Design Systems for AI

Design_System.md

A comprehensive guide on how to structure and document a design system for AI.

Table of Contents

This is a "Show, don't tell" document

The document here *shows* a structure that is familiar to AI. It's contents are just enough for AI to produce consistent output using your design system. This is important because your design system codebase and/or Figma design system file via MCP is overwhelming for AI. You can read why this may result in inconsistent output in our **Context, Context, Context article**.

Download
PDF

Download
Markdown

Beginning of the specification for AI

package.json

```
{
  "name": "aidesignsystem",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start"
  },
  "dependencies": {
    "lucide-react": "^0.577.0",
    "next": "16.1.6",
    "react": "19.2.3",
    "react-dom": "19.2.3",
    "recharts": "^3.8.0"
  },
  "devDependencies": {
    "@tailwindcss/postcss": "^4",
```

```
    "tailwindcss": "^4"
  }
}
```

Component Inventory

All components are exported from the barrel file and can be imported individually.

Component	Import	Other imports as necessary
Button	`import { Button } from 'my_ds_name'`	`import Button from 'my_ds_name/Button/Button'`
TextInput	`import { TextInput } from 'my_ds_name'`	`import TextInput from 'my_ds_name/TextInput/TextInput'`
Checkbox	`import { Checkbox } from 'my_ds_name'`	`import Checkbox from 'my_ds_name/Checkbox/Checkbox'`
Toggle	`import { Toggle } from 'my_ds_name'`	`import Toggle from 'my_ds_name/Toggle/Toggle'`
Select	`import { Select } from 'my_ds_name'`	`import Select from 'my_ds_name/Select/Select'`
Dropdown	`import { Dropdown } from 'my_ds_name'`	`import Dropdown from 'my_ds_name/Dropdown/Dropdown'`
DatePicker	`import { DatePicker } from 'my_ds_name'`	`import DatePicker from 'my_ds_name/DatePicker/DatePicker'`
Search	`import { Search } from 'my_ds_name'`	`import Search from 'my_ds_name/Search/Search'`
Tag	`import { Tag } from 'my_ds_name'`	`import Tag from 'my_ds_name/Tag/Tag'`

Component	Import	Other imports as necessary
Modal	`import { Modal } from 'my_ds_name'`	`import Modal from 'my_ds_name/Modal/Modal'`
DataTable	`import { DataTable } from 'my_ds_name'`	`import DataTable from 'my_ds_name/DataTable/DataTable'`
Pagination	`import { Pagination } from 'my_ds_name'`	`import Pagination from 'my_ds_name/Pagination/Pagination'`
Tabs	`import { Tabs } from 'my_ds_name'`	`import Tabs from 'my_ds_name/Tabs/Tabs'`
Header	`import { Header } from 'my_ds_name'`	`import Header from 'my_ds_name/Header/Header'`
SideNav	`import { SideNav } from 'my_ds_name'`	`import SideNav from 'my_ds_name/SideNav/SideNav'`
Breadcrumb	`import { Breadcrumb } from 'my_ds_name'`	`import Breadcrumb from 'my_ds_name/Breadcrumb/Breadcrumb'`
OverflowMenu	`import { OverflowMenu } from 'my_ds_name'`	`import OverflowMenu from 'my_ds_name/OverflowMenu/OverflowMenu'`
Notification	`import { Notification } from 'my_ds_name'`	`import Notification from 'my_ds_name/Notification/Notification'`
Toast	`import { Toast } from 'my_ds_name'`	`import { Toast } from 'my_ds_name/Notification/Notification'`
Banner	`import { Banner } from 'my_ds_name'`	`import { Banner } from 'my_ds_name/Notification/Notification'`
Form	`import { Form } from 'my_ds_name'`	`import Form from 'my_ds_name/Form/Form'`
FormGroup	`import { FormGroup } from 'my_ds_name'`	`import { FormGroup } from 'my_ds_name/Form/Form'`

Component	Import	Other imports as necessary
FormRow	`import { FormRow } from 'my_ds_name'`	`import { FormRow } from 'my_ds_name/Form/Form'`
FormActions	`import { FormActions } from 'my_ds_name'`	`import { FormActions } from 'my_ds_name/Form/Form'`
Spinner	`import { Spinner } from 'my_ds_name'`	`import { Spinner } from 'my_ds_name/Loading/Loading'`
Skeleton	`import { Skeleton } from 'my_ds_name'`	`import { Skeleton } from 'my_ds_name/Loading/Loading'`
SkeletonText	`import { SkeletonText } from 'my_ds_name'`	`import { SkeletonText } from 'my_ds_name/Loading/Loading'`
TableSkeleton	`import { TableSkeleton } from 'my_ds_name'`	`import { TableSkeleton } from 'my_ds_name/Loading/Loading'`

Component Props

Every component accepts `className` and forwards extra props via `...props`. All components use `forwardRef`.

