Gemini

2.5 Pro

PRO

Percakapan baru

Jelajahi Gem

**Terbaru**

[ 060 ] App By Flutter

@[ 135 ] Industri Kreatif

@API Publik: Deskripsi dan Ide Aplikasi

Keajaiban Angka dalam Matematika

@9 Bahasa

[034] Studi Matematika

50 Virus

[ 110 ] Usaha Pensiunan

@[ 125 ] Bisnis Digital

----[ 013 ] NET Framework

Tokoh Anti-Penindasan dan Penjajahan

[ 050 ] Diktator

[ 100 ] Usaha Kreatif Gen Z

[ 060 ] App By React

@[ 100 ] Lulusan SMA

@[ 60 ] Ide Startup UMKM: Daftar Lengkap

@[ 100 ] Ide Makanan Halal Viral

@[ ::: ] Fitur Gemini

@[ 035 ] Aplikasi Potensial AI

@[:::] TEKNIK KHUSUS GEMINI

@[ ::: ] INTERMEDIATE GEMINI

@[ ::: ] ADVANCED GEMINI

@[:::] BASIC Gemini

Setelan & bantuan

**Percakapan dengan Gemini**

Gemini anda sekarang bertindak sebagai developer aplikasi mobile dengan flutter.

Sebutkan daftar 10 aplikasi mobile yang bisa dibangun dengan flutter dengan tingkat kesulitan yang ringan ( tanpa penjelasan)

Tampilkan alur berpikir

Tentu, ini daftar 10 aplikasi mobile yang bisa dibangun dengan Flutter dengan tingkat kesulitan ringan:

1. Aplikasi To-Do List
2. Aplikasi Kalkulator
3. Aplikasi Cuaca Sederhana
4. Aplikasi Kuis
5. Aplikasi Catatan (Notes)
6. Aplikasi Kalkulator BMI (Indeks Massa Tubuh)
7. Aplikasi Resep Makanan
8. Aplikasi Portofolio Pribadi
9. Aplikasi Penghitung Tip
10. Aplikasi Galeri Foto Sederhana

Sebutkan 10 lagi

Tampilkan alur berpikir

Tentu, ini 10 ide aplikasi selanjutnya:

1. Aplikasi Penggulung Dadu (Dice Roller)
2. Aplikasi Konverter Satuan (misal: suhu, panjang)
3. Aplikasi "Quote of the Day"
4. Aplikasi Pemindai Kode QR
5. Aplikasi Flashcard untuk Belajar
6. Aplikasi Penghasil Password Acak
7. Aplikasi Jam Digital
8. Aplikasi Pembaca Berita dari RSS Feed
9. Aplikasi Papan Gambar Sederhana (Simple Drawing Pad)
10. Aplikasi Pelacak Pengeluaran Harian

Sebutkan daftar 10 aplikasi mobile yang bisa dibangun dengan flutter dengan tingkat kesulitan yang memengah ( tanpa penjelasan)

Tampilkan alur berpikir

Tentu, berikut adalah daftar 10 aplikasi mobile dengan tingkat kesulitan menengah yang bisa dibangun menggunakan Flutter:

1. Aplikasi Chat Sederhana (menggunakan Firebase)
2. Aplikasi E-commerce (Katalog, Keranjang, dan Wishlist)
3. Aplikasi Pelacak Kebiasaan (Habit Tracker) dengan Grafik
4. Aplikasi Media Sosial (Fitur Feed, Like, dan Komentar)
5. Aplikasi Pelacak Film/Serial TV (menggunakan API dari TMDB)
6. Aplikasi Jurnal Harian dengan Sinkronisasi Cloud
7. Aplikasi Pencari Event dengan Integrasi Peta (Google Maps)
8. Aplikasi Pelacak Latihan Fitness
9. Aplikasi Booking Sederhana (misal: booking lapangan)
10. Aplikasi Manajemen Inventaris Barang

Tambahkan 10 lagi

Tampilkan alur berpikir

Tentu, ini 10 lagi ide aplikasi dengan tingkat kesulitan menengah:

1. Aplikasi Pemutar Podcast (dengan fitur download offline)
2. Aplikasi Keuangan Pribadi (dengan pencatatan dan grafik)
3. Aplikasi Papan Proyek Kanban (seperti Trello, dengan drag-and-drop)
4. Aplikasi Pengiriman Makanan (sisi pelanggan dengan pelacakan pesanan)
5. Aplikasi Forum Komunitas berdasarkan Topik
6. Aplikasi Pembelajaran Bahasa (mirip Duolingo sederhana)
7. Aplikasi Berita dengan Mode Offline
8. Aplikasi Transportasi Online (sisi penumpang dengan peta real-time)
9. Aplikasi Penjualan Tiket Acara/Bioskop
10. Aplikasi Klien untuk Penyedia Penyimpanan Cloud (misal: Google Drive/Dropbox)

Sebutkan daftar 10 aplikasi mobile yang bisa dibangun dengan flutter dengan tingkat kesulitan yang sulit ( tanpa penjelasan)

Tampilkan alur berpikir

Anda menghentikan respons ini

Sebutkan daftar 10 aplikasi mobile yang bisa dibangun dengan flutter dengan tingkat kesulitan yang lebih lanjut ( tanpa penjelasan)

Tampilkan alur berpikir

Tentu, ini adalah daftar 10 aplikasi mobile dengan tingkat kesulitan lebih lanjut (sulit) yang bisa dibangun menggunakan Flutter:

1. Aplikasi Penyunting Video Profesional (Video Editor)
2. Aplikasi Produksi Musik (Mobile DAW)
3. Aplikasi Papan Tulis Kolaboratif Real-time (seperti Miro)
4. Aplikasi Penyunting Foto Non-Destruktif (seperti Lightroom Mobile)
5. Aplikasi Komunikasi Tim dengan Panggilan Video (menggunakan WebRTC)
6. Sistem Transportasi Online Lengkap (Aplikasi Pengemudi & Penumpang)
7. Kloning Media Sosial dengan Stories dan Filter Video Real-time
8. Aplikasi Desain Interior dengan Augmented Reality (AR)
9. Aplikasi Kontrol Perangkat Smart Home (IoT) Multi-Protokol
10. Platform E-commerce Lengkap dengan Dasbor Penjual

Lanjutkan 10 lagi

Tampilkan alur berpikir

Tentu, ini 10 ide lanjutan untuk aplikasi dengan tingkat kesulitan yang lebih lanjut:

1. Aplikasi CAD Viewer dan Editor Sederhana (2D/3D)
2. Aplikasi Code Editor dengan Terminal Terintegrasi
3. Aplikasi Simulator Penerbangan
4. Aplikasi Penampil Citra Medis (DICOM Viewer)
5. Dompet Kripto dengan Integrasi Hardware Wallet (via BLE/NFC)
6. Aplikasi Live Streaming Interaktif dengan Filter AR
7. Jejaring Sosial Berbasis Lokasi (Geofenced & Ephemeral Content)
8. Aplikasi Keamanan Mobile dengan Pemindai Jaringan
9. Alat Pembuat Animasi Vektor Interaktif (seperti editor Rive)
10. Platform Trading dengan Fitur Perancangan Algoritma Otomatis

Gabungkan semua dalam nomor urut

Tampilkan alur berpikir

Tentu, berikut adalah gabungan semua daftar ide aplikasi dalam satu nomor urut, dikelompokkan berdasarkan tingkat kesulitan.

**Tingkat Kesulitan: Ringan**

1. Aplikasi To-Do List
2. Aplikasi Kalkulator
3. Aplikasi Cuaca Sederhana
4. Aplikasi Kuis
5. Aplikasi Catatan (Notes)
6. Aplikasi Kalkulator BMI (Indeks Massa Tubuh)
7. Aplikasi Resep Makanan
8. Aplikasi Portofolio Pribadi
9. Aplikasi Penghitung Tip
10. Aplikasi Galeri Foto Sederhana
11. Aplikasi Penggulung Dadu (Dice Roller)
12. Aplikasi Konverter Satuan (misal: suhu, panjang)
13. Aplikasi "Quote of the Day"
14. Aplikasi Pemindai Kode QR
15. Aplikasi Flashcard untuk Belajar
16. Aplikasi Penghasil Password Acak
17. Aplikasi Jam Digital
18. Aplikasi Pembaca Berita dari RSS Feed
19. Aplikasi Papan Gambar Sederhana (Simple Drawing Pad)
20. Aplikasi Pelacak Pengeluaran Harian

**Tingkat Kesulitan: Menengah**

1. Aplikasi Chat Sederhana (menggunakan Firebase)
2. Aplikasi E-commerce (Katalog, Keranjang, dan Wishlist)
3. Aplikasi Pelacak Kebiasaan (Habit Tracker) dengan Grafik
4. Aplikasi Media Sosial (Fitur Feed, Like, dan Komentar)
5. Aplikasi Pelacak Film/Serial TV (menggunakan API dari TMDB)
6. Aplikasi Jurnal Harian dengan Sinkronisasi Cloud
7. Aplikasi Pencari Event dengan Integrasi Peta (Google Maps)
8. Aplikasi Pelacak Latihan Fitness
9. Aplikasi Booking Sederhana (misal: booking lapangan)
10. Aplikasi Manajemen Inventaris Barang
11. Aplikasi Pemutar Podcast (dengan fitur download offline)
12. Aplikasi Keuangan Pribadi (dengan pencatatan dan grafik)
13. Aplikasi Papan Proyek Kanban (seperti Trello, dengan drag-and-drop)
14. Aplikasi Pengiriman Makanan (sisi pelanggan dengan pelacakan pesanan)
15. Aplikasi Forum Komunitas berdasarkan Topik
16. Aplikasi Pembelajaran Bahasa (mirip Duolingo sederhana)
17. Aplikasi Berita dengan Mode Offline
18. Aplikasi Transportasi Online (sisi penumpang dengan peta real-time)
19. Aplikasi Penjualan Tiket Acara/Bioskop
20. Aplikasi Klien untuk Penyedia Penyimpanan Cloud (misal: Google Drive/Dropbox)

**Tingkat Kesulitan: Lanjutan / Sulit**

1. Aplikasi Penyunting Video Profesional (Video Editor)
2. Aplikasi Produksi Musik (Mobile DAW)
3. Aplikasi Papan Tulis Kolaboratif Real-time (seperti Miro)
4. Aplikasi Penyunting Foto Non-Destruktif (seperti Lightroom Mobile)
5. Aplikasi Komunikasi Tim dengan Panggilan Video (menggunakan WebRTC)
6. Sistem Transportasi Online Lengkap (Aplikasi Pengemudi & Penumpang)
7. Kloning Media Sosial dengan Stories dan Filter Video Real-time
8. Aplikasi Desain Interior dengan Augmented Reality (AR)
9. Aplikasi Kontrol Perangkat Smart Home (IoT) Multi-Protokol
10. Platform E-commerce Lengkap dengan Dasbor Penjual
11. Aplikasi CAD Viewer dan Editor Sederhana (2D/3D)
12. Aplikasi Code Editor dengan Terminal Terintegrasi
13. Aplikasi Simulator Penerbangan
14. Aplikasi Penampil Citra Medis (DICOM Viewer)
15. Dompet Kripto dengan Integrasi Hardware Wallet (via BLE/NFC)
16. Aplikasi Live Streaming Interaktif dengan Filter AR
17. Jejaring Sosial Berbasis Lokasi (Geofenced & Ephemeral Content)
18. Aplikasi Keamanan Mobile dengan Pemindai Jaringan
19. Alat Pembuat Animasi Vektor Interaktif (seperti editor Rive)
20. Platform Trading dengan Fitur Perancangan Algoritma Otomatis

Secara bertahap satu persatu dari nomor 1 sampai 57 , buat BraInstorming tiap aplikasi dan tampilan UInya dalam mode karakter .Jangan pindah ke nomor berikutnya sebelum saya meminta yahok

Tampilkan alur berpikir

Tentu, saya siap. Mari kita mulai dari nomor 1. Saya akan menunggu aba-aba Anda untuk melanjutkan ke nomor berikutnya.

**1. Aplikasi To-Do List**

**Brainstorming**

* **Tujuan Utama:** Membuat aplikasi sederhana bagi pengguna untuk mencatat, melacak, dan mengelola daftar tugas mereka.
* **Fitur Inti:**
  1. **Tambah Tugas:** Pengguna bisa memasukkan tugas baru melalui sebuah input field atau dialog.
  2. **Lihat Daftar Tugas:** Semua tugas ditampilkan dalam sebuah daftar yang bisa di-scroll.
  3. **Tandai Selesai:** Setiap tugas memiliki checkbox. Saat dicentang, tugas tersebut secara visual ditandai sebagai selesai (misalnya, dengan coretan pada teks).
  4. **Hapus Tugas:** Setiap tugas memiliki tombol untuk menghapusnya dari daftar.
* **Logika & State:**
  1. Aplikasi akan mengelola sebuah List dari objek Task.
  2. Setiap objek Task memiliki properti seperti title (string) dan isDone (boolean).
  3. Setiap aksi (tambah, centang, hapus) akan memanipulasi List ini dan memperbarui UI menggunakan setState.
* **Tantangan Potensial (Ringan):**
  1. Menjaga state tetap sinkron dengan UI.
  2. Mengelola tampilan AlertDialog atau BottomSheet untuk menambah tugas baru.
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib, bisa dibangun murni dengan widget bawaan Flutter.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Tugas)**

+------------------------------------------+

| My To-Do List [Opsi Sortir]|

+==========================================+

| |

| [x] ~~Beli bahan makanan~~ [🗑️] |

| ---------------------------------------- |

| [ ] Selesaikan brainstorming App #1 [🗑️] |

| ---------------------------------------- |

| [ ] Bayar tagihan listrik [🗑️] |

| ---------------------------------------- |

| [ ] Olahraga 30 menit [🗑️] |

| ---------------------------------------- |

| |

| |

| |

| (+) |

+------------------------------------------+

**Dialog Tambah Tugas (Saat tombol (+) ditekan)**

+------------------------------------------+

| [ Tambah Tugas Baru ] |

| |

| Deskripsi Tugas: |

| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

| |

| ( Batal ) [ Simpan Tugas ] |

+------------------------------------------+

Silakan berikan aba-aba jika sudah siap untuk lanjut ke nomor 2.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**2. Aplikasi Kalkulator**

**Brainstorming**

* **Tujuan Utama:** Membuat kalkulator fungsional dengan operasi aritmatika dasar yang memiliki antarmuka yang familiar bagi pengguna.
* **Fitur Inti:**
  1. **Layar Display:** Area di bagian atas untuk menampilkan angka yang sedang diketik (input) dan hasil perhitungan.
  2. **Tombol Angka (0-9):** Tombol utama untuk input numerik.
  3. **Tombol Operasi:** Tombol untuk operasi tambah (+), kurang (-), kali (\*), dan bagi (/).
  4. **Tombol Fungsi:** Tombol C (Clear) untuk menghapus semua input dan hasil, tombol = (Sama Dengan) untuk mengeksekusi perhitungan.
  5. **Tombol Tambahan:** Tombol desimal (.), tombol hapus satu karakter (<- atau DEL).
* **Logika & State:**
  1. Mengelola state untuk *string input* pengguna dan *string hasil*.
  2. Saat tombol angka atau operator ditekan, string input akan diperbarui.
  3. Saat = ditekan, string input akan di-parse dan dievaluasi untuk menghasilkan output. Ini adalah logika inti dari aplikasi.
  4. Untuk mempermudah, evaluasi ekspresi matematika bisa menggunakan paket math\_expressions.
* **Tantangan Potensial (Ringan):**
  1. Membuat layout grid tombol yang rapi dan responsif menggunakan Column, Row, dan Expanded.
  2. Menangani logika evaluasi string matematika menjadi hasil numerik.
  3. Memastikan UI diperbarui dengan benar setiap kali tombol ditekan.
* **Paket Flutter yang Mungkin Digunakan:** math\_expressions untuk mempermudah proses kalkulasi.

**Tampilan UI (Mode Karakter)**

**Layar Utama Kalkulator**

+------------------------------------+

| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

| | | |

| | Input: 1.234 + 567 | |

| | Hasil: 1799 | |

| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

+====================================+

| |

| ( C ) ( +/- ) ( % ) [ / ] |

| |

| ( 7 ) ( 8 ) ( 9 ) [ \* ] |

| |

| ( 4 ) ( 5 ) ( 6 ) [ - ] |

| |

| ( 1 ) ( 2 ) ( 3 ) [ + ] |

| |

| ( 0 ) ( . ) [ = ] |

| |

+------------------------------------+

*(Keterangan: Tombol operator [] dan = diberi style berbeda untuk membedakannya dari tombol angka ())*

Baik, saya siap untuk nomor 3 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Oke, siap. Mari kita bedah aplikasi berikutnya.

**3. Aplikasi Cuaca Sederhana**

**Brainstorming**

* **Tujuan Utama:** Menampilkan informasi cuaca terkini untuk lokasi pengguna atau lokasi yang dicari, dengan antarmuka yang bersih dan mudah dipahami.
* **Fitur Inti:**
  1. **Tampilan Cuaca Saat Ini:** Menampilkan suhu (°C), kondisi cuaca (misal: "Cerah", "Hujan Ringan"), dan ikon visual yang sesuai.
  2. **Informasi Lokasi:** Menampilkan nama kota dan negara.
  3. **Data dari API:** Mengambil data cuaca secara *real-time* dari layanan API eksternal (contoh: OpenWeatherMap, WeatherAPI).
  4. **Tombol Refresh:** Memungkinkan pengguna memuat ulang data untuk mendapatkan informasi terbaru.
* **Logika & State:**
  1. Aplikasi akan memiliki state untuk menampung data cuaca (objek WeatherModel berisi suhu, kondisi, dll).
  2. Saat aplikasi dimulai atau di-refresh, ia akan memanggil API cuaca. Selama proses ini, UI akan menampilkan indikator Loading....
  3. Setelah data JSON diterima dari API, data tersebut akan di-parsing ke dalam WeatherModel.
  4. UI akan diperbarui (menggunakan FutureBuilder atau setState setelah await) untuk menampilkan data yang sudah di-parsing.
  5. Penanganan error jika API gagal atau tidak ada koneksi internet.
* **Tantangan Potensial (Ringan):**
  1. Melakukan panggilan HTTP ke API eksternal.
  2. Mengelola API Key dengan aman.
  3. Parsing data JSON menjadi objek Dart yang bisa digunakan.
  4. Menangani state asinkron (loading, data, error).
* **Paket Flutter yang Mungkin Digunakan:** http untuk panggilan API, lottie untuk animasi cuaca yang menarik.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Saat Data Berhasil Dimuat)**

+------------------------------------------+

| [ Cari Lokasi... 🔍 ] [ 🔄 ] |

+==========================================+

| |

| 📍 Banda Aceh, Indonesia |

| Sabtu, 21 Juni |

| |

| |

| .--. |

| .-( )-. |

| (\_\_\_.\_\_)\_\_) | <- Ikon Awan & Matahari

| |

| |

| 30°C |

| |

| Cerah Berawan |

| |

| +------------------------------------+ |

| | 💨 12 km/j | 💧 78% | 🌡️ 33°C | | <- Angin | Lembap | Terasa Seperti

| +------------------------------------+ |

| |

+------------------------------------------+

**Layar Utama (Saat Memuat Data)**

+------------------------------------------+

| [ Cari Lokasi... 🔍 ] [ 🔄 ] |

+==========================================+

| |

| |

| |

| |

| |

| Memuat data cuaca... |

| ( / | \ - ) | <- Indikator loading

| |

| |

| |

| |

| |

| |

| |

+------------------------------------------+

Siap melanjutkan ke nomor 4 jika Anda berkenan.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke nomor 4.

**4. Aplikasi Kuis**

**Brainstorming**

* **Tujuan Utama:** Membuat sebuah aplikasi kuis interaktif di mana pengguna bisa menjawab serangkaian pertanyaan pilihan ganda dan mendapatkan skor di akhir.
* **Fitur Inti:**
  1. **Layar Pertanyaan:** Menampilkan satu pertanyaan beserta beberapa pilihan jawaban (biasanya dalam bentuk tombol).
  2. **Mekanisme Jawaban:** Pengguna memilih satu jawaban. Aplikasi memberikan feedback visual instan (misal, tombol menjadi hijau jika benar, merah jika salah).
  3. **Navigasi Otomatis:** Setelah menjawab, aplikasi secara otomatis menampilkan pertanyaan berikutnya.
  4. **Pelacakan Skor:** Aplikasi menghitung jumlah jawaban yang benar.
  5. **Layar Hasil:** Setelah semua pertanyaan selesai, menampilkan skor total pengguna dan opsi untuk mengulang kuis.
* **Logika & State:**
  1. Data kuis (daftar pertanyaan, pilihan jawaban, dan kunci jawaban) disimpan dalam sebuah List di dalam kode.
  2. State utama yang dikelola adalah \_questionIndex (untuk melacak pertanyaan saat ini) dan \_totalScore.
  3. Setiap kali jawaban dipilih, sebuah fungsi akan dipanggil untuk:
     + Memeriksa kebenaran jawaban.
     + Menambah skor jika benar.
     + Menaikkan \_questionIndex.
     + Memperbarui UI (setState) untuk menampilkan pertanyaan baru atau layar hasil jika kuis sudah selesai.
* **Tantangan Potensial (Ringan):**
  1. Mengelola alur aplikasi dari pertanyaan ke pertanyaan hingga ke layar hasil.
  2. Memastikan state (skor dan indeks) diperbarui dengan benar.
  3. Memisahkan data kuis dari logika UI agar lebih rapi.
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib, sangat cocok untuk melatih fundamental setState.

**Tampilan UI (Mode Karakter)**

**Layar Pertanyaan (Saat Kuis Berlangsung)**

+------------------------------------------+

| Kuis Sejarah Dunia ( 3 / 10 )|

+==========================================+

| |

| Di negara manakah Tembok Besar |

| berada? |

| |

| |

| +------------------------------------+ |

| | A. Jepang | |

| +------------------------------------+ |

| |

| +------------------------------------+ |

| | B. India | | <- Jawaban salah (merah)

| +------------------------------------+ |

| |

| +------------------------------------+ |

| | C. Tiongkok | | <- Jawaban benar (hijau)

| +------------------------------------+ |

| |

+------------------------------------------+

**Layar Hasil (Setelah Kuis Selesai)**

+------------------------------------------+

| Selamat! |

+==========================================+

| |

| |

| 🏆 |

| |

| Anda Menyelesaikan Kuis! |

| |

| Skor Anda: |

| |

| 70 |

| (7 dari 10 benar) |

| |

| |

| [ Main Lagi ] |

| |

+------------------------------------------+

Monggo, jika sudah siap kita lanjut ke nomor 5.

Yahok

Tampilkan alur berpikir

Baik, kita masuk ke nomor 5.

**5. Aplikasi Catatan (Notes)**

**Brainstorming**

* **Tujuan Utama:** Menyediakan platform bagi pengguna untuk membuat, melihat, mengedit, dan menghapus catatan digital dengan cepat dan mudah, serta menyimpannya secara lokal.
* **Fitur Inti:**
  1. **Layar Daftar Catatan:** Halaman utama yang menampilkan semua catatan dalam bentuk daftar atau grid. Setiap item menampilkan judul dan mungkin beberapa baris pertama isi catatan.
  2. **Buat & Edit Catatan:** Halaman editor terpisah dengan kolom input untuk judul dan isi catatan (konten). Halaman ini digunakan untuk membuat catatan baru dan mengedit yang sudah ada.
  3. **Simpan Otomatis/Manual:** Opsi untuk menyimpan perubahan secara manual dengan tombol atau secara otomatis saat pengguna kembali ke layar daftar.
  4. **Hapus Catatan:** Fungsi untuk menghapus catatan dari daftar.
  5. **Penyimpanan Lokal (Persistence):** Catatan disimpan di perangkat, sehingga data tidak hilang saat aplikasi ditutup.
* **Logika & State:**
  1. Aplikasi ini memperkenalkan alur **CRUD** (Create, Read, Update, Delete).
  2. Menggunakan navigasi multi-halaman (Master-Detail). Dari layar daftar (master), pengguna menavigasi ke layar editor (detail) untuk melihat atau mengubah catatan.
  3. Data perlu dilewatkan antara layar (misalnya, objek Note yang akan diedit).
  4. Untuk penyimpanan, Hive atau sqflite adalah pilihan yang lebih baik daripada shared\_preferences karena lebih cocok untuk menyimpan daftar objek yang terstruktur.
* **Tantangan Potensial (Ringan):**
  1. Implementasi operasi CRUD penuh pada database lokal.
  2. Mengelola navigasi dan passing data antar layar.
  3. Memastikan daftar di layar utama diperbarui secara otomatis setelah ada perubahan (penambahan, editan, atau penghapusan) di layar detail.
* **Paket Flutter yang Mungkin Digunakan:** hive & hive\_flutter (database NoSQL lokal yang cepat dan mudah), atau sqflite (database SQL). intl untuk memformat tanggal.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Catatan)**

+------------------------------------------+

| CatatanKu [🔍] |

+==========================================+

| |

| +------------------------------------+ |

| | Judul: Rencana Hari Ini | |

| | Isi: Meeting jam 10, lalu... | |

| | Dibuat: 21 Jun 2025, 09:15 | |

| +------------------------------------+ |

| |

| +------------------------------------+ |

| | Judul: Daftar Belanja | |

| | Isi: Kopi Gayo, mie Aceh, telur... | |

| | Dibuat: 20 Jun 2025, 17:30 | |

| +------------------------------------+ |

| |

| (+) |

+------------------------------------------+

**Layar Editor Catatan (Saat membuat atau mengedit catatan)**

+------------------------------------------+

| [<-] Editor Catatan [Simpan ✔️]|

+==========================================+

| |

| Judul: |

| |Rencana Hari Ini\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

| Isi Catatan: |

| |------------------------------------| |

| |Meeting jam 10 pagi dengan tim. | |

| |Setelah itu, lanjut coding | |

| |aplikasi catatan Flutter. | |

| |Jangan lupa makan siang! | |

| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

+------------------------------------------+

Silakan beri tahu saya jika Anda siap untuk melanjutkan ke nomor 6.

Yahok

Tampilkan alur berpikir

Baik, kita gas ke nomor 6.

