Okay, let's compile the detailed specifications for the Vercel (v0) frontend components. This focuses on the user interface elements needed for an initial, functional prototype, prioritizing the core user flow: discovery -> basic prioritization output -> basic reporting view. We'll assume a modern framework like React/Next.js, commonly used with Vercel, and potentially a UI component library like Shadcn/ui or Material UI for speed.

---

## Frontend Component Specifications (v0 Prototype for Vercel)

\*\*Objective:\*\* Define the core frontend components needed for a Minimum Viable Product (MVP) / v0 prototype of the AI Transformation Prioritization Tool, suitable for deployment on Vercel. This version prioritizes the wizard-based data capture and the display of initial prioritization results. Post-deployment tracking features will be represented by placeholders initially.

\*\*Assumed Stack:\*\* Next.js (React), Tailwind CSS or a component library (e.g., Shadcn/ui, MUI).

\*\*Core User Flows for v0:\*\*

1. User starts a new assessment via the Wizard.

2. User progresses through Wizard steps, inputting data.

3. User submits Wizard data.

4. User views a generated report showing prioritization (Heatmap/List) and basic AI suggestions.

5. (Optional v0) Basic viewing of static Job Role / AI Capability libraries.

---

### 1. Global Components

These components provide the overall structure and consistent elements across the application.

\* \*\*`<AppLayout>`\*\*

\* \*\*Purpose:\*\* Main container for all authenticated views. Sets up the primary layout structure.

\* \*\*Structure:\*\* Typically includes `<Header>`, `<SidebarNav>`, and a main content area where specific page components are rendered.

\* \*\*Props:\*\* `children` (the page content).

\* \*\*Notes:\*\* Handles overall padding, background colors.

\* \*\*`<Header>`\*\*

\* \*\*Purpose:\*\* Displays top-level navigation and branding.

\* \*\*Elements:\*\* Application Logo, possibly User Profile dropdown (static display for v0, e.g., "Consultant User"), Logout button (non-functional or basic implementation for v0).

\* \*\*Styling:\*\* Clean, fixed or sticky at the top.

\* \*\*`<SidebarNav>`\*\*

\* \*\*Purpose:\*\* Primary navigation between major sections of the application.

\* \*\*Elements:\*\* Links (using Next.js `<Link>`) to:

\* Dashboard / Reports (`/`)

\* Start New Assessment / Wizard (`/wizard`)

\* Libraries (`/library` - potentially combining Role & AI matrix views for v0)

\* \*\*Styling:\*\* Collapsible or fixed sidebar, clear indication of the active section.

\* \*\*`<LoadingSpinner>`\*\*

\* \*\*Purpose:\*\* Provide visual feedback during asynchronous operations (e.g., submitting wizard, loading report).

\* \*\*Elements:\*\* Animated spinner or loading indicator.

\* \*\*Props:\*\* `isLoading` (boolean).

\* \*\*`<Notification>` / `<Toast>`\*\*

\* \*\*Purpose:\*\* Display brief success, error, or warning messages.

\* \*\*Elements:\*\* Message text, icon, close button.

\* \*\*Props:\*\* `message`, `type` ('success', 'error', 'warning'), `isVisible`.

\* \*\*Notes:\*\* Use a library component for this (e.g., `react-hot-toast`).

---

### 2. Discovery Wizard Components (`/wizard`)

Components specific to the multi-step assessment process.

\* \*\*`<WizardLayout>`\*\*

\* \*\*Purpose:\*\* Manages the flow and structure of the wizard steps.

\* \*\*Elements:\*\*

\* `<ProgressIndicator>` (see below).

\* Container for the current `<WizardStep>` content.

\* Navigation Buttons (`<Button>`) - "Next", "Previous", "Submit" (conditionally shown).

\* \*\*State:\*\* Manages the current step index. Handles navigation logic.

\* \*\*Props:\*\* `steps` (array of step configurations/content), `onSubmit` (function called on final step).

\* \*\*`<ProgressIndicator>`\*\*

\* \*\*Purpose:\*\* Shows the user their progress through the wizard.

\* \*\*Elements:\*\* Visual representation of steps (e.g., numbered circles, progress bar). Highlights the current step.

\* \*\*Props:\*\* `currentStep`, `totalSteps`.

\* \*\*`<WizardStep>`\*\*

\* \*\*Purpose:\*\* Container for the content of a single step in the wizard.

\* \*\*Elements:\*\* Step Title (`<h2>`), Step Description/Instructions (`<p>`), container for `QuestionCard`s or other step-specific content.

\* \*\*Props:\*\* `title`, `description`, `children`.

\* \*\*`<QuestionCard>`\*\*

\* \*\*Purpose:\*\* Displays a single question and its input mechanism.

\* \*\*Elements:\*\* Question Text (`<label>`), Guidance/Help Text (`<p>` - optional), Input Element (varies based on type), Validation Message display area.

\* \*\*Props:\*\* `questionId`, `questionText`, `guidanceText`, `inputType`, `options` (for select/radio/rank), `value`, `onChange`, `isRequired`.

\* \*\*Variants (Input Types):\*\*

\* \*\*`<TextInput>` / `<Textarea>`:\*\* For open-ended text (goals, pain points).

\* \*\*`<RatingScale>`:\*\* Likert scale (e.g., 1-5 radio buttons or clickable stars). Needs labels for scale ends (e.g., "Low Impact" / "High Impact").

\* \*\*`<RankingList>`:\*\* Draggable list items for rank-order questions. Use a library like `react-beautiful-dnd`. Needs clear numbering/visual cues for rank. Limit items (e.g., 5-7).

