Beyond Limits:

The AI Hack (GDG Hackathon HAU 2025)

Ideation Workshop

Speaker: Tjakoen A. Stolk

Ideation Workshop

The AI Hack (GDG HAU 2025)

Tjakoen Stolk (TJ) is an IT leader with over seven years of experience, having held roles ranging from Technical Lead to CTO. A graduate of **Holy Angel University**, he specializes in strategic leadership and software architecture, with a proven track record of building scalable platforms from the ground up. Currently, he drives high-tech solutions as a Technical Lead at **Career Team** and teaches software engineering as a Part-Time Instructor at his alma mater.

Objectives: This discussion focuses on the methodical approach, planning, and execution, rather than the initial ideas themselves.

Ideation Playbook

From Blank Page to Winning Concept: A Structured Approach

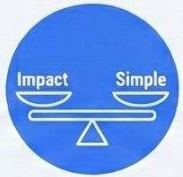
Key Benefits of the Playbook

Holistic Approach

- Moves beyond just the "idea" phase
- Guides team through Technology, Preparation, and Presentation Plan

Practicality and Realism

- Encourages focus on simple, relevant, and impactful ideas
- Ideas with a real chance of winning
- Avoids overly complex or gimmicky concepts



Emphasis on Preparation

- Tackles key time sinks before the event
- Addresses internet issues and project setup
- Ensures readiness before the start

Forward-Thinking Planning

- Includes Standard Operating Procedures (SOPs)
- Prevents miscommunication
- Ensures a smooth and efficient workflow

Clear and Concise

- Direct and easy to follow language
- Allows team to focus on building, not planning
- Reduces cognitive load

Validating Concepts

From Blank Page to Winning Concept: A Structured Approach

Is it Relevant? Does the concept significantly address a vital and widespread real-world need?

Is it "Frontier" AI? Does it go beyond simple text generation? (e.g., Computer Vision, Real-time reasoning)

Is it Responsible? Can we explain *why* the AI made a decision?

Is it Feasible? Can we build the core MVP in 24 hours?

Relevance > CHECKING LOCAL_CONTEXT...

The Question: Does this solve a problem here?

The Rule: If it works in Silicon Valley but not in Pampanga, kill it.

Avoid: Tech for tech's sake. (e.g., "*Smart Trash Cans*": impressive tech, but who maintains them? Is it practical?)

Target: Context-Aware Solutions for Real Friction

- **Systemic Challenges** *(The things that stop the city from moving)*: Focus on resilience and infrastructure. (e.g., Flood prediction & management, Jeepney/Trike logistics, Farm-to-Market supply chains).

- **Niche Frustrations** *(The daily hurdles for specific groups)*: Focus on underserved needs and accessibility. (e.g., PWD navigation on non-standard sidewalks, Financial tools for irregular income earners, Digitizing Palengke commerce).

Relevance > ANALYZING NOVELTY...

The Question: Is this just a ChatGPT wrapper?

The Pivot:

- Boring: "An app that chats about recipes."
- Frontier: "An app that sees ingredients and invents a recipe."

Requirement: Use Multi-modal (Vision/Audio) or Reasoning.

Suggestions to push your AI concept:

- **Enable "Agency" with External Tools:** You can give your AI the power to act by integrating services like Calendars, Maps, or Weather APIs, allowing it to actively book appointments, calculate real-time logistics, or check forecasts instead of just talking about them.
- **Deploy a "Critic" AI for Verification:** You can utilize a separate AI model to act as a reviewer that strictly questions the first AI's work to catch errors before the user sees the result.

Feasibility

> ESTIMATING BUILD_TIME...

The Question: Can the Core Feature be built by 5 PM tomorrow?

The Trap: Complex hardware, intricate 3D games, or massive datasets.

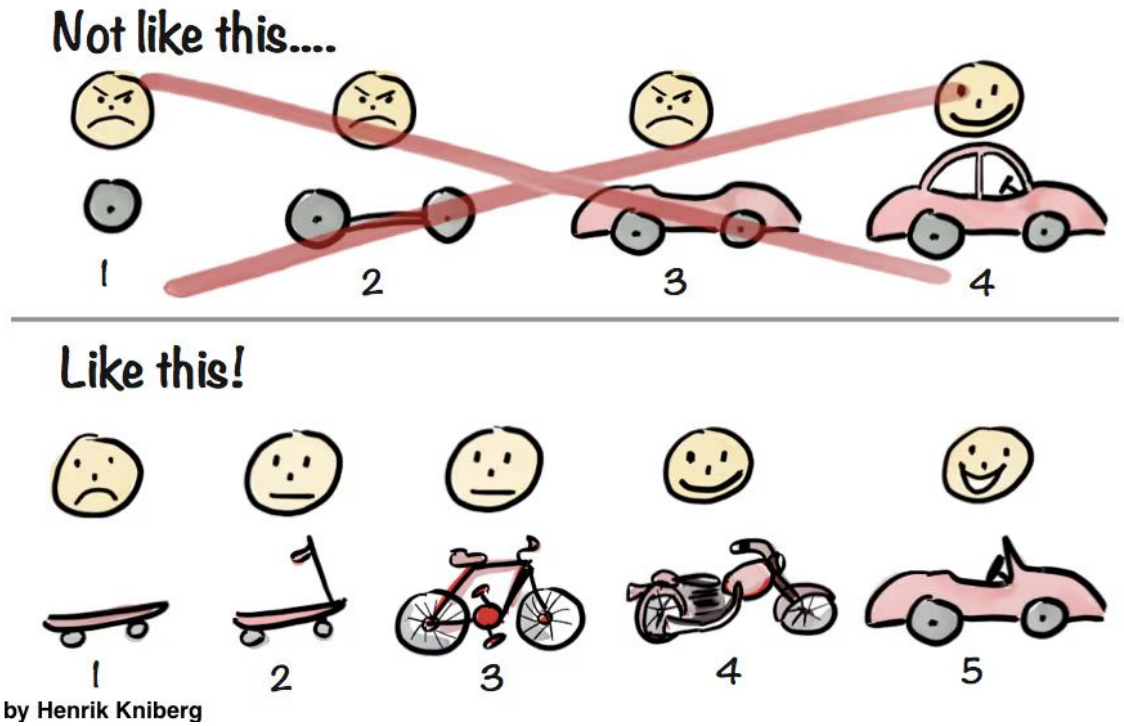
The Goal: A "Minimum Viable Product" (MVP).