_Button

```
<Button
  variant="primary"       // "primary" | "secondary" |
"tertiary" | "danger" | "ghost"
  size="md"               // "sm" | "md" | "lg"
  disabled={false}
  loading={false}         // shows Loader2 spinner, disables
button
```

```
  icon={<IconComponent />}
  iconPosition="left"     // "left" | "right"
  fullWidth={false}
  type="button"           // "button" | "submit" | "reset"
  onClick={fn}
/>
```

_TextInput

```
<TextInput
  label="Email"
  placeholder="you@example.com"
  size="md"               // "sm" | "md" | "lg"
  disabled={false}
  required={false}
  helperText="We'll never share your email."
  errorText="Invalid email"   // triggers error state (red
border + icon)
  successText="Looks good!"   // triggers success state
(green border + icon)
  icon={<IconComponent />}    // left icon
  wrapperClassName=""
  // All native <input> props: value, onChange, type, name,
etc.
/>
```

_Select (native)

```
<Select
  label="Country"
  options={[                  // string[] or { value,
label }[]
    { value: 'us', label: 'United States' },
```

```
    { value: 'uk', label: 'United Kingdom' },
  ]}
  value="us"
  onChange={fn}
  placeholder="Select an option"
  size="md"                       // "sm" | "md" | "lg"
  disabled={false}
  required={false}
  errorText=""
  helperText=""
  wrapperClassName=""
/>
```

_Dropdown (custom styled, non-native)

```
<Dropdown
  label="Role"
  options={[                      // string[] or { value,
label, disabled? }[]
    { value: 'admin', label: 'Admin' },
    { value: 'user', label: 'User' },
  ]}
  value="admin"
  onChange={(value) => {}}
  placeholder="Select..."
  size="md"                       // "sm" | "md" | "lg"
  disabled={false}
  required={false}
  errorText=""
  helperText=""
  wrapperClassName=""
/>
```

_Checkbox

```
<Checkbox
  label="Accept terms"
  checked={false}
  indeterminate={false}    // shows minus icon (mixed state)
  disabled={false}
  size="md"                // "sm" | "md" | "lg"
  onChange={fn}
/>
```

_Toggle

```
<Toggle
  label="Dark mode"
  checked={false}
  disabled={false}
  size="md"               // "sm" | "md" | "lg"
  onChange={fn}
/>
```

_Modal

```
<Modal
  open={false}
  onClose={fn}
  title="Confirm Action"
  size="md"               // "sm" | "md" | "lg" | "xl" |
"full"
  closeOnOverlay={true}
  closeOnEsc={true}
  showCloseButton={true}
```

```
  footer={<>
    <Button variant="tertiary" onClick=
{onClose}>Cancel</Button>
    <Button onClick={onConfirm}>Confirm</Button>
  </>}
>
  <p>Modal body content here.</p>
</Modal>
```

Note: Modal renders via `createPortal` to `document.body`. It locks body scroll while open.

_Tag

```
<Tag
  color="default"          // "default" | "brand" |
"success" | "warning" | "danger" | "info"
  size="md"                // "sm" | "md" | "lg"
  dismissible={false}
  onDismiss={fn}
  icon={<IconComponent />}
  outline={false}          // outline variant: transparent
bg with colored border
>
  Label
</Tag>
```

_Notification (inline)

```
<Notification
  type="info"              // "success" | "warning" |
"error" | "info"
  title="Heads up"
  dismissible={true}
```

```
  onDismiss={fn}
>
  Descriptive message body.
</Notification>
```

_Toast (floating)

```
<Toast
  type="success"           // "success" | "warning" |
"error" | "info"
  title="Saved"
  visible={true}
  duration={5000}          // ms, 0 = persistent
  onClose={fn}
>
  Optional body text.
</Toast>
```

Renders fixed at bottom-right.

