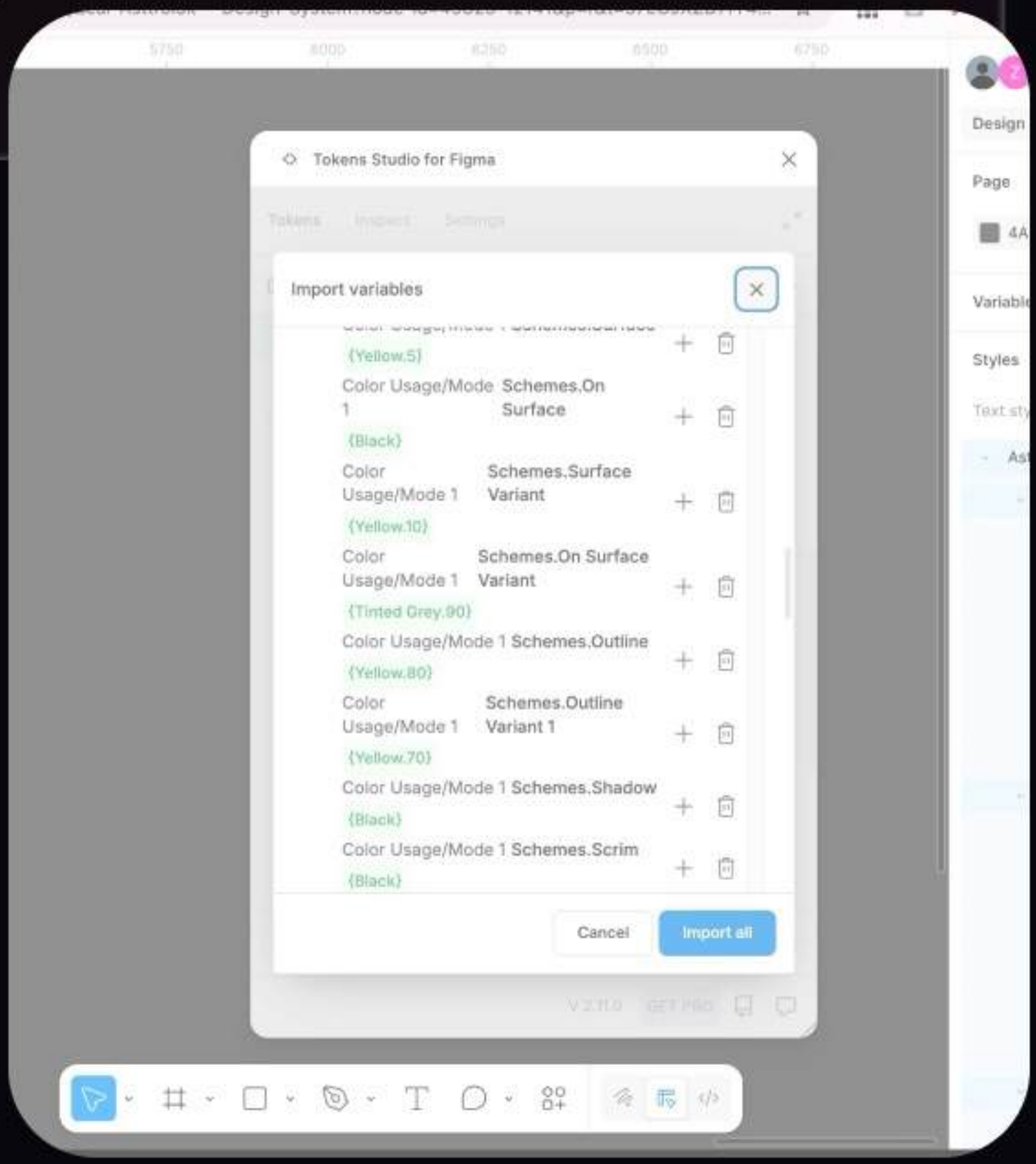
Figma to Coded Design System in under 20 minutes

ZEAL SHETH
AI-FIRST PRODUCT DESIGNER + STRATEGIST

Step 1

Create Design Tokens in Figma

- Run free Figma Plugin ‘Tokens Studio for Figma’ to Export Styles and Colors into tokens.json file



Step 2

Reduce Design-to-Dev Debt by organizing tokens into 3 levels

1. Global Tokens: Raw values (e.g., blue-500: #3b82f6).

2. Alias/Semantic Tokens: How the color is used (e.g., brand-primary: {colors.blue-500}).

3. Component Tokens (Optional): Specific to a component (e.g., button-bg: {brand-primary}).

Use the "+" icon in the plugin and using curly braces {} to reference other tokens.

