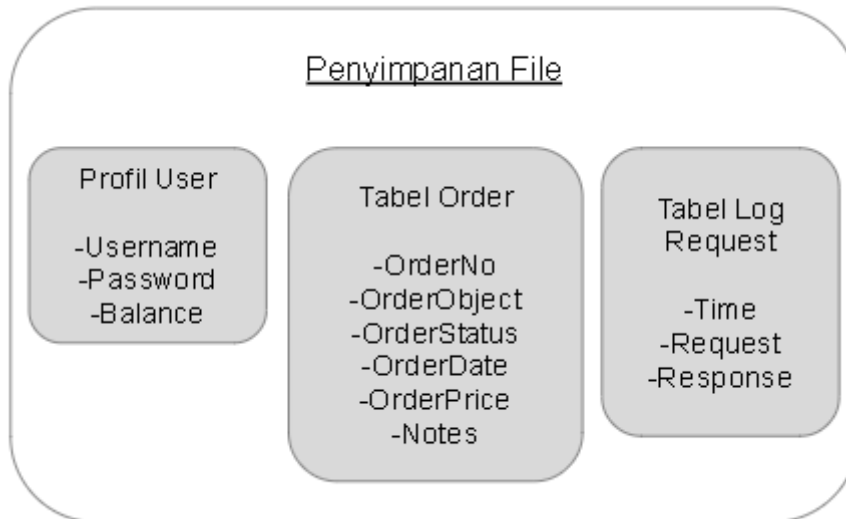
Gambar 6 Blok Diagram Subsistem Komputasi Client

Seperti gambar di atas, terdapat empat buah modul yang saling terikat dan berhubungan satu sama lain, yaitu sebagai berikut.

1. Enkripsi/dekripsi. Modul ini berfungsi untuk menambahkan fitur keamanan pada sistem sehingga data yang dikirimkan ke server dan diterima kembali oleh client dapat dilakukan enkripsi/dekripsi. Untuk keperluan enkripsi/dekripsi, digunakan SSL yang terintegrasi pada socket programming di python. Selain itu, modul ini juga berfungsi melakukan hashing pada kata sandi sehingga kata sandi tidak dapat dilihat secara eksplisit pada server.
2. State Handler. Modul ini berfungsi untuk menentukan posisi state saat ini pada client. State ini dapat berpindah-pindah sesuai dengan masukan pengguna pada sub sistem antarmuka dan tampilan antarmuka dapat berpindah pindah sesuai dengan perintah dari modul ini.
3. *Network Communications*. Modul ini bertanggungjawab terhadap proses komunikasi antara client dan server. Ketika terdapat informasi yang dibutuhkan oleh sistem client, modul ini berfungsi untuk melakukan *request* ke server sesuai dengan format JSON yang telah disepakati sehingga server dapat segera memberikan informasi yang dibutuhkan tersebut.

4. *Data Logging*. Modu ini bertanggung jawab untuk menuliskan ke file beberapa informasi pada client dan melakukan pembacaan. Dengan adanya modul ini, client tidak perlu melakukan permintaan secara terus menerus terhadap keseluruhan informasi yang dibutuhkan oleh pengguna karena ada beberapa informasi yang disimpan secara lokal pada perangkat client ini.

Selain itu, terdapat subsistem penyimpanan file pada client. File inilah yang dapat diakses oleh modul data logging pada subsistem komputasi Client. Adapun konfigurasi sub sistem ini dapat dijelaskan melalui diagram berikut.



Gambar 7 Konfigurasi Penyimpanan File pada Client

Sesuai dengan gambar di atas, konfigurasi file ini terdiri dari beberapa informasi, yaitu sebagai berikut.

- Informasi profil user, merupakan informasi login pengguna, berupa username, password, dan saldo.
- Tabel order, merupakan tabel yang menyimpan data setiap kali order dilakukan, yaitu orderno, paket makanan yang dipesan, status pemesanan, tanggal pemesanan, harga pemesanan, dan notes.
- Tabel Log Request, yaitu tabel yang menyimpan data-data yang dikirimkan ke server dan diterima dari server, yaitu waktu pengiriman request ke server, pesan yang dikirimkan, dan pesan yang diterima.

3.3 Typical Message

Message akan dikirim dan diterima antara server dan client dengan format JSON. Format request yang akan dikirimkan client adalah sebagai berikut.

{ request: ... , ... (atribut lain berdasarkan request) }

Tipe dari request dapat berupa register, login, pesan, update, delete, search, dan topup. Atribut lainnya dari *message* akan bergantung dengan tipe request seperti daftar berikut.

- Register: username, password

- Login: username, password
- Pesan: paket, quantity, status, doo, username, notes, totalHarga
- Update: orderno, paket, quantity, status, doo, username, notes, totalHarga
- Delete: orderno
- Search: tipe, username/doo/orderno/all
- Topup: username, topup

Format response yang akan dikirimkan server adalah sebagai berikut.

{ status: ... , ... (atribut lain berdasarkan response) }

Status akan menandakan apakah request yang dikirimkan client berhasil dijalankan atau tidak. Atribut lain berfungsi untuk mengakomodasi request client seperti search yang membutuhkan feedback data dari server.

4 Implementasi

4.1 Petunjuk Penggunaan

Pada masing-masing program, yaitu server, client kasir, dan client android, terdapat tiga buah prosedur yang berbeda-beda untuk menyesuaikan konfigurasi yang digunakan. Untuk menyiapkan server, dilakukan beberapa langkah sebagai berikut.

1. Pastikan bahwa pada bagian “server preparation”, IP Address dan Port yang tertulis ada sama dengan IP Address pada komputer untuk menjalankan program server ini. Misal, “192.168.100.8”.
2. Tetapkan nilai port. Pada proyek ini, digunakan port=3456.
3. Pada bagian “Constant untuk mySQL”, pastikan bahwa IP Address yang tertulis adalah sama dengan IP Address pada komputer yang menjalankan database mySQL tersebut. Misal, “192.168.100.6”.
4. Pastikan bahwa *certificate* untuk verifikasi SSL, yaitu “server.key”, “server.pem”, dan “client.pem” telah berada pada satu *directory* dengan program server.py ini.
5. Pastikan bahwa database mySQL telah dijalankan terlebih dahulu sebelum menjalankan program server ini.
6. Untuk menjalankan program server ini, masukkan perintah pada *command line* : **server.py**.

Untuk membuat database mySQL, dilakukan beberapa prosedur sebagai berikut.

1. Pastikan Mysql telah terinstall dengan baik
2. Load database yang sudah ada sesuai pada directory. Apabila database telah berhasil dimuat, instruksi nomor 7 dan 10-12 tidak perlu dijalankan. Sedangkan, apabila database gagal dimuat, ikut seluruh instruksi.
3. Jalankan [mysqld] pada cmd
4. Login ke mysql dengan [mysql -u root -p]
5. Masukan password
6. Buat user untuk server dengan [CREATE USER 'root'@'192.168.100.8' IDENTIFIED BY 'root';]

7. Buat database dengan [CREATE DATABASE restaurant;]
8. Grant akses server ke database dengan [GRANT ALL PRIVILEGES ON restaurant.* TO 'root'@'localhost';]
9. Jalankan [Flush privileges;]
10. Jalankan [use restaurant;] untuk persiapan pembuatan tabel pada database
11. Jalankan [create table if not exists ordertable(orderno INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT, paket VARCHAR(30), quantity VARCHAR(30), status VARCHAR(10), doo date, totalHarga INT(30), username VARCHAR(30), notes VARCHAR(30))AUTO_INCREMENT=1;]
12. Jalankan [create table if not exists usertable(Id INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT, username VARCHAR(30), password VARCHAR(65), saldo INT(30))AUTO_INCREMENT=1;]

Untuk menyiapkan client pada kasir, dilakukan beberapa langkah sebagai berikut.

1. Pastikan bahwa pada bagian “Konfigurasi Socket”, IP Address dan Port yang tertulis ada sama dengan IP Address pada komputer untuk menjalankan program server. Misal, “192.168.100.8”.
2. Tetapkan nilai port. Pada proyek ini, digunakan port=3456.
3. Pastikan bahwa *certificate* untuk verifikasi SSL, yaitu “client.key”, “client.pem”, dan “server.pem” telah berada pada satu *directory* dengan program login.py dan main.py.
4. Pastikan program server telah dijalankan terlebih dahulu sebelum menjalankan program client ini.
5. Untuk menjalankan program client ini, masukkan perintah pada *command line* : **restaurant_client.py**.

Untuk menyiapkan client pada Android, dilakukan beberapa langkah sebagai berikut.

1. Buka *project* Android menggunakan Android Studio.
2. Pada browser, terdapat file Constant.java. Pada file tersebut, pastikan bahwa IP Address dan Port server sesuai dengan IP Address yang tertulis pada konstanta server_addr dan server_port. Dalam hal ini, digunakan server_addr=”192.168.100.8” dan server_port=3456.
3. Untuk menjalankan aplikasi ini, dapat digunakan menu Run → Run App. Selain itu, aplikasi ini juga dapat diekspor menjadi APK dengan Build → Build Bundle(s)/Apk(s) → Build Apk(s)

4.2 Sourcecode

4.2.1 Sourcecode Server

Program server dituliskan menggunakan bahasa python dengan nama file server.py. Source code ini dilampirkan di Lampiran.

4.2.2 Sourcecode Client Kasir

Pada client kasir, digunakan *source code* dari 3 file berbeda yang dipisahkan berdasarkan tampilan UI. Tiga file tersebut adalah `restaurant_client.py`, `login.py`, dan `main.py`. Ketiga *source code* ini dapat dilihat pada Lampiran.

4.2.3 Sourcecode Client Pelanggan

Pada client pelanggan, terdapat 14 file berekstensi `.java` yang digunakan untuk mendefinisikan *activity-activity* pada client pelanggan dan dilengkapi dengan 9 file *layout .xml*. File *source code java* ini dapat dilihat pada Lampiran.

4.3 Tampilan UI

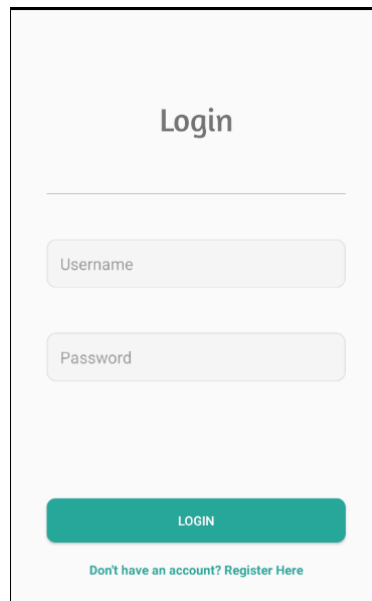
4.3.1 Client Pelanggan

Pada saat pengguna membuka aplikasi, maka akan muncul *landing page* sebagai berikut.



Gambar 8 Tampilan awal aplikasi client pelanggan

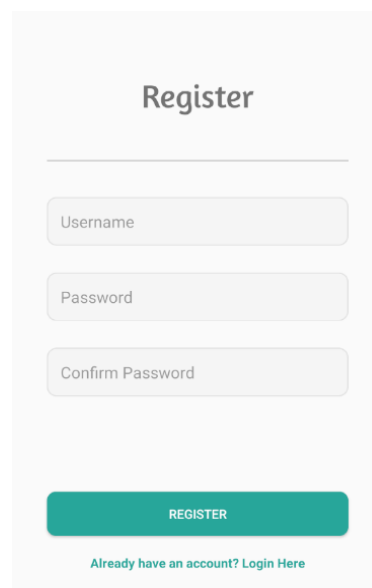
Setelah itu, apabila pelanggan sudah login, maka pengguna akan dialihkan langsung masuk ke halaman utama. Jika belum, maka pengguna akan ditampilkan halaman login sebagai berikut.

The image shows a login form with a light gray background. At the top, the word "Login" is centered in a bold, dark gray font. Below it is a horizontal separator line. There are two input fields: "Username" and "Password", both with light gray borders and placeholder text. Below the password field is a teal-colored button with the word "LOGIN" in white, uppercase letters. At the bottom, there is a link that says "Don't have an account? Register Here" in a small, teal-colored font.

Gambar 9 Tampilan halaman login aplikasi pelanggan

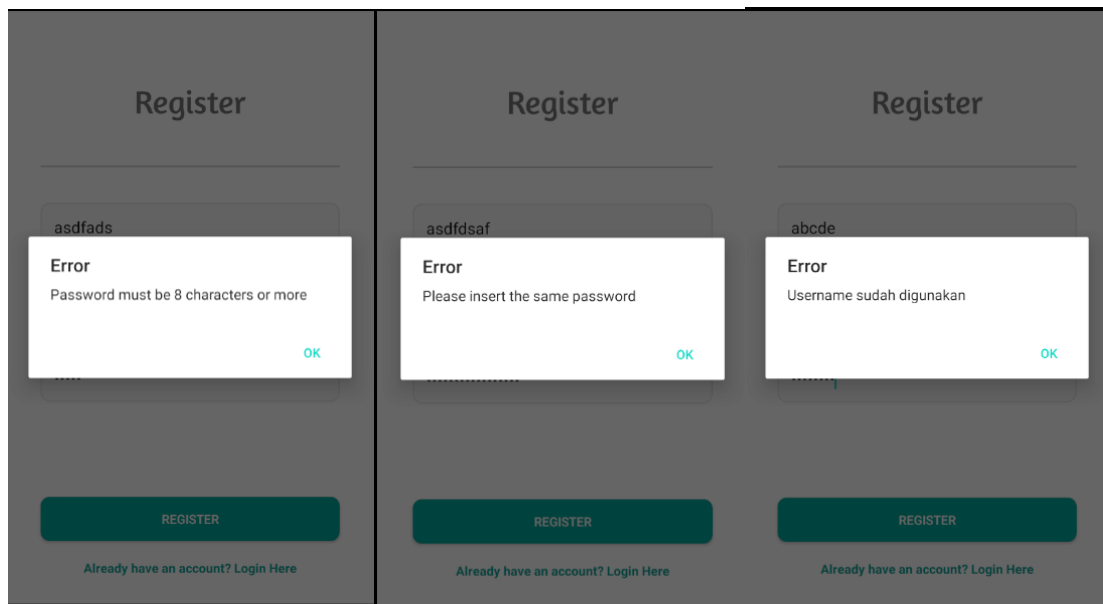
Pada halaman ini, terdapat dua *field* yang perlu diisi yaitu username dan password. Jika sudah diisi dengan benar, maka pengguna dapat langsung masuk ke halaman utama. Jika tidak, maka pengguna dapat mendaftar terlebih dahulu dengan menekan tulisan "Register here" di bagian bawah tombol login.

Tampilan dari halaman register dapat dilihat sebagai berikut.

The image shows a register form with a light gray background. At the top, the word "Register" is centered in a bold, dark gray font. Below it is a horizontal separator line. There are three input fields: "Username", "Password", and "Confirm Password", all with light gray borders and placeholder text. Below the "Confirm Password" field is a teal-colored button with the word "REGISTER" in white, uppercase letters. At the bottom, there is a link that says "Already have an account? Login Here" in a small, teal-colored font.

Gambar 10 Tampilan halaman pendaftaran akun client pelanggan

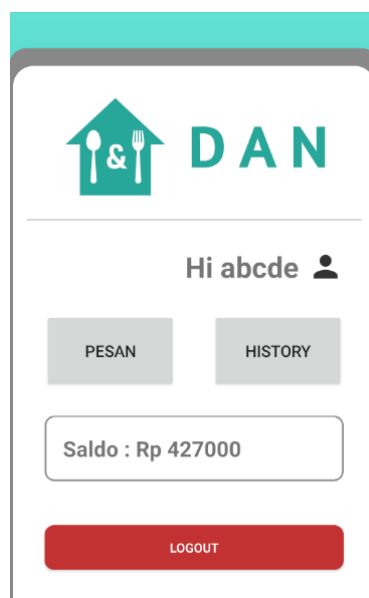
Tampilan dari halaman register ini cukup mirip dengan halaman login, hanya saja ada tambahan satu *field* yaitu confirm password yang digunakan untuk menulis ulang password. Jika pengguna berhasil melakukan pendaftaran, maka pengguna akan langsung masuk ke halaman utama. Jika tidak, maka dapat muncul pesan kesalahan sebagai berikut.



Gambar 11 Tampilan verifikasi username/password saat register

Pesan kesalahan pada gambar kiri muncul ketika pengguna memasukkan password yang panjangnya kurang dari 8 karakter. Pesan kesalahan gambar tengah muncul ketika pengguna memasukkan hal yang berbeda pada *field* password dan *field* confirm password. Pesan kesalahan gambar kanan muncul ketika pengguna hendak memasukkan username yang sudah ada pada *database*.

