Proyek Mobile Programming

(Aplikasi *Kuliah.ku*)

Disusun Oleh :

Kevin Ardi Setyawan :2313025057

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS LAMPUNG

2025

BAB I
PENDAHULUAN

I. Latar Belakang

Perkembangan teknologi mobile memberikan peluang baru di dunia pendidikan, khususnya dalam mendukung aktivitas belajar mahasiswa. Tapi, masih banyak mahasiswa yang kesulitan dalam mengatur jadwal kuliah, mencatat materi, mengelola tugas, dan meengelola keuangan dengan baik. Saya membuat aplikasi bernama *Kuliah.ku* yang dirancang untuk menjadi solusi digital yang membantu mahasiswa dalam mengatur perkuliahan, manajemen waktu, serta meningkatkan efektivitas di dunia perkuliahan. Dengan fitur seperti jadwal kuliah, manajemen tugas, catatan digital, dan manajemen keuangan, aplikasi ini diharapkan dapat menjadi asisten belajar pribadi mahasiswa.

II. Rumusan Masalah

Beberapa permasalahan yang sering dihadapi mahasiswa:

1. Kesulitan mengatur jadwal kuliah dan pengingat waktu.
2. Sulit memantau jadwal, deadline tugas dan ujian secara terorganisir.
3. Mahasiswa terkadang merasa sulit untuk mengatur keuangan.
4. Tidak adanya aplikasi terpadu yang memadukan jadwal, tugas, catatan, dan keuangan dalam satu platform.

III. Timeline

Minggu	Kegiatan	Keterangan
1-2	Analisis fitur dan kebutuhan, pembuatan konsep rancangan awal.	✓
3-4	Perancangan struktur aplikasi (Use case diagram, flowchart, dan ERD database)	✓
5	Membuat desain UI/UX dan database	✓
5-7	Implementasi pembuatan fitur dan halaman jadwal, to do list, dan manajemen keuangan	✓
8-13	Melanjutkan untuk mengembangkan desain UI/UX agar lebih menarik dan interaktif	✓
14	Pengujian dan perbaikan bug.	✓
15	Finalisasi aplikasi dan laporan akhir.	✓

LINK github:

https://github.com/Kevinardi17/myapp.git

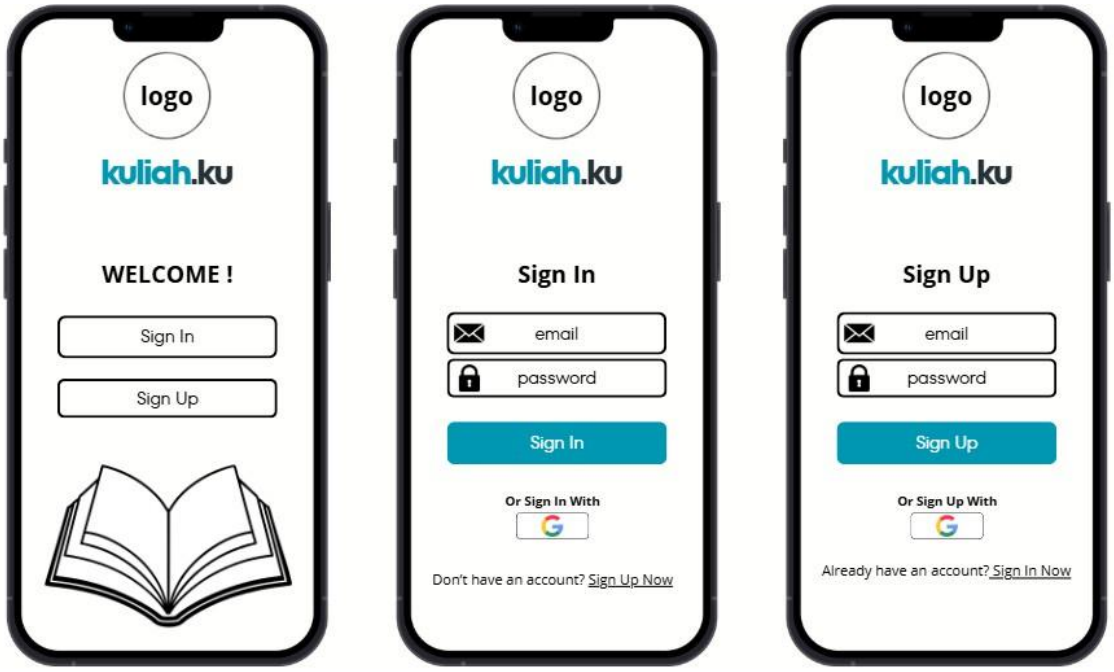
Link Video UTS:

https://youtu.be/_fUpYCjQSjI?si=VLfUi2cFRoTzFnEs

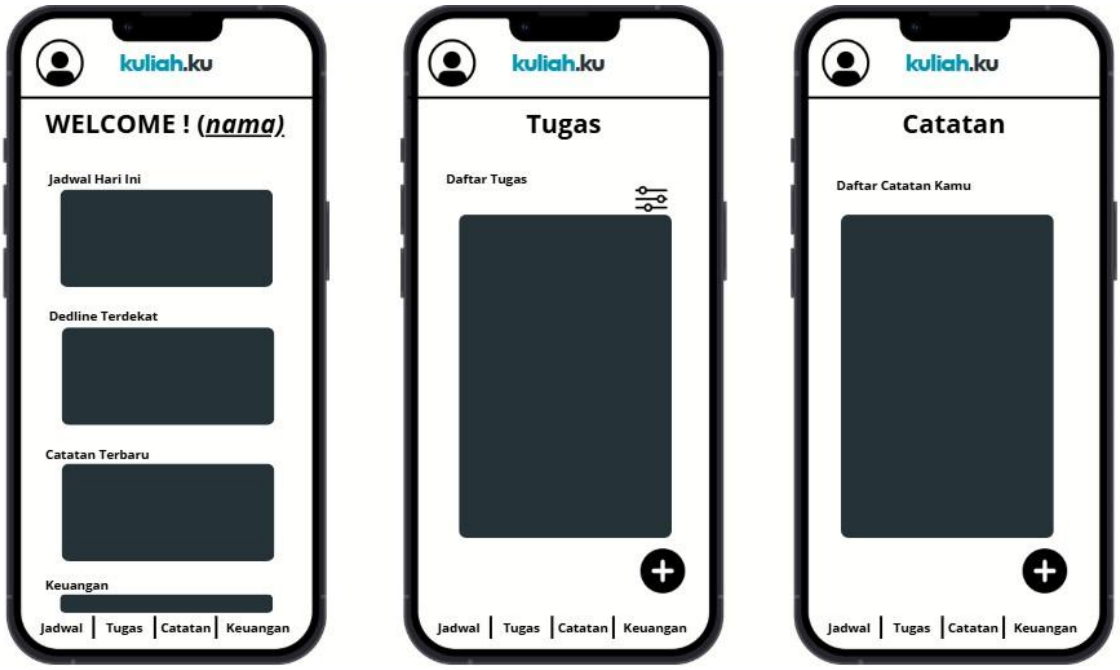
BAB II
PEMBAHASAN
(Pelaksanaan Timeline)

A. Minggu 1 :Rancangan Desain Awal (Mockup) 1.

Halaman Welcome (sign in/sign up)