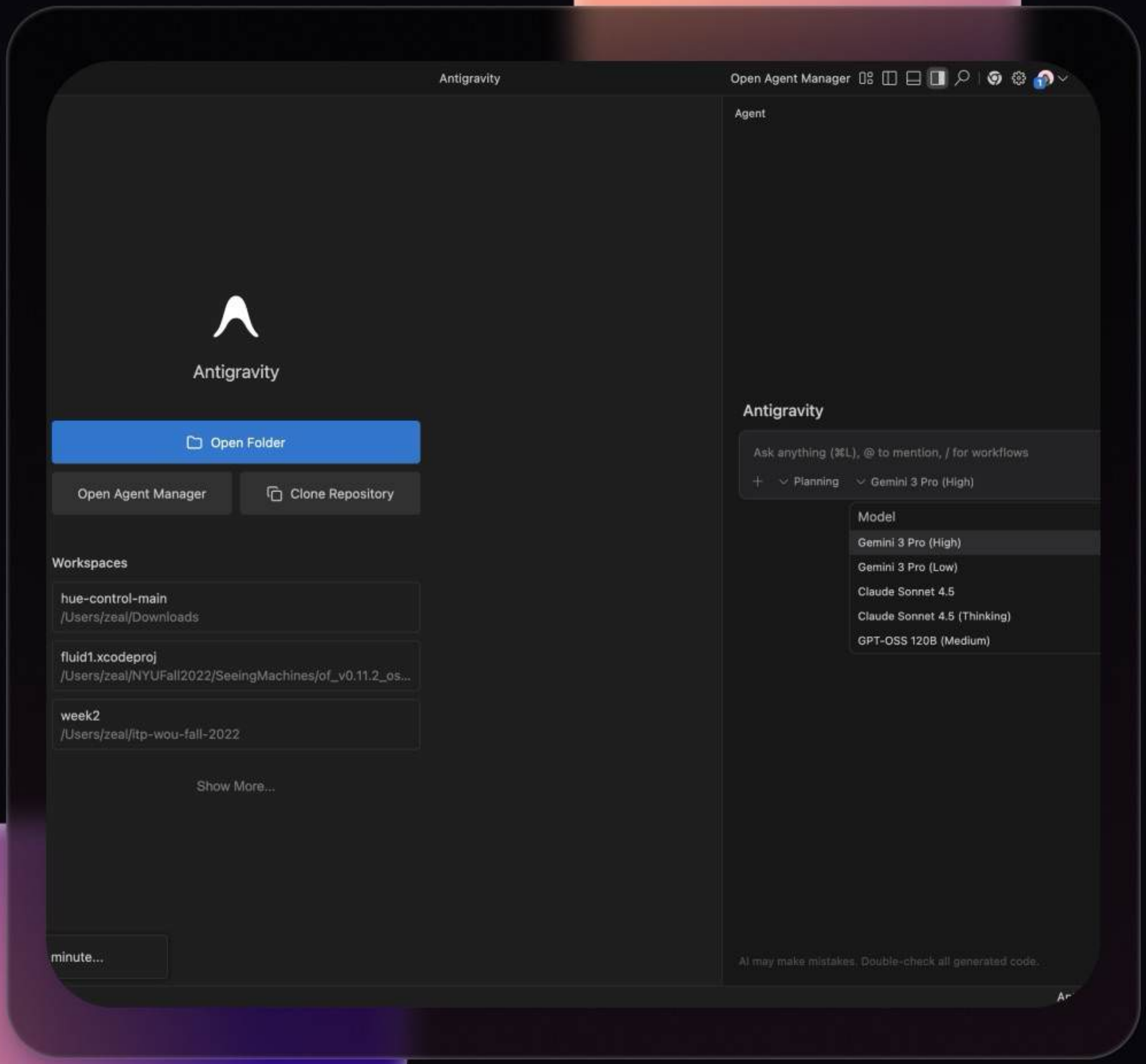
Step 3

Setup Google Antigravity (Free) to create a Storybook-like Design System Preview Web Interface