Setelah berhasil melakukan semua hal itu, maka pengguna dapat masuk ke halaman utama dengan tampilan berikut.

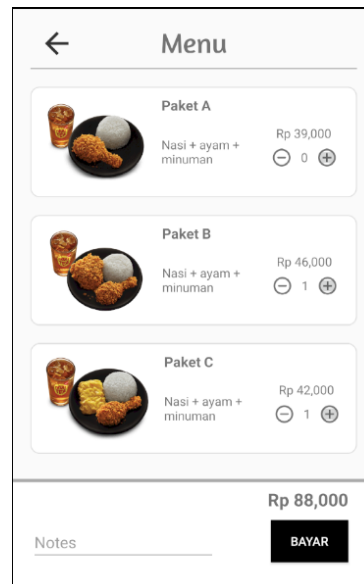


Gambar 12 Tampilan utama aplikasi Android pelanggan

Pada halaman utama, terdapat beberapa elemen utama. Di bagian tengah kanan, terdapat kata-kata “Hi <username>” untuk hiasan semata. Di bagian bawah dari tulisan tersebut,

terdapat dua tombol yaitu tombol pesan untuk memesan makanan dan tombol history untuk melihat history pemesanan makanan. Di bawah kedua tombol tersebut, terdapat sebuah elemen yang menunjukkan jumlah saldo yang dimiliki saat ini. Pada bagian paling bawah dari halaman tersebut, terdapat tombol logout untuk logout akun.

Saat pengguna menekan tombol pesan, maka akan muncul halaman pesan seperti pada gambar di bawah ini.



Gambar 13 Pilihan menu

Pada halaman tersebut, akan muncul beberapa pilihan menu. Masing-masing dari menu tersebut dapat ditambah ataupun kurang melalui tombol kurang dan tambah yang bersesuaian. Total harga dari menu yang dipesan akan muncul pada bagian kanan bawah dari halaman tersebut. Di bagian kiri bawah, terdapat sebuah *field* yang bisa digunakan untuk menulis notes dari pesanan. Jika sudah selesai, maka pengguna dapat menekan tombol bayar pada pojok kanan bawah.

Setelah menekan tombol pembayaran, maka pengguna akan diantarkan ke halaman checkout dengan tampilan sebagai berikut.

The screenshot shows a mobile application interface for a checkout page. At the top, there is a teal header. Below it, the title 'Halaman Checkout' is centered. A white box contains the following information: Subtotal : Rp 88,000, Saldo : Rp 427,000, and Sisa Saldo : Rp 339,000. Below this box, the text 'Masukkan kata sandi :' is followed by a password input field labeled 'Password'. At the bottom, there is a grey button labeled 'KONFIRMASI PEMBAYARAN'.

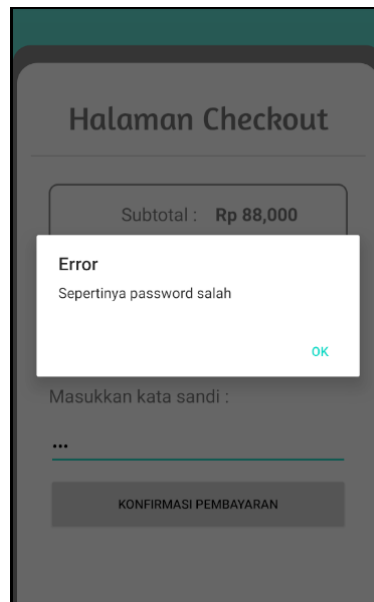
Gambar 14 Tampilan konfirmasi pembayaran

Pada halaman ini, pengguna akan diminta untuk mengkonfirmasi kembali pesanannya. Untuk itu, ditampilkan nilai dari saldo pengguna, total yang harus dibayarkan, dan sisa saldo pengguna apabila pengguna jadi memesan. Untuk mengkonfirmasi, pengguna diminta untuk memasukkan kembali password pengguna pada *field* yang disediakan kemudian menekan tombol konfirmasi pembayaran di bawahnya. Apabila password benar, maka pengguna akan diberikan pesan konfirmasi beserta nomor ordernya seperti pada gambar berikut.

This screenshot shows the same checkout page as Gambar 14, but with a white modal overlay in the center. The modal has the title 'Berhasil' and the text 'Pesananmu sudah tercatat dengan order no 32'. There is an 'OK' button in the bottom right corner of the modal. The background of the page is dimmed.

Gambar 15 Tampilan pemesanan

Namun, apabila terjadi kesalahan pada penginputan password, maka akan muncul pesan kesalahan sebagai berikut.



Gambar 16 Verifikasi password saat pemesanan

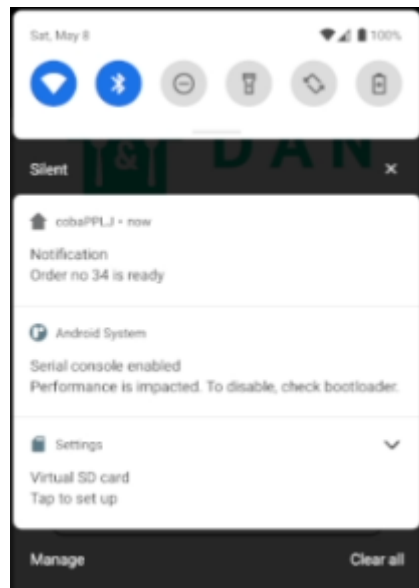
Jika pengguna kembali ke halaman utama dan menekan tombol history, pengguna dapat melihat kembali history pesanannya dengan format sebagai berikut.

The screenshot shows a mobile application interface for a history page. At the top, there is a back arrow and the title 'History'. Below the title, a list of orders is displayed. Each order entry includes the order number, date, status, items, notes, and price.

Orderno	Date	Status	Pesanan	Notes	Harga
Orderno : 13	2021-04-25	Status : selesai	Pesanan : B:1, C:2, A:3	Notes : abcde	Harga : Rp 110000
Orderno : 14	2021-04-25	Status : selesai	Pesanan : B:1, C:2	Notes :	Harga : Rp 110000
Orderno : 15	2021-04-25	Status : selesai	Pesanan : B:1, C:1	Notes :	Harga : Rp 78000
Orderno : 16	2021-04-25	Status : selesai	Pesanan : A:2, B:1	Notes : Tambah 1	Harga : Rp 124000
Orderno : 17	2021-04-25	Status : selesai	Pesanan : A:4, B:4, C:4	Notes : bjk	Harga : Rp 468000
Orderno : 19	2021-05-02				

Gambar 17 Tampilan history pesanan

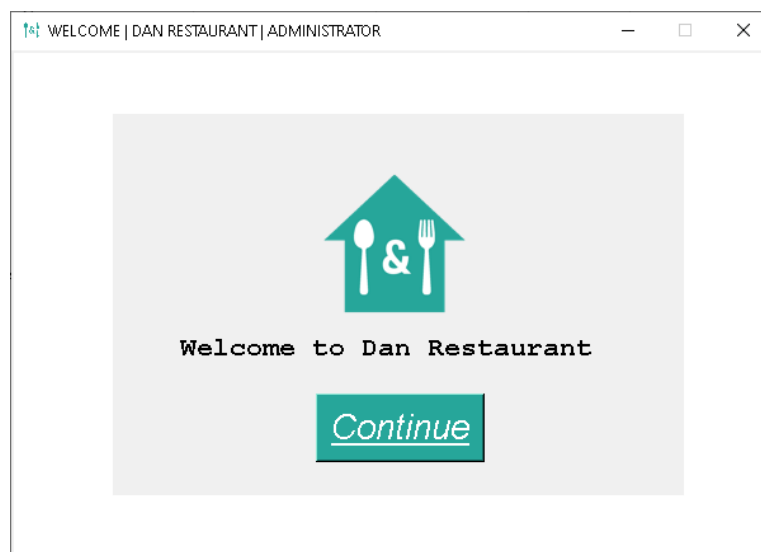
Apabila order telah dilakukan update, terdapat notifikasi pada ponsel seperti gambar berikut ini.



Gambar 18 Notifikasi apabila pesanan telah siap

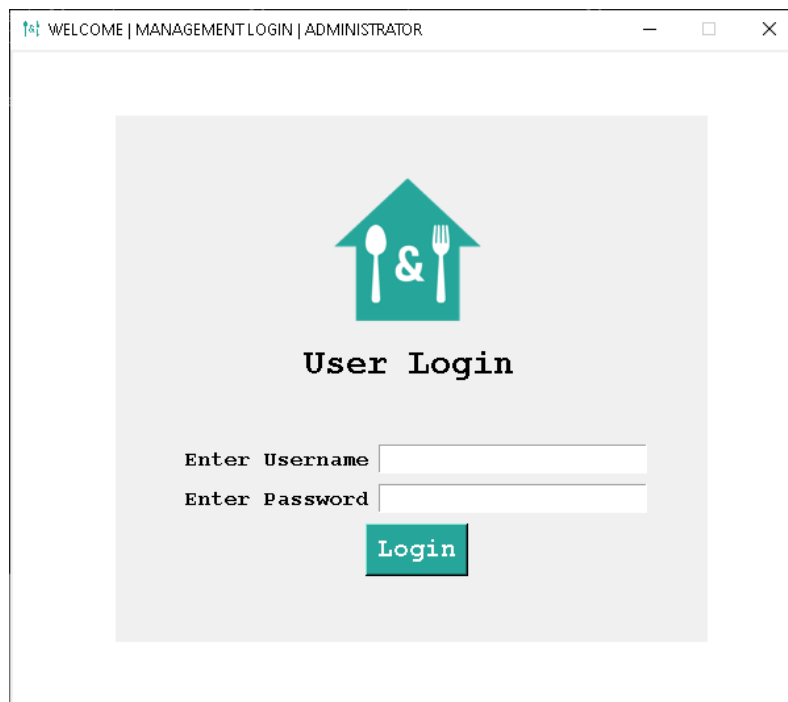
4.3.2 Client Kasir

Pada client kasir, terdapat beberapa tampilan UI. Tampilan tersebut adalah layar sambutan, layar login, layar utama, dan layar top-up. Berikut adalah tampilan dan penjelasan lebih lanjut dari tiap layar.



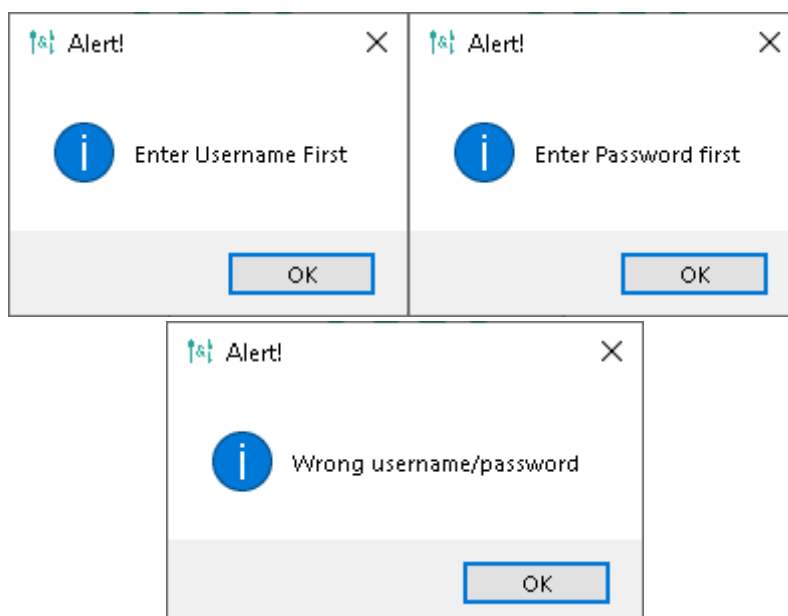
Gambar 19 Tampilan awal program client kasir

Layar sambutan adalah layar pertama yang akan ditampilkan saat user membuka aplikasi client kasir. Layar ini hanya berfungsi sebagai penanda bahwa aplikasi berhasil dijalankan dan layar pengantar ke layar login. Jika ingin melanjutkan ke layar login, user harus menekan tombol 'continue'. Ketika tombol tersebut ditekan, layar ini akan otomatis tertutup.



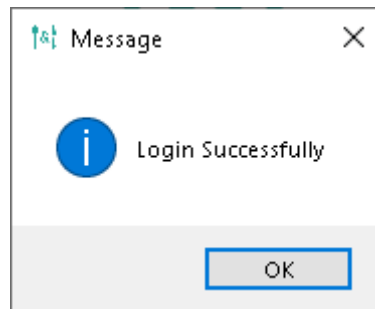
Gambar 20 Tampilan login kasir

Layar login adalah layar dimana user dapat melakukan login. Untuk melakukan login, user harus memasukan username dan password. Username dan password yang dimasukan harus terdaftar sebagai client kasir di database server. Untuk melakukan request login, tombol 'login' harus ditekan setelah username dan password diisi.



Gambar 21 Verifikasi username password pada kasir

Jika username tidak dimasukan, password tidak dimasukan, atau username dan password tidak sesuai dengan yang ada di database, maka akan ditampilkan pesan sebagai berikut.



Gambar 22 Pop up window apabila login berhasil

Jika login berhasil, maka akan ditampilkan pesan login berhasil dan layar ini akan otomatis tertutup.

DAN RESTAURANT | ORDER MANAGEMENT | ADMINISTRATOR

Order Management System

Paket:

B

C

Quantity:

1

1

Order Status:

ready

▼

Date of Order:

2021-05-02

▼

User:

abcde

Notes:

Update

Delete

Clear

Show All

Please select one record below to update or delete

orderno	Paket	Quantity	Status	Date of Order	Harga	user	notes
32	B;C	1;1	paid	2021-05-08	88000	abcde	
31	B;C	1;1	paid	2021-05-02	88000	abcde	
30	B;C	1;1	ready	2021-05-02	88000	abcde	
29	A;B	1;1	ready	2021-05-02	85000	abcde	
28	C	2	ready	2021-05-02	84000	abcde	
27	C	2	ready	2021-05-02	84000	abcde	
26	B	1	ready	2021-05-02	46000	abcde	
25	B	1	ready	2021-05-02	46000	abcde	

Please Enter Date of Order:

2021-05-08

▼

Search

Please Enter Order No:

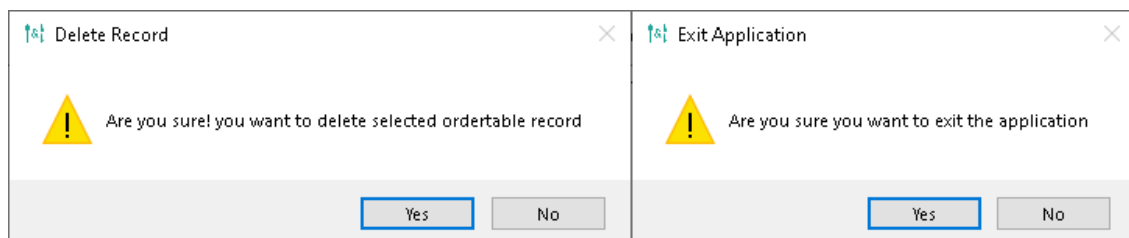
Search

Exit

Gambar 23 Tampilan utama program kasir

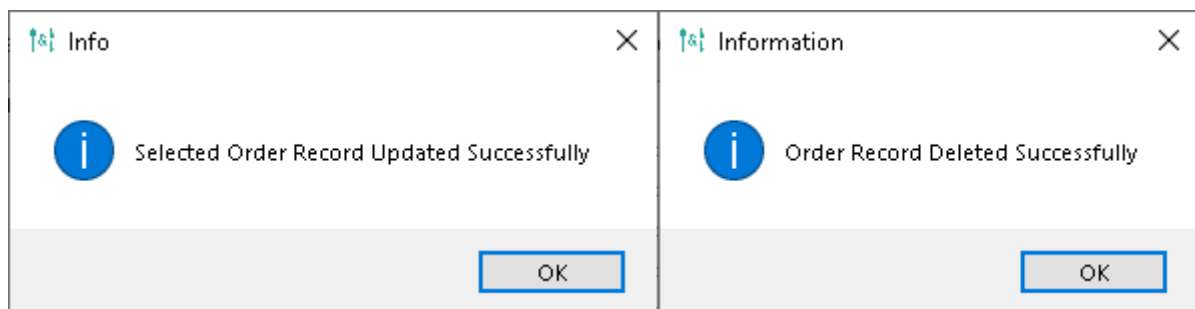
Layar utama adalah layar dimana user kasir akan melakukan sebagian besar dari aktivitas utama. Pada layar ini terlihat beberapa bagian yang memiliki fungsi yang berbeda. Pada bagian atas, tampak beberapa tempat input kosong yang memiliki label atribut dari pesanan. Tempat input ini akan digunakan untuk menampilkan atribut dari suatu pesanan tertentu dan dapat juga digunakan untuk mengganti atribut dari pesanan tersebut. Pada bagian bawahnya, terdapat beberapa tombol, yaitu tombol 'update', 'delete', 'clear', dan 'show all'. Tombol

'update' berfungsi untuk mengirimkan request memperbaharui atribut dari suatu pesanan tertentu. Tombol 'delete' berfungsi untuk mengirimkan request menghapus suatu pesanan tertentu. Tombol 'clear' berfungsi untuk menghapus teks yang ada pada tempat input yang telah terisi. Tombol 'show all' berfungsi untuk mengirimkan request menampilkan daftar pesanan yang ada. Pada bagian bawahnya lagi, terdapat tempat untuk menampilkan daftar pesanan yang ada yang diperoleh dari server. Tempat ini mempunyai fitur scroll vertikal dan horizontal pada bagian kanan dan bawahnya. Pada bagian bawahnya, terdapat bagian untuk melakukan pencarian pesanan yang spesifik. Pencarian pesanan dapat dilakukan dengan menginput atribut tanggal pesanan atau nomor pesanan. Tombol 'search' pada bagian kanannya berfungsi untuk mengirimkan request pencarian berdasarkan atribut yang telah dimasukkan. Hasil pencarian akan ditampilkan pada tempat menampilkan daftar pesanan. Pada bagian paling bawah, terdapat tombol 'exit' yang berfungsi untuk menutup aplikasi client kasir. Pada bagian kanan atas, terdapat tombol 'top-up' yang berfungsi untuk membuka layar top-up. Jika layar top-up dibuka, layar utama tidak otomatis tertutup.



