



skillz

Design System

TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings



OVERVIEW

Background

Skillz is a gaming platform that helps over **30,000** game developers bring their games to life.

At the time, our web services team had just migrated over to Figma as our design tool, and we needed to migrate over our Sketch components to a new Figma native system.

Objective

The aim of this project was to revamp the previous **Design System** to not only be Figma native, but also have more variants defined out.

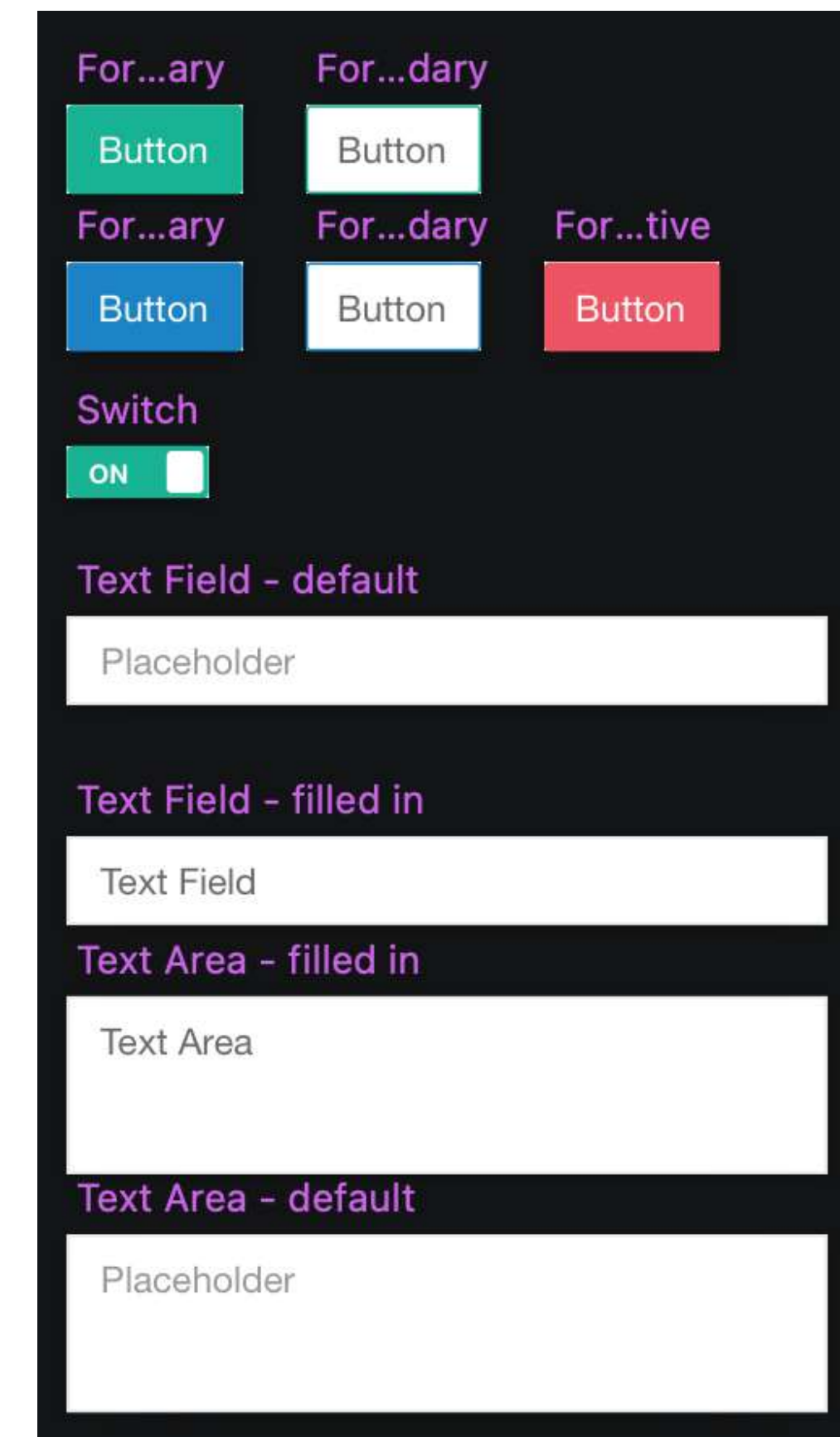
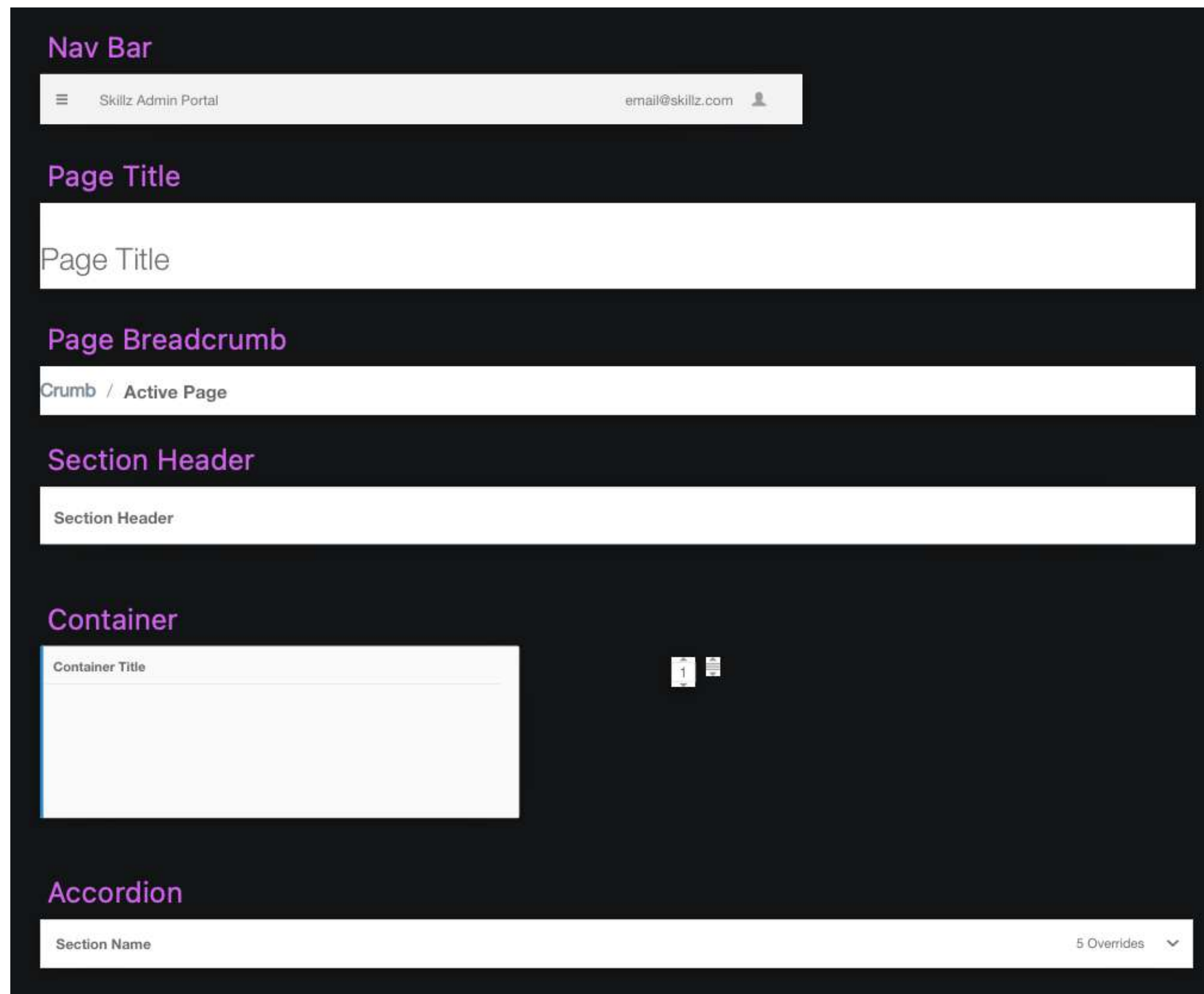
Role

With the rest of the Web Services team designers, as a group of 3 we decided to divide and conquer tackling creating a new design system.



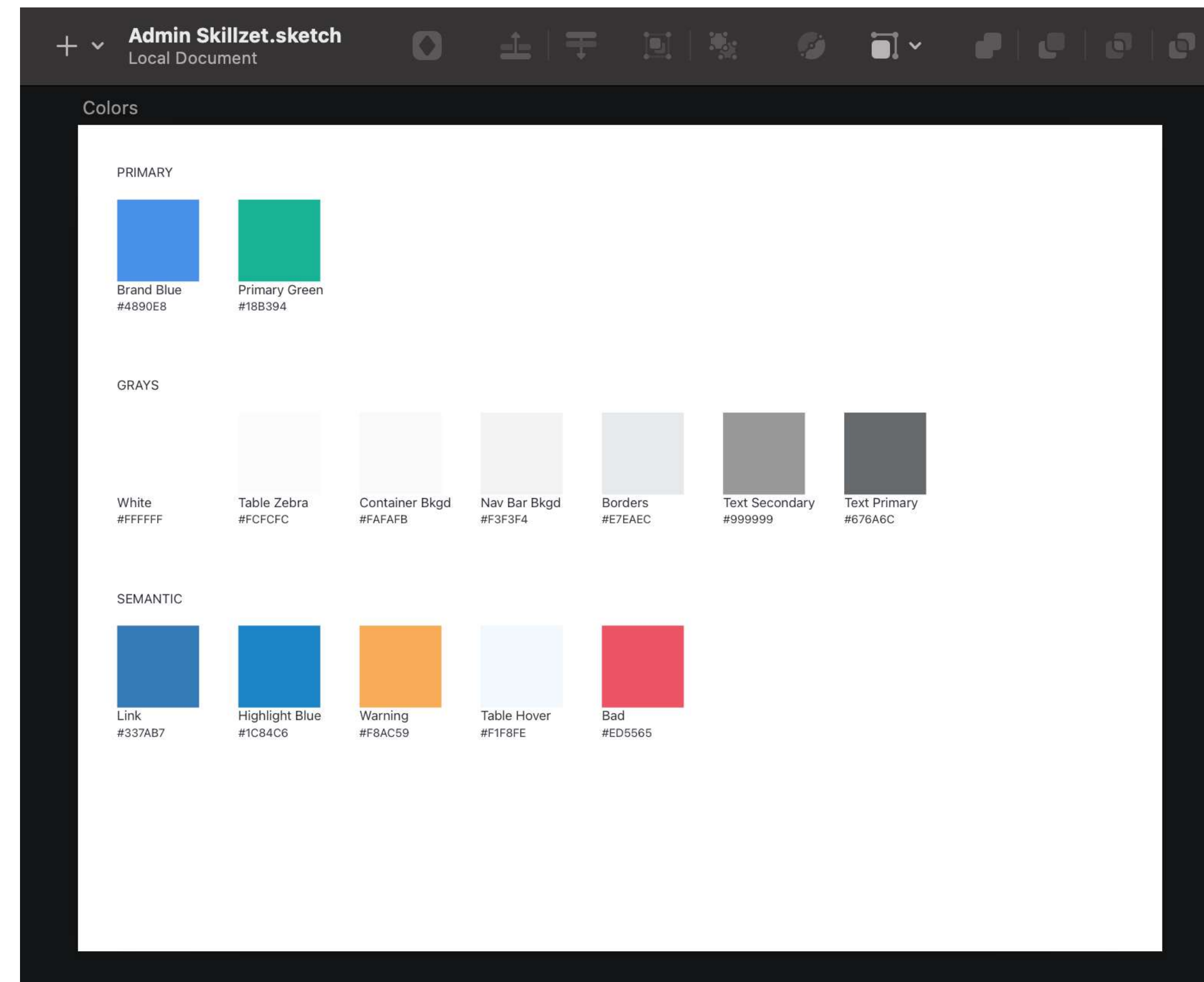
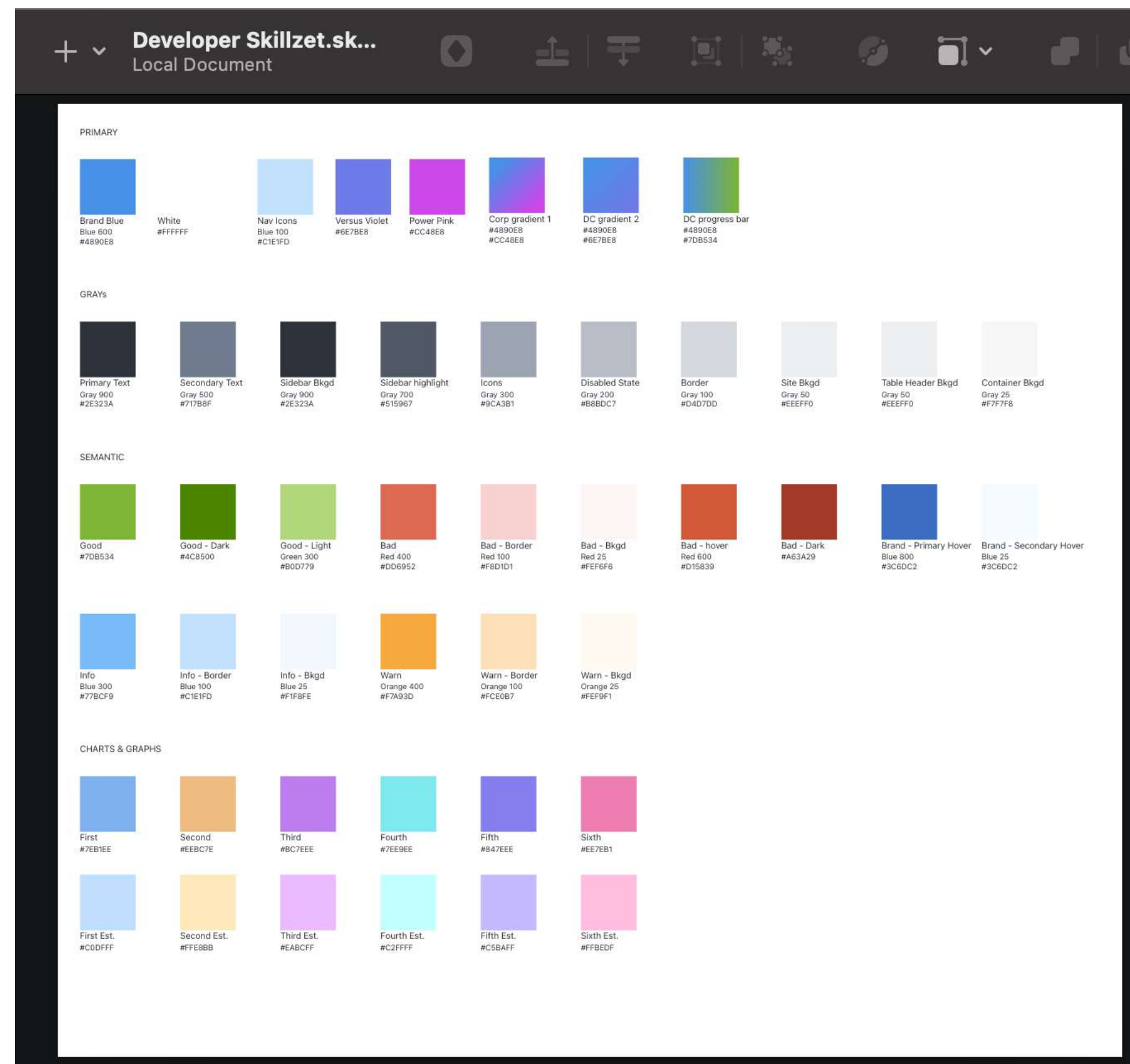
PREVIOUS DESIGN SYSTEM