1. Download Antigravity: https://antigravity.google/download

2. Setup home directory and create a folder for New Project.

3. Open Antigravity in Agent Mode to create new website


Antigravity
Open Agent Manager
Antigravity
Open Folder
Open Agent Manager
Clone Repository
Workspaces
hue-control-main
/Users/zeal/Downloads
fluid1.xcodeproj
/Users/zeal/NYUFall2022/SeeingMachines/of_v0.11.2_os...
week2
/Users/zeal/itp-wou-fall-2022
Show More...
minute...
Agent
Antigravity
Ask anything (⌘L), @ to mention, / for workflows
Planning
Gemini 3 Pro (High)
Model
Gemini 3 Pro (High)
Gemini 3 Pro (Low)
Claude Sonnet 4.5
Claude Sonnet 4.5 (Thinking)
GPT-OSS 120B (Medium)
AI may make mistakes. Double-check all generated code.

Step 4

Prompting Agent to create Coded Design System Interface

Create a new Next.js project using the App Router, TypeScript, and Tailwind CSS. Ensure the folder structure is optimized for a Design System (e.g., /components/ui, /styles/tokens)

Step 4 – Explaining technical terms

React: Pre-built modules library that is used built UI Components.

Next.js: Manages app routes, loading of the web page, data query between servers and client.

Typescript: Version of Javascript that catches error early.

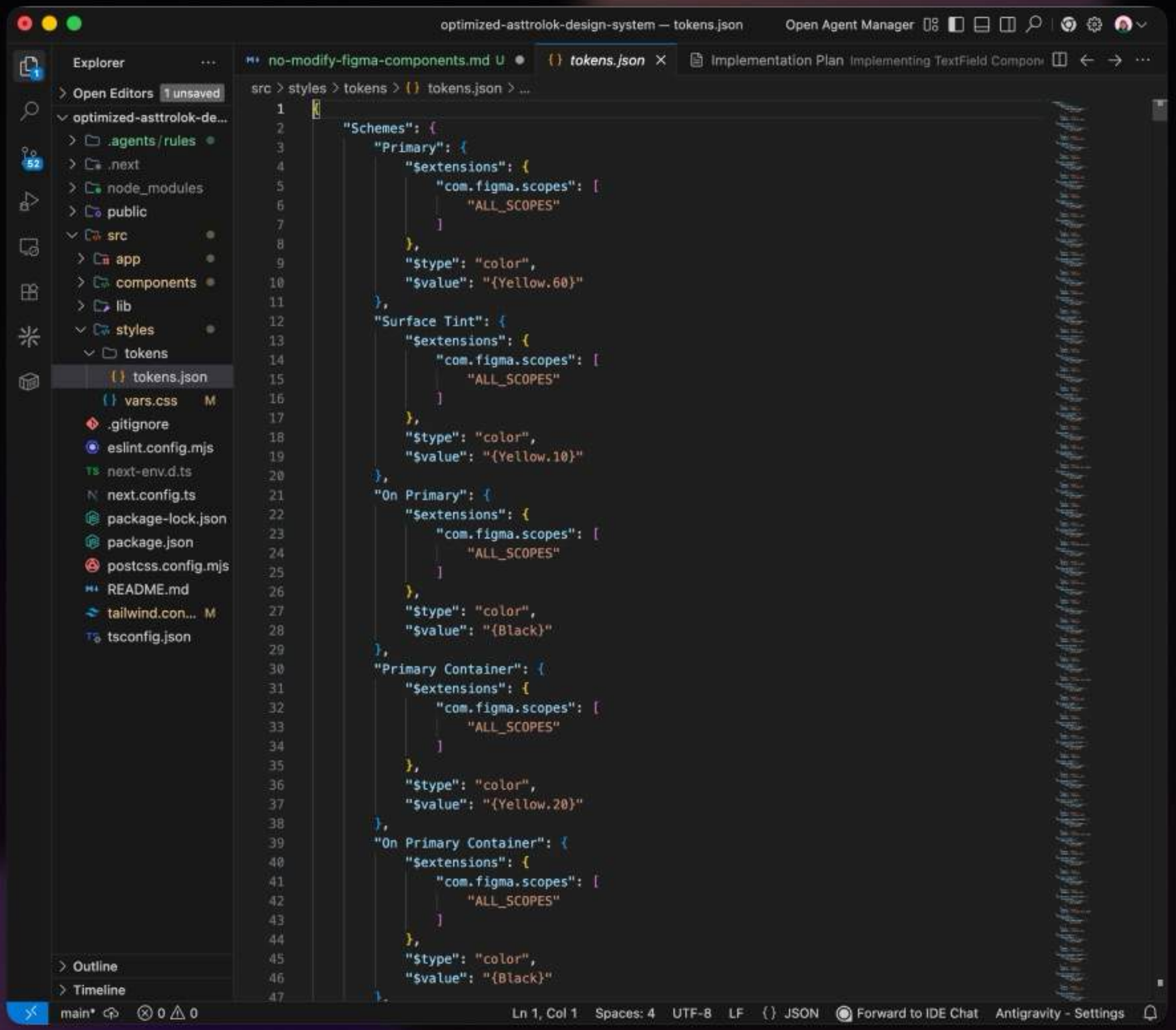
.tsx: File format where you can define components, Tailwind CSS and Javascript interaction-behavior together in the same file.

