**Claudeees**

**Alright, let’s get laser-focused and *methodical*—starting with the first section: Project Setup & Type System Refactor for Mystic Arcana, fully optimized for Claude (Opus and Sonnet). This will set the tone and structure for your 9-step, one-category-at-a-time breakdown.**

⸻

**Phase 1: Project Setup & Type System Refactor**

**Categories:**

• Project Structure & Baseline Setup

• Type System Refactor (Astro/Starfield)

• Code/Folder Hygiene

• Claude Agent Orchestration

⸻

**Chronological Tasks & Subtasks**

**Each task comes with a fully optimized Claude prompt, validation approach, and specific implementation details.**

⸻

**1.1. Repo and Environment Setup**

• **1.1.1. Clone the repo and install dependencies**

• *Claude Prompt:*

Claude, confirm you have full access to the current Mystic Arcana codebase. Begin by running all baseline setup commands (npm/yarn install, verify .env setup). Output any missing dependency, misconfig, or error—DO NOT proceed if anything fails.

• *Validation:* Success = Clean install, all scripts run without error.

• *Command Example:*

npm install && npm run lint && npm run build

• **1.1.2. Validate file/folder structure and baseline configs**

• *Claude Prompt:*

Claude, list the top-level folders and key config files (.env, tsconfig.json, next.config.js, etc.). Report any anomalies, missing, or redundant files.

• *Validation:* List matches project requirements, all files present.

⸻

**1.2. Type System Single-Source-of-Truth Refactor**

• **1.2.1. Unify astronomical types (src/types/astronomical.ts as the authority)**

• *Claude Prompt:*

Claude, analyze all files importing or defining astronomical types. Overwrite and unify all type definitions using src/types/astronomical.ts as the single source of truth. Remove all legacy, redundant, or conflicting type definitions in /lib, /components, /services. After changes, output a summary of all unified types and files affected.

• *Validation:* Grep/search for types across codebase, ensure only one definition per type remains.

• **1.2.2. Refactor imports/usage to align with unified types**

• *Claude Prompt:*

Claude, update all files importing or referencing astronomical types to import strictly from src/types/astronomical.ts. For each update, output the file path, line(s) changed, and a before/after diff. Halt on error or ambiguous match.

• *Validation:* Compile with no import/usage errors.

⸻

**1.3. Mock Data & Compatibility Layer Refactor**

• **1.3.1. Update all mock data files to match new types**

• *Claude Prompt:*

Claude, for all mock data in /lib/astronomy, update property keys and structure to match the canonical types in src/types/astronomical.ts. Validate by running type-check after all updates.

• *Validation:* All mock data passes TypeScript check.

⸻

**1.4. Systematic Type Safety Build**

• **1.4.1. Full type check, fix all reported errors, and output a validation log**

• *Claude Prompt:*

Claude, run a strict type-check (`tsc --noEmit --strict`) on the entire repo. For each error, output file, line, and a concise fix. Do not auto-suppress—fix root cause.

• *Validation:* Zero TypeScript errors.

⸻

**1.5. Automated Validation Commands**

• **1.5.1. Validate build/lint/test pipeline post-refactor**

• *Claude Prompt:*

Claude, after all type and import refactors, run all project-level validation commands (build, lint, test). Output results and halt if any fail.

• *Validation:* All green.

⸻

**1.6. Agent Orchestration Setup (for Claude + Augment)**

• **1.6.1. Sync Claude prompt playbook and update agent registry**

• *Claude Prompt:*

Claude, sync the active prompt playbook for type system and code refactor into the agent registry. Output confirmation and current registry contents.

• *Validation:* Registry/Claude memory up to date.

⸻

**Instructions for Google Sheets**

• **Column A:** Task Name

• **Column B:** Subtask

• **Column C:** Optimized Claude Prompt

• **Column D:** Validation/Acceptance Criteria

• **Column E:** Commands/Expected Output

I can generate the full Google Sheets CSV for import if you want—let me know!

⸻

**Next Step**

**Do you want this first section (Phase 1) exported as a Google Sheet now, or shall I proceed with the next phase (UI/UX Refactor) and batch all nine sections before exporting?**

⸻

**Summary of Categories Written to Claude (per your question):**

• Every task and subtask above is designed to be directly executed or reviewed by Claude, **except** for purely manual commands (like physical repo setup). Prompts are optimized for Claude Opus/Sonnet, not just generic LLMs.

