

The Antigravity Developer

Jago Bikin Aplikasi Tanpa Jago Coding



Kisah Nyata: 10 Aplikasi dalam 1 Bulan

Catatan Kecil Penulis (Disclaimer)

Buku ini ditulis bukan oleh seorang Expert Programmer atau Full Stack Developer yang hafal ribuan syntax.

Saya hanyalah seorang "Pemecah Masalah" (Problem Solver) yang kebetulan menemukan alat bantu ajaib bernama Antigravity.

Tujuan buku ini bukan untuk menggurui, melainkan berbagi pengalaman nyata (studi kasus) bagaimana orang awam bisa merealisasikan idenya menjadi aplikasi yang berguna sehari-hari.

Anggaplah buku ini sebagai catatan perjalanan teman ngopi Anda. Jika ada teknik yang kurang "standar" menurut kaidah IT, mohon dimaklumi. Masukan dan koreksi sangat saya hargai agar kita bisa belajar sama-sama.

Bab 1: The "Lazy" Mindset

(Atau: Kenapa Aku Berhenti Belajar Syntax)

Jujur saja, aku ini orang yang paling bodoh kalau urusan Matematika.

Jangankan disuruh ngerti algoritma coding yang njelimet, ngajarin anak PR matematika SD saja aku butuh effort luar biasa. Padahal itu cuma hitung-hitungan sederhana. Rasanya otak ini nggak didesain buat mikir yang terlalu abstrak.

Tapi anehnya, aku punya "penyakit" yang mungkin teman-teman juga rasakan: Aku suka banget sama dunia analisis.

Di sela-sela rutinitas kerja kantor (yang scope-nya itu-itu aja sesuai SOP), aku sering iseng ngulik data. Entah itu data penjualan, data pengeluaran bulanan, atau sekadar data nggak penting lainnya.

Aku mencoba belajar Excel dari nol. Mulai dari fungsi sederhana SUM, lanjut ke menengah kayak VLOOKUP, sampai level "dewa" kayak Pivot Table, fungsi IF bertingkat, dan Macro. Tapi jujur, otak saya sering nggak nyampe. Baru sampai Macro, saya sering stres di tengah jalan. Error sedikit, bingung benerinnya. Hahaha.

Pantang menyerah, saya coba belajar Power BI. Belajar tools-nya, syntax DAX-nya, sampai akhirnya—entah gimana caranya—jadi sebuah Dashboard. Wah, itu senangnya bukan main! Rasanya puas banget lihat grafik warna-warni yang bisa gerak-gerak.

Tapi masalahnya, setiap kali aku coba naik level lagi—misalnya mau bikin aplikasi beneran (Web App) pencatat otomatis—ujung-ujungnya balik lagi ke satu kata: KECEWA.

Ada satu pengalamanku yang paling memorable (baca: menyedihkan). Aku pernah coba bikin Kalkulator Hitung Harga Logistik. Tujuannya simpel: Logic logistik itu rumit, jadi aku mau alat yang tinggal isi data, BOOM, keluar hasilnya. Biar kerjaan mudah.

Aku develop ini pakai Excel. Berapa lama? Berbulan-bulan! Hahaha. Padahal waktu itu aku sudah pakai AI lho (Gemini Gratisan). Aku minta AI bikinin rumus/fungsi, terus aku copy-paste ke sel Excel. Tapi tetap saja, rasanya kayak Excel punya batasan. Logic-nya terlalu ribet buat diakomodasi, dan tampilannya ya... tetep Excel. Nggak bisa dipoles jadi "Aplikasi Ganteng" kecuali pakai tools berbayar.

Aku pernah tahu juga waktu periode itu, ada AI tools yang bisa collect beberapa file, lalu dianalisis, dan BOOM outputnya bisa jadi report yang ringkas. Tapi ya gitu, berbayar. Padahal ini AI yang aku butuhin banget.

Sampai akhirnya di bulan November, muncullah Antigravity. Rasanya kayak ketemu malaikat yang ngerti banget kebutuhan saya.

Kenapa? Karena begitu mentok di teknis (error, rumus salah, coding merah semua), motivasi aku biasanya langsung terjun bebas. Tapi dengan Antigravity, rasanya beda. "Sebenarnya gue lagi ngapain sih? Udah buang waktu berjam-jam, hasilnya nihil. Apa faedahnya?"

Sampai akhirnya aku sadar satu hal.

Sebenarnya nggak ada yang salah dengan menjadi Tukang. Tapi masalahnya, aku nggak sadar kalau teknologi (AI) di luar sana sudah segila itu. Maklum lah ya, boro-boro mikirin perkembangan AI, mikirin kerjaan kantor aja gak beres-beres. Dasar mindset pekerja. Hehehe.

Jujur sih, dulu aku agak boring ngikutin AI. Karena kelihatannya gak jelas: cuma dipake buat bikin lucu-lucuan, curhat gak jelas, atau tanya hal-hal aneh. Sekalinya ada yang bisa bikin gambar atau video, eh banyak batasan. Masih belum sempurna. Voiceover-nya juga kaku. Jadinya aku wajar kalau kurang respect. "Ah, mainan doang," pikirku.

Tapi sekarang? Gila sih. AI itu udah gila, beneran gila. (Hahaha). Sekarang kondisinya sudah beda total. Massive banget perkembangannya.

Di situlah aku sadar: Selama ini aku capek karena memaksa diriku menjadi TUKANG, padahal harusnya aku naik pangkat jadi MANDOR.

Tukang vs Mandor (Project Manager)

Coba bayangkan kamu mau bangun rumah. Kalau kamu mau bangun rumah sendiri (tanpa tukang), kamu harus tahu:

- Takaran semen vs pasir yang pas.
- Cara masang bata biar lurus.
- Cara nyambung kabel listrik biar nggak korslet.

Salah sedikit? Rumahmu roboh. Itu yang terjadi saat aku belajar coding manual (belajar syntax). Salah titik koma (;) satu biji aja, aplikasinya meledak.

Tapi, gimana kalau kamu jadi Mandor atau Project Manager? Kamu nggak perlu tahu cara ngaduk semen. Yang kamu perlu tahu cuma: "Pak, saya mau tembok sebelah sini warnanya putih, tingginya 3 meter, dan ada jendela di tengahnya."

Kalau temboknya miring? Kamu tinggal teriak: "Pak, itu miring. Bongkar, lurusin lagi!"

Nah, AI (Artificial Intelligence) hari ini—entah itu ChatGPT, Claude, atau Gemini—adalah Tukang Paling Cerdas dan Paling Sabar sedunia.

Mereka hafal jutaan baris kode. Mereka tahu takaran semen yang pas. Mereka nggak pernah capek disuruh bongkar pasang 100 kali.

Kelemahan mereka cuma satu: Mereka nggak pernah inisiatif. Mereka butuh Mandor. Mereka butuh KAMU.

The "Lazy" Mindset

Jadi, rahasia pertamaku bisa bikin aplikasi kayak ZenSheet atau Kalkulator Eksport Impor itu bukan karena aku jago coding. Justru karena aku MALAS.

Aku malas ngafalin syntax HTML/CSS/JS. Aku malas nyari error di Google (StackOverflow).

Karena malas, aku mendelegasikan semua kerjaan "kuli" itu ke AI. Tugasku sekarang berubah drastis:

- Dulu: Ngetik kode baris per baris (Stess level: 100).
- Sekarang: Memerintah, Mengecek, dan Meminta Revisi (Stress level: 10).

Di buku ini, aku nggak akan ngajarin kamu jadi programmer yang jago algoritma. Itu urusan orang-orang jenius di Silicon Valley.

Di sini, aku bakal ajarin kamu cara jadi Project Manager yang Galak tapi Efektif. Gimana caranya ngasih perintah (Prompt) biar si "Tukang AI" ini nggak halu, nggak malas, dan beneran bikin aplikasi yang kita mau.

Bab 2: Persiapan "Senjata" (Antigravity App)

Instalasi Bengkel Ajaib Kita

Di bab ini, kita akan langsung fokus menyiapkan satu-satunya alat tempur yang paling penting: **Antigravity App**.

Ini adalah "bengkel" tempat kita akan menyuruh AI bekerja. Tanpa ini, kode-kode yang dibuat AI cuma akan jadi teks bisu yang nggak bisa dipakai.

Cara Install Antigravity App

Ikuti panduan langkah demi langkah ini. Tenang, cuma butuh waktu 5 menit.

Langkah 1: Cari di Google

Buka Google, ketikan kata kunci: "**antigravity google**".

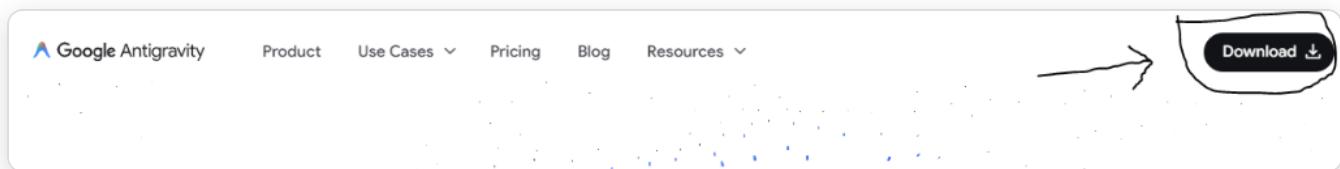
Pilih hasil pencarian paling atas yang mengarah ke [antigravity.google](#).



Google Search Antigravity

Langkah 2: Download

Begitu masuk ke website resminya, cari tombol **Download** di pojok kanan atas (atau di tengah halaman). Klik tombol tersebut.



Tombol Download

Langkah 3: Pilih Versi Windows (PENTING!)

Nah, di sini biasanya banyak yang bingung. Akan muncul pilihan untuk MacOS, Windows, dan Linux.

Karena kita pengguna Windows, perhatikan bagian **Windows**. Di sana ada pilihan "Download for x64" dan "Download for ARM64".

Pilih yang: **Download for x64**.

The screenshot shows the download section of the Antigravity website. It has three main sections: 'MacOS' with icons for Apple Silicon and Intel, 'Windows' with icons for x64 and ARM64, and 'Linux' with a penguin icon. Each section includes a 'Minimum Requirements' link and a 'Download' button.

Operating System	Architectures	Download Links
MacOS	Apple Silicon, Intel	Download for Apple Silicon , Download for Intel
Windows	x64, ARM64	Download for x64 , Download for ARM64
Linux	-	Download

Minimum Requirements
macOS versions with Apple security update support. This is typically the current and two previous versions. Min Version 12 (Monterey), X86 is not supported

Minimum Requirements
Windows 10 (64 bit)

Minimum Requirements
glibc >= 2.28, glibcxx >= 3.4.25 (e.g. Ubuntu 20, Debian 10, Fedora 36, RHEL 8)

Pilih Versi x64

"Mas, x64 itu apa sih?"

Penjelasan singkatnya: **x64** adalah arsitektur prosesor standar yang dipakai oleh hampir semua laptop/PC Windows (Intel atau AMD) modern. Kalau laptopmu beli tahun 2010 ke atas, 99% pasti x64.

Jadi nggak usah pusing mikirin teknisnya. Pokoknya klik **x64**.

Langkah 4: Install

Setelah file ter-download (biasanya berakhiran **.exe** atau **.msi**), klik dua kali file tersebut. Ikuti saja instruksinya (Next, Next, Install) sampai selesai.

Begitu selesai, buka aplikasinya. Selamat! Bengkel ajaibmu sudah siap.

Mengenal Interface Antigravity

Sebelum mulai bekerja, ada baiknya kita kenali dulu tampilan Antigravity. Ini akan membantu Anda lebih nyaman saat nanti mulai "tek-tok" dengan AI.

Dua Panel Utama

Begitu aplikasi terbuka, Anda akan melihat layar terbagi menjadi dua bagian besar:

The screenshot shows the Antigravity code editor interface. On the left side, there is a 2-panel workspace. The top panel contains the code for a modal window, specifically a section for a dynamic preview. The bottom panel contains the generated HTML output for the same section. On the right side, there is a large panel for the AI chat interface. It includes a header with the title "Cleaning up Mutasi Tab" and a message about the grep tool failing. Below this is a "Progress Updates" section with several numbered items, each with a search query and results count. At the bottom of the right panel, there is a footer with various settings and status indicators.