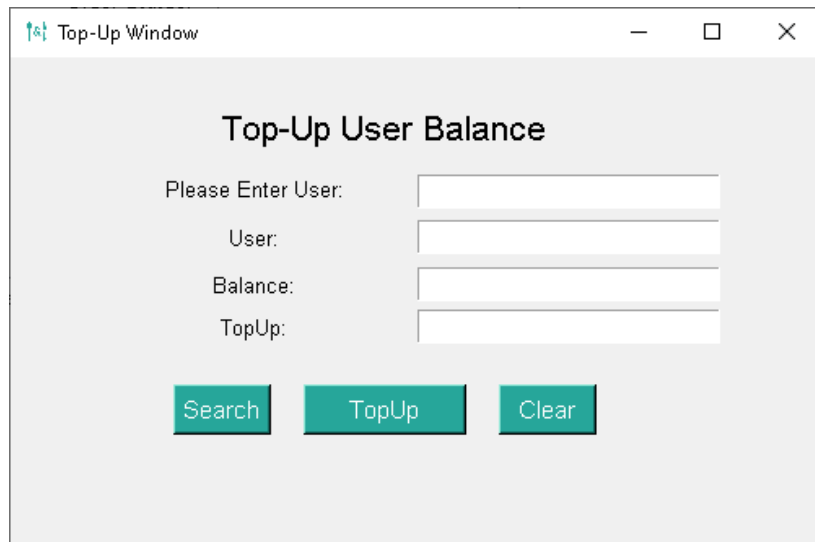
Gambar 24 Pop up window untuk menghapus order dan keluar

Jika tombol 'delete' atau 'exit' ditekan, maka akan ada konfirmasi seperti yang ditampilkan diatas. Jika user menekan 'yes' maka command dari tombol akan dieksekusi. Jika user menekan 'no' maka command dari tombol tidak akan dieksekusi.



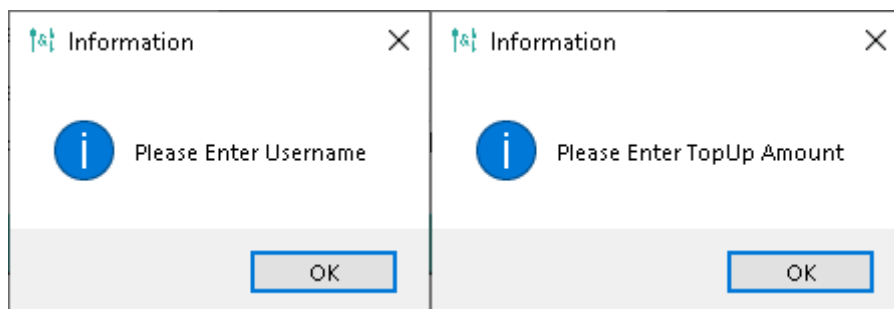
Gambar 25 Pop up window apabila order telah di *update* atau dihapus

Jika command update dan delete telah berhasil dilaksanakan, akan ditampilkan pesan berhasil seperti pada gambar diatas.



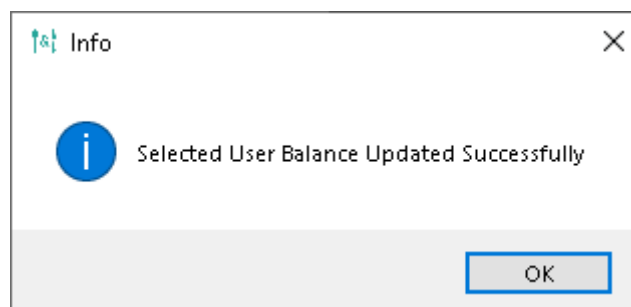
Gambar 26 Jendela untuk melakukan *top up* saldo

Layar top-up adalah layar dimana user kasir dapat melakukan berbagai aktivitas untuk fitur top-up. Pada bagian paling atas, terdapat tempat input untuk melakukan pencarian username tertentu. Pada bagian bawahnya, terdapat tempat untuk menampilkan informasi username dan balance dari user yang dicari. Pada bagian bawahnya, terdapat tempat input untuk jumlah angka yang akan di topup ke balance dari user yang sedang ditampilkan. Pada bagian paling bawah, terdapat 3 tombol, yaitu 'search', 'topup', dan 'clear'. Tombol 'search', berfungsi untuk memberikan request pencarian user. Tombol 'topup' berfungsi untuk memberikan request penambahan balance user berdasarkan jumlah topup yang diinput. Tombol 'clear' berfungsi untuk menghapus teks yang ada pada semua tempat input yang ada pada layar top-up.



Gambar 27 Verifikasi pengisian informasi saat top up saldo

Jika username atau topup amount tidak dimasukan pada saat menekan tombol 'search' atau 'topup', maka akan ditampilkan pesan sebagai berikut.



Gambar 28 Pop up window bahwa

Jika command topup telah berhasil dilaksanakan, akan ditampilkan pesan berhasil seperti pada gambar diatas.

4.4 Hasil Ujicoba

Uji coba dilakukan menggunakan video dengan nama file “demoRestoMgmtDAN.mp4”.

LAMPIRAN

Server

server.py

```
import socket
import os
from _thread import *
from datetime import datetime
import sys
import json
import mysql.connector
import ssl
import traceback

#server preparation
ServerSideSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '192.168.100.8'
port = 3456
serverkey='server.key'
serverpem='server.pem'
clientpem='client.pem'
ssl_version=None

#Constant untu mySQL
hostSQL="192.168.100.6",

#Fungsi Pendukung
#SSL
def ssl_wrap_socket(sock, ssl_version=None, keyfile=None, certfile=None,
ciphers=None):

    #1. init a context with given version(if any)
    if ssl_version is not None and ssl_version in version_dict:
        #create a new SSL context with specified TLS version
        sslContext = ssl.SSLContext(version_dict[ssl_version])
        if option_test_switch == 1:
            print ("ssl_version loaded!! =", ssl_version)
    else:
        #if not specified, default
        sslContext = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)

    if ciphers is not None:
        #if specified, set certain ciphersuite
        sslContext.set_ciphers(ciphers)
        if option_test_switch == 1:
            print ("ciphers loaded!! =", ciphers)

    #server-side must load certfile and keyfile, so no if-else
    sslContext.load_cert_chain(certfile, keyfile)
    print ("ssl loaded!! certfile=", certfile, "keyfile=", keyfile)

    try:
        return sslContext.wrap_socket(sock, server_side = True)
    except ssl.SSLError as e:
```

```

        print ("wrap socket failed!")
        print (traceback.format_exc())

#Mencari maximum order number di mysql
def fetch_max_order_no():
    if db_connection.is_connected() == False:
        db_connection.connect()
    #db_cursor.execute("use restaurant") # Interact with restaurant databse
    orderno = 0
    query1 = "SELECT orderno FROM ordertable order by orderno DESC LIMIT 1"
    # implement query Sentence
    db_cursor.execute(query1)
    print("No of Record Fetched:" + str(db_cursor.rowcount))
    if db_cursor.rowcount == 0:
        orderno = 1
    else:
        rows = db_cursor.fetchall()
        for row in rows:
            orderno = row[0]
        orderno = orderno + 1
    print("Max Order Id: " + str(orderno))
    return orderno

#definiskan fungsi callbacknya
#Kompilasi callback
def registerCb(jsonData): #untuk menerima request dari state register
    if db_connection.is_connected() == False:
        db_connection.connect()
    username=jsonData['username']
    password=jsonData['password']
    saldo='0'
    #verifikasi apakah username sudah terdaftar di mysql
    sqlQuery="SELECT * from usertable where username='%s'" %(username)
    db_cursor.execute(sqlQuery)
    data=db_cursor.fetchall()
    if (data!=[]):
        skirim={
            'response':'register',
            'status':'gagal',
            'keterangan' : "username sudah terdaftar sebelumnya"
        }
        return skirim
    try:
        #daftarkan
        sqlQuery="INSERT INTO usertable (username, password, saldo) VALUES
(%s, %s, %s)"
        db_cursor.execute(sqlQuery, (username, password, saldo))
        db_connection.commit()
    except:
        skirim={
            'response':'register',
            'status':'gagal'
        }
    else:
        skirim={
            'response':'register',
            'status':'berhasil'
        }
    print("username "+username+" berhasil didaftarkan dengan saldo

```



```

"+str(saldo))
    return skirim

def loginCb(jsonData): #memproses request dari state login
    #ambil data password
    if db_connection.is_connected() == False:
        db_connection.connect()
    username=jsonData["username"]
    password=jsonData["password"]
    try:
        #ambil password
        sqlQuery="SELECT password from usertable where username='%s'" %(
            username)
        db_cursor.execute(sqlQuery)
        passwordRef=db_cursor.fetchone()[0]
        #ambil saldo
        sqlQuery="SELECT saldo from usertable where username='%s'" %(
            username)
        db_cursor.execute(sqlQuery)
        saldo=db_cursor.fetchone()[0]
    except:
        skirim={
            'response': 'login',
            'status' : 'gagal',
            'keterangan' : 'umum'
        }
        print("login gagal umum")
    else :
        if (passwordRef==password):
            skirim={
                'response': 'login',
                'status' : 'berhasil',
                'saldo' : saldo
            }
        else :
            skirim={
                'response': 'login',
                'status' : 'gagal',
                'keterangan': 'password salah'
            }
            print("Password salah")
    return skirim

def pesanCb(jsonData): #memproses request dari state order/pesan
    if db_connection.is_connected() == False:
        db_connection.connect()
    orderno=int(fetch_max_order_no())
    paket=jsonData['paket']
    quantity=jsonData['quantity']
    status='paid'
    doo=str(datetime.now().strftime("%Y-%m-%d"))
    username=jsonData['username']
    password=jsonData['password']
    notes=jsonData['notes']
    totalHarga=jsonData['totalHarga']
    #Verifikasi password
    try:
        sqlQuery="SELECT password from usertable where username='%s'" %(
            username)
        db_cursor.execute(sqlQuery)

```

```

        passwordRef=db_cursor.fetchone()[0]
    except:
        skirim={
            'response': 'pesan',
            'status' : 'gagal',
            'keterangan' : 'verifikasi password gagal'
        }
        print("verifikasi pesan gagal umum")
        return skirim
    else :
        if (passwordRef!=password):
            skirim={
                'response': 'pesan',
                'status' : 'gagal',
                'keterangan': 'password salah'
            }
            return skirim
            print("Password salah")
#ambil saldo saat ini
sqlQuery="SELECT saldo from usertable where username='%s'" %(
    username)
db_cursor.execute(sqlQuery)
saldo=int(db_cursor.fetchone()[0])
if (saldo>=int(totalHarga)):
    saldo=saldo-int(totalHarga)
else:
    skirim={
        'response': 'pesan',
        'status': 'gagal',
        'keterangan' : 'saldo tidak cukup'
    }
    return skirim
#update saldo
sqlQuery="Update usertable set saldo='%s' where username='%s'" %(
    saldo, username)
db_cursor.execute(sqlQuery)
db_connection.commit()
#Menambahkan Order
try:
    sqlQuery="INSERT INTO ordertable (orderno, paket, quantity, status,
doo, totalHarga, username, notes) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    db_cursor.execute(sqlQuery, (orderno, paket, quantity, status, doo,
totalHarga, username, notes))
    db_connection.commit()
except mysql.connector.Error as err:
    print(err)
else:
    print("Order "+str(orderno)+ " berhasil ditambahkan")
#pesan berhasil
skirim={
    'response': 'pesan',
    'status': 'berhasil',
    'orderno': str(orderno)
}
return skirim

def updateCb(jsonData): #memproses request saat melakukan update
    if db_connection.is_connected() == False:
        db_connection.connect()
    orderno=jsonData['orderno']

```

```

paket=jsonData['paket']
quantity=jsonData['quantity']
status=jsonData['status']
username=jsonData['username']
notes=jsonData['notes']
totalHarga=jsonData['totalHarga']
#mengambil data harga sebelumnya
sqlQuery="SELECT totalHarga from ordertable where orderno=%s"%(
    orderno)
db_cursor.execute(sqlQuery)
data=db_cursor.fetchone()[0]
prevHarga=int(data)
#mengambil saldo saat ini
try:
    sqlQuery="SELECT saldo from usertable where username='%s'"%(
        username)
    db_cursor.execute(sqlQuery)
    prevsaldo=int(db_cursor.fetchone()[0])
    saldo=prevsaldo-int(totalHarga)+prevHarga
    if (saldo<0):
        skirim={
            'response': 'update',
            'status' : 'saldo kurang'
        }
        print("saldo kurang")
        return skirim
    #update saldo
    sqlQuery="Update usertable set saldo=%s where username='%s'" %(
        saldo, username)
    db_cursor.execute(sqlQuery)
    db_connection.commit()
    #update ordertable
    #db_cursor.execute("use ordertable")
    sqlQuery="Update ordertable set paket='%s', quantity='%s',
status='%s', username='%s', totalHarga='%s', notes='%s' where orderno=%s" %(
        paket, quantity, status, username, totalHarga, notes, orderno)
    db_cursor.execute(sqlQuery)
    db_connection.commit()
except:
    skirim={
        'response': 'update',
        'status' : 'gagal'
    }
else:
    skirim={
        'response': 'update',
        'status' : 'berhasil'
    }
    print("Berhasil update ordertable untuk orderno="+str(orderno))
return skirim

def deleteCb(jsonData): #memproses request delete order
    orderno=jsonData['orderno']
    #Pastikan data ada
    sqlQuery="SELECT * from ordertable where orderno=%s" %(orderno)
    db_cursor.execute(sqlQuery)
    data=db_cursor.fetchall()
    if (data!=[]):
        sqlQuery="DELETE from ordertable where orderno=%s" %(orderno)
        db_cursor.execute(sqlQuery)

```

```

        db_connection.commit()
        print("Berhasil menghapus orderno="+str(orderno))
        skirim={
            'response': 'delete',
            'status': 'berhasil'
        }
    else:
        skirim={
            'response': 'delete',
            'status': 'berhasil',
            'keterangan': 'data tidak ada'
        }
        print ("Gagal menghapus data karena orderno="+str(orderno)+" tidak
ada")
    return skirim

def searchCb(jsonData): #memproses request saat diminta search
    if (jsonData['tipe']=='orderno'):
        orderno=jsonData['orderno']
        sqlQuery="SELECT * from ordertable where orderno=%s" %(orderno)
        db_cursor.execute(sqlQuery)
        data=db_cursor.fetchall()
        skirim={
            'response': 'search',
            'status' : 'berhasil',
            'tipe' : 'orderno',
            'isi' : data
        }
        print("Pencarian orderno="+str(orderno)+" selesai.")
    elif (jsonData['tipe']=='tanggal'):
        tanggal=jsonData['tanggal']
        sqlQuery="SELECT * from ordertable where doo='%s'" %(tanggal)
        db_cursor.execute(sqlQuery)
        data=db_cursor.fetchall()
        skirim={
            'response': 'search',
            'status' : 'berhasil',
            'tipe' : 'tanggal',
            'isi' : data
        }
        print("Pencarian tanggal="+tanggal+" selesai.")
    elif (jsonData['tipe']=='all'):
        sqlQuery="SELECT * from ordertable"
        db_cursor.execute(sqlQuery)
        data=db_cursor.fetchall()
        print(data)
        skirim={
            'response': 'search',
            'status' : 'berhasil',
            'tipe' : 'all',
            'isi' : data
        }
    elif (jsonData['tipe']=='username'):
        username=jsonData['username']
        sqlQuery="Select saldo from usertable where username='%s'"
%(username)
        db_cursor.execute(sqlQuery)
        data=db_cursor.fetchone()
        if (data!=None):
            skirim={