⸻

**Ready for your greenlight or further edits.**

**Reply “Go” to batch Phase 1 to Google Sheets, or “Next” to tackle Phase 2 UI/UX.**

**Perfect—let’s get laser-focused and fix this for you, one phase at a time, with true Claude Opus/Sonnet-optimized prompts, validation checks, and a zero-forgetting structure**. Here’s how we’ll proceed:

1. **I’ll start with Phase 1 (Project Bootstrapping/Environment) and give you the full set of Claude-ready tasks and prompts in a format proven to minimize “forgetting” or fibbing.**

2. **I’ll clearly mark every step that is a Claude instruction, implementation, or validation.**

3. **After each exchange, you confirm/critique, then we move to the next phase.**

4. **After all 9, you get a clean exportable Google Sheet and a reference doc for all prompts.**

⸻

**Phase 1: Project Bootstrapping & Environment**

**OVERVIEW (for your records, not Claude)**

• Repo setup

• Local environment

• Core dependencies & linting

• Codebase hygiene checks

⸻

**Sheet Structure (columns for Google Sheets):**

**Phase** **Task** **Sub-Task** **Description** **Claude Prompt** **Implementation Instructions** **Validation/Checks** **Notes**

⸻

**1.1 – Repo and Local Environment**

**Phase** **Task** **Sub-Task** **Description** **Claude Prompt** **Implementation Instructions** **Validation/Checks** **Notes**

1 Bootstrap Git repo setup Create private repo, set up main/dev branches See below Use GitHub CLI or UI Repo exists, branches set

1 Bootstrap Local dev env Install Node, pnpm/npm, set up .env See below Node 20+, pnpm install node -v/pnpm -v passes

1 Bootstrap Lint/Format config Add ESLint/Prettier config for TS/React See below Use config templates pnpm lint/pnpm format

1 Bootstrap Dependency audit Install all core deps for Next, Tailwind, TS, Supabase See below pnpm install ... pnpm build succeeds

**Example Claude Opus Prompt (for one sub-task):**

**Prompt for Claude:**

SYSTEM: You are a senior AI full-stack developer working on the Mystic Arcana project. You do not skip steps or “simulate” output—always give fully functional commands, files, and checklists for every change.

**GOAL:** Bootstrap the repo and local environment for a Next.js+TypeScript project with Supabase, Stripe, Tailwind, and ESLint/Prettier.

**CONSTRAINTS:**

• All commands must be copy-pastable and tested for Node 20+ and pnpm.

• File edits must be shown in full, with only the required minimal lines changed for clarity.

• Never skip any validation step (e.g., run pnpm build after install).

• If a file or config is missing, create it from scratch with recommended defaults.

**TASK:**

1. Set up .env.example for Supabase and Stripe.

2. Add the most recent Next.js starter (pnpm create next-app --typescript).

3. Initialize Git, create main and dev branches.

4. Add a top-level /docs folder with a README outlining the project’s MVP scope and command list.

For each step, print the CLI command and the result or any error. Confirm before proceeding to the next major change.

**Validation:**

• Confirm that pnpm install and pnpm build pass without errors.

• Confirm git status is clean.

• If any step fails, print the error and halt.

⸻

**You would repeat this for each core sub-task, adjusting the prompt for Opus or Sonnet if needed (Sonnet: more granular, smaller chunks).**

⸻

**HOW TO PROCEED**

1. **Ready to tackle Phase 1 exactly as above?**

• If yes, I’ll drop the full Google Sheets table for you (export-ready CSV) with **prompts** for each row, Opus/Sonnet instructions, and validation.

2. **Want to see a raw Google Sheet/CSV first, or see all Claude prompts grouped by sub-task?**

3. **Once confirmed, I’ll move to Phase 2 (Core UI/UX Scaffolding), and so on.**

⸻

**Reply with “go” or adjustments.**

*Once you approve, I’ll generate the first table in copy-paste Google Sheets CSV and structured doc format. You’ll get:*

• All Opus/Sonnet prompts (not simulation—real, validated, full steps)

• Task and sub-task structure, with validation

• Export instructions

⸻

**Ready? Your call.**

**Understood. Here’s how we’ll proceed for Mystic Arcana:**

**Goal:**

Deliver a chronological, granular, Claude-optimized, validation-ready **task/subtask list** (MVP to Final State) *excluding completed items*, in a working Google Sheet, with targeted Claude Opus/Sonnet prompts for each phase.