2. Dashboard dan Menu



B. Minggu 2-4: Analisis fitur dan kebutuhan, pembuatan konsep rancangan awal dan

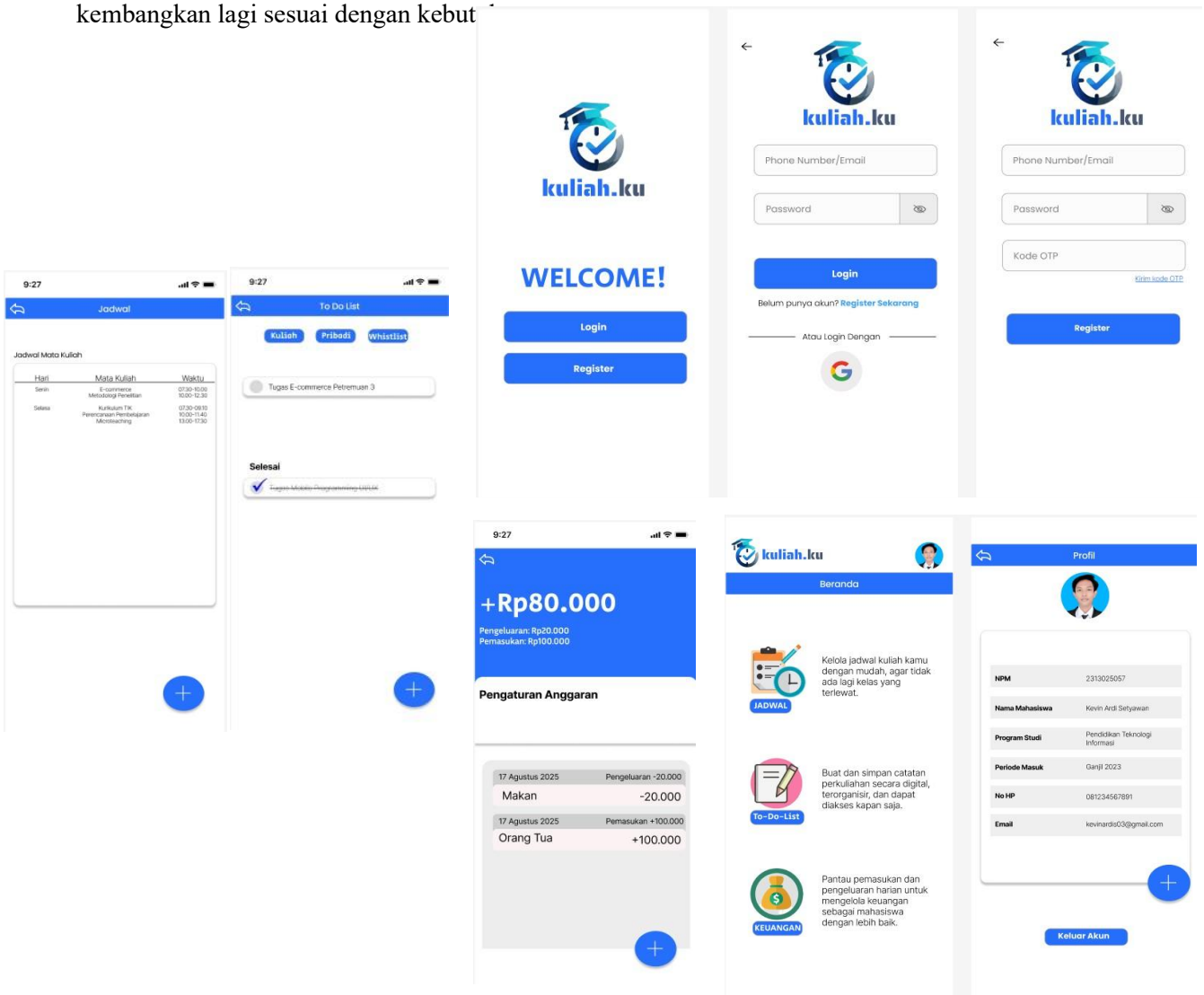
Perancangan struktur aplikasi (Use case diagram, flowchart, dan ERD database)

Link dokumen SRS:
https://drive.google.com/drive/folders/1FhaMQq9TmdoUJ7RrK9WJ5DklOSZRjagU?usp=sharing

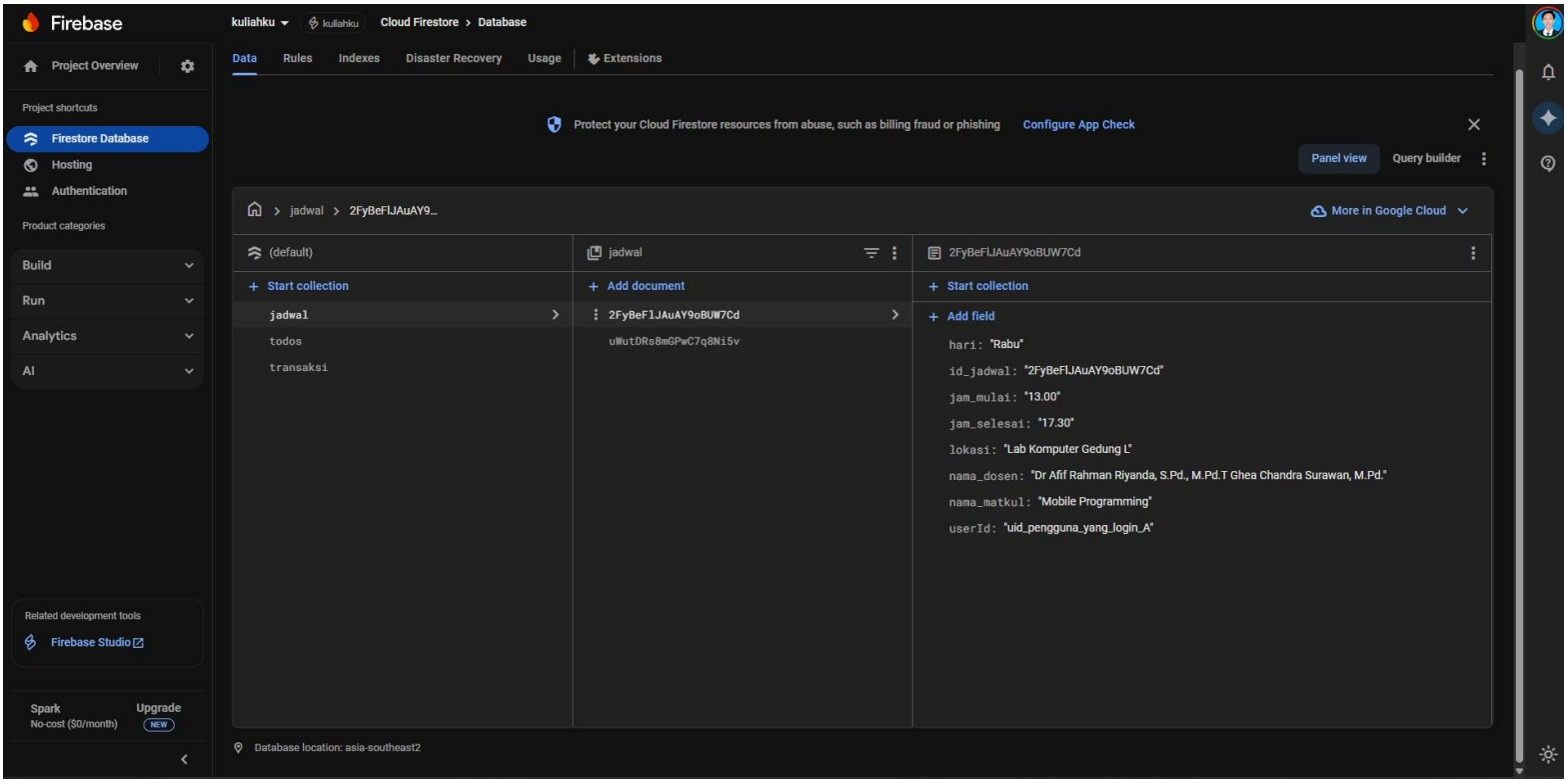
C. Minggu 5: Membuat desain UI/UX dan Database