**6. Aplikasi Kalkulator BMI (Indeks Massa Tubuh)**

**Brainstorming**

* **Tujuan Utama:** Memberikan alat yang mudah digunakan bagi pengguna untuk menghitung Indeks Massa Tubuh (BMI) mereka dan memahami hasilnya berdasarkan kategori standar kesehatan.
* **Fitur Inti:**
  1. **Input Tinggi & Berat:** Menyediakan kolom input atau slider untuk tinggi badan (dalam cm) dan berat badan (dalam kg).
  2. **Tombol Hitung:** Tombol utama untuk memproses kalkulasi.
  3. **Tampilan Hasil:** Menampilkan nilai BMI yang dihitung (misal: 22.5).
  4. **Interpretasi Hasil:** Menampilkan kategori kesehatan berdasarkan nilai BMI (misal: "Berat Badan Normal", "Obesitas", dll.) dengan warna yang sesuai untuk feedback visual.
* **Logika & State:**
  1. State akan menyimpan nilai height, weight, bmiResult (double), dan bmiCategory (string).
  2. Rumus inti: BMI = berat (kg) / (tinggi (m))^2. Perlu ada konversi dari cm ke meter sebelum kalkulasi.
  3. Saat tombol "Hitung" ditekan:
     + Ambil nilai height dan weight.
     + Lakukan validasi (pastikan tidak nol).
     + Hitung BMI.
     + Gunakan struktur if-else if-else untuk menentukan kategori berdasarkan rentang nilai BMI (misal: < 18.5 adalah Kurus, 18.5 - 24.9 adalah Normal, dst.).
     + Perbarui UI (setState) untuk menampilkan hasil dan kategori.
* **Tantangan Potensial (Ringan):**
  1. Mendesain UI yang interaktif dan mudah untuk input, misalnya menggunakan Slider selain TextField.
  2. Mengimplementasikan logika kondisional untuk menentukan kategori dan warna hasil.
  3. Melakukan validasi input agar aplikasi tidak crash karena input yang salah (misal: text atau nol).
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib, latihan yang bagus untuk Form, Slider, dan logika Dart.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Input & Hasil dalam satu layar)**

+------------------------------------------+

| Kalkulator Indeks Massa Tubuh (BMI) |

+==========================================+

| |

| Tinggi Badan (cm) |

| <----O----------------------------> 175 |

| |

| Berat Badan (kg) |

| <---------O-----------------------> 70 |

| |

| |

| +------------------+ |

| | HITUNG BMI | |

| +------------------+ |

| |

| |

| ...................................... |

| : : |

| : Hasil Perhitungan : |

| : : |

| : BMI Anda: : |

| : 22.9 : |

| : : |

| : Kategori: Berat Badan Normal : |

| : : |

| ...................................... |

+------------------------------------------+

**Tampilan Alternatif (Layar Hasil terpisah)**

+------------------------------------------+

| Hasil Analisis BMI Anda |

+==========================================+

| |

| 🎉 |

| |

| Nilai BMI: 22.9 |

| |

| BERAT BADAN NORMAL |

| |

| < Kurus |==== Normal ====| Gemuk | Obes >|

| ^ |

| |

| Tips: Pertahankan gaya hidup sehat! |

| |

| [ Hitung Ulang ] |

+------------------------------------------+

Baik, saya tunggu aba-aba Anda untuk melanjutkan ke nomor 7.

Yahok

Tampilkan alur berpikir

Siap, mari kita lanjutkan.

**7. Aplikasi Resep Makanan**

**Brainstorming**

* **Tujuan Utama:** Menjadi buku resep digital yang memungkinkan pengguna untuk menelusuri berbagai resep masakan, melihat detail bahan, dan mengikuti langkah-langkah pembuatannya.
* **Fitur Inti:**
  1. **Galeri Resep:** Layar utama menampilkan daftar atau grid resep, masing-masing dengan gambar yang menarik dan judul yang jelas.
  2. **Halaman Detail Resep:** Ketika sebuah resep dipilih, pengguna akan dibawa ke halaman detail yang berisi:
     + Gambar besar dari masakan tersebut.
     + Daftar bahan-bahan yang dibutuhkan.
     + Petunjuk pembuatan yang terstruktur (langkah demi langkah).
  3. **Data Statis:** Untuk tingkat kesulitan ringan, semua data resep (termasuk teks dan path gambar) disimpan secara lokal di dalam folder assets aplikasi, biasanya dalam format file JSON.
* **Logika & State:**
  1. Aplikasi ini adalah contoh klasik dari alur **Master-Detail**.
  2. Saat aplikasi dimulai, ia akan memuat data dari file JSON di assets. Data ini di-decode menjadi List objek Recipe.
  3. Layar utama (Master) akan me-render daftar ini menggunakan ListView.builder atau GridView.builder.
  4. Saat pengguna mengetuk sebuah item, aplikasi akan bernavigasi ke layar Detail, sambil mengirimkan objek Recipe yang dipilih sebagai argumen.
  5. Layar Detail kemudian menggunakan data dari objek yang diterima untuk menampilkan informasi resep.
* **Tantangan Potensial (Ringan):**
  1. Membaca dan mem-parsing (decode) file JSON dari aset lokal.
  2. Membuat layout yang menarik untuk kartu resep dan halaman detail.
  3. Mengelola aset gambar dan memastikan mereka ditampilkan dengan benar.
  4. Mengimplementasikan navigasi Master-Detail dengan passing data.
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib. Ini adalah latihan yang sangat baik untuk Layouting, Asset Management, dan Navigation.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Galeri Resep)**

+------------------------------------------+

| Dapur Khas Aceh [🔍] |

+==========================================+

| Pagi Sabtu di Banda Aceh, cari inspirasi|

| masak untuk keluarga? |

| |

| +------------------+ +------------------+ |

| | [Gambar Mie Aceh]| |[Gambar Ayam Tngkp]| |

| | Mie Aceh | | Ayam Tangkap | |

| | ⭐️⭐️⭐️⭐️⭐️ | | ⭐️⭐️⭐️⭐️ | |

| +------------------+ +------------------+ |

| |

| +------------------+ +------------------+ |

| | [Gambar Kuah Plk]| | [Gambar Sie Itk] | |

| | Kuah Pliek U | | Sie Itek | |

| | ⭐️⭐️⭐️⭐️ | | ⭐️⭐️⭐️⭐️⭐️ | |

| +------------------+ +------------------+ |

| |

+------------------------------------------+

**Layar Detail Resep**

+------------------------------------------+

| [<-] Resep Mie Aceh |

+==========================================+

| |

| +------------------------------+ |

| | | |

| | [Foto close-up Mie Aceh] | |

| | | |

| +------------------------------+ |

| |

| Bahan-bahan: |

| - 500 gr mie kuning basah |

| - 200 gr daging sapi/udang |

| - Bumbu halus khas (kapulaga, dll) |

| - Tauge, kol, daun bawang |

| |

| Langkah-langkah: |

| 1. Tumis bumbu halus sampai wangi. |

| 2. Masukkan daging, masak hingga matang|

| 3. Tambahkan sayuran dan mie. |

| 4. Aduk rata, sajikan dengan emping. |

| |

+------------------------------------------+

Saya siap untuk nomor 8 kapan pun Anda siap.

YAHOX

Tampilkan alur berpikir

Siap, kita lanjutkan ke nomor 8.

**8. Aplikasi Portofolio Pribadi**

**Brainstorming**

* **Tujuan Utama:** Membuat sebuah "kartu nama digital" atau CV interaktif yang menampilkan profil, keahlian, pengalaman, dan proyek dari seseorang (misalnya seorang developer, desainer, atau fotografer).
* **Fitur Inti:**
  1. **Halaman Utama (Tentang Saya):** Menampilkan foto profil, nama, jabatan, dan bio singkat yang menarik.
  2. **Halaman Keahlian (Skills):** Daftar keahlian yang dikuasai, bisa disajikan dengan bar persentase atau sekadar daftar ikon/teks.
  3. **Halaman Proyek:** Galeri dari proyek-proyek yang pernah dikerjakan. Setiap item bisa diketuk untuk menampilkan detail lebih lanjut.
  4. **Halaman Kontak:** Informasi kontak seperti email, nomor telepon, dan tautan ke media sosial profesional (GitHub, LinkedIn, Dribbble).
* **Logika & State:**
  1. Aplikasi ini sebagian besar bersifat **statis**. Semua data (teks, path gambar, URL) di-hardcode langsung di dalam kode.
  2. State management yang dibutuhkan sangat minimal, umumnya hanya untuk mengelola halaman yang sedang aktif di BottomNavigationBar atau TabBar.
  3. Logika utamanya adalah navigasi antar halaman dan membuka tautan eksternal.
* **Tantangan Potensial (Ringan):**
  1. **Desain UI/UX:** Tantangan terbesar adalah merancang antarmuka yang profesional, bersih, dan secara visual merepresentasikan identitas pribadi pemiliknya.
  2. **Layouting:** Menggunakan widget seperti Stack, Card, CircleAvatar, dan ListView untuk menciptakan tata letak yang menarik.
  3. **Integrasi Link:** Menggunakan paket untuk membuka URL eksternal (email, website, profil sosial media) di luar aplikasi.
* **Paket Flutter yang Mungkin Digunakan:** url\_launcher untuk membuka URL, font\_awesome\_flutter untuk ikon sosial media yang lengkap.

**Tampilan UI (Mode Karakter)**

**Halaman Utama ("Tentang Saya")**

+------------------------------------------+

| |

| .................... |

| --: :-- |

| --: [Foto Profil] :-- |

| --: :-- |

| .................... |

| |

| Ahmad Zulfikar |

| < Mobile Developer > |

| |

| "Developer asal Banda Aceh yang fokus |

| menciptakan solusi mobile inovatif." |

| |

| |

+==========================================+

| [Profil] [Keahlian] [Proyek] [Kontak] |

| ^^^ |

+------------------------------------------+

**Contoh Tampilan Halaman "Proyek"**

+------------------------------------------+

| Proyek Unggulan |

+==========================================+

| |

| +------------------------------------+ |

| | [Icon] Aplikasi Dapur Aceh | |

| | <Aplikasi resep masakan khas...> | |

| +------------------------------------+ |

| |

| +------------------------------------+ |

| | [Icon] Aplikasi Info Cuaca | |

| | <Menampilkan cuaca real-time...> | |

| +------------------------------------+ |

| |

| +------------------------------------+ |

| | [Icon] Kalkulator BMI | |

| | <Alat bantu cek kesehatan...> | |

| +------------------------------------+ |

| |

+==========================================+

| [Profil] [Keahlian] [Proyek] [Kontak] |

| ^^^ |

+------------------------------------------+

Siap untuk nomor 9 jika Anda juga sudah siap.

Yahok

Tampilkan alur berpikir

Oke, kita hajar nomor 9.

**9. Aplikasi Penghitung Tip**

**Brainstorming**

* **Tujuan Utama:** Membuat alat bantu cepat untuk menghitung jumlah tip dan membagi total tagihan secara adil di antara beberapa orang setelah makan di restoran atau kafe.
* **Fitur Inti:**
  1. **Input Total Tagihan:** Kolom untuk memasukkan jumlah tagihan dari struk.
  2. **Pilih Persentase Tip:** Slider atau tombol untuk memilih persentase tip yang umum (misal: 10%, 15%, 20%).
  3. **Jumlah Pembagi:** Stepper (+/-) atau slider untuk menentukan jumlah orang yang akan patungan.
  4. **Tampilan Hasil Real-time:** Menampilkan rincian biaya secara dinamis setiap kali input diubah:
     + Jumlah tip.
     + Total keseluruhan (tagihan + tip).
     + Jumlah yang harus dibayar per orang.
* **Logika & State:**
  1. State akan menyimpan billAmount, tipPercentage, numberOfPeople, dan variabel-variabel hasil perhitungan.
  2. Logika utama akan dijalankan setiap kali ada perubahan pada salah satu input.
  3. Fungsi kalkulasi akan menghitung total per orang: (billAmount \* (1 + tipPercentage / 100)) / numberOfPeople.
  4. UI diperbarui secara instan menggunakan setState untuk memberikan pengalaman interaktif.
* **Tantangan Potensial (Ringan):**
  1. Membuat UI yang responsif terhadap perubahan input secara real-time.
  2. Memformat angka menjadi format mata uang lokal (misalnya, Rupiah) dengan benar.
  3. Menangani kasus pembagian dengan nol untuk mencegah error.
* **Paket Flutter yang Mungkin Digunakan:** intl untuk memformat angka menjadi format mata uang (misal: "Rp 43.125").

**Tampilan UI (Mode Karakter)**

**Layar Utama Kalkulator**

*(Konteks: Selesai makan sore di sebuah kafe di Banda Aceh)*

+------------------------------------------+

| Tip & Patungan Kalkulator |

+==========================================+

| |

| Masukkan Total Tagihan: |

| Rp | 165.000,-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

|------------------------------------------|

| |

| Persentase Tip: |

| <----------O--------------------> 15% |

| |

| Bagi untuk: |

| ( - ) 4 orang ( + ) |

| |

+==========================================+

| |

| BIAYA PER ORANG |

| |

| Rp 47.438,- |

| |

|------------------------------------------|

| Rincian: |

| Tagihan Awal : Rp 165.000 |

| Tip (15%) : Rp 24.750 |

| Total Bayar : Rp 189.750 |

+------------------------------------------+

Saya siap untuk nomor 10, yang terakhir untuk kategori ringan.

Yahok

Tampilkan alur berpikir

Oke, ini dia yang terakhir dari kategori ringan.

**10. Aplikasi Galeri Foto Sederhana**

**Brainstorming**

* **Tujuan Utama:** Menampilkan sekumpulan gambar yang tersimpan di dalam aplikasi dalam sebuah galeri yang rapi dan memungkinkan pengguna untuk melihat setiap gambar secara penuh.
* **Fitur Inti:**
  1. **Tampilan Grid:** Layar utama menampilkan semua gambar dalam format GridView yang ringkas.
  2. **Tampilan Layar Penuh:** Saat sebuah thumbnail gambar di-tap, gambar tersebut akan terbuka dalam layar penuh.
  3. **Navigasi Geser (Swipe):** Di layar penuh, pengguna dapat menggeser ke kiri atau kanan untuk bernavigasi ke gambar selanjutnya atau sebelumnya dalam koleksi.
  4. **Animasi Transisi:** Menggunakan Hero widget untuk menciptakan animasi yang mulus saat gambar "terbang" dari grid ke posisi layar penuh.
* **Logika & State:**
  1. Data utama adalah sebuah List yang berisi path ke file gambar di dalam folder assets.
  2. Layar utama menggunakan GridView.builder untuk menampilkan gambar-gambar ini.
  3. Setiap gambar di dalam grid dibungkus dengan Hero widget yang memiliki *tag* unik (misalnya, path gambarnya).
  4. Saat di-tap, aplikasi bernavigasi ke layar detail baru, mengirimkan List gambar dan index gambar yang dipilih.
  5. Layar detail menggunakan PageView untuk memungkinkan navigasi geser dan juga memiliki Hero widget dengan *tag* yang sama untuk menciptakan animasi.
* **Tantangan Potensial (Ringan):**
  1. Mengimplementasikan GridView.
  2. Menyiapkan PageView untuk galeri geser.
  3. Mengonfigurasi Hero animation dengan benar agar transisinya mulus.
  4. Mengelola aset gambar di dalam proyek.
* **Paket Flutter yang Mungkin Digunakan:** photo\_view bisa ditambahkan jika ingin ada fitur *pinch-to-zoom* pada gambar di layar penuh.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Grid Galeri)**

*(Konteks: Album foto dari liburan keliling Banda Aceh)*

+------------------------------------------+

| Album Kenangan Aceh 📸 |

+==========================================+

| +----------------+ +----------------+ |

| | [Foto Masjid ] | | [Foto Pantai ] | |

| | Baiturrahman | | Lampuuk | |

| +----------------+ +----------------+ |

| |

| +----------------+ +----------------+ |

| | [Foto Museum ] | | [Kapal di atas] | |

| | Tsunami | | rumah | |

| +----------------+ +----------------+ |

| |

| +----------------+ +----------------+ |

| | [Foto Kopi ] | | [Foto Pulau ] | |

| | Solong | | Weh | |

| +----------------+ +----------------+ |

+------------------------------------------+

**Layar Penuh (Saat salah satu gambar di-tap)**

+------------------------------------------+

| |

| <------------ geser ------------> |

| |

| +------------------------------+ |

| | | |

| | [Gambar Masjid Baiturrahman]| |

| | dalam ukuran penuh | |

| | | |

| +------------------------------+ |

| |

| |

| |

| [ 1 dari 6 ] |

| |

+------------------------------------------+

Ini adalah akhir dari kategori ringan. Beri tahu saya jika Anda siap untuk memulai kategori menengah dengan nomor 11.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan. Sebagai koreksi, kita masih dalam kategori **Ringan**. Kategori Menengah akan dimulai dari nomor 21.

Mari kita mulai dengan nomor 11.

**11. Aplikasi Penggulung Dadu (Dice Roller)**

**Brainstorming**

* **Tujuan Utama:** Membuat aplikasi sederhana yang meniru aksi menggulung satu atau dua buah dadu secara acak.
* **Fitur Inti:**
  1. **Tampilan Dadu:** Menampilkan satu atau dua gambar dadu di tengah layar.
  2. **Tombol Aksi:** Sebuah tombol besar "Gulung Dadu" atau "Roll".
  3. **Logika Acak:** Saat tombol ditekan, gambar dadu berubah secara acak untuk menampilkan sisi yang berbeda (angka 1 sampai 6).
  4. **Tampilan Hasil (Opsional):** Jika menggunakan dua dadu, tampilkan total angka dari kedua dadu.
* **Logika & State:**
  1. State yang perlu dikelola sangat sederhana: satu atau dua variabel integer untuk menyimpan hasil dari setiap dadu, misal diceOneResult dan diceTwoResult.
  2. Ketika tombol ditekan, sebuah fungsi akan menghasilkan angka acak antara 1 dan 6 untuk setiap dadu. Ini bisa dicapai dengan Random().nextInt(6) + 1.
  3. Nilai integer hasil acak ini kemudian digunakan untuk menentukan gambar mana yang akan ditampilkan (misal, jika hasilnya 4, tampilkan dadu\_4.png).
  4. UI diperbarui menggunakan setState.
* **Tantangan Potensial (Ringan):**
  1. Menggunakan kelas Random di Dart untuk menghasilkan angka acak.
  2. Mengelola aset gambar untuk setiap sisi dadu (6 gambar per dadu).
  3. Memetakan nilai integer ke path aset gambar yang sesuai.
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib. Ini adalah proyek klasik untuk melatih fundamental setState, Image.asset, dan logika dasar.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Versi Dua Dadu)**

*(Konteks: Untuk menentukan siapa yang traktir kopi sore di Banda Aceh)*

+------------------------------------------+

| Ayo Gulung Dadu! |

+==========================================+

| |

| |

| +----------------+ +----------------+ |

| | o o | | | |

| | o | | o | |

| | o o | | | |

| +----------------+ +----------------+ |

| (Dadu 5) (Dadu 1) |

| |

| |

| Total Poin: 6 |

| |

| |

| +----------------------+ |

| | GULUNG LAGI! | |

| +----------------------+ |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 12 jika sudah siap.

Yahok

Tampilkan alur berpikir

Oke, kita lanjutkan.

**12. Aplikasi Konverter Satuan (Unit Converter)**

**Brainstorming**

* **Tujuan Utama:** Membuat alat bantu praktis untuk mengonversi nilai antara berbagai satuan pengukuran (seperti panjang, berat, suhu) dengan cepat.
* **Fitur Inti:**
  1. **Input Nilai:** Satu kolom input untuk memasukkan angka yang akan dikonversi.
  2. **Pemilihan Satuan:** Dua menu Dropdown untuk memilih satuan asal ("Dari") dan satuan tujuan ("Ke").
  3. **Hasil Real-time:** Hasil konversi ditampilkan secara otomatis setiap kali nilai input atau pilihan satuan diubah.
  4. **Kategori Konversi:** Pengguna bisa memilih jenis konversi, misalnya antara Suhu (Celcius, Fahrenheit), Panjang (Meter, Kaki), atau Berat (Kilogram, Pon).
* **Logika & State:**
  1. State akan menyimpan nilai input, unit asal yang dipilih, unit tujuan yang dipilih, dan hasil konversi.
  2. Inti dari aplikasi adalah serangkaian **rumus matematika** untuk setiap kemungkinan konversi. Ini bisa diorganisir dalam sebuah Map atau fungsi-fungsi terpisah.
  3. Misalnya, untuk suhu: Fahrenheit = (Celcius \* 9/5) + 32.
  4. Setiap perubahan pada input akan memicu fungsi kalkulasi ulang dan memperbarui UI dengan setState.
* **Tantangan Potensial (Ringan):**
  1. Mengelola state dari beberapa DropdownButton.
  2. Membuat struktur data yang logis untuk menyimpan rumus-rumus konversi.
  3. Memastikan kalkulasi berjalan akurat dan UI diperbarui secara instan.
* **Paket Flutter yang Mungkin Digunakan:** Tidak ada yang wajib, ini adalah latihan yang bagus untuk mengelola input pengguna dan menerapkan logika matematika di Dart.

**Tampilan UI (Mode Karakter)**

**Layar Utama Konverter**

+------------------------------------------+

| Konverter Satuan Cepat |

+==========================================+

| [ Suhu ] ( Panjang ) ( Berat ) |

+------------------------------------------+

| |

| Masukkan nilai untuk dikonversi: |

| |

| +------------------------------------+ |

| | 25 [v] | | <- Nilai & Dropdown Satuan Asal

| | Celcius | |

| +------------------------------------+ |

| |

| ⇅ |

| |

| +------------------------------------+ |

| | 77 [v] | | <- Hasil & Dropdown Satuan Tujuan

| | Fahrenheit | |

| +------------------------------------+ |

| |

+------------------------------------------+

Siap untuk nomor 13 jika Anda berkenan.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**13. Aplikasi "Quote of the Day"**

**Brainstorming**

* **Tujuan Utama:** Menampilkan kutipan (quote) yang inspiratif, motivasional, atau bijak kepada pengguna, bisa berganti setiap hari atau setiap kali aplikasi dibuka.
* **Fitur Inti:**
  1. **Tampilan Kutipan:** Area utama layar menampilkan teks kutipan dengan tipografi yang menarik.
  2. **Tampilan Penulis:** Menampilkan nama penulis atau sumber kutipan di bawah teks kutipan.
  3. **Sumber Data:**
     + **Versi Statis:** Kutipan diambil dari daftar (List) yang sudah disiapkan di dalam kode aplikasi.
     + **Versi API:** Kutipan diambil dari API online gratis (misal: quotable.io).
  4. **Tombol Aksi:** Tombol "Refresh" untuk memuat kutipan baru dan tombol "Bagikan" untuk menyalin atau mengirim kutipan ke aplikasi lain.
* **Logika & State:**
  1. State utama adalah objek Quote yang berisi text dan author.
  2. Jika menggunakan data statis, logika saat refresh adalah memilih item acak dari List.
  3. Jika menggunakan API, logika saat refresh adalah memanggil ulang endpoint API dan mem-parsing respons JSON, mirip dengan aplikasi cuaca.
  4. Fungsi "Bagikan" akan menggunakan paket eksternal untuk memicu dialog *sharing* bawaan OS.
* **Tantangan Potensial (Ringan):**
  1. Desain UI yang berfokus pada tipografi dan estetika minimalis.
  2. Mengimplementasikan fungsi "Bagikan" menggunakan paket share\_plus.
  3. Jika menggunakan API, menangani state *loading* dan *error*.
* **Paket Flutter yang Mungkin Digunakan:** share\_plus untuk fitur berbagi, http jika mengambil data dari API.

**Tampilan UI (Mode Karakter)**

**Layar Utama Aplikasi**

*(Konteks: Sebuah pengingat di sore hari yang tenang di Banda Aceh)*

+------------------------------------------+

| Kata Mutiara Hari Ini ✨ |

+==========================================+

| |

| |

| |

| |

| "Orang yang berilmu dan beradab |

| tidak akan diam di kampung halaman. |

| Tinggalkan negerimu dan merantaulah |

| ke negeri orang." |

| |

| - Imam Syafi'i |

| |

| |

| |

| |

| +----------------+ +----------------+ |

| | [Refresh 🔄] | | [Bagikan ✉️] | |

| +----------------+ +----------------+ |

+------------------------------------------+

Kita bisa lanjut ke nomor 14 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke nomor 14.

**14. Aplikasi Pemindai Kode QR (QR Code Scanner)**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah utilitas yang menggunakan kamera ponsel untuk secara cepat memindai dan mengekstrak informasi dari Kode QR.
* **Fitur Inti**:
  1. **Tampilan Kamera Langsung**: Aplikasi membuka kamera dan menampilkannya sebagai latar belakang utama.
  2. **Deteksi Otomatis**: Secara aktif mencari dan mendeteksi Kode QR dalam bingkai kamera.
  3. **Menampilkan Hasil**: Setelah kode terdeteksi, menampilkan data yang diekstrak (bisa berupa URL, teks, info kontak, dll.) dalam sebuah dialog atau halaman baru.
  4. **Aksi Kontekstual**: Menyediakan tombol aksi yang relevan dengan data. Jika data adalah URL, sediakan tombol "Buka Browser". Jika teks, sediakan "Salin ke Clipboard".
* **Logika & State**:
  1. Fungsionalitas inti dari pemindaian kamera dan dekode QR hampir sepenuhnya ditangani oleh **paket eksternal**. Tugas developer adalah mengintegrasikan paket ini.
  2. State utama yang dikelola adalah scannedData (string).
  3. Paket akan menyediakan sebuah *stream* atau *callback* yang akan memberikan data saat kode berhasil dipindai.
  4. Aplikasi akan "mendengarkan" *stream* ini. Ketika data diterima, aplikasi akan memicu setState untuk menampilkan hasil kepada pengguna.
* **Tantangan Potensial (Ringan)**:
  1. **Integrasi Paket**: Menambahkan dependensi paket (seperti mobile\_scanner) dan melakukan konfigurasi izin platform yang diperlukan (izin akses kamera di Android AndroidManifest.xml dan iOS Info.plist).
  2. **Menangani Hasil**: Membuat UI untuk menampilkan hasil pindai dan logika untuk menentukan aksi kontekstual yang sesuai.
  3. Membuat UI overlay (seperti bingkai target) di atas tampilan kamera.
* **Paket Flutter yang Mungkin Digunakan**: **mobile\_scanner** (pilihan modern dan sangat direkomendasikan), url\_launcher (untuk membuka link), clipboard (untuk menyalin teks).