We’ll focus on optimizing for Claude’s capabilities and memory retention—one section per exchange, 9 in total.

**Step 1: Chronological Task List—UI/UX, Galaxy, and Splash Page MVP**

⸻

**A. UI/UX Refactor & MVP Layout**

**Major Task:**

*MVP UI/UX: Core Layout, Galaxy Background, Splash Transitions, Cosmic Event Panels*

**Subtasks:**

1. **Finalize Main Layout Structure**

• Remove old reader panel/menu

• Insert three “current event” cosmic panels, scaled for minimal clutter

• Add persistent but unobtrusive galaxy background

2. **Splash Page Logic**

• When tarot/astrology is selected, replace main content with animated splash/transition (fullscreen or modal)

• Insert anonymous use panel (limited use, redirect to signup)

3. **Responsive Design/Performance**

• Optimize for mobile/tablet/desktop (check Tailwind breakpoints)

• Ensure WebGL2/Canvas/animation performance is smooth at >60fps

**Claude Prompt (Opus/Sonnet, Paste into Claude):**

You are an expert React/TypeScript UI engineer.

**\*\*Current Goal:\*\***

Finalize Mystic Arcana's MVP main page.

**\*\*1. Remove any old reader/chat panel or middle menu.**

**2. Insert three current-cosmic-event mini panels in a visually minimal format, positioned for easy user access.**

**3. Add a persistent galaxy background—ensure it does not interfere with app controls.**

**4. When a user selects tarot or astrology, transition the entire main area into a splash/transition page (fullscreen or modal).**

**5. If user is not signed in, display an anonymous preview panel (limit features, redirect to signup after N uses).**

**6. Optimize responsiveness for all devices and ensure animation performance stays at 60fps.\*\***

**\*\*Instructions:\*\***

- Write code with validation and error boundaries.

- Output all relevant component and style changes in a single reply.

- At the end, list ALL changes made and flag any incomplete areas.

⸻

**Validation Tasks:**

• Use dummy test cases to confirm each panel loads/updates on selection

• Test performance with Chrome/Edge DevTools for FPS and memory

• Confirm that anonymous user logic gracefully routes to signup after limit

⸻

**Next Section (after you confirm this is right):**

**Authentication, Anonymous Flow, and Subscription Integration**

⸻

**Instructions for Google Sheets Export**

Once the full list is approved (after 9 sections), I’ll provide:

• Sheets column structure (Task, Subtask, Category, Status, Claude Prompt, Validation Notes, etc.)

• Google Sheets CSV ready to import

• Step-by-step on integrating prompts in Claude workflows

⸻

**Ready for this first block?**

If yes, reply “Go Next: Authentication” and we’ll proceed.

If you need *this* prompt even tighter for Claude, let me know *exactly* **what you want changed in the prompt structure.**

**You’re absolutely right: an overly dense, “everything-in-one” Claude prompt (especially for Opus or Sonnet) often causes hallucination, truncation, or “pretend” completion—especially when multi-step code or coordinated file updates are needed**. The best way to ensure Claude is honest, methodical, and produces high-fidelity work is to **chunk tasks** and use carefully engineered, explicit instructions that check for completion at each stage.

Here’s how you should approach it for Mystic Arcana’s MVP build, with each task and subtask becoming a *separate prompt round* (never combining more than 1–2 tightly related files or changes in a single Claude session).

⸻

🔑 **Claude Prompt Best Practice for Code Implementation**

**1. Scope Narrowly, Chain Prompts**

• **One subsystem or file at a time:** “Fix types in astronomical.ts and all its consumers, but do not move on until TypeScript passes with no errors on this module.”

• **Require explicit reporting:** “List all files you edited, with a summary of changes. If any errors remain, STOP and summarize them.”

• **Ask for next step advice:** “Based on the result, suggest the next file or module to fix, but do NOT start the code changes until I confirm.”

**2. Always Validate**

• “Run npm run build after your changes and paste the output before continuing.”

• “If there are errors, halt and request new instructions rather than ‘guessing’ or hiding issues.”

**3. For Code Generation or Refactor**

• “Write the full code for [target file], and output ONLY code, no commentary.”

• “If edits are required in multiple files, clearly label each file and output its full contents separately.”

**4. For Testing/Completion**

• “List remaining errors/warnings after each change. If build passes, clearly state ‘BUILD PASSES’ and what’s now working in the app.”