Tampilan 2-Panel Antigravity

Panel Kiri: Workspace (Area Kerja)

Ini tempat di mana kode dan file project Anda ditampilkan. Ada dua tab biasanya:

- **Code:** Menampilkan source code (HTML, CSS, JavaScript, dll)
- **Preview:** Menampilkan hasil aplikasi yang sedang dibuat

Fokus Anda di panel kiri ini adalah untuk **MELIHAT HASIL**, bukan edit manual. Biarkan AI yang ngurusin codenya.

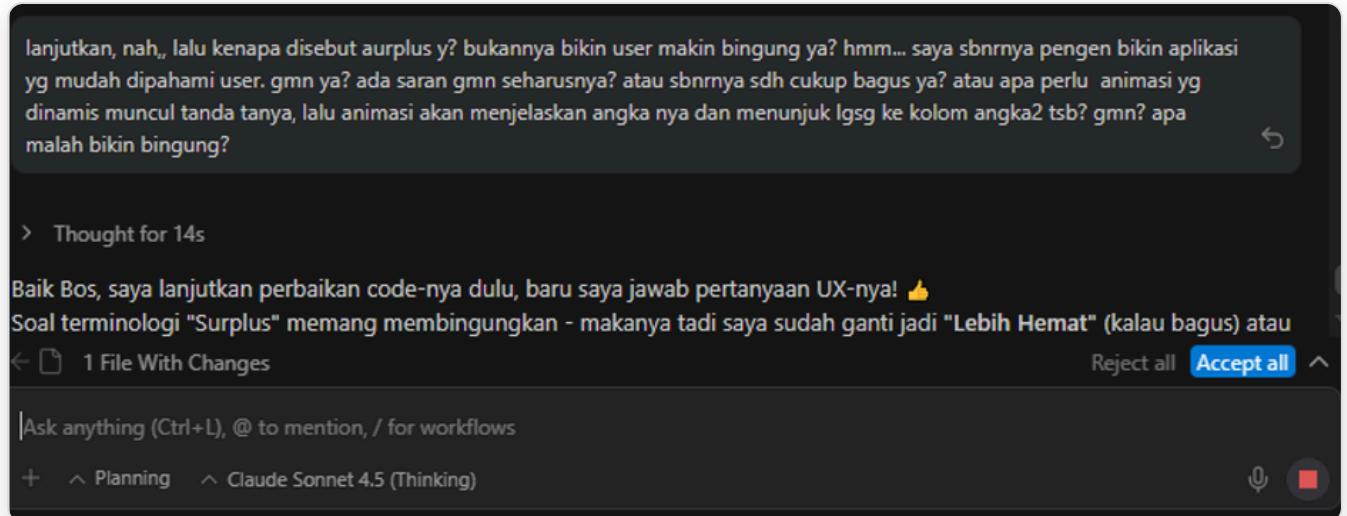
Panel Kanan: Chat (Obrolan dengan AI)

Ini adalah tempat Anda mengetik perintah. Interface-nya mirip WhatsApp:

- Ada kolom kecil di bawah untuk menulis prompt
- AI akan merespons di atas kolom tersebut
- Conversation history tersimpan, bisa scroll ke atas untuk lihat percakapan sebelumnya

Workflow: Accept, Reject, atau Always Proceed?

Saat AI selesai memproses request Anda dan menghasilkan code, biasanya akan muncul notifikasi yang menanyakan persetujuan.



Accept Button Workflow

Anda akan melihat 2 pilihan utama:

"Accept" (biru) - Klik ini kalau Anda setuju dengan perubahan yang diusulkan AI

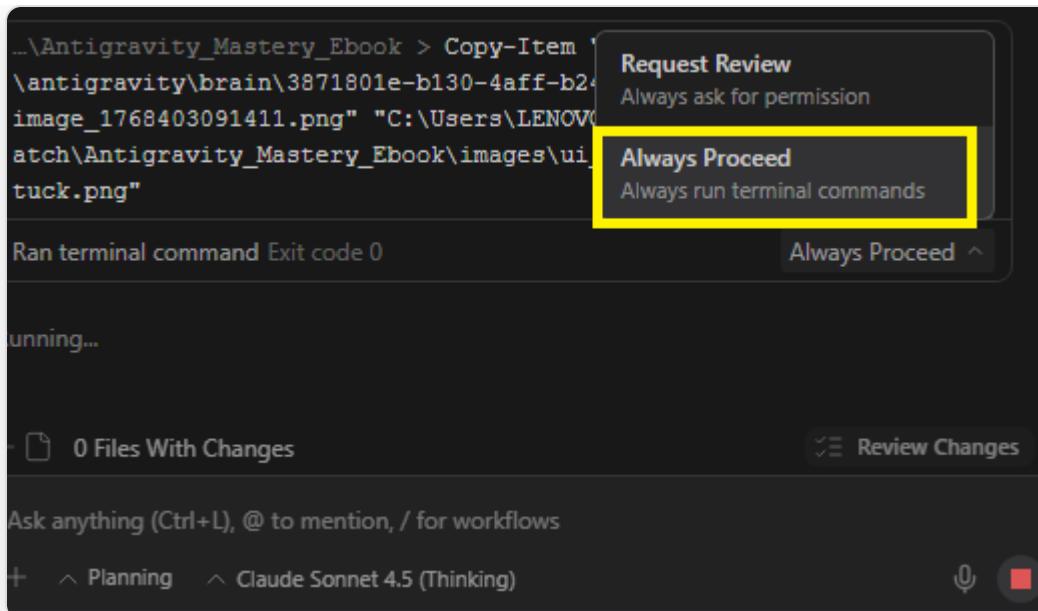
"Reject all" - Klik ini kalau Anda tidak setuju dan ingin batalkan perubahan

Tip Praktis:

Untuk project pembelajaran atau eksperimen, biasanya langsung klik **Accept** saja. AI jarang ngasih code yang merusak. Kalau ada bug, nanti tinggal minta AI buat fix lagi.

Opsi "Always Proceed" untuk Workflow Lebih Cepat

Kalau Anda merasa bolak-balik klik Accept itu memperlambat workflow, ada solusinya:



Always Proceed Option

Pada notification box, ada opsi "**Always Proceed**" - ketika Anda enable ini, AI akan otomatis apply semua perubahan tanpa konfirmasi lagi.

Kapan pakai Always Proceed?

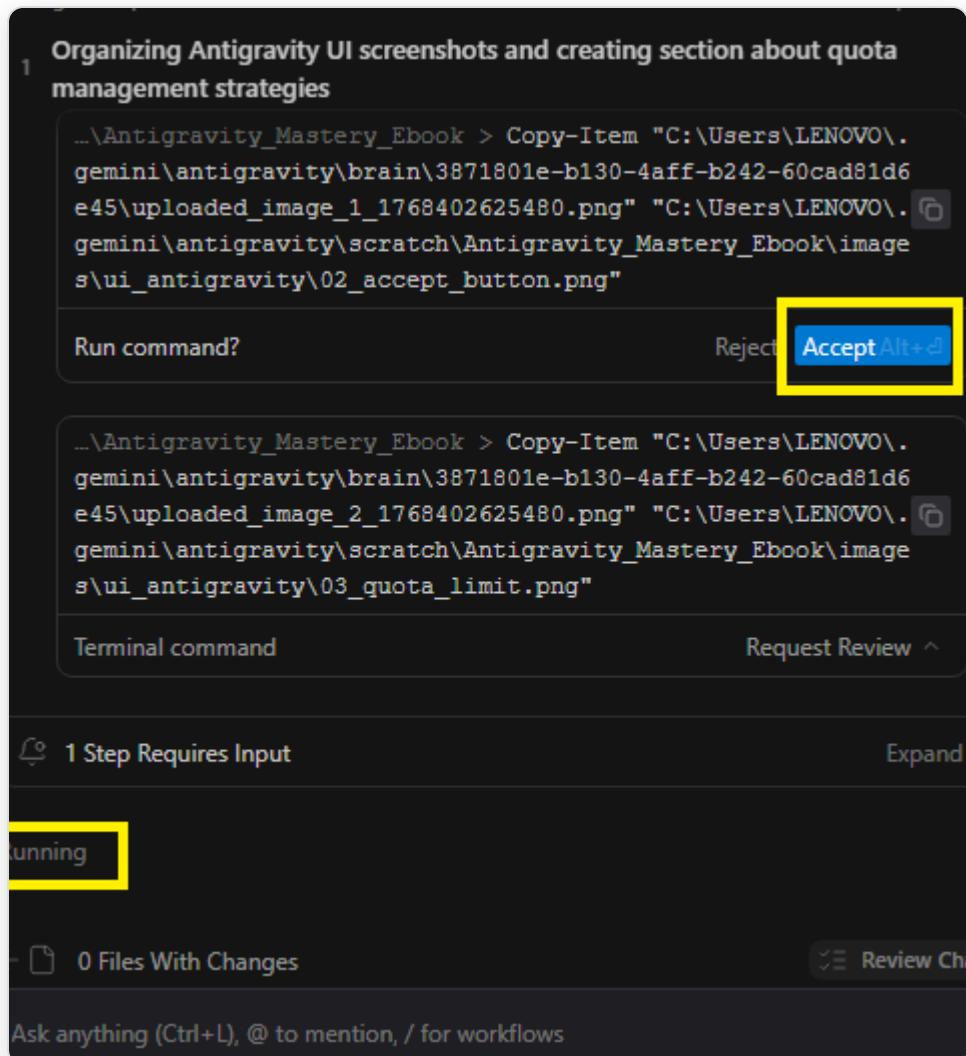
- Ketika Anda sedang iterasi cepat (testing ide, coba-coba fitur)
- Ketika Anda percaya diri dengan prompt yang Anda berikan
- Ketika Anda tidak khawatir dengan perubahan yang destructive (karena bisa di-undo)

Kapan JANGAN pakai Always Proceed?

- Ketika project sudah production/dipakai orang lain
- Ketika request Anda kompleks dan butuh review manual
- Ketika Anda masih belajar dan ingin tahu apa yang AI ubah

Troubleshooting: Masalah Umum dan Solusinya

1. Stuck di "Running..." Tidak Bergerak



Running Stuck Issue

Masalah: Anda lihat tulisan "Running..." di bawah layar, tapi tidak ada progress sama sekali.

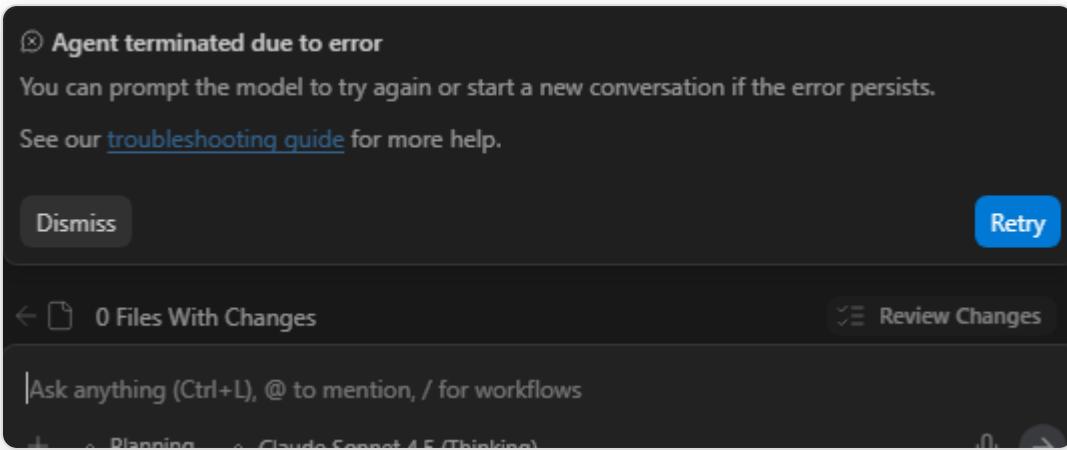
Penyebab: Biasanya ini karena ada **Accept button yang belum diklik** di atas.

Solusi:

Scroll ke atas di panel chat → Cari kotak biru "Accept" → Klik "Accept"

Setelah diklik, AI akan lanjut proses. Jadi kalau stuck, **selalu cek dulu ke atas** apakah ada yang perlu di-approve.