Our previous design system was a collection of base components on Sketch that had minimal variants.



PREVIOUS DESIGN SYSTEM

We were also losing a lot of design efficiency since we had **2 separate design systems** to maintain for each of our products. Below are the color definitions for our **Developer Console** and **Admin Portal** which utilized different atomic elements despite being under the same brand.



PREVIOUS DESIGN SYSTEM

Despite our front end partners using the **MUI React Library**, our previous design system didn't take advantage of the naming conventions that our developers were already using.

The image is a composite of two parts. On the left is a code editor window titled "Basic button". It contains the text: "The `Button` comes with three variants: text (default), contained, and outlined." Below this text are three button variants: "TEXT" (a simple text label), "CONTAINED" (a solid light blue button), and "OUTLINED" (a white button with a light blue border). Below the buttons is a toolbar with icons for code, copy, paste, undo, redo, and a menu. At the bottom of the editor is a code block with the following JSX:

```
<Button variant="text">Text</Button>
<Button variant="contained">Contained</Button>
<Button variant="outlined">Outlined</Button>
```

 On the right is a grid of five button variants, each with a label above it: "Forms/Buttons/Green Primary" (a solid green button), "Forms/Buttons/Green Secondary" (a white button with a green border), "Forms/Buttons/Blue Primary" (a solid blue button), "Forms/Buttons/Blue Secondary" (a white button with a blue border), and "Forms/Buttons/Destructive" (a solid red button).



PROBLEMS

Problem #1:

Fragmented System

Despite reusing many components between our Developer Experience and Admin Portal products, we referred to separate libraries for both of them.

Problem #2:

Sketch Native

With our old design system on Sketch, there was no way for us to actually reuse the components in our new workflow.

Problem #3:

No Shared Language

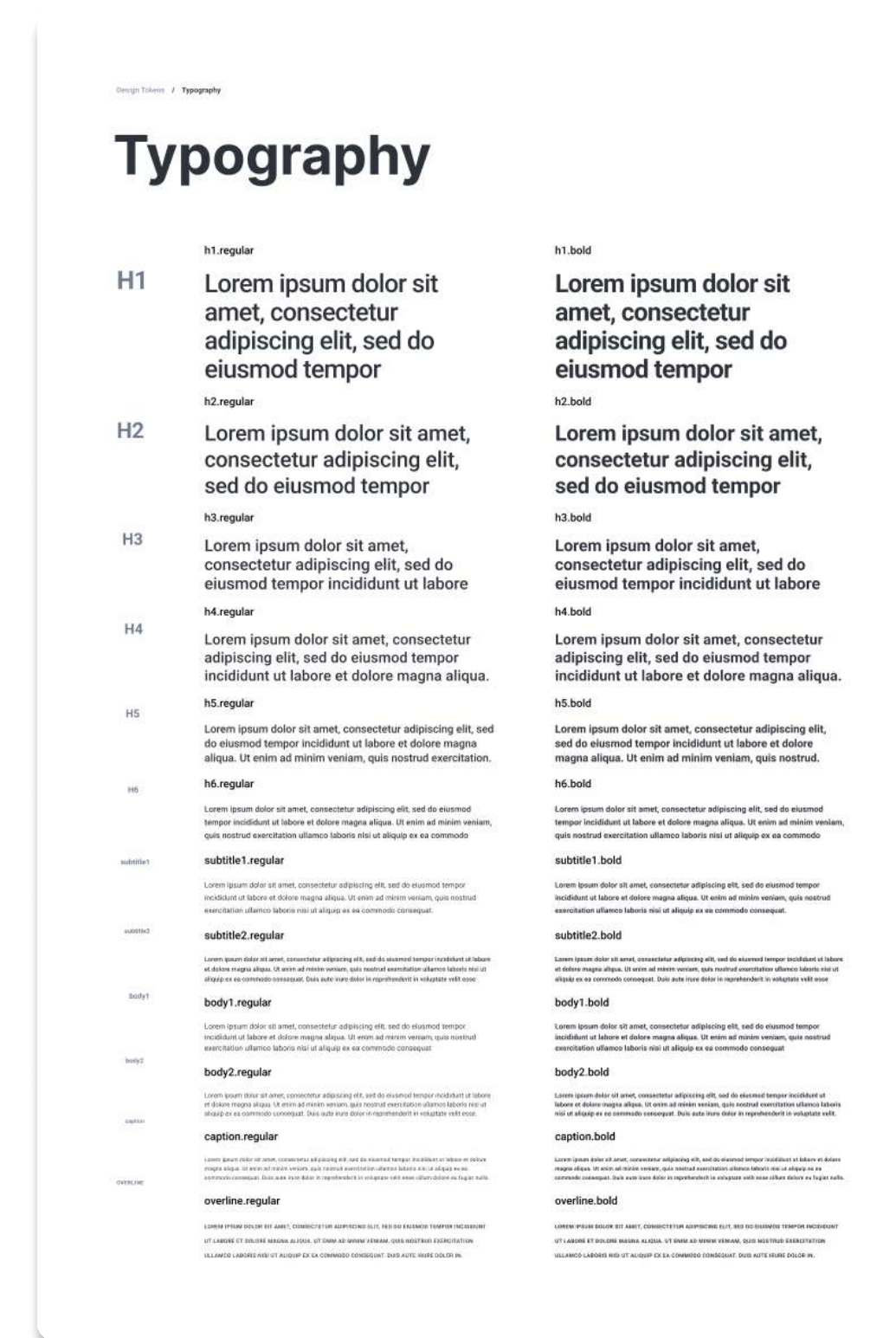
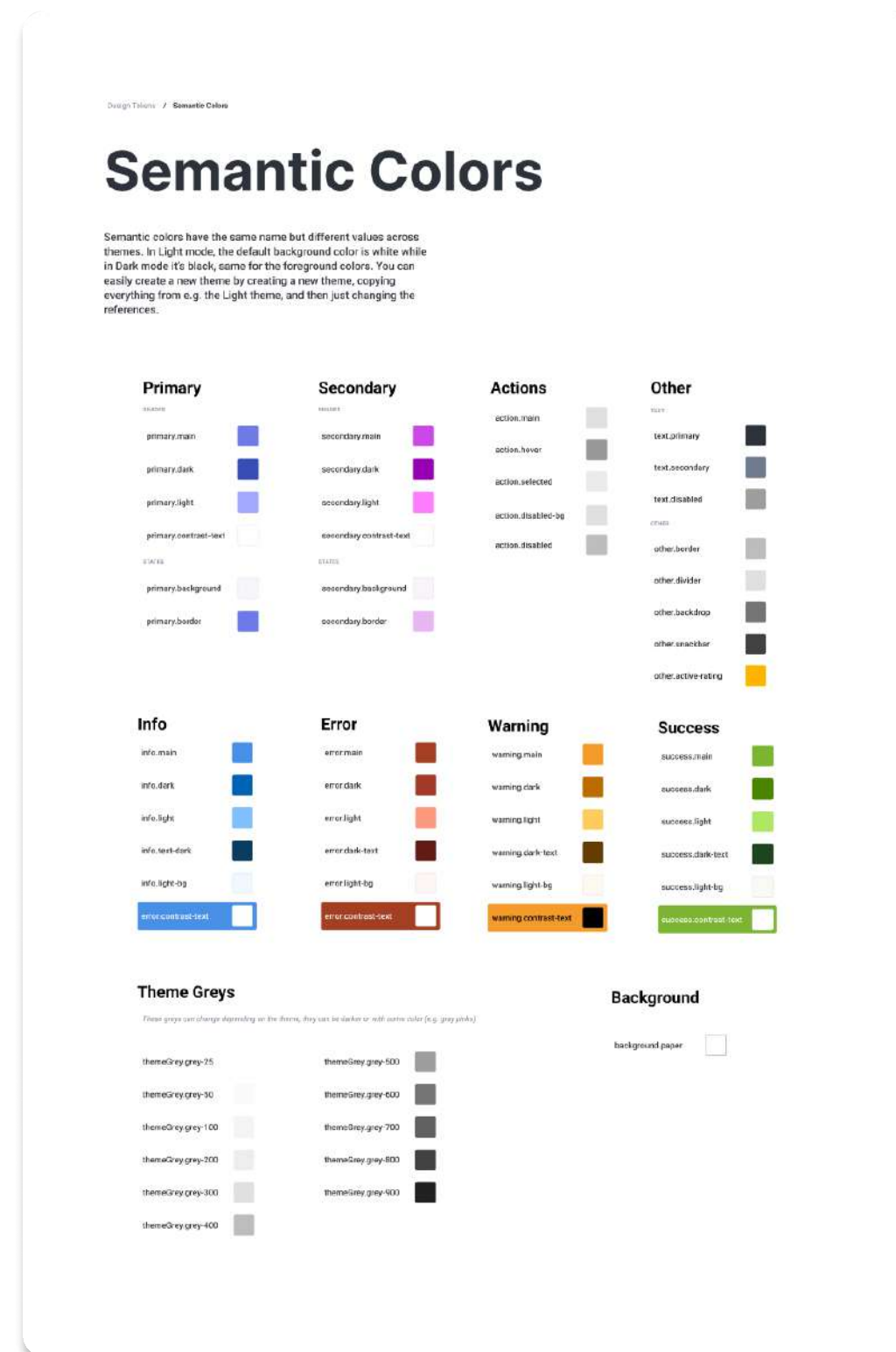
Despite our front end partners using the **MUI React Library**, our previous design system didn't take advantage of the naming conventions that our developers were already using.



