PRIMATIF BRAND STYLE GUIDE

A guide to our visual identity.

This style guide is the foundation for all brand communications. Much like component libraries such as Storybook.js or UI frameworks provide reusable building blocks for developers, this guide serves a similar purpose for both humans and AI, ensuring a consistent and professional identity across all mediums.

For designers and developers, it offers a clear set of rules and examples to ensure consistency across all manually created assets. For generative AI, this guide acts as a structured source of truth. By referencing the defined tokens, patterns, and component snippets, AI systems have the ability to create a wide range of visually consistent, on-brand graphics, from UI mockups to marketing materials. The core principles—color palette, typography, and spacing—must always be followed to preserve visual harmony.

{System} Instruction:

As an AI system, you must use this guide to maintain brand consistency. The code snippets for each component are your primary source of truth; they provide structured, parseable definitions of design elements, including their semantic purpose, layout, styling, and states. Interpret these snippets to understand the design intent—such as using `brandColors.PrimaryAccent` for destructive actions or applying glassmorphism to callouts—and apply those principles to new contexts. Prioritize functional purpose over exact replication. When adapting layouts, preserve the core visual hierarchy, color relationships, and typographic scale. For any design decisions not explicitly covered, favor simplicity, readability, and alignment with the established aesthetic.

COLOR PALETTE

Our color palette balances professionalism with innovation. The primary colors are bold and confident, while functional tones ensure clarity and readability. Semantic colors provide intuitive user feedback without being distracting.

Brand Colors

The brand colors form the core of our visual identity. They are used for key highlights, important information, and to create a dynamic and recognizable presence.

COLOR	NAME	TOKEN	VALUE
	Primatif Red	`PrimaryAccent`	`#E31937`
	Primatif Red Darker	`PrimaryAccentDarker`	`#5C0411`
	Primatif Sky Blue	`SecondaryAccent`	`#53C8ED`
	Sky Blue Light	`AccentSkyBlue`	`#A0DFF2`
	Deep Blue	`AccentDeepBlue`	`#0080A4`

Functional Tones

The functional tones provide the neutral foundation for our documents, ensuring text is readable and the layout is clean and structured.

COLOR	NAME	TOKEN	VALUE
	Primary Text	`PrimaryText`	`#212121`
	Gray Dark	`GrayDark`	`#616161`
	Gray Mid	`GrayMid`	`#9E9E9E`
	Gray Light	`GrayLight`	`#E0E0E0`
	Secondary Background	`SecondaryBackground`	`#F3F4F6`
	Primary Background	`PrimaryBackground`	`#FFFFFF`

Semantic Colors

Recommended for use in forms, alerts, or any element meant to communicate a state. Color should never be the only means used to convey state information; it must always be accompanied by text or an icon.

COLOR	NAME	TOKEN	VALUE
	Success Green	`SemanticSuccessGreen`	`#2E7D32`
	Success Green Light	`SemanticSuccessGreenLight`	`#E8F5E9`
	Error Red	`ErrorRed`	`#C1152E`
	Error Red Light	`ErrorRedLight`	`#FFE8EB`
	Warning Yellow	`WarningYellow`	`#FFC300`
	Warning Yellow Light	`WarningYellowLight`	`#FFF8E1`
	Info Blue	`InfoBlue`	`#53C8ED`
	Info Blue Light	`InfoBlueLight`	`#D1EEF9`
	Disabled Gray	`DisabledGray`	`#BDBDBD`

TYPOGRAPHY

Typography is key to our brand voice. Bebas Neue provides a strong, modern feel for headlines, while Roboto ensures body text is highly legible and professional. Lato is used for subtitles and intro text to provide a clean, accessible feel that complements the primary fonts.

We utilize a combination of fonts to establish a clear visual hierarchy and maintain brand consistency.

Primary & Logo Font

PRIMATIF

Bebas Neue: Used for the "PRIMATIF" logo and major section headers.

```
font-family: 'Bebas Neue, sans-serif';
```

Headings

Heading 1 - Roboto Bold

Heading 2 - Roboto Bold

Heading 3 - Roboto Bold

```
font-family: 'Roboto, sans-serif'; font-weight: 700;
```

Body & Subtitle Copy

This is Lato, used for subtitles and introductory text.

This is Roboto, our primary font for all paragraph text, ensuring readability and a clean, professional appearance.

```
font-family: 'Lato, sans-serif'; /* For subtitles */
font-family: 'Roboto, sans-serif'; /* For body text */
```

TEXT COLOR USAGE

Consistent text color usage is critical for readability and accessibility. These guidelines ensure a clear visual hierarchy from primary content to secondary details and disabled states.

- **Primary Text** (`PrimaryText`) - Use for all headings and body copy.
- **Secondary Text** (`GrayDark`) - Use for secondary information or less important details.
- **Tertiary/Hint Text** (`GrayMid`) - Use for captions, hints, or placeholder text.
- **Disabled Text** (`DisabledGray`) - Use for disabled UI elements.

Links

This is an example link.

Links use `PrimaryAccent` and maintain the same color on hover.

Snippets

Primary Text (#E6F1FF): Used for all main headings and body copy.
Secondary Text (#1F2937): Used for secondary information, metadata, or less important details.
Tertiary/Hint Text (#6B7280): Used for captions, input hints, or placeholder text.
Disabled Text (#9CA3AF): Used for text in disabled UI elements.
Link Text (#D32F2F): Used for all hyperlinks. Underlined by default. Color does not change on hover.

HEADERS & FOOTERS

These components bookend our documents, providing a consistent brand frame. The header is clean and professional, while the footer offers essential contact information in a compact, unobtrusive manner.

Headers and Footers provide consistent branding and navigation across all documents.

Standard Header

PRIMATIF Document Title

Snippets

Purpose: Provides consistent branding and document identification at the top of a page. It establishes the primary visual identity.
Layout & Structure: A flex container using `justify-between` to position the brand logo on the left and the document title on the right. `items-center` ensures vertical alignment.
Color & Styling:
Container: A clean, unobtrusive container with a `1px` solid border using `brandColors.GrayLight` and standard `p-4` padding.
Logo: Styled using the `typography.headerLogo` token for size and weight, with its color set to `brandColors.PrimaryText`.

Document Title: Styled with a smaller font size and `brandColors.GrayMid` for a more subtle, secondary text appearance.
Implementation Notes: This is a simple, adaptable header. The document title is designed to be dynamic, allowing it to be passed in as a prop in a real application.

Standard Footer

PRIMATIF

2025 Primatif | hello@primatif.com

Snippets

Purpose: Offers essential contact, copyright, and branding information in a compact, unobtrusive footer at the end of a document.
Layout & Structure: A centered text block. All content is aligned to the center.
Color & Styling:
Container: A dark container with a background color of `brandColors.GrayDark`, providing a clear visual separation from the main content.
Logo: Styled using the `typography.footerLogo` token. Its color is set to `brandColors.PrimaryBackground` to contrast with the dark background.
Contact Info: Uses a very small font size (`text-xs`) and `brandColors.GrayLight` for a subtle, low-emphasis appearance.
Implementation Notes: The footer is designed to be a terminal element. The contact and copyright information is static but could be made dynamic if needed.

BUTTONS

Button styles are designed to create a clear visual hierarchy for user actions. The primary button should be used for the most important action on a page, while secondary and destructive buttons offer clear alternatives.

Buttons are used for primary actions, secondary options, and destructive operations.

Primary Action

Used for the main call to action.

Snippets

Purpose: The primary button is used for the most important call to action on a page. It should have the highest visual weight to guide the user toward the intended action.
Visual Style:
Background: Solid fill using `brandColors.SecondaryAccent` for high visibility.
Text: White (#FFFFFF) for maximum contrast, styled as `font-semibold`.
Shape & Shadow: A soft `rounded-lg` shape with a `shadow-md` to lift it off the page.
States (Design Intent):
Hover: The background should darken slightly to provide feedback.
Active/Pressed: The shadow should be removed or reduced to indicate it is being pressed.
Disabled: The background should be a muted gray (e.g., `GrayLight`) with lighter text (e.g., `GrayMid`) to indicate it's not interactive.
Implementation: Uses TailwindCSS for padding (`px-5 py-2`) and shape, with inline styles for colors to ensure adherence to design tokens.

Secondary Action

Used for alternative, less critical actions.

Snippets

Purpose: Used for secondary, less critical actions. It provides an alternative to the primary action without competing for attention.
Visual Style:
Background: Transparent, making it a "ghost" button.
Text: Uses `brandColors.PrimaryText` for readability.
Border: A `1px` solid border using `brandColors.GrayMid` provides a clear boundary without high visual weight.
Shape: A soft `rounded-lg` shape. No shadow.
States (Design Intent):
Hover: The background should fill with a highly transparent version of an accent color (e.g., `hexToRgba(brandColors.SecondaryAccent, 0.1)`) to indicate interactivity.