**Tampilan UI (Mode Karakter)**

**Layar Pemindaian (Saat Kamera Aktif)**

+------------------------------------------+

| Arahkan ke Kode QR |

+==========================================+

| |

| |

| +----------------------------+ |

| | | |

| | [ Tampilan Kamera ] | |

| | | |

| | <----------------------> | |

| | | |

| | | |

| +----------------------------+ |

| (Area Pindai) |

| |

| [Hidupkan Flash 🔦] |

+------------------------------------------+

**Dialog Hasil (Setelah Kode Berhasil Dipindai)**

*(Konteks: Memindai QR di sebuah situs wisata di Banda Aceh)*

+------------------------------------------+

| |

| +--------------------------+ |

| | ✅ Kode Ditemukan | |

| |--------------------------| |

| | | |

| | Tipe: URL | |

| | Data: | |

| | https://disbudpar.acehprov.go.id | |

| | | |

| | | |

| | [ Buka Link ] ( Salin ) | |

| +--------------------------+ |

| |

+------------------------------------------+

Monggo, kita lanjut ke nomor 15 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**15. Aplikasi Flashcard untuk Belajar**

**Brainstorming**

* **Tujuan Utama**: Menyediakan alat bantu belajar berbasis kartu (flashcard) untuk membantu pengguna menghafal informasi seperti kosakata, istilah, definisi, atau rumus.
* **Fitur Inti**:
  1. **Tampilan Kartu**: Menampilkan satu kartu di layar. Satu sisi menampilkan pertanyaan atau istilah, sisi lain menampilkan jawaban atau definisi.
  2. **Aksi Balik Kartu**: Pengguna bisa mengetuk kartu untuk "membaliknya" dan melihat sisi seberangnya.
  3. **Navigasi Antar Kartu**: Tombol "Berikutnya" dan "Sebelumnya" untuk berpindah-pindah kartu dalam satu tumpukan (deck).
  4. **Data Statis**: Untuk versi ringan, satu atau beberapa tumpukan kartu sudah disiapkan di dalam kode aplikasi.
* **Logika & State:**
  1. Data utama adalah List dari objek Flashcard, di mana setiap objek memiliki properti seperti question dan answer.
  2. State yang perlu dikelola: cardIndex (indeks kartu yang sedang ditampilkan) dan isFlipped (boolean untuk menentukan sisi kartu yang terlihat).
  3. Saat kartu diketuk, isFlipped dibalik nilainya (!isFlipped).
  4. Saat tombol navigasi ditekan, cardIndex diubah, dan isFlipped direset ke false.
  5. UI diperbarui menggunakan setState untuk mencerminkan perubahan ini.
* **Tantangan Potensial (Ringan)**:
  1. Mengimplementasikan animasi saat kartu dibalik. Ini bisa dicapai dengan widget AnimatedSwitcher atau Transform.
  2. Mengelola state isFlipped agar tidak tercampur antar kartu.
  3. Membuat layout kartu yang menarik dan mudah dibaca.
* **Paket Flutter yang Mungkin Digunakan**: Tidak ada yang wajib. Proyek ini sangat bagus untuk melatih animasi dasar dan manajemen state.

**Tampilan UI (Mode Karakter)**

**Tampilan Kartu (Sisi Depan)**

*(Konteks: Seorang mahasiswa UIN Ar-Raniry sedang menghafal kosakata Bahasa Arab)*

+------------------------------------------+

| Deck: Kosa Kata Dasar B. Arab |

+==========================================+

| ( 7 / 50 ) |

| |

| +----------------------------+ |

| | | |

| | | |

| | مدرسة | |

| | (Madrasatun) | |

| | | |

| | | |

| +----------------------------+ |

| < Ketuk untuk melihat arti > |

| |

| [ << Kembali ] [ Lanjut >> ] |

+------------------------------------------+

**Tampilan Kartu (Sisi Belakang, Setelah Diketuk)**

+------------------------------------------+

| Deck: Kosa Kata Dasar B. Arab |

+==========================================+

| ( 7 / 50 ) |

| |

| +----------------------------+ |

| | | |

| | | |

| | SEKOLAH | |

| | (School) | |

| | | |

| | | |

| +----------------------------+ |

| < Ketuk untuk kembali > |

| |

| [ << Kembali ] [ Lanjut >> ] |

+------------------------------------------+

Kita lanjutkan ke nomor 16 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**16. Aplikasi Penghasil Password Acak**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah alat bantu keamanan untuk menghasilkan kata sandi (password) yang kuat, acak, dan sulit ditebak sesuai dengan preferensi pengguna.
* **Fitur Inti**:
  1. **Pengaturan Panjang**: Sebuah Slider untuk memungkinkan pengguna memilih panjang kata sandi (misal: 8-32 karakter).
  2. **Opsi Karakter**: Serangkaian Checkbox atau Switch untuk memilih jenis karakter yang akan disertakan: Huruf Besar (ABC), Huruf Kecil (abc), Angka (123), dan Simbol (@#$).
  3. **Tombol Generate**: Tombol utama untuk memicu pembuatan kata sandi baru.
  4. **Tampilan Hasil**: Sebuah area teks yang menampilkan kata sandi yang baru saja dibuat.
  5. **Tombol Salin (Copy)**: Tombol untuk menyalin kata sandi ke clipboard perangkat dengan sekali sentuh.
* **Logika & State**:
  1. State akan menyimpan semua opsi yang dipilih pengguna (length, includeUppercase, includeSymbols, dll.) dan generatedPassword.
  2. Saat tombol "Generate" ditekan:
     + Sebuah String berisi kumpulan karakter yang valid akan dibuat berdasarkan opsi yang aktif.
     + Sebuah loop akan berjalan sebanyak length yang ditentukan.
     + Di setiap iterasi, satu karakter acak akan dipilih dari kumpulan karakter tersebut dan ditambahkan ke string password.
     + Hasil akhir ditampilkan di UI menggunakan setState.
* **Tantangan Potensial (Ringan)**:
  1. Menyusun logika untuk membangun set karakter secara dinamis.
  2. Mengimplementasikan fungsi "Salin ke Clipboard".
  3. Memastikan setidaknya satu opsi karakter dipilih untuk menghindari error.
* **Paket Flutter yang Mungkin Digunakan**: clipboard untuk fungsionalitas salin.

**Tampilan UI (Mode Karakter)**

**Layar Utama Aplikasi**

+------------------------------------------+

| Generator Password Aman 🔐 |

+==========================================+

| |

| Panjang Karakter: |

| <------------------O------------> 20 |

| |

| Sertakan: |

| [✔️] Huruf Besar (A-Z) |

| [✔️] Huruf Kecil (a-z) |

| [✔️] Angka (0-9) |

| [✔️] Simbol (!@#$%) |

| |

| +--------------------+ |

| | BUAT PASSWORD | |

| +--------------------+ |

| |

| +--------------------------------------+ |

| | | |

| | jR8$zL#bV@kF4!pG7wNq | |

| | | |

| +---------------------------[Salin 📋]--+ |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 17 jika sudah siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**17. Aplikasi Jam Digital**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi layar penuh yang berfungsi sebagai jam meja digital, menampilkan waktu saat ini dengan jelas dan terus diperbarui.
* **Fitur Inti**:
  1. **Tampilan Waktu (HH:MM:SS)**: Menampilkan jam, menit, dan detik dalam format digital yang besar.
  2. **Tampilan Tanggal**: Menampilkan informasi tanggal, hari, bulan, dan tahun.
  3. **Pembaruan Otomatis**: Waktu di layar harus diperbarui setiap detik tanpa intervensi pengguna.
* **Logika & State**:
  1. State utama adalah objek DateTime yang akan menyimpan waktu terkini.
  2. Kunci dari aplikasi ini adalah penggunaan **Timer.periodic**. Sebuah Timer akan diatur untuk berjalan setiap detik (Duration(seconds: 1)).
  3. Setiap kali timer berjalan, ia akan memanggil setState untuk mendapatkan DateTime.now() dan membangun ulang UI dengan waktu yang baru.
  4. Sangat penting untuk mengelola *lifecycle* dari Timer:
     + Inisialisasi Timer di dalam metode initState().
     + Membatalkan Timer (timer.cancel()) di dalam metode dispose() untuk mencegah *memory leak*.
* **Tantangan Potensial (Ringan)**:
  1. Mengimplementasikan Timer.periodic dengan benar.
  2. Mengelola *lifecycle* dari StatefulWidget (initState dan dispose).
  3. Memformat objek DateTime menjadi string yang mudah dibaca (misalnya, menggunakan paket intl).
* **Paket Flutter yang Mungkin Digunakan**: intl untuk memformat tanggal dan waktu secara advanced (misal: nama hari dan bulan dalam Bahasa Indonesia).

**Tampilan UI (Mode Karakter)**

**Layar Utama Aplikasi**

*(Menampilkan waktu sesuai konteks saat ini)*

+------------------------------------------+

| |

| |

| |

| |

| |

| 16:14:12 |

| |

| |

| Selasa, 1 Juli 2025 |

| |

| |

| 📍 Banda Aceh (WIB) |

| |

| |

| |

| |

| (Sentuh untuk ganti warna tema) |

+------------------------------------------+

Kita lanjutkan ke nomor 18 jika Anda mengizinkan.

Yahok

Tampilkan alur berpikir

Oke, kita lanjutkan.

**18. Aplikasi Pembaca Berita dari RSS Feed**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah agregator berita sederhana yang mengambil dan menampilkan artikel terbaru dari satu atau lebih sumber berita melalui RSS (Really Simple Syndication) feed mereka.
* **Fitur Inti**:
  1. **Ambil Data Feed**: Melakukan panggilan jaringan untuk mengambil data XML dari URL RSS feed.
  2. **Tampilkan Daftar Berita**: Menampilkan daftar judul berita, mungkin beserta tanggal publikasi atau ringkasan singkat, dalam sebuah ListView.
  3. **Buka Artikel Lengkap**: Saat item berita diketuk, aplikasi akan membuka link artikel asli di browser web eksternal.
* **Logika & State**:
  1. Mirip dengan aplikasi cuaca, ini adalah aplikasi yang bergantung pada **panggilan jaringan** dan **parsing data**.
  2. Perbedaan utamanya adalah format data. RSS menggunakan format **XML**, bukan JSON. Oleh karena itu, diperlukan parser yang berbeda.
  3. State akan menyimpan List dari objek berita yang sudah di-parsing. Setiap objek bisa berisi title, link, pubDate, dll.
  4. FutureBuilder adalah widget yang ideal untuk menangani siklus hidup permintaan data: memuat, menampilkan data, atau menampilkan error.
* **Tantangan Potensial (Ringan)**:
  1. Menemukan paket yang andal untuk mem-parsing RSS feed (format XML).
  2. Menggunakan url\_launcher untuk membuka tautan eksternal.
  3. Menampilkan daftar yang diambil dari sumber data asinkron.
* **Paket Flutter yang Mungkin Digunakan**: **webfeed** (pilihan populer untuk parsing RSS & Atom feed), **url\_launcher** (untuk membuka link artikel), http (untuk request jaringan).

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Berita)**

*(Konteks: Mengambil berita dari RSS feed portal berita lokal)*

+------------------------------------------+

| Kabar Aceh Terkini [🔄] |

+==========================================+

| |

| +-------------------------------------+ |

| | Judul: Pemprov Aceh Tingkatkan | |

| | Promosi Wisata Sabang | |

| | Sumber: Antara News Aceh | |

| | <Selasa, 1 Juli 2025, 16:00> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | Judul: Festival Kopi Gayo Kembali | |

| | Digelar Bulan Depan | |

| | Sumber: Serambi News | |

| | <Selasa, 1 Juli 2025, 14:30> | |

| +-------------------------------------+ |

| |

| (Saat item diketuk, artikel asli akan |

| terbuka di browser) |

+------------------------------------------+

Monggo, kita lanjutkan ke nomor 19 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke nomor 19.

**19. Aplikasi Papan Gambar Sederhana (Simple Drawing Pad)**

**Brainstorming**

* **Tujuan Utama**: Menyediakan sebuah kanvas digital kosong bagi pengguna untuk menggambar atau membuat coretan bebas menggunakan jari mereka.
* **Fitur Inti**:
  1. **Kanvas Gambar**: Area layar utama yang berfungsi sebagai kanvas.
  2. **Deteksi Gerakan**: Melacak pergerakan jari pengguna di atas kanvas untuk membuat goresan.
  3. **Kontrol Kuas (Brush)**:
     + **Pilih Warna:** Palet sederhana untuk mengubah warna goresan.
     + **Ubah Ukuran:** Slider atau beberapa tombol untuk mengubah ketebalan garis.
  4. **Tombol Hapus (Clear)**: Tombol untuk membersihkan seluruh kanvas dan memulai dari awal.
* **Logika & State**:
  1. Ini adalah pengenalan yang bagus untuk GestureDetector dan CustomPainter.
  2. GestureDetector akan membungkus kanvas untuk menangkap input sentuhan (onPanStart, onPanUpdate, onPanEnd).
  3. State akan menyimpan sebuah List dari goresan. Setiap goresan adalah sebuah objek yang berisi List titik (Offset) beserta color dan strokeWidth-nya.
  4. CustomPainter: Sebuah widget kustom yang akan mengambil List goresan tersebut dan menggambarnya ke Canvas menggunakan metode paint().
  5. Setiap kali jari pengguna bergerak (onPanUpdate), titik baru ditambahkan ke goresan saat ini, dan setState dipanggil untuk memicu CustomPainter agar menggambar ulang kanvas dengan goresan yang diperbarui.
* **Tantangan Potensial (Ringan ke Menengah)**:
  1. Memahami dan mengimplementasikan CustomPainter untuk pertama kalinya.
  2. Menangkap dan mengelola aliran titik dari GestureDetector.
  3. Menyusun struktur data yang efisien untuk menyimpan semua goresan.
* **Paket Flutter yang Mungkin Digunakan**: flutter\_colorpicker bisa digunakan untuk menyediakan dialog pemilih warna yang lebih canggih, namun tidak wajib untuk versi sederhana.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Kanvas dan Toolbar)**

+------------------------------------------+

| Kanvas Kreasiku |

+==========================================+

| |

| |

| /| |\ |

| / | | \ |

| / | | \ O\_o |

| / | | \ ( .\_.) | <- Area Kanvas

| / |\_| \ > ^ < |

| |

| |

| |

| |

+------------------------------------------+

| Warna: [⚫] | Ukuran: [•] | [Hapus 🗑️] |

+------------------------------------------+

**Dialog Opsi (Saat "Warna" atau "Ukuran" ditekan)**

+------------------------------------------+

| [ Pilih Ukuran Kuas ] |

| |

| . • ● ⬤ |

| |

| [ Tutup ] |

+------------------------------------------+

Siap untuk nomor 20, yang terakhir dari kategori ringan, jika Anda berkenan.

Yahok

Tampilkan alur berpikir

Oke, kita tuntaskan kategori ringan dengan nomor 20.

**20. Aplikasi Pelacak Pengeluaran Harian**

**Brainstorming**

* **Tujuan Utama**: Membantu pengguna mencatat dan memantau pengeluaran harian mereka dengan cara yang sederhana dan cepat.
* **Fitur Inti**:
  1. **Input Pengeluaran**: Sebuah form atau dialog untuk memasukkan deskripsi pengeluaran (misal: "Makan Siang") dan jumlah nominalnya.
  2. **Daftar Transaksi**: Menampilkan daftar semua pengeluaran yang telah dicatat pada hari itu.
  3. **Total Harian**: Menampilkan jumlah total dari semua pengeluaran yang tercatat untuk hari ini.
  4. **Penyimpanan Lokal**: Data pengeluaran disimpan di perangkat agar tidak hilang.
* **Logika & State**:
  1. Aplikasi ini adalah implementasi **CRUD** (Create, Read, Update, Delete) sederhana, mirip dengan aplikasi To-Do List atau Catatan.
  2. State akan menyimpan List dari objek Expense, yang masing-masing berisi description, amount, dan timestamp.
  3. Saat pengguna menambahkan item baru, objek Expense akan dibuat dan ditambahkan ke List.
  4. UI akan diperbarui untuk menampilkan item baru di daftar dan menghitung ulang total pengeluaran harian.
  5. Database lokal seperti Hive atau sqflite cocok untuk menyimpan data pengeluaran.
* **Tantangan Potensial (Ringan)**:
  1. Mengimplementasikan fungsi CRUD.
  2. Memfilter daftar untuk hanya menampilkan transaksi hari ini.
  3. Memformat angka menjadi format mata uang (IDR) menggunakan paket intl.
  4. Menghitung total dari sebuah List secara dinamis.
* **Paket Flutter yang Mungkin Digunakan**: hive atau sqflite untuk database, intl untuk format angka dan tanggal.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Laporan Harian)**

*(Konteks: Mencatat pengeluaran sehari-hari di Banda Aceh)*

+------------------------------------------+

| DompetKu - Selasa, 1 Juli 2025 |

+==========================================+

| |

| Total Pengeluaran Hari Ini: |

| |

| Rp 57.000 |

| |

|------------------------------------------|

| Rincian: |

| |

| - Nasi campur di warung ... Rp 20.000 |

| - Kopi sanger sore ini Rp 15.000 |

| - Isi bensin pertalite Rp 10.000 |

| - Ongkos labi-labi pulang Rp 5.000 |

| - Parkir Rp 2.000 |

| - Kerupuk Rp 5.000 |

| |

| (+) |

+------------------------------------------+

**Dialog Tambah Pengeluaran**

+------------------------------------------+

| [ Catat Pengeluaran Baru ] |

| |

| Deskripsi: |

| |Beli martabak\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

| Jumlah (Rp): |

| |25000\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| |

| ( Batal ) [ Simpan ] |

+------------------------------------------+

Ini adalah akhir dari kategori ringan. Beri tahu saya jika Anda siap untuk naik tingkat ke kategori menengah, dimulai dengan nomor 21.

Yahok

Tampilkan alur berpikir

Baik, kita naik tingkat. Selamat datang di kategori **Menengah**. Aplikasi di sini akan melibatkan konsep yang lebih kompleks seperti backend, database, dan manajemen state yang lebih canggih.

Mari kita mulai dengan nomor 21.

**21. Aplikasi Chat Sederhana (dengan Firebase)**

**Brainstorming**

* **Tujuan Utama**: Membangun aplikasi pesan instan fungsional di mana pengguna bisa login, memilih kontak, dan bertukar pesan secara real-time.
* **Fitur Inti (Level Menengah)**:
  1. **Autentikasi Pengguna**: Pengguna harus mendaftar dan login. Firebase Authentication sangat cocok untuk ini (misal: dengan email & password).
  2. **Daftar Percakapan**: Layar utama menampilkan daftar orang yang pernah berinteraksi dengan pengguna.
  3. **Layar Obrolan (Chat Room)**: Antarmuka percakapan satu lawan satu, menampilkan pesan dalam bentuk gelembung (chat bubbles).
  4. **Pesan Real-time**: Pesan baru dari pengirim langsung muncul di layar penerima tanpa perlu refresh. Ini dicapai dengan *real-time listener* dari database.
* **Logika & State (Level Menengah)**:
  1. **Backend**: Aplikasi ini wajib menggunakan backend. **Firebase** adalah pilihan utama untuk level ini.
     + **Firebase Auth**: Mengelola pengguna (login, register, logout).
     + **Cloud Firestore (Database)**: Menyimpan data pengguna, daftar chat, dan riwayat pesan. Struktur datanya perlu dirancang dengan baik (misal: koleksi users, koleksi chats yang berisi sub-koleksi messages).
  2. **Real-time Listener**: Menggunakan StreamBuilder di Flutter yang terhubung ke snapshots() dari Cloud Firestore. Ini adalah inti dari fitur real-time, di mana UI akan otomatis diperbarui setiap kali ada data baru di database.
* **Tantangan Potensial (Menengah)**:
  1. **Konfigurasi Firebase**: Menghubungkan proyek Flutter dengan proyek Firebase untuk Android dan iOS.
  2. **Struktur Data NoSQL**: Merancang skema database di Firestore yang efisien untuk aplikasi chat.
  3. **Manajemen State Autentikasi**: Mengelola alur aplikasi antara layar login dan layar utama.
  4. **Menggunakan StreamBuilder**: Memahami konsep *streams* untuk data real-time.
  5. Membuat UI chat bubble yang membedakan pesan terkirim dan diterima.
* **Paket Flutter yang Mungkin Digunakan**: firebase\_core, firebase\_auth, cloud\_firestore.

**Tampilan UI (Mode Karakter)**

**Layar Daftar Percakapan**

+------------------------------------------+

| Obrolan [+👤] |

+==========================================+

| |

| +-------------------------------------+ |

| | Fatimah | |

| | < Oke, sampai jumpa di sana! | |

| | 16:16 WIB | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | Grup Kuliah | |

| | < Zaki: Tugasnya dikumpul besok. | |

| | 15:30 WIB | |

| +-------------------------------------+ |

| |

+------------------------------------------+

**Layar Obrolan (Chat Room)**

*(Konteks: Ahmad dan Fatimah merencanakan pertemuan di Banda Aceh)*

+------------------------------------------+

| [<-] Fatimah |

+==========================================+

| |

| +-------------------------+ |

| | Assalamualaikum, jadi | 16:15 |

| | ketemu sore ini? | |

| +-------------------------+ |

| |

| +---------------------+ |

| 16:15 | Waalaikumsalam. Jadi!| |

| | Mau di mana? | |

| +---------------------+ |

| |

| +-------------------------+ |

| | Di Solong Kopi Ulee | 16:16 |

| | Kareng aja gimana? | |

| +-------------------------+ |

| |

| +---------------------+ |

| 16:16 | Boleh. Aku OTW! | |

| +---------------------+ |

| |

| [Ketik pesan Anda...] [Kirim >] |

+------------------------------------------+

Kita bisa lanjutkan ke nomor 22 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke level menengah berikutnya.

**22. Aplikasi E-commerce (Katalog & Keranjang Belanja)**

**Brainstorming**

* **Tujuan Utama**: Membuat kerangka aplikasi toko online di mana pengguna bisa melihat katalog produk, menambahkannya ke keranjang, dan melihat rincian belanja mereka. (Catatan: Checkout dan pembayaran adalah fitur level sulit).
* **Fitur Inti (Level Menengah)**:
  1. **Katalog Produk**: Menampilkan banyak produk dalam bentuk GridView. Data produk (nama, harga, gambar) diambil dari backend seperti **Cloud Firestore**.
  2. **Halaman Detail Produk**: Halaman terpisah untuk setiap produk yang menampilkan deskripsi lengkap, galeri gambar, dan tombol "Tambah ke Keranjang".
  3. **Keranjang Belanja (Shopping Cart)**: Halaman yang menampilkan semua item yang telah ditambahkan. Pengguna bisa mengubah jumlah (quantity) atau menghapus item dari keranjang.
  4. **State Keranjang Global**: Jumlah item di keranjang (misalnya, ikon keranjang di AppBar) harus selalu ter-update dan bisa diakses dari halaman mana pun.
* **Logika & State (Level Menengah)**:
  1. **State Management Lanjutan**: Ini adalah tantangan utama. Mengelola state keranjang yang bersifat global tidak efisien jika hanya menggunakan setState. Ini adalah saat yang tepat untuk belajar **State Management Solution** seperti Provider atau Riverpod.
  2. Provider akan digunakan untuk "menyediakan" sebuah CartModel ke seluruh aplikasi. CartModel ini akan berisi List item di keranjang dan logika untuk menambah, mengurangi, atau menghapus item.
  3. Widget seperti ikon keranjang di AppBar akan "mendengarkan" perubahan di CartModel dan otomatis memperbarui tampilannya.
* **Tantangan Potensial (Menengah)**:
  1. Mempelajari dan mengimplementasikan Provider atau Riverpod untuk manajemen state global.
  2. Mengambil data produk dari sumber eksternal seperti Cloud Firestore.
  3. Merancang logika untuk keranjang belanja (menambah item baru vs menambah jumlah item yang sudah ada, menghitung total harga).
  4. Navigasi yang lebih kompleks antara tiga layar utama (Katalog -> Detail -> Keranjang).
* **Paket Flutter yang Mungkin Digunakan**: **provider** atau **flutter\_riverpod** (wajib untuk latihan ini), cloud\_firestore, intl (untuk format harga).

**Tampilan UI (Mode Karakter)**

**Layar Katalog Produk**

*(Konteks: Toko online yang menjual produk khas Aceh)*

+------------------------------------------+

| Pasar Aceh Digital [🛒 2] |

+==========================================+

| +----------------+ +----------------+ |

| | [Gambar Kopi] | | [Gambar Kain] | |

| | Kopi Gayo 250gr| | Songket Aceh | |

| | Rp 120.000 | | Rp 350.000 | |

| +----------------+ +----------------+ |

| |

| +----------------+ +----------------+ |

| | [Gambar Kue] | | [Gambar Tas] | |

| | Kue Keukarah | | Tas Pintu Aceh | |

| | Rp 45.000 | | Rp 180.000 | |

| +----------------+ +----------------+ |

+------------------------------------------+

**Layar Keranjang Belanja**

+------------------------------------------+

| [<-] Keranjang Belanja |

+==========================================+

| |

| - Kopi Gayo 250gr [Hapus 🗑️]|

| Rp 120.000 |

| Jumlah: ( - ) 1 ( + ) |

| -------------------------------------- |

| - Kue Keukarah (1 Toples) [Hapus 🗑️]|

| Rp 45.000 |

| Jumlah: ( - ) 2 ( + ) |

| -------------------------------------- |

| |

+==========================================+

| |

| Total Belanja: Rp 210.000 |

| |

| [ Proses ke Pembayaran ] |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 23 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke level menengah berikutnya.