Tailwind CSS: Compact way assigning CSS properties for object without defining specific object class.

Step 5

Automated Syncing tokens to CSS variables

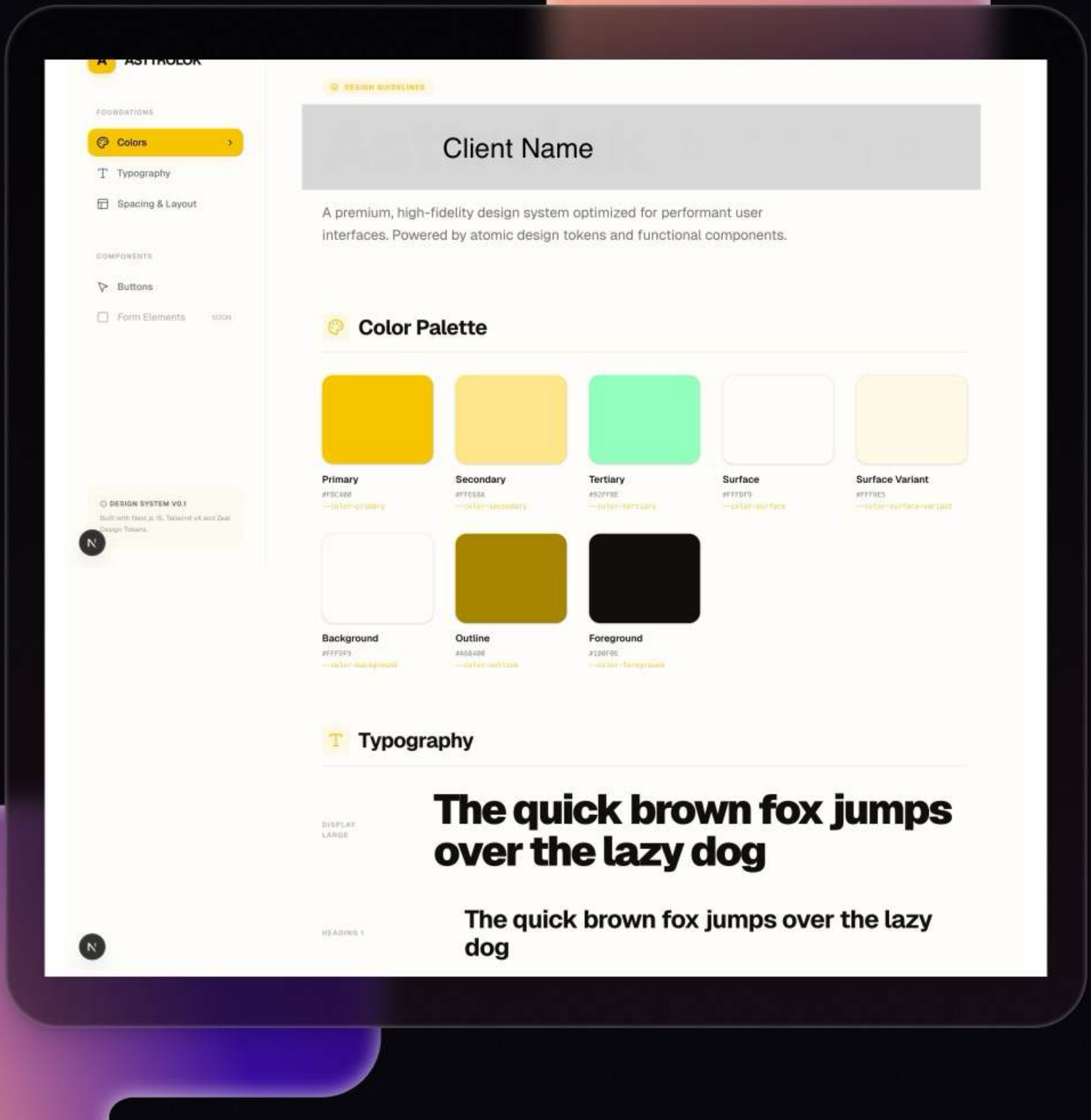
1. Copy tokens.json file into /styles/tokens/ folder.
2. Give prompt to Google Antigravity Agent
   "Analyze the tokens.json file. Update the tailwind.config.ts to extend the theme with these specific values for colors, font sizes, and spacing. Create a vars.css file in /styles that maps these tokens to CSS variables for maximum flexibility"


