USER INTERFACE / USER EXPERIENCE DESIGN SPECIFICATIONS (UI/UX DS) : WEB

PROJECT	***PERKHIDMATAN* DESIGN, DEVELOPMENT, INSTALLATION, CONFIGURATION, TESTING, COMMISSIONING AND PROVISIONING OF SUPPORT AND MAINTENANCE FOR THE NEW AGIHAN SYSTEM (NAS) *UNTUK* LEMBAGA ZAKAT SELANGOR**
DATE	**02/05/2025**
VERSION	**1.0**
REF NO	**LZS/NAS/UI/UXDS**

Prepared by:

Data Science Sdn. Bhd

dataScience

DOCUMENT REVISION HISTORY

Prepared By:

Version	Date	Author	Changes
1.0	05/05/2025	Mohd Zahirul Iman Bin Mohammad Rafee	Draft

Reviewed By:

Version	Reviewer Name	Role
1.0	Rozita Binti Radzuan Norida Ayu Binti Ismail Ridzuan Bin Mohammad Muhammad Shah Izlan Bin Ismail	Project Manager, DSSB Quality Assurance, DSSB Risk Management, DSSB Solution Architect, DSSB

Approved By:

Version	Approver Name	Role
1.1		
1.1		
1.1		

TABLE OF CONTENT

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to define the standards, principles, and guidelines for the design and implementation of the User Interface (UI) and User Experience (UX) for the New Agihan System (NAS) under the Zakat Management System initiative by Lembaga Zakat Selangor (LZS). This specification ensures that the system's interfaces meet the expectations of usability, accessibility, and consistency across both web and mobile platforms. By adhering to these standards, developers and designers can create a cohesive and user-friendly interface that meets the needs of all users.

1.2 REFERENCES

No	Link Name	Description
1.	**Web Content Accessibility Guidelines (WCAG) 2.1** *[https://www.w3.org/TR/WCAG21/]*	W3C Recommendation defining success criteria and techniques to make web content accessible to all users, including those with disabilities
2.	**WAI-ARIA Authoring Practices Guide (APG)** *[https://www.w3.org/WAI/ARIA/apg//]*	W3C guide outlining patterns and best practices for applying ARIA roles, states, and properties to build accessible widgets and components
3.	**Material Design Guidelines Overview** *[https://m2.material.io/design/guidelines-overview]*	Google's adaptable system of principles, components, and tools for designing cohesive, high-quality digital experiences

1.3 ACRONYM

Acronym	Description
UI	User Interface
UX	User Experience
NAS	New Agihan System
LZS	Lembaga Zakat Selangor
DSSB	Data Science Sdn. Bhd

WCAG	Web Content Accessibility Guidelines
PWD	Persons With Disabilities
API	Application Programming Interface
KP	Kad Pengenalan (National ID)
ELT	Extract, Load, Transform
FAB	Floating Action Button
H1, H2, H3	Header Levels 1, 2, and 3 in HTML Typography
ARIA	Accessible Rich Internet Applications
CRUD	Create, Read, Update, Delete (implied through system actions)
KPI	Key Performance Indicator
CMD	Command (typically Cmd key on Mac, used in shortcuts)
UXDS	User Experience Design Specification
SVG	Scalable Vector Graphics (implied from icon usage)
DOM	Document Object Model (related to accessibility and scripting standards)
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
API	Application Programming Interface
ID	Identifier (e.g., ID pemohonan)
ARIA	Accessible Rich Internet Applications
CDN	Content Delivery Network (implied for UI resources and icons)

2 OBJECTIVES AND SCOPE

2.1 OBJECTIVES

This document aims to establish the user interface (UI) and user experience (UX) design standards for the NAS system to ensure:

- Alignment with Lembaga Zakat Selangor's corporate identity.
- User-friendly interfaces for all user levels.
- Ensured usability and accessibility of the system.
- Adoption of a user-centered design approach.

2.2 SCOPE

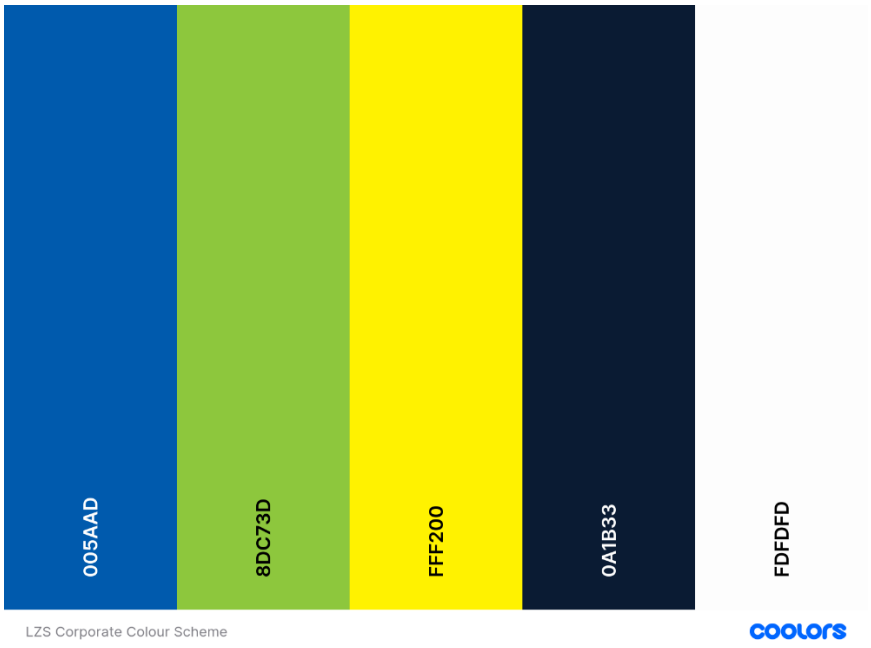
This document covers UI/UX guidelines for:

- Web-based system
- Mobile applications

3 UI / UX DESIGN PRINCIPLES

Principle	Description
Consistency	All UI elements should be consistent in terms of color, icons, layout, and fonts.
Simplicity & Focus	Information displayed must be clear and not overwhelming.
Mobile-Friendly	Interfaces should be responsive to various mobile screen sizes.
Accessibility for PWD	Comply with WCAG 2.1 standards for accessibility, specifically for users with color blindness.
Clear Navigation	Navigation should be intuitive and access to core information should not exceed 3 clicks.

4 CORPORATE COLOR SCHEMA



LZS corporate color schema

1. **Primary Color (Blue)**
- Hex: #005AAD
- RGB: 0, 90, 173
2. **Secondary Color (Green)**
- Hex: #8DC73D
- RGB: 141, 199, 61
3. **Accent Color (Yellow)**
- Hex: #FFF200
- RGB: 255, 242, 0

5 DASHBOARD OVERVIEW (VISUAL LAYOUT & BEST PRACTICES)

The dashboard is a central space designed to present key performance information in a clear and meaningful way. Its purpose is to support fast understanding, performance tracking, and confident decision-making—whether by operational teams or leadership. To achieve this, every part of the layout must be intentional, and every component must be matched appropriately with the type of data being displayed.

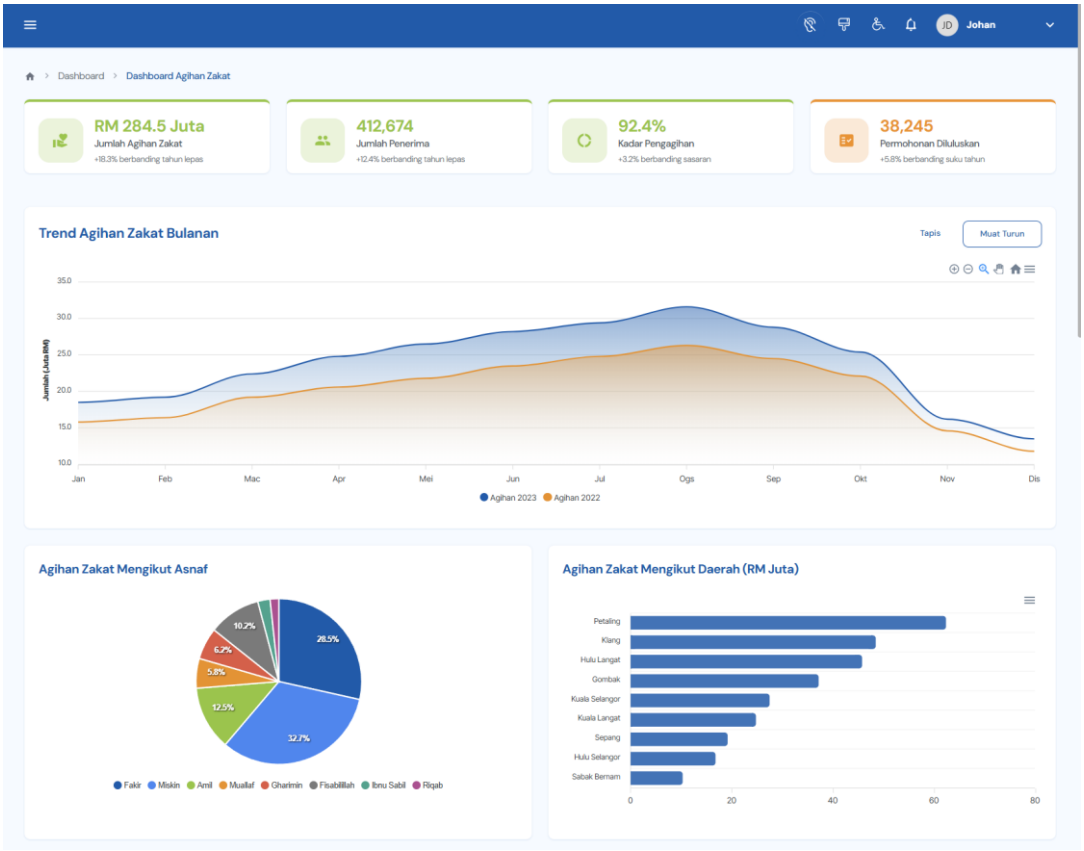
As this dashboard will be powered by an embedded Power BI experience, it is essential that its **visual and interactive elements match the design language already used throughout the LZS system**. The result should feel native—not like a separate tool, but an integrated part of the platform.

To maintain visual harmony and functional clarity, follow these design principles:

- Start with a row of **summary panels** that highlight the most critical metrics—totals, trends, or percentages. Keep the format uniform across cards, using the same icon placement, background color tone, and label structure. These should communicate the “big picture” immediately.
- Use **data-specific visual types**. For example, trend data over time should use line or area charts, while comparisons between categories work better with bars or pie charts. Do not use visuals just for style—every chart should be chosen based on what it helps the user understand quickly.
- Group charts and content into **modular sections**, each with a clear title. Avoid mixing multiple ideas in a single area. For example, time-based performance, geographic breakdowns, and recipient categories should be shown separately.
- Ensure **consistent spacing and alignment**. Margins, padding, and card size should follow the system’s grid and spacing standards to maintain rhythm and balance throughout the screen.
- Apply **typographic hierarchy**. Use larger, bolder text for key figures, and keep supporting labels in smaller, regular fonts. This helps guide attention naturally to what matters most.
- Place action elements—such as filters or export options—**subtly and predictably**. These should support the viewing experience, not compete with the visuals.

- Avoid overcrowding. Prioritize clarity over quantity. It's better to show a few well-designed insights than to overwhelm users with too many blocks on a single screen.
- Finally, **match each visual component to the nature of the content**. If a value is best communicated as a simple number, don't force it into a chart. If a pattern over time is important, ensure there's enough horizontal space for a timeline to be effective. Let the content guide the format.

By respecting these practices, the embedded dashboard will not only deliver accurate insights—it will also reflect the same user-first, clean, and trusted design approach that defines the LZS digital experience. Power BI will provide the analytics power, but it's the thoughtful design that ensures the results are easy to understand and take action on.



Dashboard interface with Power BI-embedded visual components and metric cards.

6 GRID LAYOUT STANDARD

Grid layout is the foundation for creating balanced, consistent, and scalable page structures across the system. A good grid ensures that all elements—such as cards, buttons, text, and components—are properly aligned, evenly spaced, and visually predictable. It improves

readability, strengthens hierarchy, and allows the interface to adapt well across various screen sizes.

The layout grid should be applied consistently across both web and mobile views, aligning with the overall LZS design style. The following sections explain how the grid system is structured and how to use it effectively.

6.1 GRID STRUCTURE AND SIZING

This section defines the column structure, gutter width, and margin values that form the core of the grid system. It ensures that screen layouts remain visually balanced, structured, and aligned with established design principles.

- A **12-column grid system** shall be used for desktop layouts. This provides sufficient flexibility for arranging content in full-width, half-width, or multi-column configurations.
- For tablets and smaller viewports, the grid should responsively adjust to **8 or 4 columns**, depending on the interface complexity and space constraints.
- All columns must maintain **equal width**, and gutters (the spacing between columns) should be consistently set to **16px or 24px**, based on the screen size and design density.
- **Outer margins** must be applied to separate the content from the screen edges. For desktop layouts, a margin of **24px** shall be used, while mobile layouts should use a **16px** margin.
- Content modules, such as cards, charts, or sections, must align to the grid and occupy a defined number of columns (e.g., 3, 6, 12), depending on their relative importance or content size.
- The grid must not be manually overridden or altered per screen. Instead, designers and developers must place all interface elements within the defined column structure.
- No content shall extend outside the grid boundaries. Overlapping columns or misaligned content is prohibited as it negatively impacts layout consistency and user experience.

Grid layout guide showing multiple column combinations based on a 12-column system.

6.2 ALIGNMENT, SPACING, AND RESPONSIVE BEHAVIOR

This section outlines how elements should be aligned within the grid, how vertical spacing is maintained, and how the layout behaves across responsive breakpoints.

- All user interface components must be placed **within column boundaries** and aligned precisely to the grid. Arbitrary positioning or free-floating elements are not permitted.
- **Vertical spacing** between stacked elements must be consistent and follow an **8px or 16px spacing scale**, depending on the design density and visual hierarchy.
- On smaller viewports, elements that appear side-by-side on desktop (e.g., two 6-column blocks) must **stack vertically and occupy the full width** (12 columns) to ensure mobile usability.

- Grid overlays or alignment tools should be used during design and development stages to verify accuracy and ensure adherence to spacing rules.
- While the height of content may vary depending on use case, **top and bottom internal padding** within components (e.g., cards, sections) must be consistent to preserve visual rhythm.
- All content—headers, footers, filters, charts, and forms—must respect the left and right grid boundaries and must not exceed or fall short of the grid's outer margins.
- **Fixed-width containers** should be avoided. All layouts must use **percentage-based sizing** that scales with the screen, maintaining grid alignment at all times.

Dashboard layout with four metric cards, a main chart section, a summary panel, and three smaller charts.

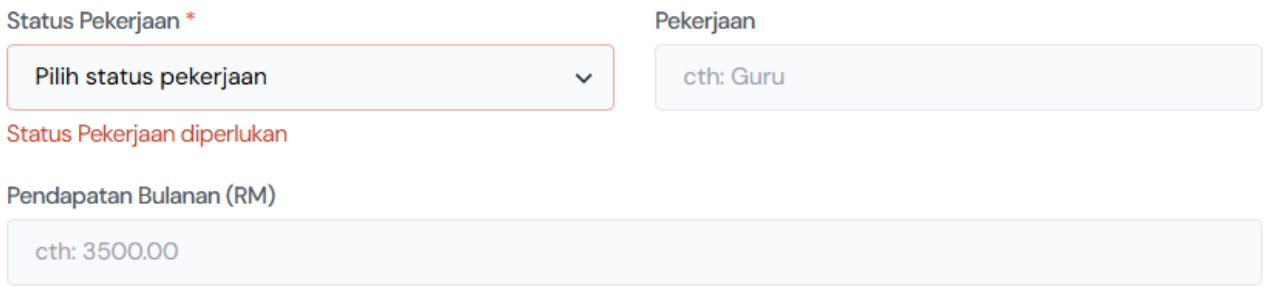
7 FORM DESIGN STANDARDS

7.1 MANDATORY FIELDS

Clearly indicating mandatory fields is critical to prevent incomplete submissions and reduce user frustration. All required fields must display a red asterisk (*) directly after the field label, using a consistent color code (e.g., #D32F2F) and spacing. The asterisk should be accessible, with screen readers announcing "wajib diisi" (required or suitable text based on situtations). Below each required input, provide concise helper text in a neutral color (e.g., "Wajib diisi" or "Required field") to reinforce the requirement. Tooltips should be available on hover or focus, offering additional context or format guidance (e.g., "Nombor Kad Pengenalan seperti di MyKad" for identification fields).

For example, in the Profil Pemohon module, the "No. Kad Pengenalan" field must show a red asterisk, helper text, and a tooltip explaining the expected input. In the Semakan & Kelulusan module, "Tarikh Kelulusan" should be marked as required, with helper text specifying the format ("Format: DD/MM/YYYY") and a tooltip if further clarification is needed.

- Place a red asterisk (*, color: #D32F2F) immediately after all required field labels.
- Ensure asterisks are accessible (screen reader text: "wajib diisi").
- Add helper text below each required input (e.g., "Wajib diisi").
- Provide tooltips for additional field guidance or format requirements.
- Never use only color to indicate required status—always pair with text or icon.
- For iconography, use a standard icon set (e.g., IcoMoon or Material Icons) for consistency if an info icon is needed.



Example "Status Pekerjaan" dropdown with required field error indicator.

7.2 LOGICAL FIELDS GROUPING

Organizing form fields into logical groups is essential for reducing cognitive load, improving data entry accuracy, and creating a visually structured interface. Group related fields together within visually distinct containers, such as <Card> components or clearly defined sections, to help users understand the context and flow of information.

Each group should be introduced with a subheader using a semibold weight and a 16px font size, ensuring clear hierarchy and scannability. For example, in the Profil Pemohon module, all address-related fields (Jalan, Bandar, Poskod) should be grouped under a card

titled "Alamat Pemohon," while personal details (Nama, No. KP, Tarikh Lahir) are grouped under "Maklumat Peribadi."

- Organizing form fields into clearly defined groups is essential to reduce user effort, improve input accuracy, and support a clean and intuitive interface. A well-grouped form layout helps users quickly understand the purpose of each section and follow the natural flow of data entry, particularly for multi-step or multi-part forms.
- All forms within the system must follow a structured grouping method that aligns with the overall LZS UI/UX design language.

Grouping Approach and Structure

- Group related input fields into visually distinct sections using card-based containers or bordered panels. Each group should present a single topic or data category, such as personal details, contact information, or address.
- Begin each group with a **section heading** that is left-aligned, styled in **semibold 16px font size**, and clearly indicates the content of the group. This heading serves as both a visual anchor and an accessibility label.
- Example groupings may include:
    - “Maklumat Peribadi” for fields like name, identification number, birth date, and gender.
    - “Alamat Pemohon” for address-related fields such as street address, city, postcode, and state.
    - “Maklumat Perhubungan” for email, phone number, and preferred communication method.
- Maintain a **consistent vertical spacing of 24px** between each section to create clear separation and prevent the form from appearing cluttered or overwhelming.
- Within each group, align input fields to a **12-column responsive grid layout**. Use even column spans (e.g., 6 + 6 for side-by-side fields, or 12 for full-width) to ensure alignment and consistency across different devices.
- When using multiple columns within a section, ensure that all form fields are baseline-aligned and maintain equal vertical spacing between rows.
- Clearly identify any optional field groups by appending "(Pilihan)" to the section heading (e.g., “Maklumat Tambahan (Pilihan)”) to guide users without causing confusion.