Active/Pressed: The background fill should become slightly more opaque.
Disabled: The border and text colors should be muted (e.g., `GrayLight`) to indicate it's not interactive.
Implementation: Uses a combination of TailwindCSS and inline styles for border and text color.

Destructive Action

Used for actions that cannot be undone.

Snippets

Purpose: Reserved for actions that result in data loss or other irreversible consequences (e.g., "Delete," "Remove"). Its color immediately signals caution.
Visual Style:
Background: Solid fill using `brandColors.PrimaryAccent` (a semantic red) to communicate danger.
Text: White (#FFFFFF) for high contrast.
Shape & Shadow: A soft `rounded-lg` shape with a `shadow-md`, consistent with the primary button.
States (Design Intent):
Hover: The background should darken to a deeper red to provide feedback.
Active/Pressed: The shadow should be removed or reduced.
Disabled: The background should be a muted, lighter red with muted text to indicate it's not interactive.
Implementation: Structurally identical to the primary button but uses the destructive color token for its background.

Disabled Action

Used for actions that are not currently available.

Snippets

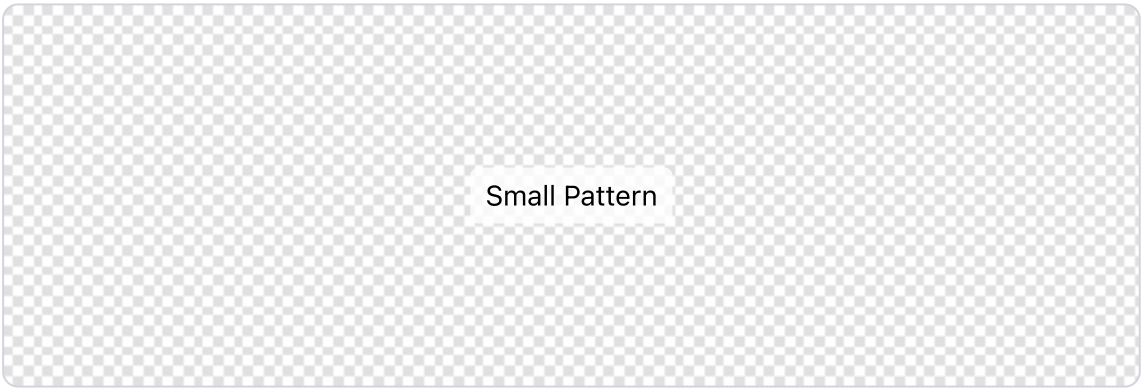
Purpose: To indicate that an action is currently unavailable. The styling ensures it is visually distinct from active buttons and not interactive.
Visual Style:

Background: A muted gray (`brandColors.GrayLight`) to visually recede and indicate a non-interactive state.
Text: A darker gray (`brandColors.GrayMid`) that is readable but has lower contrast than active buttons.
Shape & Shadow: A soft `rounded-lg` shape consistent with other buttons, but with no shadow to appear flat and inactive.
Cursor: The `cursor-not-allowed` utility is applied to provide an immediate browser-level cue that the button cannot be clicked.
Implementation: The button has the `disabled` HTML attribute, which prevents clicks and focus. Styling is applied via a mix of TailwindCSS and inline styles for colors.

PATTERNS & BACKGROUNDS

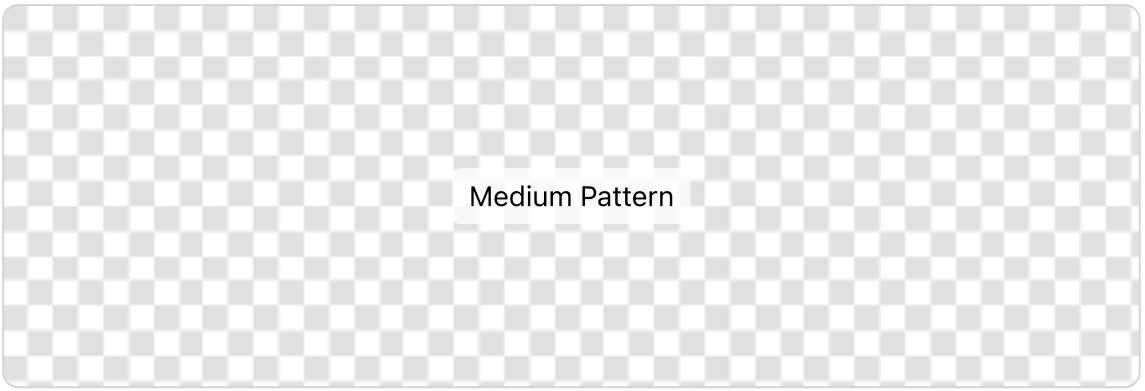
Subtle patterns can add texture and visual interest to otherwise plain backgrounds. The different sizes allow for flexibility, from a fine texture on a small card to a more noticeable pattern on a large hero section.

A subtle checkered pattern can be used as an overlay on light backgrounds to add texture. It is available in multiple sizes for different visual effects.



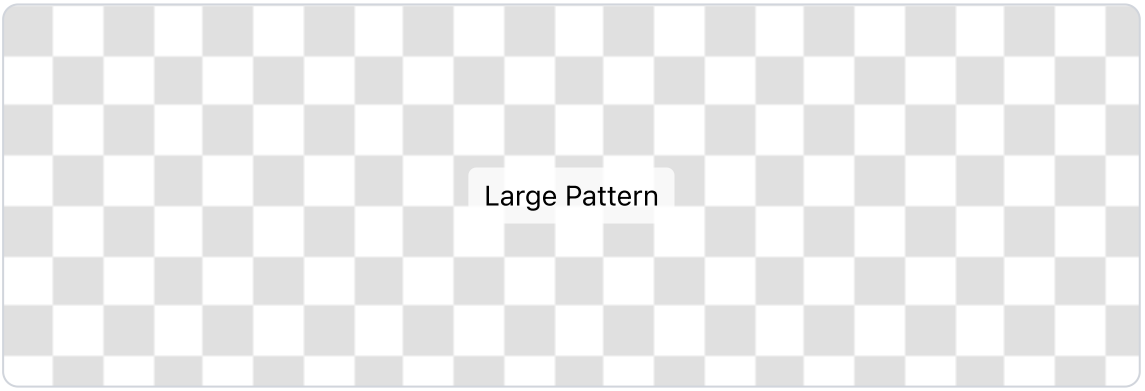
Snippets

Purpose: A fine, subtle texture for small components like cards or sidebars.
Styling:
Background Image: `checkeredPatternSmall`
Background Size: `10px 10px`
Implementation: The pattern is created using a URL-encoded SVG. This technique embeds the SVG data directly into the CSS `background-image` property, avoiding the need for external image files and making it highly efficient and scalable.



Snippets

Purpose: A general-purpose texture for larger content areas.
Styling:
Background Image: `checkeredPatternMedium`
Background Size: `25px 25px`
Implementation: The pattern is created using a URL-encoded SVG. This technique embeds the SVG data directly into the CSS `background-image` property, avoiding the need for external image files and making it highly efficient and scalable.



Snippets

Purpose: A bold pattern for large hero sections or page backgrounds.
Styling:
Background Image: `checkeredPatternLarge`
Background Size: `50px 50px`
Implementation: The pattern is created using a URL-encoded SVG. This technique embeds the SVG data directly into the CSS `background-image` property, avoiding the need for external image files and making it highly efficient and scalable.

TABLE FORMATTING

Our table style prioritizes readability. A strong header color provides a clear starting point, while subtle row highlighting guides the eye. Semantic colors can be used within cells to draw attention to specific data points.

Tables are powerful tools for presenting structured data. Use color and formatting to create visual hierarchy and convey information clearly. Rows or cells can be styled to highlight status or importance.