optimized-asttrolok-design-system — tokens.json
Open Agent Manager
Explorer
Open Editors 1 unsaved
optimized-asttrolok-de...
.agents/rules
.next
node_modules
public
src
app
components
lib
styles
tokens
tokens.json
vars.css M
.gitignore
eslint.config.mjs
next-env.d.ts
next.config.ts
package-lock.json
package.json
postcss.config.mjs
README.md
tailwind.con... M
tsconfig.json
Outline
Timeline
no-modify-figma-components.md U
tokens.json
Implementation Plan Implementing TextField Compon
src > styles > tokens > tokens.json > ...
{
"Schemes": {
"Primary": {
"$extensions": {
"com.figma.scopes": [
"ALL_SCOPES"
]
},
"$type": "color",
"$value": "{Yellow.60}"
},
"Surface Tint": {
"$extensions": {
"com.figma.scopes": [
"ALL_SCOPES"
]
},
"$type": "color",
"$value": "{Yellow.10}"
},
"On Primary": {
"$extensions": {
"com.figma.scopes": [
"ALL_SCOPES"
]
},
"$type": "color",
"$value": "{Black}"
},
"Primary Container": {
"$extensions": {
"com.figma.scopes": [
"ALL_SCOPES"
]
},
"$type": "color",
"$value": "{Yellow.20}"
},
"On Primary Container": {
"$extensions": {
"com.figma.scopes": [
"ALL_SCOPES"
]
},
"$type": "color",
"$value": "{Black}"
},
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
{} JSON
Forward to IDE Chat
Antigravity - Settings

Step 6

Build components with Storybook-like Preview

1. Prompt to Google Antigravity Agent
   "Build a React component for a 'Primary Button' in /components/ui/Button.tsx. Use Next.js 14+ standards and Tailwind CSS. Reference the theme values we just added to the tailwind config. Ensure it is accessible (ARIA labels), supports multiple states (hover, active, disabled), and is responsive."
2. *"Run a local dev server and use the integrated browser to verify the visual fidelity of the generated components against the Figma design. Create a Storybook-style 'Preview' page at /design-system to showcase all components."*