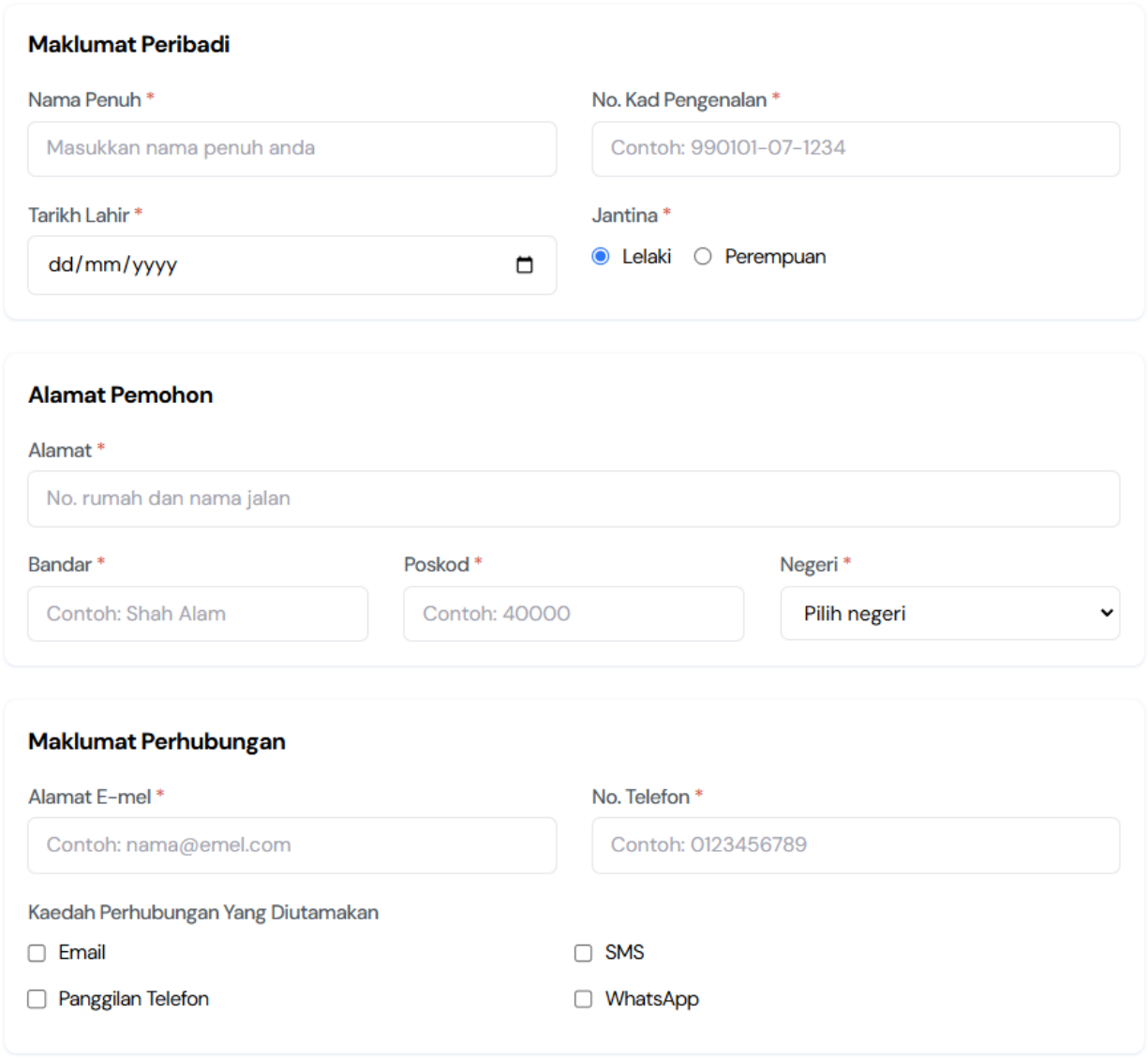
- Apply subtle visual dividers, borders, or background shading within card containers when forms become long or contain more than three field groups. This improves visual hierarchy without overwhelming the design.
- For accessibility compliance, ensure that each field group is **programmatically associated** with its section heading using semantic HTML and ARIA attributes as required.

Summary of Implementation Principles

- Group fields using container components or bordered visual sections.
- Label each group using a consistent, visible subheader styled at 16px semibold, aligned to the left.
- Maintain at least 24px vertical spacing between field groups for clarity.
- Apply the 12-column grid system within each group for layout alignment.
- Use consistent spacing and sizing for field pairs and rows.
- Clearly mark optional groups and preserve semantic accessibility through proper markup.

This standard ensures that all form designs across the system remain organized, easy to scan, and accessible—especially in scenarios that require users to complete multiple data categories within a single view. By following this approach, teams can maintain a structured, professional appearance while supporting user clarity and task efficiency.

.



Grouped form sections for "Maklumat Peribadi", "Alamat Pemohon", and "Maklumat Perhubungan" with labeled input fields.

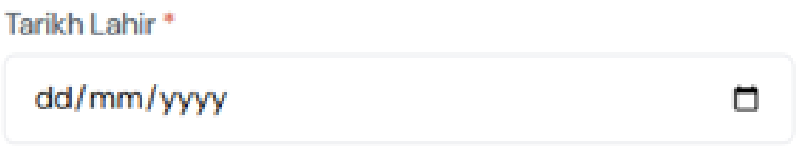
7.3 PLACEHOLDER TEST : BEST PRACTICES

Placeholder text in form fields should be treated as a subtle, supplementary hint—never as a replacement for a visible label. Placeholders are intended to provide brief, context-specific guidance on the expected input format or example values, helping users complete forms more efficiently. However, over-reliance on placeholder text can lead to usability and accessibility issues, such as confusion when the placeholder disappears on typing or when users rely on it as the only indicator of field purpose.

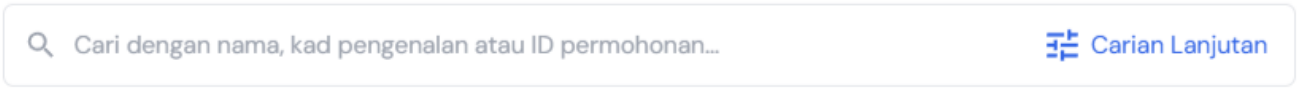
Therefore, always pair placeholders with persistent, visible labels.

a. Key Rules and Micro-Standards:

- Purpose: Use placeholder text only to clarify input format or provide a short example (e.g., DD/MM/YYYY for date fields, or Masukkan No. KP atau Nama for search fields).
- Clarity & Length: Keep placeholder text concise (ideally under 40 characters, never exceeding 50), written in sentence case, and free of unnecessary words or jargon.
- Accessibility: Ensure that placeholder text is not the sole means of conveying required information. Screen readers may not always announce placeholders, so all critical instructions must be available in labels or helper text.
- Visual Style: Use a lighter text color (e.g., #BDBDBD) for placeholders to distinguish them from user input, but maintain sufficient contrast for readability (minimum 4.5:1 ratio).
- Localization: Always localize placeholder text to the system language, and avoid abbreviations unless they are universally understood by the user base.
- Examples:
  - For a date of birth field:
    - Label: "Tarikh Lahir"
    - Placeholder: DD/MM/YYYY



  - For a search field in Carian Pemohon:



Placeholder: “Cari dengan name, kad pengenalan atau ID pemohonan.”

b. Do Not:

- Do not use placeholder text as a substitute for field labels.
- Do not include validation messages or error prompts in placeholder text.

- Do not use all caps or excessive punctuation.

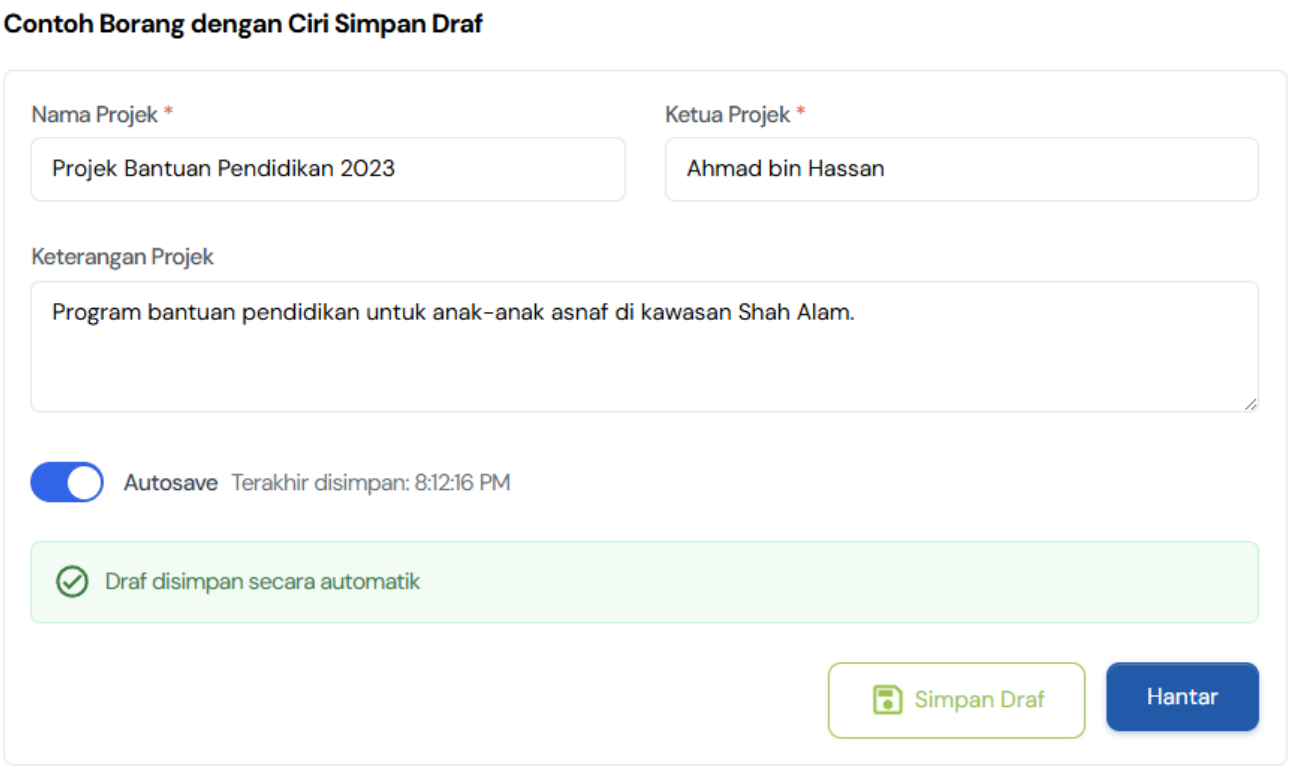
c. Micro-Behavior Recommendations:

- When a field is focused and the placeholder disappears, ensure that the field label remains visible (e.g., floating label pattern).
- If a field is required, indicate this in the label or helper text, not in the placeholder.
- For fields with specific formatting requirements, supplement the placeholder with an info icon (using a standard icon set such as IcoMoon or Material Icons) and a tooltip for additional guidance.

7.4 SAVE DRAFT & AUTOSAVE

The ability to save drafts and autosave is essential for user convenience and data integrity. A Save Draft button should be fixed to the bottom-right of long forms, such as those in the Profil Pemohon module. Autosave should be triggered on blur or every 30 seconds, with a non-intrusive toast notification stating "Draf disimpan secara automatik". In the event of a network failure, an inline alert should inform the user: "Gagal menyimpan draf. Sila cuba lagi." In the Semakan & Kelulusan module, a toast should confirm that the draft approval has been saved.

- Fix the Save Draft button to the bottom-right of long forms.
- Trigger autosave on blur or every 30 seconds.
- Provide toast notifications for autosave and network failures.

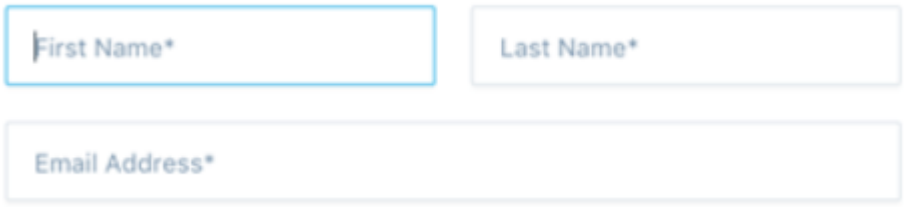


Example form with toast notification for autosave, and fixed-position "Simpan Draf" button.

7.5 MINOR USABILITY PATTERNS

Minor usability patterns are essential micro-behaviors that significantly improve the efficiency, accessibility, and satisfaction of form interactions. These patterns address common pain points and ensure that users can complete forms quickly and with minimal friction, especially in complex workflows or high-stakes environments.

- Auto-Focus on First Input:
  When a form page or modal dialog opens, the first interactive input field should automatically receive focus. This reduces unnecessary clicks and guides users to begin data entry immediately. For example, in the status-change modal of the Semakan & Kelulusan module, the cursor should be placed in the first editable field. For accessibility, ensure that auto-focus does not disrupt screen reader users—use it judiciously and test with assistive technologies.



First name field are focused on first input

- Scroll and Focus on Validation Errors:
  Upon form submission, if validation fails, the interface must automatically scroll to the first invalid field and set keyboard focus there. This helps users quickly identify and correct errors, reducing frustration. The invalid field should be visually highlighted (e.g., with a 2px red border and an error icon from the IcoMoon or Material Icons set), and an error message should be programmatically associated using aria-describedby for screen readers.

- Prefill Known User Data:
  Where possible, prepopulate form fields with known user data (e.g., from the profile API in the Profil Pemohon module). This minimizes redundant data entry, reduces errors, and speeds up the process. Always allow users to review and edit prefilled values, and clearly indicate which fields have been auto-filled.

- Keyboard Shortcuts for Efficiency:
  Provide keyboard shortcuts for common actions, such as Ctrl+S (or Cmd+S on Mac) to save drafts. When triggered, display a non-intrusive toast notification confirming the action (e.g., "Draf disimpan"). Ensure that shortcuts do not conflict with browser or assistive technology defaults, and document available shortcuts in a help tooltip or modal.

- Micro-Behavior Enhancements:
  - When a field is auto-focused, ensure that any floating label or placeholder remains visible for context.
  - If a user attempts to submit a form with multiple errors, consider displaying a summary of errors at the top, with anchor links to each invalid field.

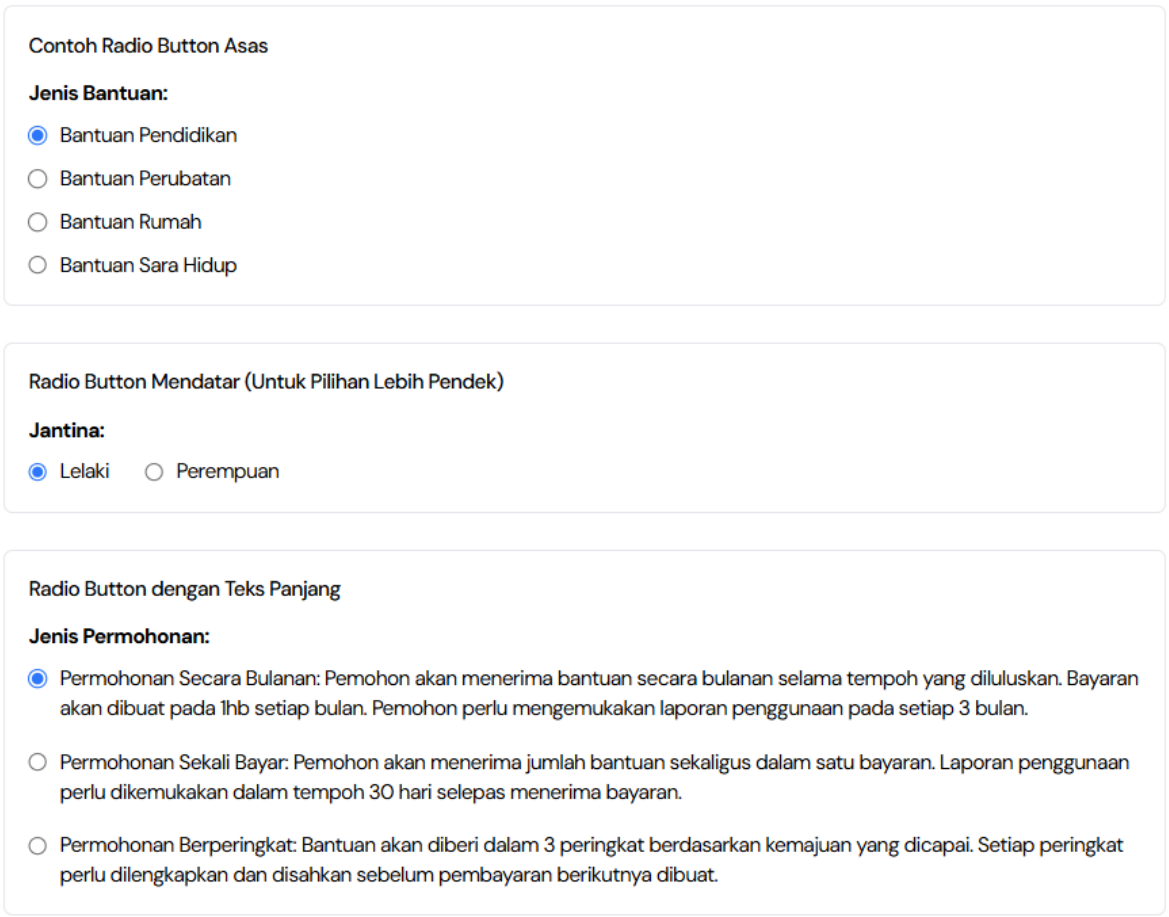
- For long forms, provide a persistent "Save Draft" button fixed to the bottom-right, ensuring it is always accessible without scrolling.

8 DATA SELECTION CONTROLS : RADIO BUTTONS, CHECKBOXES AND DROPDOWNS

8.1 PROPER SELECTION LOGIC

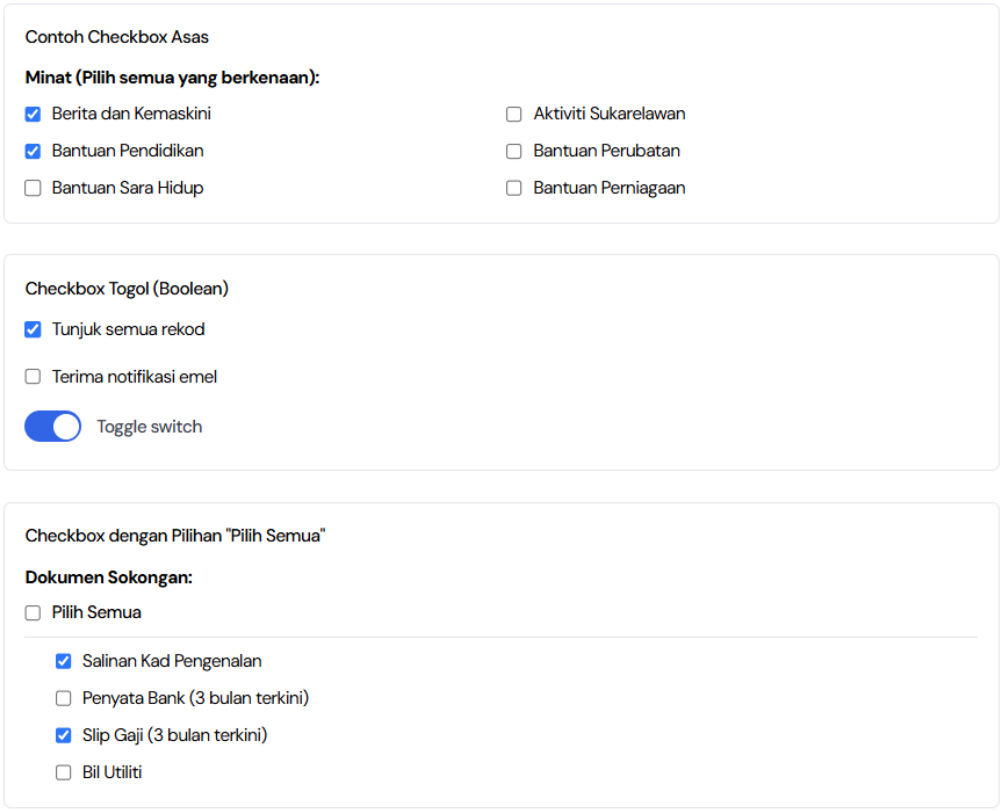
Choosing the correct selection control is critical for usability and clarity. The following rules must be applied:

- Radio Buttons
  - Use for single selection among a small, visible set of options (ideally 2–5 choices).
  - All options should be visible at once, allowing users to compare and select quickly.
  - Example: Selecting "Jenis Bantuan" in a form where only one type can be chosen.
  - Avoid using radio buttons for more than 5 options; this can overwhelm users and clutter the interface.
  - Always provide a clear label and group related radio buttons within a <fieldset> and <legend> for context and accessibility.