2. Error "Agent Terminated Due to Error"



Retry Button for Connection Error

Masalah: Muncul notifikasi error merah dengan tulisan "Agent execution terminated due to error"

Penyebab:

Biasanya ini terjadi karena:

- Koneksi internet terputus/tidak stabil
- Server AI sedang overload
- Request terlalu kompleks/berat untuk diproses

Solusi:

Klik tombol "Retry" (biru) di pojok kanan bawah notification.

AI akan mencoba ulang request terakhir Anda. Kalau masih error setelah 2-3x retry, coba:

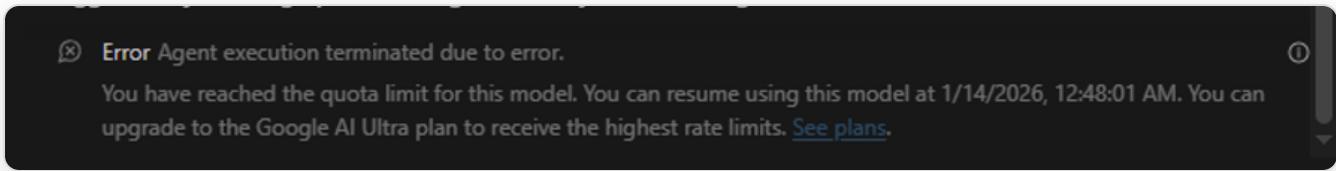
Refresh browser/restart Antigravity

Cek koneksi internet Anda

Coba ganti model AI (misal dari Gemini ke Claude)

Sederhanakan prompt Anda (pecah jadi beberapa step kecil)

3. Quota Limit Habis



Quota Limit Error

Masalah: Muncul error "You have reached the quota limit for this model"

Penyebab: Setiap model AI punya limit penggunaan gratis (daily/weekly)

Solusi:

Ganti model lain - Kalau Gemini 3 Pro habis, coba pakai Claude Sonnet

Tunggu reset - Quota akan reset sesuai periode (harian/mingguan)

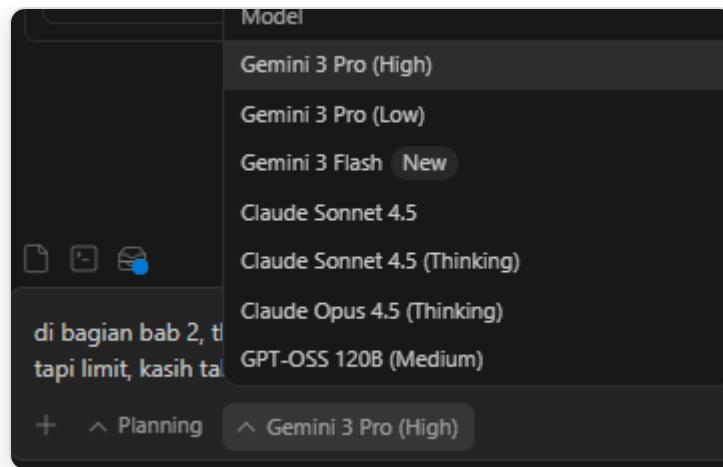
Pakai model lebih hemat - Untuk task sederhana, pakai Gemini Flash

Strategi Hemat Quota:

- Brainstorming awal pakai ChatGPT/Gemini web dulu (di luar Antigravity)
- Baru masuk Antigravity ketika konsep sudah matang
- Pakai model "Flash" atau "Low" untuk perubahan simpel (UI, styling)
- Save model "High" atau "Thinking" untuk logic kompleks saja

Memilih "Otak" AI (PENTING!)

Di bagian bawah aplikasi (lihat baris paling bawah), kamu akan melihat sebuah menu pilihan. Ini adalah tempat kita memilih **Model AI** alias "Otak" yang akan mengerjakan perintah kita.



Model Selector UI

Saat ini ada 3 "Dewa AI" utama di dunia:

Gemini (Buatan Google)

Claude (Buatan Anthropic)

GPT (Buatan OpenAI)

Bingung pilih yang mana?

Saran Saya: Pakai saja "Gemini 3".

Kenapa?

- **Cepat & Pintar:** Cocok banget buat koding sehari-hari.
- **GRATIS:** Kamu bisa pakai ini tanpa bayar sepeser pun.

"Mas, kok pilihannya banyak banget? Bedanya apa?"

Kalau kamu klik menunya, mungkin akan muncul daftar panjang seperti:

- Gemini 3 Pro (High)
- Gemini 3 Pro (Low)
- Gemini 3 Flash
- Claude Sonnet 4.5 (Thinking)
- GPT OSS 120B

Rumus gampangnya begini:

Makin "Tinggi" levelnya (High/Opus/Thinking), makin **PINTAR** dia. Makin jago memahami logika yang njelimet.

Tapi... ada harganya:

Kuota lebih cepat habis.

Mikirnya lebih lama.

Bocoran Limit (Kuota Gratis):

Gemini 3 Pro (High):

- Sistem kuotanya **MINGGUAN (7 Hari)**.
- Jadi kalau hari Senin kamu pakai habis-habisan sampai limit, kamu baru bisa pakai lagi Senin depan. *Hati-hati ya!*

Gemini 3 Pro (Low) / Flash:

- Sistem kuotanya **HARIAN (Reset tiap pagi/tengah malam)**.
- Ini yang paling aman buat eksperimen barbar. Habis hari ini, besok isi lagi.

Claude & Thinking Models (Opus/Sonnet):

- Limitnya **SANGAT SEDIKIT (Harian)**.
- Karena ini model mahal, gunakan hanya untuk "kartu As" di momen krusial (misal: saat Gemini High gak nemu solusi).

Panduan Memilih:

- **Gemini 3 Pro (Low) / Flash:** Pakai ini buat sehari-hari. Cepat, irit kuota. Cocok buat bikin tampilan (HTML/CSS) atau fitur simpel.
- **Gemini 3 Pro (High) / Claude Sonnet:** Pakai ini kalau si Low mulai "bego" atau error terus. Ini ibarat "Mandor Senior".
- **Thinking Models (Opus/Sonnet Thinking):** Ini "Profesor". Pakai ini kalau kamu mentok total atau mau bikin logika bisnis yang super rumit. Dia bakal mikir lamaaa banget sebelum jawab, tapi hasilnya biasanya presisi.

Strategi Hemat:

*Mulai dulu pakai yang **Low/Flash**. Kalau error, baru naik kelas ke **High**. Kalau masih error, panggil sang **Profesor (Thinking)**.*

Jadi, pastikan di pojok bawah aplikasimu terpilih: **Gemini 3 Pro (Low/High)** untuk awal.

Rahasia "Premium Gratisan" (Gemini Advanced)

Saat buku ini ditulis, Google menawarkan paket **Gemini Advanced** seharga **Rp 309.000 / bulan**.

Tapi tenang, biasanya ada **Free Trial 30 Hari Pertama**.

Tips "Nakal" Hemat Budget:

Promosi ini berlaku per akun Google.

Jadi, kalau masa trial 30 harimu habis dan kamu belum rela bayar...

1. *Bikin akun Google (Gmail) baru.*
2. *Login ulang di Antigravity pakai akun baru itu.*
3. **Boom!** Dapat gratis 30 hari lagi.

(Jangan bilang siapa-siapa ya, ini rahasia kita aja.)

Mari kita mulai "mempekerjakan" mereka!

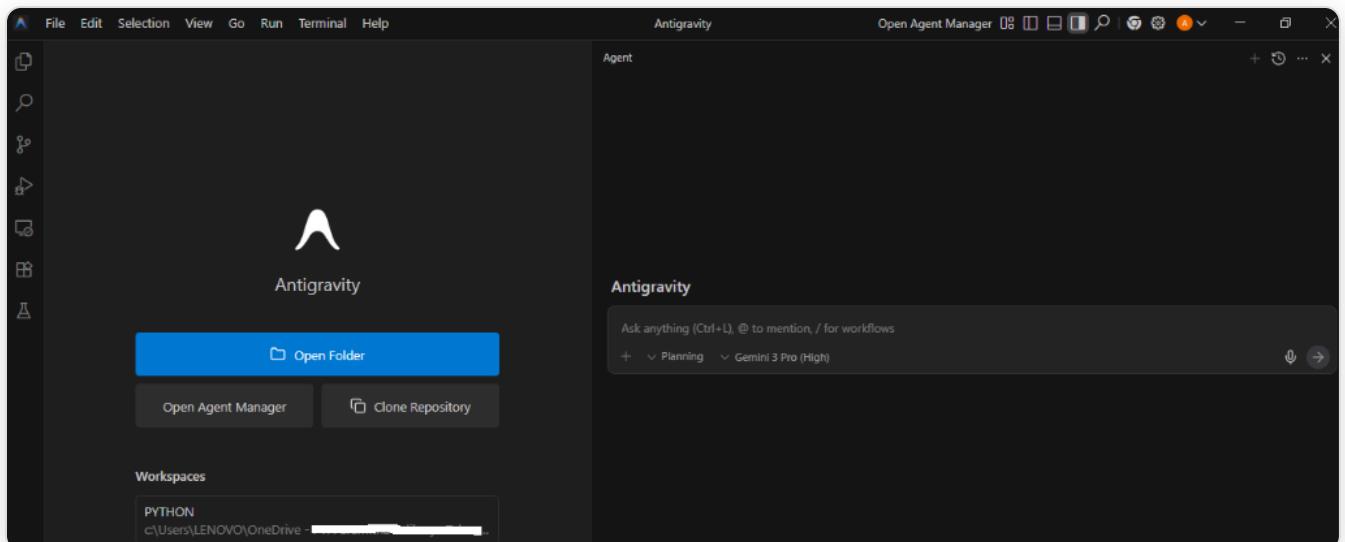
Bab 3: Cara Kerja & Jalan Pintas

(Atau: Jangan Dilihat Kodenya, Nanti Pusing)

Kalau di bab sebelumnya kita sudah setup "Senjata", sekarang kita belajar cara menembaknya.

Workflow: Kanan (Perintah) -> Kiri (Hasil)

Karena layarmu cuma dua kolom, anggap saja ini seperti **Remote TV** dan **Layar TV**.



Tampilan Antigravity App

- **Kanan (Remote):** Tempat kamu pencet-pencet (ngetik prompt).
- **Kiri (Layar TV):** Tempat kamu nonton hasilnya.

Langkah 1: Memerintah (Kolom Kanan)

Di kolom Chat, ketik keinginanmu:

"Buatkan saya kalkulator sederhana."

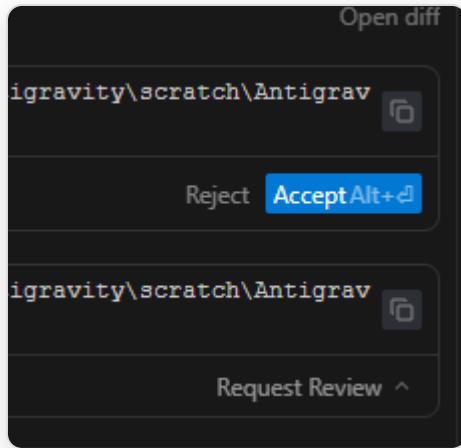
Langkah 2: Layar Kiri Berubah ("Tukang" Bekerja)

Otomatis layar sebelah kiri akan bergerak sendiri. AI sedang menulis kode.

Mungkin kamu akan melihat tulisan warna-warni (Code Editor).

PENTING: Tombol REVIEW (Accept/Reject)

Kadang-kadang, AI akan berhenti sejenak dan meminta persetujuanmu. Akan muncul tombol **Accept** (Terima) atau **Reject** (Tolak).



Tombol Accept Reject

Bingung harus ngapain?

JURUS GASPOL: Langsung klik **Accept** (atau tekan **Alt + Enter**).

Nggak usah dibaca kodenya. Percaya saja sama "tukang"-nya. Kalau nanti hasilnya salah, tinggal kita marahi lagi. Jadi, **GASS TERUS!**

Langkah 3: Munculkan Hasil (Preview)

Tujuan kita adalah melihat aplikasinya, bukan kodenya.

Biasanya AI akan otomatis mengubah layar kiri jadi Preview (Tampilan Aplikasi).

Tapi kalau layar kirinya masih nyangkut di kode (tulisan ruwet), kamu harus menegurnya.

Ketik di chat kanan:

"Tolong bukakan aplikasinya."

atau

"Mana preview-nya?"

AI akan otomatis "mengganti channel" di layar kiri dari Mode Kode ke Mode Preview.