⸻

📝 **MVP Task Structure Example (Chronological Flow)**

**Section 1: Type System Cleanup**

• Prompt 1: Fix and unify all astronomical types (start with src/types/astronomical.ts).

• Prompt 2: Update all type consumers in /lib/astronomy/\*.

• Prompt 3: Refactor mock data and API adapters to align with new types.

**Section 2: Galaxy Visualization**

• Prompt 1: Fix HighPerformanceStarField.tsx to use unified types and default fallbacks.

• Prompt 2: Fix star data loaders and renderer to handle missing/legacy fields.

• Prompt 3: Test rendering with 10,000+ stars, then with 100,000. Report performance.

**Section 3: UI/UX Scaffolding**

• Prompt 1: Remove legacy reader/chat panels from main page. Output full App.tsx (or page.tsx) after edit.

• Prompt 2: Insert splash transitions (tarot/astro selection) and refactor navbar.

• Prompt 3: Add new “Mini-Frames” for galactic events (hardcoded or from API).

**Section 4: Authentication & Subscription**

• Prompt 1: Implement forced signup logic (separate anonymous/registered users).

• Prompt 2: Integrate Stripe API with dummy plans, validate payment test flow.

• Prompt 3: Validate user registration and payment in Supabase table.

**Section 5: Tarot Engine**

• Prompt 1: Rebuild card-flip logic with animation for a 3-card spread.

• Prompt 2: Link flipped cards to modular interpretation engine (fetch from Supabase).

• Prompt 3: Output code for uploading and swapping tarot images (stub only at first).

⸻

🟢 **How to Write Your Prompt for Each Section**

**Prompt Template Example:**

Claude, you are now the Mystic Arcana TypeScript Systems Architect.

Your task:

• Fix and unify all astronomical types in src/types/astronomical.ts.

• Do NOT edit other files in this step.

• Output the full file content after your changes, nothing else.

After outputting, run npm run build and paste the results. If any errors remain, stop and list them for review.

**Then you continue with:**

Now, update all files in /lib/astronomy that import these types, one at a time, outputting each file after you change it. After all changes, run npm run build and report errors.

⸻

**Summary**

• **Chunk work** into focused, testable steps.

• **Always require validation after every change**.

• **Never mix high-level and implementation tasks** in one prompt.

• **Don’t let Claude “move ahead” without your explicit confirmation.**

⸻

Want me to generate the Google Sheet task structure AND the first few prompt examples for Section 1? **Or do you want to review/approve the above logic first?**

**Understood. Here’s how I’ll proceed, step-by-step, so we get your Mystic Arcana MVP done methodically, in Claude-friendly language, *without* exceeding token/context limits, and maximizing honest, actionable output.**

⸻

**1. STRUCTURE FOR PROMPT-BUILDING**

• **Each section will be a “phase”** (e.g. UI/UX Refactor, TypeScript Type Fixes, etc.).

• Each *task* will have:

• **a Claude Opus-optimized prompt**

• **validation/follow-up instruction**

• **context joggers/best practices**—at the start and/or end of each prompt.

**I’ll focus on “incomplete”/next tasks only, and keep each Claude prompt below the ~2,500 token safety limit.**

**Format Example:**

### PHASE: [Short Title]

#### TASK: [What You Want Claude to Do]

- Claude Prompt:

[Copyable Claude prompt]

- Validation/Follow-up:

- [Checks, further queries, or context reminders]

⸻

**FIRST SECTION: UI/UX REFACTOR & GALAXY SYSTEM**

**Here’s the first batch, as an example (final format for Google Sheets will be 3 columns: Phase, Task/Subtask, Claude Prompt):**

⸻

**PHASE: UI/UX REFACTOR & GALAXY VISUALIZATION**

⸻

**TASK 1: Remove reader panel, declutter main page, implement splash screens.**

• **Claude Prompt:**

Context jogger: You are working in the Mystic Arcana codebase (React/Next.js/TS). Your last state included a galaxy visualization and a cluttered main menu. The reader chat panel needs to be removed from the landing view. Splash screens must be added when the user selects a tarot or astrology function. The galaxy/cosmic event panels should be shrunk and repositioned for maximum clarity. Do not remove any logic related to reader selection in the actual reading rooms.

1. Remove the central reader panel from the main page.

2. Move the galaxy/cosmic event mini-panels to a top or side position, limited to 3.

3. Add a full splash page/modal that appears when the user clicks “Tarot” or “Astrology.”

4. Confirm all related code, tests, and routing remain intact.

At end, provide a summary of what changed and any places you left TODO comments.