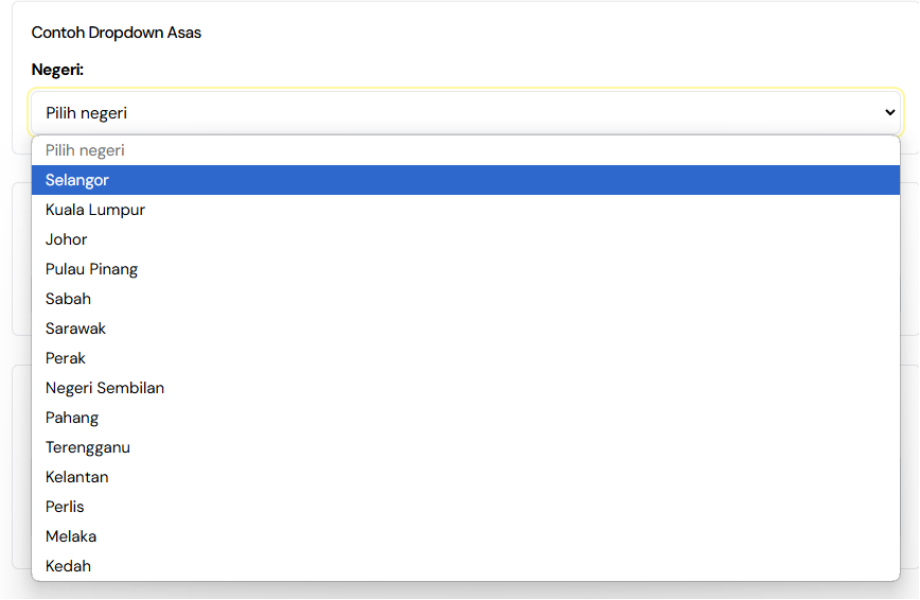


Grouped examples of vertical, horizontal, and long-text radio buttons for single-option selection.

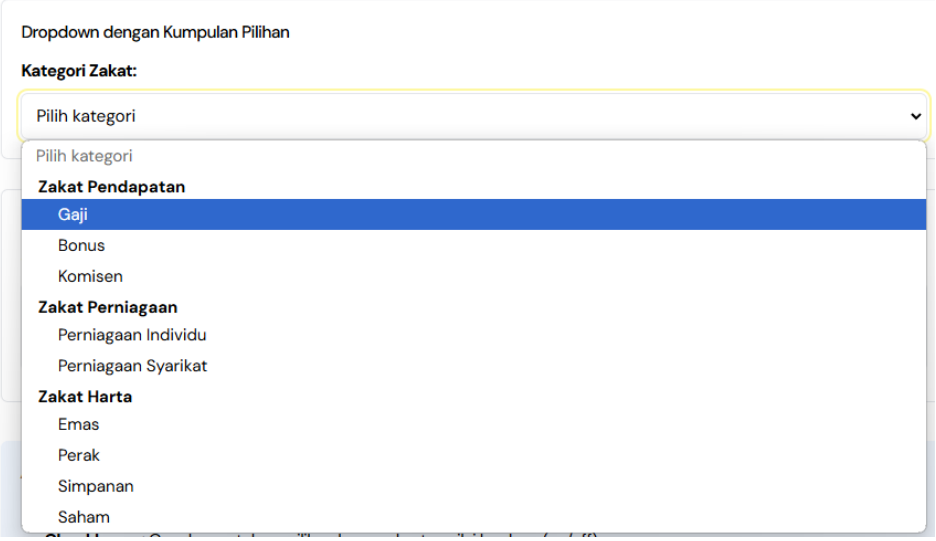
- Checkboxes
  - Use for multi-select scenarios, where users can choose any number of options, or for boolean toggles (on/off).
  - Example: "Tunjuk semua rekod" (Show all records) toggle in an audit log, or selecting multiple interests in a profile form.
  - For a single boolean value, use a single checkbox with a clear label.
  - For multiple related options, group checkboxes together and provide a group label.
  - If "Select All" is provided, ensure it is visually and programmatically linked to the group.



- Dropdowns (Select Menus)
  - Use for single selection when there are more than 5 options, or when space is limited.
  - Example: "Kategori Zakat" filter in a search form with many categories.
  - For multi-select dropdowns, ensure clear affordances (e.g., checkboxes inside the dropdown, clear selected state).
  - Avoid using dropdowns for only 2–3 options; prefer radio buttons or checkboxes for better visibility and fewer clicks.
  - Always provide a default unselected state or a clear placeholder (e.g., "Sila pilih kategori").



Basic dropdown for selecting "Negeri" with a long vertical list of states.

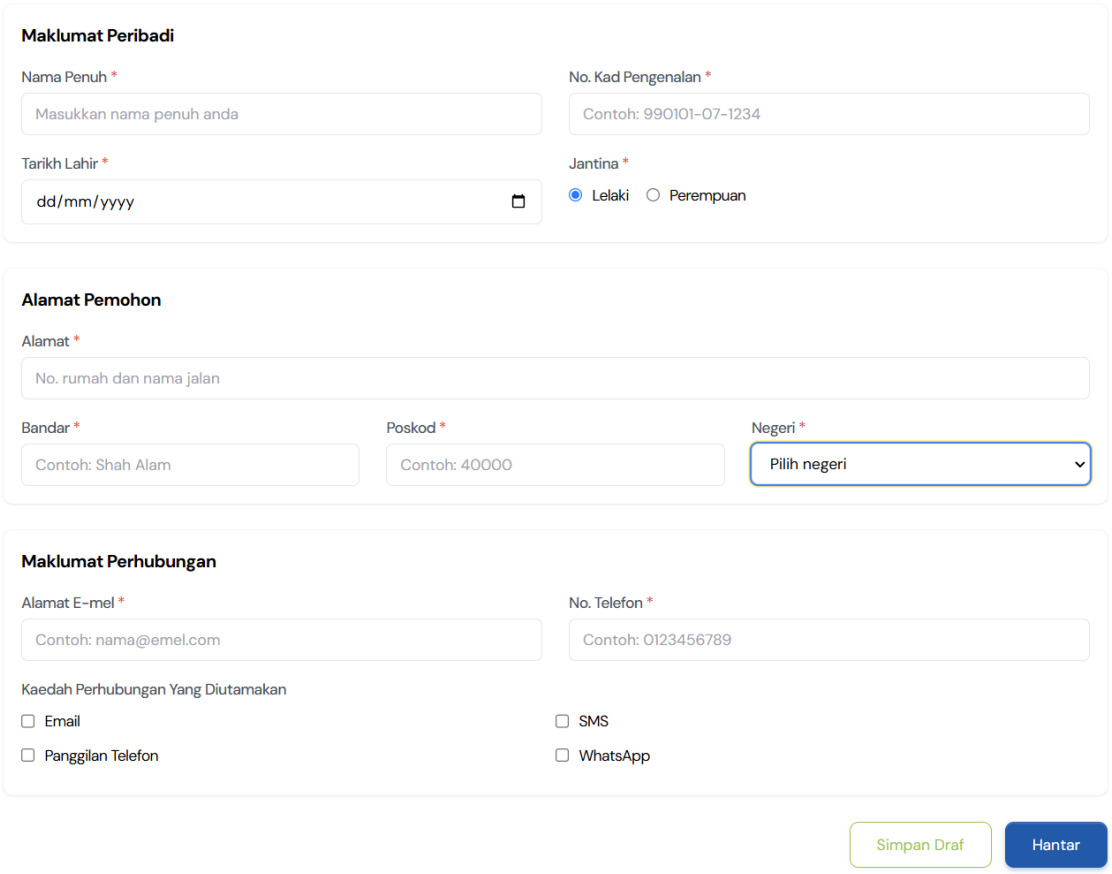


Grouped dropdown for "Kategori Zakat" with categorized options under income, business, and assets.

- Additional Micro-Standards and Best Practices
  - Never disable options unless absolutely necessary; instead, explain why an option is unavailable.
  - For all controls, ensure keyboard navigation is fully supported (Tab, Arrow keys, Space/Enter to select).
  - When using icons (e.g., info or help), use a consistent icon set such as IcoMoon or Material Icons, and provide accessible tooltips.

- Visually indicate the selected state with clear contrast and a 2px primary-blue outline on focus.
- For long lists, consider grouping options with subheadings or dividers for easier scanning.

- Example
  - In an applicant profile form, use **radio buttons** for "Jantina" (Lelaki, Perempuan), **checkboxes** for "Kaedah Perhubungan Yang Diutamakan" (Email, Panggilan Telefon, SMS, WhatsApp), and a **dropdown** for "Negeri" (many options)



Example of the usage of radio buttons and checkboxes

8.2 SEARCH BARS INSIDE DROPDOWNS

Search bars embedded within dropdown menus are essential for improving usability, especially when users must select from a long list of options or when option labels are

lengthy and hard to scan. Implementing a search bar inside dropdowns allows users to quickly filter and locate the desired item, reducing cognitive load and time spent scrolling.

- When to Include a Search Bar:
  - Always provide a search bar in dropdowns containing more than 10 options.
  - Consider adding a search bar for dropdowns with complex or lengthy option labels, even if the total count is below 10.
  - For hierarchical or grouped dropdowns (e.g., country > state), ensure the search covers all visible and nested options.

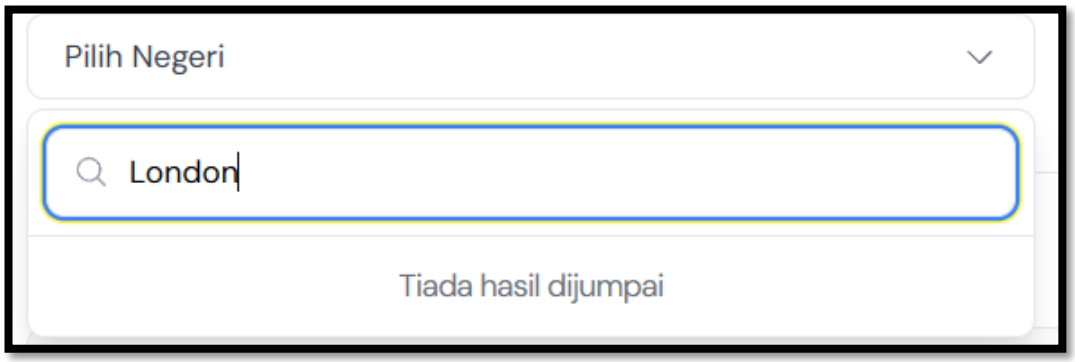
- Behavioral Standards:
  - Debounce Input: Implement a 300ms debounce on the search input to prevent excessive filtering operations and improve performance, especially for large datasets.
  - Highlight Matches: Visually highlight the matching substring within each filtered option (e.g., bold or colored text) to help users quickly identify relevant results.
  - No Results State: If no options match the search, display a clear, neutral message such as "Tiada hasil dijumpai" in grey text. Ensure this message is accessible to screen readers.
  - Keyboard Accessibility: The search input must be focusable via keyboard (Tab), and users should be able to navigate filtered results using arrow keys and select with Enter.
  - Clear Search Button: Provide a small, accessible "clear" (×) icon inside the search input to reset the filter instantly.
  - Consistent Placement: Place the search bar at the top of the dropdown, with sufficient padding and clear visual separation from the options list.
  - Mobile Responsiveness: Ensure the search bar remains usable and visible on mobile devices, with touch-friendly sizing and spacing.

- Micro-Standards and Usability Enhancements:

- Autofocus the search input when the dropdown opens, so users can immediately start typing.
- If the dropdown is closed and reopened, reset the search input unless the user is expected to make multiple selections.
- For multi-select dropdowns, show selected items as removable chips above the search bar for clarity.
- Use a consistent icon (e.g., Material Icons' "search") for the search input, and ensure all icons are accessible with appropriate aria-labels.
- Avoid truncating long option labels; instead, allow horizontal scrolling or tooltips for full label visibility.

- Example:
  - In the "Negeri" dropdown of the Carian Pemohon module, a search bar should be present at the top. As the user types "Selangor," the list filters in real time, highlighting the matching text. If the user types an invalid string, "Tiada hasil dijumpai" appears in muted grey.

- A dropdown menu with a search bar at the top, filtered options below, highlighted matches, and a "no results" state message.



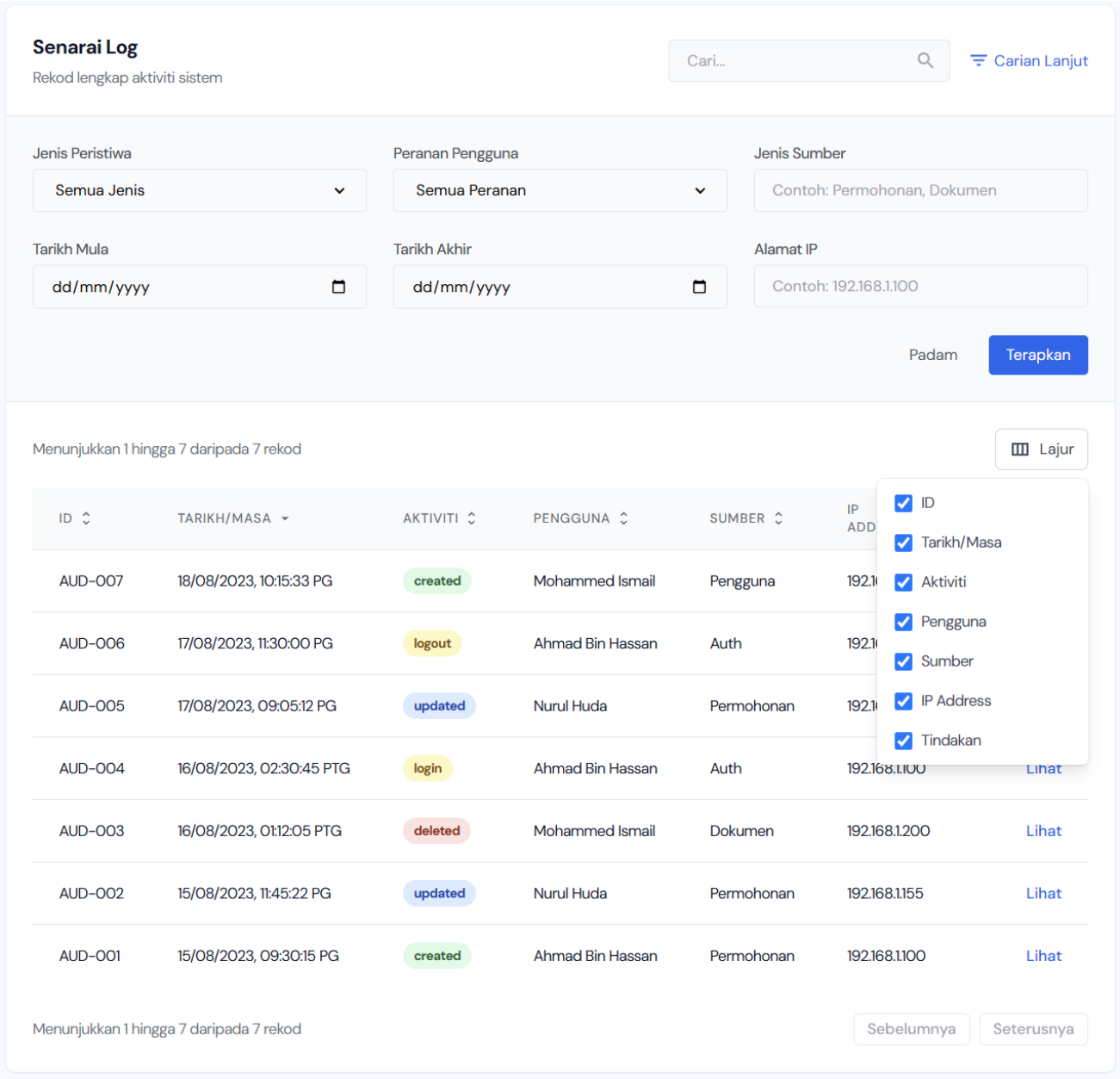
9 TABLE DESIGN STANDARD

Tables are used to present structured information in a clear and efficient format, allowing users to scan, sort, filter, and interact with data easily. This standard ensures all data tables across the system maintain usability and consistency, particularly in use cases such as logs, listings, records, and system reports.

9.1 GENERAL DESIGN PRINCIPLES

- Column headers must use clear, descriptive labels written in sentence case. Allow space for sorting icons and interaction without disrupting the header alignment.
- Tables should avoid heavy visual borders. Use light separators, alternating row backgrounds, or hover highlights to support visual scanning without introducing visual noise.
- Data must be aligned appropriately: left-aligned for textual content (e.g. names, roles), and right-aligned for numerical values where applicable.
- Display only essential columns by default. Support column visibility management via a column toggle dropdown. This allows users to customize their view based on task requirements.
- Status indicators such as system activities or record states should be presented using subtle, color-coded badges or labels. Use the defined system color palette to maintain visual consistency.

- Filtering tools should appear above the table, including dropdowns, input fields, and date range pickers. Group filters clearly and apply all changes using a single confirmation button (e.g. “Terapkan”).
- Row-level actions must be visible yet minimal. Use text links for common actions (e.g. “Lihat”) instead of stacking buttons, unless the context justifies otherwise.
- Pagination controls should be displayed at the bottom right of the table, using clear labels such as “Sebelumnya” and “Seterusnya.” Avoid infinite scrolling to preserve navigation context.
- Maintain consistent vertical spacing within rows (minimum height of 48px) to support readability and touch interaction. Apply horizontal padding uniformly across all columns.
- On mobile devices or smaller viewports, implement responsive behavior by prioritizing essential columns, enabling horizontal scroll, or stacking content for improved accessibility.



Log activity table with column filters, color-coded status badges, and customizable column visibility dropdown.

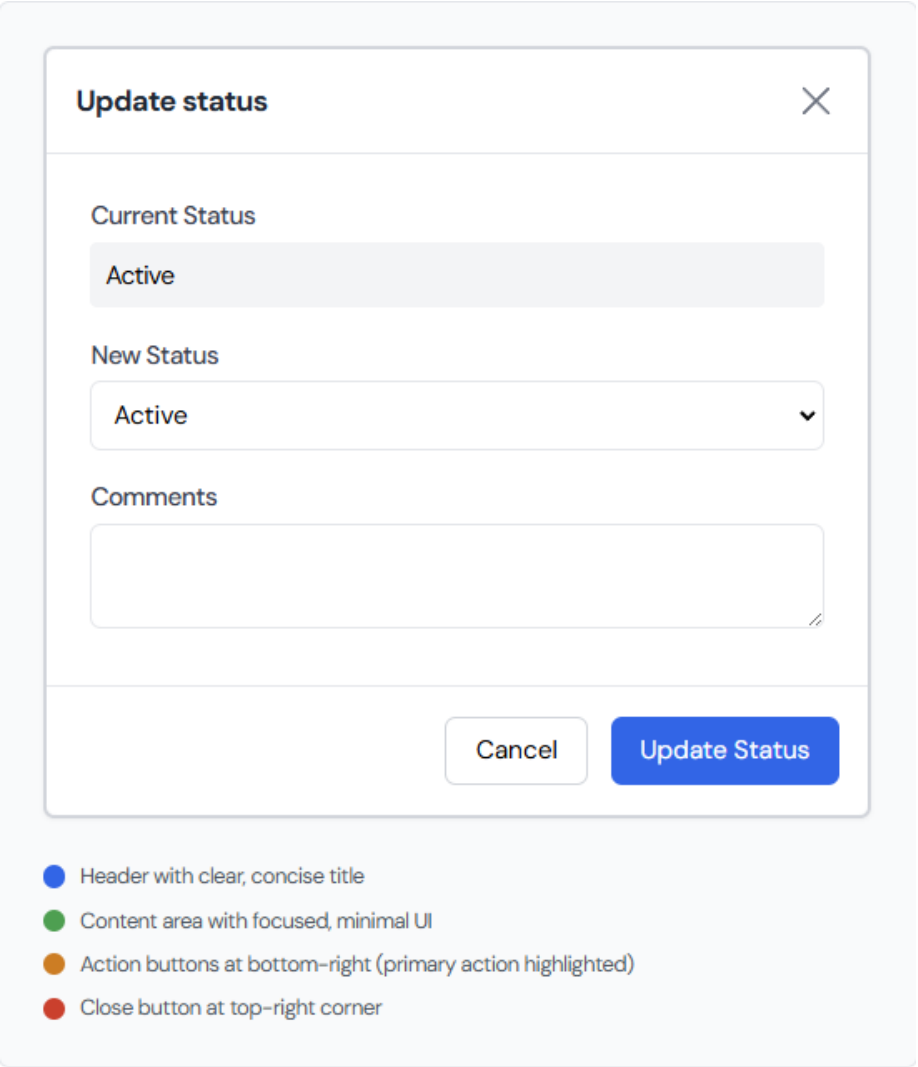
10 MODAL DIALOG DESIGN STANDARDS

Modals are temporary dialog boxes that appear over the current page to capture user input, confirm actions, or display important messages without requiring page navigation. When used correctly, modals help users stay focused on a single task while maintaining the context of the main page.

To ensure consistency and clarity, every modal should follow a standard structure and behavior. The design must be clean, with clear hierarchy and accessible controls that are easy to understand even for non-technical users.