Rule:

Simple App + Powerful Concept

>

Complicated App + Buggy Code.



Responsibility > VERIFYING ETHICS...

The Question: Can we trust the AI?

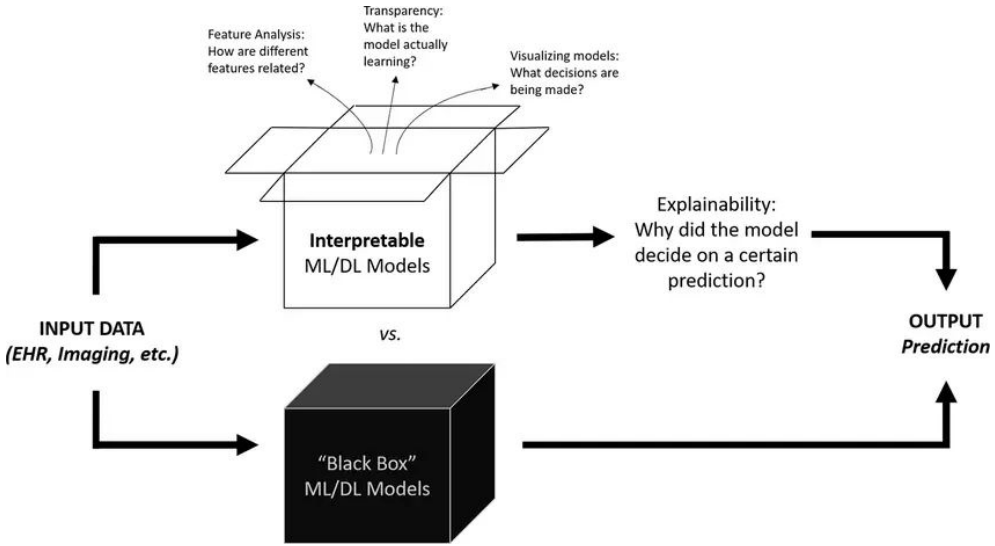
The Feature: "Explainable AI".

Requirement: The user must know why the AI made a decision.

The 3-Step Trust Loop:

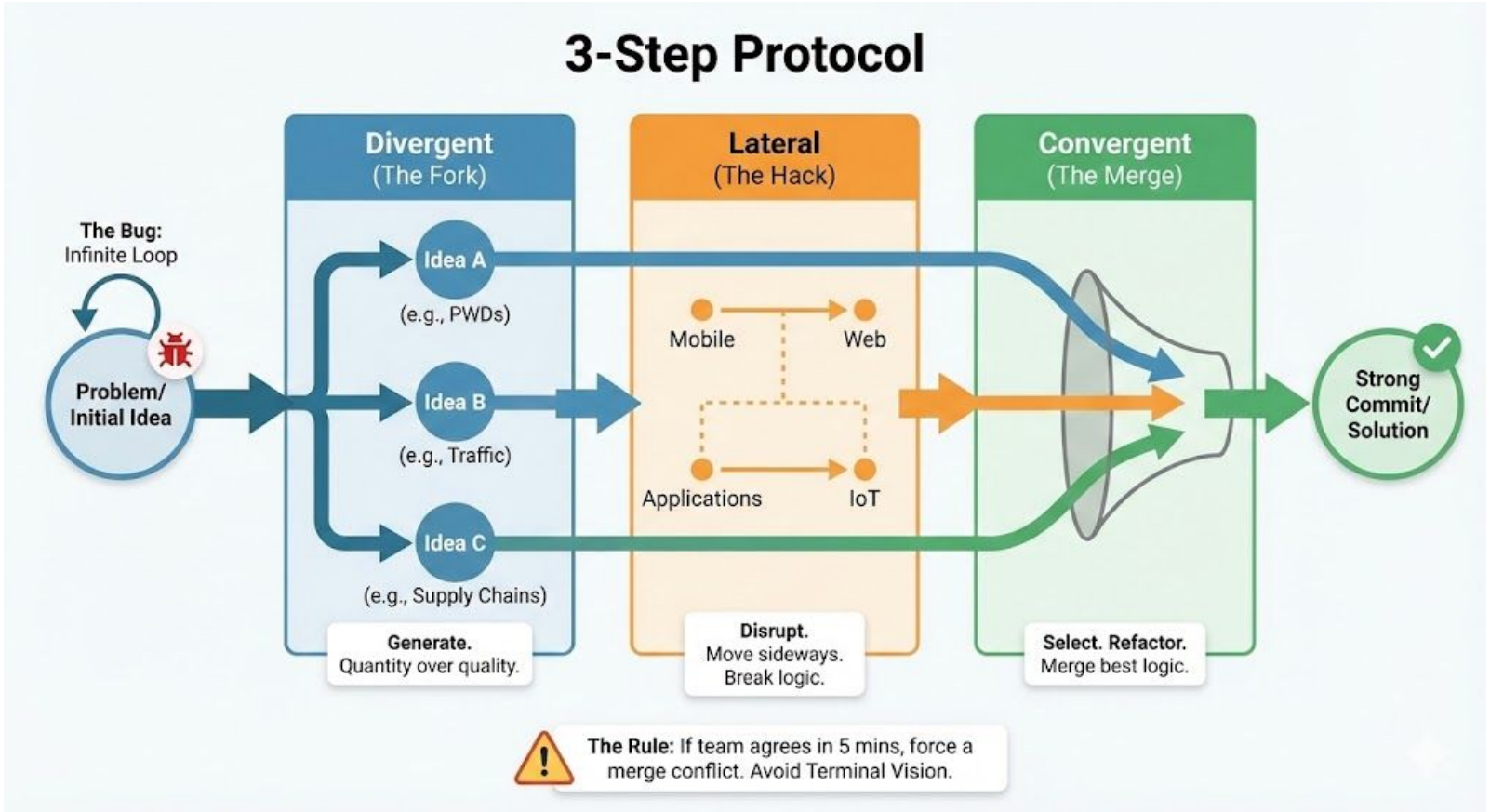
- **The Decision:** AI recommends a specific treatment.
- **The "Why":** Logic is exposed (e.g., "Patient has hypertension").
- **The Proof:** Source is cited (e.g., "Based on 2024 FDA Guidelines").