_Banner (full-width)

```
<Banner
  type="warning"           // "success" | "warning" |
"error" | "info"
  dismissible={true}
  onDismiss={fn}
>
  System maintenance scheduled for tonight.
</Banner>
```

_DataTable

```
<DataTable
  columns={[
    { key: 'name', header: 'Name', sortable: true, width:
'200px' },
    { key: 'email', header: 'Email' },
    { key: 'status', header: 'Status', render: (val, row) =>
<Tag>{val}</Tag> },
  ]}
  data={[
    { id: 1, name: 'Alice', email: 'alice@co.com', status:
'Active' },
  ]}
  sortable={true}
  defaultSortColumn="name"
  defaultSortDirection="asc"    // "asc" | "desc"
  onSort={(column, direction) => {}}
  selectable={false}
  selectedRows={[]}             // array of row indices
  onSelectionChange={(indices) => {}}
  batchActions={<Button size="sm">Delete</Button>}  // shown
when rows selected
```

_Pagination

```
<Pagination
  currentPage={1}
  totalPages={10}
  totalItems={100}
  pageSize={10}
  onPageChange={(page) => {}}
  onPageSizeChange={(size) => {}}
  pageSizeOptions={[10, 25, 50, 100]}
  siblingCount={1}
  showPageSizeSelector={false}
```

```
  showItemCount={false}
/>
```

_Tabs

```
<Tabs
  tabs={[
    { id: 'tab1', label: 'General', content: <div>...</div>
},
    { id: 'tab2', label: 'Settings', icon: <Settings size=
{16} />, badge: 3 },
    { id: 'tab3', label: 'Disabled', disabled: true },
  ]}
  defaultActiveTab="tab1"
  activeTab={controlled}        // optional controlled mode
  onChange={(tabId) => {}}
  variant="underline"           // "underline" | "pill"
  size="md"                     // "sm" | "md" | "lg"
  fullWidth={false}
/>
```

_Header

```
<Header
  logo={<img src="/logo.svg" alt="Logo" />}
  productName="Product"
  navItems={[
    { label: 'Dashboard', href: '/', active: true, icon:
<Home size={16} /> },
    { label: 'Settings', href: '/settings', onClick: fn },
  ]}
  actions={<Button size="sm">Sign out</Button>}
```

```
/>
```

_SideNav

```
<SideNav
  collapsed={false}          // collapsed = icon-only (w-16), expanded = full (w-60)
  header={<Logo />}
  footer={<UserMenu />}
  items={[
    { label: 'Dashboard', href: '/', icon: <Home size={18} />, active: true, badge: '3' },
    { label: 'Analytics', icon: <BarChart size={18} />, children: [
      { label: 'Overview', href: '/analytics' },
      { label: 'Reports', href: '/reports' },
    ], defaultExpanded: true },
    { divider: true, label: 'Settings' },   // section divider with optional label
    { label: 'Preferences', icon: <Settings size={18} />, href: '/settings' },
  ]}
/>
```

_Breadcrumb

```
<Breadcrumb
  items={[
    { label: 'Home', href: '/', icon: <Home size={14} /> },
    { label: 'Projects', href: '/projects' },
    { label: 'Current' },     // last item renders as plain text (aria-current="page")
  ]}
```

```
  separator={<CustomSeparator />}   // optional, defaults to
ChevronRight
/>
```

_Search

```
<Search
  value={controlled}              // optional controlled mode
  onChange={(val) => {}}
  onSearch={(val) => {}}          // fires on Enter or after
debounce
  onClear={fn}
  placeholder="Search..."
  suggestions={[                  // string[] or { label,
description }[]
    { label: 'Result 1', description: 'Description' },
  ]}
  onSuggestionSelect={(suggestion) => {}}
  loading={false}
  size="md"                       // "sm" | "md" | "lg"
  disabled={false}
  scope="Projects"                // optional scope badge
inside input
  debounceMs={300}
  wrapperClassName=""
/>
```

