Nama : Amarramitha Poodja Thantawi

NIM : H1D022064

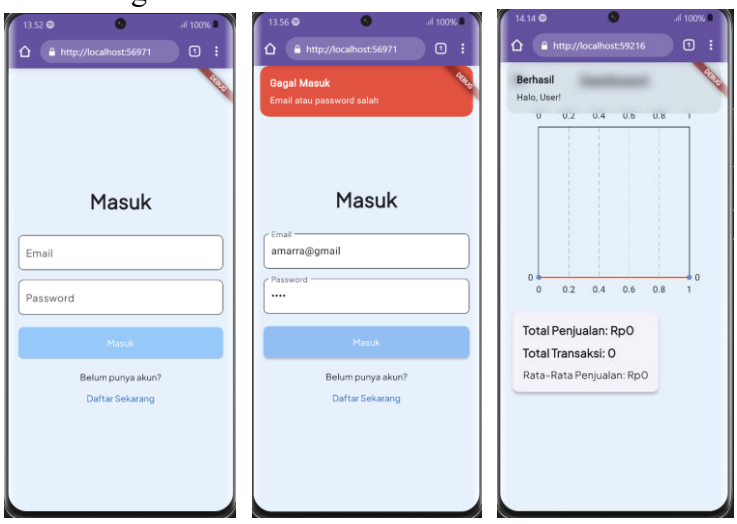
Laporan UAS Pemograman Mobile

Penjelasan Program

1. Proses Registrasi

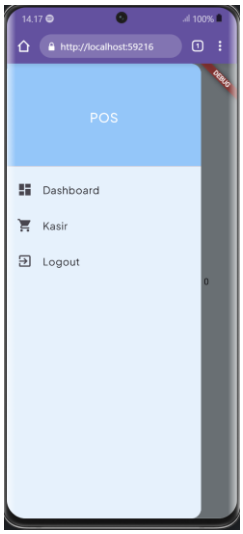
- Pengguna diminta untuk memasukkan email melalui **Textfield** yang dikendalikan oleh **_emailController**, dan password melalui **Textfield** yang dikendalikan oleh **_passwordController**.
- Ketika tombol "Daftar" ditekan, fungsi **registerWithEmailPassword** dipanggil. Fungsi ini memanfaatkan Firebase Authentication **(_auth.createUserWithEmailAndPassword)** untuk membuat pengguna baru dengan email dan password yang dimasukkan. Jika berhasil, Firebase akan mengembalikan objek **UserCredential** yang berisi informasi pengguna, termasuk **user** yang berisi ID unik pengguna (**uid**).
- Jika registrasi berhasil, sebuah snackbar muncul dengan pesan "Akun kamu berhasil dibuat!" menggunakan **Get.snackbar**. Kemudian, halaman berpindah ke layar login dengan menggunakan **Get.off(() => Login())**, yang menavigasi pengguna ke halaman login dan menghapus halaman registrasi dari stack navigasi.
- Di bagian bawah halaman registrasi, ada opsi untuk pengguna yang sudah memiliki akun untuk menuju halaman login menggunakan tombol **TextButton**. Ketika tombol ini ditekan, aplikasi akan berpindah ke halaman login.