\* \*\*`<RoleSelector>`:\*\* Multi-select dropdown, checklist, or searchable list to pick relevant roles from the library (use mock data for v0).

\* \*\*`<SingleChoice>` / `<MultipleChoice>`:\*\* Radio buttons or checkboxes for selecting predefined options.

---

### 3. Reporting Engine Components (`/reports` or `/reports/[reportId]`)

Components for displaying the analysis results. v0 focuses on static display of the \*initial\* recommendations.

\* \*\*`<ReportView>`\*\*

\* \*\*Purpose:\*\* Main container for displaying a single generated report.

\* \*\*Elements:\*\* Report Title, Date Generated, Sections for different report parts (Summary, Heatmap, Details, etc.), "Export to PDF" button (placeholder/non-functional for v0).

\* \*\*Props:\*\* `reportData` (object containing all necessary information).

\* \*\*Notes:\*\* Fetches or receives the processed report data.

\* \*\*`<SummaryCard>`\*\*

\* \*\*Purpose:\*\* Display key text sections of the report.

\* \*\*Elements:\*\* Card container with a title (e.g., "Executive Summary", "Scope") and formatted text content.

\* \*\*Props:\*\* `title`, `content` (string, potentially supports basic markdown).

\* \*\*`<HeatmapDisplay>`\*\*

\* \*\*Purpose:\*\* Visualize the prioritization matrix (e.g., Value vs. Effort). \*\*CRITICAL VISUAL.\*\*

\* \*\*Elements (v0 Simple):\*\* An HTML `<table>` or a grid layout (`div`s).

\* Axis Labels (e.g., "Business Value", "Implementation Effort").

\* Cells representing intersections (e.g., High Value / Low Effort). Cell background color determined by priority score (e.g., Red = High, Yellow = Medium, Green = Low). Cell contains the name(s) of the role/task/function prioritized there.

\* \*\*Props:\*\* `data` (array of prioritized items with scores/coordinates), `xAxisLabel`, `yAxisLabel`.

\* \*\*Notes:\*\* Keep v0 simple. No complex interactivity. Focus on clear visual representation of the priority levels.

\* \*\*`<PrioritizationList>`\*\*

\* \*\*Purpose:\*\* Display the prioritized items as an ordered list.

\* \*\*Elements:\*\* Ordered list (`<ol>`) where each list item (`<li>`) shows the rank, name of the prioritized role/task/function, and maybe its core score.

\* \*\*Props:\*\* `items` (sorted array of prioritized items).

\* \*\*`<AISuggestionCard>`\*\*

\* \*\*Purpose:\*\* Show potential AI capabilities mapped to a prioritized item.

\* \*\*Elements:\*\* Card associated with a high-priority item, listing relevant AI functions (e.g., "Potential AI Solution: Content Generation, Summarization").

\* \*\*Props:\*\* `itemName`, `suggestedCapabilities` (array of strings).

\* \*\*`<KPIPlaceholderCard>`\*\*

\* \*\*Purpose:\*\* Represent where post-deployment KPIs and ROI analysis will be displayed in future versions.

\* \*\*Elements:\*\* Card with a title (e.g., "Performance Tracking (Post-Deployment)", "ROI Analysis") and placeholder text/graphics indicating this section is for future data.

\* \*\*Props:\*\* `title`.

\* \*\*`<CommentaryBox>`\*\*

\* \*\*Purpose:\*\* Display the consultant's added narrative.

\* \*\*Elements:\*\* Formatted text area (read-only display for v0).

\* \*\*Props:\*\* `commentaryText`.

---

### 4. Library Management Components (`/library`)

Simplified view for v0, potentially just displaying static lists.

\* \*\*`<LibraryLayout>`\*\*

\* \*\*Purpose:\*\* Container for library views, potentially using tabs.

\* \*\*Elements:\*\* Tabs (e.g., "Job Roles", "AI Capabilities"). Content area displaying the selected library.

\* \*\*Props:\*\* `children`.

\* \*\*`<DataTable>`\*\*

\* \*\*Purpose:\*\* Display lists of roles or AI capabilities in a structured table.

\* \*\*Elements:\*\* Standard HTML `<table>` or a data table component from a library. Columns for key information (e.g., Role Name, Function, Key Responsibilities Snippet; AI Capability Name, Category, Description).

\* \*\*Props:\*\* `columns` (array of column definitions), `data` (array of row data - use mock data for v0).

\* \*\*Notes:\*\* Basic sorting might be included via library component. No editing/adding in v0 unless essential for testing flow.

---

\*\*v0 Specific Considerations & Simplifications:\*\*

\* \*\*Data:\*\* Use hardcoded mock data or simple JSON files initially to simulate API responses for roles, capabilities, wizard submission processing, and report generation.

\* \*\*Authentication:\*\* Can be skipped or mocked (assume a logged-in user) for the v0 prototype.

\* \*\*Backend Integration:\*\* Focus on building the UI components. API integration can come later. Button clicks might just log to the console or update local state in v0.

\* \*\*Responsiveness:\*\* Ensure basic responsiveness, but pixel-perfect polish across all devices is not the primary goal for v0.

\* \*\*Error Handling:\*\* Implement basic error states/messages, but complex edge-case handling can be deferred.

\* \*\*Dynamic Population:\*\* The UI won't trigger spidering/scraping in v0. It will consume static data representing what \*would\* be populated dynamically later.

\* \*\*Reporting Complexity:\*\* The v0 report is static. No interactive charts, filtering, or real-time data updates. KPI/ROI sections are placeholders.

This specification provides a blueprint for building the essential frontend components for the v0 prototype on Vercel, focusing on getting the core user journey functional.