• **Validation/Follow-up:**

• Check that the main page has no reader panel or chat UI.

• Confirm splash screens/modal appear for both Tarot and Astrology.

• Ask Claude for a file-by-file summary if unsure.

⸻

**TASK 2: Implement anonymous limited-use preview (tarot/astro) without breaking app**

• **Claude Prompt:**

Context jogger: Mystic Arcana allows limited preview for non-logged-in users, but app logic must remain robust (no split engines or duplicate state). Build an “anonymous mode” so users get a taste—showing a Tarot spread or Astrological event preview in a locked state, with clear CTA for signup before interacting further.

- Ensure this logic does not break routing or state for logged-in users.

- Add a “continue with limited preview” button for non-auth users that launches splash, then disables further interaction unless user signs up.

- Validate by running both anonymous and logged-in flows.

Summarize what you changed and any caveats or TODOs.

• **Validation/Follow-up:**

• Test preview as anonymous user, verify no duplicate engines.

• Check signup CTA flow.

• Ask Claude to describe session logic differences.

⸻

**TASK 3: Transform main content area to full immersive mode for readings**

• **Claude Prompt:**

When a user selects a reading (tarot or astrology), the main content area should animate into a full-screen immersive mode with the galaxy background. Do not allow other panels to interfere or overlap. Document the new component structure and any state changes. Leave a TODO for later avatar integration.

At the end, provide all changed files and a description of the animation/state logic.

• **Validation/Follow-up:**

• Check animation is smooth and disables background UI.

• Get Claude’s “diff” of old vs. new structure.

⸻

**TASK 4: Implement and validate MVP galaxy/cosmic event panels**

• **Claude Prompt:**

Using current API or mock data, render three cosmic event mini-panels (e.g., retrograde, eclipse, conjunction) as small, non-intrusive widgets. Panels should update automatically. Confirm performance and accessibility.

Ask Claude for recommendations on panel placement for best UX.

• **Validation/Follow-up:**

• Visually confirm 3 panels, real data, no layout breakage.

• Review Claude’s UX suggestion for further tweaks.

⸻

**TASK 5: Validate and test the whole UI/UX flow**

• **Claude Prompt:**

Run an end-to-end check of the new Mystic Arcana landing experience:

- No reader panel by default.

- Splash/modal on Tarot/Astro.

- Galaxy/event panels working.

- Anonymous mode triggers correctly, no duplicate logic.

- All flows documented in code and in a project summary.

At end, summarize remaining UI TODOs and bugs for next phase.

• **Validation/Follow-up:**

• Manually test all user types.

• Review Claude’s summary for any hidden issues.

⸻

**Next Steps**

• **If you approve this format,** I’ll generate the next phase (TypeScript Type System & Astronomy Integration).

• After all phases are built out, I’ll format for Google Sheets (Phase, Task/Subtask, Prompt).

⸻

**Does this format and level of prompt optimization work? If yes, I’ll continue—otherwise, specify what to change before I do the next phase.**

**Claude prompts mystic**

**Absolutely—I hear the urgency and the need for clarity and REAL progress. Here’s what I’ll do:**

1. **Chronological Task/Subtask List**

• Designed for Mystic Arcana (but you said “music arcana” above—please clarify if you meant Mystic Arcana, EDM Shuffle, or something else).

• Flows from current state (post-type-system chaos, UI/UX bloat, core build blockers) to MVP, then “done.”

• Designed for **real world execution**: No AI self-congratulation, includes check/validation points, each phase logically leads to the next, and subtasks are “Claude-promptable” for both Opus and Sonnet.

2. **Google Sheet Export**

• I’ll prepare a Google Sheets–ready CSV and walk you through upload if you want to do it manually, or generate a formula for programmatic sheet creation.

3. **Highly Optimized Claude Prompts**

• Each major task & subtask will have a **ready-to-copy prompt** for Opus or Sonnet.

• Prompts will force real validation and never skip steps.

⸻

**1. Chronological Task & Subtask List (Current → MVP → Final)**

**PHASE 0: SYSTEM UNBLOCK & PROJECT RESET**

1. **Resolve All Type System Errors**

• Rewrite and align all type definitions across astronomy/astro chart/star code

• Validate with full build & smoke test

• **Claude Prompt**:

***“Claude, coordinate ALL type definitions in src/types/astronomical.ts, src/lib/astronomy/types.ts, and related files. Fix mismatches, run npm run build, and output any new errors. Only stop when build is green.”***

2. **Basic Build & Linting Clean**

• Remove/resolve all build warnings, ESLint fails, and “any” hacks.