2. Proses Login



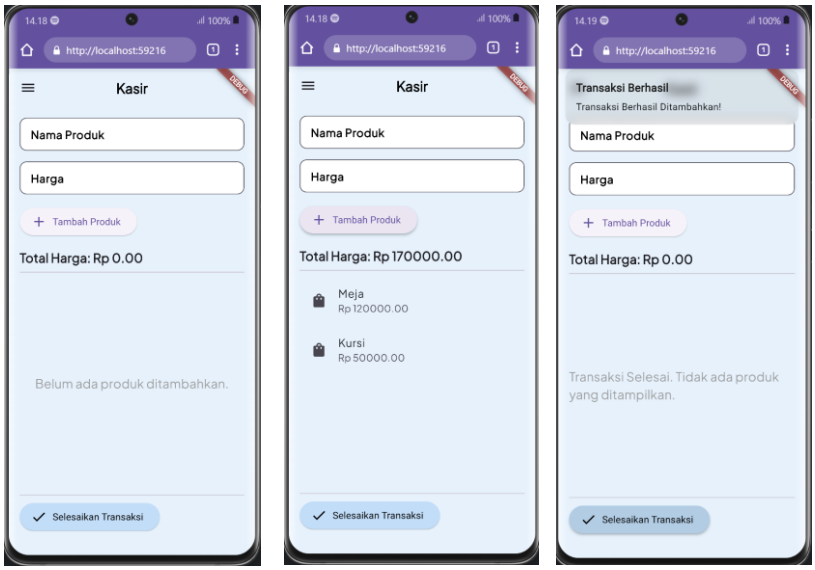
- Ketika pengguna memasukkan email dan password, controller **LoginController** akan memeriksa apakah keduanya sudah diisi.
- Jika email, password kosong atau salah, muncul pesan kesalahan menggunakan **Get.snackbar**.
- Jika keduanya diisi, aplikasi akan mencoba melakukan login dengan menggunakan **FirebaseAuth** dan metode **signInWithEmailAndPassword** untuk memverifikasi kredensial.
- Jika login berhasil, data pengguna (seperti nama) akan diambil dan status login disimpan dalam **SharedPreferences** dengan nilai true.
- Aplikasi kemudian menampilkan pesan sukses menggunakan **Get.snackbar** dan mengalihkan pengguna ke halaman utama (dashboard) menggunakan **Get.offAllNamed**.
- Jika login gagal, pesan kesalahan akan ditampilkan dengan informasi bahwa email atau password salah.

3. Sidebar



- **Drawer** adalah widget yang menyediakan menu samping yang dapat dibuka dengan menggeser atau menekan ikon menu di aplikasi. Sidebar ini berisi beberapa opsi navigasi yang mengarahkan pengguna ke halaman-halaman aplikasi yang berbeda.
- Bagian pertama dari **Drawer** adalah **DrawerHeader**, yang merupakan area header sidebar. Di sini, teks "POS" ditampilkan di tengah dengan latar belakang biru muda. Ini berfungsi sebagai header visual untuk menunjukkan bahwa sidebar ini berhubungan dengan sistem POS (Point of Sale).
- Setiap **ListTile** memiliki properti **onTap** yang memanggil fungsi navigasi dengan menggunakan **Get.offNamed()**. Fungsi **Get.offNamed()** digunakan untuk berpindah halaman menggunakan routing di aplikasi yang sudah diatur sebelumnya.
- Fungsi ini digunakan untuk mengarahkan pengguna ke halaman tertentu. **Get.offNamed('/dashboard')** akan mengarahkan pengguna ke halaman dashboard, **Get.offNamed('/cashier')** untuk halaman kasir, dan **Get.offAllNamed('/login')** untuk halaman login.
- Fungsi logout akan memanggil **FirebaseAuth.signOut()** untuk keluar dari Firebase.
- Status login dihapus dari **SharedPreferences**, dan aplikasi akan mengarahkan pengguna kembali ke halaman login.

