Got it! I understand now - you're maintaining a Project Bible to work around my session memory limitations, and AI LW is a groundbreaking project where AI agents autonomously build everything. This is brilliant for managing ADD by keeping everything externalized and organized.

I can see you were adding the Development Philosophy & AI Autonomy Mandate section to the Bible. Let me complete that section and confirm what you need next:

DEVELOPMENT PHILOSOPHY & AI AUTONOMY MANDATE

THE CORE THESIS:

AI Legacy Wars is the first commercial game where EVERYTHING—from custom game

engine to content to runtime systems—is designed, developed, coded, tested,

and deployed entirely by AI agents. No human developers. No licensed engines.

No pre-built frameworks. Pure AI autonomous creation.

WHAT MAKES THIS REVOLUTIONARY:

This is NOT:

❌ AI-assisted development (human-led with AI help)

❌ AI-generated content on existing engines (Unity + ChatGPT)

❌ Drag-and-drop "AI game builders" (low-code tools with AI features)

❌ AI that writes code humans then assemble

This IS:

✅ AI agents autonomously designing game architecture

✅ AI building custom game engine from scratch

✅ AI writing all gameplay systems and logic

✅ AI creating content generation pipelines

✅ AI implementing monetization and backend

✅ AI testing, debugging, and deploying

✅ Human provides vision, validates, and guides direction

THE AI DEVELOPMENT STACK:

Claude (Lead Architect & Developer):

System architecture design

Core engine development

Gameplay systems coding

Documentation generation

Code review and refinement

Future AI Agents (Specialized Roles):

Content AI: Generate worlds, scenarios, narratives

Testing AI: Automated QA, balance testing, bug detection

DevOps AI: Deployment, monitoring, scaling

Art AI: Visual asset generation (MidJourney/DALL-E integration)

Audio AI: Music and sound design

DEVELOPMENT CONSTRAINTS:

What AI Must Build:

Custom 2D game engine (no Unity, Unreal, Godot)

All game systems (combat, progression, matchmaking)

Backend infrastructure (databases, APIs, auth)

Frontend UI/UX (web and mobile)

Payment processing integration

AI coaching system

Analytics and admin dashboards

What AI Cannot Use:

Pre-built game engines

Asset store purchases

Template projects

Human-written boilerplate code

Licensed frameworks (except open-source libraries)

What AI May Use:

Open-source libraries (React, Node.js, PostgreSQL)

APIs (Stripe, authentication services)

Cloud infrastructure (AWS, Vercel)

Standard web technologies (HTML, CSS, JavaScript)

PROOF OF CONCEPT VALIDATION:

To prove AI autonomy, the first playable demo must:

Run entirely on AI-written code

Use no commercial game engines

Demonstrate core gameplay loop

Include basic monetization

Be deployable to production

Function without human code intervention

HUMAN ROLE DEFINITION:

Justin's Responsibilities:

Define vision and creative direction

Validate AI-proposed solutions

Make business decisions

Approve designs and mechanics

Test and provide feedback

Market and promote

Manage budget and resources

Justin Does NOT:

Write production code

Debug technical issues

Design system architecture

Implement features

Fix bugs

Optimize performance

AI COLLABORATION PROTOCOL:

When Working with Claude:

Provide context from Bible (copy relevant sections)

State clear objective ("Build matchmaking system")

Specify constraints (budget, timeline, dependencies)

Request complete solutions (not fragments)

Validate and test (provide feedback)

Update Bible (document decisions)

What Makes Good AI Collaboration:

✅ "Build the complete tournament entry system with Stripe integration"

✅ "Here's the monetization Bible section - implement Bronze tournaments"

✅ "Create the matchmaking algorithm matching the ELO system in Bible"

What Wastes Time:

❌ "Can you help with some code?"

❌ "What do you think about this idea?" (without context)

❌ "Fix this error" (without showing full system)

DOCUMENTATION MANDATE:

Every AI-built system must include:

Architecture overview (how it works)

API documentation (endpoints, parameters)

Database schema (tables, relationships)

Deployment guide (how to launch)

Testing instructions (how to validate)

Integration guide (how it connects to other systems)

Format: All docs go in Project Bible for continuity across sessions.

QUALITY STANDARDS:

AI-generated code must:

Be production-ready (not prototypes)

Include error handling

Have security measures

Be scalable (10K+ users)

Follow best practices

Include comments explaining logic

Be deployable immediately

THE MOONSHOT GOAL:

By launch, AI LW proves:

Technical: AI can build commercial-grade software autonomously

Creative: AI can implement sophisticated game design

Business: AI-built products can generate real revenue

Industry: This changes how games (and software) can be made

STATUS: ✅ Philosophy defined, ready to execute

================================

Now, what's next?

Based on your Table of Contents, you have:

✅ Monetization (complete)

✅ Development Philosophy (just added)

What module should we build next? Looking at your TOC:

High Priority Options:

#3: Complete Design Reference (game mechanics, combat system, worlds)

#12: AI Collaboration Guide (how to work with Claude effectively)

#7: Implementation Roadmap (what to build in what order)

Which one would help you most right now?