1. Desain UI/UX

Dalam Pembuatan desain UI/UX, saya menggunakan aplikasi Figma. Saya menggunakan refrensi dari aplikasi-aplikasi yang sudah ada di playstore lalu saya implementasikan dan kembangkan lagi sesuai dengan kebut

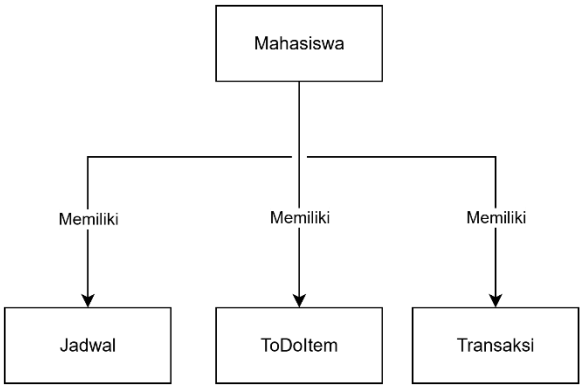


2. Desain database



Dalam merancang database, terlebih dahulu saya sudah membuat SRS/SKPL untuk kebutuhan sistem saya (link dokumen saya sertakan di laporan perkembangan projek timeline 2-4). Di dalam dokumen SRS, saya merancang diagram ERD untuk databasenya sesuai dengan kebutuhan fungsional, non fungsional. Berikut adalah penjelasan dari database saya:

1) **ERD 1**



Penjelasan Hubungan (Kardinalitas):

- Seorang **Mahasiswa** dapat **memiliki BANYAK** (<) **Jadwal**.
- Seorang **Mahasiswa** dapat **memiliki BANYAK** (<) **ToDoItem**.
- Seorang **Mahasiswa** dapat **memiliki BANYAK** (<) **Transaksi**.

Ini adalah hubungan **satu-ke-banyak (one-to-many)**.

2) ERD 2

Jadwal		
Nama Kolom	Tipe Data	Keterangan
id_jadwal	STRING	Primary Key (ID Dokumen unik dari Firestore)
userId	STRING	**Foreign Key (FK)** 🔑 ke ID Pengguna
nama_matkul	STRING	Nama mata kuliah
hari	STRING	Hari perkuliahan
jam_mulai	STRING	Waktu mulai kuliah (format HH:MM)
jam_selesai	STRING	Waktu selesai kuliah (format HH:MM)
lokasi	STRING	Lokasi atau ruang kelas
nama_dosen	STRING	Nama dosen pengampu (opsional)

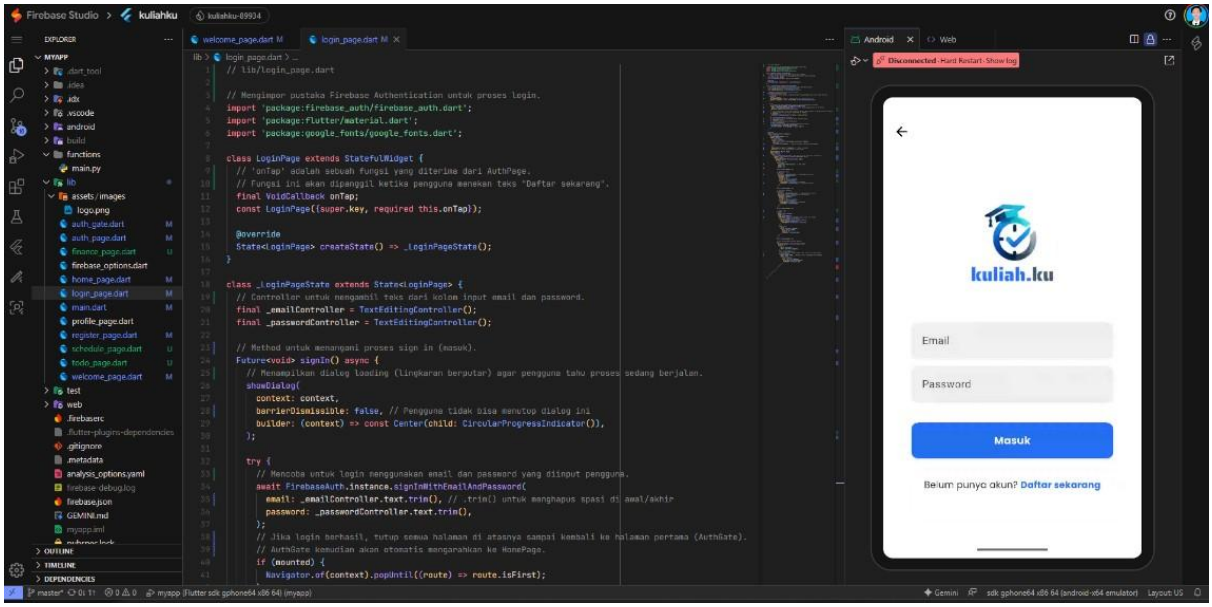
ToDo_Item		
Nama Kolom	Tipe Data	Keterangan
id_todo	String	Primary Key (PK)
userId	String	**Foreign Key** ke ID pengguna dari Authentication
judul	String	judul atau nama tugas
kategori	String	"Kuliah", "Pribadi", atau "Wishlist"
tenggat_waktu	Timestamp	Tipe data tanggal/waktu di Firebase
status_selesai	BOOLEAN	`true` jika selesai, `false` jika belum

Transaksi		
Nama Kolom	Tipe Data	Keterangan
id_transaksi	String	Primary Key (PK)
userId	String	**Foreign Key** ke ID pengguna dari Authentication
tipe	String	"Pemasukan" / "Pengeluaran"
jumlah	NUMBER	Nominal uang
keterangan	STRING	deskripsi singkat transaksi
tanggal	TIMESTAMP	Tipe data tanggal/waktu standar Firebase

D. Minggu 5:

Implementasi Pembuatan Halaman Login, Register dan Halaman Dashboard

1) Fitur Halanman Login



Saya mengimplementasikan pembuatan halaman **Login** untuk aplikasi saya "Kuliah.ku" yang terhubung dengan **Firebase Authentication**. Halaman ini memungkinkan pengguna untuk masuk ke dalam aplikasi menggunakan alamat email dan password mereka. Jika login berhasil, pengguna akan diarahkan langsung ke halaman utama melalui mekanisme pemantauan status autentikasi.

1. **Autentikasi dan Inisialisasi**

   Halaman ini menggunakan **Firebase Authentication** untuk proses login. Saat pengguna menekan tombol "Masuk", fungsi signIn() akan dijalankan, yang akan mengirimkan email dan password ke Firebase untuk proses verifikasi.