Use the following guidelines when designing modal dialogs:

- Always include a **clear and concise title** at the top of the modal to indicate its purpose, using semibold text styling and sentence case. For example, use “Update Status” instead of generic titles like “Form Dialog.”
- Provide a **close button (×)** at the top-right corner of the modal. This button must be easy to access using a mouse or keyboard, and should be labeled for screen readers (e.g., “Close dialog”).
- The **main content area** of the modal should be focused and simple. Display relevant form fields, messages, or options using proper spacing. Avoid overloading the modal with too many elements.
- The **action buttons** must be placed at the bottom-right corner. The primary action (e.g., Save, Confirm) should be visually more prominent, followed by secondary options (e.g., Cancel). Use consistent button colors, spacing, and icons.
- Use modals only when necessary. Avoid using modals for content that could be displayed inline or on a separate page. Reserve modals for actions that require the user's immediate attention or confirmation.
- When the modal opens, it should **automatically focus on the first input field** or interactive element. This helps guide users to begin their task immediately without extra clicks.
- Prevent user frustration by **warning users before closing the modal** if there are unsaved changes. Show a prompt like “Are you sure you want to discard changes?” when they attempt to close it.
- Ensure the modal is **keyboard accessible**, meaning users can navigate using the Tab key, and close it with the Escape key. All interactive elements should follow a logical tab order.
- When the modal closes, **return focus to the button or link that opened it** so that users can continue where they left off without confusion.
- For accessibility, always include proper **ARIA roles** and label the modal for screen readers using role="dialog" and aria-labelledby attributes.



Modal dialog for updating status with form fields, top-right close icon, and bottom-right action buttons.

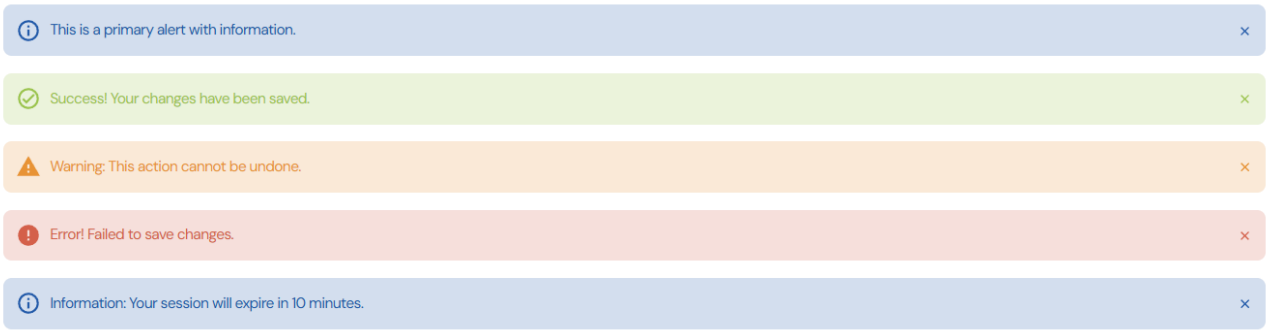
Following these principles ensures that modal dialogs are not only visually consistent but also accessible, user-friendly, and appropriate for both desktop and mobile environments.

11 ALERTS AND NOTIFICATIONS

Alerts are used to provide immediate feedback to users about the status of their actions, system events, or important information that requires attention. They serve a crucial role in guiding users through both successful workflows and error recovery. A well-structured alert not only informs but also helps users decide what to do next, reducing confusion and enhancing the overall experience.

Alerts must be visually distinct, easy to read, and consistent in placement and behavior. Each alert type should be differentiated by background color, icon, and message tone. For example, information alerts should use a soft blue background, success alerts a light green, warnings a soft yellow, and errors a muted red. These colors help users quickly recognize the nature of the message without needing to read every word.

All alerts must contain the following elements: a meaningful icon on the left to indicate type, a short and clear message in sentence case, and a dismiss icon (×) on the far right. The dismiss icon must be keyboard accessible and support screen reader labels such as "Close alert".



Alert components showcasing primary, success, warning, error, and info messages with dismiss icons

Use these alert types consistently across the application:

- Use **information alerts** for neutral system updates (e.g., “Your session will expire in 10 minutes”).
- Use **success alerts** to confirm completed actions (e.g., “Your changes have been saved”).
- Use **warning alerts** to caution users before performing critical or irreversible actions (e.g., “This action cannot be undone”).
- Use **error alerts** to indicate problems or failed actions (e.g., “Failed to save changes”).

Key behavior guidelines include:

- All alerts should be dismissible unless absolutely required to persist.
- Success and info alerts may auto-dismiss after a short duration (e.g., 5–10 seconds).

- Warning and error alerts should remain until manually dismissed, especially when user action is needed.
- Alerts should appear close to the content they relate to—typically above forms or in the top-right corner of the screen for toast-style notifications.

For usability and accessibility:

- Always pair color with icon and text—never use color alone to convey meaning.
- Use clear, action-oriented language. Avoid generic labels like "Success!" or "Error!" without context.
- Alerts should use role="alert" so screen readers can announce them immediately.
- Avoid using alerts for non-critical messages that could be conveyed through helper text or subtle UI changes.

12 CALENDAR INPUT STANDARDS (DATE PICKER USAGE)

Calendar input components play a vital role in enhancing form usability, particularly for workflows involving application periods, reporting filters, or scheduling. Consistency in their structure and behavior is essential to support accurate data entry, reduce user confusion, and maintain a professional interface.

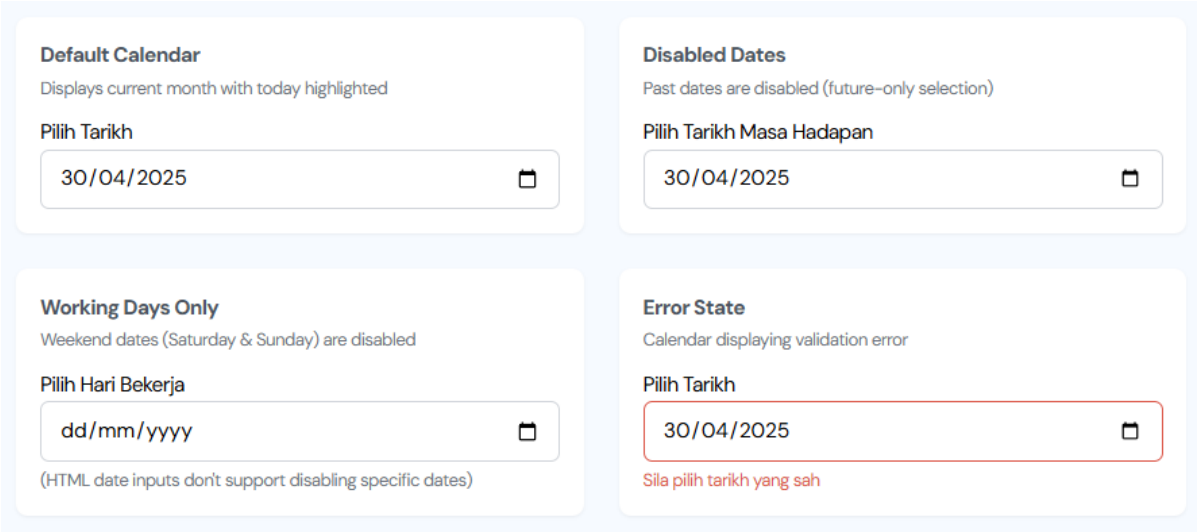
12.1 CALENDAR DISPLAY AND LAYOUT

The calendar input shall adopt a **monthly layout grid** (7-column structure, Monday to Sunday) to ensure a familiar and intuitive user experience. All date pickers must:

- Display the current month by default, with **today's date clearly highlighted** using the LZS-accredited visual cue.
- Allow navigation across months and years using accessible arrow controls, placed consistently.
- Respect the **minimum touch/click area of 44x44px** per date cell for accessibility compliance.
- Dynamically disable dates that fall outside the permitted range. This includes:
    - Past dates (if only future scheduling is valid),
    - Future dates (in reporting contexts),
    - Weekends or non-working days (based on configuration).

Misapplication to avoid:

- Inconsistent start-of-week across modules.
- Over-styled calendars that compromise readability or tappable accuracy.
- Displaying unselectable dates without visual differentiation.



Calendar input states for default, future-only, working days, and validation error handling.

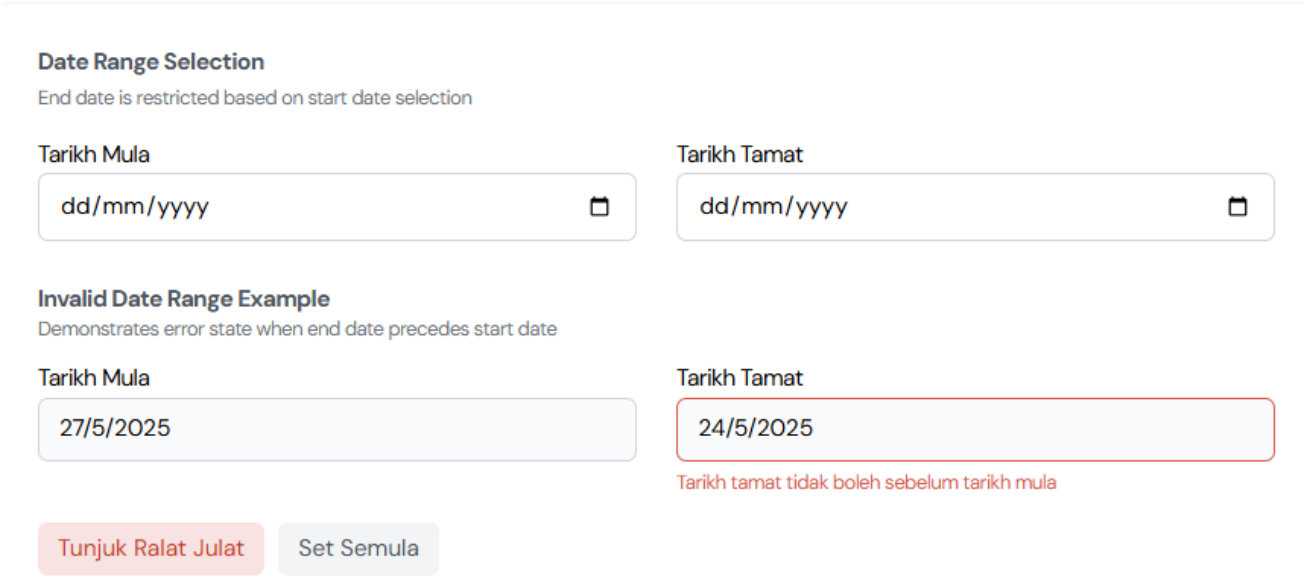
12.2 START AND END DATE USAGE

Where date ranges are required (e.g., "Tarikh Mula" and "Tarikh Tamat"), the system must enforce **logical sequence and visual pairing**.

Required behavior:

- "Tarikh Mula" must be selected before enabling "Tarikh Tamat."
- The second calendar (for end date) must restrict selectable values to those ≥ start date.
- Once both are selected, the range must be visually indicated using a background highlight.
- If the start date is changed post-selection, and the end date becomes invalid, it must be cleared or updated automatically with contextual guidance.

This approach prevents invalid entries such as backward ranges, which are common when such logic is not enforced.



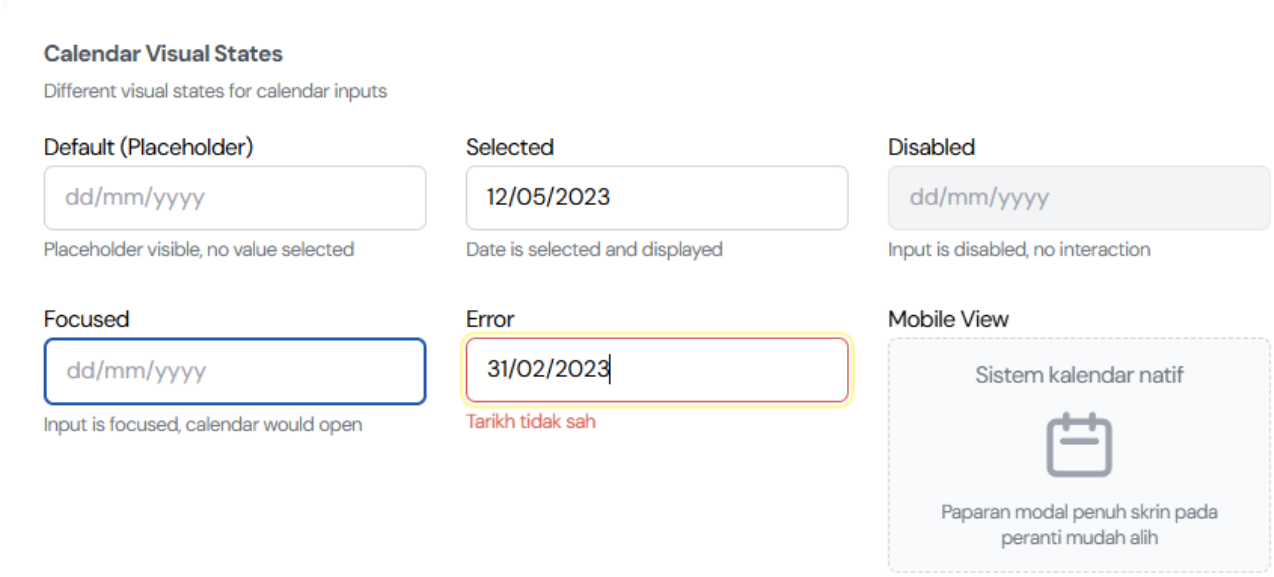
Date range picker with start/end date validation and error messaging.

12.3 VISUAL STATES AND RESPONSIVENESS

Calendar fields must support and display clear interaction states, including:

- **Default**: Placeholder (e.g., dd/mm/yyyy) visible; no value selected.
- **Focused**: Calendar opens upon user interaction.
- **Selected**: Date displayed using the system's formatted date style.
- **Error**: Red border with helper text (e.g., "Tarikh tamat tidak boleh sebelum tarikh mula").
- **Disabled**: Non-editable fields appear greyed out with no interaction.

On mobile, calendar inputs must trigger either a full-screen calendar modal or system-native input for smaller screen compatibility.



Visual states of calendar fields including placeholder, selected, disabled, focus, and error.

13 SEARCH FIELD BEHAVIOR STANDARDS

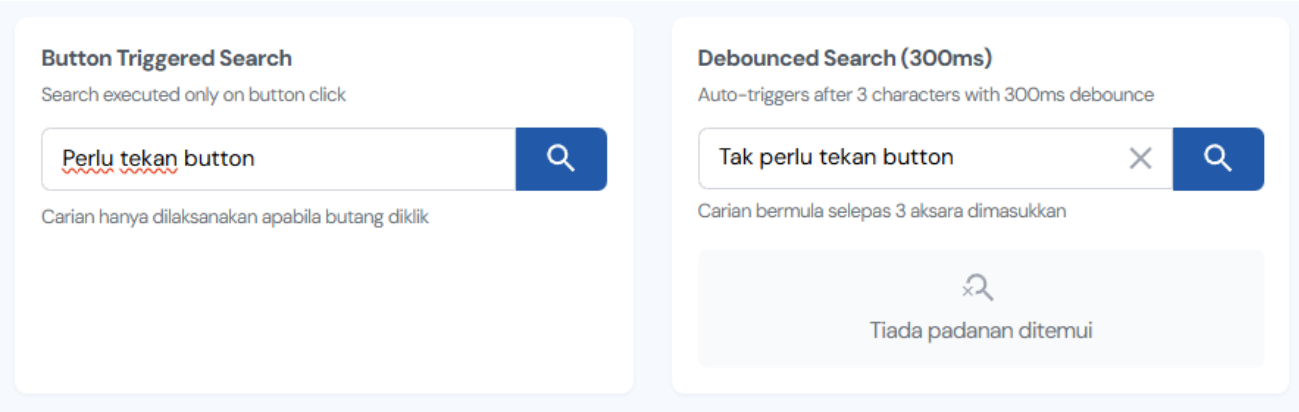
Search functionality allows users to locate, filter, or interact with large datasets. Poorly implemented search may impact system performance, flood users with irrelevant data, or create accessibility barriers. To avoid this, all search inputs must follow structured interaction patterns and validation rules.

13.1 INPUT TRIGGER

Search actions shall only be triggered through **user intent**—not per keystroke. Acceptable methods include:

- A visible **Search** button (magnifying glass or “Cari” label).
- Auto-trigger after a **minimum of 3 valid characters** with 300ms debounce to prevent excessive backend queries.

Input must never trigger search on every typed character. Additionally, users should be guided with contextual placeholder text (e.g., “Cari nama, emel atau ID permohonan”).



Search fields with button-triggered and debounce-triggered behaviors.

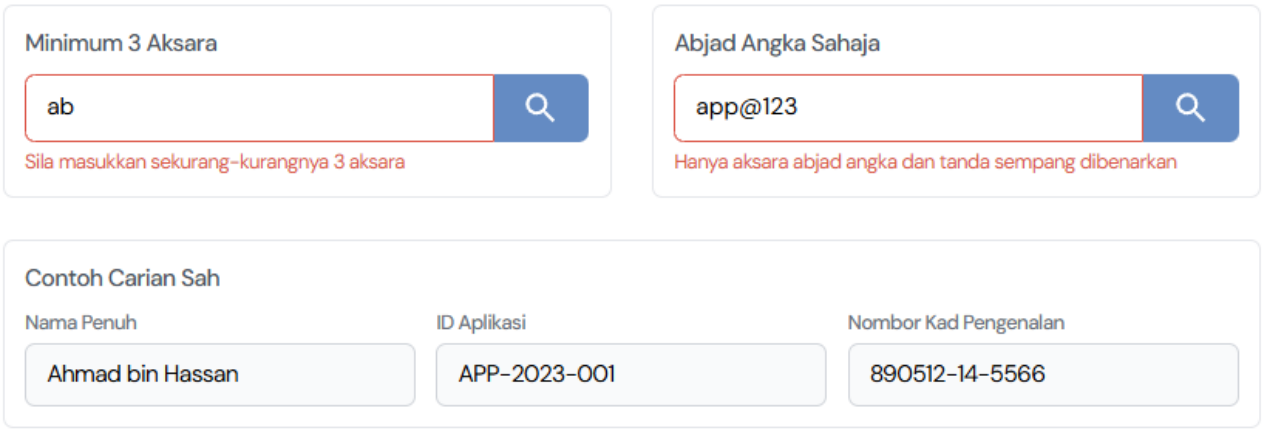
Failure to implement controlled triggers may result in:

- Server strain due to frequent database calls.
- Reduced user trust due to irrelevant or unstable results.
- **Pattern Matching and Validation**

Input values must be validated against known patterns before search execution. This includes:

- Restricting 1-character or symbol-only queries.
- Supporting alphanumeric inputs and approved symbols only (e.g., hyphenated IDs).
- Applying **regex-based logic** to:
    - Match full words (e.g., “Ahmad”) or partial but structured data (e.g., “APP-2025”).
    - Exclude loosely matched results such as entries matching any single letter.

These measures improve search relevance and help users retrieve specific records efficiently.

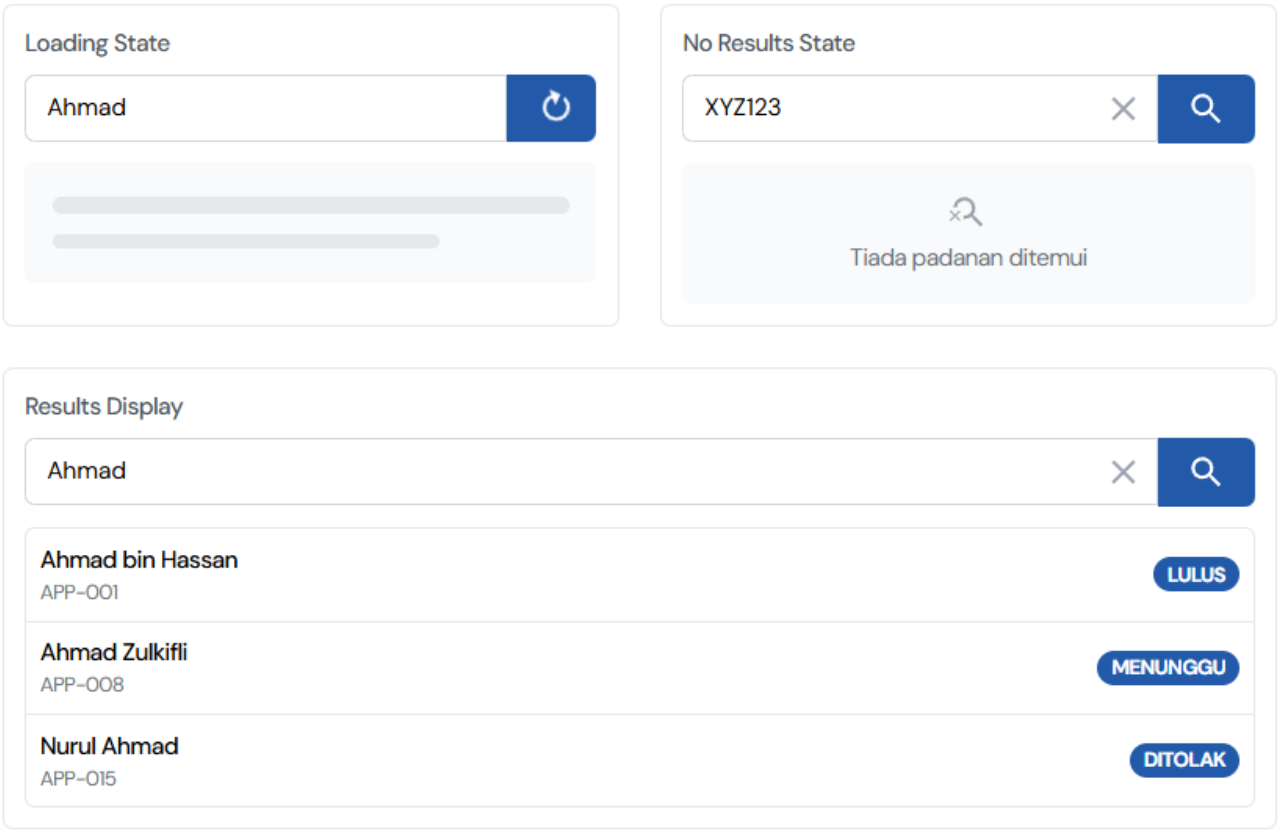


Search input validations for minimum characters, allowed characters, and valid examples.