**23. Aplikasi Pelacak Kebiasaan (Habit Tracker) dengan Grafik**

**Brainstorming**

* **Tujuan Utama**: Membantu pengguna membangun kebiasaan baik dengan melacak aktivitas harian mereka dan memberikan visualisasi data untuk melihat kemajuan dari waktu ke waktu.
* **Fitur Inti (Level Menengah)**:
  1. **CRUD Kebiasaan**: Pengguna dapat membuat, membaca, mengedit, dan menghapus kebiasaan yang ingin dilacak (misal: "Olahraga", "Baca Buku").
  2. **Pelacakan Harian**: Antarmuka untuk menandai kebiasaan sebagai "selesai" untuk hari ini.
  3. **Visualisasi Data**: Ini adalah fitur kunci yang menempatkannya di level menengah.
     + **Kalender Heatmap**: Menampilkan kalender bulanan, di mana setiap tanggal diberi warna berdasarkan apakah kebiasaan dilakukan atau tidak.
     + **Grafik Statistik**: Menampilkan diagram batang atau garis untuk menunjukkan konsistensi atau *streak* (jumlah hari berturut-turut) dari sebuah kebiasaan.
  4. **Penyimpanan Persisten**: Semua data kebiasaan dan riwayat penyelesaiannya harus disimpan di database lokal.
* **Logika & State (Level Menengah)**:
  1. **Database Lokal**: Memerlukan database yang lebih terstruktur. Hive atau sqflite sangat cocok untuk menyimpan data kebiasaan (misal: Habit(id, name)) dan data penyelesaiannya (misal: HabitCompletion(habitId, date)).
  2. **Logika Tanggal**: Banyak bekerja dengan objek DateTime untuk menandai, mengambil, dan menganalisis data berdasarkan tanggal.
  3. **Logika Statistik**: Menghitung metrik seperti *current streak*, *longest streak*, dan total penyelesaian dalam sebulan untuk ditampilkan di grafik.
  4. **State Management**: Provider atau Riverpod akan sangat membantu untuk mengelola data kebiasaan dan menyediakannya ke berbagai layar (layar harian, layar statistik, dll).
* **Tantangan Potensial (Menengah)**:
  1. **Visualisasi Data**: Mengintegrasikan dan mengonfigurasi paket grafik seperti fl\_chart atau syncfusion\_flutter\_charts.
  2. **Desain Database**: Merancang struktur database yang efisien untuk menyimpan riwayat data yang bisa bertambah banyak.
  3. Menerapkan logika kalkulasi tanggal dan statistik yang bisa jadi cukup rumit.
* **Paket Flutter yang Mungkin Digunakan**: hive atau sqflite (database), fl\_chart (grafik), table\_calendar (kalender).

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Kebiasaan Hari Ini)**

*(Konteks: Memulai kebiasaan baru di awal bulan Juli)*

+------------------------------------------+

| KebiasaanKu - Selasa, 1 Juli |

+==========================================+

| |

| [✔️] Minum 2L air |

| -------------------------------------- |

| [ ] Olahraga 20 menit |

| -------------------------------------- |

| [✔️] Membaca Al-Qur'an (setelah Subuh) |

| -------------------------------------- |

| [ ] Belajar Flutter 1 Jam |

| |

| [+ Tambah] |

+------------------------------------------+

**Layar Detail & Statistik (Saat item "Olahraga" diketuk)**

+------------------------------------------+

| [<-] Statistik: Olahraga 20 menit |

+==========================================+

| |

| Streak Saat Ini 🔥: 0 hari |

| Rekor Streak 🏆: 15 hari |

|------------------------------------------|

| |

| Aktivitas Bulan Ini (Juli 2025): |

| +------------------------------------+ |

| | || ||| | | | | <- Diagram Batang

| | ||||| | ||||| | ||||| | | | |

| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

| 1 2 3 4 5 6 7 8 9 10 ... 31 |

| |

+------------------------------------------+

Kita bisa lanjut ke nomor 24 jika Anda siap.

Yahok

Baik, kita lanjutkan.

**24. Aplikasi Media Sosial (Fitur Feed, Like, dan Komentar)**

**Brainstorming**

* **Tujuan Utama**: Membangun versi dasar dari aplikasi media sosial, di mana pengguna dapat melihat *feed* postingan dari pengguna lain, memberikan "suka" (like), dan meninggalkan komentar.
* **Fitur Inti (Level Menengah)**:
  1. **Autentikasi & Profil Pengguna**: Pengguna harus bisa mendaftar/login (Firebase Auth) dan memiliki profil sederhana (nama, foto profil).
  2. **Membuat Postingan**: Pengguna bisa membuat postingan baru yang berisi teks dan mungkin satu gambar.
  3. **Feed Utama**: Layar utama menampilkan postingan dari semua pengguna secara kronologis terbalik (yang terbaru di atas).
  4. **Fitur "Like"**: Tombol untuk menyukai postingan. Jumlah "suka" ditampilkan dan diperbarui secara real-time.
  5. **Fitur Komentar**: Pengguna bisa mengetuk postingan untuk membuka layar detail di mana mereka bisa melihat dan menambahkan komentar.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firebase)**:
     + **Cloud Firestore**: Struktur data menjadi lebih kompleks. Diperlukan koleksi untuk users, posts, dan sub-koleksi comments di dalam setiap post.
     + **Firebase Storage**: Diperlukan untuk menyimpan file gambar yang diunggah pengguna.
  2. **State Management**: Provider atau Riverpod sangat disarankan untuk mengelola state pengguna yang sedang login (currentUser).
  3. **Logika "Like"**: Mengelola "like" bisa rumit. Cara umum adalah menyimpan List dari userId yang menyukai postingan di dalam dokumen post. Saat tombol "like" ditekan, userId pengguna ditambahkan atau dihapus dari List tersebut.
  4. **Real-time Updates**: StreamBuilder digunakan secara ekstensif untuk feed, jumlah "like", dan daftar komentar agar semua data diperbarui secara instan di seluruh aplikasi.
* **Tantangan Potensial (Menengah)**:
  1. **Struktur Data Firestore yang Kompleks**: Merancang skema yang efisien untuk relasi antar pengguna, postingan, dan komentar.
  2. **Image Upload**: Mengimplementasikan alur untuk memilih gambar dari galeri perangkat, mengunggahnya ke Firebase Storage, dan mendapatkan URL-nya untuk disimpan di Firestore.
  3. Mengelola logika "like" secara efisien.
  4. Mengamankan data dengan **Security Rules** di Firestore (misalnya, pengguna hanya bisa mengedit postingan mereka sendiri).
* **Paket Flutter yang Mungkin Digunakan**: firebase\_auth, cloud\_firestore, firebase\_storage, image\_picker (untuk memilih gambar), provider.

**Tampilan UI (Mode Karakter)**

**Layar Feed Utama**

+------------------------------------------+

| Aceh Social [+ Post] |

+==========================================+

| |

| +-------------------------------------+ |

| | (👤) Zulfikar | |

| | Sore yang indah di Pantai ... | |

| | [Gambar sunset di Lampuuk] | |

| | | |

| | [❤️ 15] [💬 3] < 5 menit lalu | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | (👤) Fatimah | |

| | Menikmati Kopi Sanger terenak! | |

| | #KopiAceh | |

| | [❤️ 28] [💬 7] < 1 jam lalu | |

| +-------------------------------------+ |

| |

+------------------------------------------+

**Layar Detail Postingan & Komentar**

+------------------------------------------+

| [<-] Postingan Zulfikar |

+==========================================+

| +-------------------------------------+ |

| | (👤) Zulfikar | |

| | Sore yang indah di Pantai ... | |

| | [Gambar sunset di Lampuuk] | |

| | | |

| | [❤️ 15 Suka] | |

| +-------------------------------------+ |

| Komentar (3): |

| -------------------------------------- |

| (👤) Ahmad: Mantap sekali fotonya! |

| (👤) Lisa: Jadi pengen ke sana lagi. |

| (👤) Budi: Keren bro! |

| |

| [Tulis komentar...] [Kirim] |

+------------------------------------------+

Kita bisa lanjutkan ke nomor 25 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Oke, kita lanjutkan.

**25. Aplikasi Pelacak Film/Serial TV (dengan API)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi bagi pencinta film untuk menemukan film baru, melihat informasi detail, dan melacak apa yang sudah atau ingin mereka tonton.
* **Fitur Inti (Level Menengah)**:
  1. **Data dari API Eksternal**: Mengambil semua data film (populer, rating tertinggi, pencarian, detail) dari API publik yang kaya fitur seperti **TMDB (The Movie Database)**.
  2. **Eksplorasi & Pencarian**: Halaman utama untuk menemukan film populer dan fungsi pencarian untuk mencari film atau serial TV tertentu.
  3. **Halaman Detail Komprehensif**: Menampilkan poster, sinopsis, rating, tanggal rilis, daftar aktor, dan informasi relevan lainnya untuk setiap film.
  4. **Watchlist Pribadi**: Pengguna (yang sudah login) dapat menambahkan film ke daftar tontonan pribadi mereka ("My Watchlist").
* **Logika & State (Level Menengah)**:
  1. **Integrasi API & Backend**: Ini adalah inti kompleksitasnya. Aplikasi akan berkomunikasi dengan dua sistem berbeda:
     + **TMDB API**: Sebagai sumber data publik untuk semua informasi film.
     + **Firebase (Firestore)**: Sebagai backend pribadi untuk menyimpan data spesifik pengguna, yaitu *watchlist* mereka. Firestore akan menyimpan List berisi movie\_id untuk setiap user\_id.
  2. **Panggilan Jaringan (Networking)**: Melakukan berbagai jenis panggilan HTTP GET ke banyak *endpoint* TMDB dan mengelola API Key.
  3. **Parsing JSON Kompleks**: Respons dari TMDB seringkali bersarang (*nested*) dan kompleks, memerlukan model data Dart (Movie, Actor, dll.) yang dirancang dengan baik.
  4. **State Management**: Provider atau Riverpod digunakan untuk mengelola state autentikasi pengguna dan data watchlist mereka.
* **Tantangan Potensial (Menengah)**:
  1. Bekerja dengan dokumentasi API pihak ketiga yang ekstensif (TMDB).
  2. Mengelola dan menyinkronkan data dari dua sumber yang berbeda (API publik dan backend pribadi).
  3. Merancang UI yang padat informasi namun tetap bersih dan mudah digunakan.
  4. Menggunakan paket seperti cached\_network\_image untuk mengelola pemuatan gambar dari internet secara efisien.
* **Paket Flutter yang Mungkin Digunakan**: http atau dio (panggilan API), provider (state management), firebase\_auth, cloud\_firestore, cached\_network\_image.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Eksplorasi Film)**

+------------------------------------------+

| Movie Mania [👤][🔍]|

+==========================================+

| Sedang Tren Saat Ini |

| ---------------------------------------- |

| +----------+ +----------+ +----------+ |

| | [Poster] | | [Poster] | | [Poster] | |

| | Film A | | Film B | | Film C | |

| +----------+ +----------+ +----------+ |

| |

| Rating Tertinggi Sepanjang Masa |

| ---------------------------------------- |

| +----------+ +----------+ +----------+ |

| | [Poster] | | [Poster] | | [Poster] | |

| | Film X | | Film Y | | Film Z | |

| +----------+ +----------+ +----------+ |

+------------------------------------------+

**Layar Detail Film**

+------------------------------------------+

| [<-] Detail Film |

+==========================================+

| |

| +----------------------------------+ |

| | [Gambar backdrop besar dari film] | |

| +----------------------------------+ |

| |

| Judul Film Keren (2025) ⭐️ 8.9/10 |

| < Sci-Fi, Action | 2h 15m > |

| |

| [+ Tambah ke Watchlist] |

| |

| Sinopsis: |

| Sebuah petualangan epik melintasi |

| galaksi untuk menemukan... |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 26 jika Anda siap.

Yahok

Tentu, mari kita lanjutkan.

**26. Aplikasi Jurnal Harian dengan Sinkronisasi Cloud**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah buku harian digital pribadi di mana pengguna dapat menulis entri jurnal harian mereka dan data tersebut aman tersimpan serta dapat diakses dari berbagai perangkat.
* **Fitur Inti (Level Menengah)**:
  1. **Autentikasi Pengguna**: Pengguna harus login untuk memastikan privasi jurnal mereka.
  2. **Editor Jurnal**: Sebuah antarmuka yang bersih dan bebas gangguan untuk menulis entri jurnal, mirip dengan aplikasi catatan.
  3. **Tampilan Kalender/Daftar**: Pengguna bisa menavigasi entri mereka berdasarkan tanggal melalui tampilan kalender atau daftar kronologis.
  4. **Sinkronisasi Cloud**: Ini adalah fitur kunci. Setiap entri yang dibuat atau diedit secara otomatis disimpan ke **Cloud Firestore**. Ini memastikan data tidak hilang jika aplikasi dihapus dan memungkinkan sinkronisasi antar perangkat.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firebase)**:
     + **Firebase Auth**: Untuk mengelola akun pengguna.
     + **Cloud Firestore**: Struktur datanya cukup sederhana. Bisa berupa koleksi users, di mana setiap dokumen pengguna memiliki sub-koleksi journal\_entries. Setiap dokumen entri akan memiliki content dan timestamp.
  2. **Logika CRUD**: Aplikasi akan mengimplementasikan operasi Create, Read, Update, dan Delete pada data jurnal yang disimpan di Firestore.
  3. **Akses Data Offline**: Untuk pengalaman pengguna yang lebih baik, fitur *offline persistence* dari Firestore bisa diaktifkan. Ini memungkinkan pengguna untuk membaca dan bahkan menulis entri saat tidak ada koneksi internet. Perubahan akan disinkronkan secara otomatis saat koneksi kembali.
* **Tantangan Potensial (Menengah)**:
  1. Mengamankan data pengguna dengan **Firestore Security Rules** agar setiap pengguna hanya bisa mengakses jurnal miliknya sendiri.
  2. Mengaktifkan dan mengelola *offline persistence*.
  3. Merancang UI yang efisien untuk menavigasi entri berdasarkan tanggal.
* **Paket Flutter yang Mungkin Digunakan**: firebase\_auth, cloud\_firestore, intl (untuk format tanggal), table\_calendar (untuk navigasi kalender).

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Entri Jurnal)**

+------------------------------------------+

| Jurnal Pribadiku 📖 [👤] |

+==========================================+

| |

| +-------------------------------------+ |

| | Selasa, 1 Juli 2025 | |

| | <Hari yang cukup produktif. Berhasil> | |

| | <menyelesaikan beberapa tugas...> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | Senin, 30 Juni 2025 | |

| | <Menghabiskan sore di pantai sambil> | |

| | <menikmati senja. Sungguh...> | |

| +-------------------------------------+ |

| |

| [+ Tulis] |

+------------------------------------------+

**Layar Editor Jurnal**

+------------------------------------------+

| [<-] Entri untuk 1 Juli 2025 |

+==========================================+

| |

| Hari yang cukup produktif. Berhasil |

| menyelesaikan beberapa tugas penting di |

| kantor. Sore harinya, saya mencoba resep|

| baru untuk makan malam dan ternyata |

| hasilnya tidak mengecewakan. Merasa |

| bersyukur untuk hari ini. |

| | |

| | |

| | |

| | |

| |

| [Simpan ✔️] |

+------------------------------------------+

Kita bisa lanjutkan ke nomor 27 jika Anda sudah siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita lanjutkan ke nomor 27.

**27. Aplikasi Pencari Event dengan Peta (Integrasi Maps)**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah platform bagi pengguna untuk menemukan acara atau kegiatan (event) yang terjadi di sekitar mereka, dengan visualisasi lokasi pada peta interaktif.
* **Fitur Inti (Level Menengah)**:
  1. **Daftar & Peta Event**: Menampilkan event dalam dua mode: sebagai daftar kronologis dan sebagai penanda (marker) di atas peta (misalnya, Google Maps).
  2. **Data dari Backend**: Informasi event (nama, tanggal, deskripsi, lokasi) diambil dari backend seperti Cloud Firestore.
  3. **Lokasi Geografis**: Setiap event harus memiliki data koordinat (latitude, longitude) agar bisa ditempatkan di peta. Firestore mendukung tipe data GeoPoint untuk ini.
  4. **Detail Event**: Saat pengguna mengetuk event (baik di daftar atau di peta), aplikasi akan menampilkan halaman detail yang berisi informasi lengkap.
* **Logika & State (Level Menengah)**:
  1. **Integrasi Peta**: Ini adalah tantangan utama dan inti dari aplikasi. Menggunakan paket seperti Maps\_flutter yang memerlukan:
     + Mendapatkan dan mengonfigurasi API Key dari platform cloud (Google Cloud Platform).
     + Menampilkan peta dan mengontrol kamera (zoom, posisi awal).
     + Secara dinamis menambahkan Marker ke peta berdasarkan data GeoPoint yang diambil dari Firestore.
  2. **Backend (Firestore)**: Membutuhkan koleksi events, di mana setiap dokumen menyimpan detail event termasuk field location bertipe GeoPoint.
  3. **State Management**: Provider atau Riverpod bisa digunakan untuk mengambil data event sekali dan menyediakannya ke kedua widget, yaitu ListView dan GoogleMap, agar data tetap sinkron.
* **Tantangan Potensial (Menengah)**:
  1. **Konfigurasi API Peta**: Proses mendapatkan dan meng-setup API Key Google Maps bisa menjadi rumit dan seringkali menjadi titik hambatan bagi pemula.
  2. **Bekerja dengan Widget Peta**: Mempelajari cara memanipulasi peta, marker, dan menangani event klik pada marker.
  3. Mengelola data geografis.
* **Paket Flutter yang Mungkin Digunakan**: **Maps\_flutter**, cloud\_firestore, geolocator (jika ingin ada fitur "event di dekat saya").

**Tampilan UI (Mode Karakter)**

**Tampilan Peta**

*(Konteks: Mencari acara akhir pekan di sekitar Banda Aceh)*

+------------------------------------------+

| Temukan Event Seru [List ☰]|

+==========================================+

| |

| /-----------------------------\ |

| | Peta Area Banda Aceh | |

| | | |

| | [📍A] | |

| | [📍B] | |

| | | |

| | [📍C] | |

| \-----------------------------/ |

| |

|------------------------------------------|

| Info Marker A: |

| > Pameran Seni di Museum Tsunami |

+------------------------------------------+

**Tampilan Daftar (Saat tombol [List ☰] ditekan)**

+------------------------------------------+

| Temukan Event Seru [Peta 🗺️]|

+==========================================+

| |

| +-------------------------------------+ |

| | [A] Pameran Seni di Museum Tsunami | |

| | <Jumat, 4 Juli 2025> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | [B] Festival Kuliner Aceh | |

| | <Sabtu, 5 Juli 2025> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | [C] Konser Musik Etnik | |

| | <Sabtu, 5 Juli 2025> | |

| +-------------------------------------+ |

+------------------------------------------+

Kita lanjutkan ke nomor 28 jika Anda siap.

Yahok

Tampilkan alur berpikir

Oke, kita lanjutkan.

**28. Aplikasi Pelacak Latihan Fitness**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi panduan latihan di mana pengguna dapat memilih sebuah program latihan dan mengikuti setiap gerakannya dengan panduan waktu.
* **Fitur Inti (Level Menengah)**:
  1. **Daftar Program Latihan**: Menampilkan berbagai pilihan program (misal: "Pemanasan Pagi", "Latihan Perut 7 Menit").
  2. **Layar Sesi Latihan Aktif**: Antarmuka interaktif yang menampilkan:
     + Nama dan gambar/animasi gerakan yang sedang dilakukan.
     + **Timer hitung mundur** besar untuk durasi gerakan atau waktu istirahat.
     + Navigasi otomatis ke gerakan berikutnya setelah timer selesai.
  3. **Kontrol Sesi**: Tombol untuk menjeda (pause), melanjutkan (resume), dan melewati (skip) gerakan.
  4. **Riwayat Latihan**: Menyimpan catatan setiap sesi latihan yang berhasil diselesaikan oleh pengguna.
* **Logika & State (Level Menengah)**:
  1. **Manajemen State Sesi Aktif**: Ini adalah inti kesulitannya. State selama sesi latihan sangat dinamis (gerakan saat ini, sisa waktu, status jeda/berjalan). Menggunakan **State Management Solution** seperti Provider atau Riverpod sangat disarankan untuk mengelola state ini secara terpusat.
  2. **Logika Timer**: Mengimplementasikan Timer yang akurat, bisa dijeda, dan bisa direset adalah tantangan tersendiri. Ini lebih kompleks dari sekadar jam digital.
  3. **Data Program Latihan**: Daftar gerakan, durasi, dan waktu istirahat untuk setiap program bisa disimpan di file JSON lokal atau di database seperti Hive atau Firestore.
  4. **Database**: Hive atau sqflite digunakan untuk menyimpan riwayat latihan pengguna.
* **Tantangan Potensial (Menengah)**:
  1. Mengelola state yang kompleks dan berubah dengan cepat selama sesi latihan.
  2. Mengimplementasikan logika timer yang bisa dijeda dan dilanjutkan.
  3. Merancang alur otomatis dari satu latihan ke istirahat, lalu ke latihan berikutnya.
* **Paket Flutter yang Mungkin Digunakan**: provider (state management), hive (database), audioplayers (untuk memainkan suara notifikasi timer).

**Tampilan UI (Mode Karakter)**

**Layar Sesi Latihan Aktif**

*(Konteks: Melakukan latihan sore di rumah)*

+------------------------------------------+

| Program: Pemanasan Pagi |

+==========================================+

| Gerakan 3/10: High Knees |

| [|||||.........................] 25% |

| |

| +--------------------------+ |

| | | |

| | [Animasi/GIF gerakan] | |

| | | |

| +--------------------------+ |

| |

| 00:18 |

| |

| |

| [ Jeda || ] [ Lewati >> ] |

| |

| Selanjutnya: Istirahat (10 detik) |

+------------------------------------------+

**Layar Ringkasan Setelah Selesai**

+------------------------------------------+

| Kerja Bagus! 💪 |

+==========================================+

| |

| Anda telah menyelesaikan: |

| Program: Pemanasan Pagi |

| |

| Durasi Total: 7 menit 10 detik |

| Kalori Terbakar: ~85 kcal |

| |

| |

| |

| [ Kembali ke Menu ] |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 29 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**29. Aplikasi Booking Sederhana (misal: booking lapangan)**

**Brainstorming**

* **Tujuan Utama**: Membuat sistem di mana pengguna dapat melihat ketersediaan jadwal dan memesan (booking) slot waktu untuk fasilitas tertentu, seperti lapangan futsal, studio musik, atau ruang rapat.
* **Fitur Inti (Level Menengah)**:
  1. **Pemilihan Fasilitas & Tanggal**: Pengguna dapat memilih fasilitas yang ingin dibooking dan tanggalnya.
  2. **Tampilan Jadwal (Time Slots)**: Menampilkan grid slot waktu untuk tanggal yang dipilih, dengan indikator visual yang jelas untuk slot yang sudah terisi dan yang masih tersedia.
  3. **Proses Booking**: Pengguna dapat memilih satu atau lebih slot waktu yang tersedia dan mengonfirmasi pesanan mereka.
  4. **Backend untuk Jadwal**: Semua data booking disimpan dan dikelola secara terpusat di backend agar data selalu konsisten untuk semua pengguna.
  5. **Riwayat Booking Pengguna**: Pengguna dapat melihat daftar booking yang pernah mereka lakukan.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**: Sangat krusial untuk aplikasi ini.
     + **Struktur Data**: Diperlukan koleksi bookings. Setiap dokumen akan menyimpan facility\_id, user\_id, date, dan time\_slot.
     + **Query Ketersediaan**: Saat pengguna memilih tanggal, aplikasi akan melakukan *query* ke Firestore untuk mengambil semua booking pada tanggal tersebut untuk fasilitas yang dipilih. Logika di aplikasi kemudian akan menentukan slot mana yang masih kosong.
  2. **Transaksi Atomik**: Untuk mencegah dua pengguna memesan slot yang sama secara bersamaan (*race condition*), proses penyimpanan booking baru harus menggunakan **Firebase Transactions**. Transaksi memastikan bahwa operasi baca dan tulis terjadi sebagai satu kesatuan yang tidak dapat diganggu.
* **Tantangan Potensial (Menengah)**:
  1. **Merancang Skema Database**: Mendesain struktur di Firestore yang efisien untuk mengelola jadwal dan ketersediaan.
  2. **Logika Ketersediaan**: Menerjemahkan data booking dari database menjadi UI grid jadwal yang mudah dipahami.
  3. **Mengimplementasikan Transaksi Firestore**: Mempelajari dan menggunakan runTransaction untuk memastikan integritas data saat booking.
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, table\_calendar (untuk pemilihan tanggal yang lebih baik), intl.

**Tampilan UI (Mode Karakter)**

**Layar Pemilihan Slot Waktu**

*(Konteks: Booking lapangan futsal di Banda Aceh untuk main nanti malam)*

+------------------------------------------+

| [<-] Booking Lapangan Futsal Lampineung |

+==========================================+

| |

| Pilih Tanggal: [ Selasa, 1 Juli v ] |

| |

|------------------------------------------|

| Slot Waktu Tersedia (1 jam/sesi): |

| |

| [ 16:00 ] ( 17:00 ) [ 18:00 ] |

| |

| [ 19:00 ] [ 20:00 ] ( 21:00 ) |

| |

| ( 22:00 ) ( 23:00 ) |

| |

| Ket: [ ] Tersedia ( ) Dipesan |

| |

|------------------------------------------|

| Pilihan Anda: 19:00 |

| Total Biaya : Rp 100.000 |

| |

| [ Konfirmasi Booking ] |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 30 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**30. Aplikasi Manajemen Inventaris Barang**

**Brainstorming**

* **Tujuan Utama**: Membuat alat bantu bagi pemilik usaha kecil untuk melacak stok barang, mencatat barang masuk dan keluar, dan mengetahui kapan harus melakukan pemesanan ulang.
* **Fitur Inti (Level Menengah)**:
  1. **Manajemen Produk (CRUD)**: Pengguna bisa menambah, melihat, mengedit, dan menghapus produk dalam inventaris mereka (misal: nama, SKU, stok awal).
  2. **Pencatatan Transaksi**: Fitur untuk mencatat "Stok Masuk" (saat barang diterima dari pemasok) dan "Stok Keluar" (saat barang terjual atau digunakan). Setiap transaksi ini akan otomatis memperbarui jumlah stok produk terkait.
  3. **Riwayat Stok**: Melihat histori pergerakan stok untuk setiap item.
  4. **Peringatan Stok Rendah**: Memberikan notifikasi visual jika stok suatu barang berada di bawah ambang batas yang telah ditentukan.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**:
     + **Struktur Data**: Membutuhkan koleksi products (berisi info produk dan jumlah stok saat ini) dan koleksi stock\_movements (berisi riwayat semua transaksi masuk/keluar).
     + **Integritas Data**: Saat mencatat transaksi, jumlah stok di dokumen produk harus diperbarui secara andal. Menggunakan **Firebase Transactions** atau FieldValue.increment() sangat penting untuk mencegah data yang tidak konsisten akibat beberapa operasi yang terjadi bersamaan.
  2. **State Management**: Provider atau Riverpod berguna untuk mengelola dan menampilkan daftar produk yang selalu ter-update di seluruh aplikasi.
* **Tantangan Potensial (Menengah)**:
  1. Memastikan konsistensi dan integritas data antara tabel transaksi dan tabel produk.
  2. Merancang skema database yang efisien.
  3. Membuat antarmuka yang cepat dan mudah untuk input data transaksi stok.
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, provider, flutter\_barcode\_scanner (untuk fitur tambahan pemindaian barcode produk).