   Untuk memastikan kenyamanan pengguna:

   1) Ditampilkan **loading indicator** saat proses login berlangsung.
   2) Jika terjadi error (misalnya password salah atau email tidak ditemukan), akan ditampilkan **SnackBar** berisi pesan error dari Firebase.

   Selain itu, halaman ini juga menerima callback onTap dari halaman AuthPage, yang digunakan untuk berpindah ke halaman registrasi jika pengguna belum memiliki akun.

2. **Tampilan Utama dan Struktur Halaman**

   Tampilan halaman dibangun menggunakan Scaffold, dengan komponen utama sebagai berikut:

1) AppBar transparan, dengan ikon kembali (back) untuk kembali ke halaman sebelumnya (biasanya halaman selamat datang / WelcomePage).
2) Body utama menggunakan SingleChildScrollView, agar tampilan tetap bisa discroll saat layar terlalu kecil (misalnya di HP dengan resolusi kecil).
3) Logo aplikasi ditampilkan di bagian atas sebagai branding.

3. Form Input Email dan Password

Pengguna dapat memasukkan **email** dan **password** melalui dua buah TextField:

1) Field **Email** menerima teks biasa.
2) Field **Password** menggunakan properti obscureText: true untuk menyembunyikan karakter demi keamanan.

Kedua field menggunakan TextEditingController agar nilai yang dimasukkan bisa diakses di dalam fungsi signIn().

4. Tombol Login

Tombol "Masuk" menggunakan widget ElevatedButton, yang ketika ditekan akan memanggil fungsi signIn.

Fitur tambahan:

1) Desain tombol sudah disesuaikan agar konsisten dengan tema aplikasi (warna biru utama, border melengkung, teks putih).
2) Tombol mengisi lebar penuh layar (width: double.infinity) agar mudah ditekan.

5. Navigasi ke Halaman Registrasi

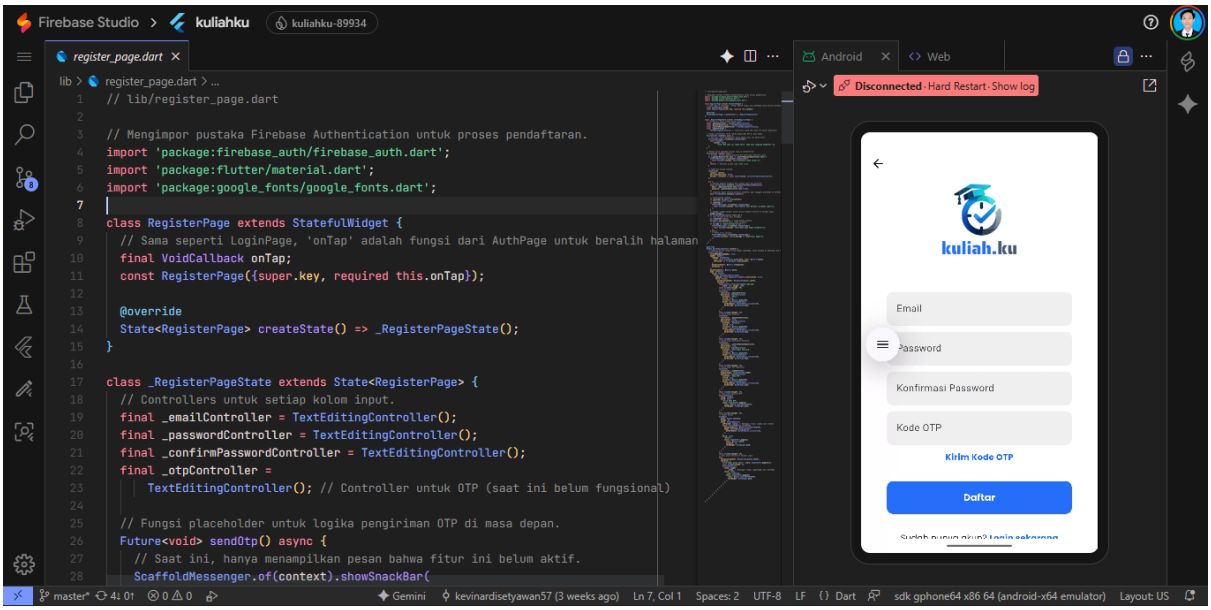
Di bagian bawah halaman terdapat teks:

"Belum punya akun? Daftar sekarang"

Teks "Daftar sekarang" dapat diketuk oleh pengguna dan menggunakan GestureDetector untuk menangani tap gesture.

Saat ditekan, fungsi widget.onTap akan dipanggil — ini merupakan callback dari AuthPage yang berfungsi menukar tampilan antara halaman login dan register.

2) Fitur Halaman Register



Saya mengimplementasikan pembuatan halaman **Login** untuk aplikasi saya "Kuliah.ku" yang terhubung dengan **Firebase Authentication**. Halaman ini memungkinkan pengguna membuat akun baru menggunakan email dan password. Proses validasi dan feedback dilakukan secara langsung, termasuk pengecekan kesesuaian password dan penanganan error dari Firebase. Selain fitur inti, halaman ini juga telah disiapkan dengan struktur untuk mendukung **fitur OTP**, meskipun saat ini masih bersifat placeholder.

1. Autentikasi dan Inisialisasi

Pada halaman ini, saya menggunakan instance dari FirebaseAuth untuk membuat akun pengguna baru berdasarkan email dan password yang dimasukkan.
Sebelum melakukan pendaftaran:

- Sistem memvalidasi bahwa password dan konfirmasi password harus **cocok**.
- Setelah pendaftaran berhasil, pengguna akan langsung **di-logout** agar dapat login secara manual, sesuai praktik umum untuk verifikasi identitas.
- Callback onTap dari AuthPage dipanggil untuk **mengalihkan pengguna ke halaman login** setelah sukses mendaftar.

Fitur **OTP** juga telah ditambahkan dalam bentuk input dan tombol, namun fungsinya masih belum diaktifkan (akan dikembangkan di masa depan).

2. Tampilan Utama dan Struktur Halaman

Struktur halaman dibangun menggunakan Scaffold, dengan layout dan gaya visual yang **konsisten dengan halaman login**:

- AppBar transparan dengan ikon kembali untuk navigasi ke halaman sebelumnya.

- Body menggunakan SingleChildScrollView agar tetap bisa di-scroll jika layar terlalu kecil.
- Logo aplikasi ditampilkan di bagian atas sebagai identitas visual.

3. Form Input

Terdapat **empat kolom input utama**, masing-masing dikontrol oleh TextEditingController:

1. **Email**
   - Menerima alamat email pengguna.
2. **Password**
   - Diinput secara tersembunyi (obscureText: true) demi keamanan.
3. **Konfirmasi Password**
   - Digunakan untuk memastikan password yang dimasukkan benar dan cocok.
4. **OTP (Kode)**
   - Input tambahan untuk kode OTP, namun saat ini belum terhubung ke sistem verifikasi.

Terdapat juga tombol "Kirim Kode OTP" yang saat ini hanya menampilkan pesan bahwa fitur tersebut belum aktif.

4. Tombol Daftar

Tombol **"Daftar"** disiapkan menggunakan ElevatedButton, yang saat ditekan akan menjalankan fungsi signUp().