FOUNDATIONS
Colors
Typography
Spacing & Layout
COMPONENTS
Buttons
Form Elements
SOON
DESIGN SYSTEM V0.1
Client Name
A premium, high-fidelity design system optimized for performant user interfaces. Powered by atomic design tokens and functional components.
Color Palette
Primary
Secondary
Tertiary
Surface
Surface Variant
Background
Outline
Foreground
Typography
DISPLAY LARGE
The quick brown fox jumps over the lazy dog
HEADING 1
The quick brown fox jumps over the lazy dog

Step 7

Manual Verification Check

1. For accessibility - Visual Readability, Color Contrast
2. Component Accuracy
    a. States and Interaction
    b. Visual
    c. Variants
3. Webapp Interaction

BONUS I

Interface Experience Enhancement

Incremental Prompting to go from 0 → 1 → beyond

add the button different states Default, Active, Enabled, Focus, Disabled for the Button Component and preview it on the design system webpage under Button section

Thought for 3s

Implementing Button States
I've researched the current button implementation and tokens. I've identified that the { } tokens.json contains state layer opacities that can be used for hover, focus, and pressed states. I'm now creating an implementation plan.

Files Edited

Also add the Secondary Button (Tonal) along with Primary, Elevated, Ghost Button.

Thought for 12s

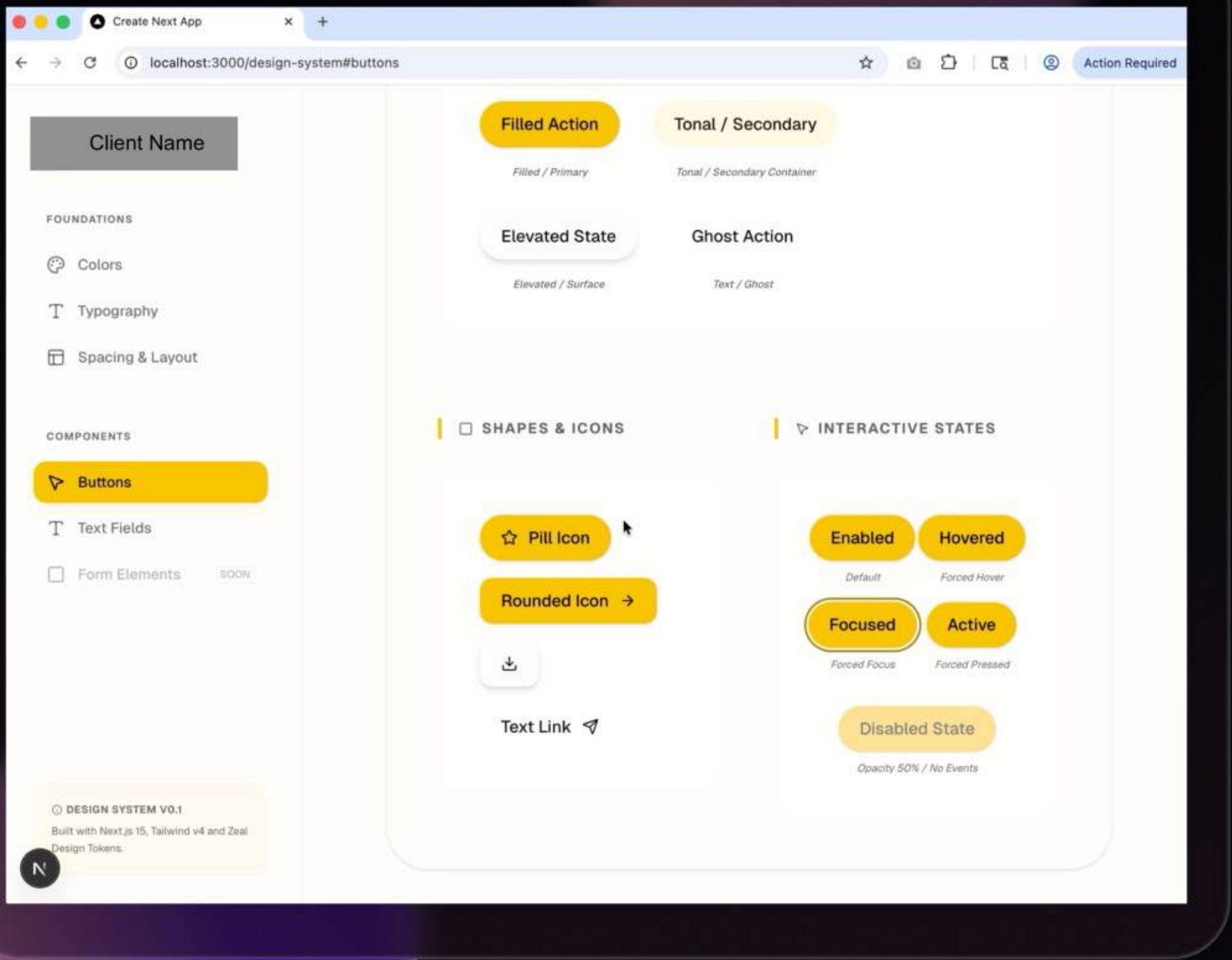
Implementing Tonal Button Variant
I have successfully implemented the "Secondary Button (Tonal)" variant and verified it in the browser. The new variant is now visible in the design system preview page. I'm now finalizing the documentation and notifying the user.