```

```

        'response': 'search',
        'status' : 'berhasil',
        'tipe' : 'username',
        'isi' : str(data[0])
    }
    else :
        skirim={
            'response': 'search',
            'status' : 'gagal',
            'tipe' : 'username',
            'isi' : ''
        }

    else:
        print("Search gagal, tipe tidak terdefinisi")
        skirim={
            'response': 'search',
            'status': 'gagal'
        }
    return skirim

def topupCb(jsonData): #memproses request untuk topup
    username=jsonData['username']
    topup=jsonData['topup']
    sqlQuery="SELECT saldo from usertable where username='%s'"%(
        username)
    db_cursor.execute(sqlQuery)
    prevsaldo=int(db_cursor.fetchone()[0])
    if (prevsaldo==None):
        skirim={
            'response' : 'topup',
            'status' : 'gagal',
            'keterangan' : 'username tidak ditemukan'
        }
        return skirim
    saldo=prevsaldo+int(topup)
    #update saldo
    sqlQuery="Update usertable set saldo=%s where username='%s' " %(
        saldo, username)
    db_cursor.execute(sqlQuery)
    db_connection.commit()
    skirim={
        'response' : 'topup',
        'status' : 'berhasil',
        'saldo' : str(saldo)
    }
    return skirim

def showHistory(jsonData): #memproses request untuk update history
    if db_connection.is_connected() == False:
        db_connection.connect()
    username=jsonData['username']
    sqlQuery = "SELECT * FROM ordertable where username='%s' order by
cast(orderno as unsigned) " %(username)
    # implement query Sentence
    db_cursor.execute(sqlQuery)
    data=db_cursor.fetchall()
    skirim={
        'response': 'showHistory',
        'isi' : data,

```

```

        'jumlah' : len(data)
    }
    #db_connection.close()
    return skirim

def switchReq(jsonData): #memproses banyaknya request ke fungsi callback
masing-masing
    db_cursor.execute("use restaurant")
    if (jsonData["request"]=="register"):
        x=registerCb(jsonData)
    elif (jsonData["request"]=="login"):
        x=loginCb(jsonData)
    elif (jsonData["request"]=="pesan"):
        x=pesanCb(jsonData)
    elif (jsonData["request"]=="update"):
        x=updateCb(jsonData)
    elif (jsonData["request"]=="delete"):
        x=deleteCb(jsonData)
    elif (jsonData["request"]=="search"):
        x=searchCb(jsonData)
    elif (jsonData["request"]=="topup"):
        x=topupCb(jsonData)
    elif (jsonData["request"]=="showHistory"):
        x=showHistory(jsonData)
    else:
        x="Invalid Request Data"
    return x

def multi_threaded_client(connection, address): #fungsi multi threading
    try:
        data = connection.recv(4096)
        file_object=open('logFile.txt', 'a')
        sdata=str(data,'utf-8')
        receiveTime=str(datetime.now())

        print("Pesan dari",address[0],":",address[1]," =", sdata)
        jsonData=json.loads(sdata)
        print("jsonData = "+str(jsonData))
        skirim=switchReq(jsonData)
        sendTime=str(datetime.now())
        writeTxt="Client :
"+str(address[0])+" : "+str(address[1])+"\nR"+receiveTime+"
"+sdata+"\nS"+sendTime+" "+str(skirim)+"\n\n"
        skirim["receiveTime"]=receiveTime
        skirim["sendTime"]=sendTime

        file_object.write(writeTxt)
        print("skirim : "+str(skirim))
        connection.sendall(bytes(json.dumps(skirim, sort_keys=True,
default=str), 'utf-8'))
    finally:
        connection.close()
        print("koneksi dengan "+address[0]+" ditutup\n")
        file_object.close()

#melakukan akses ke database mysql
db_connection = mysql.connector.connect(
    host=hostSQL,
    user="root",
    password="password",

```

```
    auth_plugin='mysql_native_password')
# creating database_cursor to perform SQL operation
db_cursor = db_connection.cursor(buffered=True)

try:
    ServerSideSocket.bind((host, port))
except socket.error as e:
    print(str(e))

print('Socket is listening..'+"PORT="+str(port))
ServerSideSocket.listen(5)

while True:
    Client, address = ServerSideSocket.accept()
    connectionSocket = ssl_wrap_socket(Client, ssl_version, serverkey,
serverpem)
    #print('Connected to: ' + address[0] + ':' + str(address[1]))
    if connectionSocket != None:
        print('Connected to:', address[0], ':', address[1])
        # Start a new thread and return its identifier
        start_new_thread(multi_threaded_client, (connectionSocket,
address))
    else:
        Client.close()

ServerSideSocket.close()
```

Client : Kasir Restoran

login.py

```
### login.py ###

### LIBRARY YANG DIGUNAKAN ###
from tkinter import *
from tkinter import messagebox
import db.db
import main
import json
import socket
import hashlib
from PIL import Image, ImageTk # pip install pillow
import ssl

# Konfigurasi SSL
pemServer = 'server.pem'
keyClient = 'client.key'
pemClient = 'client.pem'

## Konfigurasi Socket
HOST = '192.168.100.8' #'localhost' # The server's hostname or IP address
PORT = 3456 # The port used by the server

### Class Aplikasi Utama ###
class LoginWindow:

    def __init__(self):
        self.win = Tk()
        self.canvas = Canvas(self.win, width=600, height=500, bg='white')
        self.canvas.pack(expand=YES, fill=BOTH)

        width = self.win.winfo_screenwidth()
        height = self.win.winfo_screenheight()
        x = int(width / 2 - 600 / 2)
        y = int(height / 2 - 500 / 2)
        str1 = "600x500+" + str(x) + "+" + str(y)
        self.win.geometry(str1)
        ico = Image.open('images/logo4.png')
        photo = ImageTk.PhotoImage(ico)
        self.win.wm_iconphoto(False, photo)

        self.win.resizable(width=False, height=False)

        self.win.title("WELCOME | MANAGEMENT LOGIN | ADMINISTRATOR")

        # fungsi untuk membuat frame di dalam window #
        def add_frame(self):
            self.frame = Frame(self.win, height=400, width=450)
            self.frame.place(x=80, y=50)

            x, y = 70, 20

            load = Image.open("images/logo1.png")
            resized_load = load.resize((round(load.size[0]*0.25),
```



```

round(load.size[1]*0.25)))

photo = ImageTk.PhotoImage(resized_load)
self.background = Label(self.frame, image=photo)
self.background.place(x=160,y=40)

self.label = Label(self.frame, text="User Login")
self.label.config(font=("Courier", 20, 'bold'))
self.label.place(x=140, y = y + 150)

self.emlabel = Label(self.frame, text="Enter Username")
self.emlabel.config(font=("Courier", 12, 'bold'))
self.emlabel.place(x=50, y= y + 230)

self.email = Entry(self.frame, font='Courier 12')
self.email.place(x=200, y= y + 230)

self.pslabel = Label(self.frame, text="Enter Password")
self.pslabel.config(font=("Courier", 12, 'bold'))
self.pslabel.place(x=50, y=y+260)

self.password = Entry(self.frame, show='*', font='Courier 12')
self.password.place(x=200, y=y+260)

self.button = Button(self.frame, text="Login", font='Courier 15 bold',
                      command=self.login, bg='#26a69a', fg='white')
self.button.place(x=190, y=y+290)

self.win.mainloop()

# Fungsi untuk login ketika tombol ditekan #
def login(self):
    email = self.email.get()
    password = self.password.get()

    if self.email.get() == "":
        messagebox.showinfo("Alert!", "Enter Username First")
    elif self.password.get() == "":
        messagebox.showinfo("Alert!", "Enter Password first")
    else:
        h = hashlib.sha224(bytes(password, 'utf-8'))
        send_obj = {
            'request': 'login',
            'username': email,
            'password': h.hexdigest()
        }
        jsonResult = json.dumps(send_obj)

        # Membuka log file #
        with open("request_response_history.txt", "a") as f:
            f.write(str(jsonResult) + "\n")
        b = bytes(jsonResult, 'utf-8')

        # Koneksi dengan server menggunakan socket SSL #
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((HOST, PORT))
            context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
            context.verify_mode = ssl.CERT_REQUIRED
            context.load_verify_locations(pemServer)
            context.load_cert_chain(certfile=pemClient,

```

```

keyfile=keyClient)
    if ssl.HAS_SNI:
        secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
    else:
        secure_sock = context.wrap_socket(s, server_side=False)

    cert = secure_sock.getpeercert()

    if not cert: raise Exception("ERROR")

    secure_sock.send(b)
    response = secure_sock.recv(4096)
    secure_sock.close()
    s.close()

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(response.decode("utf-8")) + "\n")
    py_obj = json.loads(response.decode("utf-8"))
    if py_obj['status'] == 'berhasil' :
        messagebox.showinfo("Message", "Login Successfully")
        self.win.destroy()
        x = main.restaurantApp()
    else:
        messagebox.showinfo("Alert!", "Wrong username/password")

```

main.py

```

### main.py ###

### LIBRARY YANG DIGUNAKAN ###
from tkcalendar import Calendar, DateEntry
import tkinter as tk
import tkinter.messagebox as mb
import tkinter.ttk as ttk
import socket
import json
import datetime
from PIL import Image, ImageTk
import ssl

## Konfigurasi Socket
HOST = '192.168.100.8'
PORT = 3456

# Konfigurasi SSL
pemServer = 'server.pem'
keyClient = 'client.key'
pemClient = 'client.pem'

### Class Aplikasi Utama ###
class restaurantApp(tk.Tk):
    def __init__(self):

        # Pembuatan Penampilan Layar Utama #

```

```

super().__init__()
self.title("DAN RESTAURANT | ORDER MANAGEMENT | ADMINISTRATOR")
width = self.winfo_screenwidth()
height = self.winfo_screenheight()
x = int(width / 2 - 800 / 2)
y = int(height / 2 - 650 / 2 - 50)
str1 = "800x650+" + str(x) + "+" + str(y)
self.geometry(str1)
ico = Image.open('images/logo4.png')
photo = ImageTk.PhotoImage(ico)
self.wm_iconphoto(False, photo)

self.lblTitle = tk.Label(self, text="Order Management System",
font=("Helvetica", 16), fg="black")
self.lblPaket = tk.Label(self, text="Paket:", font=("Helvetica", 10),
fg="black")
self.lblQuan = tk.Label(self, text="Quantity:", font=("Helvetica",
10), fg="black")
self.lblStatus = tk.Label(self, text="Order Status:",
font=("Helvetica", 10), fg="black")
self.lblDOO = tk.Label(self, text="Date of Order:", font=("Helvetica",
10), fg="black")
self.lblUser = tk.Label(self, text="User:", font=("Helvetica", 10),
fg="black")
self.lblNotes = tk.Label(self, text="Notes:", font=("Helvetica", 10),
fg="black")
self.lblSelect = tk.Label(self, text="Please select one record below
to update or delete", font=("Helvetica", 10), fg="black")
self.lblSearchDate = tk.Label(self, text="Please Enter Date of
Order:", font=("Helvetica", 10), fg="black")
self.lblSearch = tk.Label(self, text="Please Enter Order
No:", font=("Helvetica", 10), fg="black")

self.entPaket1 = tk.Entry(self)
self.entPaket2 = tk.Entry(self)
self.entPaket3 = tk.Entry(self)
self.entQuan1 = tk.Entry(self)
self.entQuan2 = tk.Entry(self)
self.entQuan3 = tk.Entry(self)
n3 = tk.StringVar()
self.entStatus = ttk.Combobox(self, textvariable = n3)
self.entStatus['values'] = ('paid',
                             'ready',
                             'selesai')
self.calDOO = DateEntry(self, width=12, background='darkblue',
foreground='white', borderwidth=2,
year=2021, locale='en_US', date_pattern='y-mm-dd')
self.entUser = tk.Entry(self)
self.entNotes = tk.Entry(self)
self.searchDOO = DateEntry(self, width=12, background='darkblue',
foreground='white', borderwidth=2,
year=2021, locale='en_US', date_pattern='y-mm-dd')
self.entSearch = tk.Entry(self)

self.btn_topup = tk.Button(self, text="Top-Up", font=("Helvetica",
11), bg='#26a69a', fg='white',
command =self.openNewWindow)

self.btn_update =
tk.Button(self, text="Update", font=("Helvetica", 11), bg='#26a69a',

```

```

fg='white',command=self.update_order_data)
    self.btn_delete = tk.Button(self, text="Delete", font=("Helvetica",
11), bg='#26a69a', fg='white',
                                command=self.delete_order_data)
    self.btn_clear = tk.Button(self, text="Clear", font=("Helvetica", 11),
bg='#26a69a', fg='white',
                                command=self.clear_form)
    self.btn_show_all = tk.Button(self, text="Show All",
font=("Helvetica", 11), bg='#26a69a', fg='white',
                                command=self.load_order_data)
    self.btn_search_date = tk.Button(self, text="Search",
font=("Helvetica", 11), bg='#26a69a', fg='white',
                                command=self.show_search_date_record)
    self.btn_search = tk.Button(self, text="Search", font=("Helvetica",
11), bg='#26a69a', fg='white',
                                command=self.show_search_record)
    self.btn_exit = tk.Button(self, text="Exit", font=("Helvetica", 16),
bg='#26a69a', fg='white',command=self.exit)

    columns = ("#1", "#2", "#3", "#4", "#5", "#6", "#7", "#8")
    self.tvorder= ttk.Treeview(self,show="headings",height="5",
columns=columns)
    self.tvorder.heading('#1', text='orderno', anchor='center')
    self.tvorder.column('#1', width=60, anchor='center', stretch=False)
    self.tvorder.heading('#2', text='Paket', anchor='center')
    self.tvorder.column('#2', width=10, anchor='center', stretch=True)
    self.tvorder.heading('#3', text='Quantity', anchor='center')
    self.tvorder.column('#3',width=10, anchor='center', stretch=True)
    self.tvorder.heading('#4', text='Status', anchor='center')
    self.tvorder.column('#4', width=10, anchor='center', stretch=True)
    self.tvorder.heading('#5', text='Date of Order', anchor='center')
    self.tvorder.column('#5', width=10, anchor='center', stretch=True)
    self.tvorder.heading('#6', text='Harga', anchor='center')
    self.tvorder.column('#6',width=10, anchor='center', stretch=True)
    self.tvorder.heading('#7', text='user', anchor='center')
    self.tvorder.column('#7',width=10, anchor='center', stretch=True)
    self.tvorder.heading('#8', text='notes', anchor='center')
    self.tvorder.column('#8',width=10, anchor='center', stretch=True)

    #Scroll bars are set up below considering placement position(x&y
,height and width of treeview widget
    vsb= ttk.Scrollbar(self,
orient=tk.VERTICAL,command=self.tvorder.yview)
    vsb.place(x=40 + 640 + 1, y=310, height=180 + 20)
    self.tvorder.configure(yscroll=vsb.set)
    hsb = ttk.Scrollbar(self, orient=tk.HORIZONTAL,
command=self.tvorder.xview)
    hsb.place(x=40 , y=310+200+1, width=620 + 20)
    self.tvorder.configure(xscroll=hsb.set)
    self.tvorder.bind("<<TreeviewSelect>>", self.show_selected_record)

    self.lblTitle.place(x=200, y=30, height=27, width=300)
    self.lblPaket.place(x=175, y=70, height=23, width=100)
    self.lblQuan.place(x=175, y=100, height=23, width=100)
    self.lblStatus.place(x=171, y=129, height=23, width=104)
    self.lblDOO.place(x=175, y=158, height=23, width=104)
    self.lblUser.place(x=175, y=187, height=23, width=100)
    self.lblNotes.place(x=163, y=217, height=23, width=128)
    self.lblSelect.place(x=150, y=280, height=23, width=400)
    self.lblSearchDate.place(x=147, y=535, height=23, width=160)