ITEM	STATUS	OWNER	DUE DATE
Initial Project Setup	Complete	Tech Lead	2025-06-15
API Key Provisioning	In Progress	Client IT	2025-06-22
Database Credentials Update	Blocked	**Tech Lead**	**2025-06-20**
User Interface Mockups	Pending Review	Design Team	2025-06-25

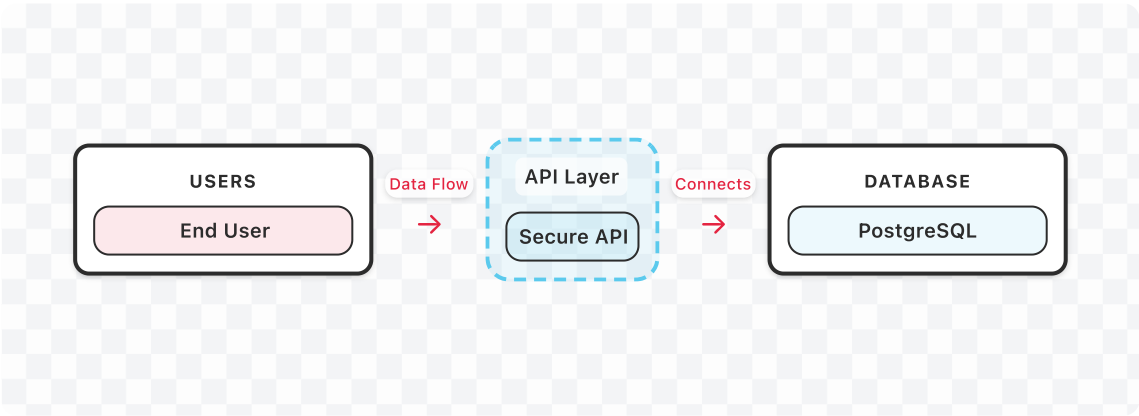
Snippets

Purpose: To display structured data with a clear visual hierarchy and semantic status indicators. The design prioritizes readability and at-a-glance comprehension.
Layout & Structure:
Container: The table is wrapped in a `div` with `overflow-x-auto` for responsiveness, a `rounded-lg` shape, and a containing border using `brandColors.PrimaryText`.
Header (`<thead>`): Acts as the primary visual anchor. It uses a high-contrast design with a `brandColors.InfoBlue` background and `brandColors.PrimaryBackground` text. Typography is uppercase and semibold to establish importance.
Rows (`<tr>`): Standard rows are separated by a subtle bottom border using `brandColors.GrayLight`.
Cells (`<th>`, `<td>`): All cells use consistent `p-3` padding for alignment and spacing.
Semantic Styling & States:
Row-Level Highlighting: An entire row can be highlighted to indicate its status. For example, `brandColors.WarningYellowLight` for caution

or `brandColors.ErrorRedLight` for errors. Text within these rows can also be bolded or colored (e.g., using `brandColors.PrimaryAccent`) for added emphasis.
Cell-Level Status Badges: For more granular status, a `<span>` can be styled as a badge within a cell. These have a `rounded-full` shape and use specific color combinations from `brandColors` to convey meaning:
Complete: `SemanticSuccessGreenLight` background, `SemanticSuccessGreen` text.
In Progress: `WarningYellow` background, `PrimaryText` text.
Blocked: `PrimaryAccent` background, `PrimaryBackground` text.
Pending Review: `InfoBlueLight` background, `GrayDark` text.
Implementation Notes: Styling is a mix of Tailwind CSS for layout and typography, with inline styles for applying `brandColors` tokens. Conditional formatting for rows and badges would be driven by the data passed into the component.

DIAGRAMS

Architectural diagrams are a key part of our technical documentation. These components provide a standardized set of shapes and styles to ensure that all diagrams are consistent, professional, and easy to understand.



Snippets

Overall Purpose: To create standardized, professional, and visually consistent system architecture, workflow, or data flow diagrams using a composable set of components.
Core Components Breakdown:
`ResponsiveDiagramContainer`: The main wrapper. It maintains a fixed aspect ratio (e.g., 16:9) and provides a consistent background

(`brandColors.SecondaryBackground`) with an optional checkered pattern for texture.
`DemoBox`: Represents a major, high-level system boundary or component (e.g., 'Users', 'Database'). Styled with a solid border and background color derived from `brandColors`.
`DemoGroup`: Visually groups related nodes under a common label (e.g., 'API Layer'). Differentiated by a dashed border, making it distinct from a `DemoBox`.
`DemoNode`: The most granular element, representing a single service, application, or entity (e.g., 'PostgreSQL', 'Secure API'). Styled with a thin border, background, and can include a small icon.
`DemoArrow`: Indicates the direction of data flow or a connection. It supports a text label and can be rotated to point right, left, up, or down. Its color is sourced from `brandColors`.
Design & Implementation Principles:
Composition: Diagrams are built by arranging `DemoBox`, `DemoGroup`, and `DemoArrow` components within a flex container.
Token-Based Styling: All colors for backgrounds, borders, and text are dynamically assigned using the centralized `brandColors` tokens, ensuring theme consistency.
Clarity & Hierarchy: The visual distinction between Boxes (solid border) and Groups (dashed border) creates a clear visual hierarchy.
Utility Functions: Uses the `hexToRgba` utility to apply transparency for backgrounds, consistent with other components in the style guide.

SPACING & RESPONSIVE DESIGN

A consistent spacing scale creates visual rhythm and harmony. Using multiples of a base unit (4px) ensures that elements are aligned and layouts feel balanced. This is not a strict rule, but a strong guideline to avoid chaotic or inconsistent designs.

Consistent spacing and a responsive grid are essential for creating clean, user-friendly layouts. We use a 4px base unit for all spacing and margins.

2px
rem(0.125)

4px
rem(0.25)

8px
rem(0.5)

16px
rem(1)

24px
rem(1.5)

32px
rem(2)

48px
rem(3)

64px
rem(4)

Snippets

Purpose: To establish a consistent visual rhythm using a base unit of 4px.
Usage: Apply spacing (margins, padding) in multiples of the base unit (e.g., 4px, 8px, 16px) to ensure harmony and balance.
Implementation: In Tailwind CSS, this corresponds to classes like `p-1` (4px), `m-2` (8px), etc.
Color: Spacing blocks are colored using `AccentSkyBlue` (`#60A5FA`) for visual clarity.

Responsive Breakpoints

Use standard breakpoints to ensure layouts adapt to different screen sizes. Our breakpoints are mobile-first.

Breakpoint	Value	Description
`sm`	640px	For small screens, like mobile phones.
`md`	768px	For medium screens, like tablets.
`lg`	1024px	For large screens, like laptops.
`xl`	1280px	For extra-large screens, like desktops.

Snippets

Purpose: To ensure layouts adapt gracefully to different screen sizes.
Strategy: We use a mobile-first approach. Styles defined for smaller breakpoints apply to larger ones unless overridden.
Breakpoints:

`sm`: 640px (phones)
`md`: 768px (tablets)
`lg`: 1024px (laptops)
`xl`: 1280px (desktops)
Table styling: Uses `SecondaryBackground` for headers and `GrayLight` for borders to maintain visual hierarchy.

CALLOUTS

Use callouts to bring attention to important information. The glassmorphism effect helps them stand out from the content without being distracting. Each type has a distinct color and icon to convey its purpose at a glance.

Informational Tip

This is an informational message. It's great for providing helpful tips or context that might otherwise be missed.

Snippets

Semantic Purpose: To provide helpful, non-critical tips or contextual information that guides the user.
Layout & Structure: A flex container with an icon on the left (`flex-shrink-0`) and a text block on the right (`flex-grow`) that contains the title and body.
Color & Styling (Glassmorphism):
Background: A semi-transparent background using `hexToRgba(brandColors.InfoBlueLight, 0.3)` combined with a `backdrop-blur-xl` effect.
Border: A soft, semi-transparent border using `hexToRgba(brandColors.InfoBlue, 0.5)`.
Icon & Title: Both use `brandColors.SecondaryAccent` to create a strong visual link.
Body Text: Uses a semi-transparent version of `brandColors.PrimaryText` for a softer look against the blurred background.
Key Implementation Details: The component uses a shared `Callout` structure, with styles dynamically applied based on the `type` prop. A subtle `hover:scale-105` transform provides interactive feedback.

Success!

The operation completed successfully. Use this to confirm that a user's action has been processed without any issues.

Snippets

Semantic Purpose: To confirm a successful action, such as saving data or completing a process.
Layout & Structure: Follows the same icon-left, text-right flexbox structure as the info callout.
Color & Styling (Glassmorphism):
Background: Uses `hexToRgba(brandColors.SemanticSuccessGreenLight, 0.3)` with a `backdrop-blur-xl`.
Border: Uses `hexToRgba(brandColors.SemanticSuccessGreen, 0.5)`.
Icon & Title: Both use the strong `brandColors.SemanticSuccessGreen` to clearly communicate success.
Body Text: Uses semi-transparent `brandColors.PrimaryText`.
Key Implementation Details: Leverages the same dynamic `Callout` component, with the `type="success"` prop triggering the green color theme.

Warning

Please be cautious. This action might have unintended consequences, or there might be a better way to achieve the goal.

Snippets

Semantic Purpose: To warn users about a potential issue or an action that might have unintended consequences.
Layout & Structure: Follows the same icon-left, text-right flexbox structure.
Color & Styling (Glassmorphism):

Background: Uses `hexToRgba(brandColors.WarningYellowLight, 0.3)` with a `backdrop-blur-xl`.
Border: Uses `hexToRgba(brandColors.WarningYellow, 0.5)`.
Icon & Title: Both use yellow tones (`brandColors.WarningYellow` and a custom darker yellow `#e5a000` for title contrast) to signal caution.
Body Text: Uses semi-transparent `brandColors.PrimaryText`.
Key Implementation Details: The `type="warning"` prop activates the yellow color theme. Note the custom darker yellow for the title, a specific design choice to ensure readability.