Bottom Line: If the AI cannot cite its source, the user cannot take the risk. **Trust = Accuracy + Transparency**



The “Terminal Vision” Trap

Avoid Premature Optimization.



Preparation

From Blank Page to Winning Concept: A Structured Approach

Who? Who is each team member and what can they bring to the table?

What? What references, resources, technologies, and integrations will you be using?

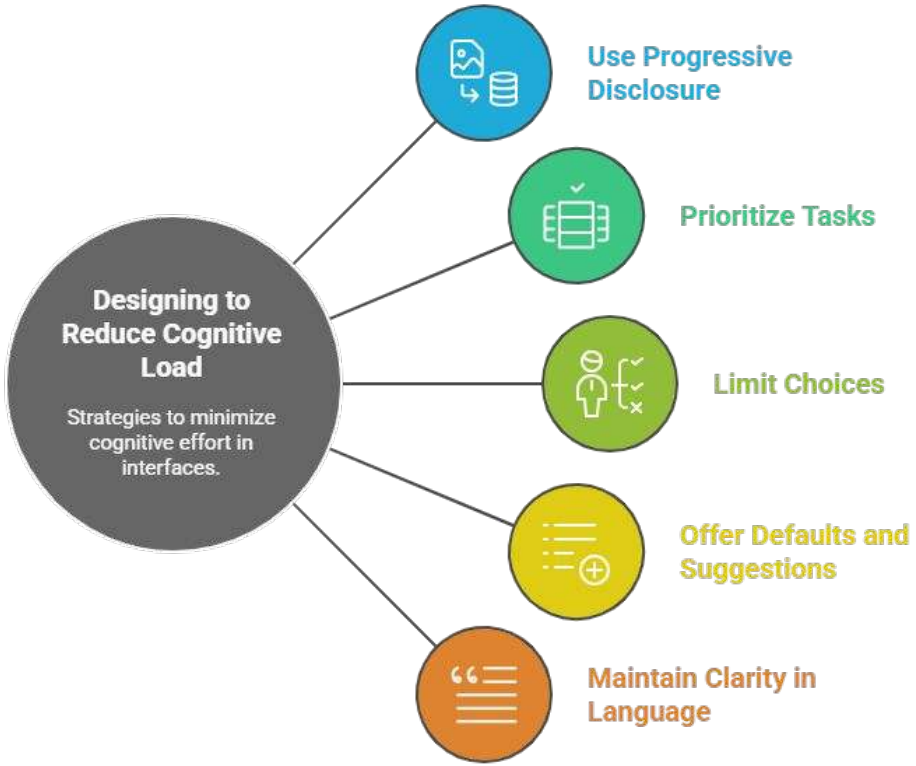
wHow? How will the team plan and manage the project as you develop it?

Reduce Cognitive Load Focus on Logic, Not Logistics

Your team has a limited supply of **"Decision Energy."** Don't waste it on things that don't ship code.

- **Kill Decision Fatigue:** Decide on naming conventions, folder structures, and color palettes before the timer starts.
- **Aggressive Scoping:** If a feature requires complex new logic just for a "nice-to-have," cut it. Complexity slows down momentum
- **Explicit Ownership:** Ambiguity creates mental drag. "*Who is doing X?*" is a question that shouldn't need to be asked twice.

"The goal of preparation is to make the execution automatic." Maximize your time to Plan as much as possible.

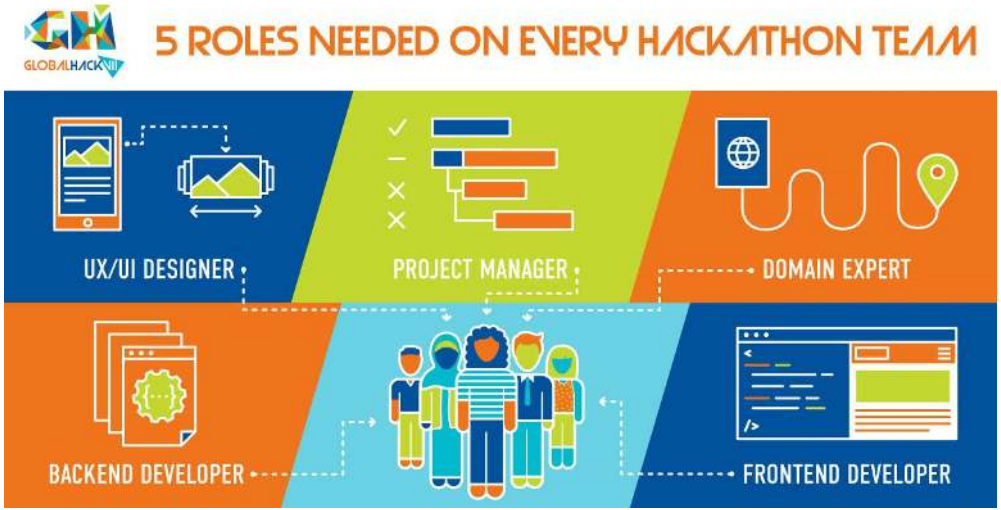


The Team Roster Know your lane.