SOLUTION

Unified System

Instead of having different atomic units defined for each individual product, we wanted to create a system where every Web Services product referred to the same set of colors, shadows, and typography to make design system maintenance more streamlined.



PROBLEMS

Problem #1:

Fragmented System

Despite reusing many components between our Developer Experience and Admin Portal products, we referred to separate libraries for both of them.

Problem #2:

Sketch Native

With our old design system on Sketch, there was no way for us to actually reuse the components in our new workflow.

Problem #3:

No Shared Language

Despite our front end partners using the **MUI React Library**, our previous design system didn't take advantage of the naming conventions that our developers were already using.



SOLUTION

Figma Native

By redefining our **atomic units** to be **Figma Native** we'd be able to easily reuse colors, shadows, and typography in all out Figma files. If we ever wanted to change one, by updating the master atomic unit, it would cascade to all of our Figma files.

Primary

SHADES

primary.main



primary.dark



primary.light



primary.contrast-text



STATES

primary.background



primary.border



Color Styles

Search

primary

contrast-text

main

light

dark

background

border

H1

h1.regular

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

H2

h2.regular

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

H3

h3.regular

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Text Styles

Search

h1

Ag regular · 56/112

Ag bold · 56/112

Ag italic · 56/112

h2

Ag regular · 48/120

Ag bold · 48/120

Ag italic · 48/120



PROBLEMS

Problem #1:

Fragmented System

Despite reusing many components between our Developer Experience and Admin Portal products, we referred to separate libraries for both of them.

Problem #2:

Sketch Native

With our old design system on Sketch, there was no way for us to actually reuse the components in our new workflow.

Problem #3:

No Shared Language

Despite our front end partners using the **MUI React Library**, our previous design system didn't take advantage of the naming conventions that our developers were already using.




SOLUTION

Shared Language

The biggest benefit of this initiative beyond the design team was improving cross-functional collaboration. This focus was on making sure the properties in our React components corresponded to how we named our variants.

Basic button

The `Button` comes with three variants: text (default), contained, and outlined.



```
<Button variant="text">Text</Button>
<Button variant="contained">Contained</Button>
<Button variant="outlined">Outlined</Button>
```



Button

- variant
 - ✓ contained
 - outlined
 - text
- size
- content type
- color
 - primary
- state
 - default



SOLUTION

Examples

Here are a couple examples of fully built out components in Figma in the new design system.

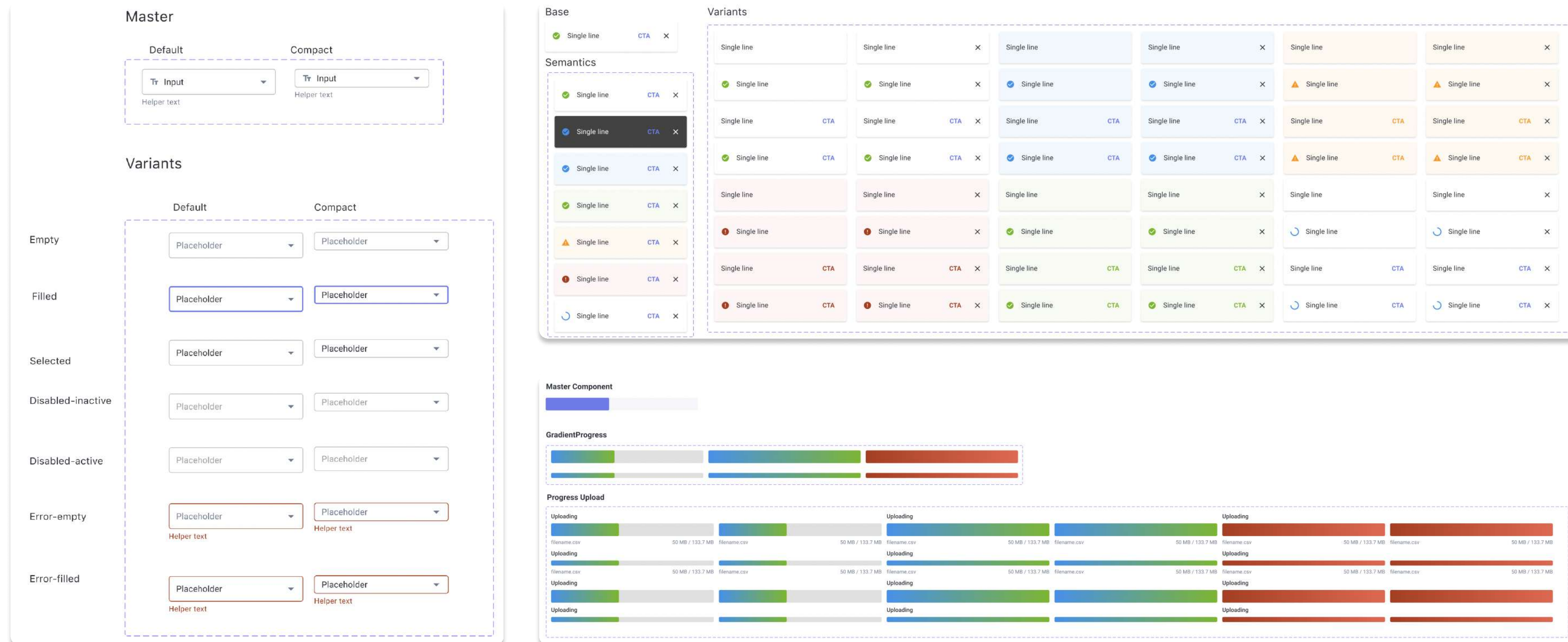


TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings



ATOMIC UNITS

Foundation

The foundation of our design system were the atomic units – properties we defined that were applied to every single component that was created.

Semantic Colors

Semantic colors have the same name but different values across themes. In Light mode, the default background color is white while in Dark mode it's black, same for the foreground colors. You can easily create a new theme by creating a new theme, copying everything from e.g. the Light theme, and then just changing the references.

Primary	Secondary	Actions	Other
primary.main	secondary.main	action.main	text.primary
primary.dark	secondary.dark	action.hover	text.secondary
primary.light	secondary.light	action.selected	text.disabled
primary.contrast.high	secondary.contrast.high	action.disabledBg	other.border
primary.contrast.low	secondary.contrast.low	action.disabled	other.disabled
primary.background	secondary.background		other.backdrop
primary.border	secondary.border		other.separator
			other.activeRing

Info	Error	Warning	Success
info.main	error.main	warning.main	success.main
info.dark	error.dark	warning.dark	success.dark
info.light	error.light	warning.light	success.light
info.contrast.high	error.contrast.high	warning.contrast.high	success.contrast.high
info.contrast.low	error.contrast.low	warning.contrast.low	success.contrast.low

Theme Greys	Background
themeGrey-15	background.paper
themeGrey-30	
themeGrey-60	
themeGrey-100	
themeGrey-200	
themeGrey-300	
themeGrey-400	

Shadows

Shadow elevations are defined by their elevation value. The elevation value is used to determine the shadow's opacity and blur radius. The elevation value is also used to determine the shadow's color. The elevation value is also used to determine the shadow's spread.

Elevation
Elevation.1
Elevation.2
Elevation.3
Elevation.4
Elevation.5
Elevation.6
Elevation.7
Elevation.8
Elevation.9
Elevation.10
Elevation.11
Elevation.12
Elevation.13
Elevation.14
Elevation.15
Elevation.16
Elevation.17
Elevation.18
Elevation.19
Elevation.20
Elevation.21
Elevation.22
Elevation.23
Elevation.24

Typography

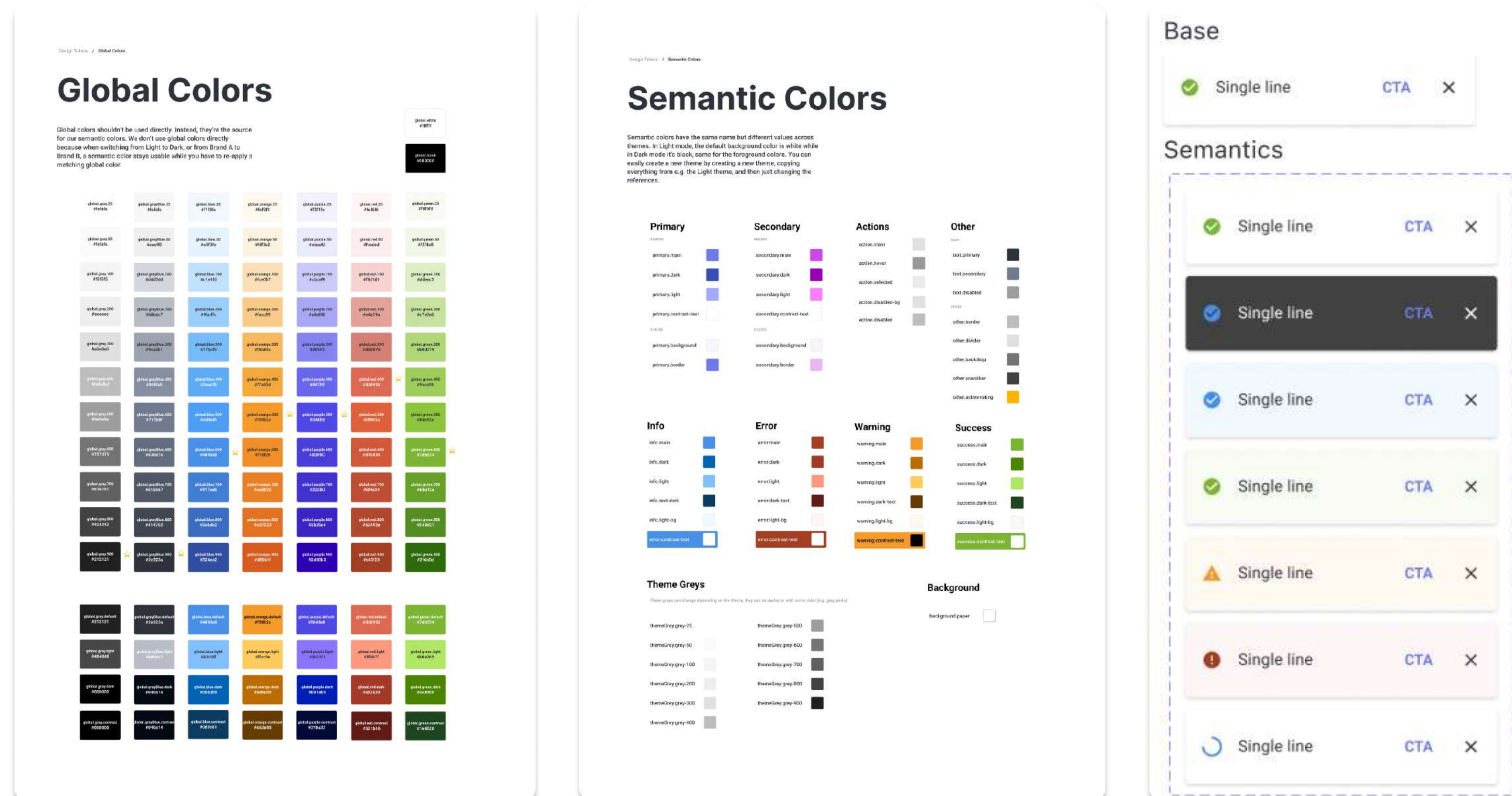
Font Weight	Font Style	Text
h1	regular	h1.regular
h1	bold	h1.bold
h2	regular	h2.regular
h2	bold	h2.bold
h3	regular	h3.regular
h3	bold	h3.bold
h4	regular	h4.regular
h4	bold	h4.bold
h5	regular	h5.regular
h5	bold	h5.bold
h6	regular	h6.regular
h6	bold	h6.bold
subheader	regular	subheader.regular
subheader	bold	subheader.bold
body	regular	body1.regular
body	bold	body1.bold
body	regular	body2.regular
body	bold	body2.bold
caption	regular	caption.regular
caption	bold	caption.bold
overline	regular	overline.regular
overline	bold	overline.bold