```

```

self.lblSearch.place(x=174, y=560, height=23, width=134)

self.entPaket1.place(x=277, y=72, height=21, width=60)
    self.entPaket2.place(x=339, y=72, height=21, width=60)
self.entPaket3.place(x=401, y=72, height=21, width=60)
self.entQuan1.place(x=277, y=100, height=21, width=60)
self.entQuan2.place(x=339, y=100, height=21, width=60)
self.entQuan3.place(x=401, y=100, height=21, width=60)
self.entStatus.place(x=277, y=129, height=21, width=186)
self.calDOO.place(x=277, y=158, height=21, width=186)
self.entUser.place(x=278, y=188, height=21, width=186)
self.entNotes.place(x=278, y=218, height=21, width=186)
self.searchDOO.place(x=310, y=535, height=21, width=186)
self.entSearch.place(x=310, y=560, height=21, width=186)

self.btn_topup.place(x=548, y=72, height=25, width=76)
self.btn_update.place(x=290, y=245, height=25, width=76)
self.btn_delete.place(x=370, y=245, height=25, width=76)
self.btn_clear.place(x=460, y=245, height=25, width=76)
self.btn_show_all.place(x=548, y=245, height=25, width=76)
self.btn_search_date.place(x=498, y=532, height=26, width=60)
self.btn_search.place(x=498, y=558, height=26, width=60)
self.btn_exit.place(x=320, y=610, height=31, width=60)
self.tvorder.place(x=40, y=310, height=200, width=640)

self.load_order_data()

self.refresh()

self.mainloop()

# Pembuatan Tampilan Layar Top Up #
def openNewWindow(self):
    self.newWindow = tk.Toplevel(self)
    self.newWindow.title("Top-Up Window")

    width = self.wininfo_screenwidth()
    height = self.wininfo_screenheight()
    x = int(width / 2 - 500 / 2)
    y = int(height / 2 - 300 / 2 - 50)
    str1 = "500x300+" + str(x) + "+" + str(y)
    self.newWindow.geometry(str1)
    ico = Image.open('images/logo4.png')
    photo = ImageTk.PhotoImage(ico)
    self.newWindow.wm_iconphoto(False, photo)

    self.lblTopUpTitle = tk.Label(self.newWindow, text="Top-Up User
Balance", font=("Helvetica", 16), fg="black")
    self.lblSearchUser = tk.Label(self.newWindow, text="Please Enter
User:", font=("Helvetica", 10), fg="black")
    self.lblTopUpUser = tk.Label(self.newWindow, text="User:",
font=("Helvetica", 10), fg="black")
    self.lblBalance = tk.Label(self.newWindow, text="Balance:",
font=("Helvetica", 10), fg="black")
    self.lblTopUp = tk.Label(self.newWindow, text="TopUp:",
font=("Helvetica", 10), fg="black")

    self.entSearchUser = tk.Entry(self.newWindow)
    self.entTopUpUser = tk.Entry(self.newWindow)
    self.entBalance = tk.Entry(self.newWindow)

```

```

self.entTopUp = tk.Entry(self.newWindow)

self.btn_searchUser = tk.Button(self.newWindow, text="Search",
font=("Helvetica", 11), bg='#26a69a', fg='white',
                                command=self.show_search_username)

self.btn_topup =
tk.Button(self.newWindow, text="TopUp", font=("Helvetica", 11), bg='#26a69a',
fg='white',
                                command=self.update_user_balance)

self.btn_clear_topup =
tk.Button(self.newWindow, text="Clear", font=("Helvetica", 11), bg='#26a69a',
fg='white',
                                command=self.clear_topup)

self.lblTopUpTitle.place(x=30, y=30, height=27, width=400)
self.lblSearchUser.place(x=50, y=70, height=23, width=200)
self.lblTopUpUser.place(x=100, y=100, height=23, width=100)
self.lblBalance.place(x=100, y=129, height=23, width=100)
self.lblTopUp.place(x=100, y=155, height=23, width=100)

self.entSearchUser.place(x=250, y=72, height=21, width=186)
self.entTopUpUser.place(x=250, y=100, height=21, width=186)
self.entBalance.place(x=250, y=129, height=21, width=186)
self.entTopUp.place(x=250, y=155, height=21, width=186)

self.btn_searchUser.place(x=100, y=201, height=31, width=60)
self.btn_topup.place(x=180, y=201, height=31, width=100)
self.btn_clear_topup.place(x=300, y=201, height=31, width=60)

# Fungsi Update User Balance #
def update_user_balance(self):
    User = self.entTopUpUser.get()
    TopUp = self.entTopUp.get()
    if User == "":
        mb.showinfo('Information', "Please Enter User Name",
parent=self.newWindow)
        self.entTopUpUser.focus_set()
        return
    if TopUp == "":
        mb.showinfo('Information', "Please Enter TopUp Amount",
parent=self.newWindow)
        self.entTopUp.focus_set()
        return
    send_obj = {
        'request': 'topup',
        'username': User,
        'topup': TopUp
    }
    jsonResult = json.dumps(send_obj)

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(jsonResult) + "\n")
    b = bytes(jsonResult, 'utf-8')

    # Koneksi dengan server menggunakan socket SSL #
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
        context.verify_mode = ssl.CERT_REQUIRED

```

```

        context.load_verify_locations(pemServer)
        context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
        if ssl.HAS_SNI:
            secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
        else:
            secure_sock = context.wrap_socket(s, server_side=False)

        cert = secure_sock.getpeercert()

        if not cert: raise Exception("ERROR")

        secure_sock.send(b)
        response = secure_sock.recv(4096)
        secure_sock.close()
        s.close()

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(response.decode("utf-8")) + "\n")
py_obj = json.loads(response.decode("utf-8"))
if py_obj['status'] == 'berhasil' :
    mb.showinfo("Info", "Selected User Balance Updated
Successfully", parent=self.newWindow)
    self.entSearchUser.delete(0, tk.END)
    self.entSearchUser.insert(0, User)
    self.show_search_username()
else:
    mb.showinfo('Information', "Balance update failed!!!",
parent=self.newWindow)

# Fungsi melakukan pencarian informasi user balance berdasarkan
username #
def show_search_username(self):
    username_entry = self.entSearchUser.get()
    if username_entry == "":
        mb.showinfo('Information', "Please Enter
Username", parent=self.newWindow)
        self.entSearchUser.focus_set()
        return
    self.entTopUpUser.delete(0, tk.END)
    self.entBalance.delete(0, tk.END)
    self.entTopUp.delete(0, tk.END)
    send_obj = {
        'request': 'search',
        'tipe': 'username',
        'username': username_entry
    }
    jsonResult = json.dumps(send_obj)

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(jsonResult) + "\n")
b = bytes(jsonResult, 'utf-8')

# Koneksi dengan server menggunakan socket SSL #
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    context.verify_mode = ssl.CERT_REQUIRED

```

```

        context.load_verify_locations(pemServer)
        context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
        if ssl.HAS_SNI:
            secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
        else:
            secure_sock = context.wrap_socket(s, server_side=False)

        cert = secure_sock.getpeercert()

        if not cert: raise Exception("ERROR")

        secure_sock.send(b)
        response = secure_sock.recv(4096)
        secure_sock.close()
        s.close()

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(response.decode("utf-8")) + "\n")
py_obj = json.loads(response.decode("utf-8"))
if py_obj['status'] == 'berhasil' :

    self.entTopUpUser.insert(0, username_entry)
    self.entBalance.insert(0, py_obj['isi'])
else:
    mb.showinfo('Information', "Username Not Found!!!",
parent=self.newWindow)

# Fungsi Untuk Command Clear pada Layar TopUp #
def clear_topup(self):
    self.entTopUpUser.delete(0, tk.END)
    self.entBalance.delete(0, tk.END)
    self.entSearchUser.delete(0, tk.END)
    self.entTopUp.delete(0, tk.END)

# Fungsi Untuk Command clear pada layar utama #
def clear_form(self):
    self.entPaket1.delete(0, tk.END)
    self.entPaket2.delete(0, tk.END)
    self.entPaket3.delete(0, tk.END)
    self.entQuan1.delete(0, tk.END)
    self.entQuan2.delete(0, tk.END)
    self.entQuan3.delete(0, tk.END)
    self.entUser.delete(0, tk.END)
    self.entNotes.delete(0, tk.END)
    self.entStatus.delete(0, tk.END)
    self.calDOO.delete(0, tk.END)

# Fungsi untuk button exit pada layar utama #
def exit(self):
    MsgBox = mb.askquestion('Exit Application', 'Are you sure you want to
exit the application', icon='warning')
    if MsgBox == 'yes':
        self.destroy()

# Fungsi untuk mengirimkan request delete suatu pesanan #
def delete_order_data(self):
    MsgBox = mb.askquestion('Delete Record', 'Are you sure! you want to
delete selected ordertable record', icon='warning')

```



```

if MsgBox == 'yes':
    send_obj = {
        'request': 'delete',
        'orderno': order_no
    }
    jsonResult = json.dumps(send_obj)

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(jsonResult) + "\n")
    b = bytes(jsonResult, 'utf-8')

    # Koneksi dengan server menggunakan socket SSL #
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
        context.verify_mode = ssl.CERT_REQUIRED
        context.load_verify_locations(pemServer)
        context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
        if ssl.HAS_SNI:
            secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
        else:
            secure_sock = context.wrap_socket(s, server_side=False)

        cert = secure_sock.getpeercert()

        if not cert: raise Exception("ERROR")

        secure_sock.send(b)
        response = secure_sock.recv(4096)
        secure_sock.close()
        s.close()

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(response.decode("utf-8")) + "\n")
    py_obj = json.loads(response.decode("utf-8"))
    if py_obj['status'] == 'berhasil' :
        mb.showinfo("Information", "Order Record Deleted Successfully")
        self.load_order_data()
        self.entPaket1.delete(0, tk.END)
        self.entPaket2.delete(0, tk.END)
        self.entPaket3.delete(0, tk.END)
        self.entQuan1.delete(0, tk.END)
        self.entQuan2.delete(0, tk.END)
        self.entQuan3.delete(0, tk.END)
        self.entUser .delete(0, tk.END)
        self.entNotes.delete(0, tk.END)
        self.entStatus.delete(0, tk.END)
        self.calDOO.delete(0, tk.END)
    else:
        messagebox.showinfo("Alert!", "Cannot Delete Record")

    # Fungsi untuk melakukan pencarian pesanan berdasarkan nomor order #
    def show_search_record(self):
        o_order_no = self.entSearch.get() # Retrieving entered orderno
        print(o_order_no)
        if o_order_no == "":
            mb.showinfo('Information', "Please Enter Order No")

```

```

        self.entSearch.focus_set()
        return
self.tvorder.delete(*self.tvorder.get_children())
send_obj = {
    'request': 'search',
    'tipe': 'orderno',
    'orderno': o_order_no
}
jsonResult = json.dumps(send_obj)

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(jsonResult) + "\n")
b = bytes(jsonResult, 'utf-8')

# Koneksi dengan server menggunakan socket SSL #
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    context.verify_mode = ssl.CERT_REQUIRED
    context.load_verify_locations(pemServer)
    context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
    if ssl.HAS_SNI:
        secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
    else:
        secure_sock = context.wrap_socket(s, server_side=False)

    cert = secure_sock.getpeercert()

    if not cert: raise Exception("ERROR")

    secure_sock.send(b)
    response = secure_sock.recv(4096)
    secure_sock.close()
    s.close()

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(response.decode("utf-8")) + "\n")
py_obj = json.loads(response.decode("utf-8"))
rows = py_obj['isi']

orderno = ""
Paket = ""
Quantity = ""
User = ""
Status = ""
Notes = ""
DOO = ""
Harga = ""
for row in reversed(rows):
    orderno = row[0]
    Paket = row[1]
    Quantity = row[2]
    Status = row[3]
    DOO = row[4]
    Harga = row[5]
    User = row[6]
    Notes = row[7]

```

```

        self.tvorder.insert("", 'end', text=orderno, values=(orderno,
Paket, Quantity, Status, DOO, Harga, User, Notes))

    # Fungsi untuk melakukan pencarian pesanan berdasarkan tanggal
    pemesanan #
    def show_search_date_record(self):
        date_of_order = self.searchDOO.get() # Retrieving entered DOO
        print(date_of_order)
        if date_of_order == "":
            mb.showinfo('Information', "Please Enter Date of Order")
            self.entSearchDate.focus_set()
            return
        self.tvorder.delete(*self.tvorder.get_children()) # clears the
treeview tvorder
        send_obj = {
            'request': 'search',
            'tipe': 'tanggal',
            'tanggal': date_of_order
        }
        jsonResult = json.dumps(send_obj)

        # Membuka log file #
        with open("request_response_history.txt", "a") as f:
            f.write(str(jsonResult) + "\n")
        b = bytes(jsonResult, 'utf-8')

        # Koneksi dengan server menggunakan socket SSL #
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((HOST, PORT))
            context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
            context.verify_mode = ssl.CERT_REQUIRED
            context.load_verify_locations(pemServer)
            context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
            if ssl.HAS_SNI:
                secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
            else:
                secure_sock = context.wrap_socket(s, server_side=False)

            cert = secure_sock.getpeercert()

            if not cert: raise Exception("ERROR")

            secure_sock.send(b)
            response = secure_sock.recv(4096)
            secure_sock.close()
            s.close()

        # Membuka log file #
        with open("request_response_history.txt", "a") as f:
            f.write(str(response.decode("utf-8")) + "\n")
        py_obj = json.loads(response.decode("utf-8"))
        rows = py_obj['isi']

        orderno = ""
        Paket = ""
        Quantity = ""
        User = ""
        Status = ""
        Notes = ""

```

```

DOO = ""
Harga = ""
for row in reversed(rows):
    ordeno = row[0]
    Paket = row[1]
    Quantity = row[2]
    Status = row[3]
    DOO = row[4]
    Harga = row[5]
    User = row[6]
    Notes = row[7]
    # print( User)
    self.tvorder.insert("", 'end', text=ordeno, values=(ordeno,
Paket, Quantity, Status, DOO, Harga, User, Notes))

# Fungsi untuk menampilkan atribut dari pesanan yang diselect #
def show_selected_record(self, event):
    self.clear_form()
    for selection in self.tvorder.selection():
        item = self.tvorder.item(selection)
    global order_no
    order_no,paket,quantity,Status,doo = item["values"][0:5]
    User,Notes = item["values"][6:8]
    paket = paket.split(';')
    if len(paket) == 1:
        self.entPaket1.insert(0, paket[0])
    elif len(paket) == 2:
        self.entPaket1.insert(0, paket[0])
        self.entPaket2.insert(0, paket[1])
    elif len(paket) == 3:
        self.entPaket1.insert(0, paket[0])
        self.entPaket2.insert(0, paket[1])
        self.entPaket3.insert(0, paket[2])
    quantity = str(quantity).split(';')
    if len(quantity) == 1:
        self.entQuan1.insert(0, quantity[0])
    elif len(quantity) == 2:
        self.entQuan1.insert(0, quantity[0])
        self.entQuan2.insert(0, quantity[1])
    elif len(quantity) == 3:
        self.entQuan1.insert(0, quantity[0])
        self.entQuan2.insert(0, quantity[1])
        self.entQuan3.insert(0, quantity[2])
    self.entUser.insert(0, User)
    self.entNotes.insert(0, Notes)
    self.entStatus .insert(0, Status)
    self.calDOO.insert(0, doo)
    return order_no

# Fungsi untuk mengirimkan request update atribut dari suatu pesanan #
def update_order_data(self):
    print("Updating")
    Paket1 = self.entPaket1.get()
    Paket2 = self.entPaket2.get()
    Paket3 = self.entPaket3.get()
    Quan1 = self.entQuan1.get()
    Quan2 = self.entQuan2.get()
    Quan3 = self.entQuan3.get()
    User = self.entUser.get()
    Notes = self.entNotes.get()

```

```

Status = self.entStatus.get()
DOO = self.calDOO.get()
print(order_no)
Harga = 0
if Paket1 == "A":
    Harga = Harga + int(Quan1)*39000 #harga 1
elif Paket1 == "B":
    Harga = Harga + int(Quan1)*46000 #harga 2
elif Paket1 == "C":
    Harga = Harga + int(Quan1)*42000 #harga 3
if Paket2 != "":
    if Paket2 == "A":
        Harga = Harga + int(Quan2)*39000 #harga 2
    elif Paket2 == "B":
        Harga = Harga + int(Quan2)*46000 #harga 3
    elif Paket2 == "C":
        Harga = Harga + int(Quan2)*42000 #harga 3
if Paket3 != "":
    if Paket3 == "A":
        Harga = Harga + int(Quan3)*39000 #harga 2
    elif Paket3 == "B":
        Harga = Harga + int(Quan3)*46000 #harga 3
    elif Paket3 == "C":
        Harga = Harga + int(Quan3)*42000 #harga 3
    if Paket3 != "":
        Paket = Paket1 + ';' + Paket2 + ';' + Paket3
        Quantity = Quan1 + ';' + Quan2 + ';' + Quan3
elif Paket2 != "":
    Paket = Paket1 + ';' + Paket2
    Quantity = Quan1 + ';' + Quan2
else:
    Paket = Paket1
    Quantity = Quan1
send_obj = {
    'request': 'update',
    'orderno': order_no,
    'paket': Paket,
    'quantity': Quantity,
    'status': Status,
    'doo': DOO,
    'username': User,
    'notes': Notes,
    'totalHarga': Harga
}
jsonResult = json.dumps(send_obj)

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(jsonResult) + "\n")
b = bytes(jsonResult, 'utf-8')

# Koneksi dengan server menggunakan socket SSL #
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    context.verify_mode = ssl.CERT_REQUIRED
    context.load_verify_locations(pemServer)
    context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
    if ssl.HAS_SNI:
        secure_sock = context.wrap_socket(s, server_side=False,