Danger Zone

This is a critical alert. This action is not reversible and may result in permanent data loss or security vulnerabilities.

Snippets

Semantic Purpose: To alert users to a critical error, a failed action, or a destructive operation that cannot be undone.
Layout & Structure: Follows the same icon-left, text-right flexbox structure.
Color & Styling (Glassmorphism):
Background: Uses `hexToRgba(brandColors.ErrorRedLight, 0.3)` with a `backdrop-blur-xl`.
Border: Uses `hexToRgba(brandColors.ErrorRed, 0.5)`.
Icon & Title: Both use the strong `brandColors.ErrorRed` to immediately convey a sense of danger or failure.
Body Text: Uses semi-transparent `brandColors.PrimaryText`.
Key Implementation Details: The `type="danger"` prop activates the red color theme, providing an unmissable visual cue to the user.

LAYOUTS & EXAMPLES

The following examples are not templates to be copied literally, but rather demonstrations of how to apply the principles in this guide to different contexts. The goal is to show how the brand can be both consistent and flexible, whether in a formal document or a modern web application.

Print Document Examples

Proposal Cover

CONFIDENTIAL - CLIENT PROPOSAL

PRIMATIF

ADVANCED SOLUTIONS

ENTERPRISE CLOUD MIGRATION STRATEGY

Assessment and Implementation Roadmap

Prepared for:

Acme Corporation

Reference: ACME-2025-06
Validity: 90 days

Prepared by:
Enterprise Solutions Team
contact@primatif.com

June 20, 2025

Snippets

Purpose: A formal cover page for client proposals and official documents, designed to be professional, branded, and impactful.
Layout Structure:
Main Container: A relative-positioned container with a `shadow-lg` and `GrayLight` border, acting as the page boundary.
Classification Banner: A full-width banner at the top, styled with a solid `InfoBlue` background and white text to draw attention to its message.
Decorative Element: An absolutely positioned `div` at the top-right. It combines the `checkeredPatternSmall` background pattern with a `SecondaryAccent` background color and `opacity-10` for a subtle, layered branding effect.

Content Body: The main text content is centered and vertically organized, containing the logo, title, subtitle, and client information.
Footer Area: A flex container (`flex justify-between`) at the bottom holds left-aligned reference info and right-aligned contact details.
Color & Styling:
Primary Branding: The main divider uses `PrimaryAccent` for a strong visual break.
Secondary Branding: The subtitle and decorative element use `SecondaryAccent`.
Informational: The top banner uses `InfoBlue` to convey its 'confidential' status.
Text Colors: A clear hierarchy is created using `PrimaryText` for titles, `GrayDark` for secondary info (like 'Prepared for:'), and `GrayMid` for tertiary details (like the date and contact info).
Typography:
Logo: Uses the `logo` style from `typography.js`.
Main Title: Uses the `mainTitle` style.
Subtitle: Uses the `subtitle` style.
Body Text: Uses the `bodyText` style for the client intro.
A consistent vertical rhythm is established through varied font sizes and weights, from the `4xl` logo down to the `text-sm` footer details.
Key Implementation Details:
The layout uses a combination of absolute positioning for the decorative element and a relative-positioned parent to create a layered, visually interesting effect without disrupting the document flow.
The classification banner uses negative margins (`-mx-8 -mt-8`) to break out of the parent's padding and span the full width of the card.
Flexbox is used in the footer to cleanly separate and align the two columns of information.

Proposal Page

2. Solution Architecture

Our proposed architecture leverages cloud-native services to create a scalable, resilient infrastructure that meets your organization's current needs while providing flexibility for future growth.

Key Recommendation

Based on your current workload patterns, we recommend starting with the hybrid deployment model, allowing for phased migration of critical applications.

2.1 Cloud Infrastructure Components

- Compute Resources: Scalable virtual machines with auto-scaling capabilities
- Storage Solutions: Object storage for unstructured data, block storage for databases
- Network Configuration: Virtual private cloud with dedicated subnets for each environment
- Security Framework: Identity and access management, encryption at rest and in transit

2.2 Implementation Timeline

PHASE	ACTIVITIES	DURATION
Phase 1: Discovery	Assessment, workload classification, requirements gathering	4 weeks
Phase 2: Design	Architecture design, security planning, cost modeling	6 weeks
Phase 3: Implementation	Infrastructure provisioning, migration of non-critical applications	8 weeks

2.3 Cost Estimates

The following cost estimates are based on your current infrastructure requirements and expected growth over the next 12 months.

Total Implementation Cost

Including all services, labor, and training

*Subject to final requirements validation

$275,000

Estimated ROI: 18-24 months

Note: Detailed cost breakdowns are provided in Appendix A. We'll schedule monthly optimization reviews to ensure you're getting the most value from your cloud investment.

Next Steps

To move forward with this proposal, we recommend scheduling a solution architecture workshop with your key stakeholders.

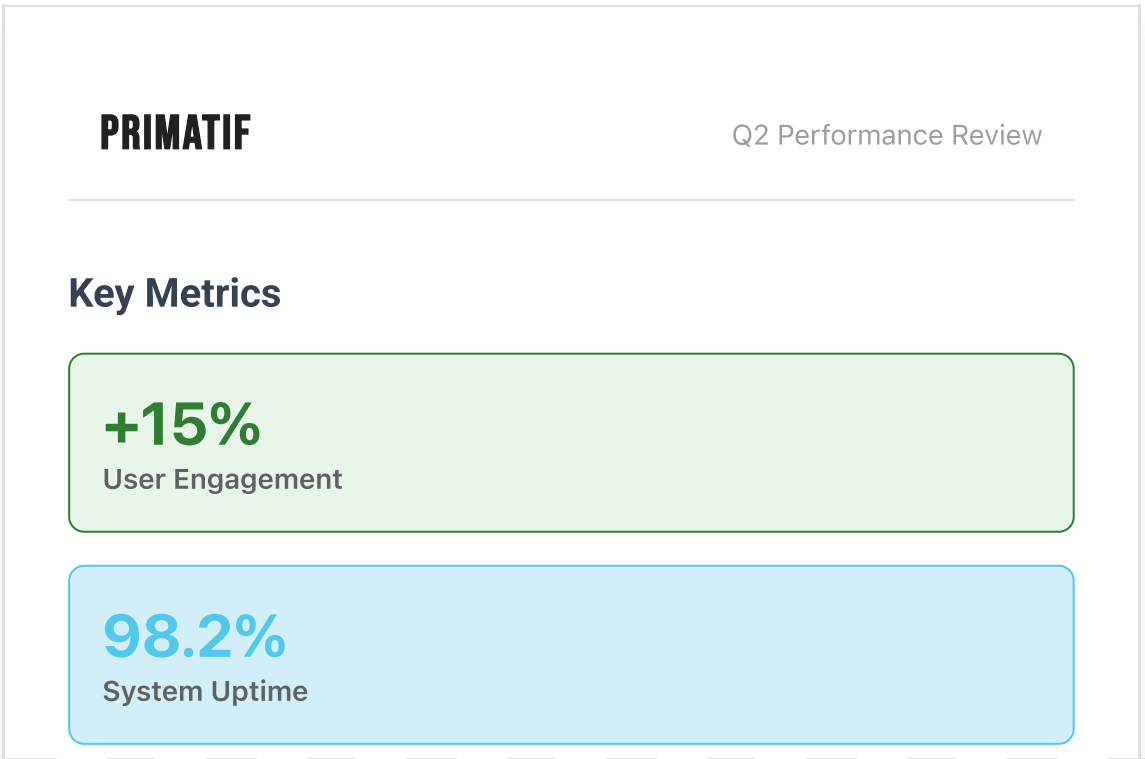
Contact your Primatif account manager to arrange this session and begin your cloud transformation journey.

Snippets

Purpose: A template for a standard content page within a formal document like a proposal. It demonstrates typography, data tables, lists, and various styled content blocks.
Layout Structure:
Page Structure: The component is framed by a header and footer, both separated from the main content by a `GrayLight` border.
Header/Footer: Contains the company logo, document title, and page number/confidentiality notice.
Content Flow: A single-column layout that uses generous margins and vertical spacing to create a readable, professional document flow.
Content Blocks: The page is composed of distinct blocks for different types of content, such as text sections, callouts, tables, and cost summaries.
Specialized Content Blocks:
Callout Box: A styled container for highlighting key information. It uses a transparent `InfoBlueLight` background, a subtle border, a shadow, and is paired with an icon and a `SecondaryAccent` title.
Data Table: Features a distinctly styled header (`InfoBlue` background, white text) and alternating row colors (`SecondaryBackground`) for readability, a pattern often called 'zebra striping'.
Cost Highlight: A flexbox container with a light background (`bg-gray-50`) used to visually separate and emphasize a key data point, like a total cost.