_DatePicker

```
<DatePicker
  label="Start date"
  value="2026-03-12"              // format: YYYY-MM-DD
  onChange={(dateStr) => {}}
```

```
  mode="single"                // "single" | "range"
  rangeEnd="2026-03-20"        // only when mode="range"
  onRangeChange={(start, end) => {}}
  placeholder="Select date"
  min="2026-01-01"             // disable dates before
  max="2026-12-31"             // disable dates after
  size="md"                    // "sm" | "md" | "lg"
  disabled={false}
  required={false}
  errorText=""
  helperText=""
  wrapperClassName=""
/>
```

_OverflowMenu

```
<OverflowMenu
  trigger={<CustomTrigger />}  // optional, defaults to
MoreVertical icon
  align="right"                // "right" | "left"
  size="md"                    // "sm" | "md" | "lg"
  items={[
    { label: 'Edit', icon: <Edit size={14} />, onClick: fn,
shortcut: '⌘E' },
    { label: 'Duplicate', icon: <Copy size={14} />, onClick:
fn },
    { divider: true },
    { label: 'Delete', icon: <Trash size={14} />, onClick:
fn, danger: true },
    { label: 'Disabled', onClick: fn, disabled: true },
  ]}
/>
```

_Form / _FormGroup / _FormRow / _FormActions

```
<Form onSubmit={(e) => {}}>
  <FormGroup legend="Personal Info">
    <FormRow>                    {/* 2-column grid on md+
screens */}
      <TextInput label="First name" />
      <TextInput label="Last name" />
    </FormRow>
    <TextInput label="Email" />
  </FormGroup>

  <FormActions align="right">  {/* "left" | "center" |
"right" | "between" */}
    <Button variant="tertiary">Cancel</Button>
    <Button type="submit">Save</Button>
  </FormActions>
</Form>
```

_Loading Components

```
<Spinner size="md" label="Loading..." />       // "sm" |
"md" | "lg" | "xl"
<Skeleton width="200px" height="1rem" variant="rectangular"
/>  // "rectangular" | "circular" | "text"
<SkeletonText lines={3} />
<TableSkeleton rows={5} columns={4} />
```

Design Tokens

All tokens are CSS custom properties defined in `src/tokens/tokens.css`. Components reference **semantic** tokens only — never primitives directly.

_Colour Palette (Primitives)

Scale	Token pattern	Range
Gray	`--ds-gray-{0,50,100..900,950}`	#ffffff → #030712
Blue (Primary)	`--ds-blue-{50..900}`	#eff6ff → #1e3a8a
Red (Danger)	`--ds-red-{50..900}`	#fef2f2 → #7f1d1d
Green (Success)	`--ds-green-{50..900}`	#f0fdf4 → #14532d
Amber (Warning)	`--ds-amber-{50..900}`	#fffbeb → #78350f
Teal (Info)	`--ds-teal-{50..900}`	#f0fdfa → #134e4a
Purple (Accent)	`--ds-purple-{50..900}`	#faf5ff → #581c87

_Semantic Tokens (use these in components)

Backgrounds: `--ds-bg-primary` `--ds-bg-secondary` `--ds-bg-tertiary` `--ds-bg-inverse` `--ds-bg-brand` `--ds-bg-brand-hover` `--ds-bg-danger` `--ds-bg-danger-hover` `--ds-bg-success` `--ds-bg-warning` `--ds-bg-info` `--ds-bg-error` `--ds-bg-overlay` `--ds-bg-hover` `--ds-bg-active` `--ds-bg-selected` `--ds-bg-disabled`

Text: `--ds-text-primary` `--ds-text-secondary` `--ds-text-tertiary` `--ds-text-inverse` `--ds-text-brand` `--ds-text-danger` `--ds-text-success` `--ds-text-warning` `--ds-text-info` `--ds-text-disabled` `--ds-text-placeholder` `--ds-text-on-brand` `--ds-text-link` `--ds-text-link-hover`

Borders: `--ds-border-primary` `--ds-border-secondary` `--ds-border-focus` `--ds-border-error` `--ds-border-success` `--ds-border-warning` `--ds-border-info` `--ds-border-disabled` `--ds-border-brand` `--ds-border-inverse`

Icons: `--ds-icon-primary` `--ds-icon-secondary` `--ds-icon-inverse` `--ds-icon-brand` `--ds-icon-danger` `--ds-icon-success` `--ds-icon-warning` `--ds-icon-info` `--ds-icon-disabled`