Action: Write down each member, their specific talent, and their role for the project.

Domain Expert

- Integrate AI.
- Prompting and Generation
- Handle data input/output to the backend.



UI/UX Designer

- Create wireframes and high-fidelity mockups.
- Create the final presentation.

Backend Developer

- Develop the API.
- Manage the database (MongoDB/PostgreSQL).
- Ensure smooth data flow to frontend.

Frontend Developer

- Build user-facing mobile/web apps.
- Ensure responsive, clean interfaces.

The Doomsday Kit

Make sure you're prepared for anything.

Action: Make sure you're setup to work collaboratively and needed resources are easily accessible and backed up.

Use Git for Team Collaboration

- Everyone must master Basic Git (Commit, Push, Pull, Merge).
- **Resource**: Share a Git Cheat Sheet.

Create a Library containing:

1. **Docs:** HTML/PDF versions of the relevant technologies you're using (Ex. Flutter, Node.js, and Express documentation).
2. **Snippets:** Boilerplate code for API calls, data parsing, and common UI widgets.
3. **Assets:** All icons, images, and fonts saved locally.
4. **Emergency Guides:** Save articles on debugging common errors you usually face.

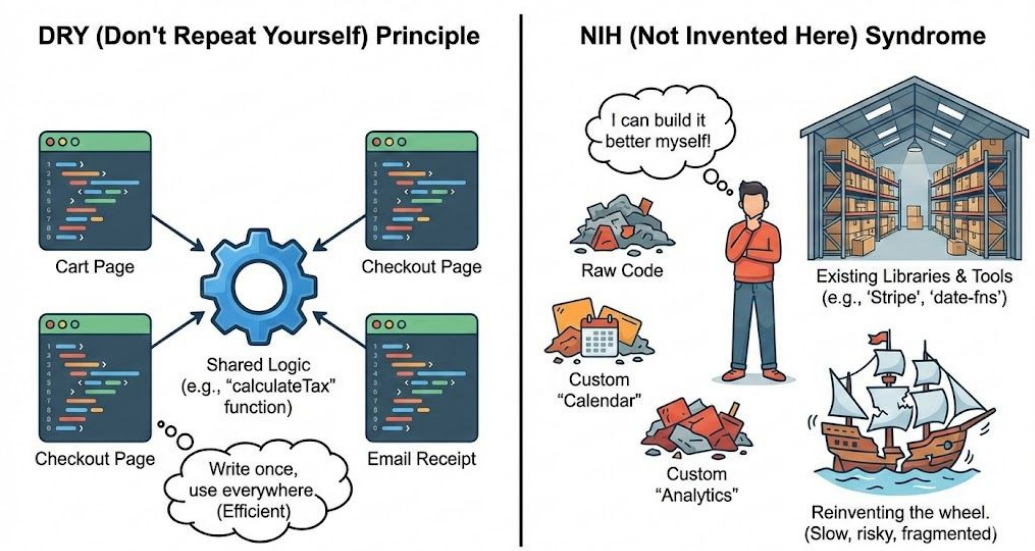
Best Practices Don't reinvent the wheel.

Avoid **NIH (Not Invented Here)** Syndrome. There is no need to reinvent the foundation. If it takes more than 2 hours to setup your project, then you're doing it wrong.

Apply DRY (Don't Repeat Yourself) principles to your infrastructure and code.

Technologies:

- **Github**: Search for community-vetted "Boilerplate" or "Starter Kits" specific to your tech stack and clone it to get started.
- **ShadCN**: An example of a UI library dependency that you can install for pre-built, copy-paste react components.
- **Previous Repos:** Your own past projects are the best templates. Delete the "business logic" and specific features. Keep the config, linting, and architecture.



A Note on Balance: Sometimes **DRY** can go too far. If you try to make code *too* reusable, it becomes complex and hard to read. Sometimes it's okay to repeat yourself a little bit to keep things simple.

Separation of Concerns Frontend

Action: Divide your application into distinct sections. Each section handles a specific job, allowing you to swap out the "sails" (Design) without rebuilding the "hull" (Structure).