4. Proses Tambah Transaksi Produk

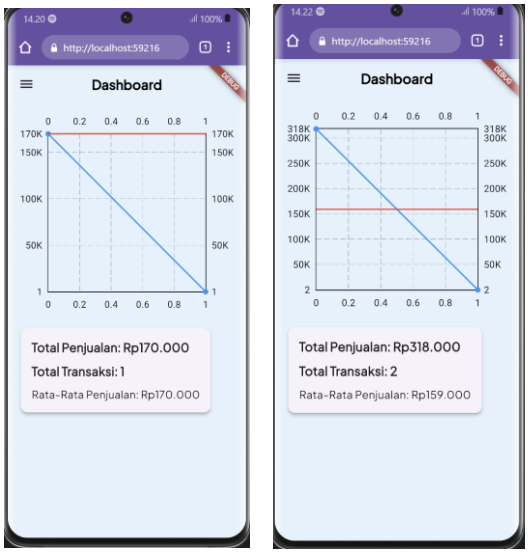


- Pada metode **onInit()**, yang dijalankan saat kontroler diinisialisasi, sistem mencoba memuat transaksi yang sedang aktif dengan memanggil fungsi **_loadActiveTransaction()**. Fungsi ini berfungsi untuk memeriksa apakah ada transaksi aktif yang sudah tersimpan di Firestore. Jika tidak ada, transaksi baru akan dimulai.
- Metode **startNewTransaction()** digunakan untuk memulai transaksi baru. Di sini, kita membuat ID transaksi baru dengan menggunakan **Firestore.instance.collection('transactions').doc().id** untuk menghasilkan ID yang unik. Selain itu, kita mengatur ulang daftar produk dan total harga menjadi kosong dan 0, serta menandakan bahwa transaksi ini belum selesai.
- Fungsi **addProduct()** digunakan untuk menambahkan produk ke dalam transaksi aktif. Produk yang ditambahkan disimpan dalam daftar products. Selain itu, total harga transaksi juga dihitung ulang dengan menambahkan harga produk yang baru. Jika transaksi sudah selesai, fungsi ini tidak akan menambahkan produk lagi.
- Metode **completeTransaction()** dipanggil ketika kasir ingin menyelesaikan transaksi. Pada langkah ini, status transaksi diubah menjadi "completed" dan data transaksi disimpan ke Firestore. Produk yang ada dalam transaksi diubah menjadi format map agar bisa disimpan di Firestore. Setelah transaksi selesai, kita menandakan bahwa transaksi ini sudah selesai dengan mengubah nilai **transactionCompleted** menjadi **true.** Setelah transaksi selesai, produk yang ada disimpan ke Firestore, dan UI akan memperbarui status transaksi.
- Pada bagian tampilan, kita menggunakan **ListView.builder()** untuk menampilkan produk yang sudah ditambahkan ke dalam transaksi. Setiap produk ditampilkan

dengan nama dan harga yang diambil dari objek Product. UI ini akan terupdate secara otomatis setiap kali ada produk yang ditambahkan ke dalam daftar.

- Untuk menampilkan total harga yang terupdate di UI, kita menggunakan widget **Obx()**, yang memungkinkan tampilan otomatis merespon perubahan pada variabel reaktif seperti **totalAmount**. Setiap kali produk baru ditambahkan, nilai **totalAmount** akan diperbarui dan UI akan menampilkan nilai terbaru.

5. Dashboard



- Pada saat controller diinisialisasi, fungsi **onInit()** dipanggil dan secara otomatis memanggil **fetchSalesSummary()**. Fungsi ini bertanggung jawab untuk mengambil data transaksi yang terkait dengan pengguna yang sedang login, menggunakan FirebaseAuth untuk mendapatkan ID pengguna yang aktif. Data transaksi tersebut kemudian diambil dari koleksi Firestore dengan mengakses dokumen berdasarkan ID pengguna, dan hasilnya digunakan untuk menghitung total penjualan serta jumlah transaksi yang terjadi.
- **FetchSalesSummary()** berfungsi untuk mengambil semua transaksi yang terkait dengan pengguna saat ini dan menghitung total penjualan serta jumlah transaksi. Setiap transaksi berisi daftar produk, yang harga produk-produk tersebut dihitung untuk mendapatkan total penjualan. Hasil perhitungan ini kemudian disimpan dalam variabel observable seperti totalSales dan totalTransactions, yang selanjutnya akan memengaruhi tampilan UI.
- Di dalam **SingleChildScrollView**, pertama-tama ditampilkan grafik **LineChart** menggunakan **fl_chart** yang menggambarkan data total penjualan dan jumlah transaksi. Grafik ini memperlihatkan dua titik utama: total penjualan pada titik

pertama dan total transaksi pada titik kedua. Grafik ini juga menampilkan garis rata-rata penjualan yang dihitung berdasarkan total penjualan dibagi dengan jumlah transaksi.

- Di bawah grafik, terdapat kartu yang menampilkan informasi penting secara ringkas: total penjualan, total transaksi, dan rata-rata penjualan. Data ini diperbarui secara dinamis menggunakan widget **Obx()** yang memungkinkan UI untuk merespon perubahan pada data observable di **DashboardController**. Setiap kali nilai total penjualan atau jumlah transaksi berubah, teks pada card ini akan langsung terupdate.