ATOMIC UNITS

Colors

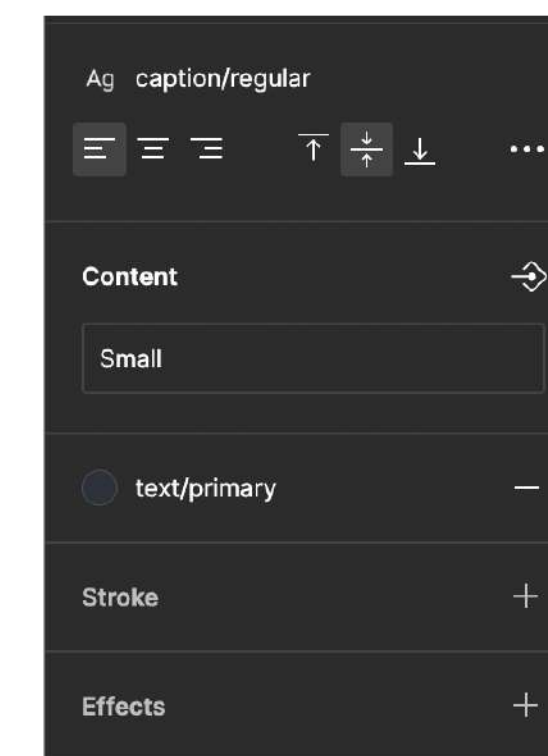
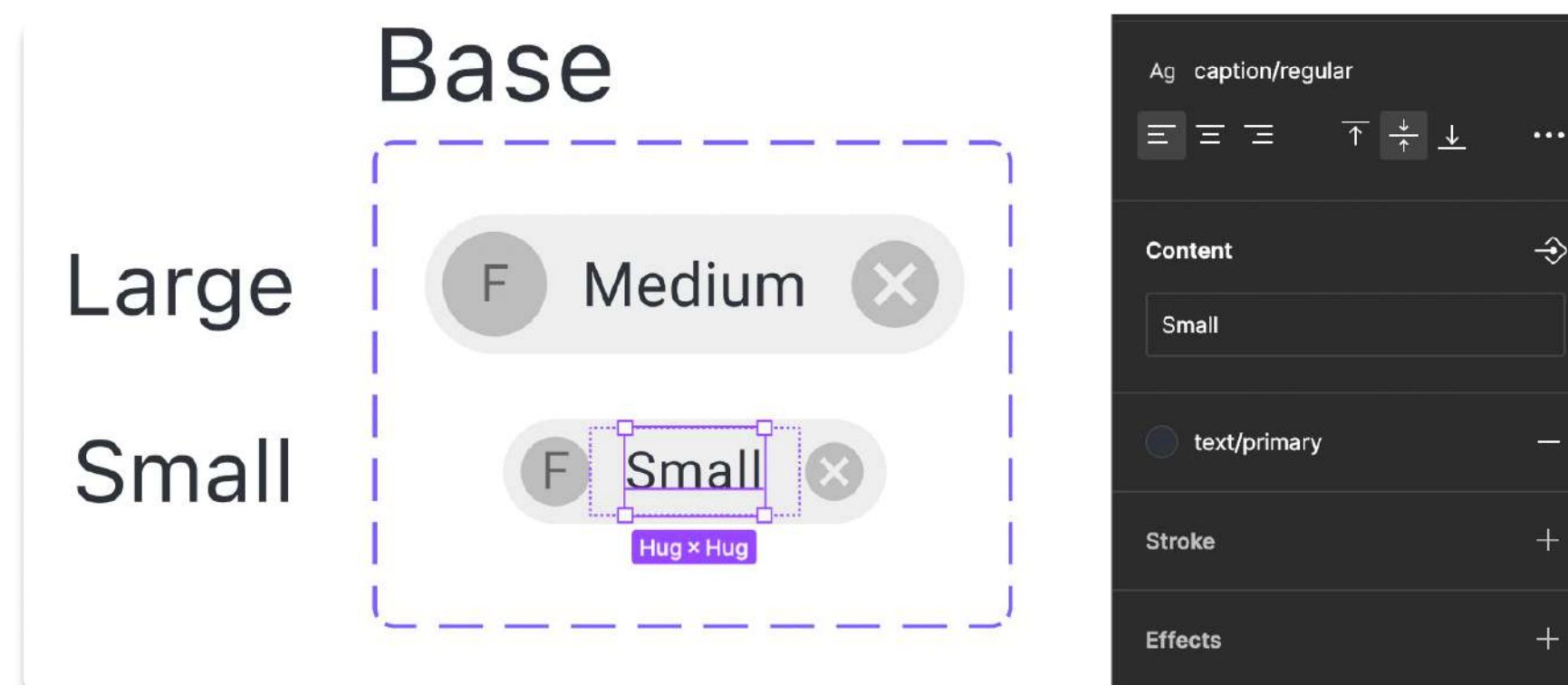
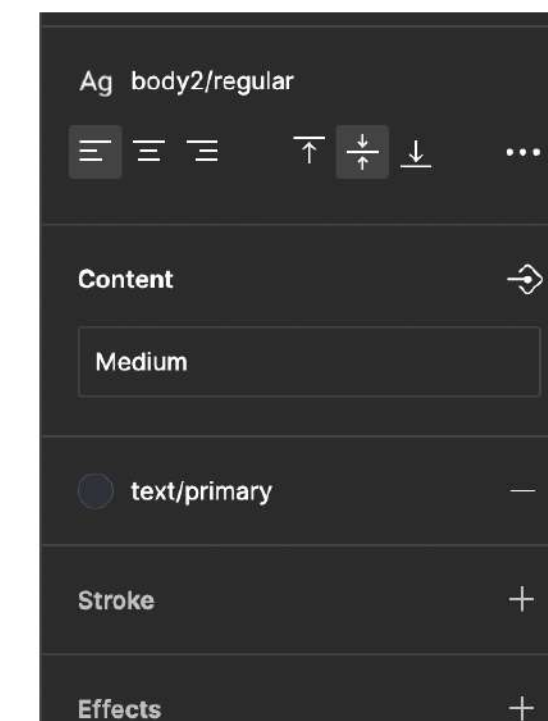
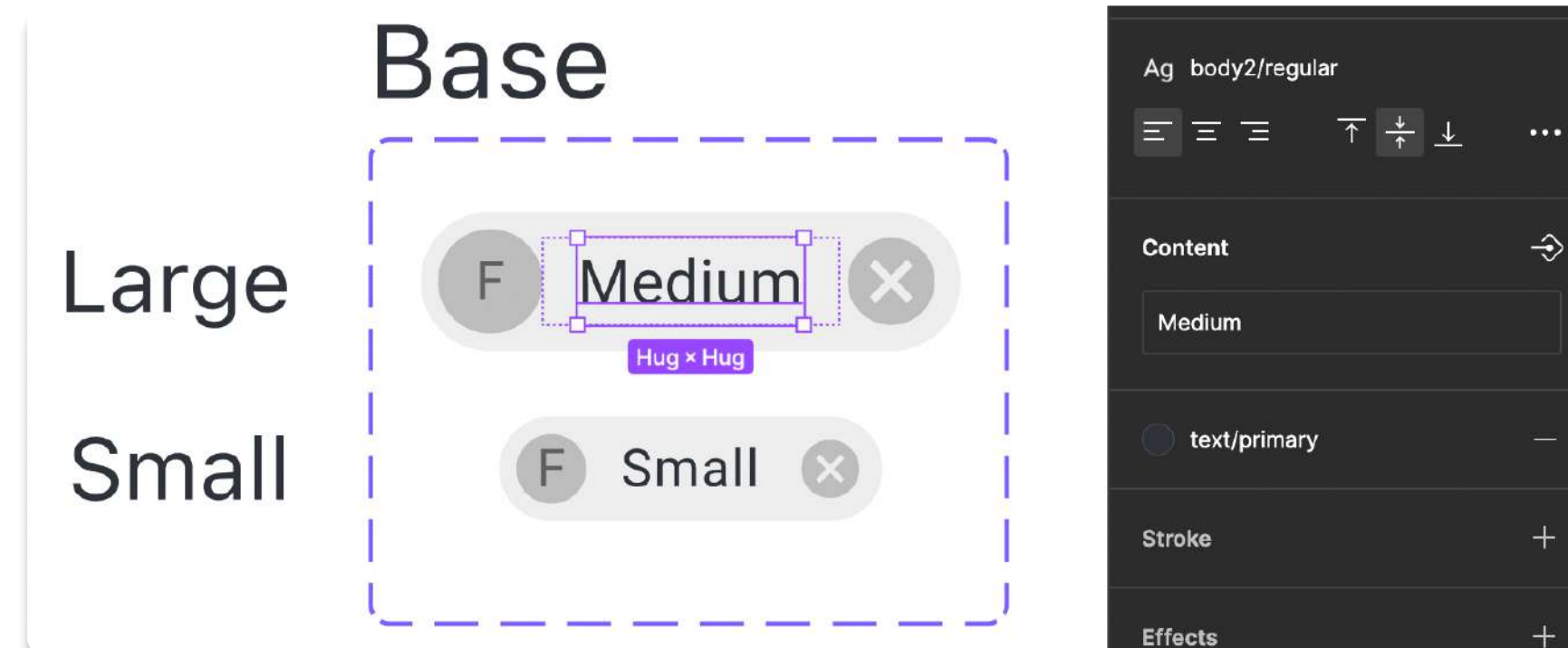
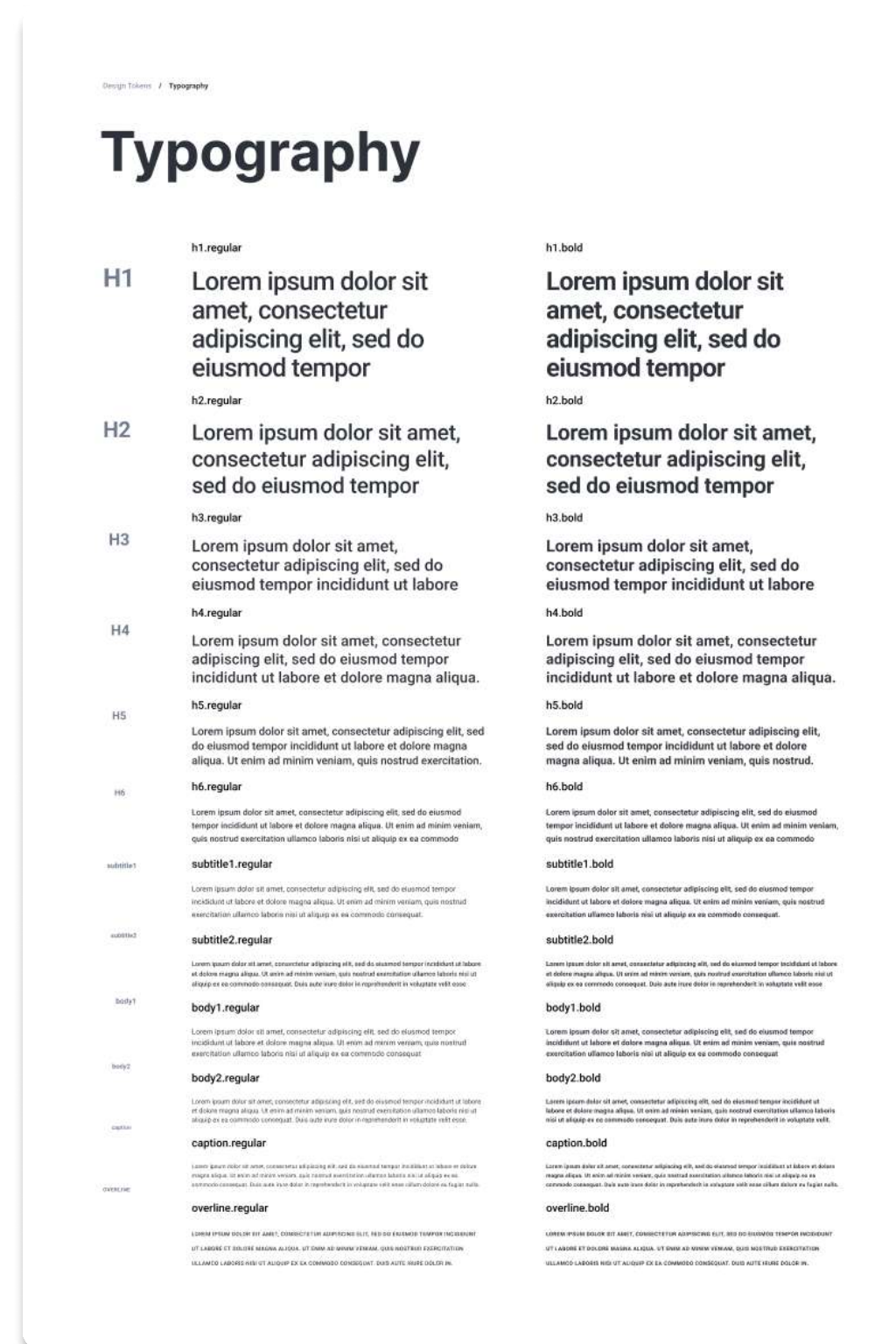
The first of these atomic units was Color since we wanted a cohesive Skillz feel to all of our web services products. We began by using our core brand colors and creating a full range of shades of each color. This allowed us to define semantic colors we could use for **info**, **success**, **warning**, and **error** states.