Fungsi ini melakukan langkah-langkah berikut:

- Validasi kecocokan password.
- Menampilkan loading dialog (CircularProgressIndicator).
- Mencoba membuat akun baru dengan Firebase.
- Logout otomatis dan menampilkan pesan berhasil kepada pengguna.
- Berpindah ke halaman login menggunakan widget.onTap().
- Jika terjadi error, misalnya:
- Email sudah digunakan: akan ditampilkan pesan khusus.
- Error lain: akan ditampilkan pesan error dari Firebase.

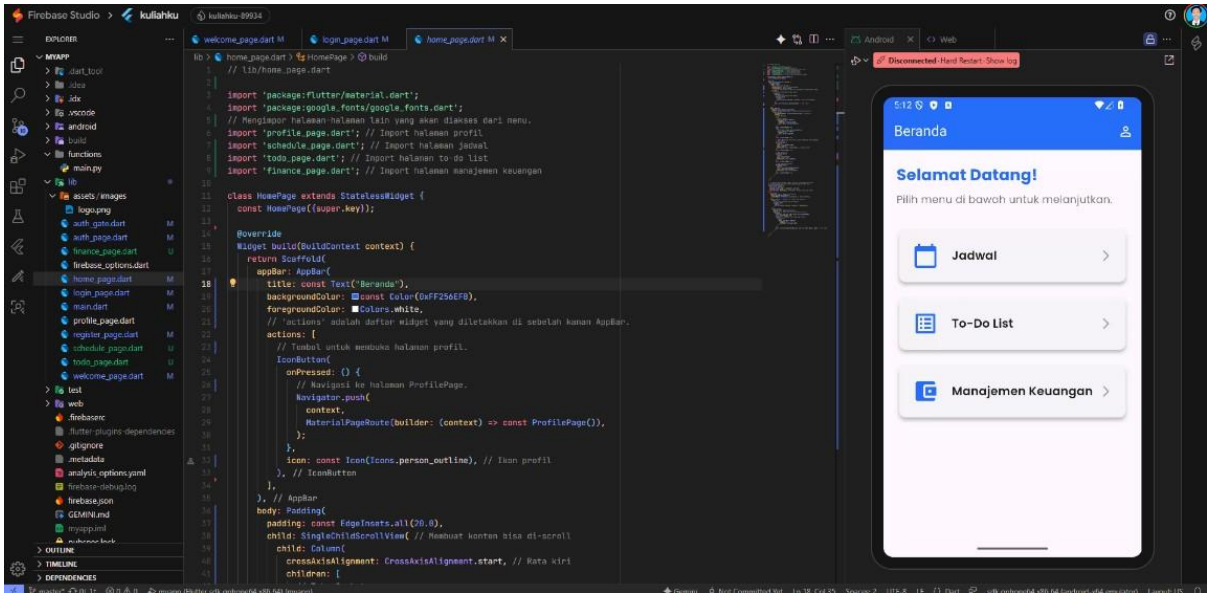
5. Navigasi ke Halaman Login

Di bagian bawah halaman, terdapat teks:

"Sudah punya akun? Login sekarang"

Teks ini dapat diklik oleh pengguna dan menggunakan GestureDetector untuk memanggil fungsi onTap, yang berfungsi mengalihkan tampilan ke **halaman login**. Ini penting untuk memungkinkan navigasi antar dua halaman otentikasi utama.

3) Fitur Halaman Dashboard



Kode ini merupakan implementasi **halaman beranda utama** dari aplikasi saya "Kuliah.ku" yang berfungsi sebagai **pusat navigasi** ke berbagai fitur utama aplikasi, seperti **Jadwal Kuliah**, **To-Do List**, **Manajemen Keuangan**, dan **Profil Pengguna**. Halaman ini dirancang agar mudah diakses, ramah pengguna, serta menyajikan tampilan yang bersih dan modern menggunakan GoogleFonts dan Material Design.

1. Tampilan dan Struktur Halaman

Halaman dibangun menggunakan widget Scaffold, yang terdiri dari dua bagian utama:

1) **AppBar**

   Terletak di bagian atas layar, AppBar menampilkan:

   - Judul halaman ("Beranda")
   - Tombol ikon profil di sebelah kanan yang mengarahkan ke halaman **ProfilePage**

2) **Body (Konten Utama)**

   Konten utama berisi:

   - Teks sambutan ("Selamat Datang!") dan instruksi singkat
   - Deretan kartu menu interaktif untuk mengakses fitur-fitur utama aplikasi

2. Navigasi Fitur

Halaman Home ini menyediakan **tiga menu utama**, masing-masing ditampilkan dalam bentuk **Card (kartu)** dengan ikon, teks, dan tombol panah:

1) Jadwal Kuliah
   - Diwakili oleh ikon kalender

- Mengarahkan ke halaman SchedulePage

2) **To-Do List**
   - Diwakili oleh ikon daftar tugas
   - Mengarahkan ke halaman TodoPage
3) **Manajemen Keuangan**
   - Diwakili oleh ikon dompet
   - Mengarahkan ke halaman FinancePage

Setiap kartu menu dibangun menggunakan metode khusus _buildMenuItemCard() untuk menjaga konsistensi dan modularitas kode.

3. Komponen Reusable: _buildMenuItemCard()

Fungsi _buildMenuItemCard() adalah helper widget yang digunakan untuk membuat tampilan setiap kartu menu. Fitur utamanya:

1) Menampilkan **ikon fitur**, **judul**, dan **ikon panah** sebagai indikator interaksi
2) Menggunakan widget InkWell agar bisa merespons sentuhan dengan efek ripple
3) Navigasi dilakukan melalui Navigator.push() ke halaman yang dituju
   Dengan pendekatan ini, proses penambahan menu baru menjadi lebih efisien dan terstruktur.