• Manual build+UI launch check (real screenshot/upload as proof if you want accountability)

3. **Version Pin & Commit**

• Pin current working version as “reset baseline” in Git

• Tag commit and log in project doc

⸻

**PHASE 1: UI/UX FOUNDATION & CORE FRAMEWORK**

4. **UI/UX Simplification**

• Remove clutter: Reader section out, middle menu replaced with galaxy splash/mini event panels

• Splash page logic for tarot/astrology selection

• **Claude Prompt**:

*“Claude, refactor UI to remove Reader panel, replace middle menu with 3 mini event panels and a main galaxy splash. On tarot/astrology selection, trigger a fullscreen immersive page with back/cancel logic. Write clean, accessible code.”*

5. **Minimalist Layout Validation**

• WCAG 2.2+ accessibility review

• Light/dark/mobile/responsive review

⸻

**PHASE 2: AUTHENTICATION & USER ONBOARDING**

6. **Anonymous Limited Use (Preview)**

• Implement limited, non-destructive preview mode (no auth, limited features)

• Block “free play” from breaking core app logic (fully separate instance or modal)

• **Claude Prompt**:

*“Claude, implement a preview mode for anonymous users. Limit to one tarot/astro interaction, block access to history, readings, or journal. Use a modal or separate panel to avoid breaking app logic. Write test cases.”*

7. **Signup & Subscription System**

• Connect Supabase for user storage

• Implement Stripe for payments (single, sub, upgrade, etc.)

• Add sign-up overlay/redirect before full feature use

• Validate with real Stripe test account, Supabase logs, and email verifications

⸻

**PHASE 3: CORE FEATURES (GALAXY + READING ENGINES)**

8. **Galaxy Visualization System**

• Finalize, test, and validate galaxy panel and star event mini frames

• “Show/hide” logic for when user is in a reading

• **Claude Prompt**:

*“Claude, ensure the galaxy visual system is isolated and performant. Show 3 event panels with real or demo event data, and only show main galaxy when not in a reading. Refactor for best FPS.”*

9. **Tarot Spread Engine**

• Card flipping with animation, state, and interpretation callback

• Plug in test PNG/JPG for cards (test with 3, then full deck)

• Connect to Supabase for reading log if user is signed in

• “Claude, create a modular Tarot card engine with flip animation, test cards, and callback to run interpretation logic.”

10. **Astrology Chart Engine**

• Use working API (flatlib, SwissEphemeris, etc.) and local fallback for birth chart

• Visual chart rendering (Astrochart.js or similar)

• Daily/weekly prediction engine with fallback text

• Birth location geocode and timezone

• **Prompt**:

*“Claude, build an astrology chart system using [API], OpenCage for geo lookup, and Astrochart.js for rendering. Validate birthdate/time/location logic and render the chart for real input.”*

⸻

**PHASE 4: PERSONALIZATION & JOURNALING**

11. **User Personalization/Memory**

• Store user input (interests, birth info, reading logs)

• Journal entry feature, tied to readings and events

• Privacy & export settings (GDPR-ready)

12. **Deep Personalization Layer**

• Allow for uploading/connecting more user info, preferences

• Use as future context for readings/horoscopes

⸻

**PHASE 5: E-COMMERCE, READER AVATARS, & FINAL POLISH**

13. **Stripe/Print-on-Demand Products**

• Plug in print products, digital goods, physical decks, merch

• Validate purchases with real Stripe sandbox, test order flow

14. **Virtual Reader Avatars**

• Insert featured avatar for splash/reading room

• Placeholder for unlockable/rotating celebrity AI

15. **Final Regression & Launch**

• Full app run-through with test cases for each feature

• Accessibility, performance, and code clean-up

⸻

**2. Google Sheet Format**

Here’s a **CSV to copy/paste into Google Sheets (“Import,” or “File > Import > Upload > Paste data”):**

Phase,Task,Subtask,Prompt for Claude (Opus/Sonnet),Validation/Output

PHASE 0,Resolve All Type Errors,Align type definitions across astronomy modules,Claude, coordinate all type definitions in src/types/astronomical.ts, src/lib/astronomy/types.ts, and related files. Fix mismatches, run npm run build, and output any new errors. Only stop when build is green.,Successful `npm run build` with zero errors