ATOMIC UNITS

Typography

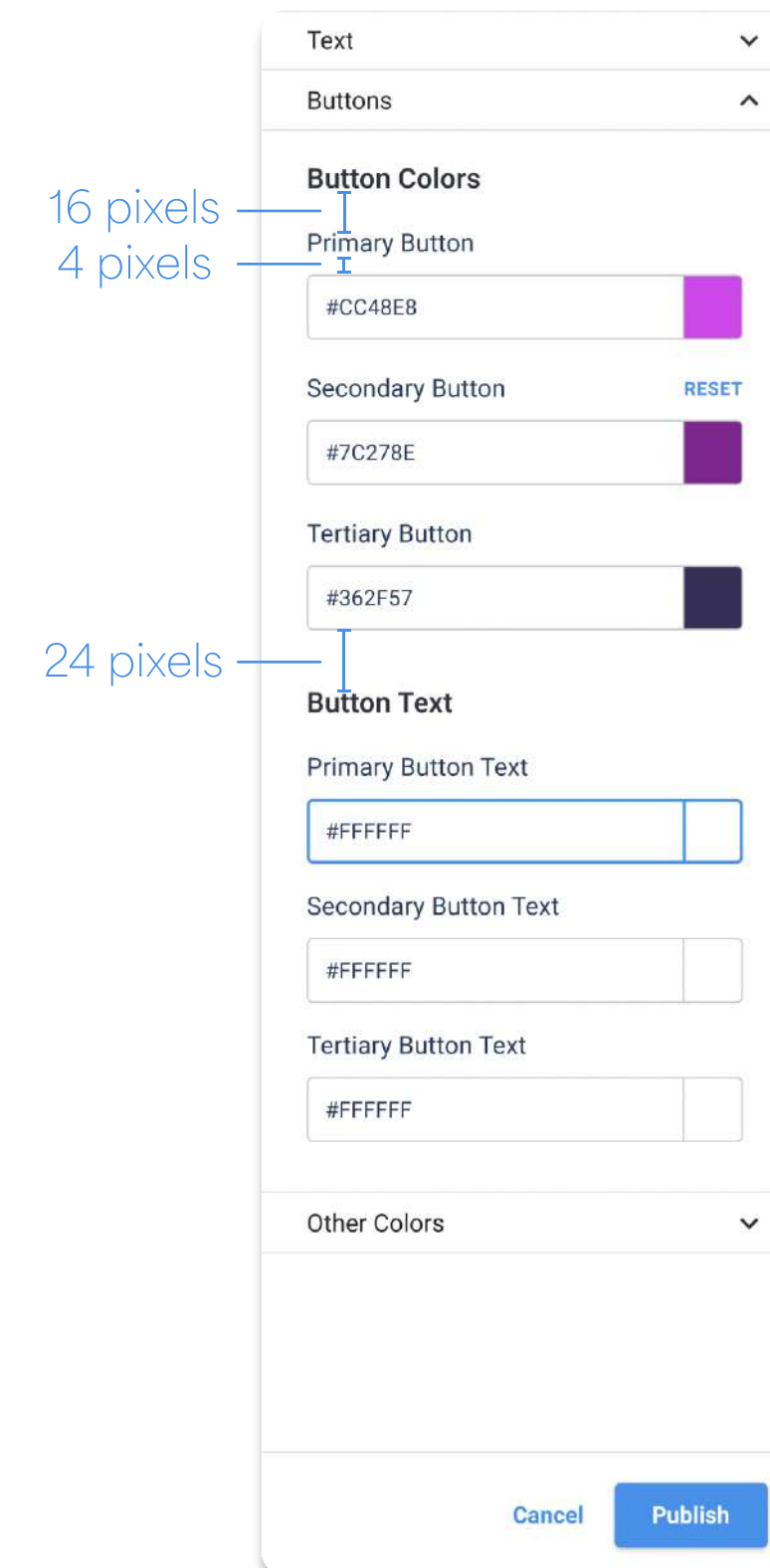
The next atomic unit we defined out was typography. This allowed us to have a consistent sense of size and spacing for all of our components that we reused. Here you can see the difference between a **Body** vs **Caption** typography component.



ATOMIC UNITS

Spacing

The last of our atomic units is size and spacing. We stuck to the 8 point grid to define these since it helps with having quarter and half steps to create a sense of grouping for our more complex components.



ATOMIC UNITS

Shadows

The last of our atomic units is shadows. Having varying depths of shadows helps create a sense of hierarchy for our overlaid components.

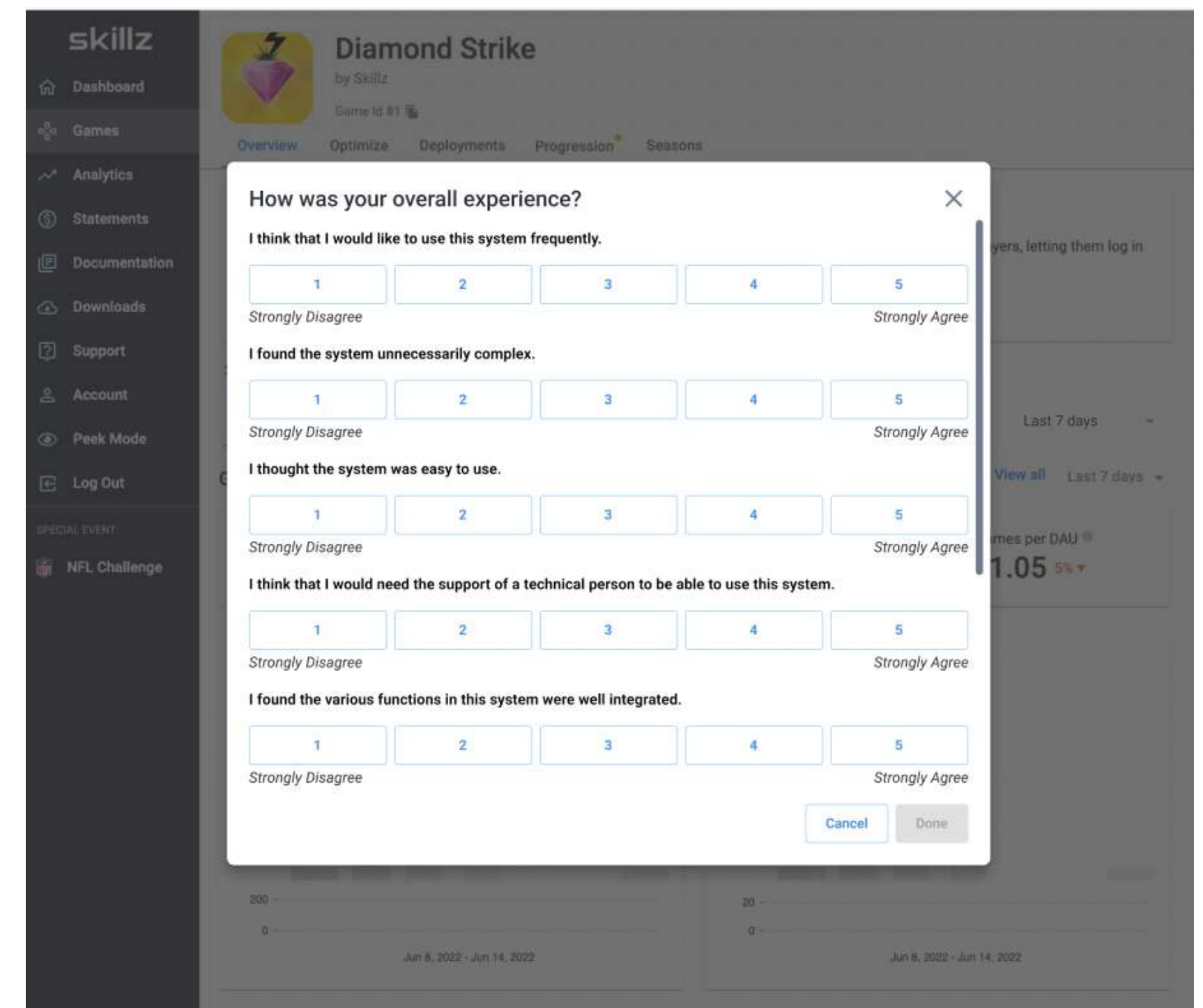
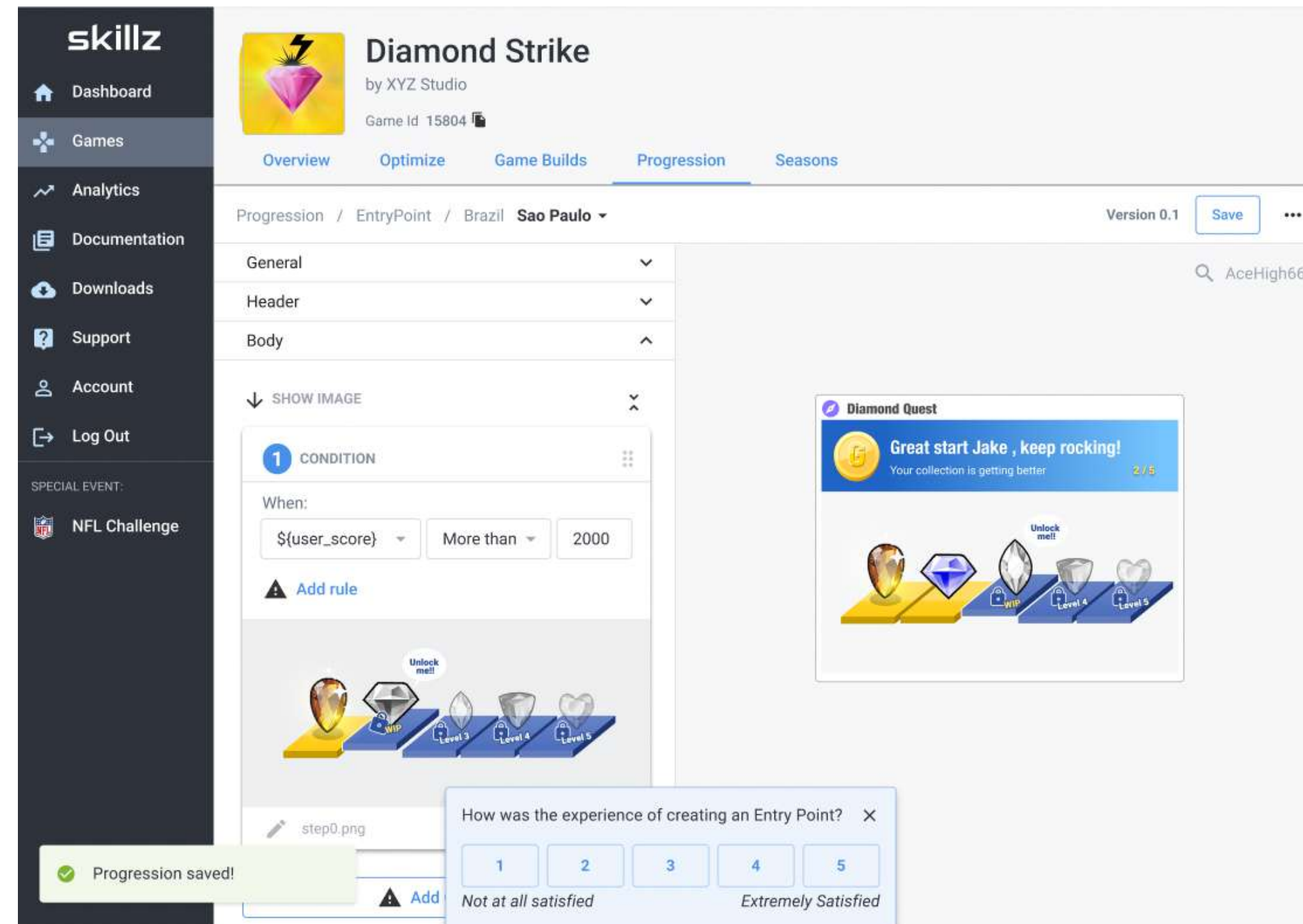
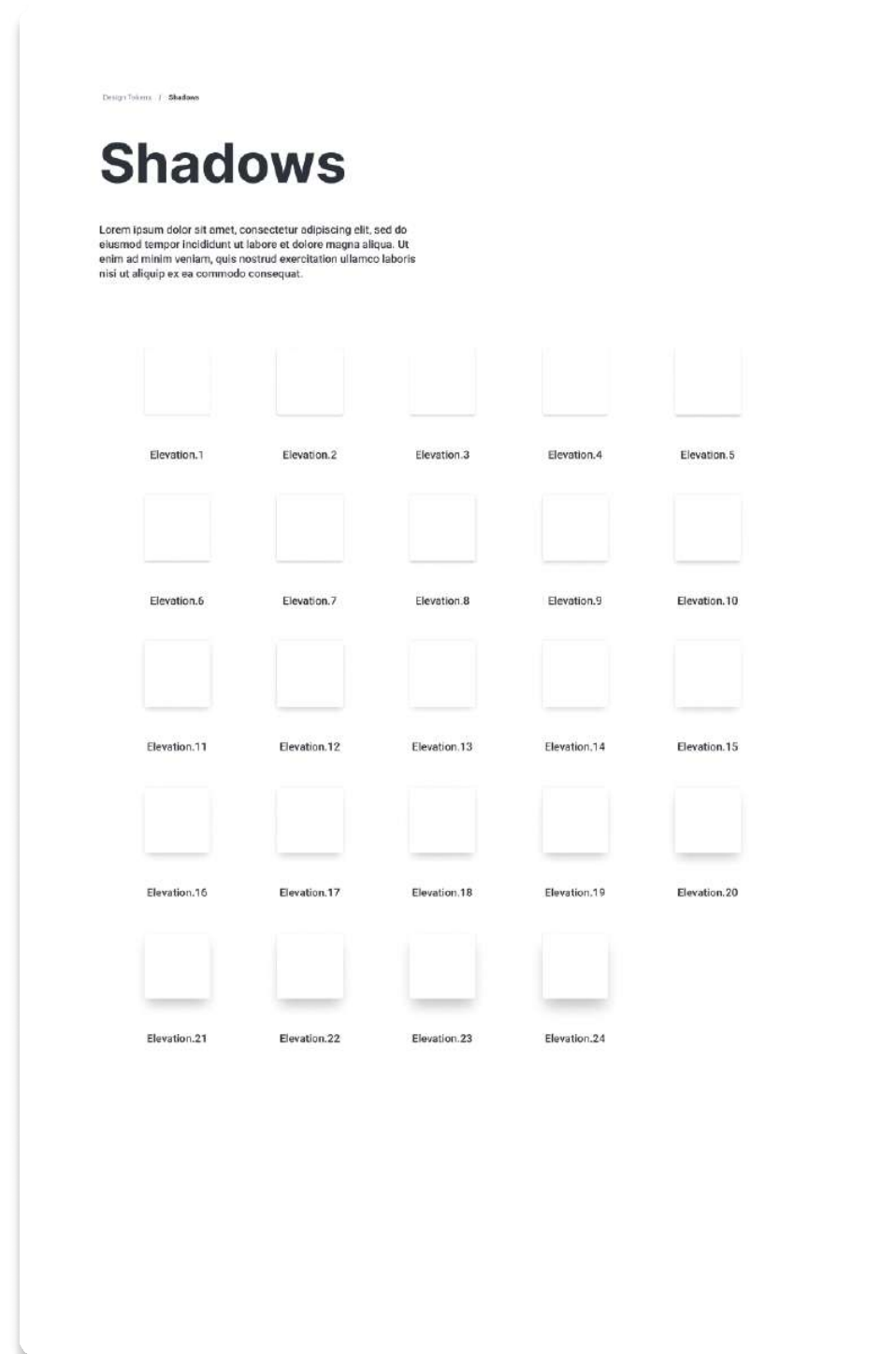


TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings



COMPONENTS

Components

The second layer to our design system was creating components. These would be built on top of the atomic units we defined and be consistent with what our developers used from the Material UI library.

The image displays a design system component library. It is organized into several sections:

- Master:** Shows a 'Default' and 'Compact' version of a 'Tr Input' component with 'Helper text'.
- Variants:** A large grid showing various states and styles for the 'Single line' component. It includes a 'Semantics' section with icons (checkmark, error, warning, info) and a grid of 'Single line' components with different colors (green, blue, orange, red, yellow) and CTA icons.
- Master Component:** Shows a 'Master Component' with a blue progress bar.
- GradientProgress:** Shows a 'GradientProgress' component with a multi-colored gradient bar.
- Progress Upload:** Shows a 'Progress Upload' component with multiple 'Uploading' bars, each with a filename and progress indicator (e.g., 'filename.csv 50 MB / 133.7 MB').



COMPONENTS

Master Components

One thing to note in the creation of components is the use of master components. This is a top level component to 'rule them all'. These define the base structures of the component itself, and like atomic units, when changed will cascade down.

Base

✔ Single line CTA ✕

Semantics

✔ Single line CTA ✕

✔ Single line CTA ✕

✔ Single line CTA ✕

✔ Single line CTA ✕

⚠ Single line CTA ✕

❗ Single line CTA ✕

🔄 Single line CTA ✕

Master Elements

🏠 Label Active Link

Variants

Breadcrumb Active	🏠 Breadcrumb Active
Breadcrumb Hover	🏠 Breadcrumb Hover
Breadcrumb Disabled	🏠 Breadcrumb Disabled

Master Component



GradientProgress



COMPONENTS

Variant Matrix and Properties

These master components would just be the first layer to the variant matrix we would create for these components. By defining out each individual property in a matrix you could easily see every single variant of a component indexed by its properties. This also helped in the maintenance of the system by being able to pinpoint which exact variant needed to be touched up.

The image displays a design tool interface with two main sections: 'Base' and 'Variants'.

Base: Shows a single 'Single line' component with a 'CTA' property. It is marked with a green checkmark icon.

Semantics: A list of 'Single line' components with 'CTA' properties, each with a different icon (checkmark, error, warning, refresh). The second item in the list is highlighted with a dark background.

Variants: A 6x6 grid of 'Single line' components, each with a 'CTA' property. The variants are color-coded and have different icons (checkmark, error, warning, refresh) indicating their status.

Row	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
1	Single line	Single line	Single line	Single line	Single line	Single line
2	Single line	Single line	Single line	Single line	Single line	Single line
3	Single line	Single line	Single line	Single line	Single line	Single line
4	Single line	Single line	Single line	Single line	Single line	Single line
5	Single line	Single line	Single line	Single line	Single line	Single line
6	Single line	Single line	Single line	Single line	Single line	Single line



TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings



MAINTENANCE

Quality Assurance

One of our biggest collaboration tools when creating this design system was a checklist for making sure our components were being designed correctly. Naming convention, appropriate use of atomic units, stress testing, and responsiveness all needed to be checked off by two designers before we published a component to the design system.

Checklist Tester 1	Checklist Tester 2	Tests	Test Description
<input type="checkbox"/>	<input type="checkbox"/>	Hide Master and Base components	Master and Base component name should be in the format .componentname
<input type="checkbox"/>	<input type="checkbox"/>	Component name: Capitalized and Camelized	.baseColorPicker
<input type="checkbox"/>	<input type="checkbox"/>	Typography	All fonts are defined by the design system
<input type="checkbox"/>	<input type="checkbox"/>	Colors	All colors are defined by the design system
<input type="checkbox"/>	<input type="checkbox"/>	Using Skillzet Icons	Check that the icons are from the Skillzet Icons set. Should we check that any icon that is a component be used in other instances - i.e. a checkbox component on a table or form
<input type="checkbox"/>	<input type="checkbox"/>	Variant name tied to MUI API	Do this by referencing the component's page on the MUI dev doc site: Example - Breadcrumbs https://mui.com/components/breadcrumbs/ and add the link to the component's page
<input type="checkbox"/>	<input type="checkbox"/>	Spacing conventions (8 Base)	
<input type="checkbox"/>	<input type="checkbox"/>	Check content responsiveness	Test expected responsiveness of component. The responsiveness requirements will vary depending on a component's expected behavior
<input type="checkbox"/>	<input type="checkbox"/>	Stress Test	
<input type="checkbox"/>	<input type="checkbox"/>	Easy Variant Switching	
<input type="checkbox"/>	<input type="checkbox"/>	Test all variants	Toggle between variants to check that toggles are working and that no error messages are shown on the variants panel
<input type="checkbox"/>	<input type="checkbox"/>	Autolayout is working	
<input type="checkbox"/>	<input type="checkbox"/>	Accessible?	Confirm Text and Background contrast



DESIGN PROCESS

Best Practices

Another ongoing effort is making sure we have well-defined best practices for each of our components – from the spacing to the usage of the component itself.

Types

Single Line

Single Line With Icon

Single Line With Action

Single Line w Action + Icon

Single Line With Action

Single Line w Action + Icon

Specs

Single Line w Action + Icon

56px, 24px, 16px

Anatomy

Single Line w Action + Icon

1 Icon (optional), 2 Text Label, 3 Action (optional)

User input processing

Needs Documentation

Text fields are a combination of the field label (the title) and the input area. Inputs can be typed text, URLs, date pickers, and more.

Do

Prize Required

Placeholder

Do

Prize Required

100

Do

Prize Required

100

Do

Prize Required

100

Prize must be under 50

Upon unfocusing a text field, do a validation check

Don't

Prize Required

Placeholder

Don't

Prize Required

100

Prize must be under 50

Don't do a validation check while the field is still updating

Do

mm/dd/yyyy

22/06/2022

22/06/2022

22/06/2022

Month must be between 1-12

Upon unfocusing a text field, do a validation check

Don't

mm/dd/yyyy

22/06/2022

22/06/2022

Month must be between 1-12

Don't do a validation check while the field is still updating



DESIGN PROCESS

Patterns

Beyond best practices on an individual component level, we also needed to define out patterns for how these components interact with one another on a page level.

Patterns / Raising Errors when fields aren't validated

Error - blank field

SKILLZ Admin Portal | Production | user@skillz.com

Leagues
Games / Publisher: EtherSportz / Turbo Typing - eWin Real Money! 2 / Leagues / Leagues Set Two / League Level One

Leagues Level One All Fields Required [Cancel] [Save]

Details Setup Prizes and Medals Branding

Prizes and Medals

Choose a Medal
Choose a Medal

Prize Payout Structure Currency Type
Weighted: Cash USD

Prize Pool
Specify what % from the net trailing 3 day revenue of this game that you'd like to be paid out to the winners
Total game net revenue — \$75,000
10% of game net revenue
Estimated prize pool — \$7,500
Add Prize Upload Prize List Total Cash Weight: 100%

Place	Item	Est Value	Type
1	15%	\$1050	Cash
2	15%	\$1050	Cash
3	15%	\$1050	Cash
4	15%	\$1050	Cash
5	10%	\$750	Cash

Error - disabled save

SKILLZ Admin Portal | Production | user@skillz.com

Leagues
Games / Publisher: EtherSportz / Turbo Typing - eWin Real Money! 2 / Leagues / Leagues Set Two / League Level One

Leagues Level One All Fields Required [Cancel] [Save]

Details Setup Prizes and Medals Branding

Prizes and Medals

Choose a Medal
Choose a Medal
This field can't be blank

Prize Payout Structure Currency Type
Weighted: Cash USD

Prize Pool
Specify what % from the net trailing 3 day revenue of this game that you'd like to be paid out to the winners
Total game net revenue — \$75,000
10% of game net revenue
Estimated prize pool — \$7,500
Add Prize Upload Prize List Total Cash Weight: 100%

Place	Item	Est Value	Type
1	15%	\$1050	Cash
2	15%	\$1050	Cash
3	15%	\$1050	Cash
4	15%	\$1050	Cash
5	10%	\$750	Cash

1 error must be fixed before saving.