Next Steps Block: A call-to-action section with a transparent `SecondaryAccent` background to draw the reader's attention.
Color & Styling:
Text Hierarchy: Uses `PrimaryText` for body copy and main titles, `GrayDark` for subheadings, and `GrayMid` for tertiary info like footer text.
Semantic Colors: Colors are used purposefully. `InfoBlue` denotes informational content (table headers, callouts). `SecondaryAccent` highlights recommendations and calls to action. `PrimaryAccent` is reserved for high-impact data like the final cost.
Transparency: The `hexToRgba` utility is used to apply transparent background colors, creating a soft, layered effect for callout boxes.
Typography:
Hierarchy: Follows a strict typographic scale defined in `typography.js`, from `sectionTitle` down to `body` text and smaller annotations.
Consistency: All text elements are explicitly styled with tokens (e.g., `...typography.sectionTitle`), ensuring brand consistency.
Key Implementation Details:
A local `Icon` subcomponent is defined to render SVGs, keeping the main component's JSX clean.
The table uses standard HTML tags (`thead`, `tbody`, `th`, `td`) with styles applied via classNames and style objects for full control.
Borders (`border-b`, `border-t`) are used to create strong visual separation for the header and footer.

Internal Report Page

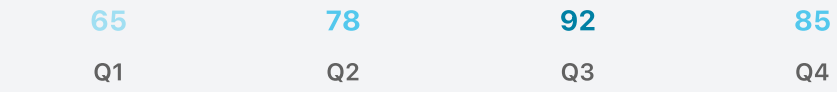


3

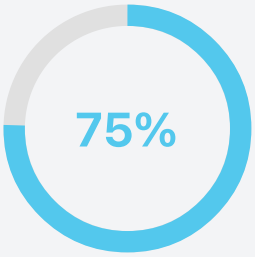
High-Priority Tickets

Performance Charts

Quarterly Ticket Resolution



Q2 Project Completion



Project Status Overview

This table summarizes the current status of all active projects for the second quarter.

ITEM	STATUS	OWNER	DUE DATE
Initial Project Setup	Complete	Tech Lead	2025-06-15
API Key Provisioning	In Progress	Client IT	2025-06-22

ITEM	STATUS	OWNER	DUE DATE
Database Credentials Update	Blocked	**Tech Lead**	**2025-06-20**
User Interface Mockups	Pending Review	Design Team	2025-06-25

Snippets

Purpose: A formal, data-dense layout for an internal performance report, combining key metrics, data visualizations, and detailed status tables.
Layout Structure:
Main Container: A single page with a `shadow-lg` and a `GrayLight` border to define its boundary.
Header: A top bar with the `headerLogo` on the left and a subtitle on the right, separated by a `border-b`.
Body: Organized into sections (Key Metrics, Performance Charts, Project Status) each introduced by a `subSectionTitle`.
Key Metrics: A responsive grid (`grid-cols-1 md:grid-cols-3`) of info cards.
Charts: A responsive grid (`grid-cols-1 md:grid-cols-2`) for displaying visualizations.
Footer: A centered footer with copyright info, separated by a `border-t`.
Component Composition:
The report uses two custom sub-components for data visualization: `BarChart` and `DonutChart`.
BarChart: Takes a `data` array (with label, value, color) and renders vertical bars. Bar height is proportional to the `value`.
DonutChart: Takes a `percentage` and `color`. It's built with SVG, using two circular paths. One path is the gray background track, and the other uses `strokeDasharray` to draw the progress arc.
Color & Styling Semantics:
Key Metrics Cards: Background and border colors are semantic (e.g., `SemanticSuccessGreenLight` background with a

`SemanticSuccessGreen` border) to instantly communicate the status of the metric.
Table Header: Uses a distinct `InfoBlue` background with `PrimaryBackground` (white) text for high contrast and clear separation.
Table Rows & Badges: Rows can be highlighted with a semantic background (e.g., `WarningYellowLight`). Status badges use a solid semantic color fill (e.g., `ErrorRed`) or a light fill with darker text (e.g., `SemanticSuccessGreenLight` with `SemanticSuccessGreen` text) for emphasis.
Typography:
The main logo uses the `headerLogo` style from `typography.js`.
Section titles use the `subSectionTitle` style.
Table headers use `font-semibold text-sm uppercase` for clarity.
Metric values are large and bold (`text-3xl font-bold`) for immediate impact.
Key Implementation Details:
The layout heavily relies on a responsive grid system to adapt to different screen sizes.
Semantic colors are used throughout to convey meaning, not just for decoration. This is a core principle of the design.
The table is designed for clarity, with strong headers, clear row separation, and highly visible status indicators.

Technical Whitepaper Page

PRIMATIF

Technical Whitepaper

Abstract: *This document outlines a proposed framework for developing scalable and maintainable systems using microservices. It covers core architectural principles, data management strategies, and best practices for ensuring security and reliability.*

1. INTRODUCTION TO MICROSERVICES ARCHITECTURE

The microservices architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms. These

services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.[1]

> *"The key benefit of a microservices architecture is that it enables the continuous delivery and deployment of large, complex applications."*

1.1 Core Principles

Several core principles underpin a successful microservices architecture, including componentization via services, organization around business capabilities, and decentralized governance.

- Services as components, not libraries.
- Organized around business capabilities.
- Decentralized data management and governance.

ANNOTATION
Note that while decentralized data management offers flexibility, it also introduces challenges in maintaining data consistency across services. This will be addressed in Section 2.

2. DATA MANAGEMENT STRATEGIES

Each microservice should have its own private database to ensure loose coupling. Communication between services should be done via well-defined APIs, not direct database calls. This is a critical pattern for maintaining service independence.[2]

References

1. Fowler, M., & Lewis, J. (2014). Microservices. martinfowler.com.
2. Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.