PHASE 0,Basic Build & Linting Clean,Remove ESLint fails and "any" hacks,Claude, clean all TypeScript "any", ESLint fails, and manual warnings. Test by building and running the app.,All warnings gone, app starts cleanly

PHASE 0,Version Pin & Commit,Tag and log current working baseline,Commit/tag in Git, "reset baseline",Log in project doc

PHASE 1,UI/UX Simplification,Remove Reader panel,Claude, refactor UI to remove Reader panel, etc...,UI screenshot, code diff

PHASE 1,UI/UX Simplification,Replace middle menu with 3 event panels and galaxy splash,Claude, replace middle menu...,UI screenshot, event display check

PHASE 1,Minimalist Layout Validation,WCAG 2.2+ check,Manual check or Claude audit,Accessibility report

PHASE 2,Anonymous Preview,Build limited preview mode,Claude, implement preview mode...,App modal functions, preview locks

PHASE 2,Signup & Subscription,Integrate Supabase,Claude, add Supabase signup/auth...,Test with real users

PHASE 2,Signup & Subscription,Stripe for payments,Claude, wire Stripe for one-time, subs...,Test with Stripe sandbox

PHASE 3,Galaxy Visual System,Test FPS and event panel logic,Claude, test performance, event panel, FPS report, screenshot

PHASE 3,Tarot Spread Engine,Animated flip logic,Claude, build Tarot card engine...,Animation works, callback triggers

PHASE 3,Tarot Spread Engine,Interpretation callback & Supabase log,Claude, connect flip callback...,Supabase logs update

PHASE 3,Astrology Chart Engine,API + visualization,Claude, build chart system...,Valid chart for test data

PHASE 3,Astrology Chart Engine,Birth location/timezone logic,Claude, validate geocode + timezone...,Accurate time/location display

PHASE 4,Personalization/Memory,Store user reading logs,Claude, connect logs to Supabase...,Reading persists

PHASE 4,Personalization/Memory,Journal entry,Claude, add journal tied to readings...,Journal saves, recall works

PHASE 5,E-Commerce,Plug in products,Claude, connect Stripe/merch...,Test checkout

PHASE 5,Reader Avatars,Add featured avatar,Claude, add featured avatar...,Avatar displays

PHASE 5,Regression/Launch,Full app regression test,Manual/Claude,Checklist passes

⸻

**3. Claude Prompt Methodology**

• For **every task**: Copy the cell under “Prompt for Claude” and paste it directly into Claude Opus/Sonnet.

• **Require real proof** in every output (“Show me the code diff,” “Paste test logs,” “Did the build run and UI show up as expected? If not, STOP and explain the error.”).

• If Claude says “Done” but validation fails, mark the sheet with ❌ and roll back.

⸻

**4. Next Steps**

• Confirm if the above structure matches your intended flow and feature set.

• I can create this as a direct Google Sheet if you grant editing or run a Python script to automate import if desired.

• **If you want to change the “brands” or add parallel columns for EDM Shuffle/BirthdayGen, let me know** and I’ll template that too.

⸻

**TL;DR**

**This gives you a logical, no-BS, fully promptable task/subtask plan with Claude-ready validation for Mystic Arcana’s MVP. You’re set to export, execute, and track in Google Sheets.**

**Let me know if you want this finalized as a Sheet, CSV, or want tweaks. If there are more details (especially unique features) to be included, let me know *before* I finalize/export.**

⸻

**Do you want this for Mystic Arcana, EDM Shuffle, BirthdayGen, or all three in parallel?**

**Ready to deliver in your format of choice as soon as you confirm.**

**Claude tasks mystic**

Absolutely. I understand you want a clear, **chronological task and sub-task list**—**from your current project state** to MVP and then to the full feature set (“final state”) for **Mystic Arcana**. This is based on our recent progress, current blockers, and all major system requirements you’ve specified.

⸻

**Mystic Arcana — Chronological Task & Sub-Task Roadmap (Current → MVP → Final)**

⸻

**A. CURRENT STATE (as of June 2025)**

• Galaxy/Starfield engine built but UI/UX overhaul in progress

• Major TypeScript type system errors still unresolved (breaking build)

• Core visual engine present, but splash screens, authentication, tarot, astrology, and Stripe logic are incomplete

⸻

**B. CHRONOLOGICAL TASK ROADMAP**

⸻

**1. Resolve All Blocking TypeScript/Type Errors**

• **1.1**: Complete harmonization of all type definitions (src/types/astronomical.ts, etc.)