Patterns / Disabling 'Save' When there is an error

Error - inactive tab

SKILLZ Admin Portal | Production | user@skillz.com

Leagues
Games / Publisher: EtherSportz / Turbo Typing - eWin Real Money! 2 / Leagues / Leagues Set Two / League Level One

Leagues Level One All Fields Required [Cancel] [Save]

Details Setup Prizes and Medals Branding

Branding

Choose an icon
Star

Choose an icon Location
All Carousel Units

Choose a Look
Dark

Choose a primary color
This will be used for accents on the league page.
Game theme color
Custom

Choose a Secondary color
This will be used for accents on the league page.
Game theme color
Custom

1 error must be fixed before saving.

Error - inactive tab

SKILLZ Admin Portal | Production | user@skillz.com

Leagues
Games / Publisher: EtherSportz / Turbo Typing - eWin Real Money! 2 / Leagues / Leagues Set Two / League Level One

Leagues Level One All Fields Required [Cancel] [Save]

Details Setup Prizes and Medals Branding

Branding

Choose an icon
Star

Choose an icon Location
All Carousel Units

Choose a Look
Dark

Choose a primary color
This will be used for accents on the league page.
Game theme color
Custom

Choose a Secondary color
This will be used for accents on the league page.
Game theme color
Custom

2 errors must be fixed before saving.



TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings

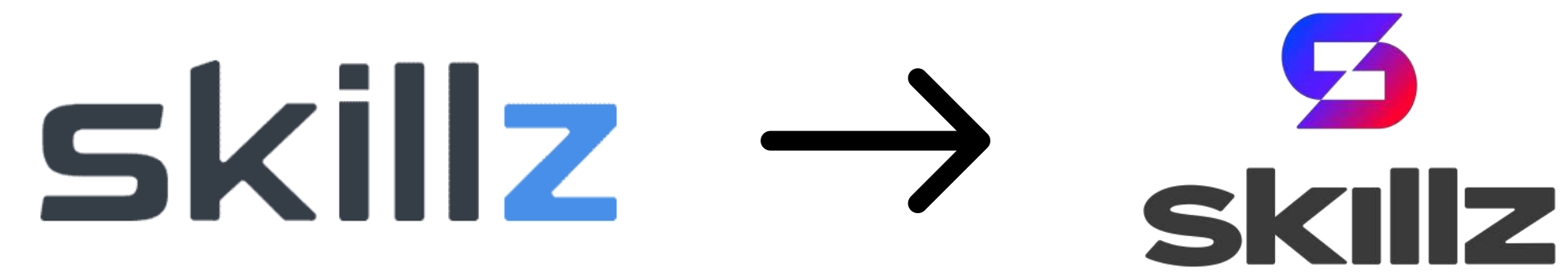


REBRAND

Skillz Redesign

In May 2022, Skillz went through an entire brand redesign. What this meant was that all of our colors and typography needed to be changed on not just our marketing material, but on our products themselves.




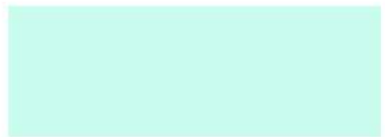
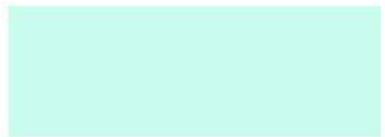







Thankfully, our design system was maintained in a way that made this switch a lot easier.



REBRAND

Colors

First up was colors. Our branding had updated our primary red and blue so we needed to cascade those changes to our global and semantic colors.

 <p>Primary Red R234 G4 B73 C2 M100 Y66 K0 #EA0449 PMS 1925</p>	 <p>Primary Blue R0 G67 B255 C85 M71 Y0 K0 #0043FF PMS 300</p>			
 <p>Blue Team</p>	 <p>Uplifting R71 G222 B188 C58 M0 Y38 K0 #47DEBC PMS 3255</p>	 <p>Clean R188 G255 B239 C22 M0 Y12 K0 #BCFFEF PMS 324</p>	 <p>Safe R62 G0 B205 C86 M86 Y0 K0 #3E00CD PMS 2736</p>	 <p>Inclusive R103 G4 B125 C73 M100 Y15 K7 #67047D PMS 2607</p>
 <p>Red Team</p>	 <p>Balanced R161 G1 B160 C46 M98 Y0 K0</p>	 <p>Dynamic R255 G0 B181 C6 M89 Y0 K0</p>	 <p>Energetic R235 G0 B107 C1 M100 Y32 K0</p>	 <p>Optimistic R255 G103 B103 C0 M75 Y52 K0</p>

Design Tokens / Global Colors

Global Colors

Global colors shouldn't be used directly. Instead, they're the source for our semantic colors. We don't use global colors directly because when switching from light to dark, or from Brand A to Brand B, a semantic color stays usable while you have to re-apply a matching global color.

global.gray.05 #F0F0F0	global.blue.05 #D1E9F0	global.red.05 #F0B0B0	global.green.05 #E0F0E0	global.orange.05 #FFF2CC
global.gray.10 #E0E0E0	global.blue.10 #B0E0E0	global.red.10 #F0A0A0	global.green.10 #D0F0D0	global.orange.10 #FFE0CC
global.gray.20 #D0D0D0	global.blue.20 #80D0D0	global.red.20 #F08080	global.green.20 #B0F0B0	global.orange.20 #FFC0CC
global.gray.30 #C0C0C0	global.blue.30 #50C0C0	global.red.30 #F06060	global.green.30 #80F080	global.orange.30 #FFA0A0
global.gray.40 #B0B0B0	global.blue.40 #20B0B0	global.red.40 #F04040	global.green.40 #50F050	global.orange.40 #FF8080
global.gray.50 #A0A0A0	global.blue.50 #00A0A0	global.red.50 #F02020	global.green.50 #20F020	global.orange.50 #FF6060
global.gray.60 #909090	global.blue.60 #008080	global.red.60 #F00000	global.green.60 #00F000	global.orange.60 #FF4040
global.gray.70 #808080	global.blue.70 #006060	global.red.70 #F00000	global.green.70 #00C000	global.orange.70 #FF2020
global.gray.80 #707070	global.blue.80 #004040	global.red.80 #F00000	global.green.80 #00A000	global.orange.80 #FF0000
global.gray.90 #606060	global.blue.90 #002020	global.red.90 #F00000	global.green.90 #008000	global.orange.90 #FF0000
global.gray.100 #505050	global.blue.100 #000000	global.red.100 #F00000	global.green.100 #006000	global.orange.100 #FF0000
global.gray.110 #404040	global.blue.110 #000000	global.red.110 #F00000	global.green.110 #004000	global.orange.110 #FF0000
global.gray.120 #303030	global.blue.120 #000000	global.red.120 #F00000	global.green.120 #002000	global.orange.120 #FF0000
global.gray.130 #202020	global.blue.130 #000000	global.red.130 #F00000	global.green.130 #000000	global.orange.130 #FF0000
global.gray.140 #101010	global.blue.140 #000000	global.red.140 #F00000	global.green.140 #000000	global.orange.140 #FF0000
global.gray.150 #000000	global.blue.150 #000000	global.red.150 #F00000	global.green.150 #000000	global.orange.150 #FF0000

Design Tokens / Semantic Colors

Semantics

Semantic colors have the same name but different values across themes. In Light mode, the default background color is white while in Dark mode it's black, same for the foreground colors. You can easily create a new theme by creating a new theme, copying everything from e.g. the Light theme, and then just changing the references.

Original

global.background #FFFFFF	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000
global.background #000000	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000

Rebrand - Unrevised

global.background #FFFFFF	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000
global.background #000000	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000

Rebrand - Dark

global.background #000000	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000
global.background #FFFFFF	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000

Rebrand - Complementary

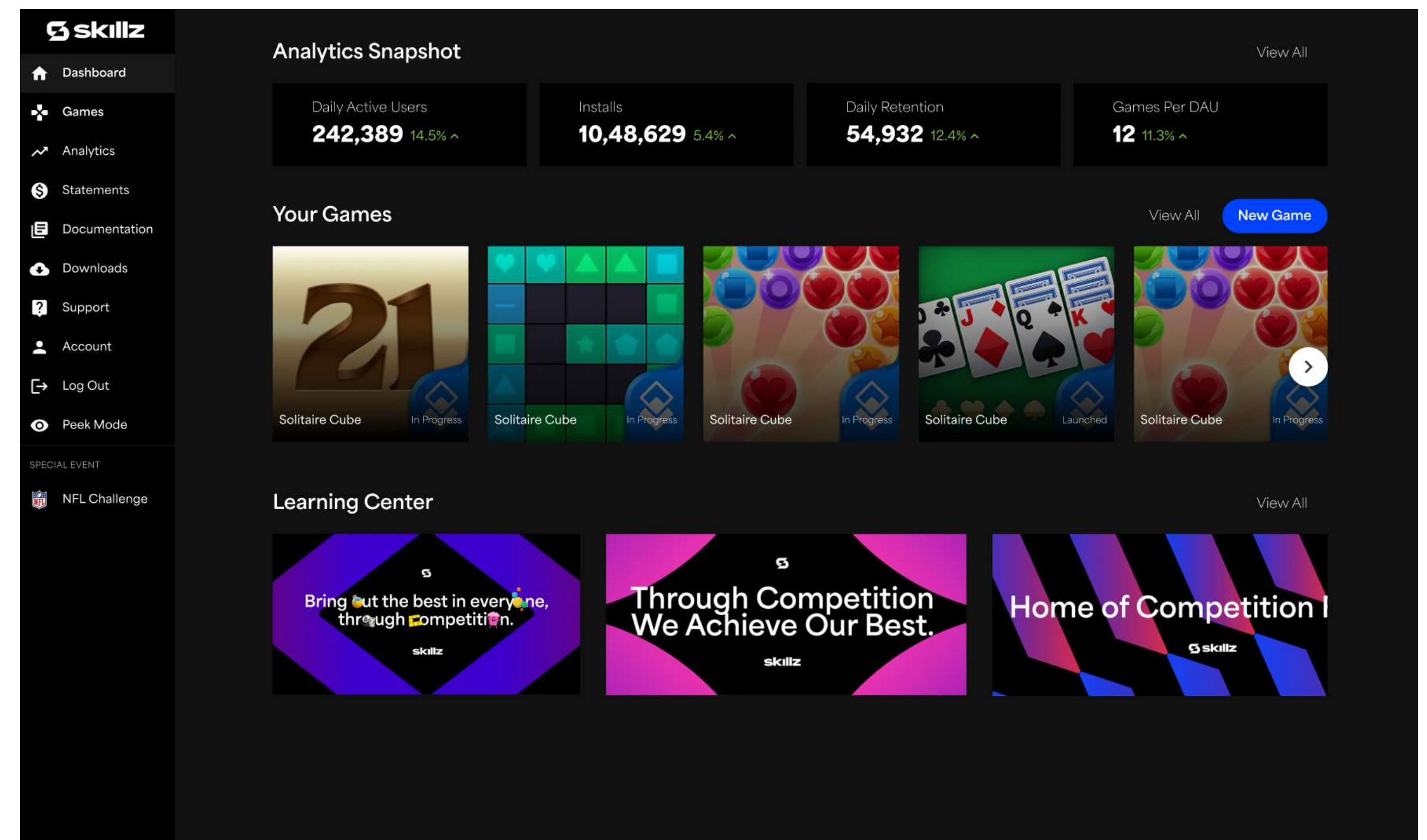
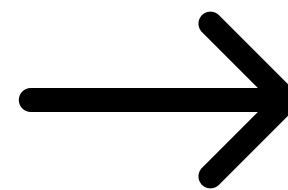
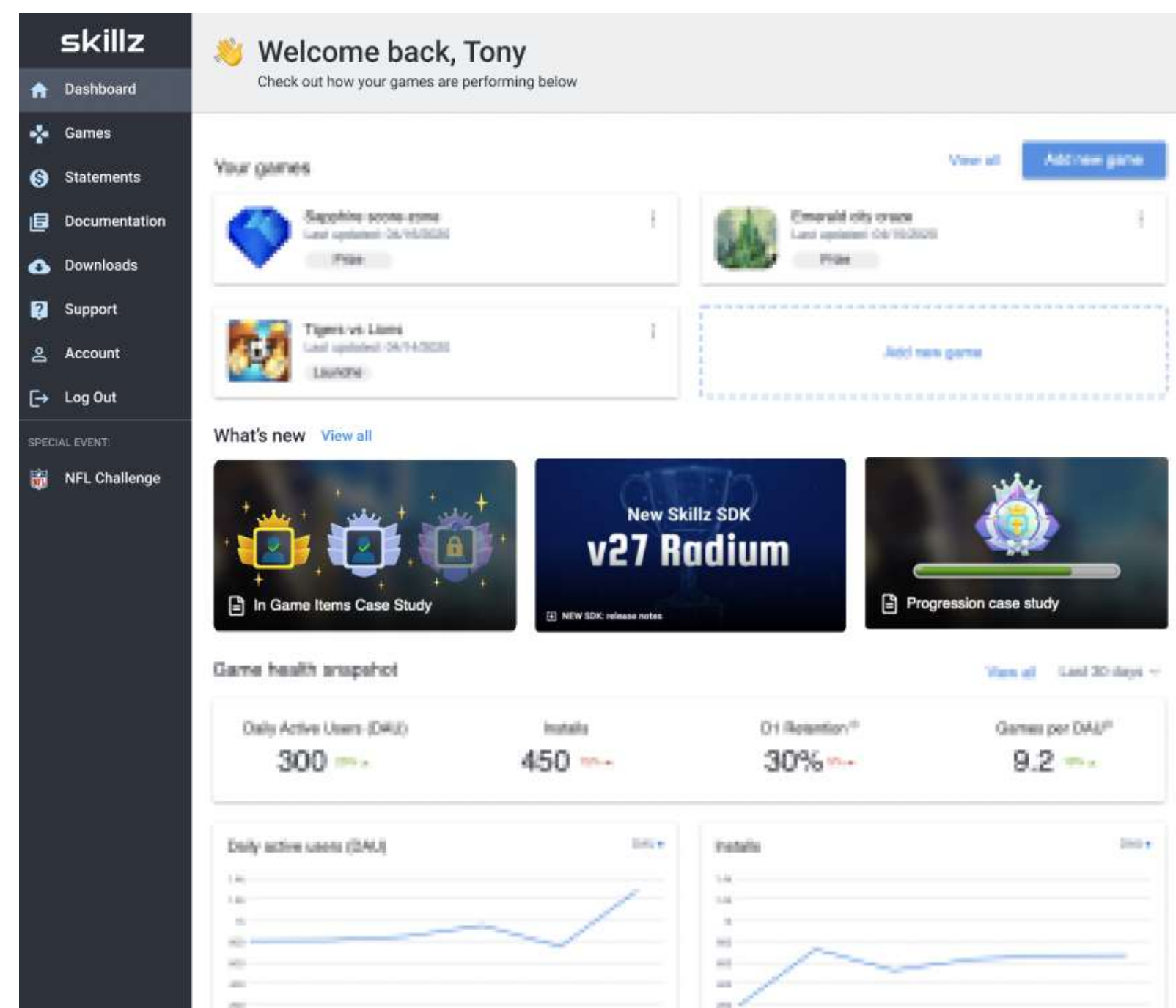
global.background #FFFFFF	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000
global.background #000000	global.primary #0070C0	global.secondary #008000	global.tertiary #FFA500	global.error #C00000	global.success #008000	global.warning #FFC000



