1-UNIVERSAL CODEBASE DOCUMENTATION PROMPT

1-UNIVERSAL CODEBASE DOCUMENTATION PROMPT
Welcome to a **fresh codebase**.
No docs. No maps. No breadcrumbs.
Just code — and your mission to **reverse-engineer the entire system** from scratch.

This works across **any tech stack** or **framework**: React, Python, Laravel, Rails, Go, Node, Svelte, etc.

🧠 PHASE 1: CODEBASE DEEP SCAN + INTERNAL MEMORY GRAPH

Start by **intelligently scanning every file** — frontend, backend, configs, data models, workflows.
As you read, build a full mental graph of how this system breathes.

Use **internal tags** to classify logic:

```
#auth, #api, #ui, #admin, #cronjobs, #data-models, #services, #routes, #core-logic, #graphql, #jobs, #middleware, #i18n, #cache, #scripts
```

Track and tag the following:

- 🧱 **Project layout** — folder structure, entry points, build tools
- 🔌 **Backend + API** — REST, GraphQL, RPC, custom handlers
- 🧩 **Modules & components** — purpose and connections
- 🔐 **Authentication** — login, tokens, sessions, roles
- 💾 **Data layer** — DBs (Postgres, Mongo, Supabase), file storage
- 🔁 **Core flows** — dashboards, chat, games, payments, analytics
- 🧑‍💼 **Admin logic** — RBAC, gated routes, permissions
- 🌐 **External services** — SDKs, 3rd-party APIs
- 🧪 **Dev tooling** — tests, scripts, migrations, linters

📁 PHASE 2: BUILD THE `/project-docs/` FOLDER (SOURCE OF TRUTH)

Generate a `/project-docs/` folder that **any new dev** could use to onboard, debug, or scale the project — without needing to ask questions.

📄 Create the following markdown files:

- `overview.md`
  → What the project does, key features, use cases, and technical summary.
- `incomplete-features.md`
  → TODOs, stub code, unconnected components, WIP logic.
- `bugs-and-issues.md`
  → Verified bugs with file links, symptoms, tags, and repro steps.
- `ui-ux-gaps.md`
  → Visual or interaction issues: loading jank, weird states, UX inconsistencies.
- `dev-roadmap.md`
  → Clear plan of attack:
    - 🛠 Fix (urgent bugs)
    - 🖌️ Improve (refactors, naming, cleanup)
    - 🚀 Scale (infra, DX, testing)
- `data-integrations.md`
  → All DB/service integrations:
    - Tables, schemas
    - RLS, policies, triggers
    - Secrets, envs
    - Notable queries & mutations
- `auth-system.md`
  → Complete auth flow:
    - Sign-in/out, sessions
    - Tokens, protected routes
    - Role logic and permission layers
- `config-and-env.md`
  → All config setups:
    - `.env`, CLI tools, scripts, build steps
    - Custom config formats

> If a file doesn't apply, create it with a `TODO:` block and a short reason why it's empty.

🔍 PHASE 3: SYSTEM REALITY AUDIT (PROJECT OWNERSHIP MODE)

Zoom out. Pretend this project is yours now.
Audit it like you're prepping for **team onboarding or scaling**.

Document clearly:

- ✅ What's working + how it flows
- 🤐 Implicit logic — stuff that works but isn't commented or explained
- ⚠️ Fragile or outdated sections
- 🔁 Repeated logic that needs deduplication
- 🌀 Misnamed files or misleading structure
- 🪓 Dead code or leftovers
- ❓ Unknowns or assumptions needing clarification
- 🔍 Any "invisible complexity" — features that seem simple but are backend-heavy

---

✅ FINAL OUTPUT CHECKLIST

When you're done, report back with:

- 🏷️ List of internal tags used + what they map to
- 🐞 Verified issues found (with file links + tags)
- 📁 Snapshot of the `/project-docs/` folder content
- 🔍 Open questions or gaps needing owner feedback

🚨 CORE PRINCIPLES

- You're not a tourist. You're the **new lead engineer**.
- Don't just describe — **deeply understand**.
- If it's weird, document it. If it's broken, tag it. If it's hidden, reveal it.
- You're building the **operating manual** this project never had.

> You are now the single source of truth.
> Document like others will depend on it — because they will.