Component-specific: `--ds-input-bg` `--ds-input-border` `--ds-input-border-hover` `--ds-table-header-bg` `--ds-table-row-hover` `--ds-table-row-selected` `--ds-table-border` `--ds-sidebar-bg` `--ds-sidebar-text` `--ds-sidebar-text-active` `--ds-sidebar-hover` `--ds-sidebar-active` `--ds-header-bg` `--ds-header-border`

_Spacing (4px base)

Token	Value
`--ds-spacing-0`	0
`--ds-spacing-1`	0.25rem (4px)
`--ds-spacing-2`	0.5rem (8px)
`--ds-spacing-3`	0.75rem (12px)
`--ds-spacing-4`	1rem (16px)
`--ds-spacing-5`	1.25rem (20px)
`--ds-spacing-6`	1.5rem (24px)
`--ds-spacing-8`	2rem (32px)
`--ds-spacing-10`	2.5rem (40px)
`--ds-spacing-12`	3rem (48px)
`--ds-spacing-16`	4rem (64px)
`--ds-spacing-20`	5rem (80px)
`--ds-spacing-24`	6rem (96px)

_Sizing

Token	Value	Use
`--ds-size-xs`	1.5rem (24px)	Small badges, tags
`--ds-size-sm`	2rem (32px)	Small buttons/inputs
`--ds-size-md`	2.5rem (40px)	Default buttons/inputs
`--ds-size-lg`	3rem (48px)	Large buttons/inputs
`--ds-size-xl`	3.5rem (56px)	Extra large

Icons: `--ds-icon-{xs,sm,md,lg,xl}` → 12px, 16px, 20px, 24px, 32px

Containers: `--ds-container-{sm,md,lg,xl,2xl}` → 640px, 768px, 1024px, 1280px, 1536px

_Typography

Token	Value
`--ds-font-sans`	Geist Sans (falls back to system sans-serif)
`--ds-font-mono`	Geist Mono (falls back to system monospace)
`--ds-text-xs`	0.75rem (12px)
`--ds-text-sm`	0.875rem (14px)
`--ds-text-md`	1rem (16px)
`--ds-text-lg`	1.125rem (18px)
`--ds-text-xl`	1.25rem (20px)
`--ds-text-2xl`	1.5rem (24px)
`--ds-text-3xl`	1.875rem (30px)
`--ds-text-4xl`	2.25rem (36px)

Token	Value
`--ds-font-regular`	400
`--ds-font-medium`	500
`--ds-font-semibold`	600
`--ds-font-bold`	700
`--ds-leading-none`	1
`--ds-leading-tight`	1.25
`--ds-leading-snug`	1.375
`--ds-leading-normal`	1.5
`--ds-leading-relaxed`	1.625

_Border Radius

`--ds-radius-none` (0) · `--ds-radius-sm` (4px) · `--ds-radius-md` (6px) · `--ds-radius-lg` (8px) · `--ds-radius-xl` (12px) · `--ds-radius-2xl` (16px) · `--ds-radius-full` (9999px)

_Shadows

`--ds-shadow-xs` · `--ds-shadow-sm` · `--ds-shadow-md` · `--ds-shadow-lg` · `--ds-shadow-xl` · `--ds-shadow-2xl`

Dark theme automatically increases shadow opacity.

_Motion

Token	Value
`--ds-duration-fast`	100ms

Token	Value
`--ds-duration-normal`	200ms
`--ds-duration-slow`	300ms
`--ds-duration-slower`	500ms
`--ds-ease-default`	cubic-bezier(0.4, 0, 0.2, 1)

_Z-Index

Token	Value
`--ds-z-dropdown`	1000
`--ds-z-sticky`	1020
`--ds-z-fixed`	1030
`--ds-z-modal-backdrop`	1040
`--ds-z-modal`	1050
`--ds-z-popover`	1060
`--ds-z-tooltip`	1070
`--ds-z-toast`	1080

Theming

_Available themes: `light` (default), `dark`, `high-contrast`

Themes are applied via `data-theme` attribute on `<html>`. Switch at runtime using the ThemeProvider context:

```
import { useTheme } from '@/context/ThemeProvider';

function ThemeSwitcher() {
  const { theme, setTheme, toggleTheme, themes } =
useTheme();
  return (
    <select value={theme} onChange={(e) =>
setTheme(e.target.value)}>
      {themes.map(t => <option key={t} value={t}>{t}
</option>)}
    </select>
  );
}
```

The `Providers` component in `src/app/providers.js` wraps the app with `ThemeProvider`. Theme persists in `localStorage` under key `ds-theme`.