13.2 FEEDBACK AND UX

Feedback is essential to inform users of the search state. Each search input must support:

- **Visual feedback** (spinner or shimmer) during result fetching.
- **No result state messaging** (“Tiada padanan ditemui”) if no matches are found.
- **Clear input (×)** icon to reset field instantly.
- Consistent layout behavior to prevent screen shifts or content reload flicker.



Search results interface with loading, empty state, and result display components.

Avoid scenarios where:

- Empty fields or invalid patterns still attempt to execute.
- No loading indicators lead users to believe the system is unresponsive.
- Irrelevant matches cause cognitive overload.

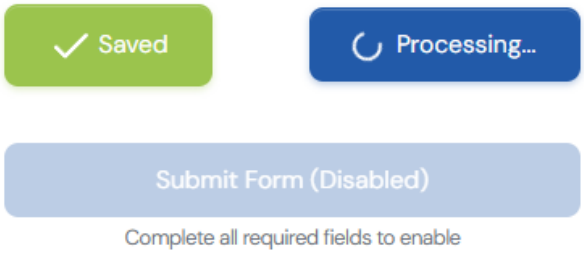
14 INTERACTION FEEDBACK PATTERNS

Interaction feedback refers to the visual or behavioral cues that inform users when their actions have been registered by the system. These small, yet essential patterns provide reassurance, improve responsiveness, and reduce uncertainty throughout the user journey. Whether it's a form submission, button click, or page transition, users should always receive immediate, meaningful feedback confirming that the system has responded.

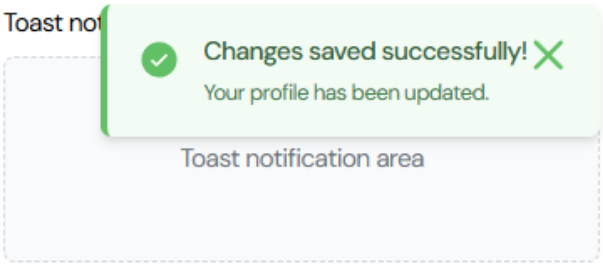
Interaction feedback patterns must be subtle yet clear, aligned with the overall design system, and consistently applied across the entire interface. They contribute to a polished, accessible experience while helping users understand system status and progress.

Use the following guidelines when designing interaction feedback:

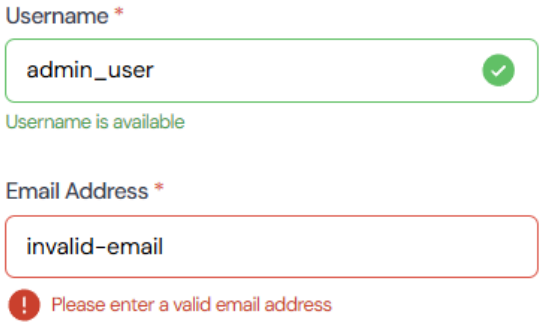
- Always provide instant visual feedback when a button is clicked, such as a loading spinner, disabled state, or color change.



- Use toast messages or inline confirmations to notify users of completed actions like saving, submitting, or updating.



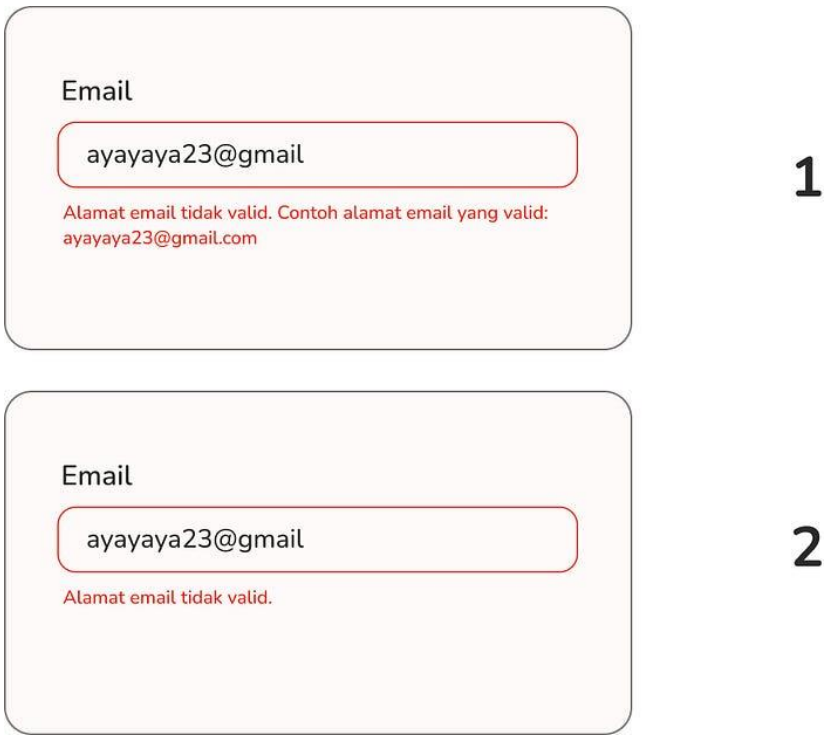
- Highlight errors or validation issues immediately after user action, using clear indicators and messages positioned near the problem.



- Apply loading indicators for any process that takes more than one second, whether it's data fetching, form submission, or content generation.
- Use motion sparingly and purposefully, such as smooth transitions for page loads or section reveals, to help guide user attention.
- For destructive actions, always provide a confirmation dialog with clear wording and distinguishable button styles.
- Provide micro-interactions like focus rings, active states, or ripple effects to show element responsiveness, especially on touch devices.
- Ensure feedback is accessible to all users by supporting screen reader alerts, keyboard navigation, and visible focus indicators.
- Avoid overloading the interface with unnecessary animations or alerts—only show feedback that adds clarity or relevance to the task.
- Keep feedback consistent across components, ensuring the same action always triggers the same visual response regardless of where it occurs in the system.

15 UX WRITING PRINCIPLES

UX writing plays a vital role in shaping how users understand, navigate, and interact with a digital product. The words used in buttons, error messages, labels, and notifications are not just content—they are part of the interface. Clear, thoughtful, and consistent writing helps users feel guided, confident, and supported throughout their journey. Good UX writing ensures clarity without overwhelming the user, especially in complex workflows or high-stakes environments such as registration, payment, or submission flows.



Comparison of two email validation error messages showing different levels of clarity and user guidance.

To maintain a consistent and user-focused experience, use the following UX writing principles across all modules and components:

- Use plain, conversational language that feels human, not robotic or overly formal.
- Write in sentence case, avoid all caps, and keep punctuation minimal unless required for clarity.
- Be concise—remove unnecessary words without sacrificing clarity or tone.
- Use active voice and action-driven verbs, especially for buttons and instructions.
- Ensure every piece of text supports user goals, not system logic—speak from the user's perspective.
- For error messages, state what went wrong and how to fix it using empathetic, non-blaming language.
- Maintain consistent terminology across screens; avoid mixing similar terms that may confuse the user.
- Use inclusive, respectful language and avoid cultural references that may not be universally understood.
- Where space is limited (e.g., mobile views), prioritize clarity over completeness.

- Validate all text for accessibility—ensure it can be read by screen readers and understood at a school reading level.

Example situations and proper UX writing in Malay:

Situation 1 – User forgot to enter their identification number

Incorrect: *Ralat 500: Data tidak lengkap!*

Correct: *Sila masukkan nombor kad pengenalan anda.*

Situation 2 – User attempted to submit a form without selecting the type of assistance

Incorrect: *Field kosong. Sila isi semua bahagian.*

Correct: *Sila pilih jenis bantuan yang ingin dipohon.*

Situation 3 – User successfully submitted the application form

Incorrect: *Submit Success!*

Correct: *Permohonan anda telah dihantar dengan berjaya.*

These examples reflect clear, user-friendly language that avoids technical errors, system jargon, or vague prompts. Each correction improves clarity, guides the user on what to do next, and aligns with the principles of accessibility, empathy, and consistency.

16 VISUAL STYLE GUIDELINES

A consistent and accessible visual style is essential for building user trust, improving usability, and ensuring a professional appearance across all modules. The following standards define the core visual elements for all user interfaces.

16.1 TYPOGRAPHY STANDARDS

Typography must establish a clear hierarchy and maximize readability. Use the Inter font family (with sans-serif fallback) for all text elements. Maintain strict consistency in font sizes, weights, and spacing to help users scan and comprehend content efficiently.

- Font Family: Use Inter, sans-serif for all UI text.

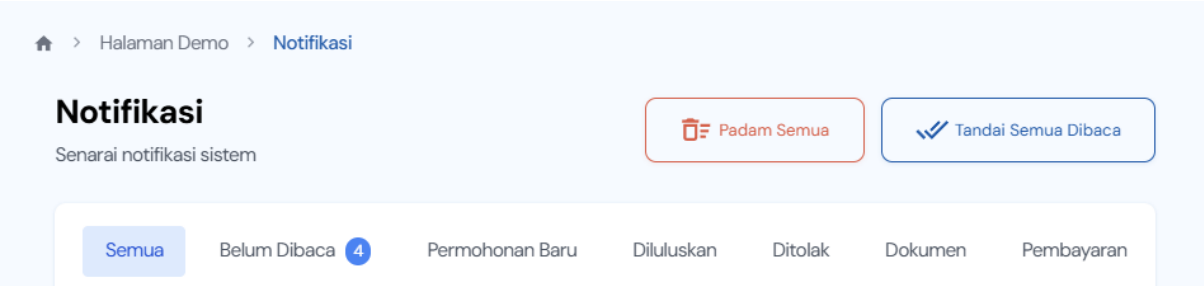
- Hierarchy:
- Page Titles (H1): 24px, bold (font-weight: 700). Example: "Dashboard Pengurusan Zakat"
- Section Headers (H2): 20px, semibold (font-weight: 600).
- Subsection Headers (H3): 16px, semibold.
- Body Text: 14px, regular (font-weight: 400).
- Captions/Helper Text: 12px, regular, with increased line height for legibility.
- Line Height: Minimum 1.5 for body text to improve readability.
- Letter Spacing: Use default or slightly increased spacing for headings to enhance clarity.
- Micro-Standard: Never use all-caps for body text; reserve for small labels or tags only.
- Accessibility: Ensure text is never rendered below 12px for any user-facing content.

Typography guideline reference showing font hierarchy, sizes, line heights, and font family standards.

16.2 BUTTON USAGE AND ACTION PLACEMENT

Buttons are a primary means of user interaction and must be implemented using the framework's <Button> component. The following standards ensure clarity, consistency, and accessibility:

- Button Types: Use the standard <Button> component for all actions. Button variants include primary, secondary, danger (for destructive actions), and text/outline styles for less prominent actions. The default size is medium; use small or large only when justified by layout or emphasis.
- Placement:
    - Primary Page Actions: Place the main action button (e.g., Add New, Export Data) at the top-right of the page content area.

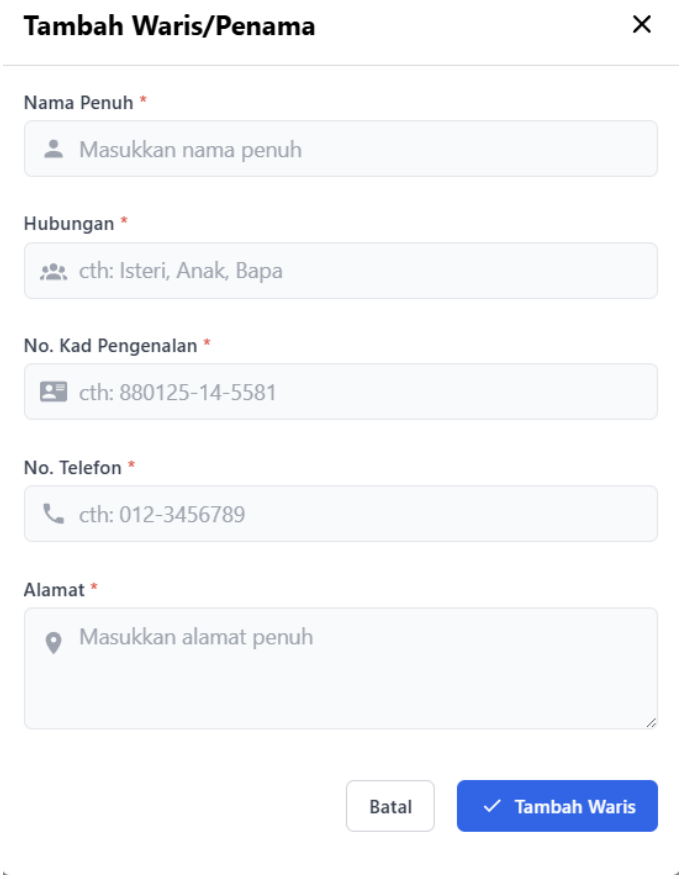


    - Form Actions: Place in a sticky <FormFooter> or at the bottom-right of the form container (<Card>) for short forms.

    - Table Row Actions: Right-align within the last column, using icon buttons for common actions (View, Edit, Delete).

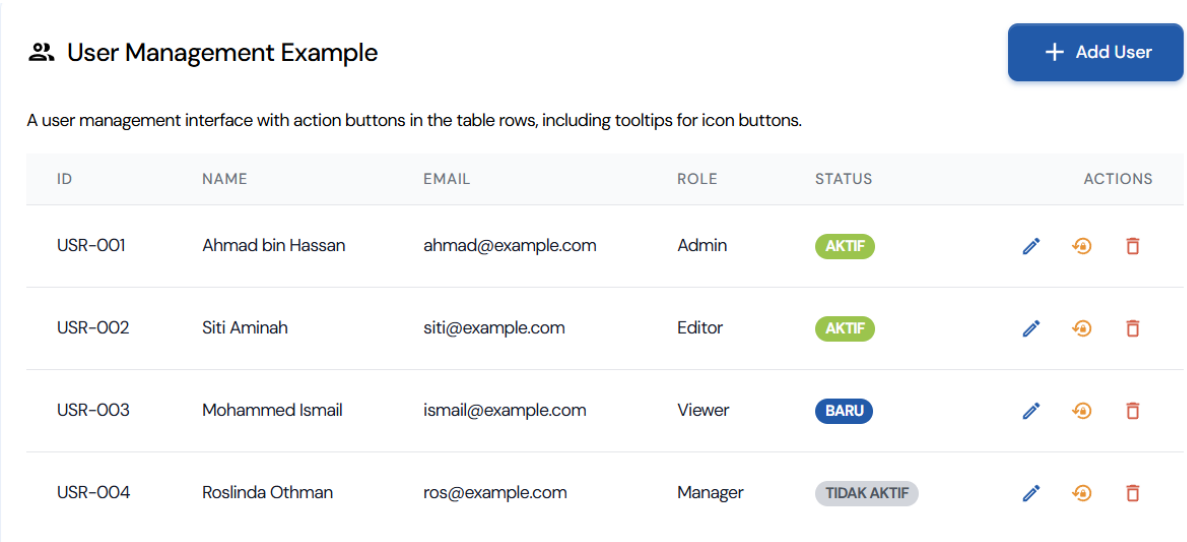
ID	NAMA	JENIS BANTUAN	STATUS	TINDAKAN
APP-001	Ahmad bin Hassan	Bantuan Perubatan	LULUS	Lihat
APP-002	Siti Aminah	Bantuan Pendidikan	MENUNGGU	Lihat
APP-003	Mohammed Ismail	Bantuan Rumah	DALAM PROSES	Lihat
APP-004	Roslinda Othman	Bantuan Perubatan	DITOLAK	Lihat

- Modal Actions: Bottom-right of the modal footer, with primary, secondary, and destructive (red) buttons ordered by importance.



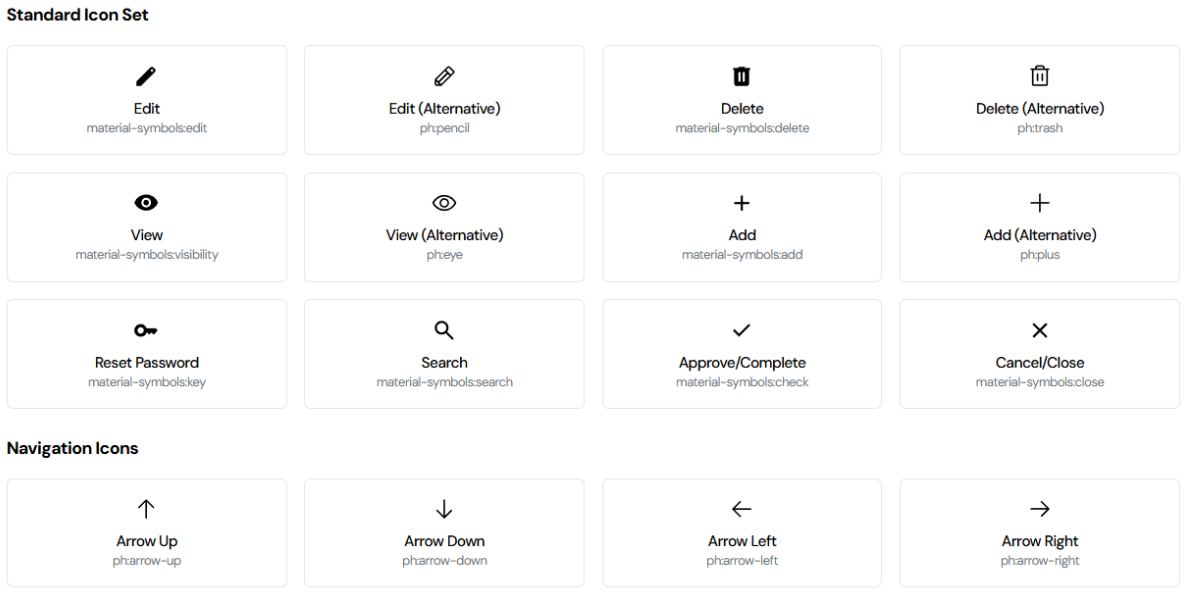
- Grouping: Group related actions together, maintaining 8px or 12px spacing between buttons. Order by importance: Primary → Secondary → Tertiary/Link.

- Labels: Use clear, action-driven text (e.g., Save, Submit, Delete). Use sentence case for button labels. Pair icons with text for clarity, especially for non-obvious actions.
- States: Indicate disabled, loading (with spinner), hover, and focus states. Use an inline spinner for loading states and disable the button during async operations.
- Prohibited: Do not use Floating Action Buttons (FABs) or scatter action buttons inconsistently across the page.
- When to Use Action Buttons:
    - Always provide a clear primary action on forms, modals, and index/listing pages.
    - Use secondary actions for less critical options (e.g., Cancel, Save as Draft).
    - For destructive actions (e.g., Delete, Remove), use a red/danger button and require confirmation.
    - In tables, use icon buttons for row-level actions and group more than three actions in a dropdown.
- Example:
    - On a user management page, place an Add User button at the top-right. In each table row, provide Edit (), Reset Password ( ), andDelete () as icon buttons, each with a tooltip



16.3 ICON USAGE AND SELECTON

Icons enhance clarity and speed of recognition for common actions. Use the IcoMoon or Material Icons set, as available in the codebase. Icons should be paired with text unless the action is universally understood (e.g., trash for delete, pencil for edit).