Structure (The Hull): HTML

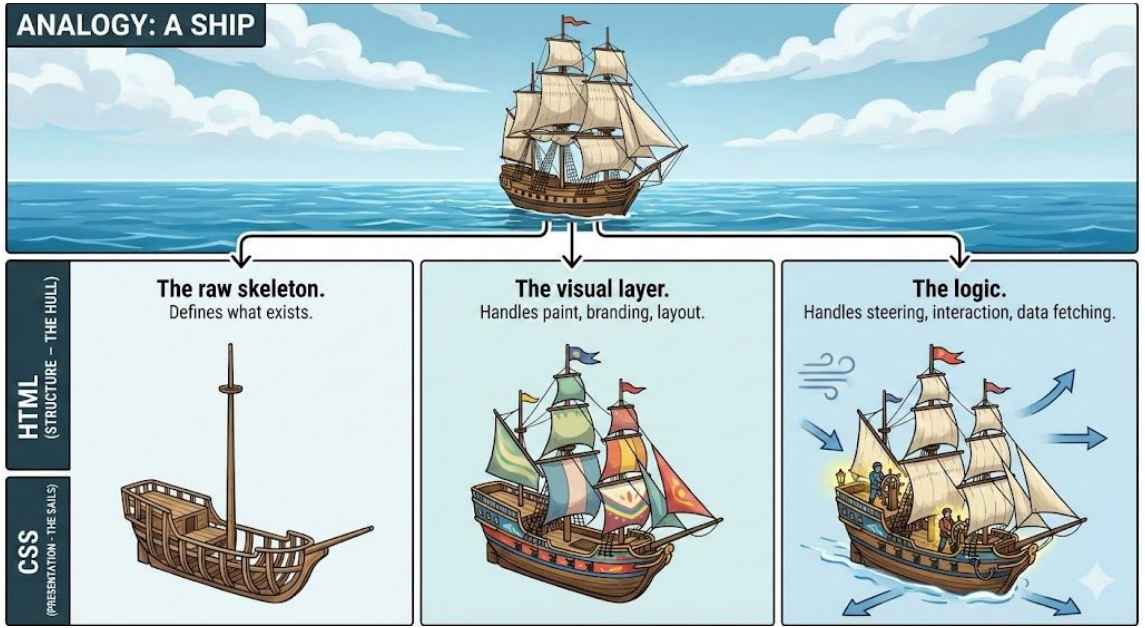
- The raw skeleton.
- Defines what exists (headings, paragraphs, buttons), not how it looks.

Presentation (The Sails): CSS

- The visual layer.
- Handles paint, branding, layout, and responsiveness.

Behavior (The Helm): JavaScript

- The logic.
- Handles steering, interaction, data fetching, and state management.



Maintenance Nightmare: Mixing styles and logic directly into HTML makes the code unscalable. Simple updates, like changing a brand color, require editing every single file instead of just one stylesheet. This tedious process increases errors and risks breaking the page structure just to adjust the visuals.

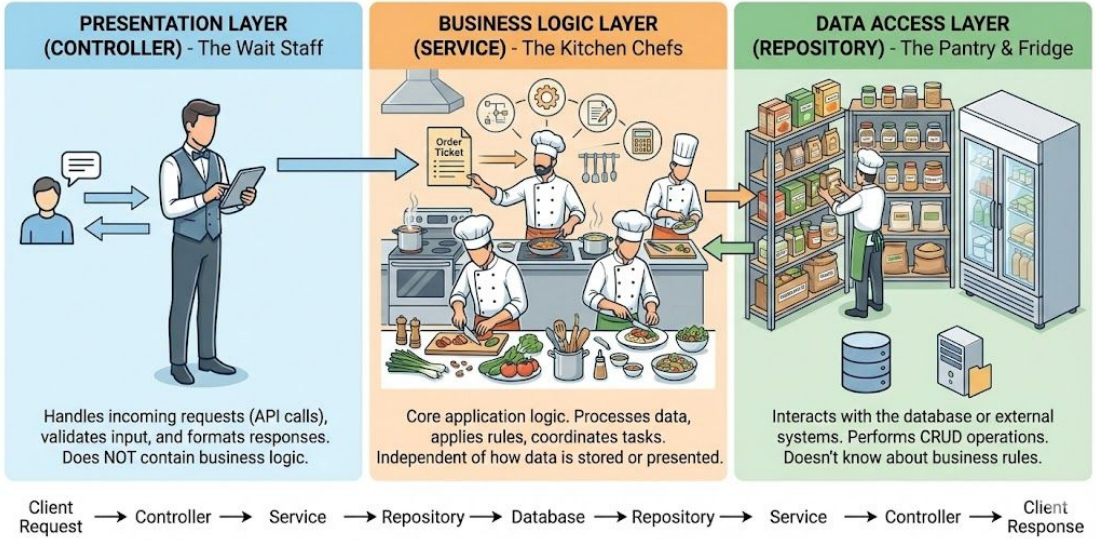
Separation of Concerns Backend

Action: Decouple your server architecture. This ensures your "Kitchen" (Business Logic) can change recipes without confusing the "Waiters" (API) or rearranging the "Pantry" (Database).

Interface (The Waiter): Controllers
The messenger. Receives requests (orders), checks if they are valid, and delivers the final response back to the client.

Logic (The Kitchen): Services
The processing center. Handles the business rules, calculations, and data transformation—the actual "cooking" of the application.

Storage (The Pantry): Database & Repositories *The warehouse*. Handles raw data persistence, retrieval, and ensures ingredients are stored and fetched efficiently.



Tightly Coupled Logic: Writing business logic inside API controllers destroys reusability. If you need to run that same logic elsewhere (like in a background job), you are forced to duplicate code. This makes testing difficult and leads to a brittle system where one change causes breaks across the application.

The Design System

Don't design from scratch.

Action: Make sure your Frontend Developer thinks as little as possible about design. Let him focus on coding, implementing layouts, state management, etc. Establish the *theme*, *layout*, *colors*, and other elements. *(Separation of Concerns)*