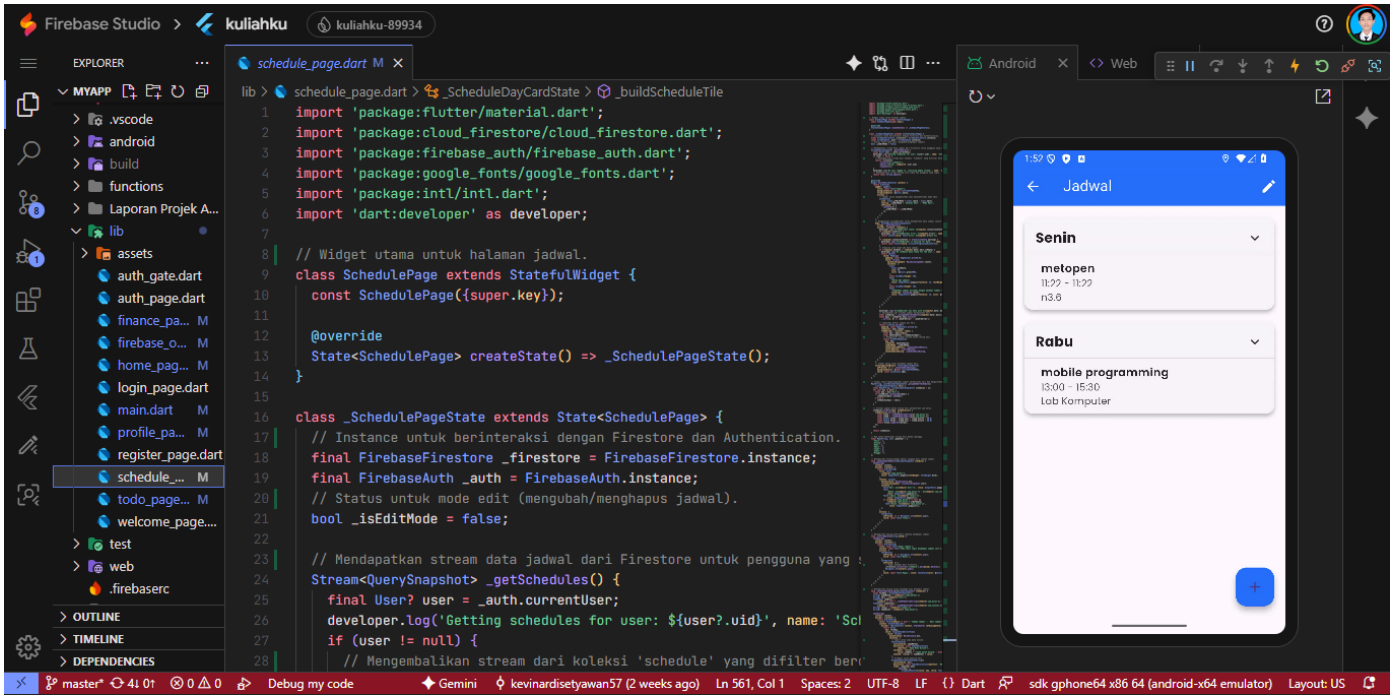
4. Navigasi ke Halaman Profil

Ikon **akun** di AppBar digunakan untuk masuk ke halaman **profil pengguna** (ProfilePage). Navigasinya dilakukan melalui Navigator.push() dengan MaterialPageRoute.

E. Minggu 5-7:

Implementasi Pembuatan Fitur dan Halaman Jadwal, To Do List, dan Manajemen Keuangan

4) Fitur Halaman Jadwal



Selanjutnya saya mengimplementasikan pembuatan halaman **Jadwal** dengan integrasi **Firebase Authentication** dan **Cloud Firestore** sebagai backend-nya. Fungsinya adalah untuk membantu pengguna mencatat, melihat, dan mengelola jadwal kuliah secara real-time, dengan data yang terorganisir berdasarkan hari dan waktu.

1. **Tampilan Utama dan Struktur Halaman**

   Halaman ini menggunakan struktur Scaffold, dengan komponen utama:

   1) **AppBar** menampilkan judul "Jadwal Kuliah" serta tombol mode edit untuk memungkinkan pengguna mengubah atau menghapus jadwal yang sudah dibuat.
   2) **FloatingActionButton (FAB)** di bagian kanan bawah digunakan untuk menambahkan jadwal baru.
   3) **Body** halaman dibangun dengan StreamBuilder, yang mengambil data jadwal dari Firestore dan akan otomatis diperbarui jika ada perubahan, tanpa perlu me-reload aplikasi.

2. **Pengelompokan dan Penampilan Jadwal**

   Semua data jadwal yang tersimpan di Firestore dikelompokkan berdasarkan **hari** (Senin–Minggu), lalu diurutkan berdasarkan **jam mulai**.

   Untuk menampilkan data ini, saya menggunakan widget khusus bernama ScheduleDayCard, yang berfungsi sebagai **kartu per hari**. Kartu ini bersifat

expandable, artinya pengguna dapat mengetuk kartu untuk melihat atau menyembunyikan detail seluruh jadwal pada hari tersebut.

3. **Menambahkan dan Mengubah Jadwal**

   Fungsi penambahan dan pengeditan jadwal dilakukan melalui **dialog form** yang ditampilkan saat pengguna menekan FAB atau tombol edit pada jadwal.

   Form input mencakup:

   - Nama mata kuliah
   - Hari kuliah
   - Jam mulai dan selesai
   - Lokasi
   - Nama dosen *(opsional)* Sebelum data dikirim ke Firestore:
   - Input divalidasi untuk memastikan tidak kosong
   - Data jam dikonversi dari String ke TimeOfDay menggunakan DateFormat agar dapat digunakan dalam pengurutan jadwal.

   Penambahan dilakukan menggunakan metode .add(), sedangkan pengeditan dilakukan dengan .update() berdasarkan ID dokumen di Firestore.

4. **Mode Edit dan Penghapusan**

   Terdapat fitur **mode edit** yang dapat diaktifkan melalui ikon di AppBar. Saat mode ini aktif:

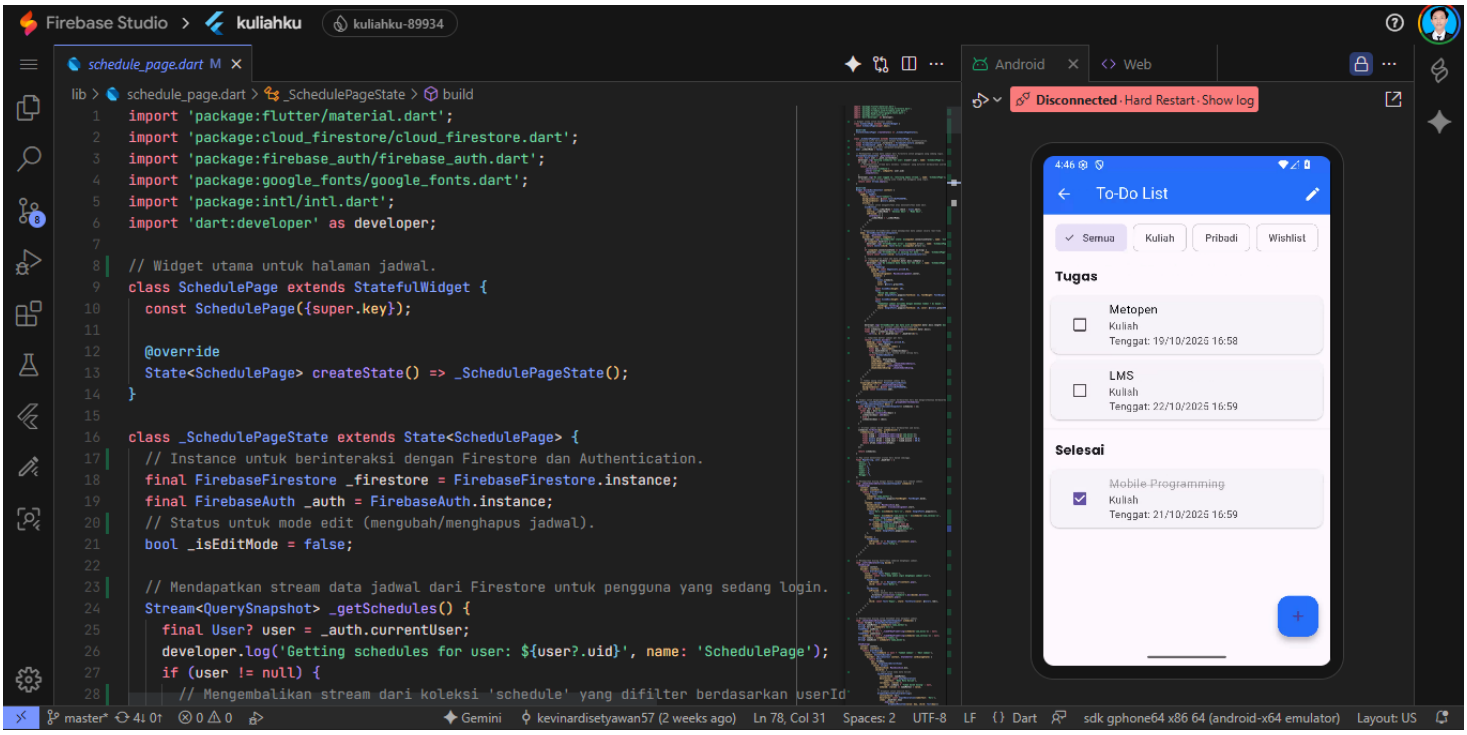
   - Setiap jadwal akan menampilkan ikon edit dan hapus
   - Pengguna bisa langsung mengubah isi jadwal atau menghapusnya dari Firestore
   - Mode ini dirancang agar tidak mengganggu tampilan utama, dan hanya aktif saat dibutuhkan pengguna.