_Creating a custom theme

Add a new `[data-theme="your-theme"]` block in `tokens.css` overriding the semantic tokens, then add the theme name to the `THEMES` array in `src/context/ThemeProvider.js`.

Style Guide

_General rules

- **Use semantic tokens** (`--ds-bg-brand`, not `--ds-blue-600`). This ensures theme compatibility.
- **Tailwind CSS 4** is the styling approach. Token values are applied inline via `var()` inside Tailwind arbitrary values: `bg-[var(--ds-bg-primary)]`, `text-[color:var(--ds-text-brand)]`.
  - For font-size tokens, use the `length` hint: `text-[length:var(--ds-text-sm)]`

  - For color tokens in `text-`, use the `color` hint: `text-[color:var(--ds-text-brand)]`
  - For colors without hint ambiguity (bg, border), no hint is needed: `bg-[var(--ds-bg-primary)]`
- **All interactive elements** must include the `ds-focus-ring` class for keyboard accessibility.
- **Transitions** use design tokens: `transition-all duration-[var(--ds-duration-normal)]`.

_Layout patterns

- **App shell:** `Header` (h-14, top) + `SideNav` (w-60 or w-16 collapsed) + main content area.
- **Forms:** Wrap in `<Form>`, group sections with `<FormGroup legend="...">`, pair fields side-by-side with `<FormRow>`, and end with `<FormActions>`.
- **Spacing:** Use Tailwind's spacing utilities. For component internals, the token scale maps to Tailwind: `p-4` = 16px = `--ds-spacing-4`.

_Component patterns

- All form components (`TextInput`, `Select`, `Dropdown`, `DatePicker`, `Checkbox`, `Toggle`, `Search`) support `size="sm|md|lg"` for consistent sizing.
- Error states: pass `errorText` to form controls. It sets red borders, shows error icons, and displays the message below the field.
- Loading states: `Button` has a `loading` prop. `DataTable` has a `loading` prop. Use `Spinner`, `Skeleton`, `SkeletonText`, or `TableSkeleton` for content loading.
- All components forward refs and spread `...props` for extensibility.

_Typography

- Page headings: `text-[length:var(--ds-text-3xl)]` or `text-[length:var(--ds-text-4xl)]` with `font-bold`
- Section headings: `text-[length:var(--ds-text-xl)]` with `font-semibold`
- Body text: `text-[length:var(--ds-text-md)]` (16px default)
- Small / helper text: `text-[length:var(--ds-text-sm)]` or `text-[length:var(--ds-text-xs)]`

- Use `--ds-text-primary` for main content, `--ds-text-secondary` for supporting text, `--ds-text-tertiary` for subdued text.

_Accessibility

- Focus rings via `ds-focus-ring` class (2px solid blue outline with 2px offset).
- Screen reader text via `ds-sr-only` class.
- All interactive components include proper ARIA attributes (`role`, `aria-label`, `aria-expanded`, `aria-modal`, etc.).
- Modal traps focus context and responds to Escape key.
- Color contrast meets WCAG AA in all three themes (high-contrast theme is designed for enhanced contrast).

Quick-Start Example

```
'use client';
import { useState } from 'react';
import {
  Header, SideNav, Button, TextInput, Modal, DataTable,
  Tag, Notification, Form, FormGroup, FormRow, FormActions
} from 'my_ds_name';
import { Home, Settings, Users } from 'lucide-react';

export default function DashboardPage() {
  const [sideNavCollapsed, setSideNavCollapsed] =
useState(false);
  const [modalOpen, setModalOpen] = useState(false);

  return (
    <div className="h-screen flex flex-col">
```

```
      <Header
        productName="Acme Admin"
        navItems={[{ label: 'Dashboard', href: '/', active:
true }]}
        actions={<Button size="sm" variant="ghost">Sign
```

End of specification for AI

‹ Design Systems for AI

Build design systems optimized for AI workfl...

AI Design System ›

A design system optimized for AI-assisted w...