**Tampilan UI (Mode Karakter)**

**Layar Daftar Inventaris**

*(Konteks: Pemilik kedai kopi di Banda Aceh mengelola stoknya)*

+------------------------------------------+

| Inventaris: Kedai Kopi Sanger [+ 📝] |

+==========================================+

| |

| -> Biji Kopi Gayo Arabika Stok: 12 kg |

| ---------------------------------------- |

| -> Biji Kopi Gayo Robusta Stok: 8 kg |

| ---------------------------------------- |

| -> Susu Kental Manis Stok: 3 kaleng |

| ( ⚠️ Stok Menipis! ) |

| ---------------------------------------- |

| -> Gula Aren Stok: 5 kg |

| ---------------------------------------- |

| |

+------------------------------------------+

**Dialog Pencatatan Transaksi (Saat tombol [+ 📝] ditekan)**

+------------------------------------------+

| [ Catat Transaksi Stok ] |

| |

| Pilih Barang: |

| | Susu Kental Manis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [v] | |

| |

| Tipe Transaksi: |

| [Stok Masuk] (Stok Keluar) |

| |

| Jumlah: |

| | 24\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | kaleng |

| |

| ( Batal ) [ Simpan ] |

+------------------------------------------+

Kita lanjutkan ke nomor 31 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita mulai blok kedua dari kategori Menengah.

**31. Aplikasi Pemutar Podcast (dengan download offline)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi pemutar podcast yang memungkinkan pengguna untuk mencari, mendengarkan streaming, dan mengunduh episode untuk didengarkan secara offline.
* **Fitur Inti (Level Menengah)**:
  1. **Penemuan Podcast**: Mengambil daftar podcast dan episode dari sumber eksternal (API podcast atau RSS feed).
  2. **Pemutar Audio Lengkap**: Antarmuka pemutar dengan kontrol play, pause, seek (geser durasi), dan kontrol kecepatan pemutaran (1x, 1.5x, dll.).
  3. **Pemutaran di Latar Belakang (Background Audio)**: Ini adalah fitur kunci. Audio harus terus berjalan saat aplikasi diminimalkan atau layar mati, lengkap dengan notifikasi kontrol media.
  4. **Download untuk Offline**: Fungsi untuk mengunduh file audio episode ke penyimpanan lokal perangkat.
* **Logika & State (Level Menengah)**:
  1. **Manajemen Audio & Background Service**: Ini tantangan utama. Memerlukan paket yang kuat seperti **just\_audio** yang dikombinasikan dengan **audio\_service**.
     + just\_audio: Menangani logika pemutaran audio (play, pause, seek, dll.).
     + audio\_service: Mengelola *background service* agar audio tetap berjalan, menampilkan notifikasi media di OS, dan menangani input dari notifikasi tersebut (misalnya, tombol play/pause di notifikasi).
  2. **Manajemen Unduhan**: Menggunakan paket seperti dio atau flutter\_downloader untuk mengelola proses unduhan, termasuk menampilkan progres, menangani kegagalan, dan menyimpan file ke direktori yang benar (menggunakan path\_provider).
  3. **State Management**: Provider atau Riverpod sangat berguna untuk menyebarkan state dari audio\_service (misal: episode apa yang sedang diputar, statusnya apa) ke seluruh UI aplikasi.
* **Tantangan Potensial (Menengah)**:
  1. **Konfigurasi Background Service**: Meng-setup audio\_service di Android (AndroidManifest.xml) dan iOS (Info.plist dengan background modes) bisa sangat rumit.
  2. Mengelola siklus hidup unduhan dan file lokal.
  3. Menjaga UI tetap sinkron dengan state pemutar yang berjalan di background.
* **Paket Flutter yang Mungkin Digunakan**: **just\_audio** dan **audio\_service** (kombinasi terbaik), webfeed (parsing RSS), dio (download), path\_provider.

**Tampilan UI (Mode Karakter)**

**Layar Daftar Episode sebuah Podcast**

*(Konteks: Podcast tentang Sejarah dan Budaya Aceh)*

+------------------------------------------+

| [<-] Podcast: Hikayat Aceh |

+==========================================+

| Episode: |

| |

| - Riwayat Laksamana Keumalahayati [▶️][⤓] |

| <42 menit | 1 Juli 2025> |

| ---------------------------------------- |

| - Jejak Kesultanan Samudera Pasai[▶️][✔️] |

| <55 menit | 24 Juni 2025> |

| (✔️ = sudah diunduh) |

| ---------------------------------------- |

| - Asal-usul Kopi Gayo [▶️][⤓] |

| <35 menit | 17 Juni 2025> |

| |

+------------------------------------------+

| Mini Player (muncul saat ada yg diputar) |

+------------------------------------------+

| > Riwayat Laksamana... [ || ] 3:14/42:00|

+------------------------------------------+

**Layar Pemutar Penuh (Saat mini player di-tap)**

+------------------------------------------+

| [ v ] Now Playing |

+==========================================+

| |

| +--------------------------+ |

| | [Cover Art Podcast] | |

| +--------------------------+ |

| |

| Riwayat Laksamana Keumalahayati |

| <Hikayat Aceh> |

| |

| < -----O---------------------- > 25:30 |

| |

| [↶15s] [ ▶️ ] [15s↷] |

| |

| Kecepatan: [ 1x ] |

+------------------------------------------+

Kita lanjutkan ke nomor 32 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**32. Aplikasi Keuangan Pribadi (Pencatatan & Visualisasi Data)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi manajemen keuangan yang tidak hanya mencatat pemasukan dan pengeluaran, tetapi juga memberikan wawasan kepada pengguna melalui visualisasi data seperti grafik dan diagram.
* **Fitur Inti (Level Menengah)**:
  1. **Pencatatan Transaksi Berbasis Kategori**: Pengguna bisa mencatat pemasukan atau pengeluaran dan memberinya kategori (misal: Gaji, Makanan, Transportasi, Hiburan).
  2. **Dasbor Ringkasan**: Halaman utama yang menampilkan ringkasan kondisi keuangan bulan ini (total pemasukan, total pengeluaran, dan sisa saldo).
  3. **Visualisasi Data**: Ini adalah fitur kunci. Menampilkan **diagram lingkaran (pie chart)** yang memecah persentase pengeluaran berdasarkan kategori.
  4. **Riwayat & Filter**: Menampilkan riwayat semua transaksi dengan kemampuan untuk memfilter berdasarkan bulan.
* **Logika & State (Level Menengah)**:
  1. **Database Lokal**: Membutuhkan database terstruktur seperti Hive atau sqflite untuk menyimpan List objek Transaction (yang berisi jumlah, tipe, kategori, dan tanggal).
  2. **Agregasi Data**: Tantangan utamanya adalah menulis logika untuk memproses data mentah dari database. Misalnya, untuk membuat pie chart, aplikasi harus:
     1. Mengambil semua transaksi pengeluaran bulan ini.
     2. Mengelompokkannya berdasarkan kategori.
     3. Menjumlahkan total untuk setiap kategori.
     4. Menyiapkan data ini agar bisa diterima oleh paket grafik.
  3. **Integrasi Grafik**: Menggunakan paket eksternal untuk mengubah data yang telah diagregasi menjadi diagram lingkaran yang interaktif.
  4. **State Management**: Provider atau Riverpod akan membantu mengelola state data transaksi dan ringkasan keuangan agar mudah diakses oleh berbagai widget.
* **Tantangan Potensial (Menengah)**:
  1. **Logika Agregasi Data**: Menulis query dan logika Dart untuk memproses data bisa menjadi rumit.
  2. **Menggunakan Paket Grafik**: Mempelajari cara mengonfigurasi dan memberikan data ke paket seperti fl\_chart.
  3. Merancang dasbor UI yang informatif dan tidak berantakan.
* **Paket Flutter yang Mungkin Digunakan**: hive atau sqflite (database), **fl\_chart** atau **pie\_chart** (untuk visualisasi), provider (state management), intl.

**Tampilan UI (Mode Karakter)**

**Layar Dasbor Utama**

*(Konteks: Laporan keuangan untuk bulan Juli)*

+------------------------------------------+

| Dompet Saya - Juli 2025 [+ Transaksi] |

+==========================================+

| Saldo Bulan Ini: Rp 1.500.000 |

| > Pemasukan: Rp 4.000.000 |

| > Pengeluaran: Rp 2.500.000 |

|------------------------------------------|

| |

| Alokasi Pengeluaran |

| |

| /-------------\ |

| / Makanan 40% \ \* Makanan |

| | Kost 30% | \* Kost |

| \ Transport 15%/ \* Transport |

| \---- Lain2 --/ \* Lainnya |

| |

|------------------------------------------|

| Transaksi Terakhir: |

| [+] Gaji Juli Rp 4.000.000 |

| [-] Bayar Kost Rp 750.000 |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 33 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**33. Aplikasi Papan Proyek (seperti Trello, dengan drag-and-drop)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi manajemen proyek visual bergaya Kanban, di mana tugas direpresentasikan sebagai "kartu" yang bisa dipindahkan antar "kolom" (misal: "Rencana", "Dikerjakan", "Selesai").
* **Fitur Inti (Level Menengah)**:
  1. **Tampilan Papan (Board)**: Tampilan utama yang bisa digeser secara horizontal untuk melihat semua kolom.
  2. **Kolom & Kartu**: Pengguna bisa membuat kolom baru dan menambahkan kartu tugas di dalam setiap kolom.
  3. **Drag-and-Drop**: Ini adalah fitur kunci dan tantangan utama. Pengguna harus bisa menekan lama sebuah kartu dan menyeretnya untuk:
     + Mengubah urutannya di dalam kolom yang sama.
     + Memindahkannya ke kolom yang berbeda.
  4. **Sinkronisasi Backend**: Perubahan posisi kartu harus disimpan ke backend (Firestore) agar status papan tetap konsisten.
* **Logika & State (Level Menengah)**:
  1. **Implementasi Drag-and-Drop**:
     + **Pendekatan Paket**: Cara yang paling disarankan untuk level ini adalah menggunakan paket seperti **drag\_and\_drop\_lists** yang sudah menyediakan sebagian besar logika UI.
     + **Pendekatan Manual**: Menggunakan kombinasi widget Draggable dan DragTarget dari Flutter. Ini memberikan kontrol penuh tetapi jauh lebih kompleks.
  2. **Backend (Firestore)**:
     + **Struktur Data**: Diperlukan struktur yang mendukung pengurutan. Misalnya, koleksi boards, lalu sub-koleksi lists (kolom). Setiap dokumen list punya sub-koleksi cards. Baik list maupun card harus memiliki field position atau order (berupa angka).
  3. **Pembaruan State & Backend**: Saat kartu berhasil dipindahkan di UI, sebuah fungsi akan dipanggil untuk memperbarui field position dan/atau listId dari kartu tersebut di Firestore.
* **Tantangan Potensial (Menengah)**:
  1. Mengimplementasikan fungsionalitas drag-and-drop yang mulus.
  2. Merancang struktur data di Firestore yang mendukung pengurutan dan relasi antar item.
  3. Menangani pembaruan data di backend secara efisien setiap kali ada perubahan di UI.
* **Paket Flutter yang Mungkin Digunakan**: **drag\_and\_drop\_lists** atau **boardview** (untuk UI), cloud\_firestore, firebase\_auth, provider.

**Tampilan UI (Mode Karakter)**

**Tampilan Papan Proyek Kanban**

*(Konteks: Tim mahasiswa Unsyiah mengelola tugas akhir mereka)*

+------------------------------------------------------------------+

| Proyek Akhir: Aplikasi UMKM Aceh (<--) Geser (-->) |

+==================================================================+

| |

| +------------------+ +------------------+ +------------------+ |

| | TO DO | | IN PROGRESS | | DONE | |

| |------------------| |------------------| |------------------| |

| | [Rancang DB ] | | [Buat UI Login ] | | [Riset Awal ] | |

| | [Tulis Bab 1 ] | | | | | |

| | [Buat Mockup UI] | | | | | |

| | | | | | | |

| +------------------+ +------------------+ +------------------+ |

| |

+------------------------------------------------------------------+

**Simulasi Aksi Drag-and-Drop**

+------------------------------------------------------------------+

| Proyek Akhir: Aplikasi UMKM Aceh (<--) Geser (-->) |

+==================================================================+

| |

| +------------------+ +------------------+ +------------------+ |

| | TO DO | | IN PROGRESS | | DONE | |

| |------------------| |------------------| |------------------| |

| | [Rancang DB ] | | [Buat UI Login ] | | [Riset Awal ] | |

| | [Tulis Bab 1 ] | | | | | |

| | | | | | | |

| +------------------+ +------------------+ +------------------+ |

| \ |

| [Buat Mockup UI] <- (sedang diseret)|

| |

+------------------------------------------------------------------+

Kita lanjutkan ke nomor 34 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan.

**34. Aplikasi Pengiriman Makanan (sisi pelanggan, dengan pelacakan)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi dari sisi pelanggan untuk memesan makanan dari restoran dan yang terpenting, bisa melacak status pesanan secara real-time dari konfirmasi hingga tiba di tujuan.
* **Fitur Inti (Level Menengah)**:
  1. **Eksplorasi & Pemesanan**: Fungsionalitas standar seperti menelusuri restoran, melihat menu, dan menambahkan item ke keranjang (mirip aplikasi E-commerce).
  2. **Checkout Sederhana**: Mengonfirmasi pesanan dan alamat (tanpa integrasi pembayaran nyata).
  3. **Pelacakan Pesanan Real-time**: Ini adalah fitur kunci. Setelah memesan, pengguna dialihkan ke layar pelacakan yang menampilkan status pesanan secara live. Statusnya bisa berupa: "Pesanan Dikonfirmasi", "Sedang Disiapkan", "Diantar Driver", "Pesanan Tiba".
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**:
     + **Struktur Data**: Diperlukan koleksi orders. Setiap dokumen pesanan akan memiliki field status (misalnya, sebuah string atau angka yang merepresentasikan status).
  2. **Real-time Listener**: Layar pelacakan pesanan akan menggunakan **StreamBuilder** yang "mendengarkan" perubahan pada **satu dokumen spesifik** (dokumen pesanan pengguna tersebut) di Firestore.
  3. **Simulasi Update**: Untuk level menengah, status pesanan bisa diubah secara manual di konsol Firebase untuk tujuan testing, atau menggunakan *Cloud Function* sederhana yang mengubah status setiap beberapa menit untuk mensimulasikan proses nyata.
* **Tantangan Potensial (Menengah)**:
  1. Merancang UI pelacakan pesanan yang informatif dan mudah dipahami (misalnya, menggunakan stepper visual).
  2. Menggunakan StreamBuilder untuk mendengarkan perubahan pada satu dokumen, bukan seluruh koleksi.
  3. Memikirkan cara mensimulasikan perubahan status dari sisi "backend" atau "driver".
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, provider (untuk keranjang belanja), lottie (untuk animasi menarik di layar pelacakan).

**Tampilan UI (Mode Karakter)**

**Layar Pelacakan Pesanan (Order Tracking)**

*(Konteks: Memesan makan malam dari restoran lokal di Banda Aceh)*

+------------------------------------------+

| Lacak Pesanan Anda (#JKL-567) |

+==========================================+

| |

| Estimasi Tiba dalam: 15 Menit |

| Driver: Bang Jali (0812-xxxx-xxxx) |

| |

|------------------------------------------|

| |

| (✔️) Pesanan Dikonfirmasi |

| 16:50 WIB |

| | |

| (✔️) Makanan Sedang Disiapkan |

| 16:52 WIB |

| | |

| (▶️) Driver Mengantar Pesanan |

| 16:55 WIB |

| | |

| ( ) Tiba di Tujuan Anda |

| |

| |

+------------------------------------------+

*(Keterangan: (✔️) Selesai, (▶️) Sedang Berlangsung, ( ) Menunggu)*

Kita lanjutkan ke nomor 35 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**35. Aplikasi Forum Komunitas Sederhana**

**Brainstorming**

* **Tujuan Utama**: Membuat platform diskusi di mana anggota komunitas dapat membuat topik pembicaraan (threads) dan berpartisipasi dengan memberikan balasan (replies).
* **Fitur Inti (Level Menengah)**:
  1. **Autentikasi Pengguna**: Pengguna harus login untuk bisa membuat topik atau membalas.
  2. **Daftar Topik**: Layar utama menampilkan semua topik yang ada, biasanya diurutkan berdasarkan aktivitas terakhir.
  3. **Buat Topik Baru**: Pengguna bisa memulai percakapan baru dengan judul dan postingan pembuka.
  4. **Halaman Detail Topik**: Menampilkan postingan pembuka secara lengkap, diikuti oleh daftar semua balasan dari pengguna lain secara kronologis.
  5. **Membalas Topik**: Terdapat kolom input di halaman detail untuk pengguna menambahkan balasan mereka.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**:
     + **Struktur Data Bersarang (Nested)**: Ini adalah tantangan utama dalam desain database NoSQL. Cara yang umum adalah:
       - Sebuah koleksi threads.
       - Setiap dokumen thread memiliki data seperti title, content, creatorId.
       - Di dalam setiap dokumen thread, terdapat **sub-koleksi** replies yang berisi semua balasan untuk thread tersebut.
  2. **Real-time Updates**: StreamBuilder digunakan untuk mendengarkan koleksi replies di halaman detail, sehingga balasan baru akan langsung muncul tanpa perlu refresh.
  3. **Query & Ordering**: Melakukan query untuk mengambil daftar threads dan mengurutkannya, serta query untuk mengambil semua replies dari sebuah thread.
* **Tantangan Potensial (Menengah)**:
  1. Merancang skema database NoSQL untuk data yang secara konseptual bersifat relasional (topik dan balasannya).
  2. Mengambil dan menampilkan data dari sub-koleksi di Firestore.
  3. Mengelola state pengguna untuk hak akses (hanya pengguna ter-login yang bisa membalas).
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, provider, intl (untuk format tanggal).

**Tampilan UI (Mode Karakter)**

**Layar Utama (Daftar Topik Forum)**

*(Konteks: Forum diskusi untuk warga kota Banda Aceh)*

+------------------------------------------+

| Forum Diskusi Warga Banda Aceh [+ Topik] |

+==========================================+

| |

| > Info pengurusan KTP baru di mana ya? |

| Oleh: Budi | 3 Balasan | 5 menit lalu |

| ---------------------------------------- |

| > Rekomendasi Servis Laptop Terpercaya |

| Oleh: Lisa | 8 Balasan | 2 jam lalu |

| ---------------------------------------- |

| > Acara Car Free Day minggu ini ada? |

| Oleh: Ahmad| 1 Balasan | 4 jam lalu |

| ---------------------------------------- |

| |

+------------------------------------------+

**Layar Detail Topik & Balasan**

+------------------------------------------+

| [<-] Info pengurusan KTP baru... |

+==========================================+

| Post oleh: Budi |

| ---------------------------------------- |

| Permisi, mau tanya, kalau mau urus KTP |

| baru untuk pemula, lokasinya di Disduk- |

| capil kota atau bisa di kecamatan ya? |

| Terima kasih infonya. |

|==========================================|

| Balasan (3): |

| |

| Fatimah: Bisa di Disdukcapil kota, Bang. |

| Bawa fotokopi KK saja. |

| |

| Zaki: Betul, di Gedung Mavista lantai 2. |

| |

| [Tulis balasan Anda...] [Kirim] |

+------------------------------------------+

Kita lanjutkan ke nomor 36 jika Anda siap.

Yahok

Tampilkan alur berpikir

Oke, kita lanjutkan.

**36. Aplikasi Pembelajaran Bahasa (seperti Duolingo sederhana)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi pembelajaran bahasa yang interaktif, menyajikan materi dalam unit-unit kecil, dan menguji pemahaman pengguna melalui berbagai jenis latihan.
* **Fitur Inti (Level Menengah)**:
  1. **Jalur Pembelajaran (Learning Path)**: Tampilan utama yang menunjukkan urutan modul atau pelajaran (misal: Dasar 1, Frasa, Makanan). Pelajaran berikutnya baru terbuka setelah pelajaran sebelumnya selesai.
  2. **Beragam Tipe Latihan**: Ini adalah fitur kunci yang membuatnya lebih kompleks dari aplikasi kuis biasa. Contohnya:
     + **Pilihan Ganda**: Mencocokkan gambar dengan kata atau menerjemahkan kata.
     + **Menyusun Kalimat**: Menyusun kata-kata yang diacak menjadi sebuah kalimat yang benar.
  3. **Sistem Progres**: Melacak pelajaran mana yang sudah diselesaikan oleh pengguna untuk membuka pelajaran baru.
  4. **Data dari Backend**: Semua materi pelajaran dan latihan disimpan di backend (Firestore) agar mudah diperbarui dan dikelola.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**:
     + **Struktur Data**: Memerlukan desain skema yang fleksibel. Misalnya, koleksi lessons, di mana setiap dokumen memiliki sub-koleksi exercises. Setiap dokumen exercise memiliki type ("multipleChoice", "arrangeSentence"), prompt, options, dan correctAnswer.
  2. **Logika Latihan**: Aplikasi harus bisa menampilkan UI yang berbeda tergantung pada type latihan yang diambil dari Firestore.
  3. **Logika Evaluasi**: Fungsi untuk memeriksa jawaban pengguna harus bisa menangani berbagai format jawaban (membandingkan string untuk pilihan ganda, membandingkan urutan List untuk menyusun kalimat).
  4. **Manajemen Progres**: Data progres pengguna (pelajaran mana yang sudah selesai) harus disimpan di dokumen pengguna di Firestore.
* **Tantangan Potensial (Menengah)**:
  1. Merancang dan membangun beberapa widget latihan yang berbeda dan dinamis.
  2. Membuat struktur data di Firestore yang bisa mengakomodasi berbagai jenis latihan.
  3. Mengelola state selama sesi pelajaran (latihan ke berapa, skor, dll.).
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, provider, audioplayers (untuk memutar audio pengucapan kata).

**Tampilan UI (Mode Karakter)**

**Layar Jalur Pembelajaran (Learning Path)**

*(Konteks: Aplikasi untuk belajar Bahasa Aceh)*

+------------------------------------------+

| Mari Belajar Bahasa Aceh |

+==========================================+

| |

| (✔️) Perkenalan |

| | |

| (▶️) Frasa Harian |

| | |

| (🔒) Anggota Keluarga |

| | |

| (🔒) Di Pasar |

| |

+------------------------------------------+

*(Keterangan: (✔️) Selesai, (▶️) Sedang Dikerjakan, (🔒) Terkunci)*

**Contoh Layar Latihan (Tipe: Menyusun Kalimat)**

+------------------------------------------+

| Pelajaran: Frasa Harian [|||||.......] |

+------------------------------------------+

| |

| Susun kalimat ini: |

| "Siapa nama kamu?" |

| |

| +--------------------------------------+ |

| | Soe nan gata | |

| +--------------------------------------+ |

| |

| |

| ( gata ) ( Nan ) ( Soe ) |

| |

| [ Periksa ] |

+------------------------------------------+

Kita lanjutkan ke nomor 37 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita lanjutkan.

**37. Aplikasi Berita dengan Fitur Offline Mode**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi pembaca berita yang andal, yang memungkinkan pengguna untuk tetap bisa membaca artikel berita bahkan saat perangkat tidak terhubung ke internet.
* **Fitur Inti (Level Menengah)**:
  1. **Pengambilan & Caching**: Saat online, aplikasi mengambil berita terbaru dari API atau RSS feed. Secara bersamaan, data berita (teks dan URL gambar) disimpan ke dalam database lokal (cache).
  2. **Mode Offline**: Saat aplikasi mendeteksi tidak ada koneksi internet, ia akan secara otomatis beralih untuk menampilkan data berita dari cache lokal, bukan mencoba menghubungi server.
  3. **Indikator Status Koneksi**: Memberikan feedback visual yang jelas kepada pengguna mengenai status koneksi mereka saat ini (Online atau Offline).
  4. **Caching Gambar**: Gambar dari artikel juga di-cache agar bisa ditampilkan saat offline.
* **Logika & State (Level Menengah)**:
  1. **Arsitektur Offline-First**: Ini adalah inti dari aplikasi. Logika aplikasi dirancang untuk selalu mencoba membaca dari database lokal terlebih dahulu.
  2. **Deteksi Konektivitas**: Menggunakan paket seperti **connectivity\_plus** untuk mendengarkan perubahan status jaringan (WiFi, Seluler, Offline) secara real-time.
  3. **Database Lokal untuk Caching**: Menggunakan database NoSQL yang cepat seperti **Hive** sangat ideal untuk menyimpan data berita dalam format yang mirip dengan JSON.
  4. **Logika Sinkronisasi**: Saat aplikasi online, ia akan mengambil data baru, membandingkannya dengan data di cache, lalu memperbarui cache dengan berita terbaru.
* **Tantangan Potensial (Menengah)**:
  1. Merancang arsitektur dan logika untuk beralih antara sumber data online dan offline dengan mulus.
  2. Mengelola database cache, termasuk kapan harus membersihkan data lama.
  3. Memberikan pengalaman pengguna yang konsisten di kedua mode.
* **Paket Flutter yang Mungkin Digunakan**: **connectivity\_plus** (wajib), **hive** (database cache), http atau webfeed, **cached\_network\_image** (untuk caching gambar secara otomatis).