```

```

server_hostname=HOST)
    else:
        secure_sock = context.wrap_socket(s, server_side=False)

        cert = secure_sock.getpeercert()

        if not cert: raise Exception("ERROR")

        secure_sock.send(b)
        response = secure_sock.recv(4096)
        secure_sock.close()
        s.close()

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(response.decode("utf-8")) + "\n")
    py_obj = json.loads(response.decode("utf-8"))
    if py_obj['status'] == 'berhasil' :
        mb.showinfo("Info", "Selected Order Record Updated Successfully
")
        self.load_order_data()
    elif py_obj['status'] == 'saldo kurang' :
        mb.showinfo("Info", "Saldo User Kurang ")
        self.load_order_data()
    else:
        mb.showinfo('Information', "Data update failed!!!")

    # Fungsi untuk merequest daftar semua pesanan yang ada #
    def load_order_data(self):
        self.tvorder.delete(*self.tvorder.get_children()) # clears the
treeview tvorder
        send_obj = {
            'request': 'search',
            'tipe': 'all'
        }
        jsonResult = json.dumps(send_obj)

    # Membuka log file #
    with open("request_response_history.txt", "a") as f:
        f.write(str(jsonResult) + "\n")
    b = bytes(jsonResult, 'utf-8')

    # Koneksi dengan server menggunakan socket SSL #
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        #print('loading')
        context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
        context.verify_mode = ssl.CERT_REQUIRED
        context.load_verify_locations(pemServer)
        context.load_cert_chain(certfile=pemClient, keyfile=keyClient)
        if ssl.HAS_SNI:
            secure_sock = context.wrap_socket(s, server_side=False,
server_hostname=HOST)
        else:
            secure_sock = context.wrap_socket(s, server_side=False)

        cert = secure_sock.getpeercert()

        if not cert: raise Exception("ERROR")

```

```

        secure_sock.send(b)
        response = secure_sock.recv(4096)
        secure_sock.close()
        s.close()

# Membuka log file #
with open("request_response_history.txt", "a") as f:
    f.write(str(response.decode("utf-8")) + "\n")
py_obj = json.loads(response.decode("utf-8"))

rows = py_obj['isi']

orderno = ""
Paket = ""
Quantity = ""
Notes = ""
Status = ""
User = ""
DOO = ""
Harga = ""
for row in reversed(rows):
    orderno = row[0]
    Paket = row[1]
    Quantity = row[2]
    User = row[6]
    Notes = row[7]
    Status = row[3]
    DOO = row[4]
    Harga = row[5]
    self.tvorder.insert("", 'end', text=orderno, values=(orderno,
Paket, Quantity, Status, DOO, Harga, User, Notes))
    #print('load complete')

# Fungsi untuk memanggil load_order_data setiap 10 detik agar data
pesanan selalu uptodate #
def refresh(self):
    print('refresh')
    try:
        self.load_order_data()
        #print('lalala')
    finally:
        self.after(10000, self.refresh) # run itself again after 1000 ms

### MAIN PROGRAM ###
if __name__ == "__main__":
    print('now running')
    app = restaurantApp()
    app.mainloop()

```

restaurant_client.py

```

### restaurant_client.py ###

### LIBRARY YANG DIGUNAKAN ###
from tkinter import *

```

```

import login
from PIL import Image, ImageTk # pip install pillow

### Class Aplikasi Utama ###
class WelcomeWindow:

    def __init__(self):
        self.win = Tk()

        self.canvas = Canvas(self.win, width=600, height=400, bg='white')
        self.canvas.pack(expand=YES, fill=BOTH)

        width = self.win.winfo_screenwidth()
        height = self.win.winfo_screenheight()
        x = int(width / 2 - 600 / 2)
        y = int(height / 2 - 400 / 2)
        str1 = "600x400+" + str(x) + "+" + str(y)
        self.win.geometry(str1)
        ico = Image.open('images/logo4.png')
        photo = ImageTk.PhotoImage(ico)
        self.win.wm_iconphoto(False, photo)

        self.win.resizable(width=False, height=False)

        self.win.title("WELCOME | DAN RESTAURANT | ADMINISTRATOR")

        # fungsi untuk membuat frame di dalam window #
        def add_frame(self):
            self.frame = Frame(self.win, height=300, width=450)
            self.frame.place(x=80, y=50)

            x, y = 70, 20

            load = Image.open("images/logo1.png")
            resized_load = load.resize((round(load.size[0]*0.25),
            round(load.size[1]*0.25)))

            photo = ImageTk.PhotoImage(resized_load)
            self.background = Label(self.frame, image=photo)
            self.background.place(x=160, y=40)

            self.labeltitle = Label(self.frame, text="Welcome to Dan Restaurant")
            self.labeltitle.config(font=("Courier", 16, 'bold'))
            self.labeltitle.place(x=50, y=y+150)

            self.button = Button(self.frame, text="Continue", font=('helvetica',
            20, 'underline italic'), bg='#26a69a', fg='white',
            command=self.login)
            self.button.place(x=x+90, y=y+200)

            self.win.mainloop()

        # Fungsi untuk membuka window baru ketika tombol ditekan #
        def login(self):
            self.win.destroy()

            log = login.LoginWindow()
            log.add_frame()

```



```
### MAIN PROGRAM ###
if __name__ == "__main__":
    x = WelcomeWindow()
    x.add_frame()
```

Client : Pelanggan

Constants.java

```
package com.example.cobapplj;

public class Constants {
    // Contains all constants used in this project

    public static final String server_addr = "192.168.100.8"; // server
address
    public static final int server_port = 3456; // server port
}
```

Functions.java

```
package com.example.cobapplj;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.util.Log;

import androidx.appcompat.app.AlertDialog;
import androidx.core.app.NotificationCompat;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;

public class Functions {
    // Contains reuseable functions accross activity

    public static void CreateAlertDialogBox(Context context, String title,
String message){
```

```

// Function to create alert dialog box

// Create builder
AlertDialog.Builder builder = new AlertDialog.Builder(context);
builder.setTitle(title);
builder.setMessage(message);
builder.setPositiveButton("Ok", new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
    }
});

// Build
AlertDialog dialog = builder.create();
dialog.show();
}

public static void CreateNotification(Context context, int id, String
title, String message){
    // Function to create notification

    Log.d("Notification",message); // debug

    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.O) {
        // Create notification channel
        NotificationChannel channel = new NotificationChannel("Channel
1","Channel 1",NotificationManager.IMPORTANCE_LOW);

        NotificationManager notificationManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.createNotificationChannel(channel);

        // Create builder
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(context,"Channel 1");

        Intent notificationIntent = new Intent(context,
HomeActivity.class);
        PendingIntent pendingIntent =
PendingIntent.getActivity(context,id,notificationIntent,PendingIntent.FLAG_U
PDATE_CURRENT);

        builder.setContentIntent(pendingIntent);
        builder.setSmallIcon(R.drawable.logo1);
        builder.setContentTitle(title);
        builder.setContentText(message);

        // Build
        notificationManager.notify(id,builder.build());
    }
}

public static String HashPassword(String password,String method){
    // Function to hash string according to method
    String hashtext;

    try {

```

```

        // getInstance() method is called with algorithm SHA-224
        MessageDigest md = MessageDigest.getInstance(method);

        // digest() method is called
        // to calculate message digest of the input string
        // returned as array of byte
        byte[] messageDigest = md.digest(password.getBytes());

        // Convert byte array into signum representation
        BigInteger no = new BigInteger(1, messageDigest);

        // Convert message digest into hex value
        hashtext = no.toString(16);

        // Add preceding 0s to make it 32 bit
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
    }
    // For specifying wrong message digest algorithms
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }

    return hashtext;
}

public static void WriteTextFile(Context context, String filename,
String content){
    // Function to write string into textfile
    try {
        // Check if file exist
        File f = new File(context.getFilesDir() + "/" + filename);
        if (!f.exists()){
            f.createNewFile();
        }

        // Prepare to write file
        FileOutputStream fout = new
FileOutputStream(context.getFilesDir() + "/" + filename);
        fout.write(content.getBytes());
        fout.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static String ReadTextFile(Context context, String filename){
    // Function to read textfile and returns the content
    String temp = "";
    try {
        // Check if file exist
        File f = new File(context.getFilesDir() + "/" + filename);
        if (f.exists()){
            // Read file
            FileInputStream fin = new
FileInputStream(context.getFilesDir() + "/" + filename);
            int c;

            while( (c = fin.read()) != -1){

```

```

        temp = temp + (char) c;
    }

    fin.close();
}
} catch (Exception e) {
    e.printStackTrace();
}

return temp;
}

public static void WriteHistoryFile(Context context, List<History>
history_list){
    // Function to write history (from list) into file
    String content = "";
    for (History history : history_list){
        String line = history.FormatToHistoryLine();
        content = content + line;
    }
    WriteTextFile(context,"log.txt",content);
}

public static List<History> ReadHistoryFile(Context context){
    // Function to read history file and returns the list of history
    List<History> history_list = new ArrayList<History>();

    // Read file
    String content = ReadTextFile(context,"log.txt");

    // File exist and successfully read
    if (content.length()>0) {
        // Take each line
        String[] lines = content.split("\n");
        for (String line : lines){
            // Parse the line
            String[] history_content = line.split("\\|");
            String orderno = history_content[0];
            String timestamp = history_content[1];
            String paket = history_content[2];
            String quantity = history_content[3];
            String harga = history_content[4];
            String status = history_content[5];

            String notes;
            try {
                notes = history_content[6];
            } catch (ArrayIndexOutOfBoundsException e){
                notes = "";
            }

            // Input into list
            History history_instance = new
History(orderno,timestamp,paket,quantity,harga,notes,status);
            history_list.add(history_instance);
        }
    }

    return history_list;
}

```

```

        public static List<String> CompareHistoryList(List<History>
before_list, List<History> after_list){
            List<String> result = new ArrayList<String>();

            for (History before_hist : before_list){
                String orderno1 = before_hist.getOrderno();
                History cohistry = null;
                for (History after_hist : after_list){
                    if (after_hist.getOrderno().equals(orderno1)){
                        cohistry = after_hist;
                        break;
                    }

                    if (cohistry!=null){
                        if
(!before_hist.getStatus().equals(cohistry.getStatus())){
                            result.add("Order no " + before_hist.getOrderno() + " is "
+ cohistry.getStatus());
                        }
                    }
                }

                return result;
            }
        }
    }
}

```

History.java

```

package com.example.cobapplj;

public class History {

    String orderno;
    String timestamp;
    String paket;
    String quantity;
    String harga;
    String notes;
    String status;

    public History(){
        this.orderno = "";
        this.timestamp = "";
        this.paket = "";
        this.quantity = "";
        this.harga = "";
        this.notes = "";
        this.status = "";
    }

    public History (String orderno,String timestamp,String paket,String
quantity,String harga,String notes,String status){
        this.orderno = orderno;
        this.timestamp = timestamp;
    }
}

```

```
this.paket = paket;
this.quantity = quantity;
this.harga = harga;
this.notes = notes;
this.status = status;
}

public String getOrderno() {
return orderno;
}

public String getTimestamp(){
return this.timestamp;
}

public String getPaket(){
return this.paket;
}

public String getQuantity(){
return this.quantity;
}

public String getHarga() {
return this.harga;
}

public String getNotes() {
return notes;
}

public String getStatus() {
return status;
}

public void setOrderno(String orderno) {
this.orderno = orderno;
}

public void setTimestamp(String timestamp){
this.timestamp = timestamp;
}

public void setPaket(String paket) {
this.paket = paket;
}

public void setQuantity(String quantity) {
this.quantity = quantity;
}

public void setHarga(String harga) {
this.harga = harga;
}

public void setNotes(String notes) {
this.notes = notes;
}

public void setStatus(String status) {
```

```

        this.status = status;
    }

    public String getPesanan(){
        String pesanan = "";

        String[] list_paket = paket.split(";");
        String[] list_quantity = quantity.split(";");

        if (list_paket.length==1){
            pesanan = paket + ":" + quantity;
        } else if (list_paket.length==2){
            pesanan = list_paket[0] + ":" + list_quantity[0] + ", " +
list_paket[1] + ":" + list_quantity[1];
        } else if (list_paket.length==3){
            pesanan = list_paket[0] + ":" + list_quantity[0] + ", " +
list_paket[1] + ":" + list_quantity[1] + ", " + list_paket[2] + ":" +
list_quantity[2];
        }

        return pesanan;
    }

    public String FormatToHistoryLine(){
        return this.orderno + "|" + this.timestamp + "|" + this.paket + "|" +
this.quantity + "|" + this.harga + "|" + this.status + "|" + this.notes +
"|\n";
    }
}

```

HistoryListAdapter.java

```

package com.example.cobapplj;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import java.util.List;

public class HistoryListAdapter extends ArrayAdapter<History> {

    private List<History> listHistory;
    private Context context;
    private int layout;

    public HistoryListAdapter(Context context, int layout, List<History>
listHistory){
        super(context, layout, listHistory);
        this.listHistory = listHistory;
    }
}

```

```

        this.context = context;
        this.layout = layout;
    }

    // https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView
    // https://agung-setiawan.com/android-membuat-custom-listview/
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        HistoryHolder holder;

        View v = convertView;

        if (v == null) {
            v = LayoutInflater.from(context).inflate(layout, parent, false);
        }

        holder = new HistoryHolder();
        holder.history_pesanan = v.findViewById(R.id.history_pesanan);
        holder.history_time = v.findViewById(R.id.history_time);
        holder.history_harga = v.findViewById(R.id.history_harga);
        holder.history_notes = v.findViewById(R.id.history_notes);
        holder.history_status = v.findViewById(R.id.history_status);
        holder.history_orderno = v.findViewById(R.id.history_orderno);

        // Ambil data scheduler
        History history = listHistory.get(position);

        //          // Ubah timestamp menjadi tanggal lalu tampilkan
        //          Date date = new Date(history.getTimestamp()*1000L);
        //          DateFormat df = new SimpleDateFormat("yyyy/MM/dd HH:mm");
        //          String string_date = df.format(date);

        holder.history_time.setText(history.getTimestamp());
        holder.history_pesanan.setText("Pesanan : "+history.getPesanan());
        holder.history_harga.setText("Harga : Rp "+history.getHarga());
        holder.history_notes.setText("Notes : "+history.getNotes());
        holder.history_status.setText("Status : "+history.getStatus());
        holder.history_orderno.setText("Orderno : "+history.getOrderno());

        return v;
    }

    static class HistoryHolder{
        TextView history_orderno;
        TextView history_time;
        TextView history_pesanan;
        TextView history_harga;
        TextView history_notes;
        TextView history_status;
    };
}

```

HomeActivity.java


```

package com.example.cobapplj;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.SocketTimeoutException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class HomeActivity extends AppCompatActivity {

    TextView saldo_text;

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

        // get username and saldo from sharedpref
        final SharedPreferences user_pref = getSharedPreferences("USER_PREF",
MODE_PRIVATE);

        String username = user_pref.getString("username", null);
        Integer saldo = user_pref.getInt("saldo", -1);

        // Try to update saldo from server
        JSONObject send_json = new JSONObject();
        try {
            send_json.put("request", "search");
            send_json.put("tipe", "username");
            send_json.put("username", username);

            String json_string = send_json.toString();

            new NetworkTask().execute(json_string);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    //Pesanan Button
    Button btn_pesanan=(Button)findViewById(R.id.but_pesanan);
    btn_pesanan.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick (View arg0) {
            Intent intent = new Intent(HomeActivity.this,
PesananActivity.class);
            startActivity(intent);
        }
    });

    //Order History Button
    Button btn_history=(Button)findViewById(R.id.but_history);
    btn_history.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick (View arg0) {
            Intent intent = new Intent(HomeActivity.this,
LihatHistory.class);
            startActivity(intent);
        }
    });

    //Logout Button
    Button btn_logout=(Button)findViewById(R.id.but_logout);
    btn_logout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick (View view) {
            SharedPreferences.Editor user_pref_editor =
user_pref.edit();
            user_pref_editor.putString("username","");
            user_pref_editor.putInt("saldo",-1);
            user_pref_editor.apply();

            MainActivity.CancelRefresh(getApplicationContext());

            try {
                File f = new File(getFilesDir() + "/log.txt");
                f.delete();
            } catch (Exception e){
                e.printStackTrace();
            }

            Intent intent = new Intent(HomeActivity.this,
LoginActivity.class);
            startActivity(intent);
            finish();
        }
    });

    TextView hi_text = (TextView) findViewById(R.id.hi_text);
    hi_text.setText("Hi "+username);

    // Update saldo
    saldo_text = (TextView) findViewById(R.id.saldo_text);
    if (saldo!=-1){
        saldo_text.setText("Saldo : Rp " + String.format("%,d",saldo));
    } else {
        saldo_text.setText("Saldo : Rp -");
    }