5. **Tampilan Real-Time dan Responsif**

   Dengan menggunakan StreamBuilder, semua data yang tampil akan **langsung diperbarui secara real-time** setiap kali ada perubahan di Firestore — baik penambahan, pengeditan, maupun penghapusan.

   Data juga ditampilkan dengan rapi dalam bentuk kartu per hari, dan diurutkan agar memudahkan pengguna melihat urutan jadwal kuliah dari pagi hingga sore.

5) Fitur Halaman To Do List



Selanjutnya saya mengimplementasikan pembuatan halaman **To-Do List** dengan integrasi **Firebase Authentication** dan **Cloud Firestore** sebagai backend-nya. Fungsinya adalah untuk memungkinkan pengguna mencatat, mengelola, dan memantau berbagai tugas mereka berdasarkan kategori yang ditentukan.

1. **Tampilan Utama dan Struktur Halaman**

   Halaman ini dibangun dengan menggunakan widget Scaffold, di mana:

   1. AppBar berisi judul "To-Do List" dan tombol untuk mengaktifkan **mode edit** (edit/hapus tugas).
   2. FloatingActionButton disediakan untuk menambahkan tugas baru.
   3. Isi halaman (body) terdiri dari dua bagian utama:
      - **Filter kategori** (_buildCategoryFilter) di bagian atas, untuk menyaring tampilan tugas berdasarkan kategori yang dipilih (Kuliah, Pribadi, Wishlist, atau Semua).
      - **Daftar tugas** (_buildTodoList) yang ditampilkan menggunakan StreamBuilder, agar data Firestore dapat dipantau dan diperbarui secara **real-time**.

2. **Filter Kategori**

   Untuk kategori, saya menggunakan FilterChip agar pengguna dapat dengan mudah memilih jenis tugas yang ingin ditampilkan. Pemilihan ini akan memengaruhi query data yang diambil dari Firestore.

3. **Menampilkan dan Memfilter Tugas**

   Semua data tugas disimpan dalam koleksi Firestore bernama ToDo_Item, dengan filter berdasarkan userId. Data kemudian dibagi menjadi dua kelompok:

   - **Tugas yang belum selesai**
   - **Tugas yang sudah selesai**

   Saya juga menerapkan logika untuk memfilter berdasarkan kategori yang sedang dipilih, misalnya hanya menampilkan tugas "Kuliah" saja. Tampilan tugas dikelompokkan dengan jelas, dan jika tidak ada data yang sesuai filter, maka ditampilkan pesan "Tidak ada tugas dalam kategori ini".

4. **Menambah dan Mengubah Tugas**

   Fungsi _showAddTaskDialog() menampilkan dialog input yang fleksibel:

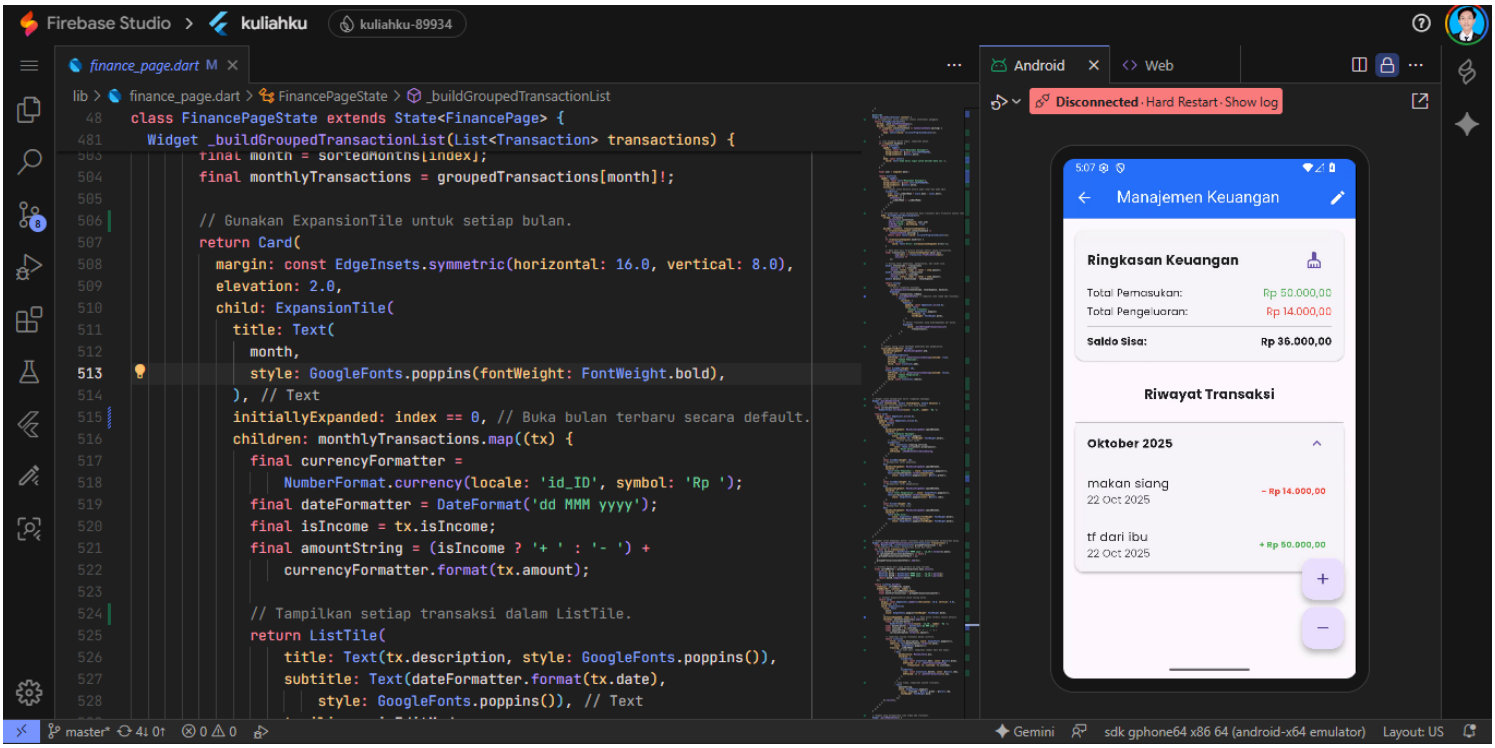
   - Pengguna dapat memilih kategori tugas, mengisi judul, deskripsi, serta tenggat waktu (kecuali untuk kategori "Pribadi", yang memang tidak memerlukan tenggat).
   - Dialog ini juga digunakan untuk edit data jika parameter task diisi.
   - Tugas baru ditambahkan dengan **.add()** ke Firestore, sementara untuk pengeditan digunakan **.update().**

   Saya juga memastikan validasi input dilakukan sebelum menyimpan, terutama agar judul tidak kosong.