Change the label grey color at different places on the web page to make it visually accessible. Make sure to NOT change any design system elements

Thought for 6s

Improving Text Accessibility
I've verified the accessibility improvements for the grey labels via browser screenshots. The contrast is now much better for all descriptive text and headers. I'm now adding this to the walkthrough and closing the task.

Review Changes


Create Next App
localhost:3000/design-system#buttons
Action Required
Client Name
FOUNDATIONS
Colors
Typography
Spacing & Layout
COMPONENTS
Buttons
Text Fields
Form Elements
SOON
DESIGN SYSTEM V0.1
Built with Next.js 15, Tailwind v4 and Zeal Design Tokens.
Filled Action
Filled / Primary
Tonal / Secondary
Tonal / Secondary Container
Elevated State
Elevated / Surface
Ghost Action
Text / Ghost
SHAPES & ICONS
INTERACTIVE STATES
Pill Icon
Rounded Icon
Text Link
Enabled
Default
Hovered
Forced Hover
Focused
Forced Focus
Active
Forced Pressed
Disabled State
Opacity 50% / No Events

BONUS II

Managing efficient AI to avoid unintended behavior

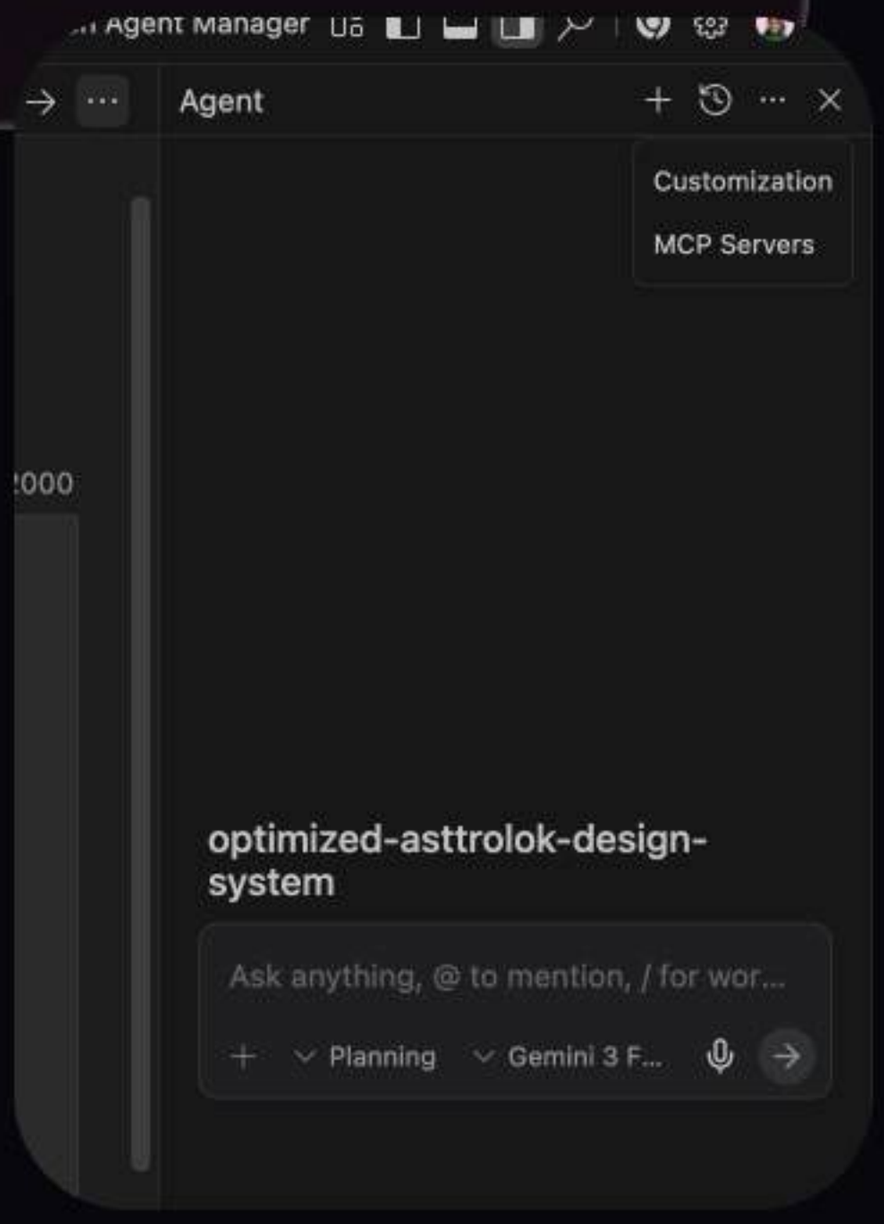
Using Skills, Rules and Workflows

You can add skills, rules and workflows to help AI agent create intended output.

- Rules: Always On Constraints

- Skills: On Demand Expertise. Not always in AI's active memory. Skills are referenced only when a particular task needs it.

- Workflows: Sequence of steps triggered on a certain event.



Example Rule: .agent/rules/token-validator.md

Trigger: File match `src/components/**/*.tsx`
Logic: If the agent detects a literal CSS color (e.g., `#FFFFFF`), it must stop and ask to replace it with a token from `src/styles/tokens.json`.