```

```

    }
    }

    private class NetworkTask extends AsyncTask<String, byte[], String> {

        SSLSocket nsocket; //Network Socket
        InputStream nis; //Network Input Stream
        OutputStream nos; //Network Output Stream

        @Override
        protected String doInBackground(String... params) { //This runs on a
different thread
            try {
                Log.i("AsyncTask", "doInBackground: Creating socket");
                SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
                sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
                    public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                    public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                    public X509Certificate[] getAcceptedIssuers() {
                        return new X509Certificate[0];
                    }
                }}, new SecureRandom());
                SSLSocketFactory socketFactory =
sslContext.getSocketFactory();
                nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
                nsocket.setSoTimeout(5000);
            } catch (SocketTimeoutException e) {
                return "timeout";
            } catch (Exception e) {
                e.printStackTrace();
            }
            String result = "";
            try {
                if (nsocket.isConnected()) {
                    nis = nsocket.getInputStream();
                    nos = nsocket.getOutputStream();
                    nos.write(params[0].getBytes());
                    byte[] buffer = new byte[4096];
                    int read = nis.read(buffer, 0, 4096); //This is blocking
                    while (read != -1) {
                        byte[] tempdata = new byte[read];
                        System.arraycopy(buffer, 0, tempdata, 0, read);
                        result = new String(tempdata);
                        read = -1;
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
                result = "";
            }
            return result;
        }

        @Override
        protected void onPostExecute(String s) {
            Log.d("AsyncTask", s);
        }
    }
}

```

```

        if (s.equals("timeout")){
            AlertDialog.Builder builder = new
AlertDialog.Builder(HomeActivity.this);
            builder.setTitle("Error");
            builder.setMessage("Please check your connection");
            builder.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog dialog = builder.create();
            dialog.show();
        } else {
            if (s.length()==0){
                return;
            }
            if (s.substring(s.length()-1)!=""){
                s = s + "}";
            }
            try {
                nis.close();
                nos.close();
                nsocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
            try {
                JSONObject json_response = new JSONObject(s);
                if (json_response.get("response").equals("search") &&
json_response.get("status").equals("berhasil")) {
                    String saldo_str =
json_response.get("isi").toString();
                    Integer saldo = Integer.parseInt(saldo_str);

                    SharedPreferences.Editor user_pref_editor =
getSharedPreferences("USER_PREF", MODE_PRIVATE).edit();
                    user_pref_editor.putInt("saldo",saldo);
                    user_pref_editor.apply();

                    saldo_text.setText("Saldo : Rp " + saldo_str);
                } else {

                    Functions.CreateAlertDialogBox(HomeActivity.this,"Error","Please check
your credentials");
                }

            } catch (JSONException e) {
                e.printStackTrace();
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
}
}

```

KonfirmasiPembayaran.java

```
package com.example.cobapplj;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.SocketTimeoutException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class KonfirmasiPembayaran extends AppCompatActivity {

    private String pesanan;
    private String quantity;
    private String notes;

    private String username;
    private String password;

    private int total;
    private int saldo;
    private int sisaSaldo;

    ProgressBar progressBar;

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

        //Print total pembayaran
        Intent rIntent = getIntent();
```

```

        total = rIntent.getIntExtra("totalPembayaran", 0);
        pesanan = rIntent.getStringExtra("pesanan");
        quantity = rIntent.getStringExtra("quantity");
        notes = rIntent.getStringExtra("notes");

        // Ambil username dari shared pref
        SharedPreferences user_pref =
getSharedPreferences("USER_PREF",MODE_PRIVATE);
        username = user_pref.getString("username","");
        saldo = user_pref.getInt("saldo",-1);

        // Update total harga text
        TextView txtTotal=findViewById(R.id.totalTxt);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);

        // Update saldo text
        TextView txtSaldo=findViewById(R.id.saldoTxt);
        String saldoStr=String.format("%,d", saldo);
        txtSaldo.setText("Rp "+saldoStr);

        // Update sisa saldo text
        sisaSaldo=saldo-total;
        TextView txtSisa=findViewById(R.id.sisaSaldoTxt);
        String sisaSaldoStr=String.format("%,d", sisaSaldo);
        txtSisa.setText("Rp "+sisaSaldoStr);

        progressBar = (ProgressBar) findViewById(R.id.progress_bar);
    }

    public void onKonfirmasi(View view){
        //Cek Password dan saldo. kalau benar, pindah ke activity selanjutnya
        if (sisaSaldo<0){

            Functions.CreateAlertDialogBox(KonfirmasiPembayaran.this,"Gagal","Maaf
saldo anda tidak cukup");
        } else {
            EditText editTextPassword = (EditText)
findViewById(R.id.editTextPassword);
            password = editTextPassword.getText().toString();

            if (password.length()==0){

                Functions.CreateAlertDialogBox(KonfirmasiPembayaran.this,"Error","Plea
se insert password");
            } else {
                String hashtext = Functions.HashPassword(password, "SHA-
224");

                JSONObject send_json = new JSONObject();
                try {
                    send_json.put("request", "pesan");
                    send_json.put("username", username);
                    send_json.put("password", hashtext);
                    send_json.put("paket", pesanan);
                    send_json.put("quantity", quantity);
                    send_json.put("notes", notes);
                    send_json.put("totalHarga", total);

                    String json_string = send_json.toString();

```

```

        progressBar.setVisibility(View.VISIBLE);

        new NetworkTask().execute(json_string);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

}

private class NetworkTask extends AsyncTask<String, byte[], String> {

    SSLSocket nsocket; //Network Socket
    InputStream nis; //Network Input Stream
    OutputStream nos; //Network Output Stream

    @Override
    protected String doInBackground(String... params) { //This runs on a
different thread
        try {
            Log.i("AsyncTask", "doInBackground: Creating socket");
            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
                public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public X509Certificate[] getAcceptedIssuers() {
                    return new X509Certificate[0];
                }
            }, new SecureRandom());
            SSLSocketFactory socketFactory =
sslContext.getSocketFactory();
            nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
        } catch (SocketTimeoutException e) {
            return "timeout";
        } catch (Exception e) {
            e.printStackTrace();
        }
        String result = "";
        try {
            if (nsocket.isConnected()) {
                nis = nsocket.getInputStream();
                nos = nsocket.getOutputStream();
                nos.write(params[0].getBytes());
                byte[] buffer = new byte[4096];
                int read = nis.read(buffer, 0, 4096); //This is blocking
                while (read != -1) {
                    byte[] tempdata = new byte[read];
                    System.arraycopy(buffer, 0, tempdata, 0, read);
                    result = new String(tempdata);
                    read = -1;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            result = "";
        }
    }
}

```

```

        }
        return result;
    }

    public void CancelRefresh(){

    }

    @Override
    protected void onPostExecute(String s) {
        Log.d("AsyncTask",s);
        if (s.equals("timeout")){

            Functions.CreateAlertDialogBox(KonfirmasiPembayaran.this,"Error","Please check your connection");
        } else {
            if (s.substring(s.length()-1)!="}") {
                s = s + "}";
            }
            try {
                nis.close();
                nos.close();
                nsocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
            try {
                JSONObject json_response = new JSONObject(s);
                if (json_response.get("response").equals("pesan") &&
                json_response.get("status").equals("berhasil")) {
                    String order_no =
                json_response.get("orderno").toString();
                    AlertDialog.Builder builder = new
                AlertDialog.Builder(KonfirmasiPembayaran.this);
                    builder.setTitle("Berhasil");
                    builder.setMessage("Pesananmu sudah tercatat dengan
                order no "+order_no);
                    builder.setPositiveButton("Ok", new
                DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
                int which) {

                        Intent intent = new
                Intent(KonfirmasiPembayaran.this,HomeActivity.class);
                        startActivity(intent);
                        finish();
                    }
                });

                    AlertDialog dialog = builder.create();
                    dialog.show();
                } else {

                    Functions.CreateAlertDialogBox(KonfirmasiPembayaran.this,"Error","Sepertinya password salah");
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }

```



```

        Functions.CreateAlertDialogBox(KonfirmasiPembayaran.this, "Maaf", "Seper
tinya terjadi kesalahan. Silakan coba lagi.");
    }
    }
    progressBar.setVisibility(View.GONE);
}
}
}

```

LauncherActivity.java

```

package com.example.cobapplj;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;

import androidx.appcompat.app.AppCompatActivity;

public class LauncherActivity extends AppCompatActivity {

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

        // Cek apakah user sudah login dari shared pref
        final SharedPreferences user_pref = getSharedPreferences("USER_PREF",
MODE_PRIVATE);

        String username_pref = user_pref.getString("username", "");

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if (username_pref.equals("")) { // belum login, alihkan ke
LoginActivity
                    Intent intent = new Intent(LauncherActivity.this,
LoginActivity.class);
                    startActivity(intent);
                    finish();
                } else { // sudah login, alihkan ke HomeActivity
MainActivity.StartRefresh(getApplicationContext());

                    int saldo = user_pref.getInt("saldo", 0);
                    Intent intent = new Intent(LauncherActivity.this,
HomeActivity.class);
                    intent.putExtra("saldo", saldo);
                    startActivity(intent);
                    finish();
                }
            }
        }, 2000);
    }
}

```

```
}
```

LihatHistory.java

```
package com.example.cobapplj;

import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.SocketTimeoutException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class LihatHistory extends AppCompatActivity {

    ProgressBar progressBar;
    List<History> listHistory = new ArrayList<History>();
    HistoryListAdapter adapter;
    TextView info_text;

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

        // Ambil username dari sharedpref
        SharedPreferences user_pref =
        getSharedPreferences("USER_PREF", MODE_PRIVATE);
        String username = user_pref.getString("username", "");

        //Back Button
        ImageView btn_kembali=(ImageView) findViewById(R.id.back_icon);
        btn_kembali.setOnClickListener(new View.OnClickListener() {
```

```

        @Override
        public void onClick (View view) {
            finish();
        }
    });

    info_text = (TextView) findViewById(R.id.info_text);
    progressBar = (ProgressBar) findViewById(R.id.progress_bar);

    // Buka history dari file history
    listHistory = Functions.ReadHistoryFile(LihatHistory.this);
    if (listHistory.size() != 0) {
        progressBar.setVisibility(View.GONE);
        info_text.setVisibility(View.GONE);
    } else {
        progressBar.setVisibility(View.INVISIBLE);
        info_text.setVisibility(View.VISIBLE);
    }

    // Siapkan listview
    ListView listView = (ListView) findViewById(R.id.listView);
    adapter = new HistoryListAdapter(LihatHistory.this,
R.layout.list_history, listHistory);
    listView.setAdapter(adapter);

    if (listHistory.size() != 0) {
        progressBar.setVisibility(View.GONE);
        info_text.setVisibility(View.GONE);
    } else {
        progressBar.setVisibility(View.INVISIBLE);
        info_text.setVisibility(View.VISIBLE);
    }

    // Request history ke server
    JSONObject send_json = new JSONObject();
    try {
        send_json.put("request", "showHistory");
        send_json.put("username", username);

        String json_string = send_json.toString();

        progressBar.setVisibility(View.VISIBLE);

        new NetworkTask().execute(json_string);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

private class NetworkTask extends AsyncTask<String, byte[], String> {

    SSLSocket nsocket; //Network Socket
    InputStream nis; //Network Input Stream
    OutputStream nos; //Network Output Stream

    @Override
    protected String doInBackground(String... params) { //This runs on a
different thread
        try {
            Log.i("AsyncTask", "doInBackground: Creating socket");

```

```

        SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
        sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
            public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
            public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
            public X509Certificate[] getAcceptedIssuers() {
                return new X509Certificate[0];
            }
        }}, new SecureRandom());
        SSLSocketFactory socketFactory =
sslContext.getSocketFactory();
        nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
        nsocket.setSoTimeout(5000);
        } catch (SocketTimeoutException e){
            return "timeout";
        } catch (Exception e){
            e.printStackTrace();
        }
        String result = "";
        try {
            if (nsocket.isConnected()) {
                nis = nsocket.getInputStream();
                nos = nsocket.getOutputStream();
                nos.write(params[0].getBytes());
                byte[] buffer = new byte[4096];
                int read = nis.read(buffer, 0, 4096); //This is blocking
                while (read != -1) {
                    byte[] tempdata = new byte[read];
                    System.arraycopy(buffer, 0, tempdata, 0, read);
                    result = new String(tempdata);
                    read = -1;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            result = "";
        }
        return result;
    }

    @Override
    protected void onPostExecute(String s) {
        Log.d("AsyncTask", s);
        if (s.equals("timeout")){

            Functions.CreateAlertDialogBox(LihatHistory.this, "Error", "Please check
your connection");
        } else {
            if (s.substring(s.length()-1)!=""){
                s = s + "}";
            }

            try {
                nis.close();
                nos.close();
                nsocket.close();
            } catch (IOException e) {

```

```

        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
        JSONObject json_response = new JSONObject(s);
        if (json_response.get("response").equals("showHistory")) {
            listHistory.clear();
            adapter.notifyDataSetChanged();

            String isi = json_response.get("isi").toString();
            Integer jumlah =
Integer.parseInt(json_response.get("jumlah").toString());
            Log.d("Asynctask", isi);
            if (jumlah<=0){
                info_text.setText("Tidak ada history");
                info_text.setVisibility(View.VISIBLE);
            } else {
                info_text.setVisibility(View.GONE);
                isi = isi.substring(1, isi.length() - 1);
                String[] arrayHistory = isi.split("]");
                for (String history : arrayHistory) {
                    String cleaned_history = history;
                    int left_index = history.indexOf("[");
                    if (left_index != -1) {
                        cleaned_history =
history.substring(left_index + 1);
                    }
                    Log.d("Asynctask", cleaned_history);
                    String[] history_content =
cleaned_history.split(",");
                    String orderno = history_content[0];
                    String paket =
history_content[1].substring(1, history_content[1].length() - 1);
                    String quantity =
history_content[2].substring(1, history_content[2].length() - 1);
                    String status =
history_content[3].substring(1, history_content[3].length() - 1);
                    String timestamp =
history_content[4].substring(1, history_content[4].length() - 1);
                    String harga = history_content[5];
                    String notes =
history_content[7].substring(1, history_content[7].length() - 1);

                    History history_instance = new
History(orderno,timestamp, paket, quantity, harga, notes, status);
                    listHistory.add(history_instance);
                    adapter.notifyDataSetChanged();
                }

                if (listHistory.size() > 0) {
Functions.WriteHistoryFile(LihatHistory.this, listHistory);
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();

            Functions.CreateAlertDialogBox(LihatHistory.this, "Maaf", "Sepertinya

```

```

        terjadi kesalahan. Silakan coba lagi.");
    }
    }
    progressBar.setVisibility(View.GONE);
}
}
}

```

LoginActivity.java

```

package com.example.cobapplj;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.SocketTimeoutException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class LoginActivity extends AppCompatActivity {

    ProgressBar progressBar;
    String username;
    String password;

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

        final SharedPreferences login_pref =
        getSharedPreferences("LOGIN_PREF", MODE_PRIVATE);

        final EditText username_field =

```

```

(EditText)findViewById(R.id.username_field);
    final EditText password_field =
(EditText)findViewById(R.id.password_field);
    Button login_button = (Button)findViewById(R.id.login_button);

    progressBar = (ProgressBar)findViewById(R.id.progress_bar);

    login_button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            // Ambil username dan password dari field nya masing
masing
            username = username_field.getText().toString();
            password = password_field.getText().toString();

            if (username.length()==0 || password.length()==0){ //
validasi panjang username password
                progressBar.setVisibility(View.GONE);

                Functions.CreateAlertDialogBox(LoginActivity.this,"Invalid","Please
fill every field");
            } else {
                // Hash password
                String hashtext = Functions.HashPassword(password,"SHA-
224");

                // Kirim ke server
                JSONObject send_json = new JSONObject();
                try {
                    send_json.put("request","login");
                    send_json.put("username",username);
                    send_json.put("password",hashtext);

                    String json_string = send_json.toString();

                    progressBar.setVisibility(View.VISIBLE);

                    new NetworkTask().execute(json_string);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
    });

    TextView login_register_text =
(TextView)findViewById(R.id.login_register_text);
    login_register_text.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(LoginActivity.this,
RegisterActivity.class);
            startActivity(intent);
            finish();
        }
    });
}