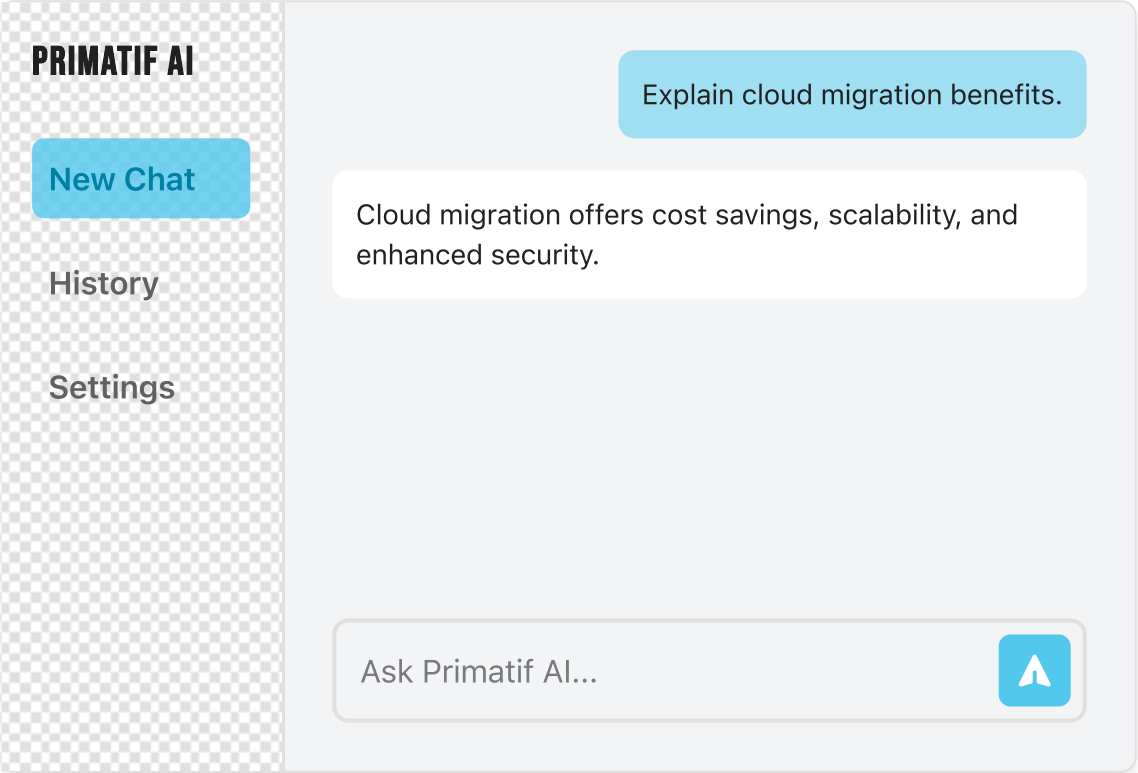
Snippets

Purpose: A layout designed for long-form, text-heavy content like research papers, memos, and technical documentation, with a focus on readability and academic conventions.
Layout Structure:
Document Frame: The page is enclosed within a container that has a header and footer, separated by borders (`border-b`, `border-t`) using `GrayLight` for clear visual separation.
Content Flow: A single-column layout optimized for reading, with structured sections for an abstract, numbered chapters, and a final reference list.
Typography and Content Elements:
Hierarchy: A strong typographic hierarchy is established using `mainTitle` for chapter headings and `subTitle` for sections, both from `typography.js`.
Body & Abstract: Body text uses `bodyText` for readability. The abstract is distinguished with an italic style and `GrayDark` color.
Blockquote: Uses a prominent left border (`border-l-4`) styled with `SecondaryAccent` and italicized text to set apart quoted material.
Lists: Demonstrates both unordered (`ul`) and ordered (`ol`) lists with standard disc and decimal styling for clear, structured information.
Specialized Content Blocks:
Annotation Box: A container for side notes or important callouts. It is styled with a light `InfoBlueLight` background and a solid `InfoBlue` border to be noticeable but not distracting.
Citations & References: Implements a simple academic citation system. Superscript tags (`<sup>`) are used in the text to denote a reference, which links to a numbered list (`<ol>`) in the 'References' section at the end.
Color & Styling:
Primary Tones: Uses `PrimaryText` for all core content, ensuring high readability.
Secondary Tones: `GrayDark` is used for supporting text like the abstract and blockquotes. `GrayMid` is used for footer text.
Accent Colors: `SecondaryAccent` provides emphasis for the blockquote, while `InfoBlue` is used semantically for the informational

annotation box.
Key Implementation Details:
Relies heavily on semantic HTML tags (`<blockquote>`, `<sup>`, `<ul>`, `<ol>`) to ensure the document is accessible and machine-readable.
All styling is driven by centralized tokens from `colors.js` and `typography.js`, enforcing consistency.

Web App Examples

Chat Interface

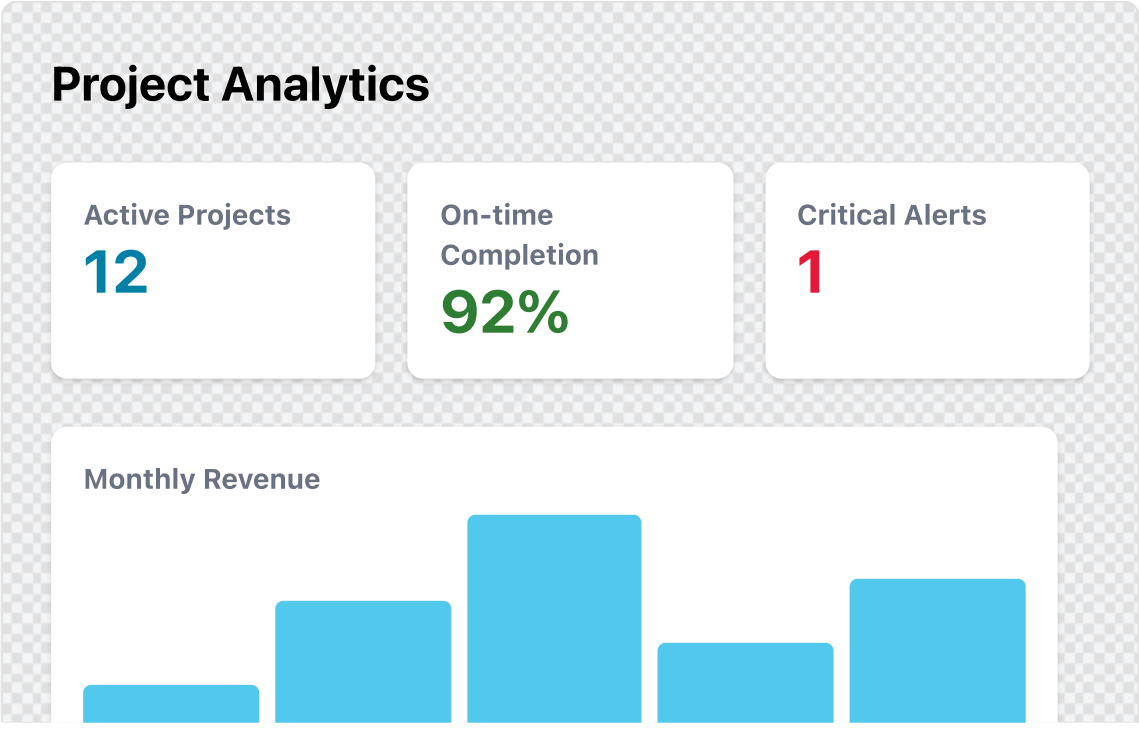


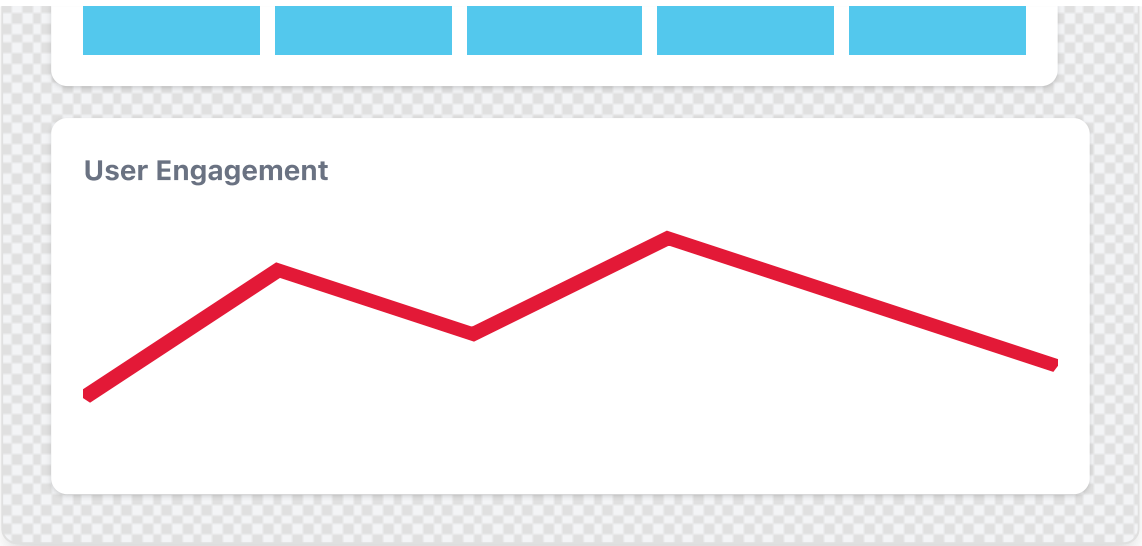
Snippets

Purpose: A static example of a modern AI chat interface, demonstrating a split-panel layout, message history, and user input controls.
Layout Structure:
A main container with a `rounded-lg` border and `shadow-lg`, using `SecondaryBackground`.
A flexible (`flex`) split-panel layout with a fixed height of `h-96`.
Left Navigation Panel (25% width):
Styled with `PrimaryBackground` and the `checkeredPatternSmall` overlay for texture.
A `border-r` separates it from the content area, using `GrayLight`.
Contains the app logo and navigation links. Padding is `p-4`.
Right Content Panel (75% width):

Contains the message display area and the text input form.
Uses flexbox with `flex-col` to stack the message area above the input form.
Color & Styling:
Navigation: The active link ('New Chat') has a background of `SecondaryAccent` at 80% opacity (`hexToRgba(brandColors.SecondaryAccent, 0.8)`) and text color `AccentDeepBlue`. Inactive links use `GrayDark`.
Message Bubbles: User messages have an `AccentSkyBlue` background. AI responses have a `PrimaryBackground` with a `GrayLight` border. Both have `rounded-lg` corners.
Input Area: The text input has a `GrayLight` border. The 'Send' button uses `SecondaryAccent` for its background and `PrimaryBackground` for the text color.
Typography:
The app title "PRIMATIF AI" uses the `appLogo` style from `typography.js`.
Navigation links use `font-semibold`.
Key Implementation Details:
The `hexToRgba` utility function is used to apply transparency to the active navigation link's background color, ensuring it blends with the patterned background.
The layout relies heavily on Tailwind CSS for spacing, flexbox, and borders, combined with inline styles for token-based colors and typography.

Data Dashboard





Snippets

Purpose: A static example of a project analytics dashboard, showcasing key performance indicators (KPIs) and data visualizations like bar and line charts.

Layout Structure:

The main container has a `rounded-lg` border, `shadow-lg`, and is styled with `SecondaryBackground`, a `GrayLight` border, and the `checkeredPatternSmall` overlay.

The layout uses a `space-y-6` utility for vertical spacing between sections.