5. **Checklist Tugas Selesai**

   Setiap item tugas dilengkapi dengan checkbox, sehingga pengguna bisa langsung menandai tugas sebagai selesai. Status ini akan otomatis diperbarui ke Firestore menggunakan metode .update().

6) Fitur Halaman Manajemen Keuangan



Selanjutnya saya mengimplementasikan pembuatan halaman **To-Do List** dengan integrasi **Firebase Authentication** dan **Cloud Firestore** sebagai backend-nya. Fungsinya adalah untuk memungkinkan pengguna mencatat pemasukan dan pengeluaran secara real-time, serta memantau kondisi keuangan mereka melalui ringkasan saldo dan riwayat transaksi.

1. **Tampilan Utama dan Struktur Halaman**

   Halaman ini dibangun dengan menggunakan widget Scaffold, dengan struktur utama sebagai berikut:

   1) **AppBar** berisi judul "Manajemen Keuangan" serta ikon edit untuk mengaktifkan **mode edit** (untuk menghapus dan mengubah transaksi).
   2) **FloatingActionButton** disediakan dua buah:
      - Tombol + untuk menambahkan pemasukan.
      - Tombol - untuk menambahkan pengeluaran.
   3) **Body** terdiri dari dua bagian utama:
      - Kartu Ringkasan Keuangan (_buildSummaryCard), menampilkan total pemasukan, pengeluaran, dan saldo saat ini.
      - Daftar transaksi (_buildGroupedTransactionList) yang ditampilkan berdasarkan bulan dan tahun, dengan real-time update menggunakan StreamBuilder.

2. **Ringkasan Keuangan**

   Di bagian atas halaman, terdapat kartu ringkasan yang menampilkan informasi:

   - Total Pemasukan
   - Total Pengeluaran
   - Saldo Sisa

   Data ini dihitung secara dinamis berdasarkan semua transaksi pengguna yang ada di Firestore.

   Selain itu, terdapat tombol reset saldo, yang memungkinkan pengguna menghapus seluruh transaksi dan memulai periode baru dengan sisa saldo sebelumnya sebagai “Saldo Awal”.

3. **Riwayat Transaksi dan Pengelompokan Bulanan**

   Semua transaksi diambil secara real-time dari Firestore menggunakan StreamBuilder, kemudian:

   - Dikelompokkan berdasarkan bulan dan tahun menggunakan ExpansionTile, agar lebih mudah dinavigasi.
   - Tiap transaksi ditampilkan dengan informasi: deskripsi, tanggal, dan jumlah uang.
   - Warna jumlah uang dibedakan: hijau untuk pemasukan, merah untuk pengeluaran.

4. **Menambah dan Mengedit Transaksi**

   Fungsi _showTransactionDialog() digunakan untuk menampilkan dialog input transaksi, yang bersifat fleksibel:

   1) Dialog ini bisa digunakan untuk **menambah** maupun **mengubah** data transaksi.
   2) Pengguna dapat mengisi:
      - Jumlah uang
      - Keterangan transaksi
      - Tanggal transaksi

   Jika dialog dipanggil dengan parameter transaksi (transaction != null), maka akan dilakukan **pengeditan**, jika tidak maka akan menambahkan data baru.

   Data kemudian dikirim ke Firestore menggunakan metode .add() untuk penambahan dan .update() untuk pengeditan.

5. **Mode Edit Transaksi**

   Dengan menekan ikon edit di AppBar, pengguna masuk ke **mode edit**, di mana:

   - Tombol edit (ikon pensil) dan hapus (ikon tempat sampah) muncul di setiap transaksi.
   - Fungsionalitas ini hanya aktif jika isEditMode == true.

   Mode ini dirancang agar tidak mengganggu pengguna yang hanya ingin melihat ringkasan keuangan tanpa melakukan perubahan.

6. **Menghapus dan Mereset Transaksi**

   Fungsi _deleteTransaction() memungkinkan pengguna menghapus transaksi tertentu dari Firestore.

   Selain itu, terdapat fungsi penting yaitu **_resetBalance()**, yang memungkinkan pengguna:

   1) Menghapus seluruh riwayat transaksi.
   2) Menyimpan saldo akhir sebagai "Saldo Awal" untuk memulai periode keuangan yang baru.
   3) Menampilkan dialog konfirmasi (_showResetConfirmationDialog) sebelum proses reset dilakukan.

7. **Tampilan Saat Tidak Ada Data**

   Jika belum ada transaksi sama sekali, halaman akan menampilkan ikon dan teks "Belum ada transaksi." melalui fungsi _buildEmptyState(). Hal ini memberikan umpan balik yang jelas kepada pengguna bahwa data mereka masih kosong.

BAB III
PENUTUP

3.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian aplikasi Kuliah.ku, dapat disimpulkan bahwa aplikasi yang saya buat berhasil dikembangkan sebagai solusi digital untuk membantu mahasiswa dalam mengelola aktivitas perkuliahan secara terintegrasi. Aplikasi Kuliah.ku menyediakan fitur utama berupa manajemen jadwal kuliah, to-do list, catatan tugas, serta manajemen keuangan yang saling terhubung dalam satu platform.

Pemanfaatan teknologi Firebase Authentication dan Cloud Firestore memungkinkan pengelolaan data pengguna secara aman dan real-time, sehingga pengguna dapat menambah, mengubah, maupun menghapus data dengan mudah. Selain itu, desain antarmuka yang dirancang menggunakan pendekatan UI/UX modern membuat aplikasi lebih ramah pengguna dan mudah dioperasikan.

Secara keseluruhan, aplikasi Kuliah.ku mampu menjawab permasalahan yang sering dihadapi mahasiswa dalam mengatur waktu, tugas, dan keuangan, serta berpotensi menjadi asisten digital yang mendukung efektivitas dan produktivitas mahasiswa dalam menjalani perkuliahan.

3.2 Saran

Untuk pengembangan selanjutnya, aplikasi Kuliah.ku masih memiliki beberapa peluang peningkatan, antara lain:

1. Menambahkan fitur notifikasi dan reminder otomatis untuk jadwal kuliah, deadline tugas, dan pengeluaran agar pengguna lebih disiplin dalam mengelola aktivitasnya.
2. Mengembangkan fitur OTP dan verifikasi akun secara penuh untuk meningkatkan keamanan sistem autentikasi.
3. Menambahkan fitur sinkronisasi kalender dengan Google Calendar atau kalender perangkat pengguna.
4. Mengembangkan fitur statistik dan grafik keuangan agar pengguna dapat memantau kondisi keuangan secara lebih visual.
5. Melakukan pengujian lebih lanjut pada berbagai perangkat dan ukuran layar untuk memastikan aplikasi berjalan optimal dan stabil.

Dengan adanya pengembangan lanjutan tersebut, diharapkan aplikasi Kuliah.ku dapat menjadi lebih lengkap, aman, dan bermanfaat bagi mahasiswa dalam jangka panjang.