```

```

private class NetworkTask extends AsyncTask<String, byte[], String> {

    SSLSocket nsocket; //Network Socket
    InputStream nis; //Network Input Stream
    OutputStream nos; //Network Output Stream

    @Override
    protected String doInBackground(String... params) { //This runs on a
different thread
        try {
            Log.i("AsyncTask", "doInBackground: Creating socket");
            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
                public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public X509Certificate[] getAcceptedIssuers() {
                    return new X509Certificate[0];
                }
            }}, new SecureRandom());
            SSLSocketFactory socketFactory =
sslContext.getSocketFactory();
            nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
            nsocket.setSoTimeout(5000);
        } catch (SocketTimeoutException e) {
            return "timeout";
        } catch (Exception e) {
            e.printStackTrace();
        }
        String result = "";
        try {
            if (nsocket.isConnected()) {
                nis = nsocket.getInputStream();
                nos = nsocket.getOutputStream();
                nos.write(params[0].getBytes());
                byte[] buffer = new byte[4096];
                int read = nis.read(buffer, 0, 4096); //This is blocking
                while (read != -1) {
                    byte[] tempdata = new byte[read];
                    System.arraycopy(buffer, 0, tempdata, 0, read);
                    result = new String(tempdata);
                    read = -1;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            result = "";
        }
        return result;
    }

    @Override
    protected void onPostExecute(String s) {
        Log.d("Asynctask", s);
        if (s.length()==0){

            Functions.CreateAlertDialogBox(LoginActivity.this, "Error", "Please try

```



```

again");
        } else if (s.equals("timeout")) {

            Functions.CreateAlertDialogBox(LoginActivity.this, "Error", "Please
check your connection");
        } else {
            if (s.substring(s.length()-1)!="}") {
                s = s + "}";
            }
            try {
                nis.close();
                nos.close();
                nsocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
            try {
                JSONObject json_response = new JSONObject(s);
                if (json_response.get("response").equals("login") &&
json_response.get("status").equals("berhasil")) {
                    String saldo_str =
json_response.get("saldo").toString();
                    Integer saldo = Integer.parseInt(saldo_str);

                    SharedPreferences.Editor user_pref_editor =
getSharedPreferences("USER_PREF", MODE_PRIVATE).edit();
                    user_pref_editor.putString("username", username);
                    user_pref_editor.putInt("saldo", saldo);
                    user_pref_editor.apply();

                    MainActivity.StartRefresh(getApplicationContext());

                    Intent intent = new Intent(LoginActivity.this,
HomeActivity.class);
                    startActivity(intent);
                    finish();
                } else {

                    Functions.CreateAlertDialogBox(LoginActivity.this, "Error", "Please
check your credentials");
                }
            } catch (JSONException e) {
                e.printStackTrace();
                Functions.CreateAlertDialogBox(LoginActivity.this,
"Maaf", "Sepertinya terjadi kesalahan. Silakan coba lagi.");
            }
        }
        progressBar.setVisibility(View.GONE);
    }
}
}

```

MainActivity.java

```

package com.example.cobapplj;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        public static void StartRefresh(Context context){
            Intent alarmIntent = new Intent(context, UpdateReceiver.class);
            final PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, UpdateReceiver.REQUEST_CODE,
alarmIntent, PendingIntent.FLAG_UPDATE_CURRENT);
            long firstMillis = System.currentTimeMillis(); // alarm is set right
away
            AlarmManager alarm = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
            // First parameter is the type: ELAPSED_REALTIME,
ELAPSED_REALTIME_WAKEUP, RTC_WAKEUP
            // Interval can be INTERVAL_FIFTEEN_MINUTES, INTERVAL_HALF_HOUR,
INTERVAL_HOUR, INTERVAL_DAY

            alarm.setInexactRepeating(AlarmManager.RTC_WAKEUP,firstMillis+30000,30
000L,pendingIntent);
            Log.d("Refresh","Start Refresh");
        }

        public static void CancelRefresh(Context context){
            Intent alarmIntent = new Intent(context, UpdateReceiver.class);
            final PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, UpdateReceiver.REQUEST_CODE,
alarmIntent, PendingIntent.FLAG_UPDATE_CURRENT);
            AlarmManager alarm = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
            alarm.cancel(pendingIntent);
            Log.d("Refresh","Cancel Refresh");
        }
    }
}

```

PesanActivity.java

```

package com.example.cobapplj;

import android.content.DialogInterface;
import android.content.Intent;

```

```

import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

public class PesanActivity extends AppCompatActivity {
    public final int harga1=39000;
    public final int harga2=46000;
    public final int harga3=42000;
    public int qty1=0;
    public int qty2=0;
    public int qty3=0;
    public int total;
    public String notesStr;

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

    public void onPlus1Clk(View view){
        qty1+=1;
        TextView txtQty1=findViewById(R.id.qty1);
        txtQty1.setText(Integer.toString(qty1));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }

    public void onMin1Clk(View view){
        qty1=(qty1-1);
        if (qty1<0){
            qty1=0;
        }
        TextView txtQty1=findViewById(R.id.qty1);
        txtQty1.setText(Integer.toString(qty1));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }

    public void onPlus2Clk(View view){
        qty2+=1;
        TextView txtQty1=findViewById(R.id.qty2);
        txtQty1.setText(Integer.toString(qty2));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }

    public void onMin2Clk(View view){

```

```

        qty2=(qty2-1);
        if (qty2<0){
            qty2=0;
        }
        TextView txtQty1=findViewById(R.id.qty2);
        txtQty1.setText(Integer.toString(qty2));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }
    public void onPlus3Clk(View view){
        qty3+=1;
        TextView txtQty1=findViewById(R.id.qty3);
        txtQty1.setText(Integer.toString(qty3));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }

    public void onMin3Clk(View view){
        qty3=(qty3-1);
        if (qty3<0){
            qty3=0;
        }
        TextView txtQty1=findViewById(R.id.qty3);
        txtQty1.setText(Integer.toString(qty3));
        //update harga
        total=qty1*harga1+qty2*harga2+qty3*harga3;
        TextView txtTotal=findViewById(R.id.totalHarga);
        String totalStr=String.format("%,d", total);
        txtTotal.setText("Rp "+totalStr);
    }

    public void onBayarClk (View view){
        TextView txtNotes=findViewById(R.id.notes);
        notesStr=txtNotes.getText().toString();

        total=qty1*harga1+qty2*harga2+qty3*harga3;

        if (total<=0){
            AlertDialog.Builder builder = new
AlertDialog.Builder(PesanActivity.this);
            builder.setTitle("Cek lagi ya");
            builder.setMessage("Belum pesan apapun ya? Yuk pesan sesuatu");
            builder.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog dialog = builder.create();
            dialog.show();
        } else {
            String pesanan = "";

```

```

        String quantity = "";

        if (qty1>0){
            pesanan += "A";
            quantity += qty1;
        }

        if (qty2>0){
            if (pesanan.length()!=0){
                pesanan += ";";
                quantity += ";";
            }
            pesanan += "B";
            quantity += qty2;
        }

        if (qty3>0){
            if (pesanan.length()!=0){
                pesanan += ";";
                quantity += ";";
            }
            pesanan += "C";
            quantity += qty3;
        }

        Intent intent = new Intent(PesanActivity.this,
KonfirmasiPembayaran.class);
        intent.putExtra("totalPembayaran", total);
        intent.putExtra("pesanan",pesanan);
        intent.putExtra("quantity",quantity);
        intent.putExtra("notes",notesStr);
        startActivity(intent);
    }
}

public void onBackButtonClk(View view){
    finish();
}
}

```

RegisterActivity.java

```

package com.example.cobapplj;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;

```

```

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.SocketTimeoutException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class RegisterActivity extends AppCompatActivity {

    ProgressBar progressBar;
    String username;
    String password;
    String confirm_password;

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

        final EditText username_field =
            (EditText)findViewById(R.id.username_field);
        final EditText password_field =
            (EditText)findViewById(R.id.password_field);
        final EditText confirm_password_field =
            (EditText)findViewById(R.id.confirm_password_field);

        progressBar = (ProgressBar)findViewById(R.id.progress_bar);

        Button register_button = (Button)findViewById(R.id.register_button);
        register_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.d("socket", "register");

                username = username_field.getText().toString();
                password = password_field.getText().toString();
                confirm_password =
confirm_password_field.getText().toString();

                if (username.length()==0 || password.length()==0 ||
confirm_password_field.length()==0) {
                    progressBar.setVisibility(View.GONE);
                    Functions.CreateAlertDialogBox(RegisterActivity.this,
"Error", "Please fill every field");
                } else if
(password.length()<8||confirm_password.length()<8){

                    Functions.CreateAlertDialogBox(RegisterActivity.this, "Error", "Password

```

```

must be 8 characters or more");
        } else if (password.equals(confirm_password)){
            String hashtext = Functions.HashPassword(password,"SHA-
224");

            JSONObject send_json = new JSONObject();
            try {
                send_json.put("request","register");
                send_json.put("username",username);
                send_json.put("password",hashtext);

                String json_string = send_json.toString();

                progressBar.setVisibility(View.VISIBLE);

                new NetworkTask().execute(json_string);
            } catch (JSONException e) {
                e.printStackTrace();
            }

            } else {

                Functions.CreateAlertDialogBox(RegisterActivity.this,"Error","Please
insert the same password");
            }

        }

    });

    TextView register_login_text =
    (TextView)findViewById(R.id.register_login_text);
    register_login_text.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(RegisterActivity.this,
LoginActivity.class);
            startActivity(intent);
            finish();
        }
    });
}

@Override
public void onBackPressed() {
    super.onBackPressed();
    Intent intent = new Intent(RegisterActivity.this,
LoginActivity.class);
    startActivity(intent);
    finish();
}

private class NetworkTask extends AsyncTask<String, byte[], String> {

    SSLSocket nsocket; //Network Socket
    InputStream nis; //Network Input Stream
    OutputStream nos; //Network Output Stream

    @Override
    protected String doInBackground(String... params) { //This runs on a
different thread

```

```

        try {
            Log.i("AsyncTask", "doInBackground: Creating socket");
            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
                public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public X509Certificate[] getAcceptedIssuers() {
                    return new X509Certificate[0];
                }
            }}, new SecureRandom());
            SSLSocketFactory socketFactory =
sslContext.getSocketFactory();
            nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
            nsocket.setSoTimeout(5000);
        } catch (SocketTimeoutException e) {
            return "timeout";
        } catch (Exception e) {
            e.printStackTrace();
        }
        String result = "";
        try {
            if (nsocket.isConnected()) {
                nis = nsocket.getInputStream();
                nos = nsocket.getOutputStream();
                nos.write(params[0].getBytes());
                byte[] buffer = new byte[4096];
                int read = nis.read(buffer, 0, 4096); //This is blocking
                while (read != -1) {
                    byte[] tempdata = new byte[read];
                    System.arraycopy(buffer, 0, tempdata, 0, read);
                    result = new String(tempdata);
                    read = -1;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            result = "";
        }
        return result;
    }

    @Override
    protected void onPostExecute(String s) {
        Log.d("AsyncTask", s);
        if (s.equals("timeout")) {

            Functions.CreateAlertDialogBox(RegisterActivity.this, "Error", "Please
check your connection");
        } else {
            if (s.substring(s.length()-1)!="") {
                s = s + " ";
            }
            try {
                nis.close();
                nos.close();
                nsocket.close();
            }
        }
    }
}

```



```

        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            JSONObject json_response = new JSONObject(s);
            if (json_response.get("response").equals("register") &&
                json_response.get("status").equals("berhasil")) {
                SharedPreferences.Editor user_pref_editor =
                getSharedPreferences("USER_PREF", MODE_PRIVATE).edit();
                user_pref_editor.putString("username", username);
                user_pref_editor.putInt("saldo", 0);
                user_pref_editor.apply();

                MainActivity.StartRefresh(getApplicationContext());

                Intent intent = new Intent(RegisterActivity.this,
                HomeActivity.class);
                startActivity(intent);
                finish();
            } else {

                Functions.CreateAlertDialogBox(RegisterActivity.this, "Error", "Username
                sudah digunakan");
            }
            } catch (JSONException e) {
                e.printStackTrace();
            }

            Functions.CreateAlertDialogBox(RegisterActivity.this, "Maaf", "Sepertiny
            a terjadi kesalahan. Silakan coba lagi.");
        }
        progressBar.setVisibility(View.GONE);
    }
}
}

```

UpdateReceiver.java

```

package com.example.cobapplj;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class UpdateReceiver extends BroadcastReceiver {
    public static int REQUEST_CODE = 1;

    @Override
    public void onReceive(Context context, Intent intent) {
        Intent i = new Intent(context, UpdateService.class);
        context.startService(i);
    }
}

```

```
}
```

UpdateService.java

```
package com.example.cobapplj;

import android.app.IntentService;
import android.content.Intent;
import android.content.SharedPreferences;
import android.util.Log;

import androidx.annotation.Nullable;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.InputStream;
import java.io.OutputStream;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.X509TrustManager;

public class UpdateService extends IntentService {

    SSLSocket nsocket; //Network Socket
    InputStream nis; //Network Input Stream
    OutputStream nos; //Network Output Stream

    String update_request_string;

    public UpdateService() {
        super("UpdateService");
    }

    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        Log.d("Refresh", "Start Intent");
        SharedPreferences user_pref =
getSharedPreferences("USER_PREF", MODE_PRIVATE);
        String username = user_pref.getString("username", "");
        JSONObject update_request_json = new JSONObject();
        try {
            update_request_json.put("request", "showHistory");
            update_request_json.put("username", username);
            update_request_string = update_request_json.toString();
        } catch (Exception e) {
            e.printStackTrace();
            return;
        }
    }
}
```

```

        try {
            Log.i("Refresh", "Creating socket");
            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, new X509TrustManager[]{new
X509TrustManager(){
                public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {}
                public X509Certificate[] getAcceptedIssuers() {
                    return new X509Certificate[0];
                }
            }}, new SecureRandom());
            SSLSocketFactory socketFactory = sslContext.getSocketFactory();
            nsocket = (SSLSocket)
socketFactory.createSocket(Constants.server_addr, Constants.server_port);
            nsocket.setSoTimeout(5000);
        } catch (Exception e) {
            e.printStackTrace();
            return;
        }
        String result = "";
        try {
            if (nsocket.isConnected()) {
                nis = nsocket.getInputStream();
                nos = nsocket.getOutputStream();
                nos.write(update_request_string.getBytes());
                byte[] buffer = new byte[4096];
                int read = nis.read(buffer, 0, 4096); //This is blocking
                while (read != -1) {
                    byte[] tempdata = new byte[read];
                    System.arraycopy(buffer, 0, tempdata, 0, read);
                    result = new String(tempdata);
                    read = -1;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            result = "";
            return;
        }

        if (result.substring(result.length()-1)!=""){
            result = result + "}";
        }
        Log.d("Refresh", result);
        try {
            JSONObject json_response = new JSONObject(result);
            String isi = json_response.get("isi").toString();
            isi = isi.substring(1, isi.length() - 1);
            String[] arrayHistory = isi.split(",");
            List<History> history_list = new ArrayList<History>();
            boolean isReady = true;
            for (String history : arrayHistory) {
                String cleaned_history = history;
                int left_index = history.indexOf("[");
                if (left_index != -1) {
                    cleaned_history = history.substring(left_index + 1);
                }
                String[] history_content = cleaned_history.split(",");
            }
        }
    }
}

```

```

        String ordeno = history_content[0];
        String paket =
history_content[1].substring(1,history_content[1].length()-1);
        String quantity =
history_content[2].substring(1,history_content[2].length()-1);
        String status =
history_content[3].substring(1,history_content[3].length()-1);
        String timestamp =
history_content[4].substring(1,history_content[4].length()-1);
        String harga = history_content[5];
        String notes =
history_content[7].substring(1,history_content[7].length()-1);

        History history_instance = new
History(ordeno,timestamp,paket,quantity,harga,notes,status);
        history_list.add(history_instance);
    }
    List<History> last_history =
Functions.ReadHistoryFile(UpdateService.this);

    List<String> result_list =
Functions.CompareHistoryList(last_history,history_list);

    if (result_list.size()!=0){
        for (String compare_message : result_list){

            Functions.CreateNotification(getApplicationContext(),1,"Notification",
compare_message);
        }
    }
    Functions.WriteHistoryFile(UpdateService.this,history_list);
} catch (JSONException e) {
    e.printStackTrace();
} catch (Exception e){
    e.printStackTrace();
}

try {
    nis.close();
    nos.close();
    nsocket.close();
} catch (Exception e) {
    e.printStackTrace();
}

}

}

```