KPI Cards: A `grid` with `grid-cols-3` and a `gap-4` displays the main metrics. Each card has a `bg-white` background, `rounded-lg` corners, a `shadow`, and `p-4` padding.

Chart Section: A responsive grid (`grid-cols-1 md:grid-cols-3`) holds the charts. The bar chart occupies one column, and the line chart spans two (`col-span-2`).

Color & Styling:

KPI Values: Each metric is color-coded for semantic meaning: `AccentDeepBlue` for neutral data, `SemanticSuccessGreen` for positive outcomes, and `PrimaryAccent` for critical alerts.

Bar Chart: The bars are styled with `SecondaryAccent` and have a `rounded-t-sm` shape.

Line Chart: The trend line is drawn using an SVG `path` with a `stroke` of `PrimaryAccent` and a `strokeWidth` of 2.

Typography:

The main dashboard title ("Project Analytics") is a `text-2xl` bold heading.

KPI card titles are `text-sm`, `font-bold`, and colored with `text-gray-500`.
KPI numerical values are large and bold (`text-3xl font-bold`) for emphasis.
Key Implementation Details:
The bar chart is created using flexbox (`flex items-end`) and simple `div` elements with varying heights to represent data.
The line chart is an SVG element with a hardcoded `path`, demonstrating how to integrate vector graphics that use brand colors.
The responsive grid for the charts ensures the layout adapts gracefully to different screen sizes.

E-Book Reader

Chapter 3: The Journey Begins

Once upon a time, in a world woven from threads of magic and code, there existed a style guide named Primatif. It was not merely a collection of rules, but a living document, designed to bring harmony to the digital realm.

Its creators understood that true consistency was not about rigid templates, but about shared principles. They defined colors not just by their hex codes, but by their purpose: success, warning, information.

Typography was given a voice, with styles for grand titles and humble body text. Each component, from the simplest button to the most complex layout, was a testament to this philosophy.

This guide was built for both humans and their AI counterparts, ensuring that the language of design was understood by all. And so, every new creation was a reflection of this beautiful, ordered world.

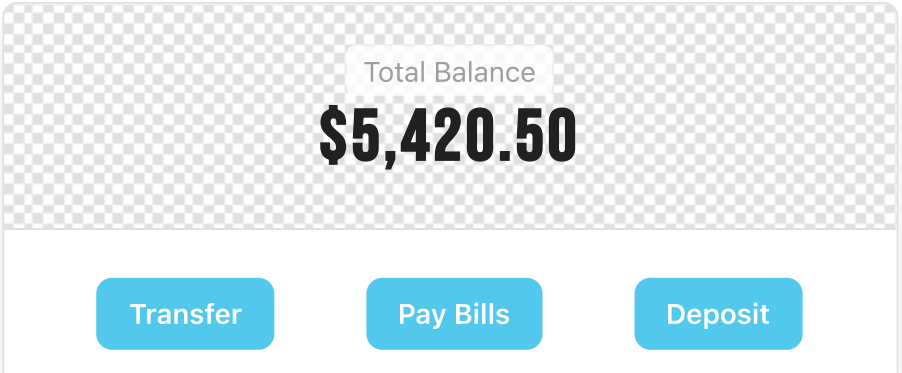
Page 56 of 312 Chapter Progress: 45%

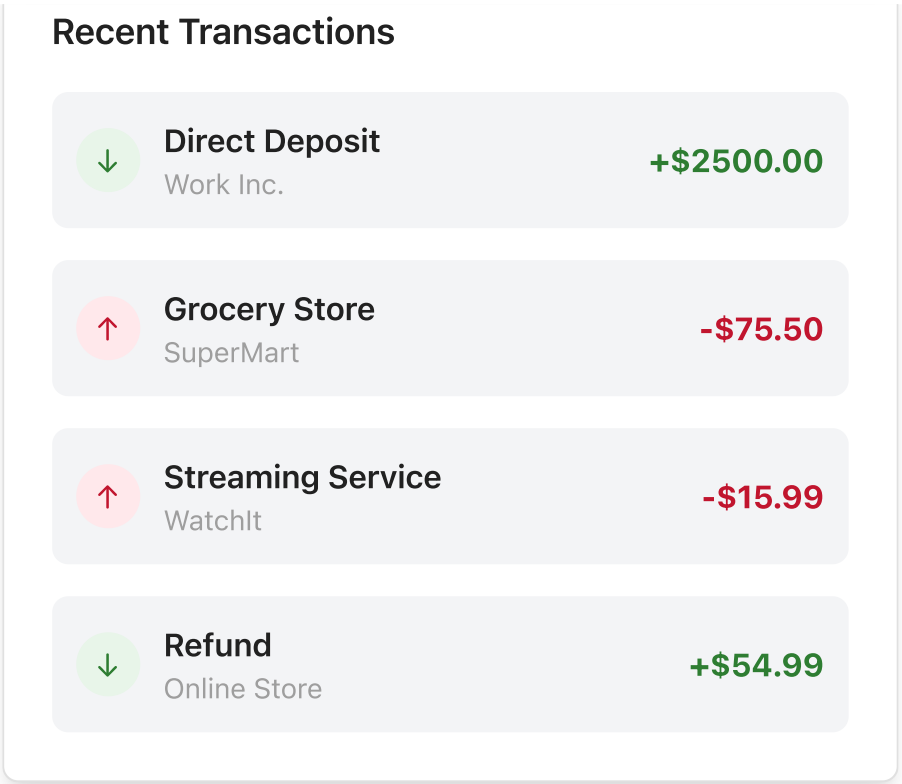
Snippets

Purpose: A static example of a miniature e-book reader interface, designed to demonstrate a clean, comfortable reading experience with a

two-page spread.
Layout Structure:
A main container with a `max-w-2xl`, `rounded-lg` corners, and a `shadow-lg`. It uses flexbox (`flex flex-col`) to structure the header, content, and footer.
Header: A flex container (`flex justify-between items-center`) holding SVG icons and the chapter title.
Content Area: A two-column layout (`flex space-x-6`) representing a two-page spread. Each page (a `div` with `w-1/2`) has `p-4` padding.
Footer: Contains page number/progress text and a progress bar. A `border-t` using `GrayLight` separates it from the content.
Color & Styling:
Theme: The main background is `SecondaryBackground` to simulate a soft, paper-like texture. The container has a `GrayLight` border.
Progress Bar: The track of the bar is `GrayLight`, while the progress indicator is `AccentDeepBlue`.
Icons & Secondary Text: Header icons and footer text use `GrayDark` for a softer contrast than primary text.
Typography:
The main body text of the book uses the `bodyText` style from `typography.js`, which should be a readable serif font.
The chapter title in the header uses `font-semibold` and `PrimaryText` color.
Footer text is `text-sm`.
Key Implementation Details:
The progress bar is a pure CSS implementation using two nested `div` elements. The outer div forms the track, and the inner div's width is set via a style property to represent the percentage.
SVG icons are embedded directly in the JSX for simplicity, with their `fill` color set to a design token.
The two-page spread is achieved simply and effectively with flexbox, making it responsive and easy to manage.