Constraint: Do not export components that lack a corresponding `.stories.tsx` file.

Example Skill: .agent/skills/accessibility-skill.md

When user asks for Accessibility check and rectification of text and color contrast, exclude the UI components that have .tsx file in /components/ui/ folder to make sure the Design System components are represented accurately visually.

Example Workflow: .agent/workflows/ship-component.md (The "Automation")

Workflow: Ship Component **Description**: Full automation for finishing a component.
Steps
1. **Lint**: Run `npm run lint` and fix any auto-fixable issues.
2. **Document**: Run the `accessibility-skill.md` skill on the current file.
3. **Commit**: Create a commit with the message "feat(ui): add [component name] and stories".

BONUS III

Sustainable AI Usage

1. High Resolution Incremental Screenshots Analysis to convert Figma screens into code can take between 20,000 - 50,000 Tokens per component. Pro-Tip - Wherever possible use meta data from the Figma MCP directly to save tokens. Other options are free plugins to convert parts of the code into .JSON file, or keep the Figma file dev mode ON in the browser so that Antigravity Agent can "read" the text in the Inspect panel just like it reads a code file. It scrapes the CSS values (padding, hex codes, font sizes) directly from the browser's memory.
2. Writing code is relatively inexpensive over reading file. Reading involves the hidden cost of thinking and context memory storage. Whenever possible provide clarification on deterministic tasks to save tokens.

Thank you !

STAY TUNED FOR
WEEKLY CONTENT.
SUBSCRIBE TO FOLLOW