Nah, di situlah kamu bisa tes aplikasinya.

Tips Pro: Jangan "Buta Map" (Hemat Kuota)

Satu kesalahan fatal pemula: **Menyuruh AI bikin aplikasi yang kamu sendiri gak paham logikanya.**

Kalau kamu gak tau maunya apa, AI akan "mengarang bebas". Hasilnya? Error, ngaco, dan kamu stress sendiri. Bukan salah AI-nya, tapi salah kita yang gak ngasih "peta" yang jelas.

Saran Saya:

Sebelum buka Antigravity, **BRAINSTORMING DULU.**

Matangkan dulu konsepnya. Mau fitur apa? Alurnya gimana?

Diskusi sama siapa? Ya sama AI juga!

Trik Hemat Limit:

Karena Antigravity (Gemini 3) ada kuota/limitnya biarpun gratis, sayang kalau dipakai cuma buat ngobrol ngalor-ngidul.

Mending brainstorming-nya dilakukan di **Luar Aplikasi** (pakai Chrome biasa).

Buka [ChatGPT.com](https://chatgpt.com) atau [Gemini.google.com](https://gemini.google.com).

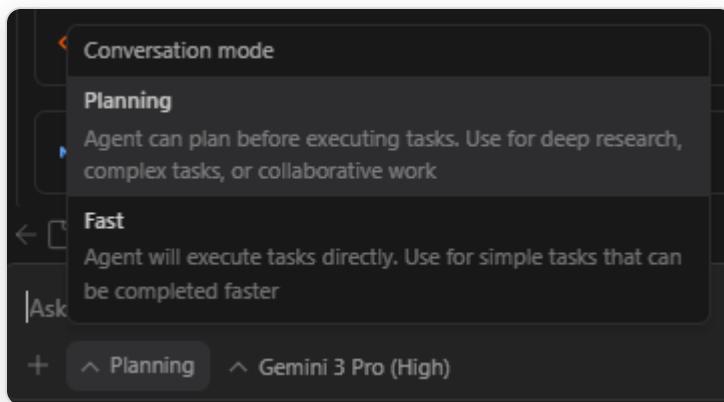
"Bro AI, saya punya ide bikin aplikasi kasir. Kira-kira butuh fitur apa aja ya biar simpel tapi kepakai?"

Diskusi sampai matang di sana. Begitu konsep sudah jadi dan kamu paham logikanya, baru bawa ke Antigravity untuk dieksekusi.

Hemat kuota, hati tenang, hasil maksimal.

Tips Pro 2: Pilih Mode "Planning" (Biar Gak Bolak-Balik)

Di sebelah kiri kolom chat, ada menu kecil bertuliskan **Mode**.



Planning vs Fast Mode

Ada dua pilihan:

Fast (Cepat): AI langsung "hajar" nulis kode. Kalau salah, ya salah.

Planning (Perencanaan): AI akan mikir dulu, bikin rencana, dan nanya ke kamu: "*Bos, saya mau bikin gini, setuju gak?*"

Saran Saya:

Gunakan selalu **PLANNING MODE**.

Kenapa? Karena AI akan menampilkan preview rencananya dulu sebelum buang-buang "tenaga" (kuota) buat nulis kode.

Kalau rencananya salah, kamu bisa koreksi di awal.

Ini jauh lebih hemat limit daripada pakai mode Fast -> Salah -> Ulangi -> Habis Limit.

Kamus Istilah (File-File Aneh)

Selama proses pembuatan, kamu mungkin akan melihat file-file aneh bermunculan di foldermu. Nggak usah panik, ini artinya:

1. File Laporan (Dokumen Proyek)

Ini adalah file komunikasi antara kamu ("Bos") dan AI ("Mandor").

- **task.md** : Ini **Daftar Tugas** (To-Do List). Isinya checklist apa saja yang harus dikerjakan AI.
- **implementation_plan.md** : Ini **Proposal Rencana**. Sebelum kerja, AI biasanya bikin rencana dulu di sini. Kalau kamu setuju, baru dia kerjakan.
- **walkthrough.md** : Ini **Laporan Pertanggungjawaban**. Setelah selesai kerja, AI akan nulis di sini: "Bos, saya sudah ngerjain A, B, C. Hasilnya begini (ada screenshot)."

2. File Bangunan (Kodingan)

Ini adalah bahan bangunan aplikasimu.

- **index.html** : Ini **Bangunan Utama**. Kalau mau buka aplikasi, cari file ini.
- **style.css** : Ini **Cat & Dekorasi**. Warna, font, tata letak, semuanya diatur di sini.
- **app.js** / **script.js** : Ini **Listrik & Otak**. Yang bikin tombol bisa diklik, hitungan bisa jalan, itu file ini.
- **widget** / **data.js** : Kadang ada file tambahan buat data atau fitur tempelan.

Kesimpulan:

Kalau kamu bingung, cukup cari **index.html**. Itu adalah pintunya. Sisanya? Biar AI yang urus.

Bab 3.5: Melihat Proses Sebenarnya - Brainstorming sampai Debugging

(Dokumentasi Lengkap: Tek-Tok dengan AI)

Di bab sebelumnya saya sudah menjelaskan teori tentang bagaimana berkomunikasi dengan AI. Sekarang saya ingin menunjukkan seperti apa proses sebenarnya ketika membuat aplikasi - dari awal sampai selesai.

Ini adalah dokumentasi nyata ketika saya membuat aplikasi "**Dr. Anti Bokek**" - sebuah personal finance tracker untuk tracking keuangan harian. Semua screenshot di sini adalah capture asli dari percakapan saya dengan Antigravity.

Apa itu Dr. Anti Bokek?

Dr. Anti Bokek adalah aplikasi web sederhana untuk membantu saya mengelola keuangan personal sehari-hari.

Masalah yang ingin diselesaikan:

- Saya punya budget bulanan yang sudah terstruktur (fixed expenses, wants, savings)
- Tapi saya kesulitan tracking pengeluaran **harian** - apakah saya masih "on track" atau sudah "tekor"
- Kadang di akhir bulan baru sadar ternyata sudah overspending

Solusi yang diusulkan AI:

- Dashboard keuangan yang menampilkan "Jatah Harian" (daily allowance)
- Input cepat untuk catat pengeluaran dengan "Fast Action Button" (tombol Suntik - Feedback real-time: masih "Lebih Hemat" atau sudah "Tekor"

- Gamifikasi dengan komentar "Dokter" yang kasih saran (makanya namanya Dr. Anti Bokek)

Ini bukan aplikasi kompleks seperti Mint atau YNAB. Ini tool personal yang sangat spesifik untuk kebutuhan saya - **tracking harian dengan mindful spending**.

Dan saya membuatnya **tanpa menulis satu baris code pun secara manual**. Semua dikerjakan AI, mulai dari brainstorming konsep sampai bug fixing.

Mari kita lihat prosesnya step-by-step:

1. Brainstorming: Meminta IDE dari AI

Saya tidak selalu tahu persis aplikasi seperti apa yang saya butuhkan. Kadang saya hanya punya problem statement yang samar-samar.

Untuk Dr. Anti Bokek, masalah saya sederhana: saya kesulitan tracking pengeluaran harian, padahal untuk budgeting bulanan saya sudah punya worksheet yang cukup detail.

Jadi saya mulai dengan bertanya ke AI:

skrgke aplikasi dr anti bokek, aku punya ide, sbnrnya pasar user yg seneng mencatat details pengeluaran harian itu msh lumayan banyak, nah mungkin gak sih menggabungkan ide worksheet daily mencatat pengeluaran dengan aplikasi dr anti bokek? mgkn kamu ada ide? coba brainstorming, cari tahu gimana konsep ny

> Thought for 5s

Brainstorming Dr. Anti Bokek Evolution

I analyzed the `Dr_Anti_Bokek` code and confirmed it focuses on monthly/macro budgeting. To integrate daily tracking, I devised the 'Rawat Jalan Harian' (Daily Outpatient Care) concept. This adds micro-transaction logging with gamified 'Doctor' feedback, bridging the gap between monthly planning and daily habits. I am now presenting this concept to the user.

Progress Updates

Expand all <

- 1 Exploring Dr. Anti Bokek codebase
- 2 Reading Dr. Anti Bokek index.html
- 3 Proposing 'Rawat Jalan Harian' concept to user

Ide Menarik, Bro! 🎉

Market "pencatat detail" itu memang besar karena rasanya lebih *mindful* saat keluar uang. Kalau digabung sama konsep Dr. Anti Bokek yang sudah *fun* dan *nyeleneh*, ini bisa jadi killer feature.

Saya ada konsep brainstorming begini:

Nama Fitur: "Rawat Jalan Harian" 🚑

Konsepnya: Keluar input bulanan itu: Checkup, input harian itu: Rawat Jalan (minum obat rutin)

← 0 Files With Changes

Review Changes

Ask anything (Ctrl+L), @ to mention, / for workflows

+ ⌂ Planning ⌂ Gemini 3 Pro (High)



Ln 1, Col 1 Spaces: 4 UTF-8 CRLF () JavaScript Antigravity - Settings

Brainstorming Phase - Meminta IDE

"Saya punya ide untuk aplikasi tracking keuangan harian yang bisa sinkron dengan worksheet budgeting bulanan. Kamu ada ide untuk brainstorming?"

Yang menarik: saya tidak langsung minta "buatkan aplikasi X dengan fitur Y dan Z".

Saya minta AI untuk ikut **berpikir** bersama saya mengenai konsep yang tepat.

2. AI Memberikan Planning Detail

Setelah memahami problem statement saya, AI tidak langsung ngoding. AI memberikan proposal konsep terlebih dahulu:

Ide Menarik, Bro!💡

Market "pencatat detail" itu memang besar karena rasanya lebih *mindful* saat keluar uang. Kalau digabung sama konsep

Dr. Anti Bokek yang sudah *fun* dan *nyeleneh*, ini bisa jadi killer feature.

Saya ada konsep brainstorming begini:

Nama Fitur: "Rawat Jalan Harian" 🚗

Konsepnya: Kalau input bulanan itu Checkup, input harian itu Rawat Jalan (minum obat rutin).

Fitur Utamanya:

1. Fast Action Button ("Suntik Pengeluaran"):

- Input cepat: Nominal & Kategori (Wajib/Napas vs Jajan/Dosa).
- Langsung *real-time* mengurangi "Nyawa" (Budget) bulan itu.

2. Gamifikasi "Daily Allowance" (Jatah Harian):

- Sistem otomatis ngitung: $(Gaji - Fixed Cost - Tabungan) / 30 \text{ hari}$.
- Misal jatah sehari Rp 50.000.
- Kalau hari ini jajan cuma 20rb -> "Sultan! Hemat 30rb." 🟢
- Kalau hari ini jajan 100rb -> "WOY! Overdosis! Besok puasa ya." 🟥

3. Komentar Dokter (The Best Part):

- Tiap kali input pengeluaran "Jajan", Dokternya ngomel.
- *Input Kopi 50rb* -> Dokter: "Air hitam doang 50rb? Ginjal aman bos?"

Koneksi ke Fitur Lama:

- Inputan "Budget Lifestyle" di menu awal jadi Pagu Anggaran.
- Inputan "Rawat Jalan" ini jadi Realisasi.
- Nanti di Dashboard ada bar Budget vs Kenyataan.

- ⚡ 0 Files With Changes

Review Changes

AI Planning Response - Concept Breakdown

Di sini AI menyarankan konsep "Rawat Jalan Harian" - sebuah metode tracking pengeluaran harian yang bisa dibridge ke monthly budgeting.

Perhatikan detailnya:

- Fast Action Button untuk input cepat
- Gamifikasi "Daily Allowance" dengan visual feedback
- Komentar "Dokter" yang memberikan saran (makanya namanya Dr. Anti Bokek)
- Dashboard progress Budget vs Kenyataan

Saya bahkan tidak memikirkan semua itu di awal. AI yang mengusulkan.

3. Approval dan Implementation Plan

Setelah membaca proposal AI, saya merasa konsepnya bagus. Jadi saya approve:

"Ide menarik, Bro! Market 'pencatatan detail' memang lebih mindful. Mari kita eksekusi dengan Rawat Jalan Harian concept."