**Tampilan UI (Mode Karakter)**

**Tampilan Saat Online**

+------------------------------------------+

| Kabar Aceh Hari Ini (🟢 Online) |

+==========================================+

| |

| +-------------------------------------+ |

| | [Gambar] Ekspor Kopi Gayo Meningkat | |

| | <1 Juli 2025> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | [Gambar] Festival Danau Laut Tawar | |

| | Siap Digelar | |

| | <1 Juli 2025> | |

| +-------------------------------------+ |

| |

| (Menyimpan berita untuk dibaca offline) |

+------------------------------------------+

**Tampilan Saat Offline**

+------------------------------------------+

| Kabar Aceh Hari Ini (🔴 Offline)|

+==========================================+

| ! Anda offline. Menampilkan berita yg |

| tersimpan dari 1 Juli, 16:30 WIB. |

|------------------------------------------|

| |

| +-------------------------------------+ |

| | [Gambar] Ekspor Kopi Gayo Meningkat | |

| | <1 Juli 2025> | |

| +-------------------------------------+ |

| |

| +-------------------------------------+ |

| | [Gambar] Festival Danau Laut Tawar | |

| | Siap Digelar | |

| | <1 Juli 2025> | |

| +-------------------------------------+ |

| |

+------------------------------------------+

Kita lanjutkan ke nomor 38 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita lanjutkan. Aplikasi ini tingkat kesulitannya sudah berada di perbatasan antara menengah dan sulit.

**38. Aplikasi Transportasi Online (sisi penumpang, dengan peta real-time)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi dari sisi penumpang untuk memesan layanan transportasi (seperti ojek atau taksi) dan melacak posisi driver di peta secara langsung dari saat penjemputan hingga tiba di tujuan.
* **Fitur Inti (Level Menengah-Sulit)**:
  1. **Pemilihan Lokasi & Tujuan**: Pengguna bisa memilih titik jemput dan tujuan di peta interaktif.
  2. **Estimasi Biaya**: Menampilkan estimasi biaya perjalanan sebelum memesan.
  3. **Pemesanan & Penugasan Driver**: Pengguna membuat pesanan, dan sistem (yang disimulasikan) akan menugaskan driver terdekat.
  4. **Pelacakan Peta Real-time**: Fitur kunci. Setelah driver ditugaskan, pengguna bisa melihat ikon driver bergerak di peta menuju lokasi penjemputan, dan kemudian menuju lokasi tujuan.
* **Logika & State (Level Menengah-Sulit)**:
  1. **Integrasi Peta Lanjutan**: Menggunakan Maps\_flutter secara ekstensif. Ini termasuk:
     + Menampilkan marker untuk pengguna dan driver.
     + **Menggambar rute (polyline)** di antara dua titik (biasanya memerlukan API tambahan seperti Google Directions API).
     + Memperbarui posisi marker driver secara dinamis.
  2. **Backend (Firestore) & Real-time Location**:
     + **Struktur Data**: Koleksi rides di mana setiap dokumen menyimpan status perjalanan, lokasi jemput/tujuan (GeoPoint), passengerId, dan driverId.
     + **Lokasi Driver**: Dokumen ride juga akan menyimpan driver\_location (GeoPoint) yang terus diperbarui oleh aplikasi driver (yang disimulasikan).
  3. **Real-time Listener**: Aplikasi penumpang menggunakan StreamBuilder untuk mendengarkan perubahan pada dokumen ride mereka. Setiap kali driver\_location diperbarui di Firestore, posisi marker driver di peta juga akan diperbarui.
* **Tantangan Potensial (Menengah-Sulit)**:
  1. Interaksi peta yang kompleks (memilih titik, menggambar rute).
  2. Mengelola state dari sebuah perjalanan yang memiliki banyak tahap (mencari, dijemput, diantar, selesai).
  3. Logika untuk memperbarui lokasi di peta secara mulus tanpa membuat aplikasi terasa berat.
* **Paket Flutter yang Mungkin Digunakan**: **Maps\_flutter**, cloud\_firestore, geolocator (mendapatkan lokasi awal pengguna), flutter\_polyline\_points (untuk membantu menggambar rute).

**Tampilan UI (Mode Karakter)**

**Layar Pelacakan Driver (Saat Driver Menuju Titik Jemput)**

*(Konteks: Memesan ojek untuk pulang kerja di sore hari)*

+------------------------------------------+

| Ojek Aceh Cepat |

+==========================================+

| |

| /------------Peta Area-------------\ |

| | | |

| | (Anda 📍) | |

| | . | |

| | . | |

| | . (🏍️) --> | |

| | . | |

| \--------------------------------/ |

| |

+==========================================+

| Status: Driver sedang menuju lokasimu! |

| Estimasi tiba: 4 menit |

|------------------------------------------|

| 👤 Bang Mamat - Vario BL 5678 AA |

| ⭐️ 4.8 [ Chat 💬 ] [ Telepon 📞] |

+------------------------------------------+

*(Keterangan: (🏍️) Posisi driver, (📍) Posisi Anda, (...) Rute yang ditempuh)*

Kita lanjutkan ke nomor 39 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**39. Aplikasi Penjualan Tiket Acara/Bioskop**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi di mana pengguna dapat melihat jadwal film atau acara, memilih tempat duduk dari denah, dan melakukan pemesanan tiket.
* **Fitur Inti (Level Menengah)**:
  1. **Jadwal Film/Acara**: Menampilkan daftar film yang sedang tayang atau acara yang tersedia, beserta pilihan tanggal dan jam pertunjukan.
  2. **Denah Tempat Duduk (Seat Map)**: Ini adalah fitur kunci. Menampilkan layout kursi dari bioskop atau venue secara visual. Pengguna bisa melihat kursi mana yang sudah terisi dan memilih kursi yang masih kosong.
  3. **Proses Pemesanan**: Setelah memilih kursi, pengguna dapat melanjutkan ke halaman konfirmasi untuk menyelesaikan pemesanan.
  4. **Tiket Digital**: Setelah pemesanan berhasil, aplikasi menghasilkan tiket digital (misalnya dalam bentuk Kode QR) yang disimpan di akun pengguna.
* **Logika & State (Level Menengah)**:
  1. **Backend (Firestore)**:
     + **Struktur Data**: Memerlukan desain yang cermat. Misalnya, koleksi showtimes. Setiap dokumen jadwal pertunjukan akan berisi movie\_id, timestamp, dan sebuah Map atau List yang merepresentasikan status setiap kursi (misal: seats: {'A1': 'available', 'A2': 'booked', ...}).
  2. **Transaksi Pemesanan Kursi**: Untuk mencegah dua orang memesan kursi yang sama persis, proses ini **wajib** menggunakan **Firebase Transactions**. Transaksi akan membaca status kursi, dan jika masih tersedia, akan mengubah statusnya menjadi "terisi" dan menyelesaikan pesanan dalam satu operasi atomik.
  3. **UI Interaktif**: Membuat widget denah kursi yang bisa di-tap, di-zoom, dan di-pan.
* **Tantangan Potensial (Menengah)**:
  1. **Membangun UI Denah Kursi**: Membuat widget kustom untuk denah kursi yang interaktif.
  2. **Mengimplementasikan Transaksi Firestore**: Memastikan tidak ada *double-booking* dengan runTransaction.
  3. Merancang skema database yang efisien untuk mengelola ribuan kursi di banyak jadwal pertunjukan.
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, **qr\_flutter** (untuk membuat QR code tiket), **interactive\_viewer** (untuk membuat denah kursi bisa di-zoom/pan).

**Tampilan UI (Mode Karakter)**

**Layar Pemilihan Kursi (Seat Selection)**

*(Konteks: Memesan tiket nonton di bioskop Hermes XXI Banda Aceh)*

+------------------------------------------+

| [<-] Pilih Kursi: Badarawuhi (19:00) |

+==========================================+

| |

| ======================= |

| [ L A Y A R / SCREEN] |

| ======================= |

| |

| A [ ] [ ] (x) (x) [ ] [ ] [ ] [ ] |

| B [ ] [ ] (x) (x) [ ] [ ] (x) (x) |

| C [ ] [s] [s] (x) [ ] [ ] [ ] [ ] |

| D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] |

| |

|------------------------------------------|

| [ ] Tersedia (x) Terisi [s] Pilihan Anda|

|------------------------------------------|

| |

| Kursi: C2, C3 Total: Rp 80.000 |

| |

| [ Lanjut ke Pembayaran ] |

+------------------------------------------+

Kita lanjutkan ke nomor 40, yang terakhir dari kategori menengah, jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, ini adalah nomor terakhir untuk kategori Menengah.

**40. Aplikasi Klien untuk Penyedia Penyimpanan Cloud (misal: Google Drive/Dropbox)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi file manager pihak ketiga yang memungkinkan pengguna untuk login ke akun penyimpanan cloud mereka (seperti Google Drive), menelusuri file dan folder, serta melakukan operasi dasar seperti mengunggah dan mengunduh file.
* **Fitur Inti (Level Menengah-Sulit)**:
  1. **Autentikasi OAuth 2.0**: Ini adalah tantangan kunci. Pengguna harus bisa login dengan aman ke akun Google/Dropbox mereka melalui alur otentikasi standar industri (OAuth 2.0) untuk memberikan izin akses kepada aplikasi.
  2. **Browser File/Folder**: Menampilkan struktur direktori dari cloud storage, memungkinkan pengguna untuk bernavigasi masuk dan keluar dari folder.
  3. **Operasi File**:
     + **Mengunduh (Download)**: Menyimpan file dari cloud ke perangkat pengguna.
     + **Mengunggah (Upload)**: Memilih file dari perangkat dan mengunggahnya ke folder tertentu di cloud.
     + **Melihat Detail**: Menampilkan metadata file (ukuran, tipe, tanggal modifikasi).
* **Logika & State (Level Menengah-Sulit)**:
  1. **Integrasi API Pihak Ketiga**: Bekerja secara ekstensif dengan API resmi dari penyedia layanan, misalnya **Google Drive API**.
     + **Konfigurasi Proyek Cloud**: Mendaftarkan aplikasi di Google Cloud Console untuk mendapatkan kredensial OAuth 2.0 (Client ID).
     + **Alur Otentikasi**: Menggunakan paket seperti google\_sign\_in untuk mengelola proses login dan mendapatkan token akses yang diperlukan untuk setiap panggilan API.
  2. **Panggilan API**: Melakukan berbagai panggilan HTTP ke endpoint API untuk files.list (melihat daftar file), files.get (mengunduh), dan files.create (mengunggah).
* **Tantangan Potensial (Menengah-Sulit)**:
  1. **Implementasi OAuth 2.0**: Memahami dan mengimplementasikan alur otentikasi yang kompleks ini seringkali menjadi bagian tersulit.
  2. Bekerja dengan dokumentasi API yang sangat luas dan teknis.
  3. Menangani upload dan download file biner, termasuk menampilkan progresnya.
* **Paket Flutter yang Mungkin Digunakan**: **google\_sign\_in**, **googleapis** & **googleapis\_auth** (untuk interaksi dengan Google API), file\_picker (memilih file lokal), path\_provider (menyimpan file yang diunduh).

**Tampilan UI (Mode Karakter)**

**Layar File Browser**

*(Konteks: Mengakses file tugas kuliah di Google Drive)*

+------------------------------------------+

| Drive > Skripsi > Bab 2 [👤] |

+==========================================+

| |

| [📁] Gambar Penelitian |

| ---------------------------------------- |

| [📄] Draft Bab 2.docx |

| <2.1 MB | Diedit kemarin> |

| ---------------------------------------- |

| [📈] Data Kuesioner.xlsx |

| <500 KB | 28 Juni 2025> |

| ---------------------------------------- |

| [🖼️] Foto Observasi 01.jpg |

| <3.5 MB | 28 Juni 2025> |

| |

| |

| [↑ Upload] |

+------------------------------------------+

*(Saat file diketuk, akan muncul opsi seperti "Download", "Ubah Nama", "Hapus")*

Ini adalah akhir dari kategori menengah. Beri tahu saya jika Anda siap untuk tantangan sesungguhnya di kategori **Lanjutan / Sulit**, dimulai dengan nomor 41.

Yahok

Tampilkan alur berpikir

Baik, kita mulai. Selamat datang di kategori **Lanjutan / Sulit**. Aplikasi di level ini melibatkan arsitektur yang kompleks, pemrosesan data yang berat, UI kustom tingkat tinggi, atau integrasi native yang mendalam.

Mari kita mulai dengan nomor 41.

**41. Aplikasi Penyunting Video Profesional (Video Editor)**

**Brainstorming**

* **Tujuan Utama**: Membuat editor video non-linear (NLE) fungsional di perangkat mobile, memungkinkan pengguna memotong, menggabungkan klip, menambahkan trek audio, teks, dan mengekspor hasilnya.
* **Fitur Inti (Level Lanjutan)**:
  1. **Timeline Multitrek**: Ini adalah inti dari aplikasi. Sebuah UI kompleks yang menampilkan beberapa trek untuk video, audio, dan teks. Pengguna bisa mengatur ulang klip di timeline.
  2. **Operasi Klip**: Fungsi dasar seperti memotong (*trim*), membelah (*split*), dan menghapus klip video atau audio.
  3. **Menambahkan Lapisan**: Kemampuan untuk menumpuk elemen, seperti menambahkan teks di atas video atau menambahkan musik latar di bawah audio asli video.
  4. **Pratinjau Real-time**: Menampilkan pratinjau video yang sedang diedit.
  5. **Proses Rendering (Export)**: Menggabungkan semua trek dan editan menjadi satu file video akhir (misalnya, MP4).
* **Logika & State (Level Lanjutan)**:
  1. **Integrasi Native Mendalam**: Pemrosesan video (decoding, filtering, encoding) sangat intensif dan **tidak mungkin** dilakukan secara efisien dengan Dart murni. Ini memerlukan penggunaan pustaka native yang kuat seperti **FFmpeg**.
     + **Tantangan Utama**: Mengompilasi FFmpeg untuk Android dan iOS dan mengintegrasikannya ke dalam proyek Flutter.
  2. **Komunikasi Dart-Native**: Menggunakan **Flutter FFI (Foreign Function Interface)** atau Platform Channels untuk mengirim perintah dari UI Flutter (misal: "potong video X dari detik 5 ke 10") ke kode native yang akan menjalankan perintah FFmpeg.
  3. **Manajemen State Kompleks**: State dari timeline (daftar semua klip, posisinya, durasinya, efek, teks, dll.) sangatlah rumit dan harus dikelola dengan sangat hati-hati.
  4. **UI Kustom Tingkat Tinggi**: Widget timeline harus dibuat dari awal menggunakan CustomPainter dan GestureDetector untuk menangani scrolling, zooming, dan interaksi drag-and-drop pada klip.
* **Tantangan Potensial (Lanjutan)**:
  1. **FFmpeg**: Mengintegrasikan dan menggunakan FFmpeg adalah tantangan rekayasa perangkat lunak yang sangat besar.
  2. **Performa**: Memastikan UI tetap responsif saat menangani file video besar dan pratinjau.
  3. **Manajemen Memori & File**: Menangani file media berukuran besar secara efisien.
* **Paket Flutter yang Mungkin Digunakan**: **ffmpeg\_kit\_flutter** (menyederhanakan integrasi FFmpeg, namun tetap kompleks), file\_picker, video\_player (untuk pratinjau).

**Tampilan UI (Mode Karakter)**

**Layar Editor Utama**

*(Konteks: Mengedit video vlog "Jelajah Kuliner Medan")*

+------------------------------------------+

| Proyek: Kuliner Medan [Ekspor 📤] |

+==========================================+

| |

| +------------------------------+ |

| | | |

| | [ Jendela Pratinjau Video ] | |

| | | |

| +------------------------------+ |

| | |

|------------------|-----------------------| <- Playhead & Timeline

| Trek Video [Klip1][ Klip 2 ][ Klip 3 ] |

| Trek Audio [ Musik Latar ... ] |

| Trek Teks [ Judul ][ ][ Penutup] |

|------------------------------------------|

| |

| [Potong ✂️] [Filter ✨] [Teks T] [Audio 🎵] |

+------------------------------------------+

Ini adalah tantangan yang sangat besar. Beri tahu saya jika Anda siap melanjutkan ke nomor 42.

Yahok

Tampilkan alur berpikir

Baik, kita lanjutkan ke tantangan berikutnya.

**42. Aplikasi Produksi Musik (Mobile DAW)**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah Digital Audio Workstation (DAW) mobile yang fungsional, memungkinkan pengguna untuk membuat musik dari awal dengan menggabungkan beberapa trek instrumen MIDI, sampel audio, dan rekaman.
* **Fitur Inti (Level Lanjutan)**:
  1. **Timeline Arranger**: Tampilan utama untuk menyusun "klip" MIDI atau audio di beberapa trek secara horizontal.
  2. **Piano Roll / MIDI Editor**: Sebuah antarmuka grid untuk menulis, mengedit, dan menghapus not-not melodi atau drum.
  3. **Instrumen Virtual & Synthesizer**: Mesin audio internal yang dapat menghasilkan suara (seperti piano, bass, atau synth) berdasarkan data MIDI.
  4. **Rekaman Audio**: Kemampuan untuk merekam suara dari mikrofon perangkat ke sebuah trek audio.
  5. **Mixer**: Kontrol dasar untuk volume dan *panning* (posisi stereo kiri/kanan) untuk setiap trek.
* **Logika & State (Level Lanjutan)**:
  1. **Pemrosesan Audio Real-time (DSP)**: Ini adalah **tantangan terberat**. Semua operasi audio (sintesis suara, mixing, efek) harus berjalan dengan **latensi sangat rendah**, yang tidak mungkin dicapai dengan Dart murni. Ini mutlak memerlukan kode **C/C++**.
  2. **Mesin Audio Kustom**: Developer pada dasarnya perlu membangun atau mengintegrasikan sebuah *custom audio engine*. Pustaka seperti Oboe (Android), CoreAudio (iOS), atau framework cross-platform seperti JUCE atau Superpowered sering digunakan untuk ini.
  3. **Flutter FFI (Foreign Function Interface)**: FFI adalah jembatan performa tinggi antara UI Flutter (yang mengelola state, UI timeline, piano roll) dan mesin audio C/C++ yang melakukan semua pekerjaan berat.
  4. **Sinkronisasi Audio-Visual**: Memastikan *playhead* (penanda waktu) di UI bergerak sinkron sempurna dengan audio yang diputar oleh mesin native adalah hal yang sangat rumit.
* **Tantangan Potensial (Lanjutan)**:
  1. **Low-latency Audio**: Ini adalah "cawan suci" dalam pengembangan audio mobile dan sangat sulit dicapai.
  2. **Pemrograman C/C++**: Memerlukan keahlian di luar ekosistem Dart/Flutter.
  3. **Manajemen State Proyek**: State sebuah proyek musik sangat kompleks (data MIDI, posisi klip, parameter mixer, dll).
* **Paket Flutter yang Mungkin Digunakan**: Sangat sedikit. Proyek ini 95% bergantung pada FFI dan kode native kustom.

**Tampilan UI (Mode Karakter)**

**Tampilan Arranger Utama**

*(Konteks: Membuat komposisi musik yang memadukan elemen modern dan tradisional Aceh)*

+------------------------------------------+

| Proyek: Etnik Fusion [▶] [■] [●] |

+==========================================+

| | | <- Playhead

| -----------|-----------------------------| <- Timeline

| T1: Drums | [----- Pola Ritme 1 ----] |

| T2: Bass | [----- Pola Bass A -----] |

| T3: Rapa'i | [ Sampel Rapa'i ] |

| T4: Melodi | [ Melodi Utama (MIDI) ] |

| T5: Vokal | [--- Rekaman Vokal ---] |

| |

|------------------------------------------|

| Instrumen: [Synth] | Mixer | Efek [FX] |

+------------------------------------------+

*(Saat klip MIDI "Melodi Utama" di-tap, akan membuka tampilan Piano Roll)*

Ini adalah salah satu proyek paling kompleks yang bisa dibayangkan. Mari kita lanjutkan ke nomor 43 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan ke level berikutnya.

**43. Aplikasi Kolaborasi Papan Tulis Real-time (seperti Miro)**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah kanvas digital tak terbatas (*infinite canvas*) di mana beberapa pengguna bisa berkolaborasi secara bersamaan untuk menggambar, menulis, dan menyusun ide dalam bentuk visual.
* **Fitur Inti (Level Lanjutan)**:
  1. **Kanvas Interaktif**: Pengguna bisa melakukan *pan* (geser) dan *zoom* (perbesar/perkecil) pada kanvas dengan bebas.
  2. **Kolaborasi Multi-pengguna Real-time**: Ini adalah inti dari aplikasi. Setiap aksi dari satu pengguna (menggambar garis, memindahkan objek) harus langsung terlihat oleh semua pengguna lain di papan yang sama dengan latensi minimal.
  3. **Beragam Alat (Tools)**: Menyediakan alat seperti pena untuk menggambar bebas, teks, bentuk dasar (kotak, lingkaran), dan catatan tempel (*sticky notes*).
  4. **Manipulasi Objek**: Pengguna bisa memilih, memindahkan, mengubah ukuran, dan menghapus objek apa pun di kanvas.
* **Logika & State (Level Lanjutan)**:
  1. **Networking Real-time Latensi Rendah**:
     + **WebSocket**: Untuk aplikasi sekelas ini, *real-time listener* dari Firestore mungkin tidak cukup cepat. Pendekatan yang lebih umum adalah menggunakan backend kustom dengan **WebSockets** untuk komunikasi dua arah yang instan. Server akan menerima event dari satu klien dan menyiarkannya ke semua klien lain.
     + **Resolusi Konflik**: Perlu strategi untuk menangani kasus saat dua pengguna mengedit objek yang sama. Ini bisa berupa *last-write-wins*, atau menggunakan struktur data canggih seperti **CRDTs (Conflict-free Replicated Data Types)**.
  2. **UI Kanvas Kustom**: Seluruh antarmuka kanvas harus dibangun dari nol menggunakan CustomPainter dan GestureDetector. Logika untuk me-render ribuan objek, menangani transformasi (pan, zoom), dan deteksi sentuhan pada objek menjadi sangat rumit.
  3. **Optimasi Performa**: Dengan potensi ribuan objek di kanvas, performa rendering menjadi krusial. Perlu teknik optimasi seperti *object culling* (hanya me-render objek yang terlihat di layar) dan *layering*.
* **Tantangan Potensial (Lanjutan)**:
  1. Membangun dan mengelola backend WebSocket yang andal.
  2. Merancang sistem sinkronisasi data yang efisien dan bebas konflik.
  3. Membangun renderer kanvas 2D yang performan dari awal.
* **Paket Flutter yang Mungkin Digunakan**: web\_socket\_channel, interactive\_viewer (sebagai dasar pan/zoom), namun sebagian besar logika adalah kustom.

**Tampilan UI (Mode Karakter)**

**Tampilan Papan Kolaboratif**

*(Konteks: Tim startup di Banda Aceh sedang brainstorming ide aplikasi)*

+------------------------------------------+

| Papan: Ide Aplikasi Wisata [👤A][👤F] |

+==========================================+

| |

| +-------------+ |

| | Sticky Note:| --- (dihubungkan)|

| | - Fitur Peta| |

| +-------------+ |

| | |

| | (Fatimah's cursor)

| | |

| +--------------------------+ |

| | Teks: Target Pengguna | |

| | - Wisatawan Domestik | |

| | - Turis Mancanegara | |

| +--------------------------+ |

| |

+------------------------------------------+

| [Pilih 🖐️] [Pena ✒️] [Teks T] [Bentuk 〇] |

+------------------------------------------+

*(Keterangan: [👤A][👤F] adalah avatar Ahmad & Fatimah yang sedang online. Kursor mereka juga terlihat di kanvas)*

Ini adalah proyek yang sangat menantang di sisi backend dan frontend. Mari lanjutkan ke nomor 44 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan ke tantangan berikutnya.

**44. Aplikasi Penyunting Foto Profesional (seperti Lightroom Mobile)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi penyunting foto *non-destruktif* yang kuat, memberikan kontrol granular kepada pengguna atas parameter gambar seperti pencahayaan, warna, dan detail.
* **Fitur Inti (Level Lanjutan)**:
  1. **Pengeditan Non-Destruktif**: Perubahan yang dibuat (misalnya, menaikkan exposure) tidak diterapkan secara permanen pada gambar asli. Aplikasi hanya menyimpan daftar "resep" atau langkah-langkah editan.
  2. **Kontrol Pencahayaan Lanjutan**: Slider untuk mengatur *Exposure, Contrast, Highlights, Shadows, Whites, Blacks*.
  3. **Kontrol Warna Presisi**: Fitur seperti *White Balance, Tint, Vibrance, Saturation*, dan yang paling kompleks, **Kurva Nada (Tone Curve)** dan panel **HSL (Hue, Saturation, Luminance)**.
  4. **Penyesuaian Selektif**: Kemampuan untuk menerapkan editan hanya pada area tertentu dari gambar menggunakan kuas (*brush*) atau gradien linier/radial.
* **Logika & State (Level Lanjutan)**:
  1. **Pemrosesan Gambar di GPU**: Untuk performa tinggi (agar pratinjau terasa instan saat slider digerakkan), semua operasi pengolahan gambar harus dieksekusi di **GPU** menggunakan *shaders*. Ini memerlukan:
     + **Integrasi Native**: Menggunakan **OpenGL** atau **Metal** melalui kode native (C/C++).
     + **Shader Programming (GLSL)**: Menulis *fragment shaders* kustom untuk setiap efek (misalnya, shader untuk menyesuaikan kontras, shader untuk kurva, dll).
  2. **Komunikasi Dart-Native**: Menggunakan **Flutter FFI** untuk mengirim parameter editan (nilai slider, titik kurva) dari UI Flutter ke kode native yang akan me-render gambar dengan *shader*.
  3. **Manajemen State Editan**: Aplikasi harus menyimpan seluruh "tumpukan" editan (daftar semua penyesuaian yang dibuat pengguna) dalam sebuah state yang kompleks. Setiap kali ada perubahan, gambar di-render ulang dari awal dengan seluruh tumpukan editan.
* **Tantangan Potensial (Lanjutan)**:
  1. **Pemrograman GPU (Shaders)**: Ini adalah keahlian yang sangat spesifik dan berada di luar lingkup pengembangan aplikasi pada umumnya.
  2. **Performa Rendering**: Memastikan pratinjau tetap mulus saat parameter diubah secara real-time.
  3. Membangun UI kustom yang kompleks seperti widget *Tone Curve*.
* **Paket Flutter yang Mungkin Digunakan**: Sedikit sekali. Mungkin image untuk operasi dasar, tetapi untuk fitur inti yang performan, solusinya adalah **kustom dengan FFI** atau paket seperti **flutter\_gl** untuk akses ke API grafis.