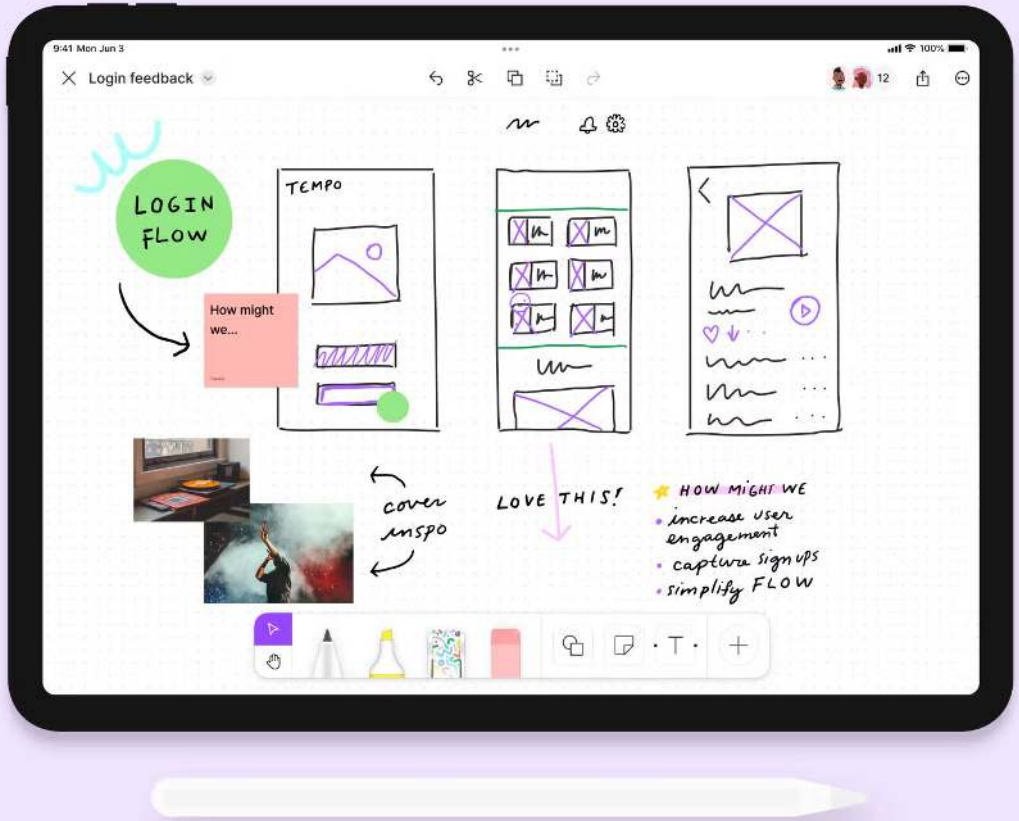
Technologies you can use:

- **Figma**: You can start from a design system template.
- **Canva**: Commonly used, easy to use and accessible.

Basic Steps:

- Low level Wireframe
- High Level UI/UX Design

Work concurrently with your Frontend, one Page/Screen at a time.



Tech Stack & Architecture

Stick to what you already know.

Action: Establish and agree on the best Tech Stack for your project (Frontend, Backend, Database, AI Integration)

Agree on a Standard Project Structure:

- Create a boilerplate folder structure
- **Data Models:** Pre-write standard classes (e.g., User.dart, string id, string name).
- **Resource**: Share a Git Cheat Sheet.

```
lib/
├── features/
│   ├── user/
│   │   ├── presentation/
│   │   │   ├── user_screen.dart
│   │   │   └── user_widget.dart
│   │   ├── data/
│   │   │   ├── user_repository.dart
│   │   │   └── user_service.dart
│   │   └── logic/
│   │       └── user_cubit.dart
│   └── products/
│       ├── presentation/
│       │   ├── product_screen.dart
│       │   └── product_widget.dart
│       ├── data/
│       │   ├── product_repository.dart
│       │   └── product_service.dart
│       └── logic/
│           └── product_bloc.dart
└── main.dart
```

The "Speed-to-Market" Stack (Serverless)
Best for: MVPs, Startups, and teams needing to ship fast with low backend maintenance.

- **Frontend:** Flutter (Mobile/Web)
- **Backend:** Firebase (BaaS - Backend as a Service)
- **Database:** Firestore (NoSQL)
- **AI Integration:** Firebase ML or Google Cloud Vertex AI API (direct call)
- **Auth:** Firebase Auth (Google, Apple, Email sign-in built-in)

The "JavaScript Everywhere" Stack (Web Standard)

- **Frontend:** React or Next.js
- **Backend:** Node.js (Express)
- **Database:** MongoDB (NoSQL)
- **AI:** OpenAI API calls via Node
- **Why:** One language (JS) for the whole app; huge community support.

The Presentation Plan

Sell the dream, not the code.

The project is secondary. A great presenter sells the vision.

The 30-Second Elevator Pitch

- Refine the core idea down to two sentences.
- Use this to quickly hook mentors and judges before the formal pitch.

Q&A Prep

Brainstorm the "Kill Questions":

- “How is this different from existing apps?"
- "How will you monetize or scale?"