Kemudian AI membuat implementation plan yang lebih teknis:

ok gass

Thought for 7s

Approving the Concept

I'm pleased to share that the user has given the green light for the "Rawat Jalan Harian" concept. We can now proceed with confidence and focus on integrating the UI update, namely, the "Input Harian" / "Daily Input" feature within the `Dr_Anti_Bokek` app. This is the first step towards a more comprehensive solution.

Outlining Implementation Strategy

I'm now formulating a detailed implementation plan for the "Rawat Jalan Harian" feature within `Dr_Anti_Bokek`. I'm going to create a `task.md` document, detailing UI updates (FAB and dashboard section), core logic (`addDailyExpense`, allowance calculation, persistence), and user feedback mechanisms. This will guide the development process.

Generating..

Rencana Eksekusi: Rawat Jalan Harian 📝

Berikut detail teknisnya:

- 1. UI Baru:**
 - Tombol Suntik (FAB): Di pojok kanan bawah, icon Jarum Suntik.
 - Panel Jatah Harian: Di Dashboard, menampilkan "Sisa Uang Jajan Hari Ini".
 - Modal Input: Simpel, pilih "Napas (Needs)" atau "Dosa (Wants)".
- 2. Logika Gamifikasi:**
 - Sistem akan menghitung $(\text{Budget Jajan Sebulan} - \text{Jajan Bulan Ini}) / \text{Sisa Hari}$.
 - Kalau hari ini jajan kebanyakan, dokter akan marah di panel dashboard.

Silakan review rencana ini. Kalau oke, saya langsung oprek kode `index.html`. 🚧

`** implementation_plan.md`

Open

Approval & Implementation Plan

AI secara otomatis membuat file `implementation_plan.md` yang berisi:

- UI Baru: Tombol Suntik (FAB), Panel Jatah Harian, Modal Input

- Logika Gamifikasi: Sistem hitung budget per hari
- Integrasi dengan code existing

Yang penting untuk dicatat: **saya tidak perlu tahu detail teknis ini.**

AI yang membuat planning, saya hanya perlu "oke gas" atau "tunggu, ini perlu diubah".

4. Implementation: Fitur Ditambahkan

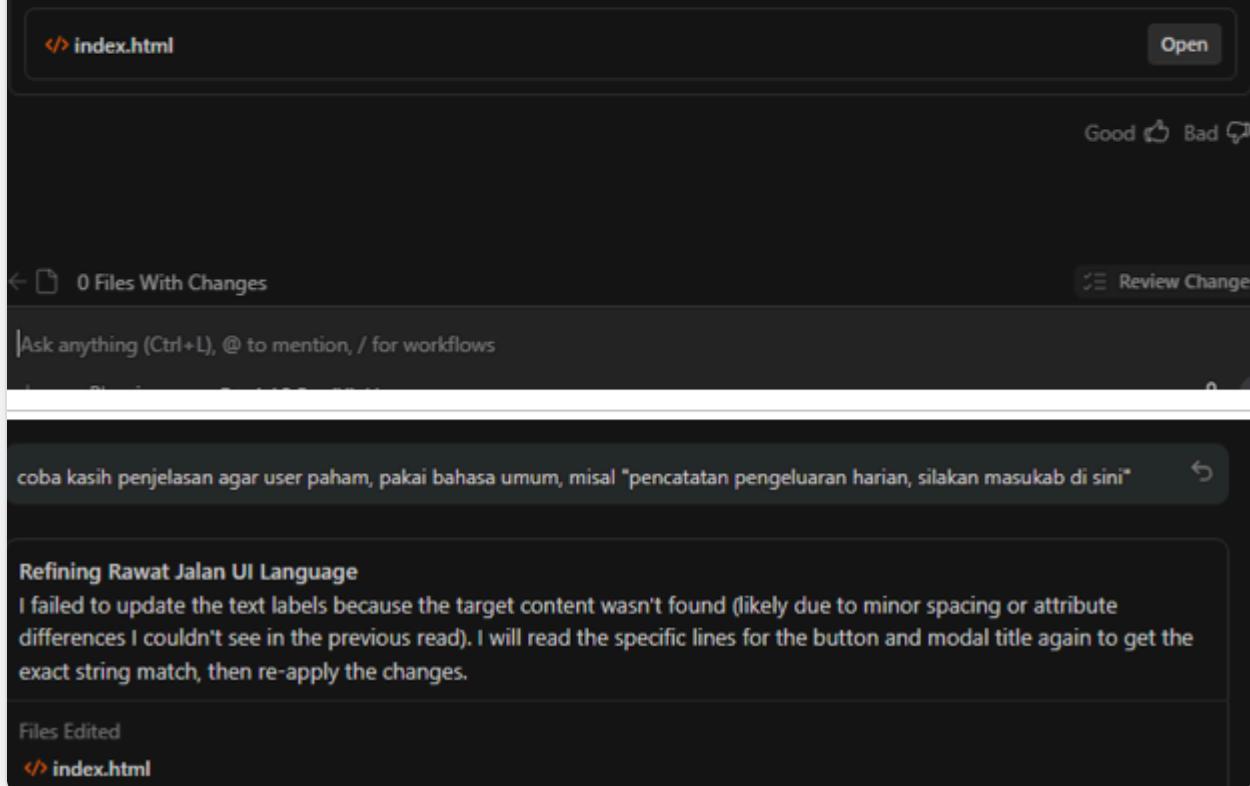
Setelah plan disetujui, AI mulai mengimplementasikan fitur-fitur tersebut:

Fitur "Rawat Jalan Harian" Siap Diuji! 🎉

Sudah saya suntikkan fitur-fitur ini ke dalam app:

1. Tombol Suntik (+): Cek pojok kanan bawah. Klik untuk input pengeluaran harian.
2. Gamifikasi Dokter:
 - Coba input 20.000 (Jajan) -> Liat reaksinya.
 - Coba input 150.000 (Jajan) -> Dokter bakal ngamuk. 😠
3. Jatah Harian Card: Di dashboard sekarang ada kartu baru yang kasih tau sisa jatah jajan hari ini biar genap sampai akhir bulan.

Silakan dicoba, Dok! Jangan lupa refresh browsernya dulu. ✨



Implementation Summary

Dalam beberapa menit, AI sudah selesai menambahkan:

Tombol Suntik (Fast Action Button) dengan icon jarum

Gamifikasi Dokter dengan real-time feedback

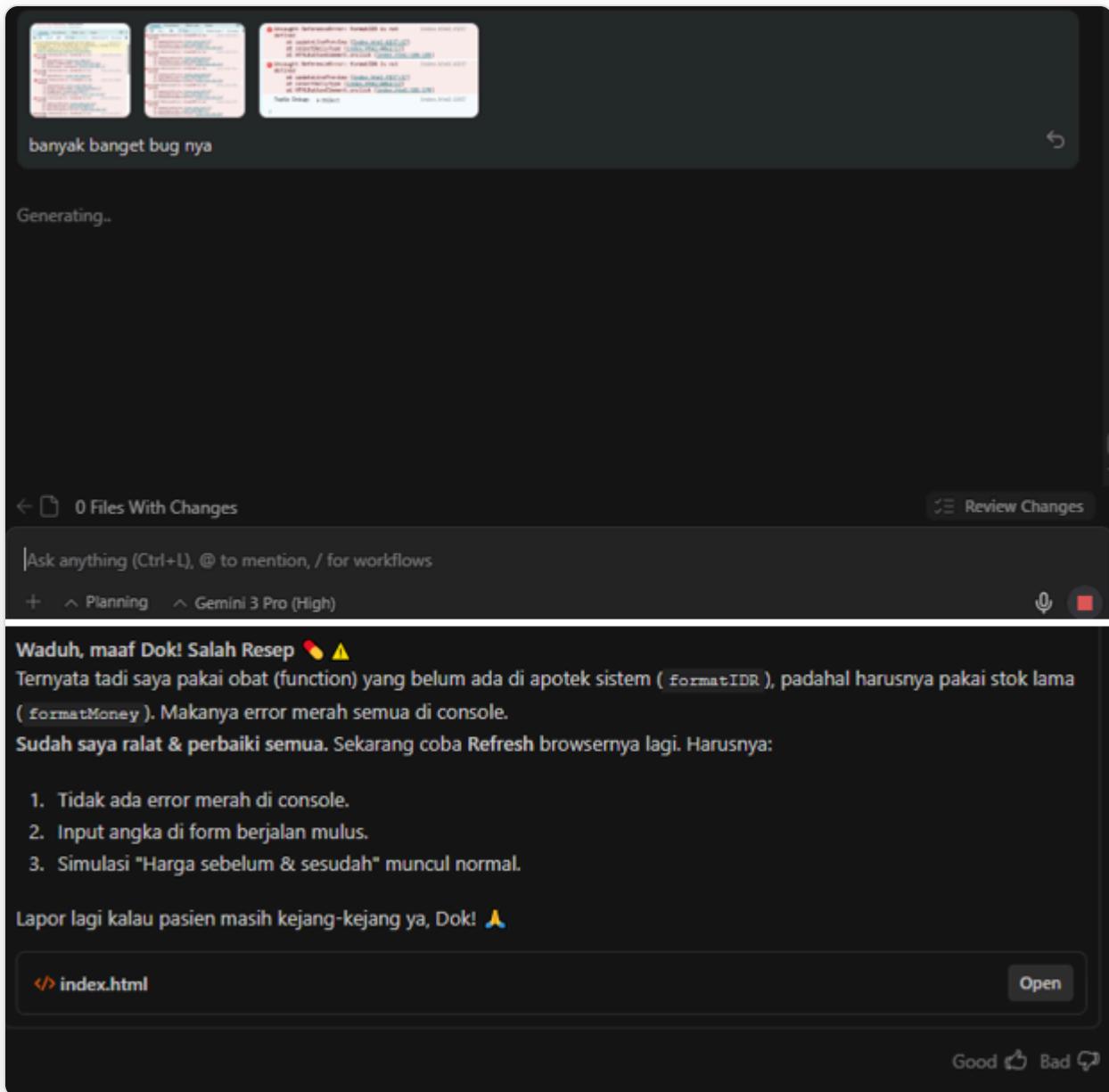
Jatah Harian Card yang menampilkan sisa budget

Saya hanya scroll membaca summary-nya. Belum ngecek hasilnya di browser.

5. Testing & Debugging: Ketika Ada Error

Ini bagian yang paling penting untuk dipahami: **tidak selalu langsung sempurna.**

Saat saya buka hasil code nya di browser dan tekan F12 (Chrome DevTools), saya melihat ada error:



Debugging Phase - Screenshot Error Console

Error: `formatMoney is not defined`

Daripada saya coba cari-cari code mana yang salah atau googling "how to fix formatMoney error", saya langsung screenshot console error nya dan paste ke Antigravity dengan caption:

"banyak banget bug nya"

Dan dalam beberapa detik, AI langsung:

Mendeteksi error dari screenshot (fungsi `formatMoney` belum didefinisikan)

Mengidentifikasi akar masalahnya

Memperbaiki code

Menjelaskan apa yang diperbaiki

Hasilnya: error hilang. Simulasi "Harga sebelum & sesudah" muncul normal.

Pelajaran dari Proses Ini

1. Brainstorming Bersama AI Itu Valuable

Saya tidak perlu datang dengan design document yang sempurna. AI bisa membantu mematangkan ide yang masih mentah.

Yang saya lakukan hanya menjelaskan masalah saya dengan jujur. AI yang suggest solusi.

2. Approval Loop Membuat Saya Tetap Kontrol

AI tidak langsung ngerjakan tanpa konfirmasi. Ada fase:

- Proposal Concept → Saya review → Approve/Revise
- Implementation Plan → Saya review → Proceed
- Code Generated → Saya test → Report bugs (kalau ada)

Jadi saya tetap yang pegang kendali, bukan AI yang ngacak-ngacak.

3. Debugging Itu Mudah dengan Screenshot

Tidak perlu paham error message yang cryptic. Screenshot console → paste → AI fix.

Ini menurunkan barrier untuk pemula. Kita tidak perlu "lancar bahasa error".

4. Proses Iteratif, Bukan One-Shot Perfect

Aplikasi tidak langsung jadi sempurna. Ada phases:

- User ask
- AI propose
- User review
- AI implement
- User test
- (kalau ada bug) User screenshot → AI fix

Dan itu normal. Yang penting prosesnya smooth.

Kesimpulan

Ini adalah dokumentasi nyata dari proses saya membuat aplikasi. Tidak di-edit, tidak dishortcut. Persis seperti ini lah cara kerjanya.

