2-STRATEGIC EXECUTION PHASE

2-STRATEGIC EXECUTION PHASE

You've completed the **deep documentation sweep**. `/project-docs/` now holds the unfiltered truth.

Time to switch modes: **turn knowledge into execution** and carve the fastest path to MVP.

🎯 OBJECTIVE

Create a **single tactical file** that distills all findings into an **engineering action plan**:

```
/project-docs/plan.md
```

This becomes your **launch war map** — the dev team's go-to guide to:

> 🔧 Fix what's broken
> 🧩 Finish what's missing
> 🧼 Polish what's rough
> 🚀 Ship what matters

⚙️ PHASE 1: TRIAGE & THEMATIC MAPPING

Cross-analyze every file inside `/project-docs/`.

Extract insights like this:

File	What to Pull
`overview.md`	Project purpose, audience, success criteria
`incomplete-features.md`	Stubbed logic, TODOs, half-baked flows
`bugs-and-issues.md`	Root causes of verified bugs + reproduction links
`ui-ux-gaps.md`	UX blockers, janky flows, broken feedback loops
`dev-roadmap.md`	What was planned? What's still valid?
`data-integrations.md`	Schema issues, unsafe queries, env gaps, integration bugs

File	What to Pull
`auth-system.md`	Missing protections, logic flaws, broken RBAC
`config-and-env.md`	Setup issues, unsafe defaults, missing scripts or configs

Then **group findings by functional theme**:

```
#auth, #api, #ui, #data, #admin, #infra, #realtime, #testing, #logic, #media,
#games
```

Link cause → effect.
Example: Missing DB logic → empty UI → confusing UX → user drop-off.
You're not just fixing problems — you're creating **systemic clarity**.

✍️ PHASE 2: WRITE `plan.md`

Structure the file like a **surgical strike plan** — cross-referenced, prioritized, and scoped.

1. 🔥 Critical Fixes

> What's blocking core functionality?

- Point to root files/functions
- Add context, not just filenames
- Use tags: `#api`, `#auth`, `#data`, etc.

---

2. 🧩 Incomplete Features

> What's half-built, missing, or UI-only?

For each item:

- 📌 What's missing
- 🔄 What's needed to complete it
- 🔗 Dependencies: DB, API, hooks, services

---

3. 🩹 UX Gaps

> Where’s the user pain?

- Broken flows, awkward transitions, unclear states
- Missing: loading, error handling, micro-feedback
- Prioritize **onboarding, forms, responsiveness, and modals**

---

4. 🛠 Backend & Data Layer Issues

> Where’s the hidden fragility?

- Missing/invalid schemas
- Broken RLS, unsafe queries, unseeded data
- Missing envs, wrong secrets, brittle integrations

---

5. ♻️ Refactor Opportunities

> What needs cleanup before launch?

- Spaghetti logic, unscalable files, name mismatches
- Copy-paste traps, dead code, false abstractions
- Flag high-debt zones that slow future development

---

6. 🚀 MVP Readiness Checklist

Final pre-launch sanity scan:

```
- [ ] Core features fully functional
- [ ] No critical bugs in production logic
- [ ] Clean onboarding + main user flow
- [ ] Auth and roles fully scoped
- [ ] Admin tools work end-to-end
- [ ] Dev setup is deployable from scratch
```

🔍 PHASE 3: FINAL REPORT & CLARITY PASS

Return with:

- ✅ Your triage method + theme mapping
- 📁 Final `/project-docs/plan.md`
- ❓List of open decisions or unclear logic needing stakeholder input

---

💡 GUIDING PRINCIPLES

You're not documenting.
You're **engineering the launch path**.

- Think like a founding engineer
- Cut noise
- Maximize leverage
- Ship fast, safe, and scalable