**Tampilan UI (Mode Karakter)**

**Layar Editor Utama**

*(Konteks: Mengedit foto sunset di Pantai Lampuuk, Aceh Besar)*

+------------------------------------------+

| [<- Impor] Editor Foto [Ekspor 📤] |

+==========================================+

| |

| |

| |

| [ Gambar sunset di pantai ] |

| |

| |

| |

| |

+------------------------------------------+

| Panel: CAHAYA |

|------------------------------------------|

| Exposure <-------O---------------> +0.3 |

| Contrast <---------O-------------> +10 |

| Highlights <----O------------------> -25 |

| |

| [Cahaya] (Warna) (Efek) (Selektif) |

+------------------------------------------+

Ini adalah proyek yang sangat mendalam di bidang pengolahan grafis. Mari lanjutkan ke nomor 45 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan ke tantangan berikutnya.

**45. Aplikasi Komunikasi Tim dengan Panggilan Video (seperti Slack/Discord)**

**Brainstorming**

* **Tujuan Utama**: Membuat platform komunikasi tim yang lengkap, menggabungkan chat berbasis kanal (*channel-based chat*), pesan langsung, dan yang terpenting, fungsionalitas panggilan suara dan video secara real-time.
* **Fitur Inti (Level Lanjutan)**:
  1. **Struktur Tim & Kanal**: Pengguna bisa bergabung dalam tim/server yang memiliki banyak kanal teks untuk diskusi terpisah.
  2. **Chat Real-time**: Fungsionalitas chat standar yang sinkron antar semua anggota tim.
  3. **Panggilan Suara & Video (WebRTC)**: Ini adalah **inti dari kesulitan aplikasi ini**. Pengguna bisa memulai panggilan suara atau video, baik satu lawan satu maupun dalam sebuah grup/kanal.
  4. **User Presence**: Menampilkan status online/offline/idle dari anggota tim.
  5. **Notifikasi Push**: Mengirim notifikasi saat pengguna disebut (*mention*) atau menerima pesan.
* **Logika & State (Level Lanjutan)**:
  1. **WebRTC (Web Real-Time Communication)**: Teknologi ini adalah standar untuk komunikasi *peer-to-peer* (P2P) dan sangat kompleks untuk diimplementasikan. Melibatkan tiga komponen utama:
     1. **Signaling Server**: Sebuah server perantara (yang harus Anda bangun, misalnya dengan WebSocket) untuk membantu dua klien "berkenalan" dan bertukar informasi koneksi.
     2. **STUN/TURN Servers**: Server publik untuk membantu klien menemukan alamat IP publik mereka dan menembus *firewall* (NAT Traversal).
     3. **Media Streams**: Mengambil aliran data dari kamera dan mikrofon, lalu mengirimkannya ke klien lain.
  2. **Backend**: Memerlukan kombinasi **Firestore** (untuk menyimpan data chat, kanal, dan user), **Firebase Messaging** (untuk notifikasi), dan **server kustom** untuk *signaling* WebRTC.
* **Tantangan Potensial (Lanjutan)**:
  1. **Implementasi WebRTC**: Ini adalah bidang rekayasa perangkat lunak yang sangat terspesialisasi dan sulit.
  2. **Membangun & Mengelola Backend Kompleks**: Terutama server signaling.
  3. **Performa & Kualitas Panggilan**: Memastikan panggilan video berjalan lancar di berbagai kondisi jaringan.
* **Paket Flutter yang Mungkin Digunakan**: **flutter\_webrtc** (menyediakan *binding* ke API WebRTC native, namun tetap sangat kompleks untuk digunakan), cloud\_firestore, firebase\_messaging.

**Tampilan UI (Mode Karakter)**

**Tampilan Kanal Chat Teks**

*(Konteks: Tim startup di Banda Aceh berdiskusi di kanal proyek)*

+------------------------------------------+

| ☰ #proyek-kopi-aceh [📞][📹] |

+==========================================+

| |

| Ahmad: |

| Desain UI untuk halaman produk sudah |

| siap di Figma. Silakan dicek. |

| |

| Fatimah: |

| Siap, sedang aku periksa. |

| |

| Anda: |

| Oke, setelah ini kita adakan panggilan |

| video singkat untuk review ya. |

| |

| [Ketik pesan...] [Kirim] |

+------------------------------------------+

*(Keterangan: [📞] Mulai Panggilan Suara, [📹] Mulai Panggilan Video)*

**Tampilan Panggilan Video Aktif**

+------------------------------------------+

| Panggilan Video: #proyek-kopi-aceh |

+==========================================+

| |

| +-------------------------------+ |

| | [ Video Ahmad ] | |

| +-------------------------------+ |

| |

| +----------------+ +----------------+ |

| | [Video Fatimah]| | [ Video Anda ] | |

| +----------------+ +----------------+ |

| |

| |

| [Hening 🔇] [Kamera 🔁] [Tutup Panggilan 📞] |

+------------------------------------------+

Ini adalah proyek *full-stack* yang sangat kompleks dengan tantangan besar di *networking real-time*. Mari kita lanjutkan ke nomor 46 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita masuki tantangan berikutnya yang lebih dalam.

**46. Sistem Transportasi Online Lengkap (Aplikasi Pengemudi & Penumpang)**

**Brainstorming**

* **Tujuan Utama**: Membangun **ekosistem** layanan transportasi online secara keseluruhan. Ini bukan hanya satu aplikasi, melainkan **dua aplikasi terpisah** (satu untuk penumpang, satu untuk pengemudi) yang diatur oleh sebuah backend yang canggih.
* **Fitur Inti (Level Lanjutan)**:
  1. **Aplikasi Penumpang**: Semua fitur dari aplikasi level menengah (No. 38), termasuk memesan dan melacak.
  2. **Aplikasi Pengemudi**:
     + Beralih status "Online/Offline" untuk menerima atau berhenti menerima pesanan.
     + Menerima notifikasi pesanan baru di sekitar mereka.
     + Pilihan untuk **Menerima** atau **Menolak** pesanan dalam batas waktu tertentu.
     + Antarmuka navigasi peta untuk menuju titik jemput, lalu ke tujuan.
     + Kemampuan untuk memperbarui status perjalanan (misal: "Sudah menjemput", "Selesai").
  3. **Backend Canggih**:
     + **Sistem Penugasan (Dispatch)**: Algoritma di backend untuk menemukan pengemudi terdekat yang aktif dan menugaskan pesanan kepada mereka.
     + **Pelacakan Lokasi Real-time Skala Besar**: Secara efisien melacak lokasi geografis semua pengemudi yang sedang online.
     + **Penghitungan Harga**: Algoritma untuk menghitung biaya berdasarkan jarak dan mungkin faktor lain.
* **Logika & State (Level Lanjutan)**:
  1. **Geo-querying di Backend**: Ini adalah tantangan besar. Memerlukan penggunaan teknologi atau layanan khusus seperti **GeoFire** yang berjalan di atas Firestore/Realtime Database untuk bisa melakukan query seperti "temukan semua driver online dalam radius 2 km dari titik A".
  2. **Manajemen Siklus Hidup Pesanan**: Sebuah pesanan memiliki banyak status (mencari, diterima, menjemput, mengantar, selesai, dibatalkan) yang harus dikelola secara andal di backend dan dipantulkan ke kedua aplikasi secara real-time.
  3. **Komunikasi Dua Arah**: Dibutuhkan komunikasi yang solid antara kedua aplikasi melalui backend. Misalnya, saat pengemudi menekan "Terima", status di aplikasi penumpang harus langsung berubah.
* **Tantangan Potensial (Lanjutan)**:
  1. **Arsitektur Backend**: Merancang dan membangun backend yang skalabel untuk menangani ribuan query lokasi dan pesanan secara bersamaan.
  2. **Algoritma Dispatch**: Membuat logika penugasan yang adil dan efisien adalah masalah rekayasa yang kompleks.
  3. **Konsumsi Baterai**: Aplikasi pengemudi harus mengirim pembaruan lokasi secara efisien agar tidak menguras baterai perangkat.
* **Paket Flutter yang Mungkin Digunakan**: Maps\_flutter, cloud\_firestore, firebase\_functions (untuk logika backend), dan pustaka geo-query seperti geoflutterfire.

**Tampilan UI (Mode Karakter)**

**Tampilan Aplikasi Pengemudi (Saat Menerima Pesanan)**

+------------------------------------------+

| Status: Online 🟢 [Profil] [Riwayat] |

+==========================================+

| |

| /------------Peta Area-------------\ |

| | | |

| | (Anda 🏍️) | |

| | . | |

| | . (Penumpang 👤) | |

| \--------------------------------/ |

| |

| +-------------------------------------+ |

| | PESANAN BARU! | |

| | Jemput: 1.5 km | Tujuan: 4 km | |

| | Potensi Pendapatan: Rp 15.000 | |

| | | |

| | ( TOLAK ❌ ) [ TERIMA ✔️ (12s)] | |

| +-------------------------------------+ |

+------------------------------------------+

*(Ini adalah sisi lain dari ekosistem, menunjukkan apa yang dilihat oleh pengemudi)*

Membangun kedua sisi platform ini adalah tugas rekayasa yang masif. Mari kita lanjutkan ke nomor 47 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**47. Kloning Media Sosial dengan Stories & Filter Video Real-time**

**Brainstorming**

* **Tujuan Utama**: Membuat kloning aplikasi media sosial modern seperti Instagram atau TikTok, dengan fokus pada fitur-fitur komputasi intensif seperti Stories yang berumur pendek dan filter video/AR yang diterapkan secara real-time.
* **Fitur Inti (Level Lanjutan)**:
  1. **Feed & Interaksi**: Fitur dasar media sosial (post, like, comment).
  2. **Stories Ephemeral**: Pengguna bisa memposting konten (foto/video) yang akan otomatis terhapus setelah 24 jam. UI untuk melihat stories (mengetuk untuk maju, menahan untuk jeda) harus dibuat kustom.
  3. **Filter Video & AR Real-time**: Ini adalah **tantangan terberat**. Saat pengguna merekam video, aplikasi harus bisa:
     + Menerapkan filter warna (seperti filter Instagram) secara langsung.
     + Menerapkan filter Augmented Reality (AR), seperti topeng, kacamata, atau distorsi wajah, yang melacak pergerakan wajah pengguna secara real-time.
* **Logika & State (Level Lanjutan)**:
  1. **Backend untuk Stories**: Memerlukan mekanisme di backend (misalnya, *Time-to-Live (TTL) policy* di Firestore atau *Cloud Function* yang berjalan secara berkala) untuk menghapus dokumen Stories yang sudah kedaluwarsa.
  2. **Pipeline Pemrosesan Grafis Real-time**:
     1. **Akses Kamera**: Mengambil *stream* frame video mentah dari kamera.
     2. **Deteksi Wajah (Machine Learning)**: Setiap frame video dilewatkan ke model Machine Learning (seperti Google ML Kit atau TensorFlow Lite) untuk mendeteksi *landmark* wajah (posisi mata, hidung, mulut).
     3. **Rendering AR**: Berdasarkan posisi landmark wajah, objek 2D/3D (misalnya, gambar kacamata) di-render di atas frame video. Ini memerlukan rendering di GPU menggunakan **OpenGL/Metal** melalui FFI untuk performa.
* **Tantangan Potensial (Lanjutan)**:
  + **Integrasi Machine Learning**: Menjalankan model deteksi wajah pada stream video di perangkat secara efisien adalah hal yang sangat sulit.
  + **Pemrograman GPU**: Menulis *shaders* (GLSL) untuk filter warna atau me-render objek AR adalah keahlian tingkat lanjut.
  + **Performa Pipeline**: Memastikan seluruh pipeline dari kamera -> ML -> GPU -> layar berjalan dengan *frame rate* yang tinggi dan tanpa jeda.
* **Paket Flutter yang Mungkin Digunakan**: **camera**, **google\_ml\_kit\_face\_detection**, dan kemungkinan besar **kode native kustom (C++/Kotlin/Swift) dengan FFI** untuk menyatukan semuanya.

**Tampilan UI (Mode Karakter)**

**Layar Perekam Video dengan Filter AR**

*(Konteks: Membuat Story Instagram-like di sebuah kafe di Banda Aceh)*

+------------------------------------------+

| [X Tutup] [Musik 🎵] |

+==========================================+

| |

| |

| |

| [ Tampilan Kamera Langsung ] |

| |

| (Wajah pengguna terlihat |

| memakai kacamata hitam AR) |

| |

| |

| |

| (🔘 Tombol Rekam) |

| |

| |

| < (Filter 1) [Filter Kacamata] (Filter 3) >|

+------------------------------------------+

Ini adalah proyek yang sangat berat, menggabungkan media sosial, backend, dan *computer vision*. Mari kita lanjutkan ke nomor 48 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita lanjutkan.

**48. Aplikasi Desain Interior dengan Augmented Reality (AR)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi yang memungkinkan pengguna untuk "meletakkan" dan memvisualisasikan model furnitur 3D virtual di dalam ruangan mereka sendiri secara real-time menggunakan teknologi Augmented Reality (AR).
* **Fitur Inti (Level Lanjutan)**:
  1. **Deteksi Bidang (Plane Detection)**: Aplikasi menggunakan kamera untuk mendeteksi permukaan datar di dunia nyata, seperti lantai atau meja, sebagai tempat untuk meletakkan objek virtual.
  2. **Katalog Objek 3D**: Menyediakan galeri furnitur dalam bentuk model 3D (misalnya, kursi, meja, sofa) yang bisa dipilih pengguna.
  3. **Penempatan Objek AR**: Setelah memilih item, pengguna bisa mengetuk bidang yang terdeteksi untuk menempatkan model 3D furnitur di lokasi tersebut.
  4. **Manipulasi Objek**: Pengguna bisa berinteraksi dengan objek 3D yang sudah ditempatkan menggunakan gestur sentuhan untuk memindahkan, memutar, dan mengubah ukurannya.
* **Logika & State (Level Lanjutan)**:
  1. **Integrasi Framework AR Native**: Ini adalah inti dari aplikasi. Flutter sendiri tidak memiliki kemampuan AR. Perlu ada jembatan ke framework AR native dari setiap platform:
     + **ARCore** di Android.
     + **ARKit** di iOS.
  2. **Jembatan Flutter-Native**:
     + **Pendekatan Paket**: Menggunakan paket seperti **ar\_flutter\_plugin**. Paket ini mencoba menyediakan widget dan API Dart untuk berinteraksi dengan ARCore/ARKit, namun seringkali memerlukan konfigurasi native yang rumit.
     + **Pendekatan Kustom**: Untuk kontrol penuh, developer harus membangun *Platform View* sendiri untuk me-render tampilan AR native di dalam UI Flutter, dan menggunakan Platform Channels untuk komunikasi.
  3. **Manajemen Model 3D**: Aplikasi perlu bisa memuat dan me-render model 3D (dalam format seperti .gltf atau .usdz). Logika rendering ini ditangani oleh ARCore/ARKit.
* **Tantangan Potensial (Lanjutan)**:
  1. **Konfigurasi Proyek AR**: Menyiapkan dependensi, izin, dan pengaturan build untuk ARCore dan ARKit adalah proses yang sangat rentan error.
  2. **Interaksi Gestur di Ruang 3D**: Menerjemahkan gestur 2D di layar (geser, cubit) menjadi transformasi (posisi, rotasi, skala) objek di dalam ruang 3D.
  3. Mengelola aset model 3D.
* **Paket Flutter yang Mungkin Digunakan**: **ar\_flutter\_plugin**, arkit\_plugin. Sebagian besar pekerjaan berat terjadi di sisi native.

**Tampilan UI (Mode Karakter)**

**Layar Utama (Mode AR)**

*(Konteks: Mencoba meletakkan kursi rotan khas Aceh di ruang tamu)*

+------------------------------------------+

| AR Interior Designer |

+==========================================+

| |

| |

| |

| [ Tampilan dari kamera ponsel ] |

| [ memperlihatkan sudut ruang tamu ] |

| [ ] |

| [ ( Kursi Rotan Virtual ) ] |

| [..................................] |

| |

| |

+------------------------------------------+

| Pilih Objek: [Kursi] (Meja) (Lampu) |

+------------------------------------------+

*(Keterangan: [...] adalah area deteksi lantai, (Kursi Rotan Virtual) adalah objek 3D yang ditempatkan pengguna)*

Ini adalah proyek yang sangat bergantung pada integrasi hardware dan SDK native yang kompleks. Mari kita lanjutkan ke nomor 49 jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita lanjutkan ke tantangan berikutnya.

**49. Aplikasi Kontrol Perangkat Smart Home (IoT) Multi-Protokol**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah dasbor terpusat yang dapat mengontrol dan memonitor berbagai perangkat *Internet of Things* (IoT) atau *smart home* yang menggunakan protokol komunikasi yang berbeda-beda.
* **Fitur Inti (Level Lanjutan)**:
  1. **Dukungan Multi-Protokol**: Ini adalah **inti kesulitan**. Aplikasi harus bisa berkomunikasi dengan:
     + Perangkat berbasis **Wi-Fi** (misalnya, melalui API cloud atau protokol lokal seperti MQTT).
     + Perangkat **Bluetooth Low Energy (BLE)** (misalnya, sensor suhu, smart lock).
     + Perangkat **Zigbee/Z-Wave** (yang biasanya memerlukan komunikasi dengan *hub* atau *gateway* perantara seperti Philips Hue Bridge).
  2. **Penemuan Perangkat**: Aplikasi bisa memindai jaringan untuk menemukan perangkat baru yang kompatibel.
  3. **Dasbor Kontrol Terpadu**: Satu antarmuka untuk mengontrol semua perangkat, terlepas dari protokol yang mereka gunakan (misal: tombol on/off untuk lampu, slider kecerahan, dll.).
  4. **Otomatisasi & Rutinitas**: Memungkinkan pengguna membuat aturan sederhana (misal: "Jika sensor pintu terbuka, nyalakan lampu teras").
* **Logika & State (Level Lanjutan)**:
  1. **Integrasi Native & Komunikasi Tingkat Rendah**:
     + **BLE**: Memerlukan pemahaman mendalam tentang konsep GATT, *Services*, dan *Characteristics*. Menggunakan paket seperti flutter\_blue\_plus yang merupakan jembatan ke API Bluetooth native.
     + **Wi-Fi/API**: Melakukan panggilan HTTP ke API cloud dari produsen perangkat atau langsung ke IP lokal perangkat.
  2. **Lapisan Abstraksi (Abstraction Layer)**: Tantangan arsitektur utama adalah membuat lapisan di dalam kode yang menyembunyikan kompleksitas protokol. UI tidak perlu tahu apakah sebuah Lampu dikontrol via BLE atau Wi-Fi; ia hanya memanggil metode lampu.turnOn(). Lapisan abstraksi inilah yang akan menerjemahkan panggilan tersebut ke protokol yang benar.
  3. **Manajemen State Real-time**: State setiap perangkat harus selalu sinkron. Ini memerlukan *listeners* (misalnya, notifikasi dari karakteristik BLE) atau *polling* untuk menjaga data tetap update.
* **Tantangan Potensial (Lanjutan)**:
  1. **Bekerja dengan Hardware**: Proyek ini tidak mungkin dilakukan tanpa perangkat keras fisik untuk pengujian.
  2. **Komunikasi Nirkabel**: Menangani koneksi, diskoneksi, dan error pada berbagai protokol nirkabel.
  3. **Arsitektur Perangkat Lunak**: Merancang lapisan abstraksi yang bersih dan bisa dikembangkan adalah tugas yang sangat sulit.
* **Paket Flutter yang Mungkin Digunakan**: **flutter\_blue\_plus** (untuk BLE), http (API), mqtt\_client (jika ada perangkat MQTT), provider/riverpod.

**Tampilan UI (Mode Karakter)**

**Layar Dasbor Utama**

+------------------------------------------+

| Rumah Pintar Saya (Banda Aceh) |

+==========================================+

| Ruang Tamu |

| ---------------------------------------- |

| - Lampu Utama (Zigbee) [ Nyala ] --O-- |

| - Smart TV (Wi-Fi) [ Mati ] |

| |

| Kamar Tidur |

| ---------------------------------------- |

| - Lampu Tidur (Wi-Fi) [ Mati ] |

| - Sensor Suhu (BLE) Suhu: 25°C |

| - AC [ Nyala | 24°C ] |

| |

| Keamanan |

| ---------------------------------------- |

| - Kunci Pintu (BLE) [ Terkunci 🔒 ] |

| - Sensor Jendela [ Tertutup ] |

| |

+------------------------------------------+

*(Keterangan dalam kurung menunjukkan contoh protokol yang berbeda untuk setiap perangkat)*

Ini adalah proyek yang sangat kompleks yang menjembatani *software* dengan dunia *hardware*. Mari kita lanjutkan ke nomor 50 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita selesaikan tantangan ini.

**50. Platform E-commerce Lengkap dengan Dasbor Penjual**

**Brainstorming**

* **Tujuan Utama**: Membangun sebuah **platform marketplace** penuh. Ini berarti membuat dua aplikasi/antarmuka utama: satu untuk **pembeli** (seperti aplikasi e-commerce biasa) dan satu lagi (bisa aplikasi terpisah atau web) untuk **penjual** mengelola toko mereka.
* **Fitur Inti (Level Lanjutan)**:
  1. **Aplikasi Pembeli**: Semua fitur dari aplikasi e-commerce level menengah (No. 22).
  2. **Dasbor Penjual (Seller Dashboard)**: Ini adalah komponen baru yang sulit. Penjual harus bisa:
     + Mendaftar dan membuat toko.
     + Melakukan CRUD (Create, Read, Update, Delete) pada produk mereka sendiri.
     + Menerima dan mengelola pesanan yang masuk untuk toko mereka.
     + Memperbarui status pesanan (misalnya, "Sedang Diproses", "Dikirim").
     + Melihat statistik penjualan sederhana.
  3. **Arsitektur Multi-Penjual (Multi-Tenant)**: Backend harus dirancang untuk menangani banyak penjual, di mana data setiap penjual (produk, pesanan) terisolasi dan aman.
* **Logika & State (Level Lanjutan)**:
  1. **Backend (Firestore)**:
     + **Struktur Data**: Sangat kompleks. Dibutuhkan koleksi sellers, products (yang masing-masing memiliki sellerId), dan orders (yang memiliki buyerId dan sellerId).
     + **Aturan Keamanan (Security Rules)**: Ini menjadi **sangat krusial dan kompleks**. Aturan harus memastikan seorang penjual hanya dapat memodifikasi produk dan pesanan di tokonya sendiri, dan tidak bisa melihat data penjual lain. Pembeli hanya bisa membuat pesanan dan melihat pesanannya sendiri.
  2. **Logika Backend Tambahan**: Kemungkinan memerlukan **Cloud Functions** untuk memproses logika yang lebih rumit, seperti mengirim notifikasi pesanan baru ke penjual, atau menghitung total pendapatan.
* **Tantangan Potensial (Lanjutan)**:
  1. **Merancang arsitektur backend multi-tenant** yang skalabel dan aman.
  2. **Menulis Firestore Security Rules yang rumit** dan memastikan tidak ada celah keamanan.
  3. Mengelola siklus hidup pesanan yang melibatkan dua pihak (pembeli dan penjual).
* **Paket Flutter yang Mungkin Digunakan**: cloud\_firestore, firebase\_auth, firebase\_storage, firebase\_functions, provider/riverpod.

**Tampilan UI (Mode Karakter)**

**Tampilan Dasbor Penjual (Tab "Pesanan Baru")**

*(Konteks: Seorang penjual Kopi Gayo di "Pasar Digital Aceh" menerima pesanan baru)*

+------------------------------------------+

| Dasbor Toko: Kopi Seulawah [⚙️] |

+==========================================+

| [Pesanan Baru (3)] (Diproses) (Selesai) |

+------------------------------------------+

| |

| Pesanan #ORD-789 | 2 item | Rp 250.000 |

| > Pembeli: Ahmad (Banda Aceh) |

| > Catatan: Tolong giling halus. |

| [ Proses Pesanan ] |

| -------------------------------------- |

| Pesanan #ORD-788 | 1 item | Rp 125.000 |

| > Pembeli: Budi (Medan) |

| [ Proses Pesanan ] |

| -------------------------------------- |

| Pesanan #ORD-786 | 5 item | Rp 600.000 |

| > Pembeli: Lisa (Jakarta) |

| [ Proses Pesanan ] |

| |

+------------------------------------------+

*(Ini menunjukkan sisi lain dari platform, yaitu antarmuka yang digunakan oleh penjual)*

Membangun platform multi-sisi seperti ini adalah tantangan rekayasa yang besar. Mari kita lanjutkan ke set terakhir dari aplikasi level lanjutan, dimulai dengan nomor 51, jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita masuki set terakhir dari aplikasi level **Lanjutan / Sulit**. Ini adalah proyek-proyek yang membutuhkan keahlian rekayasa perangkat lunak yang mendalam.