Dan yang perlu saya lakukan hanyalah:

Menjelaskan masalah dengan jelas

Membaca proposal AI dengan teliti

Testing hasil dengan jujur (screenshot kalau ada bug)

Tidak ada satu baris code pun yang saya tulis manual.

Tidak ada satu error message pun yang saya debug sendiri.

Tidak ada dokumentasi framework yang saya baca.

Semua yang saya butuhkan adalah kemampuan untuk **berkomunikasi masalah dengan baik**.

Itu saja.

Bab selanjutnya: Studi Kasus UPS Calculator - Aplikasi dengan 14 Logika Kompleks

Bab 4: Studi Kasus - Kalkulator UPS yang "Mustahil" Dikoding Manual

(Dari 14 Poin Logika Kompleks → Aplikasi Kerja dalam Waktu Singkat)

Saya ingin berbagi pengalaman membuat sebuah aplikasi yang, sejurnya, tidak mungkin saya buat sendiri dengan cara tradisional.

Ini adalah kalkulator rate UPS dengan 14 conditional logic yang saling terkait. Kalau saya harus belajar coding dari nol untuk bikin ini, mungkin butuh berbulan-bulan. Tapi dengan pendekatan "Antigravity", saya bisa menyelesaiakannya dalam waktu yang jauh lebih singkat.

Bukan karena saya jago coding. Tapi karena saya paham masalahnya dengan baik.

Memahami Kompleksitas

Sebelum mulai, saya ingin menunjukkan seberapa rumit aplikasi yang dimaksud. Ini bukan aplikasi "Hello World", tapi tool yang saya gunakan sehari-hari di bisnis logistik saya.

Berikut ini adalah planning notes saya saat pertama kali memikirkan aplikasi ini:

Requirement Planning (Dokumentasi Asli)

Buatkan kalkulator rate UPS dengan deskripsi

1. jika actual berat (bukan volumetric) $> 25 \text{ kg} < 71 \text{ kg}$, maka terkena AHS (additional handling surcharge) sebesar IDR 243,460 / collie
2. jika jenis package selain box (misal drum, silinder , maka akan terkena AHS sebesar IDR 243,460 / collie
3. jika gabungan Length + Girth ($2 \times \text{Lebar} + 2 \times \text{tinggi}$) $> 300 < 400$ maka akan terkena LPS(large Package Surcharge) sebesar IDR 1,058,200 / collie dan secara otomatis menjadikan minimum chargeable weight menjadi 40 kg / collie, serta menghilangkan AHS surcharge sebesar IDR 243,460 / collie
4. jika salah satu panjang $> 122 \text{ cm}$ dan atau lebar $> 76 \text{ cm}$, namun gabungan Length + Girth tidak melebihi 300, maka akan terkena AHS sebesar IDR 243,460 / collie
5. jika panjang melebihi 274 cm / collie , dan atau berat $> 70\text{kg} / \text{collie}$, dan atau gabungan pangjang + Girth ($\text{girth} = \{2 \times \text{height}\} + \{2 \times \text{width}\}$) $> 400 \text{ cm}$, maka akan memiliki 2 opsi
 - a. tetap di service UPS Saver / UPSExpedited, namun terkena OMX (Over Maximum) sebesar IDR 4,121,800 / collie + LPS sebesar 1,058,200 / collie, atau
 - b. ganti ke service UPS WWF,dan tidak terkena surcharge LPS, OMX
6. jika import, maka kena surcharge additional brokerage sebesar 112,890 / AWB
7. basic rate di kosongkan (diinput manual). bikin 2 jenis
 - a. $< 20 \text{ kg}$ chargeable weight = basic rate
 - b. $\geq 21 \text{ kg}$ chargeable weight = basic rate / kg

Planning Part 1 - 7 Poin Pertama

Ini baru 7 poin pertama dari total 14 requirement yang harus dipenuhi.

8. FSI (Fuel surcharge Index) diisi manual karena update weekly.

<https://www.ups.com/id/en/support/shipping-support/shipping-costs-rates/fuel-surcharges>

9. Surge fee sbb

to EUR = IDR 16,280 / kg

from EUR = IDR 7,252 / kg

from Americas = IDR 6,512 / kg

to Americas, ISMEA = IDR 16,280 / kg

to & from ASIA = IDR 1,480 / kg

to & from Israel = IDR 9,620 / kg

10. tambahkan additional surcharge optional (drop down list)

Extended Area = IDR 419,580 / kg or 7,992 / kg greater

Remote Area = IDR 468,420 / kg or 9,176 / kg greater

PEB = IDR 180,960 / AWB

Residential = IDR 56,684 / per AWB

Adult Signature = IDR 71,040 / per AWB

Delivery Confirmation = IDR 37,740 / per AWB

Alternate Broker = IDR 408,660 / per AWB

Duty Tax forward = IDR 290,080 / per AWB

Planning Part 2 - Surcharge Tambahan

Belum selesai. Ada Fuel Surcharge Index yang update mingguan, konversi mata uang untuk berbagai region...

Duty Tax forward = IDR 290,080 / per AWB

IPF (Intl Processing Fee) = IDR 37,000 / per AWB

Paper commercial Invoice = IDR 370,000 / AWB

11. buatkan kolom berat diisi manual, dimensi diisi manual, buat dalam 10 baris.lalu konversi ke berat, (length x width x height dalam cm) dibagi 5000 = menghasilkan berat. lalu ambil angka terbesar antara berat dan volumetric

12. basic rate + Surge fee + AHS surcharge + LPS + Brokerage import fee + FSI + VAT 1,1% = total rate. surcharge Additional Brokerage Import tolong tidak menjadi komponen penambahan rate untuk penghitungan FSI

13. chargeable weight ambil angka terbesar antara actual weight dengan volumetric weight. cara mengkonversi dimensi ke volumetric weight = panjang x lebar x tinggi dalam CM lalu dibagi 5000. lalu buatkan dua jenis kalkulator, 1 untuk berat 0,5 kg - 20 kg dimana basic rate nya bukan per kg; lalu yang ke-2 untuk berat >21 kg,dimana basic rate nya per kg

14. tolong tambahkan total collie di sebelah kanan tiap baris nya. lalu tambahkan nama negara origin, negara destinasi. juga pastikan jika dalam 1 baris ada berat yang melebihi 71 kg / collie, maka otomatis semua baris terkena minimum 71 kg. dan atau jika ada salah satu baris total Length + girth nya melebihi 400 , maka terkena minimum berat 71 kg di semua baris nya, serta defaultnya menjadi menggunakan service WWEF, sementara surcharge OMX dijadikan opsional 2 saja Jika menggunakan service WWEF, maka hilangkan otomatis AHS surcharge.

Planning Part 3 - Logika Final

Dan baru selesai di poin 14. Ada chargeable weight, volume weight, conditional collies, negara origin-destination...

Apa yang Membuat Ini Kompleks?

Dari requirement di atas, aplikasi ini membutuhkan:

- Perhitungan berat actual vs volumetric (ambil yang lebih besar)
- Deteksi jenis package otomatis (box vs non-box)
- Kalkulasi Length + Girth dengan multiple threshold
- Multi-tier pricing dengan nested conditions

- Service-specific rules (UPS Saver ≠ WWF ≠ Expedited)
- Auto-add surcharge untuk import shipment
- Input manual untuk basic rate dan FSI index
- Dropdown dengan 10+ opsi additional surcharge
- Konversi mata uang multi-region
- Weight/dimension converter otomatis
- Dual mode calculator (per kg vs per collie)
- Formula total calculation dengan 6+ komponen
- Conditional minimum chargeable weight
- Auto show/hide fields berdasarkan service selection

Kalau saya harus menulis semua logika if-else nya manual, saya bahkan tidak tahu dari mana harus mulai.

Pendekatan yang Saya Ambil

Saya tidak memikirkan "bagaimana cara ngoding ini." Saya hanya fokus pada satu pertanyaan:

"Apa yang harus terjadi?"

Saya sudah paham betul aturan bisnis UPS. Saya tahu kapan surcharge Additional Handling berlaku, kapan Large Package dikenakan, bagaimana menghitung chargeable weight.

Jadi saya tinggal menuliskan apa yang saya tahu, dalam bahasa yang saya pahami.

Prompt yang Saya Gunakan

Buatkan kalkulator rate UPS dengan deskripsi:

jika actual berat (bukan volumetric) >25 kg < 71 kg, maka terkena AHS

(additional handling surcharge) sebesar IDR 243,460 / collie

jika jenis package selain box (misal drum, silinder), maka akan terkena AHS

sebesar IDR 243,460 / collie

jika gabungan Length + Girth ($2 \times \text{Lebar} + 2 \times \text{tinggi}$) $> 300 < 400$ maka akan

terkena LPS(Large Package Surcharge) sebesar IDR 1,058,200 / collie dan secara

otomatis menjadikan minimum chargeable weight menjadi 40 kg / collie, serta

menghilangkan AHS surcharge sebesar IDR 243,460 / collie

[... seterusnya sampai poin 14]



Cukup itu saja. Saya tidak perlu tahu syntax `if-else`, tidak perlu tahu cara buat dropdown, tidak perlu tahu gimana caranya auto-hide field.

AI yang mengurus semua detail teknis tersebut.

Proses Pengerjaan

Setelah mengirim requirement lengkap ke AI:

AI membaca dan memahami semua 14 poin logika

AI menulis kode dengan ratusan baris conditional statement

AI membuat UI yang clean dan responsif

AI mengimplementasikan semua rumus perhitungan

Saya? Hanya menunggu sambil membaca penjelasan yang diberikan AI tentang apa yang sedang dibuatnya.

Hasil dan Pengujian

Aplikasi selesai dalam hitungan menit. Saya langsung test dengan data real dari customer:

Test 1: Berat 30kg

Expected: Kena AHS (>25kg) = Rp 243,460

Result: ✓ Benar

Test 2: Dimensi 120x60x60cm (Length+Girth = 360cm)

Expected: Kena LPS (>300) = Rp 1,058,200 + auto min weight 40kg + AHS hilang

Result: ✓ Benar, bahkan AHS otomatis terhapus

Test 3: Import shipment

Expected: Auto-add brokerage Rp 112,890/AWB

Result: ✓ Benar

Semua 14 poin logika berjalan sempurna.

Iterasi dan Perbaikan

Saat testing, saya menemukan ada yang kurang: field untuk tracking number.

Saya tinggal menambahkan satu kalimat:

"Tambahin field input 'Tracking Number' di bagian atas form."

10 detik kemudian, field baru muncul. Terintegrasi dengan baik dengan code yang sudah ada.

Kemudian saya ingat perlu tombol "Reset":

"Tambahin tombol 'Reset' untuk clear semua field."

15 detik kemudian, tombol "Reset" ditambahkan dan berfungsi sempurna.

Ini yang saya maksud dengan "iterasi tanpa kerumitan". Tinggal ngomong, langsung jadi.

Pelajaran yang Saya Ambil

Dari pengalaman membuat kalkulator UPS ini, saya menyadari beberapa hal:

Yang Penting Bukan Skill Coding

Yang penting adalah pemahaman terhadap masalah. Saya paham betul aturan surcharge UPS karena itu bisnis saya. Pemahaman itu yang menjadi nilai.

Fokus pada "What", Bukan "How"

Saya tidak perlu tahu bagaimana AI menulis `if ((weight > 25 && weight < 71) || packageType != 'box')`. Saya hanya perlu tahu: "Kapan AHS dikenakan?"

Iterasi Itu Mudah

Tidak ada takut salah atau "wah harus ngulang dari awal". Tinggal bilang apa yang kurang, langsung ditambahin.

14 Poin Kompleksitas → 1 Prompt

Requirement yang kalau dikerjakan manual mungkin butuh ratusan baris code dengan puluhan if-else bersarang, cukup ditulis dalam bentuk daftar poin yang jelas.

Yang perlu saya lakukan hanyalah memahami masalah dengan baik dan menjelaskannya dengan jelas. Sisanya? AI yang mengurus.

Itu saja.

Bab 5: Seni "Mengadu" (Debugging Tanpa Pusing)

Saat Kode Tidak Jalan, Jangan Panik. Mengadu Saja.

"Mas, pas saya klik tombol 'Hitung', kok gak muncul apa-apa ya?"

"Mas, angkanya jadi NaN (Not a Number), kenapa nih?"