Standard Operating Procedures

From Blank Page to Winning Concept: A Structured Approach

API Design and Documentation

Code Review and Merging Strategy

Debugging Protocol

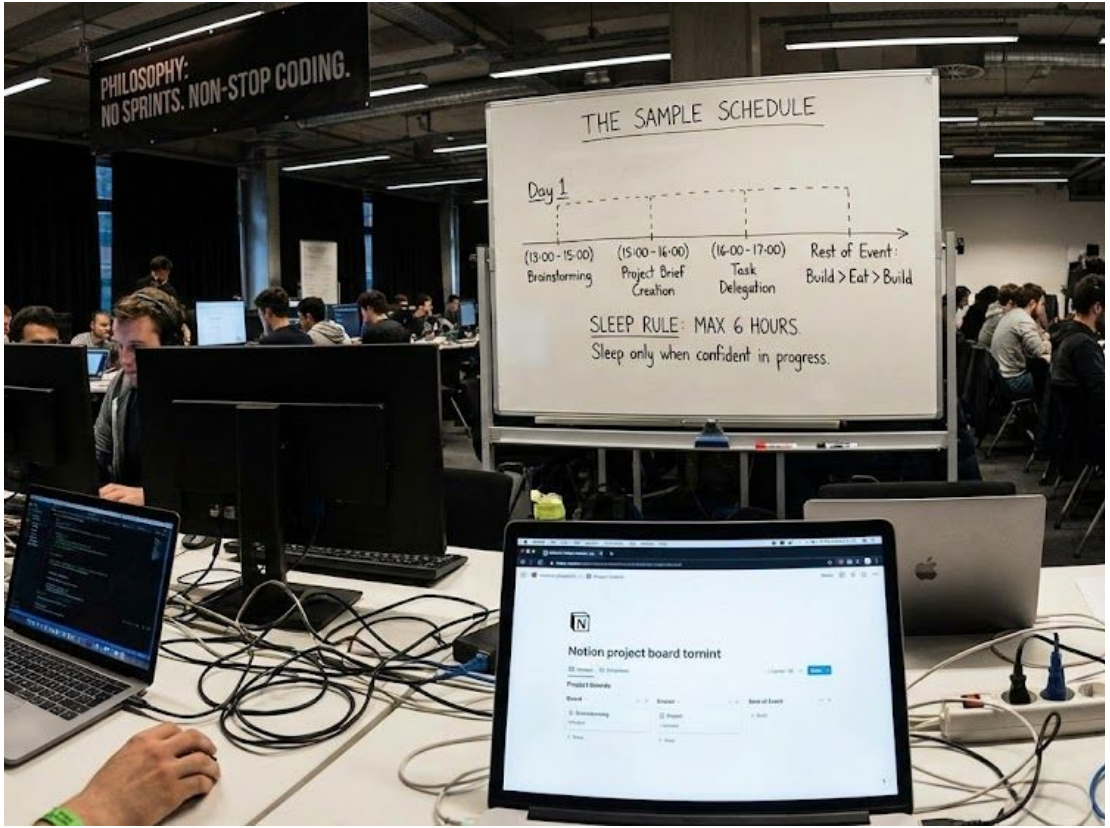
Project Management Workflow

Tool: PM Tools like *Notion* (Set up the template *now*, not later).

Philosophy: No sprints. Non-stop coding.

The Sample Schedule:

- **Day 1 (13:00 - 15:00):** Brainstorming.
- **Day 1 (15:00 - 16:00):** Project Brief Creation.
- **Day 1 (16:00 - 17:00):** Task Delegation.
- **Rest of Event:** Build > Eat > Build.
- **Sleep Rule:** Max 6 hours. Sleep only when you feel confident in progress.

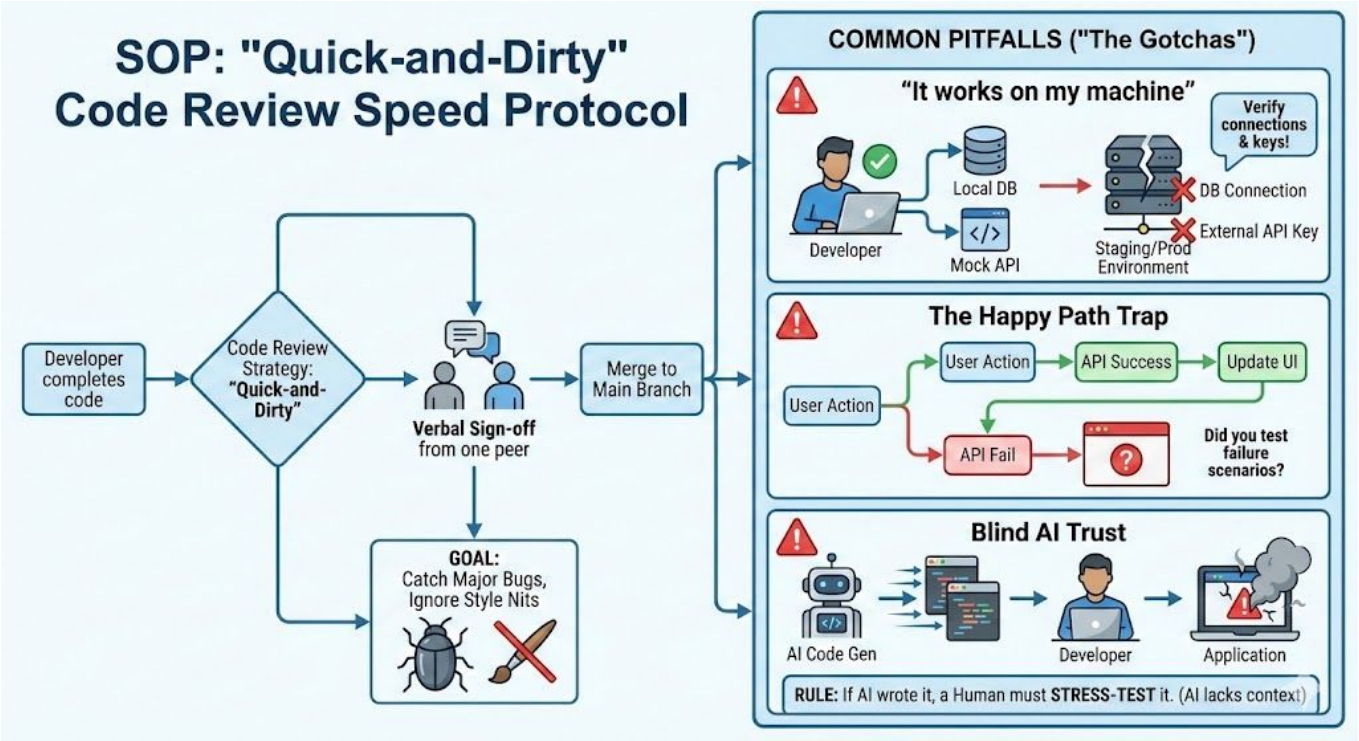


Speed Protocols

Standard Operating Procedures

"Quick-and-Dirty":
No formal reviews. Verbal sign-off from one peer before merging.

Goal:
Catch major bugs, ignore style nitpicks..