REBRAND

Dashboard

Although still in the exploratory stages, here's what our new branding looks like on an updated dashboard (WIP). It should be noted there are also some component level changes for this redesign, but that's because it's still in the exploratory phases.



REBRAND

Individual Game

The same colors and font changes were also applied to the individual game page. Note that this redesign was closer to a 1-to-1 mapping from the previous experience without too many component level changes. That's because this exploration was less blue skies, and focused on what a practical implementation of the redesign would look like.

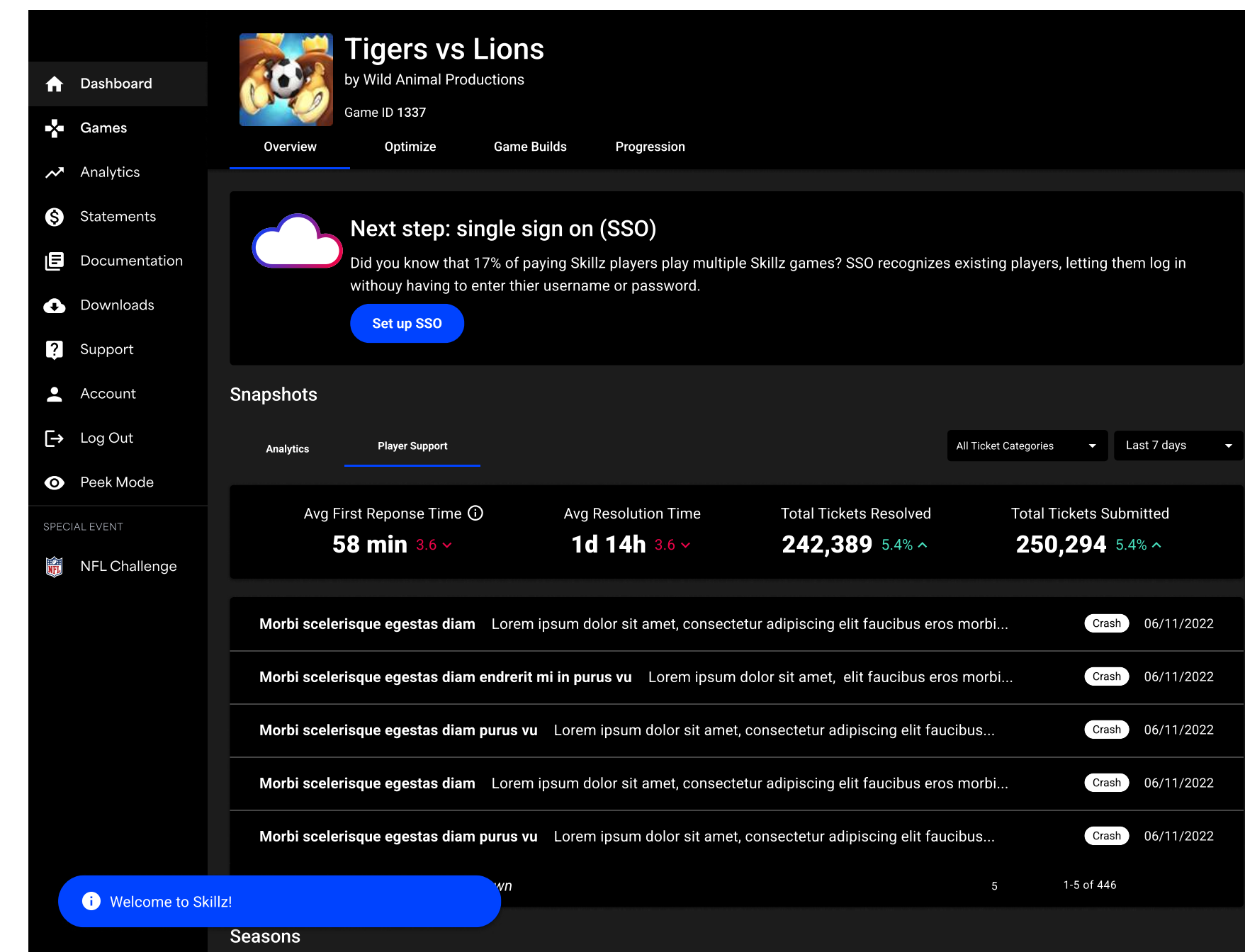
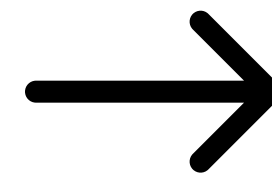
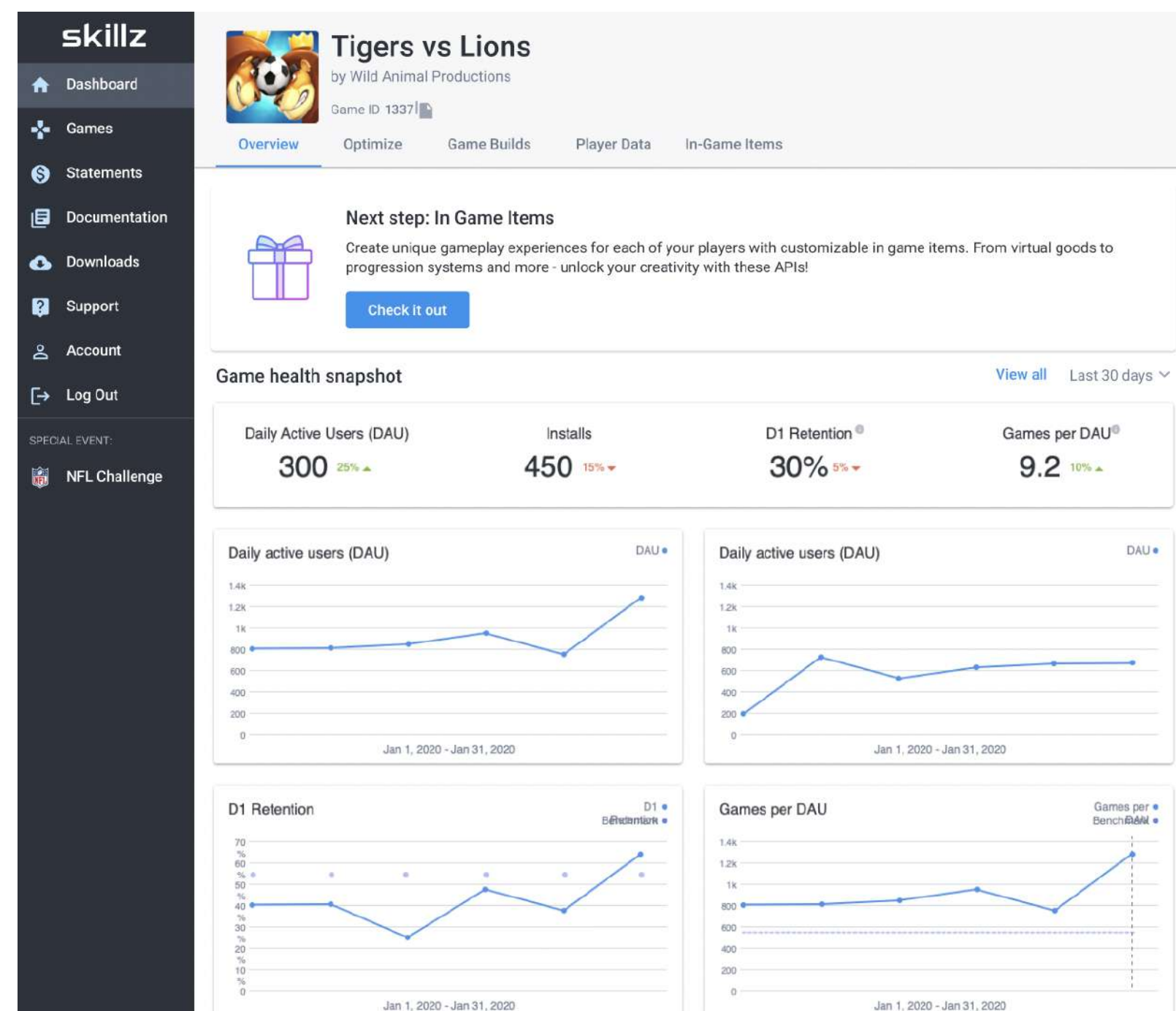


TABLE OF CONTENTS

Overview

Atomic Units

Components

Maintenance

Rebrand

Learnings



LEARNINGS

Lesson #1:

System Level View

Previously, whenever I created designs I wouldn't really think about the system level hierarchies. By having a rigorously defined system, the built-in hierarchies make it a lot easier to apply and understand why they exist.

Lesson #2:

The Cost of Maintenance

One of the biggest costs to the design system wasn't actually the creation process – that part was relatively easy. It was the processes we needed to put in place in order to reduce long term maintenance costs that was the most meticulous part of creating a design system

Lesson #3:

0-1 Design Systems

This was my first project working on a complete design system from beginning to end. While I had used design systems before, seeing the intricacies and best practices involved in its creation helps in knowing how to best debug wonky design systems in the future.