Selamat! Kamu baru saja bertemu dengan sahabat karib programmer: **BUG**.

Di dunia coding tradisional, bug adalah musuh yang harus dibasmi dengan cara menelusuri ratusan baris kode, sakit kepala, dan kurang tidur.

Di dunia **Antigravity**, bug adalah alasan kita untuk **memarahi** (**baca: memberi feedback**) ke AI.

Mindset Debugging: "Ini Salah Siapa?"

Ingin tahu, kamu adalah **Project Manager**.

Kalau bangunan miring, apakah kamu yang salah? Bukan. Tukangnya yang salah.

Apakah kamu yang harus ambil semen dan benerin temboknya? **TIDAK**.

Tugas kamu adalah **TERIAK**.

"WOI TUKANG! Temboknya miring 5 derajat ke kanan! Benerin sekarang!"

Prinsip yang sama berlaku di AI Coding.

Kalau aplikasinya error, jangan coba benerin sendiri (kecuali kamu ngerti).

Kembalikan ke AI dengan bukti errornya.

3 Langkah "Mengadu" yang Efektif

1. Cari Tahu Pesan Error-nya (Jadilah Mata-Mata)

Kalau web diklik diam saja, biasanya dia "teriak" di belakang layar.

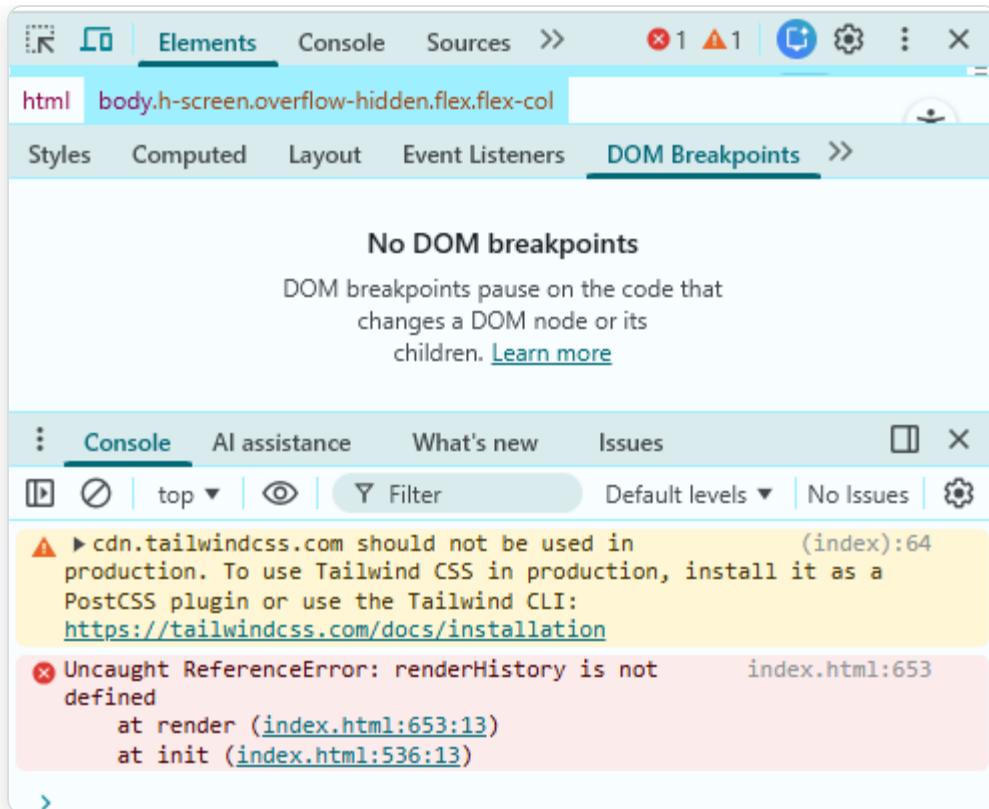
Cara melihatnya:

Tekan tombol **F12** di keyboard. (Atau Klik Kanan di layar -> Inspect).

Pilih tab bernama **Console**.

Apakah ada tulisan warna **MERAH**?

- Contoh: `Uncaught TypeError: Cannot read property 'value' of null`
- Atau: `ReferenceError: calculate is not defined`



Debug Console F12

2. Copy-Paste Error itu ke AI

Jangan cuma bilang "Gak bisa". AI bukan dukun.

Copy tulisan merah tadi, lalu chat ke AI:

"Bro, pas saya klik tombol hitung, muncul error ini di Console:

`Uncaught TypeError: Cannot read property 'value' of null`

Tolong perbaiki kodennya."

3. Jelaskan Kronologinya (Jika Tidak Ada Pesan Error)

Kadang gak ada error merah, tapi hasilnya ngaco (Logic Error).

Ceritakan detailnya:

"Gak ada error merah sih. Tapi pas saya input Berat 30 kg, harusnya kena Additional Handling Surcharge (Rp 211.640), tapi kok di hasil perhitungannya Surchargenya masih 0?"

Coba cek lagi logika `if-else` kamu untuk berat di atas 25kg."

Studi Kasus: Checkbox "Remote Area" yang Bandel

Ingat fitur checkbox Remote Area di Bab 4?

Misalnya pas kamu centang, eh total harganya gak nambah. Tetap ikut harga normal.

Jangan edit kodennya.

Langsung komplain:

"Halo. Saya udah coba kodennya.

Masalahnya: Ketika checkbox 'Remote Area' saya centang, Total Harga tidak bertambah Rp 452.290.

Sepertinya logika `if (checkbox.checked)` kamu belum jalan.

Tolong perbaiki."

AI akan langsung minta maaf, dan layar kiri (Code) akan bergerak sendiri memperbaiki logikanya.

Tugas kamu? Klik Accept lagi.

Intinya:

Di sini, **Skill Komunikasi > Skill Coding**.

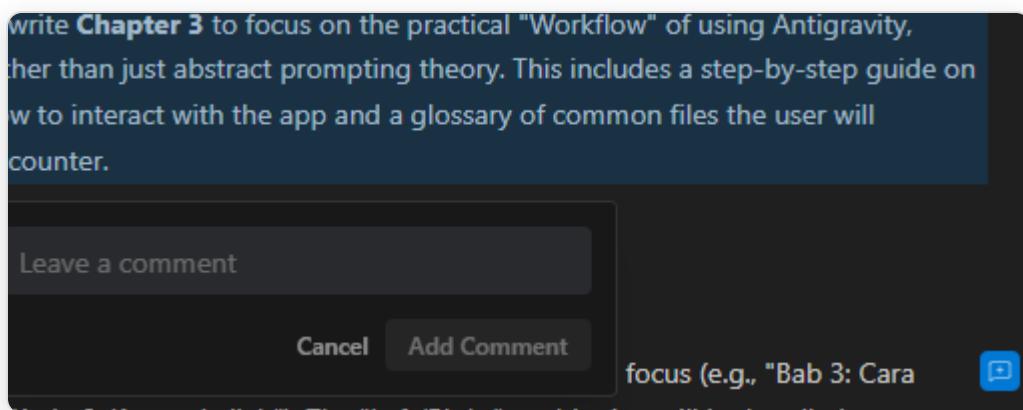
Semakin jago kamu menceritakan masalahnya (mengadu), semakin cepat aplikasinya jadi.

Jangan takut salah. AI gak bakal marah kalau kamu komplain 100 kali pun. Malah dia seneng dikasih kerjaan.

Tips Pro: Revisi Ala "Dosen Skripsi" (Tanda Biru)

Kalau kamu mau lebih detail (sedikit effort), coba perhatikan di sebelah kiri (di area Chat atau Code).

Terkadang ada **Kotak Biru Kecil** atau ikon komentar di samping teks/kode.



Fitur *Inline Comment*

Fitur apa ini?

Ini ibarat kamu lagi **mencoret-coret skripsi**.

Klik tanda birunya.

Tulis komentarmu spesifik di baris itu. Misal: *"Bagian ini tolong warnanya diganti merah."* atau *"Rumus ini salah, harusnya dikali 10."*

AI akan memperbaiki **hanya bagian itu** tanpa mengacak-acak bagian lain yang sudah benar.

Gunakan ini kalau kamu mau hasil yang super rapi dan presisi tanpa perlu menjelaskan panjang lebar di chat utama.

Bab 6: Go Live! (Dari Laptop ke Internet)

Jadikan Karyamu Abadi di Internet

Sekarang kamu punya **Kalkulator Pajak Import** yang canggih di laptopmu.

Tapi sayang dong kalau cuma kamu yang pakai. Gimana kalau teman kantormu mau pakai? Atau bosmu mau lihat?

"Waduh, harus sewa hosting mahal ya? Harus beli domain jutaan?"

TIDAK.

Ingin prinsip Antigravity: *High Impact, Low Effort, Zero Cost.*

Kita akan menayangkan website kamu ke internet dalam waktu kurang dari 60 detik. **Gratis.**

Langkah Ajaib: Netlify Drop

Kita akan menggunakan layanan bernama **Netlify**. Ini adalah "Malaikat Penolong" para web developer modern.

Step 1: Rapikan Folder

Pastikan file kodinganmu tadi (misal `kalkulator.html`) diganti namanya menjadi `index.html`.

Kenapa? Karena internet sudah sepakat bahwa "pintu gerbang" sebuah website itu namanya harus `index.html`.

Masukkan file itu ke dalam satu folder khusus, misalnya folder bernama `Project_Kalkulator`.

TIPS: Dimana Folder Saya?

Kalau kamu bingung file-nya tersimpan di mana, tanya saja ke AI:

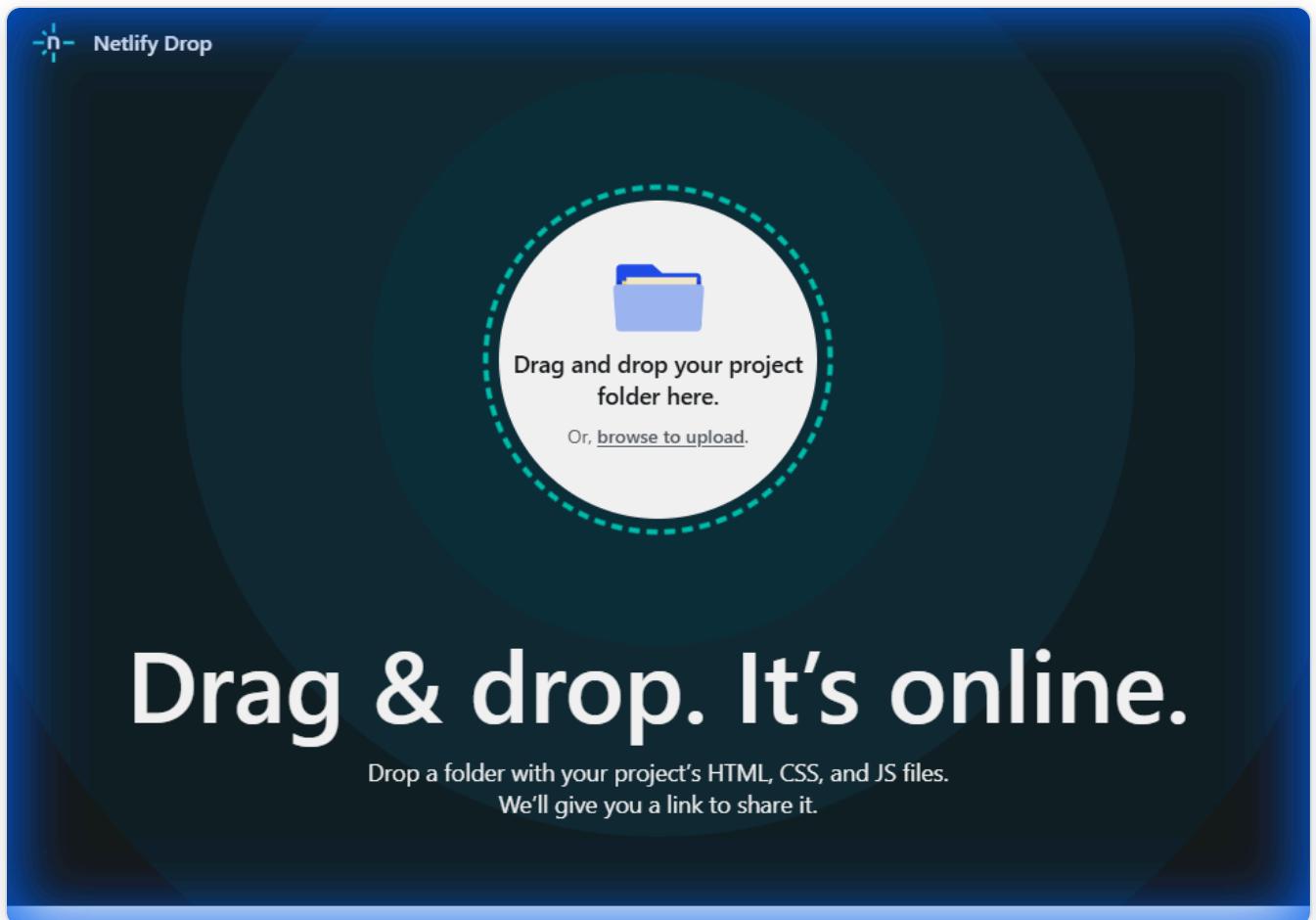
"Di mana lokasi folder proyek ini di laptop saya? Tolong buka foldernya."