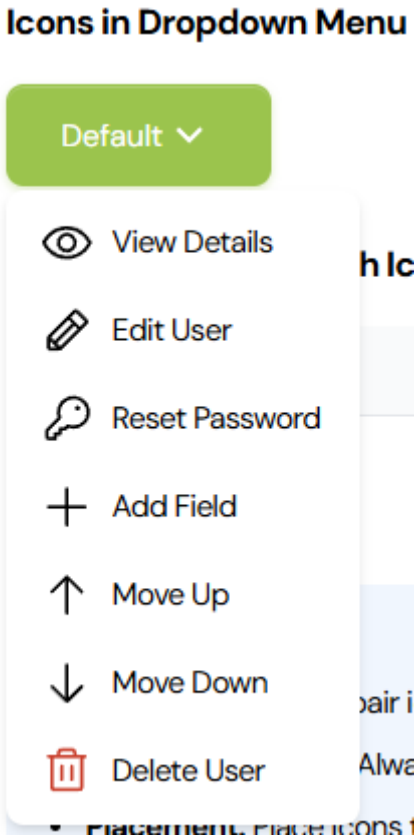


Standard icon set using IcoMoon and Material Icons for consistent action and navigation symbols.

- Standard Icons:
  - Edit: ph:pencil or material-symbols:edit
  - Delete: ph:trash or material-symbols:delete
  - View: ph:eye or material-symbols:visibility
  - Add: material-symbols:add or ph:plus
  - Reset Password: ph:key or material-symbols:key
  - Arrow/Navigation: ph:arrow-up, ph:arrow-down, ph:arrow-left, ph:arrow-right
  - Search: material-symbols:search
- Placement:
  - Place icons to the left of text in buttons (e.g., <Button><Icon name="material-symbols:add" /> Add User</Button>).
  - For icon-only buttons, always provide a tooltip and an accessible aria- label.
  - Size: Use consistent icon sizing (typically 16px or 20px for buttons, 24px for cards/headers).

- Color: Use the default icon color for normal actions. Use red for destructive actions and muted/grey for disabled states.
- Accessibility: All icons must have accessible labels or tooltips. Do not use color alone to convey meaning—pair with text or tooltips.

- Example:
  - In a dropdown menu, use ph:plus for "Add Field", ph:arrow-up/ph:arrow- down for sorting, and ph:x for delete, each with a tooltip.



Dropdown menu containing icons with action labels for user operations.

17 NAVIGATION COMPONENTS AND STANDARDS

Navigation components are essential for providing users with clear orientation, efficient movement throughout the application, and quick access to key actions. This section details the standards for implementing breadcrumbs, sidebars, headers, and related navigation/action elements to ensure consistency, accessibility, and optimal user experience.

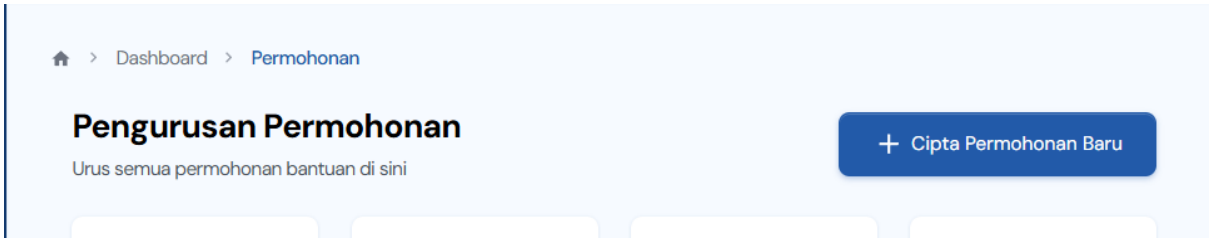
17.1 BREADCRUMBS: USAGE AND IMPLEMENTATION

Breadcrumbs are a critical navigational aid that help users understand their current location within the site hierarchy and provide a quick way to move back to parent pages. They must be implemented using the <Breadcrumbs> component and follow strict placement and accessibility rules.

- When to Use:
  - Display breadcrumbs on all pages that are more than one level deep in the site structure (e.g., detail, edit, or sub-list pages).
  - For multi-step workflows or wizards, include the current step as the final breadcrumb segment.
- Format and Content:
  - Structure breadcrumbs as Home / Parent Page / Current Page, with only the last segment (current page) rendered as plain text (not a link).
  - Use clear, human-readable, and concise names for each segment. Avoid abbreviations unless universally recognized.
- Placement:
  - Position breadcrumbs directly above the main page title (<h1>), left- aligned with the primary content area for visual consistency.
- Icons and Visuals:
  - Use a chevron or arrow icon (e.g., ic:outline-keyboard-arrow-right from IcoMoon or Material Icons) between breadcrumb segments for clear separation.
- Accessibility:
  - Wrap breadcrumbs in semantic markup: <nav aria- label="breadcrumb">.
  - Ensure all breadcrumb links are keyboard accessible and that the current page is properly indicated for screen readers.
- Micro-Behaviors:
  - Breadcrumbs should never wrap to multiple lines; truncate long segments with tooltips for full names.

- On mobile, consider collapsing intermediate segments with an ellipsis (...) for space efficiency.

- Example:
  On the "Edit User" page, display:
  `Home / Users / Edit User`
  where "Edit User" is plain text and not a link.

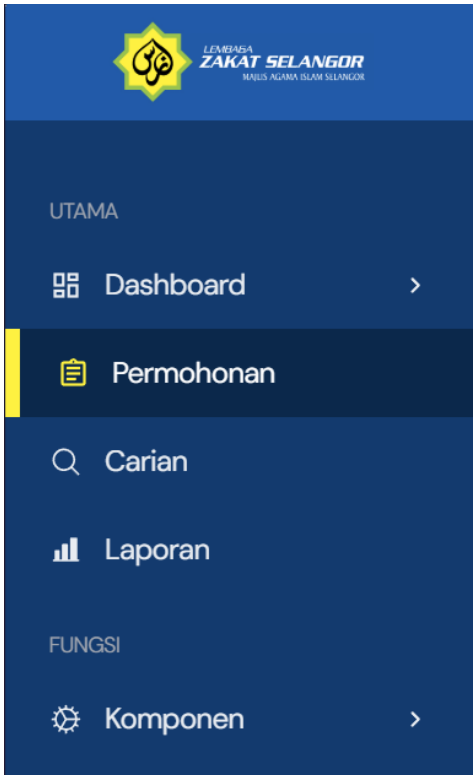


Breadcrumb navigation with page title and primary action button for creating a new application.

17.2 NAVIGATE AND ACTION COMPONENT STANDARDS

Consistent navigation and action components are essential for user orientation, efficiency, and accessibility. The following standards apply to all navigation and action- related UI elements. Where possible, always use the standard components from the internal UI framework to ensure uniformity and maintainability.

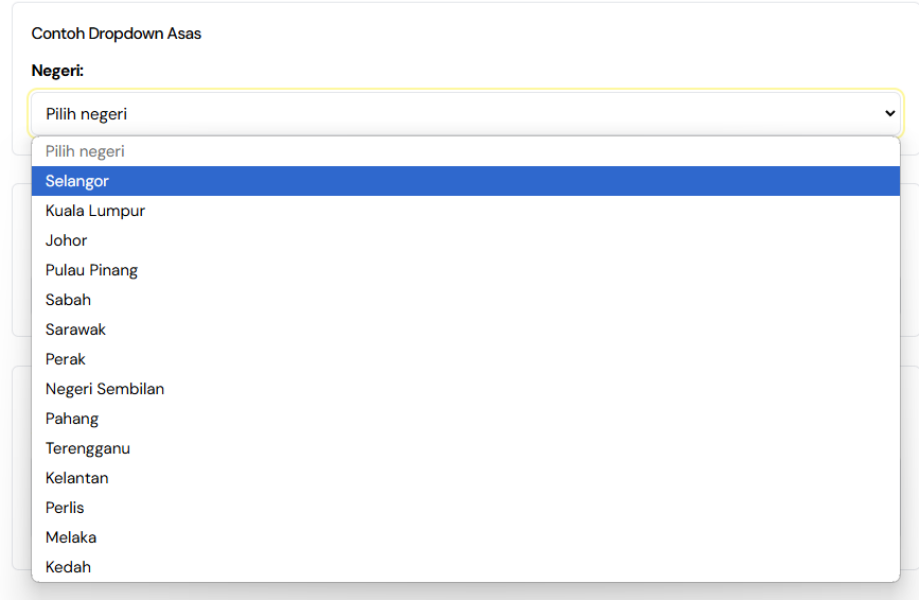
- Sidebar (<Sidebar>)
  - Use for primary navigation across the application.
  - Always visually highlight the current location (e.g., background highlight, bold text, or indicator bar).
  - Group related navigation items under clear section headers for better scanability.
  - On mobile, provide a collapsible sidebar with accessible toggle controls (ensure toggle is keyboard and screen reader accessible).
  - Sidebar navigation must be fully keyboard navigable and support screen reader landmarks.



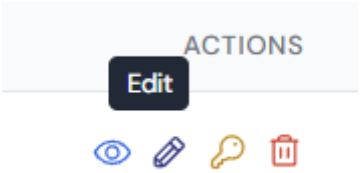
- Header (<Header>)
  - Place global actions (user account, notifications, global search) in the header.
  - Always display the current user's name or avatar, with quick access to profile/settings.
  - Ensure header actions are accessible via keyboard and have clear focus states.
  - On mobile, condense header actions into a menu or icon group for space efficiency.

- Dropdowns (<Dropdown>)
  - Use for grouped or secondary actions, especially in tables, toolbars, or card headers.
  - Each dropdown item must have a clear, concise label; include an icon where it aids recognition (use IcoMoon or Material Icons).
  - Dropdowns must be fully keyboard accessible: open/close with Enter/Escape, navigate with arrow keys, select with Enter/Space.

- Close dropdowns on outside click or Escape key.
- For destructive actions, use a red icon and confirmation dialog.



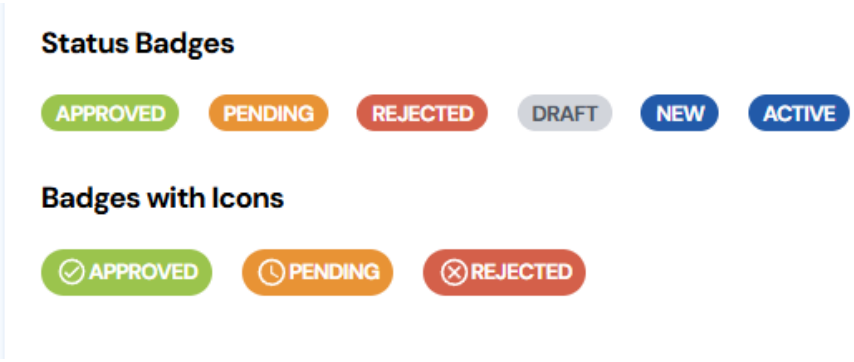
- Tooltips (<Tooltip>)
  - Use for icon-only buttons, truncated text, and supplementary hints.
  - Never use tooltips for critical instructions or error messages (these must be visible).
  - Tooltips must be accessible: trigger on keyboard focus and be announced by screen readers.
  - Position tooltips so they do not obscure related controls.



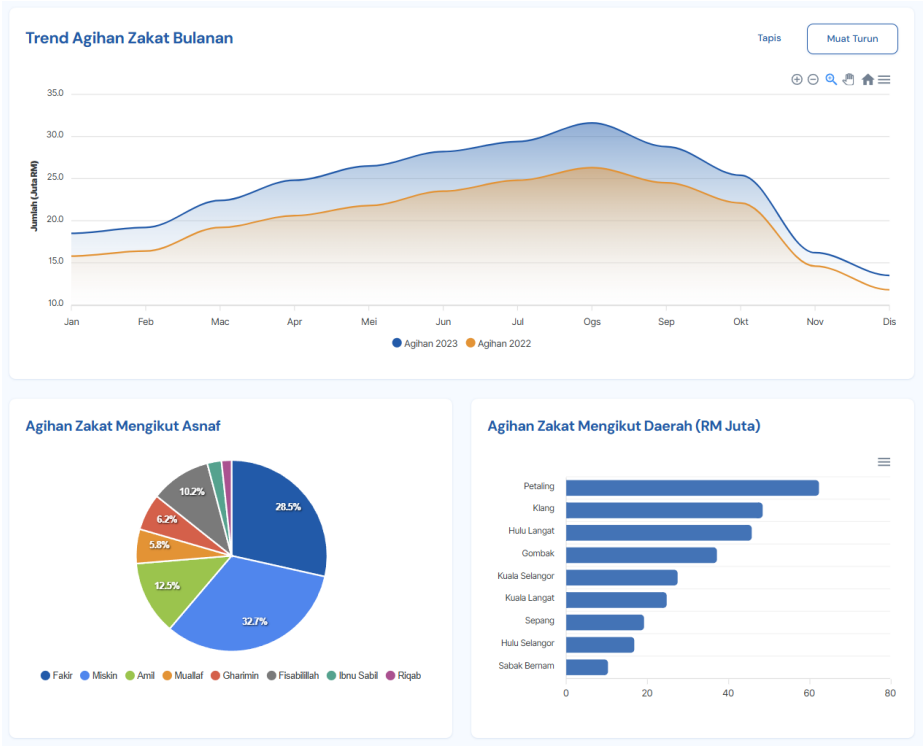
- Pagination (<Pagination>)
  - Place below tables or lists when content spans multiple pages.
  - Center or right-align pagination controls for visual balance.
  - Use clear previous/next icons (e.g., ph:arrow-left, ph:arrow-right), and always indicate the current page.

- Pagination controls must be keyboard accessible and have visible focus indicators.

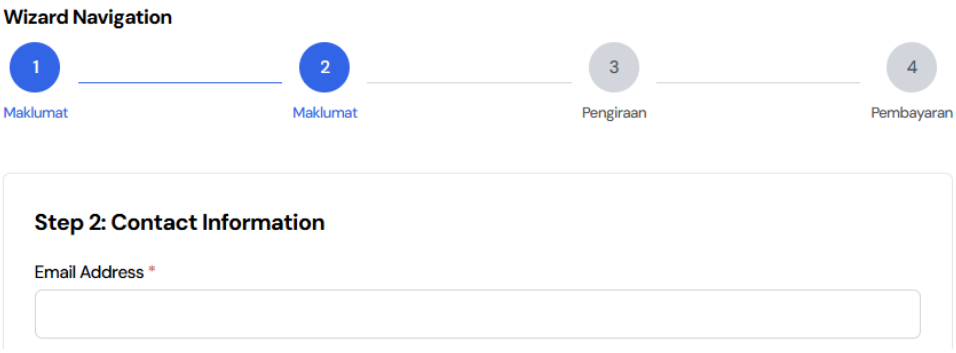
- Badges (<Badge>)
  - Use for status indicators in tables, cards, and lists.
  - Standard color mapping: green for APPROVED, red for REJECTED, yellow/orange for PENDING, etc.
  - Always pair badges with accessible text (visible or via aria-label) for screen readers.
  - Do not rely on color alone to convey meaning.



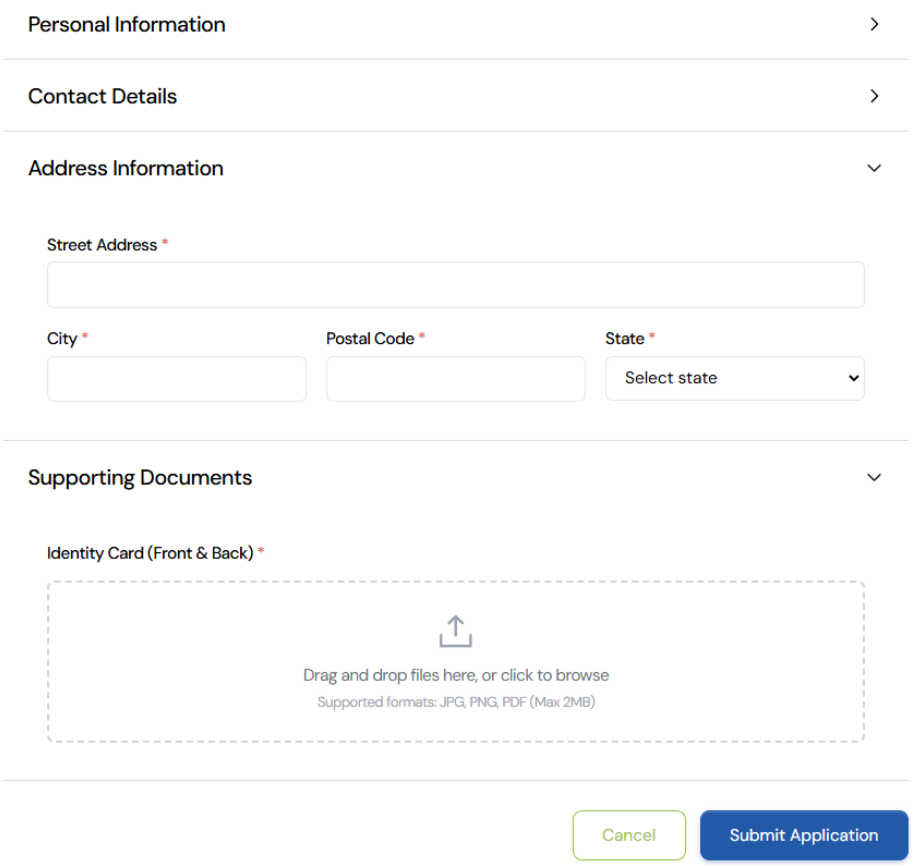
- Charts (<Chart>)
  - Use for data visualization; always include clear axis labels, legends, and accessible descriptions.
  - Ensure charts are colorblind-friendly (use patterns or textures in addition to color).
  - Provide a data table alternative or export option for accessibility and data review.



- Wizard (<Wizard>) and Accordion (<Accordion>)
  - Use <Wizard> for multi-step forms, showing progress and allowing navigation between steps (with clear step indicators).



  - Use <Accordion> for collapsible sections in long forms; only one section should be expanded at a time for clarity.



- Both components must be fully keyboard navigable and provide clear focus management.

- File Upload (<FileUpload>)
  - Use for document uploads, with clear status indicators (uploading, success, error).
  - Allow users to remove or replace files before submission.
  - Provide accessible error messages and ensure all controls are keyboard and screen reader accessible.

18 MOBILE APP SPECIFIC GUIDELINE

This section outlines mobile-specific guidance for how to use key interface components within the visual language shared across both web and mobile experiences.These examples are designed to help readers understand how, when, and why certain UI elements may be applied in mobile layouts—particularly when translating web-first designs into touch-friendly, single-column experiences. The objective is to ensure that every component used in mobile views reinforces clarity, responsiveness, and ease of interaction.

*The visuals referred to in this documentation are example concepts only, meant to demonstrate possible situational use of components in a mobile context and not the final designs. Actual implementation will follow specific project requirements, system constraints, and stakeholder approva*l.

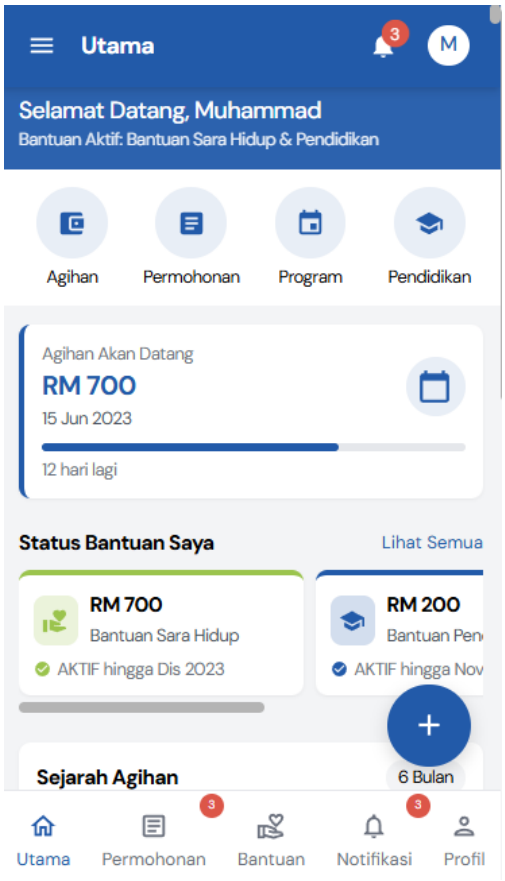
18.1 NAVIGATION STRUCTURE ON MOBILE

Navigating on mobile should feel effortless and anchored. The structure must support short attention spans, single-hand use, and fast reorientation. This can be achieved by anchoring key actions and sections through consistent navigation zones that remain available without disrupting the user's context.