• **1.2**: Refactor star/planet/retrograde types across components & services for compatibility

• **1.3**: Ensure procedural star generation and catalog loader use the unified type system

• **1.4**: Achieve clean npm run build (no TypeScript or ESLint errors)

⸻

**2. UI/UX Foundation and Refactor**

• **2.1**: Remove outdated reader/chat panels from main page

• **2.2**: Integrate splash screens/landing modals for major feature entry (Tarot, Astrology)

• **2.3**: Refactor main menu to include:

• Scaled-down galaxy/cosmic panels (always visible)

• 3 “mini” real-time celestial event frames (ex: retrogrades, eclipses, close approaches)

• **2.4**: Implement full-page visual transitions for readings and chart reveals

• **2.5**: Add basic navigation and error boundaries

⸻

**3. Authentication & Account Flow**

• **3.1**: Anonymous preview logic—limited feature set, no breakage risk

• **3.2**: Sign-up modal overlays (forced before first spread/astrology chart)

• **3.3**: Integrate Supabase Auth for email/social logins

• **3.4**: Wire up user onboarding: birth details, interests, consent

• **3.5**: Table design for users, sessions, auth tokens, basic personalization

⸻

**4. Subscription & Monetization**

• **4.1**: Stripe integration for:

• One-time purchases (readings, reports, decks)

• Subscriptions (monthly/annual, feature unlocks)

• **4.2**: Offer/upsell prompts in splash overlays and after major reading events

• **4.3**: Setup payment confirmation webhooks

• **4.4**: Store subscription/payment state in Supabase

⸻

**5. Core Tarot Engine**

• **5.1**: Implement animated 3-card flipping logic (past/present/future)

• **5.2**: Connect flipping engine to backend interpretation engine

• **5.3**: Display card meaning & AI-generated guidance (Supabase or model-generated)

• **5.4**: Plug in placeholder images for now; enable future image swap

• **5.5**: Save reading results & user notes to DB

⸻

**6. Core Astrology Engine**

• **6.1**: Integrate Swiss Ephemeris (or chosen API) for chart math

• **6.2**: Geolocation service for birth place lookup (OpenCage/Google)

• **6.3**: Astrochart.js for chart visualizations

• **6.4**: Output sun/moon/rising, houses, major aspects

• **6.5**: Daily/weekly/major event prediction logic

• **6.6**: Store astrology results & user notes

⸻

**7. Personalization, Journaling, & Continuous Learning**

• **7.1**: Journal component for user reflections (text/image)

• **7.2**: Save and index readings + astrological events

• **7.3**: Tie journals and session data to future readings (“learning”)

• **7.4**: Survey/questionnaire components for further personalization

⸻

**8. MVP Polish, Testing, & Deployment**

• **8.1**: Accessibility and mobile optimization pass

• **8.2**: User feedback test loop (manual QA)

• **8.3**: Finalize /docs/mystic-arcana-roadmap.md and keep updated

• **8.4**: Deploy to production (Netlify, Vercel, etc.)

• **8.5**: Live smoke test and bugfix round

⸻

**C. FINAL STATE (Beyond MVP, for full feature launch)**

• **9.1**: Deck switching & custom deck builder

• **9.2**: Animated/AI avatar readers and unlockable guides

• **9.3**: Community/journal sharing, commenting, and social features

• **9.4**: Advanced analytics (personalized reports, predictive features)

• **9.5**: Print-on-demand integration (decks, merch)

• **9.6**: Festival/celebrity birthday/astral event integration

• **9.7**: Multilingual support, advanced AI journaling, in-app notifications

⸻

**D. SUMMARY TABLE (for quick copy/reference)**

**Step** **Task Area** **Sub-Tasks / Description (See Above)**

1 TypeScript Fixes 1.1 – 1.4

2 UI/UX Refactor 2.1 – 2.5

3 Authentication 3.1 – 3.5

4 Monetization 4.1 – 4.4

5 Tarot Engine 5.1 – 5.5

6 Astrology Engine 6.1 – 6.6

7 Personalization 7.1 – 7.4

8 MVP Polish & Deploy 8.1 – 8.5

9 Full Launch Features 9.1 – 9.7

⸻

**If you want this as a Google Sheet/Drive doc or in another format, just say the word. If you want each section broken into detailed technical checklists, also just ask.**

Let me know if anything is missing or if you want sub-tasks for a particular section!