AI akan memberitahu jalurnya atau bahkan ngebukain File Explorer buat kamu.

Step 2: Buka Netlify Drop

Buka browser, ketik: app.netlify.com/drop.

Kamu akan melihat area kotak putus-putus bertuliskan *"Drag and drop your site folder here"*.



Tampilan Netlify Drop

Step 3: Seret dan Lepas

Buka folder **Project_Kalkulator** kamu di komputer, **klik tahan (drag)** foldernya, lalu **lepas (drop)** ke kotak di website Netlify tadi.

Tunggu 5 detik...

Uploading...

Published!

Step 4: Ganti Nama (Optional)

Websitemu sudah online! Tapi namanya biasanya aneh, misal: **bambang-ganteng-x7z89.netlify.app**.

Klik "Site Settings" -> "Change Site Name".

Ubah jadi nama yang keren, misal: **kalkulator-pajak-budi.netlify.app**.

FAQ: "Gratisnya Sampai Kapan?" (PENTING)

Banyak yang tanya: "*Mas, ini beneran gratis? Ada limitnya gak?*"

Jawabannya: **Gratis Selamanya**, tapi dengan sistem **Kuota (Bandwidth)**.

Logika Gampangnya:

Bayangkan Netlify itu seperti **Provider Internet**.

- Kamu dikasih **Kuota Data 100 GB / bulan**.
- Setiap kali ada orang yang buka websitemu, kuotanya berkurang sedikit (paling cuma beberapa Kilobyte).
- **100 GB itu Berapa Banyak?** Itu SANGAT BANYAK.
- Kalau websitemu cuma Kalkulator sederhana, butuh **Jutaan Pengunjung** buat ngabisin kuota itu dalam sebulan.

Kesimpulan:

Selama websitemu bukan sekelas Tokopedia atau Facebook yang dikunjungi satu Indonesia tiap detik, kuota gratis ini **RASANYA SEPERTI UNLIMITED**. Aman!

Penutup: Kamu Sudah Jadi Developer!

Sekarang coba buka link itu di HP. Kirim ke Grup WhatsApp kantor.

"Eh guys, gue iseng bikin alat hitung pajak nih. Cobain deh."

Lihat reaksi mereka.

Mereka akan kagum. *"Gila, lo jago coding sekarang?"*

Jawab saja dengan senyum tipis:

"Ah enggak kok. Cuma iseng aja."

Padahal di balik layar, kamu tahu rahasianya.

Kamu tidak menghapal ribuan syntax. Kamu tidak begadang debugging error titik koma.

Kamu hanya **memerintah AI** dengan cerdas (Project Manager Mindset), dan membiarkan dia melakukan kerja kerasnya.

Inilah jalan ninjamu sekarang.

Selamat datang di dunia **Antigravity Developer**.

Bukti Nyata (Result)

Masih ragu?

Sebagai bukti bahwa metode ini berhasil, berikut adalah daftar aplikasi yang berhasil saya buat sendiri dalam waktu **1 Bulan Terakhir** (tanpa basic coding).

Sebagian saya pakai sendiri, sebagian lagi **Pesanan Client (CUAN Beneran!)**.

Dan semua ini saya kerjakan **selepas jam kantor (9-to-5)**, cuma bermodal laptop dan kopi.

Berikut daftarnya:

UPS Kalkulator (Hitung harga Export/Import otomatis).

Kalkulator Duty & Tax Impor Indonesia.

Sales CRM Offline (Manajemen pelanggan tanpa internet).

Sales Dashboard (Drag & drop file Excel, langsung jadi grafik).

Aplikasi Leads Tracker (Monitoring prospek untuk Sales).

Aplikasi Kasir Warung (POS Sederhana).

Pencatatan Order Laundry Pakaian.

Pencatatan Order & Track Laundry Sepatu.

Tracker & Report Team Collector (Monitoring lapangan).

Excel Converter (Alat konversi data Excel jadi lebih compact).

Semua ini dibuat dengan modal: **Nanya ke AI**.

Penasaran detailnya?

Silakan lihat-lihat portofolio lengkap saya di: zensheet.my.id

Level Selanjutnya: Tantangan Dunia Developer

Tentu saja, perjalananmu tidak berhenti di sini.

Apa yang kamu pelajari di buku ini adalah **Level 1 (Basic)**.

Di luar sana, dunia Programming masih sangat luas dan "liar".

Jika kamu ingin naik level, inilah beberapa istilah keren yang mungkin perlu kamu pelajari (tanya ke AI) nanti:

1. Database Aman (**Supabase / Firebase**)

Aplikasi yang kita buat di buku ini datanya masih disimpan di Local Storage (browser). Kalau Cache dihapus, data hilang.

- **Tantangan:** "Halo AI, gimana caranya bikin aplikasi yang datanya tersimpan di Cloud (Database) supaya aman dan bisa login pakai email?"

2. Version Control (GitHub)

Kalau aplikasimu makin besar, kamu butuh "Mesin Waktu". Kalau kodinganmu error, kamu bisa mundur ke versi kemarin.

- **Tantangan:** "Halo AI, ajari saya cara upload kode ke GitHub biar aman dan terlihat profesional."

3. Manajement Banyak Aplikasi (Cloudflare / Vercel)

Punya 10 aplikasi? 100 aplikasi? Masa mau diurus satu-satu manual?

Ada tools canggih bernama Cloudflare atau Vercel untuk mengatur semua "kerajaan" aplikasimu dalam satu dashboard. Gratis, cepat, dan aman dari serangan hacker.

- **Tantangan:** "Halo AI, gimana caranya deploy aplikasi ke Cloudflare Pages?"

4. API Integration (Menyambungkan Sesuatu)

Bayangkan aplikasimu bisa ngirim WhatsApp otomatis? Atau bisa narik data langsung dari Google Sheets? Itu namanya API.

- **Tantangan:** "Halo AI, buatkan fitur kirim pesan WA otomatis saat tombol ini diklik."
-

What's Next?

Giliranmu. Masalah apa yang bikin pusing di sekelilingmu?

Jadikan aplikasi. Gratis. Sekarang.

"Halo AI, saya mau bikin..."

Bonus Chapter: The Antigravity Prompt Cheat Sheet

"Contekan Mantra" untuk Memerintah AI

Ini adalah "Senjata Rahasia" Anda. Tidak perlu hafal coding, cukup copy-paste mantra ini untuk membuat AI bekerja 10x lebih efektif.

1. Mantra "Arsitek" (Untuk Memulai Project)

Gunakan saat: Baru mau bikin aplikasi dari nol.

"Halo, saya ingin membuat aplikasi berbasis web untuk [Tujuan Aplikasi, misal: kasir warung].

*Tolong berikan saya **struktur folder** dan **file apa saja** yang dibutuhkan.*

Gunakan HTML, CSS, dan JS vanilla saja agar mudah. Buat desainnya modern dan responsif untuk HP."

2. Mantra "Tukang Dekor" (Untuk Mempercantik Tampilan)

Gunakan saat: Tampilan aplikasi masih kaku seperti tahun 1990.

*"Tolong ubah desain file CSS-nya. Saya ingin gayanya '**Modern & Clean**'.*

Gunakan font 'Inter' atau 'Poppins' dari Google Fonts.

Berikan efek bayangan (box-shadow) lembut pada kartu dan tombol.

Pastikan tombolnya enak dipencet di layar HP (padding besar)."

3. Mantra "Satpam Logic" (Untuk Mencegah Error)

Gunakan saat: Ada hitungan matematika yang salah atau error aneh.

"Coba cek lagi logika hitungan di fungsi `hitungTotal` .

Pastikan kamu [Jelaskan aturan, misal: tidak boleh ada angka negatif].

Buatkan saya fungsi 'Guardian' untuk memfilter input yang aneh-aneh.

Jika ada error, tolong jelaskan kenapa, dan berikan kode perbaikannya secara lengkap (jangan potong-potong)."

4. Mantra "Detektif Error" (Saat Mentok)

Gunakan saat: Aplikasi tidak jalan, layar putih, atau console merah semua.

"Saya dapat error ini di Console: [Paste Error Disini].

Tolong analisa penyebabnya. Jelaskan kepada saya seperti menjelaskan ke anak umur 5 tahun.

Lalu perbaiki kodennya. Saya tidak mahu tahu teknisnya, pokoknya harus jalan."

5. Mantra "Deploy" (Saat Mau Pamer)

Gunakan saat: Project sudah jadi dan mau di-online-kan.

"Project saya sudah selesai. Semua file ada di folder ini.

Pastikan file utamanya bernama `index.html` .

Apakah ada file gambar atau script yang path-nya salah (broken link)? Tolong cek dan perbaiki sebelum saya upload ke Netlify."

Tips Tambahan: "The Golden Rule"

Jangan pernah bilang: "*Buatin aplikasi dong.*" (Terlalu umum, AI bingung).

Tapi bilanglah: "*Buatin tombol warna merah yang kalau diklik dia akan memunculkan alert 'Halo'.*" (Spesifik, AI senang).

Jadilah Bos yang Spesifik. Anak buah (AI) akan bekerja sempurna.

Tentang Penulis

Siapa di balik buku ini?

Saya bukan Guru, Dosen, apalagi Hacker.

Saya hanya seorang praktisi Logistik yang "males" (baca: ingin kerja cerdas).

Saya percaya teknologi harusnya memudahkan hidup kita, bukan malah bikin pusing dengan coding yang rumit.

Sehari-hari, saya sibuk mengurus logistik, sambil sesekali menyeruput kopi dan menyuruh AI mengerjakan tugas-tugas teknis saya.

Butuh Teman Curhat (Soal Antigravity)?

Saya tahu belajar sendirian itu kadang membingungkan.

Kalau kamu mentok, bingung, atau berhasil bikin aplikasi keren dan mau pamer, silakan hubungi saya.

Saya belum buka kelas Zoom (karena jujur, saya mager dan agak gaptek kalau harus live), tapi kita bisa ngobrol santai via teks:

- Email: zensheet.id@gmail.com
- WhatsApp: [0812-9065-0963](tel:0812-9065-0963) (*Chat Only ya, saya pemalu kalau nelpon*)

Terima kasih sudah berinvestasi ilmu lewat buku ini.

Jika teknik ini bisa menghemat waktu kerjamu ratusan jam ke depannya, semoga ini menjadi investasi terbaikmu tahun ini.

P.S. (Jalur VIP untuk yang Super Sibuk)

Meskipun buku ini mengajarkan caranya, saya tahu ada dari kalian yang ingin hasil **INSTAN & POWERFUL**.

Kalau kamu butuh aplikasi jadi tanpa pusing mikirin prompt, **Hire Saya Saja**.

Saya membuka jasa pembuatan aplikasi custom.

Spesialisasi saya adalah membuat aplikasi yang **datanya tersimpan AMAN & RAPI** (bisa connect ke **Google Sheets** atau **Cloud Database**).

Jadi datanya nggak cuma nyangkut di HP, tapi bisa diolah lagi di Excel kantor.

Kamu tinggal duduk manis terima beres, biarkan saya dan pasukan AI saya yang bekerja keras.

Hubungi kontak di atas untuk konsultasi.

Selamat berkarya!

- The Antigravity Developer

The Antigravity Developer - Jago Bikin Aplikasi Tanpa Jago Coding