- Avoid overloading the interface with too many entry points. Limit to essential options that matter most on mobile.

- Keep navigation reachable at all times, especially at the bottom edge where thumbs naturally rest.
- Clearly separate navigation from content. Do not allow it to blend or interfere with the main content area.
- Maintain consistent active state visuals—users should always know where they are.
- Use labels alongside icons to reduce ambiguity and improve clarity for new or infrequent users.
- Any additional quick access elements (like counters or profile access) should remain subtle and non-disruptive.

This approach ensures users remain in control of their journey, even if they return days or weeks later.



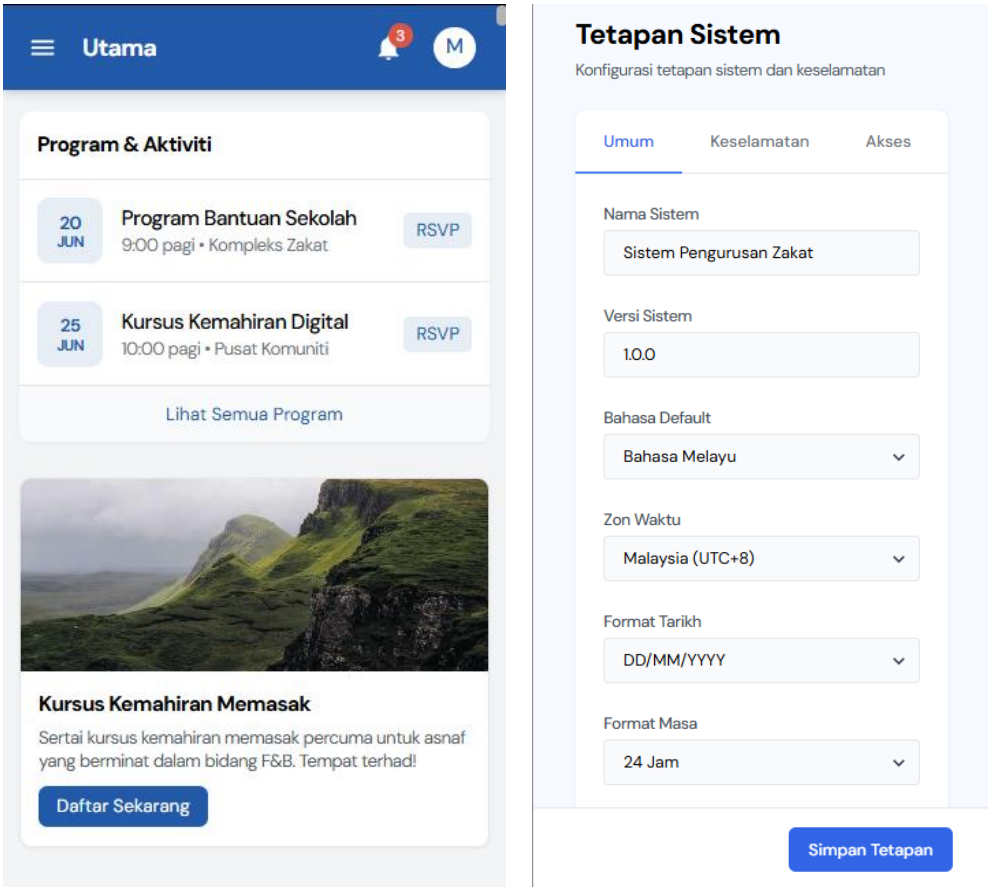
Mobile homepage concept with bottom navigation bar, top greeting header, quick access icons, and segmented assistance status cards.

18.2 LAYOUT AND CARD COMPOSITION

Cards are one of the most versatile components on mobile. They offer a containerized experience where related information and actions are visually grouped. When used correctly, they support visual clarity and encourage faster scanning.

- Every card should feel like a self-contained block with a single, clear focus. Avoid combining too many unrelated items in one card.
- Use generous spacing around and inside each card to allow the eye to breathe. On small screens, air is clarity.
- If content is presented vertically, use logical stacking—support left-aligned structures for readability.
- Cards must avoid visual crowding. Secondary actions or supporting information should not compete with the main message.
- When stacking multiple cards in a scroll, maintain consistent padding and spacing to reinforce rhythm.
- Headings, metadata, and call-to-action buttons should follow predictable placement across all cards.

A clean, intentional card layout keeps the experience feel purposeful, light, and digestible—even with dense content.



Mobile cards showcase for program listings, promotional content, and system settings form layout.

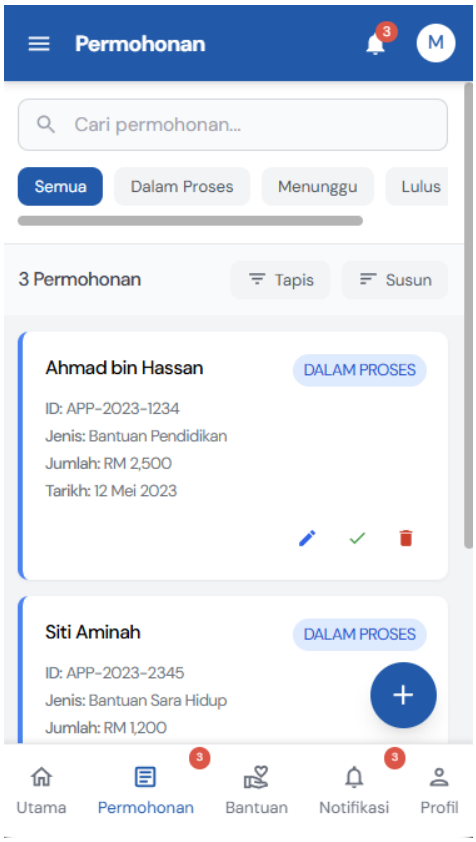
18.3 LISTS, FILTERS & PROGRESSIVE DISCLOSURE

List-based content on mobile should prioritize visibility, simplicity, and progressive detail. Filters and tabs may support this, but only when they clarify, not complicate. Lists should offer a scan-friendly overview with the ability to dig deeper when necessary.

- Keep list items concise. Each item should present just enough information to spark interest, not overwhelm.
- Use spacing and dividers to separate items clearly without relying on boxes or borders everywhere.
- Avoid loading the top of the screen with too many filter chips or tabs. Only show what is essential in mobile view.
- Always position the most relevant sorting or filtering tools within easy reach—typically above the list.

- Enable progressive disclosure. Don't present every detail upfront; allow users to explore or act as needed.
- Interactive elements within list items (like actions) must be visually balanced—not too large, not too subtle.

This ensures that the list feels alive and responsive without demanding too much decision-making upfront.



Mobile list of applications with filter tabs, action icons, and progressive detail for each item.

18.4 MOBILE-FRIENDLY DETAIL CARDS

On mobile, detail views must be sharp, easy to scroll, and segmented. Card-style presentation supports this well, as it allows information to be grouped and read at a glance—especially important for on-the-go users.

- Use visual grouping to separate different types of information. For example, current state, categories, and timing should not be mixed in a single visual line.
- Maintain a clean vertical flow—users should never need to pause and decode.

- Allow for scroll, but break content visually so users don’t lose their place.
- Buttons inside these sections should be positioned after the content they relate to—not before.
- Don't overload any one card with mixed styling (e.g. badge + button + label + icon) unless they are clearly aligned.
- All UI elements must remain touchable with appropriate spacing and clear hit zones.

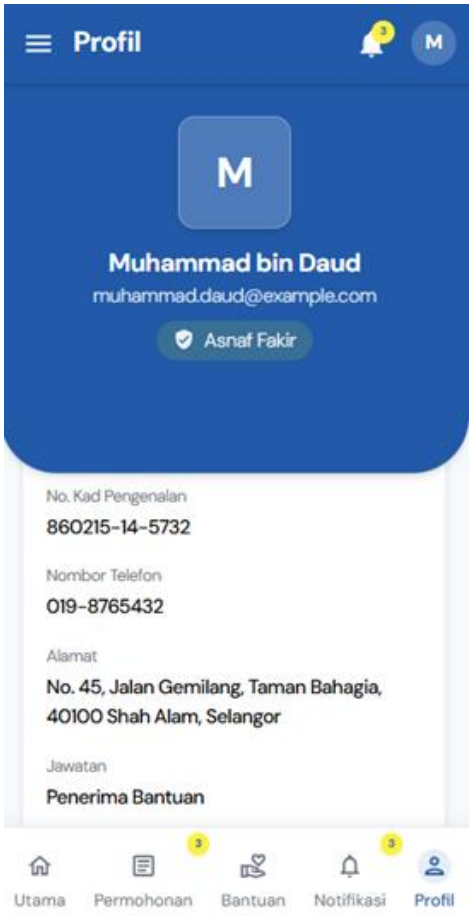
This approach gives every detail its breathing space, reducing the risk of tap errors or visual confusion.

18.5 PROFILE AND IDENTITY SECTIONS

The profile section must serve as a reliable reference point—one that feels personal, consistent, and easy to update. It should prioritize hierarchy, with clear labeling and structured metadata.

- Use top-aligned user identity visuals to reinforce personalization.
- Avoid mixing role badges or status tags directly with user data. Present them in a dedicated, clearly styled area.
- Display user data in a clean list format—no visual fluff, just clarity.
- Leave sufficient padding between data fields for legibility.
- Keep the most editable fields visually separate from the rest to avoid accidental interaction.
- Avoid using borders for every line. Rely on spacing and subtle typography changes to guide the eye.

A well-balanced profile area builds trust and makes users feel grounded in the system.



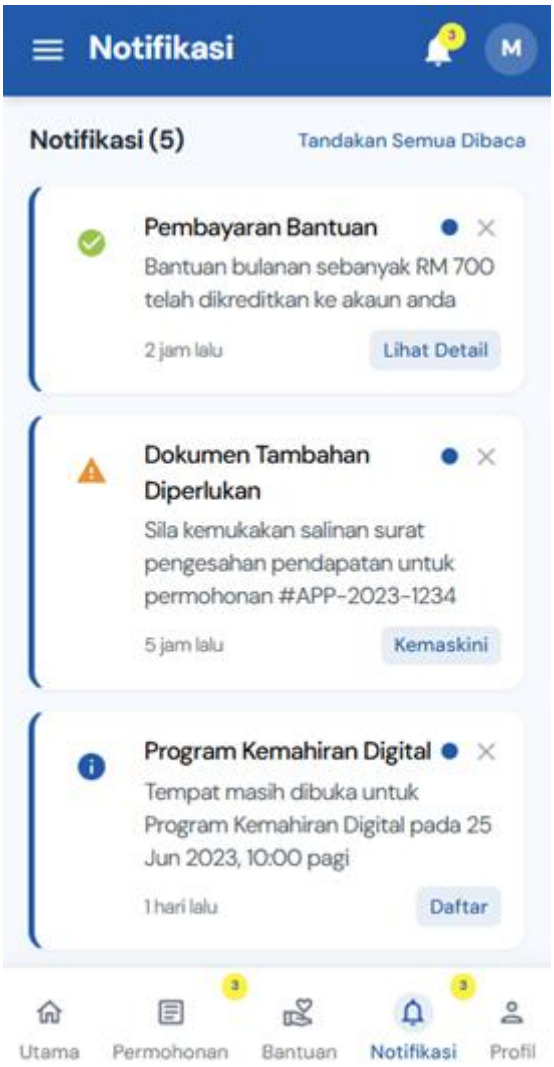
Mobile user profile with identity header, role badge, and structured personal details in vertical layout.

18.6 NOTIFICATION CARDS & STATUS FEEDBACK

Feedback and notifications on mobile must balance urgency and readability. Visual status indicators, contextual messages, and lightweight cards help deliver information without disrupting the experience.

- Use a vertical list of cards to display updates in digestible chunks.
- Include subtle status indicators (e.g., visual icons, badges) to denote the type of message at a glance.
- Allow users to dismiss items or act on them without navigating away.
- Keep interaction lightweight. Use one or two call-to-action buttons, no more.
- Ensure alerts feel timely, not repetitive. Avoid stacking too many of the same type.
- Never use overly vibrant or alarming color unless the context truly demands attention.

By presenting feedback gracefully, users feel informed and empowered, not interrupted or overwhelmed.



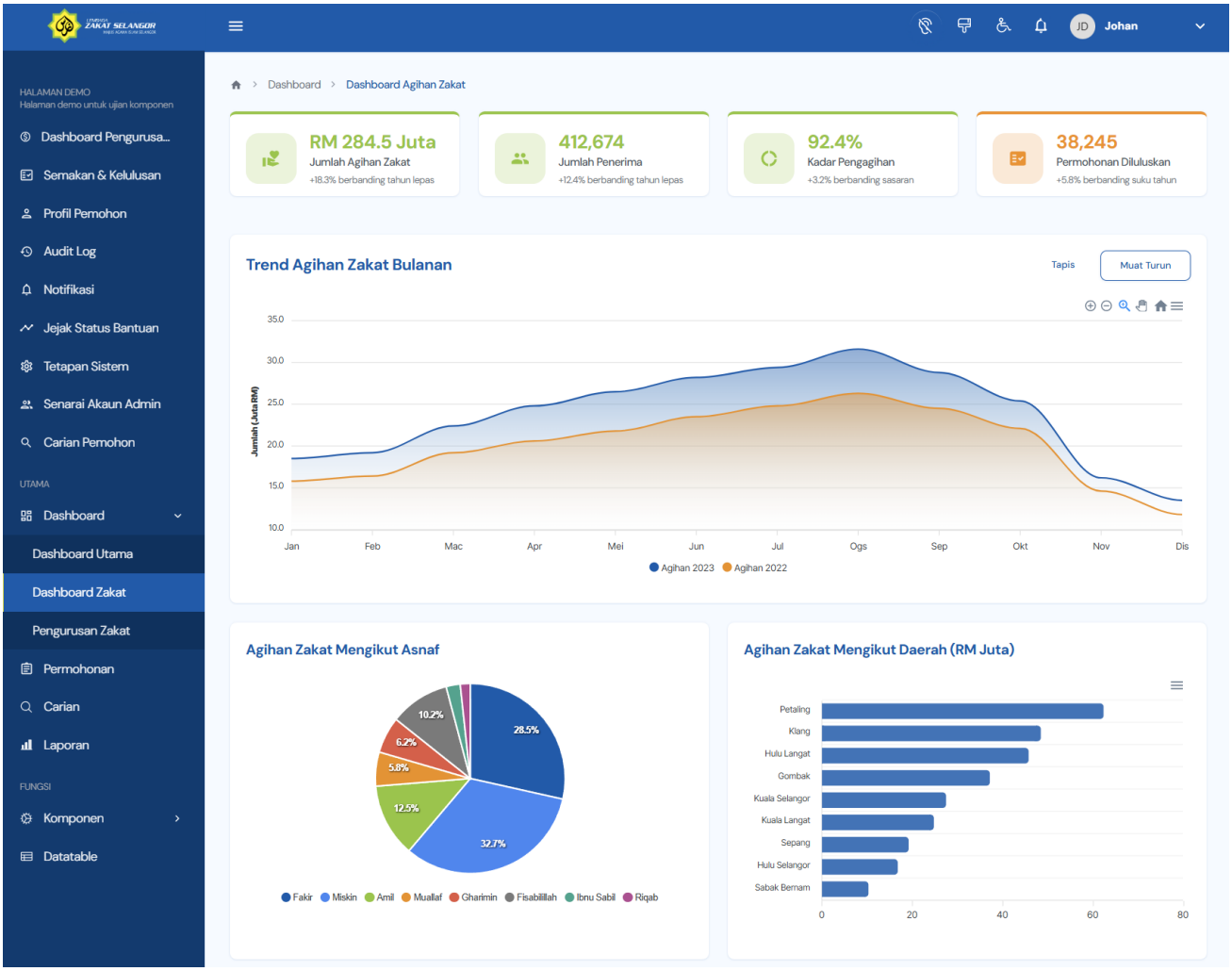
Mobile notification cards with status icons, timestamps, and inline action buttons.

19 RESPONSIVENESS & DEVICE TESTING

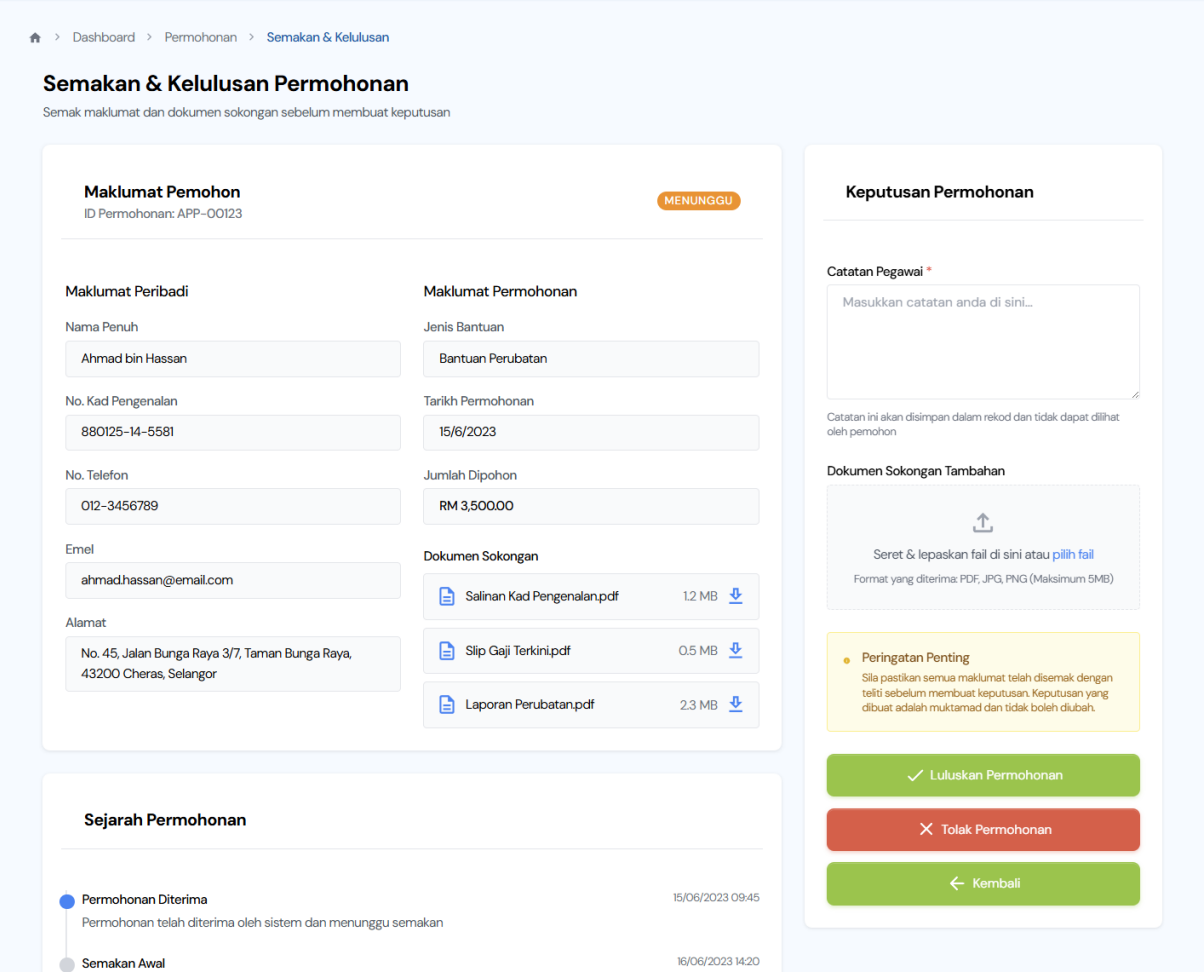
The system UI/UX must be tested for consistency and usability across the following devices:

- Desktop Browsers: Chrome, Microsoft Edge, Safari
- Tablets: iPad and Android devices 10 inches and above
- Mobile Phones: iOS and Android (multiple screen sizes)