**51. Aplikasi CAD Viewer dan Editor Sederhana (2D/3D)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi yang mampu memuat, menampilkan secara akurat, dan memungkinkan manipulasi dasar pada file *Computer-Aided Design* (CAD) seperti DXF (2D) atau STEP (3D).
* **Fitur Inti (Level Lanjutan)**:
  1. **Parsing File Kompleks**: Kemampuan untuk membaca dan mengurai (*parse*) skema file CAD yang sangat rumit. Tidak seperti JSON atau XML, format ini mendefinisikan geometri yang kompleks.
  2. **Rendering Grafis Performa Tinggi**: Ini adalah **tantangan utama**. Menampilkan jutaan garis, kurva, dan permukaan secara efisien tidak bisa dilakukan dengan CustomPainter. Ini memerlukan rendering langsung di GPU menggunakan **OpenGL/Vulkan/Metal**.
  3. **Interaksi Viewport**: Kontrol yang presisi untuk *pan*, *zoom*, dan *rotate* (untuk 3D) model yang besar.
  4. **Seleksi Objek & Properti**: Kemampuan untuk mengetuk sebuah entitas geometris (seperti garis atau lingkaran) untuk memilihnya dan menampilkan propertinya (panjang, radius, layer, dll.).
* **Logika & State (Level Lanjutan)**:
  1. **Pustaka Native & FFI**: Hampir tidak mungkin untuk menulis parser dan renderer CAD dari nol dalam Dart. Solusi yang realistis adalah:
     + Menggunakan pustaka C/C++ yang sudah ada (misalnya, Open Cascade, libdxfrw).
     + Mengintegrasikan pustaka ini ke dalam Flutter menggunakan **Flutter FFI (Foreign Function Interface)**.
  2. **Mesin Rendering Kustom**: Developer pada dasarnya membangun mesin rendering mini. UI Flutter hanya akan menjadi "kulit" dan wadah untuk kanvas rendering native.
  3. **Matematika Geometri Kompleks**: Melibatkan banyak sekali aljabar linear dan geometri komputasi untuk menangani transformasi, proyeksi, dan seleksi objek.
* **Tantangan Potensial (Lanjutan)**:
  1. **Bekerja dengan FFI dan kode C/C++**: Memerlukan keahlian di luar ekosistem Flutter.
  2. **Pemrograman Grafis Tingkat Rendah**: Berinteraksi langsung dengan OpenGL atau API grafis lainnya.
  3. Memahami format file CAD yang sangat teknis.
* **Paket Flutter yang Mungkin Digunakan**: Sangat sedikit. vector\_math untuk beberapa perhitungan. Sebagian besar adalah solusi **kustom sepenuhnya melalui FFI**.

**Tampilan UI (Mode Karakter)**

**Tampilan Viewer CAD 2D**

*(Konteks: Seorang arsitek di Banda Aceh mereview denah bangunan)*

+------------------------------------------+

| File: Denah\_Masjid\_Modern.dxf [⚙️] |

+==========================================+

| |

| +-----+-----+ |

| | | | |

| | +-----+ |

| +-----+-----+ |

| | | |

| | [Denah Arsitektur 2D yang Kompleks] |

| | | |

| +-----------+ |

| |

| |

+------------------------------------------+

| [Pilih 🖐️] [Ukur 📏] [Zoom 🔍] [Layer 겹] |

+------------------------------------------+

*(Saat sebuah garis dipilih, panel properti akan muncul)*

Ini adalah proyek yang lebih dekat ke rekayasa perangkat lunak teknis daripada pengembangan aplikasi biasa. Mari kita lanjutkan ke nomor 52 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita lanjutkan ke tantangan berikutnya.

**52. Aplikasi Code Editor dengan Terminal Terintegrasi**

**Brainstorming**

* **Tujuan Utama**: Membuat sebuah editor kode yang fungsional untuk perangkat mobile, dilengkapi dengan fitur-fitur penting seperti *syntax highlighting* dan, yang paling menantang, sebuah terminal shell yang terintegrasi.
* **Fitur Inti (Level Lanjutan)**:
  1. **Editor Kode Performa Tinggi**: Mampu membuka dan mengedit file kode berukuran besar dengan lancar, tanpa lag saat scrolling atau mengetik.
  2. **Syntax Highlighting Multi-Bahasa**: Secara otomatis mewarnai sintaks berdasarkan bahasa pemrograman file (Python, Dart, JavaScript, dll.).
  3. **Terminal Terintegrasi**: Ini adalah **inti kesulitan**. Menyediakan sebuah *pseudo-terminal* (PTY) yang menjalankan shell (seperti sh atau bash) di dalam aplikasi. Ini memungkinkan pengguna menjalankan perintah sistem file, git, ssh, atau bahkan skrip.
* **Logika & State (Level Lanjutan)**:
  1. **Editor Teks Kustom**: TextField bawaan Flutter tidak akan cukup. Diperlukan pembuatan widget editor teks kustom dari awal untuk mendapatkan kontrol penuh atas rendering teks, manajemen kursor, seleksi, dan performa.
  2. **Integrasi Syntax Highlighting**: Menulis parser untuk semua bahasa tidaklah praktis. Pendekatan yang umum adalah mengintegrasikan pustaka C/C++ yang sudah ada seperti **tree-sitter** melalui **Flutter FFI**.
  3. **Integrasi Terminal**: Ini adalah tantangan rekayasa tingkat sistem yang sangat sulit.
     + Memerlukan **kode native** untuk membuat dan mengelola proses *pseudo-terminal* (PTY).
     + Membuat jembatan komunikasi dua arah (*input/output stream*) antara proses shell native dan UI terminal di Flutter.
* **Tantangan Potensial (Lanjutan)**:
  1. **Membangun editor teks performan dari nol** adalah tugas yang sangat besar.
  2. **Integrasi terminal fungsional** memerlukan pengetahuan mendalam tentang sistem operasi tingkat rendah di Android dan iOS.
  3. Mengintegrasikan pustaka C/C++ via FFI untuk *syntax highlighting*.
* **Paket Flutter yang Mungkin Digunakan**: flutter\_highlight (untuk highlighting dasar tapi kurang performan), xterm (untuk UI terminal). Namun, solusi inti untuk performa dan fungsionalitas penuh hampir pasti **kustom via FFI**.

**Tampilan UI (Mode Karakter)**

**Layar Editor Utama**

+------------------------------------------+

| [☰] file: server.js [▶ Run] [Git] |

+==========================================+

| 1| const http = require('http'); |

| 2| |

| 3| const server = http.createServer((req, |

| 4| res.end('Hello, World!'); |

| 5| }); |

| 6| |

| 7| server.listen(3000, () => { |

| 8| console.log('Server running...'); |

| 9| }); |

|... |

|------------------------------------------|

| ▲ Terminal |

|------------------------------------------|

| $ node server.js |

| Server running... |

| $ \_ |

+------------------------------------------+

*(UI terbagi antara editor kode di atas dan panel terminal yang bisa ditarik di bawah)*

Ini adalah aplikasi yang sangat teknis dan menantang. Mari kita lanjutkan ke nomor 53 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita lanjutkan.

**53. Aplikasi Simulator Penerbangan Sederhana**

**Brainstorming**

* **Tujuan Utama**: Membuat simulator penerbangan dasar di mana pengguna dapat mengendalikan sebuah pesawat dalam lingkungan 3D, melakukan manuver sederhana, lepas landas, dan mendarat.
* **Fitur Inti (Level Lanjutan)**:
  1. **Rendering Dunia 3D**: Ini adalah **tantangan terbesar**. Mampu me-render medan (terrain), langit, landasan pacu, dan model pesawat 3D secara real-time.
  2. **Model Fisika Penerbangan**: Mensimulasikan gaya-gaya aerodinamika dasar (gaya angkat, hambat, dorong, gravitasi) untuk membuat pergerakan pesawat terasa realistis saat merespons input.
  3. **Antarmuka Kontrol (HUD)**: Menampilkan *Heads-Up Display* dengan instrumen penting seperti indikator kecepatan, ketinggian, horizon buatan (*artificial horizon*), dan kontrol untuk *throttle* (tenaga mesin).
  4. **Beberapa Sudut Pandang Kamera**: Kemampuan untuk beralih antara tampilan kokpit dan tampilan orang ketiga (di luar pesawat).
* **Logika & State (Level Lanjutan)**:
  1. **Mesin Game 3D**: Membangun ini dari nol di Flutter sangat tidak praktis. Solusi yang paling masuk akal adalah **meng-embed sebuah game engine** yang sudah matang.
     + **Unity/Unreal Engine**: Menggunakan paket seperti flutter\_unity\_widget untuk menyematkan proyek Unity di dalam aplikasi Flutter. Flutter akan bertindak sebagai lapisan UI untuk menu dan HUD, sementara Unity menangani semua rendering 3D dan simulasi fisika.
  2. **Komunikasi Flutter-Engine**: Diperlukan jembatan komunikasi dua arah untuk mengirim input kontrol (dari widget Flutter) ke game engine, dan menerima data telemetri pesawat (kecepatan, ketinggian) dari engine untuk ditampilkan di HUD Flutter.
  3. **Fisika & Matematika**: Melibatkan banyak sekali perhitungan fisika dan aljabar linear (vektor, matriks, quaternion) yang biasanya ditangani oleh mesin fisika di dalam game engine.
* **Tantangan Potensial (Lanjutan)**:
  1. **Integrasi dengan Game Engine**: Menjembatani Flutter dan Unity/Unreal adalah proses yang sangat kompleks dan seringkali tidak stabil.
  2. **Pemrograman 3D & Fisika**: Memerlukan keahlian dalam pengembangan game, yang merupakan disiplin ilmu tersendiri.
  3. **Optimasi Performa**: Mencapai *frame rate* yang mulus untuk simulasi 3D di perangkat mobile sangatlah menantang.
* **Paket Flutter yang Mungkin Digunakan**: **flutter\_unity\_widget**. Sebagian besar pekerjaan berat terjadi di dalam Unity atau game engine lain, bukan di Flutter secara langsung.

**Tampilan UI (Mode Karakter)**

**Tampilan Utama Simulasi (Dari dalam Kokpit)**

*(Konteks: Skenario lepas landas dari Bandara Sultan Iskandar Muda (BTJ), Banda Aceh)*

+------------------------------------------+

| |

| [ Tampilan 3D dari jendela kokpit ] |

| [ menghadap landasan pacu ] |

| |

| |

| |

| ----- ( ◠ ) ----- | <- Horizon Buatan

| |

| |

| |

| |

+==========================================+

| SPD: 95 kts | ALT: 150 ft | THR: 100% | <- HUD

|------------------------------------------|

| [Yoke/Stick] [Rudder] [Flaps] | <- Kontrol

+------------------------------------------+

Ini adalah proyek yang setara dengan pengembangan game sederhana. Mari kita lanjutkan ke nomor 54 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, kita lanjutkan.

**54. Aplikasi Penampil Citra Medis (DICOM Viewer)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi penampil gambar medis profesional yang mampu memuat, menampilkan, dan memanipulasi file dalam format **DICOM (Digital Imaging and Communications in Medicine)**, yang merupakan standar industri untuk gambar seperti CT scan dan MRI.
* **Fitur Inti (Level Lanjutan)**:
  1. **Parsing File DICOM**: Ini adalah **tantangan utama**. DICOM bukan hanya file gambar; ia adalah kontainer data yang kompleks berisi ratusan *tag* metadata pasien, studi, dan parameter akuisisi, selain data piksel itu sendiri.
  2. **Menampilkan Stack Gambar**: Scan medis seringkali terdiri dari ratusan "irisan" (*slices*). Aplikasi harus bisa menampilkan tumpukan gambar ini dan memungkinkan pengguna untuk men-scroll melaluinya dengan cepat.
  3. **Windowing (Window/Level)**: Fitur paling krusial bagi radiolog. Memungkinkan pengguna secara interaktif menyesuaikan kontras (*window width*) dan kecerahan (*window level*) untuk menonjolkan jaringan tubuh yang berbeda (tulang, jaringan lunak, paru-paru, dll.).
  4. **Alat Pengukuran**: Menyediakan alat dasar untuk mengukur jarak atau sudut pada gambar.
* **Logika & State (Level Lanjutan)**:
  1. **Pustaka Parsing Native**: Menulis parser DICOM dari nol sangat tidak disarankan. Pendekatan yang umum adalah mengintegrasikan pustaka C/C++ yang sudah matang seperti **DCMTK** atau **Orthanc** melalui **Flutter FFI**.
  2. **Pemrosesan Gambar Kustom**: Data piksel dalam DICOM seringkali memiliki kedalaman 12-bit atau 16-bit, bukan 8-bit seperti gambar biasa. Logika *windowing* adalah fungsi matematika untuk memetakan rentang nilai piksel yang besar ini ke rentang 8-bit (0-255) yang bisa ditampilkan layar. Untuk performa real-time, idealnya ini dilakukan di GPU dengan *custom shaders*.
  3. **Manajemen Memori**: Satu studi DICOM bisa berukuran ratusan megabyte atau bahkan gigabyte. Aplikasi harus sangat efisien dalam mengelola memori.
* **Tantangan Potensial (Lanjutan)**:
  1. **Memahami standar DICOM** yang sangat tebal dan kompleks.
  2. **Pemrosesan piksel *high-bit-depth*** secara performan.
  3. Mengintegrasikan pustaka medis C/C++ yang besar via FFI.
* **Paket Flutter yang Mungkin Digunakan**: Hampir tidak ada. Ini adalah proyek yang sangat bergantung pada **solusi kustom dengan FFI**.

**Tampilan UI (Mode Karakter)**

**Layar Viewer Utama**

*(Konteks: Seorang dokter di RSU Dr. Zainoel Abidin, Banda Aceh, mereview hasil CT scan)*

+------------------------------------------+

| Pasien: #12345 | Studi: CT Abdomen |

+==========================================+

| |

| |

| |

| +--------------------------+ |

| | | |

| | [Gambar irisan CT scan] | |

| | | |

| +--------------------------+ |

| |

| |

| |

| Irisan: 178 / 512 <-------O------------> |

| W: 350 L: 50 |

+------------------------------------------+

| [W/L 🖼️] [Ukur 📏] [Zoom 🔍] [Info] |

+------------------------------------------+

*(Keterangan: W/L adalah alat Window/Level. Pengguna bisa menggeser jari di layar untuk mengubahnya secara interaktif)*

Ini adalah proyek yang sangat terspesialisasi dan kritis. Mari kita lanjutkan ke nomor 55 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita lanjutkan.

**55. Dompet Kripto dengan Integrasi Hardware Wallet (via BLE/NFC)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi dompet kripto (*software wallet*) yang sangat aman, di mana *private key* tidak pernah disimpan di ponsel. Sebagai gantinya, semua operasi penandatanganan transaksi didelegasikan ke **dompet perangkat keras (*hardware wallet*)** eksternal melalui Bluetooth Low Energy (BLE) atau NFC.
* **Fitur Inti (Level Lanjutan)**:
  1. **Melihat Aset**: Menampilkan saldo aset kripto pengguna dengan mengambil data dari API blockchain publik.
  2. **Membuat Transaksi**: Pengguna menyusun transaksi di aplikasi (alamat tujuan, jumlah).
  3. **Komunikasi BLE/NFC**: Ini adalah **inti dari aplikasi**. Aplikasi harus bisa menemukan, terhubung, dan berkomunikasi dengan hardware wallet.
  4. **Penandatanganan di Perangkat Keras**: Data transaksi yang belum ditandatangani dikirim ke hardware wallet. Pengguna harus mengonfirmasi detail transaksi secara fisik di layar perangkat keras tersebut. Perangkat keras kemudian menandatangani transaksi secara internal.
  5. **Menyiarkan Transaksi**: Tanda tangan digital dikirim kembali ke aplikasi, yang kemudian menggabungkannya dengan data transaksi dan menyiarkannya ke jaringan blockchain.
* **Logika & State (Level Lanjutan)**:
  1. **Komunikasi Nirkabel Tingkat Rendah**: Memerlukan pemahaman mendalam tentang protokol **BLE** (GATT, Services, Characteristics) atau **NFC** (NDEF). Ini adalah komunikasi biner yang sangat teknis.
  2. **Implementasi Kriptografi**: Perlu pemahaman yang sangat kuat tentang cara kerja tanda tangan digital (misalnya, ECDSA), serialisasi format transaksi, dan protokol komunikasi spesifik dari vendor hardware wallet (setiap vendor seperti Ledger atau Trezor punya protokolnya sendiri).
  3. **Keamanan Maksimal**: Seluruh alur komunikasi harus aman dari berbagai serangan. Kesalahan sekecil apa pun dalam implementasi kriptografi bisa mengakibatkan kehilangan dana pengguna secara permanen.
* **Tantangan Potensial (Lanjutan)**:
  1. **Komunikasi BLE/NFC** yang andal dan aman.
  2. **Implementasi protokol kriptografi** yang 100% benar.
  3. Reverse engineering atau bekerja dengan dokumentasi teknis tingkat rendah dari produsen hardware wallet.
  4. Bekerja dengan perangkat keras fisik.
* **Paket Flutter yang Mungkin Digunakan**: **flutter\_blue\_plus** (untuk BLE), **nfc\_manager** (untuk NFC). Sebagian besar logika kriptografi harus menggunakan pustaka C/C++ yang sudah teruji melalui **FFI**.

**Tampilan UI (Mode Karakter)**

**Layar Konfirmasi Transaksi dengan Hardware Wallet**

+------------------------------------------+

| [<-] Tinjau & Tandatangani |

+==========================================+

| |

| Kirim: |

| 0.05 BTC |

| |

| Ke Alamat: |

| 3J98t1Wp...c4vE2 |

| |

| Biaya Jaringan: ~0.0002 BTC |

|------------------------------------------|

| |

| Silakan hubungkan dan konfirmasi |

| transaksi di perangkat Ledger Anda. |

| |

| Status: Menunggu koneksi BLE... |

| (Searching for device) |

| |

| |

| [!] Pastikan detail di layar |

| perangkat Anda sudah benar! |

| |

+------------------------------------------+

Ini adalah aplikasi dengan tingkat keamanan dan risiko tertinggi. Mari kita lanjutkan ke nomor 56 jika Anda siap.

Yahok

Tampilkan alur berpikir

Tentu, mari kita lanjutkan.

**56. Aplikasi Live Streaming dengan Filter AR Real-time**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi siaran langsung (seperti Instagram Live atau TikTok Live) di mana penyiar (*streamer*) dapat menerapkan filter Augmented Reality (AR) pada wajah mereka secara real-time, dan penonton bisa berinteraksi melalui chat.
* **Fitur Inti (Level Lanjutan)**:
  1. **Siaran Langsung (Live Streaming)**: Mengambil video dan audio dari perangkat penyiar dan menyiarkannya ke server media menggunakan protokol latensi rendah seperti **RTMP** atau **WebRTC**.
  2. **Filter AR Real-time**: Pipeline yang sama seperti aplikasi kloning media sosial (No. 47): kamera -> deteksi wajah ML -> rendering AR. Tantangannya adalah melakukan ini pada video yang sedang disiarkan secara langsung, bukan direkam.
  3. **Distribusi ke Penonton**: Penonton dapat terhubung ke siaran dan menonton video secara langsung dengan jeda minimal.
  4. **Interaksi Real-time**: Penonton bisa mengirim pesan chat atau reaksi (misalnya, emoji hati) yang muncul di layar penyiar.
* **Logika & State (Level Lanjutan)**:
  1. **Infrastruktur Backend (Media Server)**: Ini adalah komponen kunci. Aplikasi tidak bisa hanya P2P untuk siaran ke banyak orang. Diperlukan **Media Server** (seperti Nginx dengan modul RTMP, Ant Media Server, atau layanan cloud seperti AWS IVS) yang akan menerima satu stream dari penyiar dan mendistribusikannya ke ratusan atau ribuan penonton.
  2. **Protokol Streaming**: Mengimplementasikan klien RTMP atau WebRTC di Flutter sangatlah sulit dan hampir pasti memerlukan **FFI ke pustaka C/C++** (seperti librtmp).
  3. **Pipeline Komputasi di Sisi Penyiar**: Perangkat penyiar harus melakukan pekerjaan yang sangat berat: mengambil video, memprosesnya dengan model ML, me-render efek AR di atasnya, meng-encode video yang sudah dimodifikasi, dan mengirimkannya ke media server, semuanya secara real-time.
* **Tantangan Potensial (Lanjutan)**:
  1. **Menyiapkan dan mengelola Media Server** adalah tugas rekayasa backend/DevOps yang kompleks.
  2. **Implementasi protokol streaming** di sisi klien.
  3. **Optimasi performa yang ekstrim** di perangkat penyiar untuk menghindari overheating atau baterai habis dengan cepat.
* **Paket Flutter yang Mungkin Digunakan**: Sangat sedikit. Mungkin camera dan google\_ml\_kit untuk input. Sisanya, terutama untuk streaming, akan bergantung pada **solusi kustom via FFI**.

**Tampilan UI (Mode Karakter)**

**Tampilan di Layar Penyiar (Streamer)**

*(Konteks: Seorang influencer lokal melakukan siaran langsung dari Masjid Raya Baiturrahman)*

+------------------------------------------+

| [LIVE 🔴] 👁️ 215 Penonton [Tutup X] |

+==========================================+

| |

| |

| [ Tampilan kamera depan penyiar ] |

| [ dengan filter sorban virtual AR ] |

| |

| |

| |

| |

|------------------------------------------|

| ❤️ Budi: Keren sekali, Bang! |

| 👍 Lisa: Assalamualaikum dari Medan. |

| ❤️ |

|------------------------------------------|

| [Filter ✨] [Tulis pesan...] |

+------------------------------------------+

*(Komentar dan reaksi dari penonton muncul secara real-time di bagian bawah layar)*

Ini adalah salah satu aplikasi paling menantang, menggabungkan *computer vision*, rekayasa backend, dan *networking*. Mari kita lanjutkan ke nomor 57, yang terakhir, jika Anda siap.

Yahok

Tampilkan alur berpikir

Baik, mari kita selesaikan yang terakhir sesuai permintaan Anda.

**57. Jejaring Sosial Berbasis Lokasi (Geofenced & Ephemeral Content)**

**Brainstorming**

* **Tujuan Utama**: Membuat aplikasi sosial di mana konten yang dibuat pengguna (seperti postingan atau foto) hanya terlihat oleh pengguna lain yang berada dalam area geografis tertentu (*geofence*) dan konten tersebut akan hilang setelah beberapa waktu (*ephemeral*).
* **Fitur Inti (Level Lanjutan)**:
  1. **Pelacakan Lokasi Real-time**: Aplikasi secara aktif melacak lokasi pengguna dan melaporkannya ke backend.
  2. **Geo-querying & Geofencing**: Ini adalah **inti dari aplikasi**. Backend harus mampu secara efisien menjawab query seperti: "berikan saya semua postingan dalam radius 1 km dari lokasi pengguna saat ini" atau "tampilkan semua pengguna di dalam area acara Pekan Kebudayaan Aceh".
  3. **Konten Ephemeral**: Setiap postingan memiliki "waktu hidup" (misalnya, 24 jam) dan akan dihapus secara otomatis oleh backend setelahnya.
  4. **Antarmuka Peta**: UI utama adalah sebuah peta interaktif yang menampilkan lokasi teman atau "heatmap" dari postingan di sekitar.
* **Logika & State (Level Lanjutan)**:
  1. **Backend Geospasial**: Melakukan *geo-query* pada skala besar sangatlah menantang untuk database standar. Ini seringkali memerlukan:
     + Layanan atau pustaka khusus seperti **GeoFire** di atas Firestore.
     + Atau bahkan database yang dirancang khusus untuk data geospasial seperti **PostGIS**, yang berarti memerlukan backend kustom penuh di luar ekosistem Firebase.
  2. **Arsitektur Backend untuk Data Ephemeral**: Memerlukan mekanisme TTL (*Time-to-Live*) atau *Cloud Function* terjadwal untuk secara rutin membersihkan konten yang sudah kedaluwarsa.
  3. **Optimasi Baterai & Jaringan**: Pelacakan lokasi secara terus-menerus sangat menguras baterai. Aplikasi klien harus cerdas dalam mengirim pembaruan lokasi, misalnya hanya saat pengguna bergerak secara signifikan (*significant location change*).
* **Tantangan Potensial (Lanjutan)**:
  1. **Backend geospasial yang skalabel** adalah masalah rekayasa yang sangat sulit.
  2. **Menyeimbangkan antara keakuratan lokasi dan efisiensi baterai** di sisi klien.
  3. Menangani isu privasi terkait data lokasi pengguna secara bertanggung jawab.
* **Paket Flutter yang Mungkin Digunakan**: geolocator, Maps\_flutter, cloud\_firestore, firebase\_functions, dan pustaka geo-query seperti geoflutterfire.

**Tampilan UI (Mode Karakter)**

**Tampilan Peta Utama**

*(Konteks: Sebuah aplikasi khusus untuk acara "Pekan Kebudayaan Aceh" di Taman Ratu Safiatuddin)*

+------------------------------------------+

| Peta PKA 2025 [👤] |

+==========================================+

| |

| /------ Peta Area Taman Ratu ------\ |

| | | |

| | (Anjungan Pidie) [🔥] | |

| | | |

| | [💬] | |

| | (Anda 📍) | |

| | [📸] | |

| \--------------------------------/ |

| |

+------------------------------------------+

| [+ Buat Postingan] [Teman] [Filter Peta]|

+------------------------------------------+

*(Keterangan: Ikon [🔥], [💬], [📸] adalah postingan dari pengguna lain di sekitar. Saat diketuk, kontennya akan muncul)*

**Tampilan Saat Melihat Konten**

+------------------------------------------+

| |

| +---------------------------------+ |

| | Post oleh: Fatimah | |

| | Lokasi: Dekat Panggung Utama | |

| | <Hilang dalam 3 jam> | |

| |---------------------------------| |

| | | |

| | [Foto pertunjukan tari Saman] | |

| | | |

| | "Malam ini ramai sekali!" | |

| +---------------------------------+ |

| |

+------------------------------------------+

Ini adalah akhir dari sesi brainstorming kita untuk 57 ide aplikasi. Semoga ini memberikan banyak inspirasi untuk proyek Flutter Anda berikutnya!

Yahok

Tampilkan alur berpikir

Anda menghentikan respons ini



Yahok

Video

Deep Research

Canvas

Gemini dapat membuat kesalahan, jadi periksa kembali responsnya

Akun Google

Ujang Amadu

indirastarly@gmail.com