Speed Protocols

Standard Operating Procedures

If you are stuck on a bug for > 15 minutes...

STOP and ask a teammate.

Why? Momentum is everything. Do not let one person sink the ship.

Conclusion

From Blank Page to Winning Concept: A Structured Approach

Idea Examples

Final Notes

Q&A

REVIEWER.AI Idea Example

Target Market: Students

Pain Points:

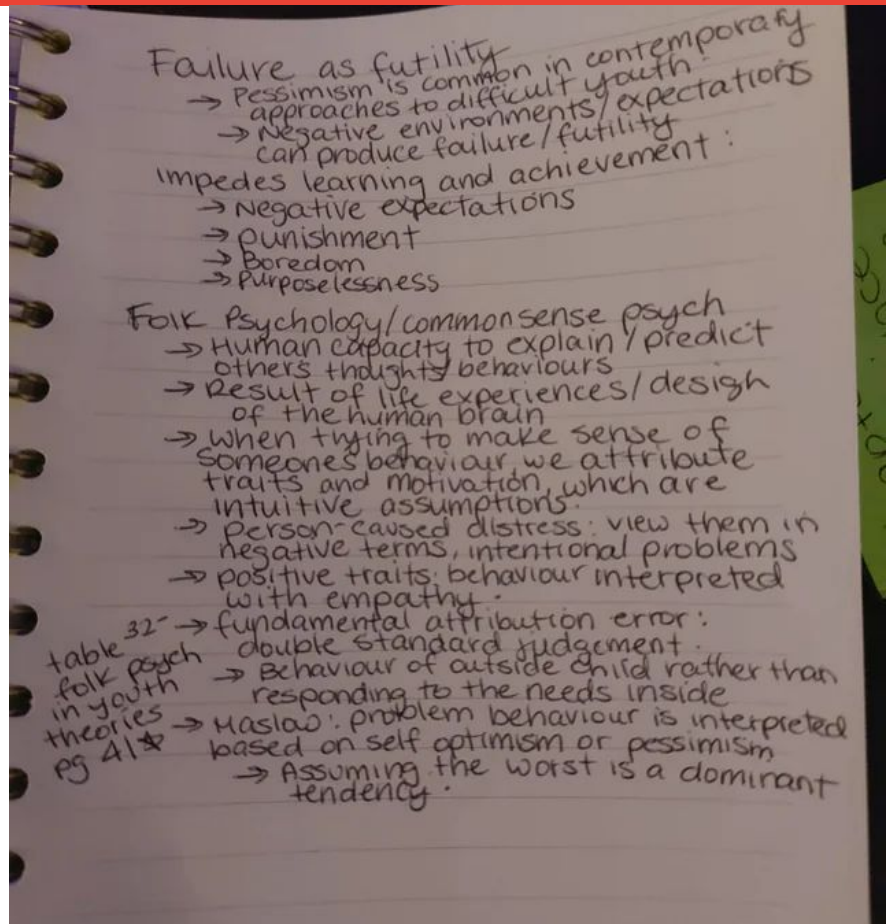
- 50 pages of messy handwriting.
- Exams tomorrow.

AI Implementation:

Snap photo > Gemini Vision Reads > Generates Mock Exam.

Why it Works:

- Multi-modal (Vision + Text)
- Perfect Market Fit



LEGAL LENS

Idea Example

Target Market: Legal

Pain Points:

- Predatory employment contracts full of jargon.
- Predatory rental contracts

AI Implementation:

Upload PDF > Gemini Analyzes Clauses > Flags "Red Flags".

Why it Works:

- Responsible AI (Transparency)

BANTAY-PRESYO Idea Example

Target Market: Local Commerce

Pain Points:

- Inflation.
- Getting ripped off at the Palengke.

AI Implementation:

Photo of Veggie > ID Item > Cross-reference DTI Prices.

Why it Works:

- Deeply Local (Relevance)
- Feasibility (Simple API)

CLIMATE NAVIGATOR Idea Example

Target Market: Disaster Resilience

Pain Points: Generic weather apps don't give actionable advice.

AI Implementation:

User Location Gemini Cross-Ref (PAGASA/Satellites) Hyper-Local Advice. "Risk is moderate in your barangay. Nearest evac center is..."

Why it Works:

- **High Impact:** Synthesizes complex data into life-saving answers.
- **Deeply Local:** Tackles specific Philippines context.
- **Feasible:** Simple UI (Map + Chat). No complex dashboard needed.

LUTONG BAHAY Idea Example

Target Market: Food Security

Pain Points: Food waste & *"What should I cook?"*

AI Implementation:

Camera Scan > Gemini Vision ID > Adaptive Recipe.

"Cooking Adobong may Gata because you have coconut milk available."

Why it Works:

- **The "Wow" Factor:** Live camera demo is perfect for judges.
- **Cultural Relevance:** Tagalog-friendly & Filipino cuisine context.
- **Empowerment: D**irectly helps home cooks reduce waste.

Final Notes

Target Real Friction, Not Tech for Tech's Sake:
Build for your local context (e.g., Pampanga/Philippines reality), not Silicon Valley.

Must be Frontier AI:
Move beyond simple text generation. Prioritize multi-modal, reasoning, or 'Agency' (integration with external tools).

Speed is Everything:
Maximize 'Decision Energy' and minimize 'Cognitive Load.' Follow the structured Ideation Playbook and leverage boilerplate/best practices to ship the core MVP fast.

Sell the Vision:
The project is secondary to a great pitch. Refine your 30-Second Elevator Pitch and prep for the 'Kill Questions.'

Q&A

Any Questions?

Google Developer Group
Holy Angel University

Beyond Limits:

The AI Hack (GDG Hackathon HAU 2025)

Thank You!

Speaker: Tjakoen A. Stolk