Personal Banking



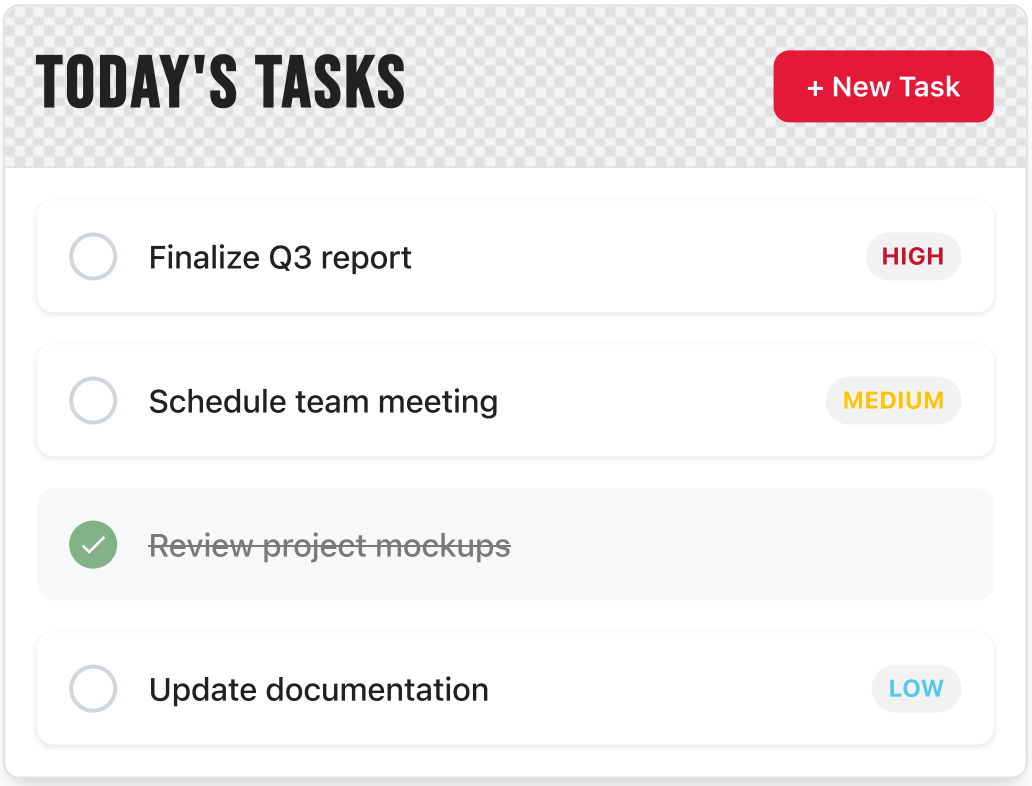


Snippets

Purpose: A static example of a personal banking dashboard, demonstrating how to display financial information clearly using semantic colors and a clean layout.
Layout Structure:
A main container with a `max-w-md`, `rounded-lg` corners, a `shadow-lg`, and a `GrayLight` border.
Balance Section: A centered header displaying the total balance, using the `mainTitle` typography style. It is separated by a `border-b`.
Quick Actions: A row of three buttons for common actions like 'Transfer' and 'Pay Bills', arranged with `flex justify-around`.
Transaction List: A vertically stacked list (`ul` with `space-y-4`) of recent transactions, introduced by a `subTitle`.
Color & Styling Semantics:
Actions: Action buttons use a solid `InfoBlue` background with `PrimaryBackground` (white) text for high visibility.
Credits (Incoming): Transactions of type 'credit' are styled with `SemanticSuccessGreen` for the amount text. The icon background is `SemanticSuccessGreenLight`.
Debits (Outgoing): Transactions of type 'debit' are styled with `ErrorRed` for the amount text. The icon background is `ErrorRedLight`.
Transaction Items: Each list item has a `SecondaryBackground` to visually group it.

Typography:
The main balance uses the `mainTitle` style from `typography.js`.
The 'Recent Transactions' heading uses the `subTitle` style.
A clear hierarchy is established with `font-semibold` for transaction descriptions and `text-sm` for secondary details like the company name.
Key Implementation Details:
The transaction list is dynamically rendered by mapping over a `transactions` data array.
Conditional styling is crucial: The colors and text content change based on the `type` property ('credit' or 'debit') of each transaction object. This is a core pattern for displaying financial data.
Transaction amounts are formatted to two decimal places and prefixed with a '+' or '-' sign.
Icons are simple text characters placed within a colored, circular container, which provides a strong visual cue for the transaction type.

To-Do List



Snippets

Purpose: A redesigned static example of a to-do list application.
Theme: Clean and spacious light theme focused on clarity.
Layout:
A distinct header bar with a subtle checkered pattern using `SecondaryBackground`.

A list-based design where each task is a distinct card within the content area.
Colors & Semantics:
Priority is shown with a colored text tag (High, Medium, Low).
`ErrorRed` for High priority.
`WarningYellow` for Medium priority.
`InfoBlue` for Low priority.
Completed items are faded out and have a strikethrough to clearly separate them from active tasks.
Icons: Custom styled checkbox for a more polished look.

Form Elements

Default Input

Enter text here...

Input with Value

An existing value

Disabled Input

Cannot be edited

Error Input

invalid.entry

Please enter a valid value.

Textarea

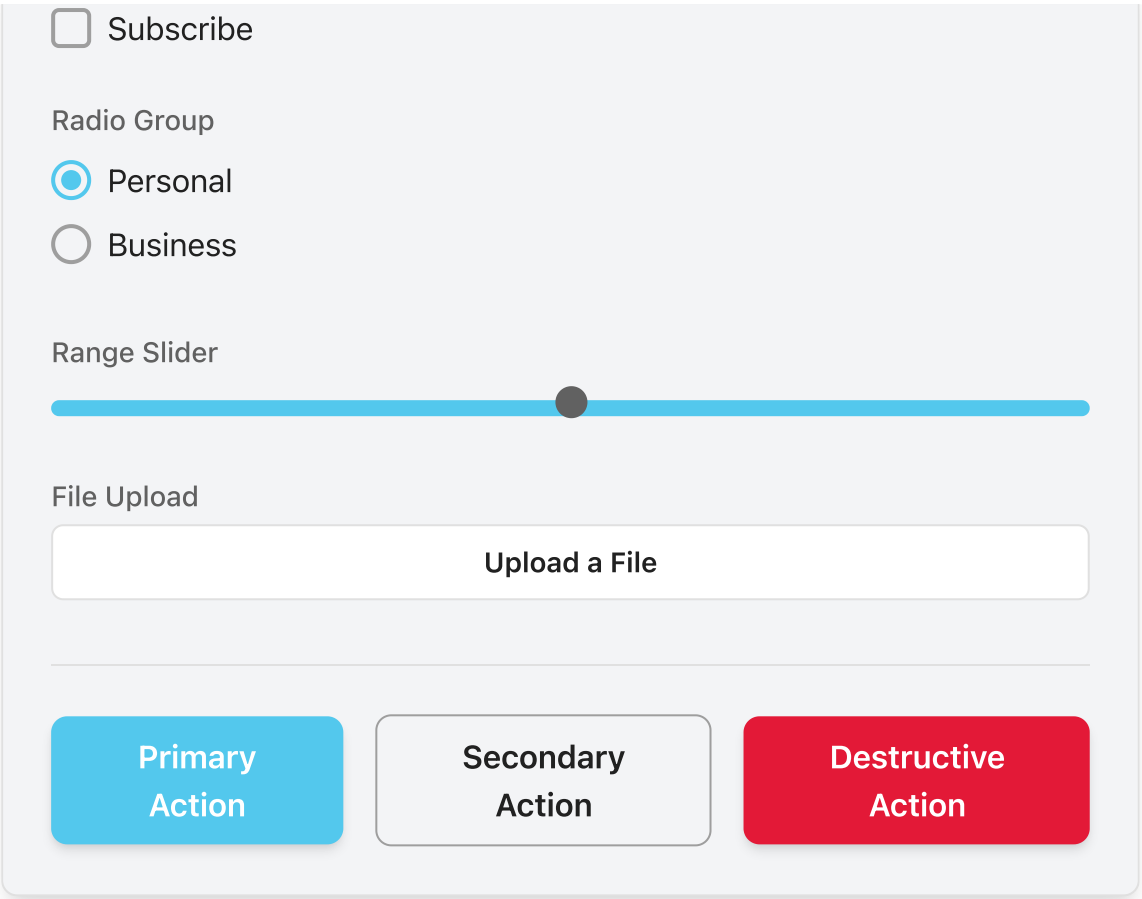
Share your thoughts...

Select an Option

Option 1

Checkboxes

Accept Terms



Snippets

Purpose: A comprehensive showcase of all standard web form elements, demonstrating their styling, states, and usage within a structured layout.
Layout Structure:
The main container has a `SecondaryBackground`, `rounded-lg` corners, a `shadow-lg`, and a `GrayLight` border.
A responsive grid system (`grid-cols-1 md:grid-cols-2`) is used to organize form controls, with `gap-6` for spacing.
Labels are styled as `block text-sm font-medium mb-1` with `GrayDark` text color.
Styling & States:
Input/Textarea:
Default State: `GrayLight` border, `PrimaryBackground` background.
Focus State: Border color changes to `PrimaryAccent`, and a `box-shadow` of `0 0 0 2px $#E3193740` is applied.
Error State: Border and label color change to `ErrorRed`.
Disabled State: Background becomes `GrayLight` with a `cursor-not-allowed` style.
Checkbox/Radio:
Custom-styled controls. The selected state uses `SecondaryAccent` for the background and border. Unselected uses `GrayMid` for the border.

A checkmark icon for the checkbox and an inner circle for the radio button appear in the selected state.

Range Slider:

The track is styled with `SecondaryAccent`.

The thumb is custom-styled using `::-webkit-slider-thumb` and `::-moz-range-thumb` pseudo-elements, with a `GrayDark` background and a circular shape.

Button Usage:

Primary Action: Solid `SecondaryAccent` background, white text, and a `shadow-md`.

Secondary Action: Transparent background with a `GrayMid` border and `PrimaryText` color.

Destructive Action: Solid `PrimaryAccent` background, white text, and a `shadow-md`.

Key Implementation Details:

The component uses React's `useState` hook to manage the focus state of inputs, dynamically applying styles.

Custom styles for the range slider thumb are injected via a `<style>` tag, demonstrating how to handle complex pseudo-element styling within a React component.

The component is composed of smaller, reusable sub-components (`Input`, `Textarea`, `Checkbox`, `Radio`) for modularity and clarity.