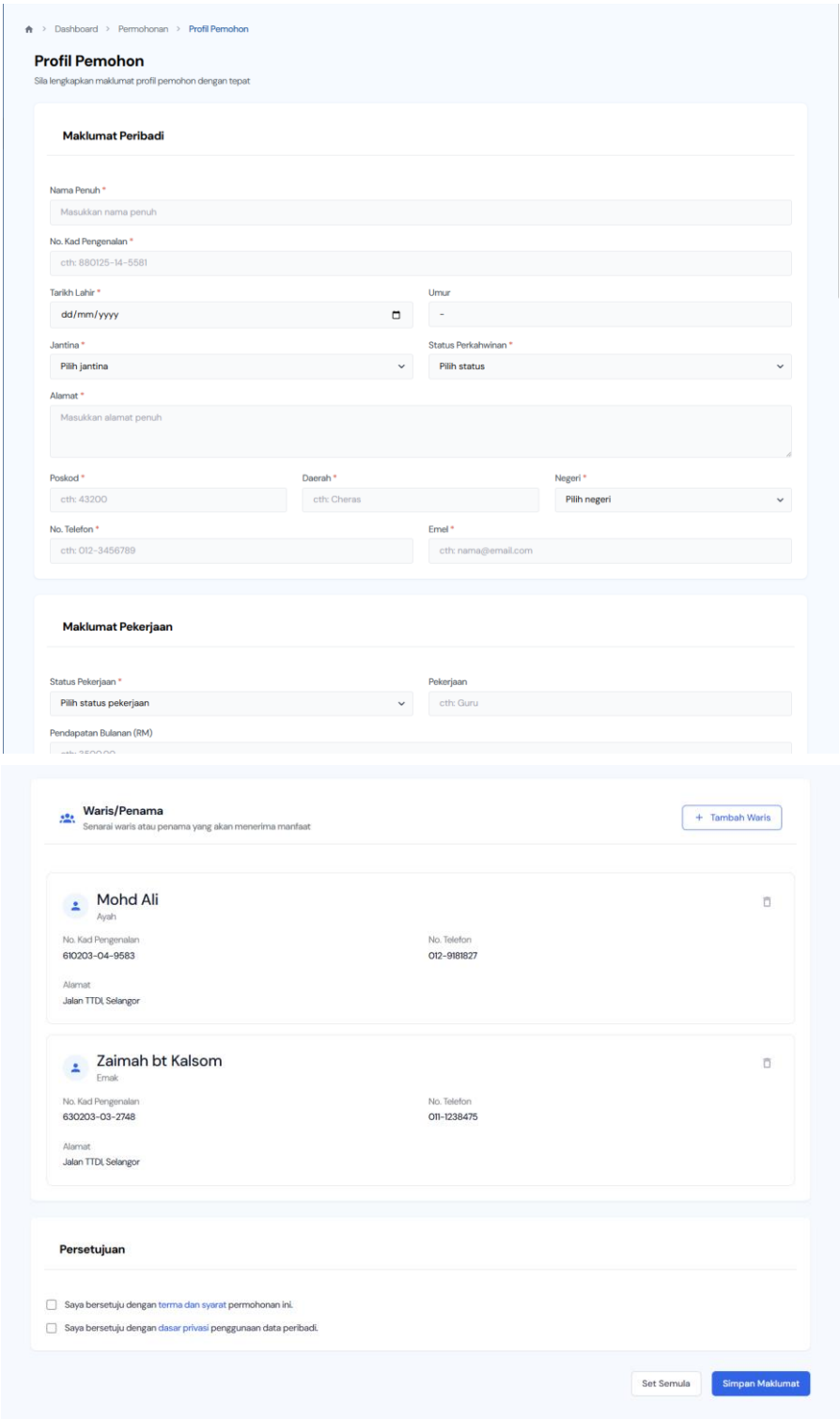
20 ANNEX



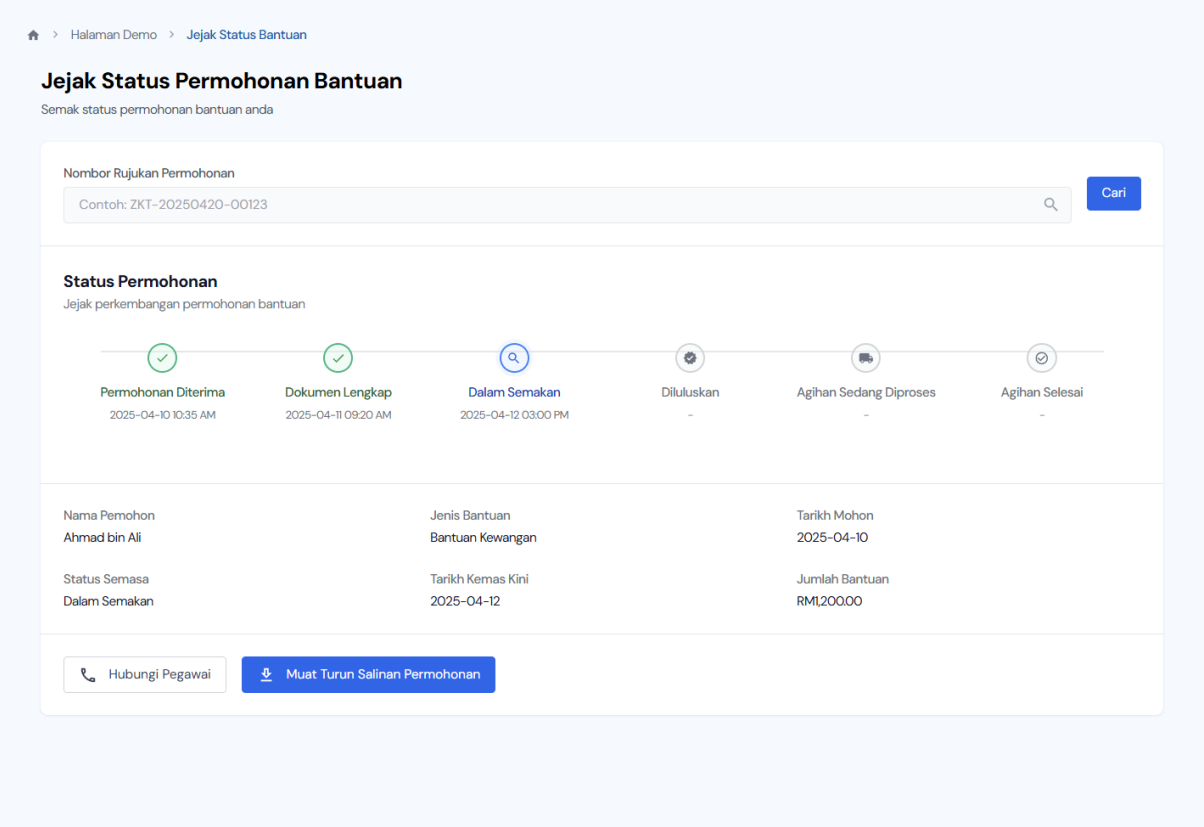
Conceptual dashboard screen with full sidebar navigation, KPI summary cards, and embedded charts for monthly trends, asnaf distribution, and regional breakdown (not final design).



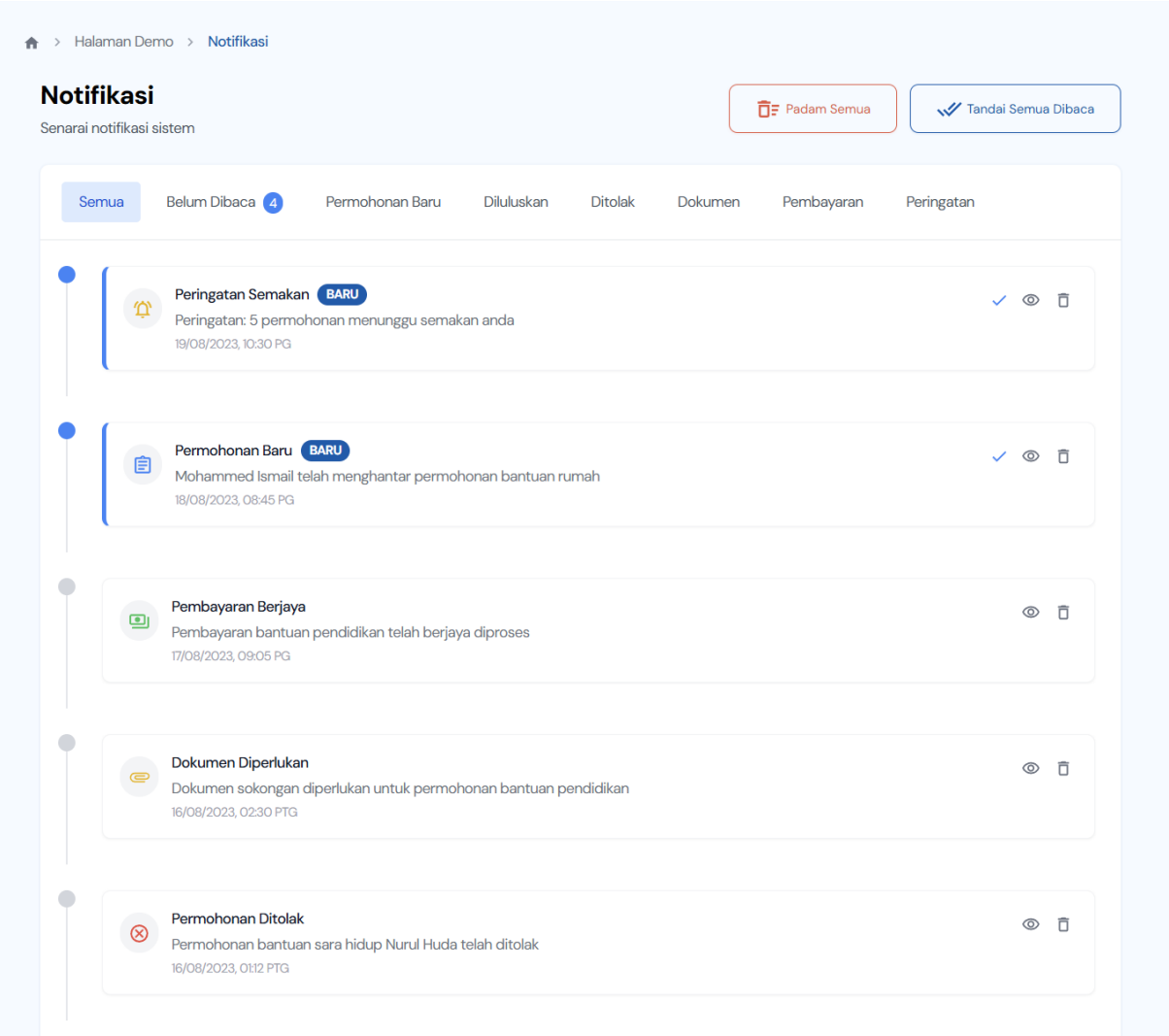
Conceptual review screen with applicant details, documents, notes, and decision actions



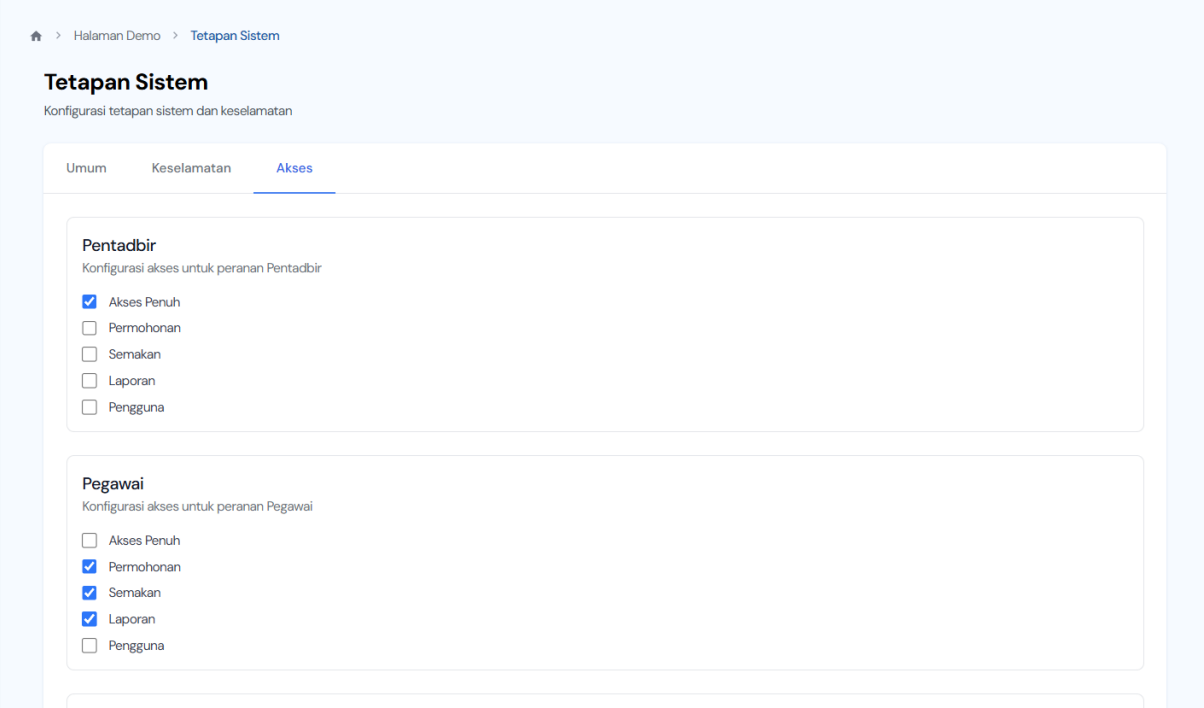
Conceptual applicant profile form with grouped personal, employment, beneficiary, and consent sections



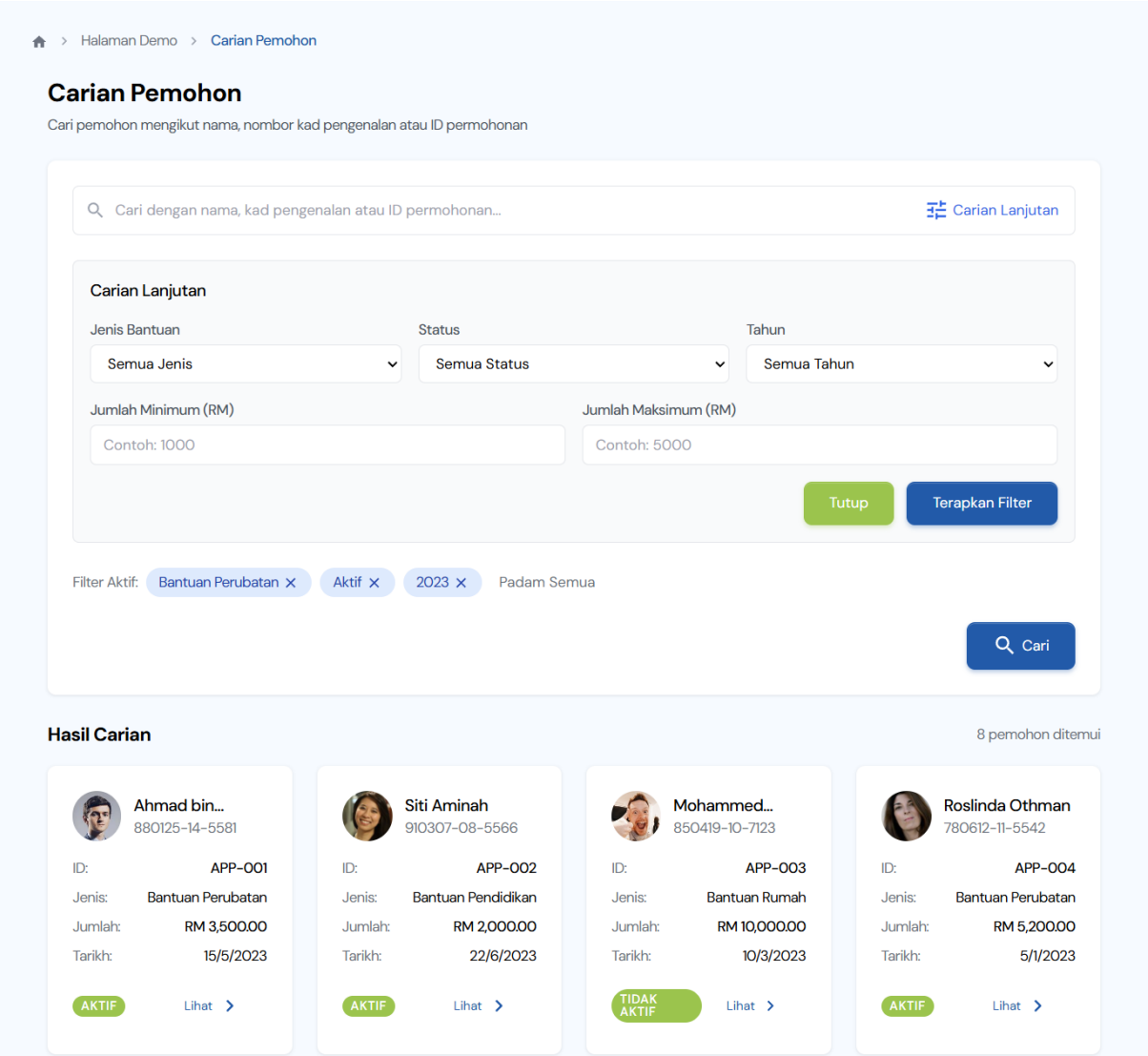
Conceptual tracking screen for bantuan application status with timeline progress, applicant details, and support actions



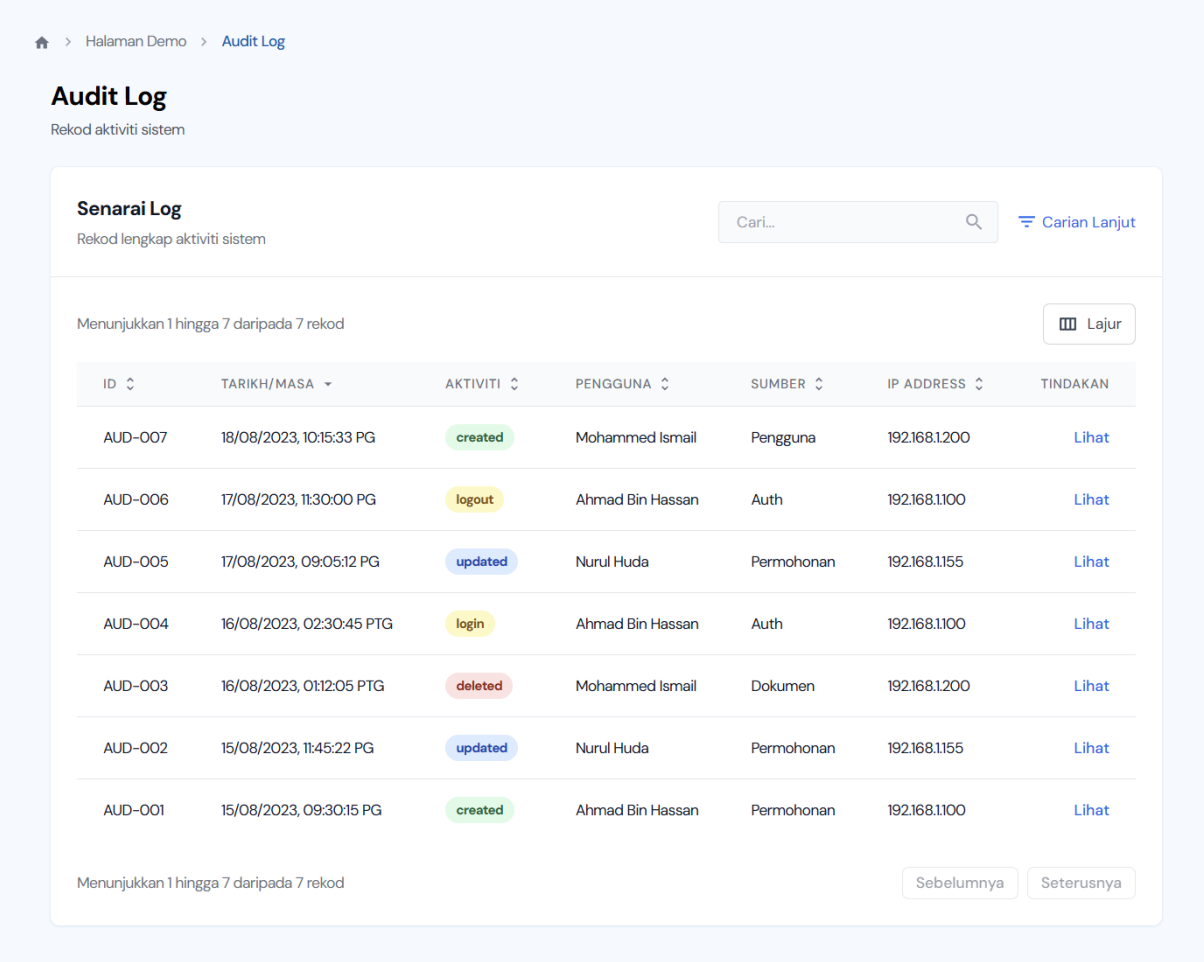
Conceptual notification list with categorized tabs, status indicators, and inline actions



Conceptual access settings screen with role-based checkbox permissions grouped by user role



Conceptual applicant search interface with advanced filters, tag-based active filters, and card-style search results



Conceptual audit log table displaying user activities with sorting, column controls, and status badges