Front-End Product Requirements Document (PRD): MCP Registry GUI

1. Introduction

This document specifies the product requirements for the Graphical User Interface (GUI) of the new MCP Registry and Management Platform. The GUI is the primary interface for users to manage their Model Context Protocol (MCP) agents and interact with them via a powerful, multimodal chat experience.

2. Goals and Philosophy

2.1. Product Goals

- **Simplify MCP Management:** Provide a clear, intuitive interface for registering, configuring, and monitoring MCP agents.
- **Enable Multimodal Interaction:** Seamlessly integrate voice, vision, and document input/output into the core chat experience.
- **Enhance Maintainability:** Utilize a modern, component-based framework (Next.js/React) and a utility-first styling approach (Tailwind CSS) to ensure a highly maintainable codebase.

2.2. UX/UI Philosophy: "Mission Control"

The design philosophy is "Mission Control"—a professional, data-rich, yet clean interface that provides the user with complete control and clear status visibility.

- **Clarity:** Information hierarchy must be simple and direct.
- **Efficiency:** Core tasks (e.g., sending a query, checking agent status) must be achievable with minimal clicks.
- **Professionalism:** The aesthetic should be modern, high-contrast, and professional, with a strong emphasis on the requested **Dark Mode**.

3. Target Users

User Type	Primary Goal	Key Needs
MCP Administrator	Register, configure, and monitor the health of MCP	Clear status dashboards, secure credential input,

	agents.	detailed error logs.
End User / Agent Operator	Interact with the most appropriate MCP agent to complete a task (text, voice, or vision).	Fast, responsive chat interface, easy access to multimodal input options, clear context display.

4. Core User Flows

4.1. Flow 1: Registering a New MCP Agent

1. User navigates to the "Registry Management" section.
2. User clicks "Add New MCP Agent."
3. A form modal appears, requesting: Agent Name, Endpoint URL, MCP Manifest (JSON/YAML upload or paste), and API Key/Credentials.
4. User submits the form.
5. The UI displays a "Testing Connection..." status.
6. Upon success, the new agent appears in the list with an "Online" status.

4.2. Flow 2: Multimodal Interaction (Voice Command)

1. User is on the main "Chat Interface" screen.
2. User clicks the "Microphone" icon.
3. The UI provides visual feedback (e.g., a pulsing waveform) indicating active listening.
4. User speaks a query (e.g., "Summarize the document I just uploaded").
5. User stops speaking; the UI displays the transcribed text (STT result).
6. The UI displays a "Routing..." status while the backend selects the agent.
7. The response is received and displayed as text, and the UI provides an option to play the audio response (TTS).

4.3. Flow 3: Dark Mode Toggle

1. User clicks the "Settings" or "Theme" icon (e.g., a sun/moon icon) in the navigation bar.
2. The UI instantly switches between the Light and Dark themes, persisting the user's preference across sessions.

5. Next Steps

The following phases will detail the specific functional requirements for the components identified in these flows.

5. Functional Requirements: MCP Registry Management GUI

This section details the requirements for the interface used by the **MCP Administrator** to manage registered agents.

5.1. Registry Dashboard (Agent List View)

ID	Requirement	Description
FR-REG-5.1.1	**Agent List Display**	The dashboard SHALL display a sortable, searchable list of all registered MCP agents.
FR-REG-5.1.2	**Real-time Status Indicator**	Each agent entry SHALL display a real-time status indicator (e.g., green for Online, red for Offline/Error, yellow for Testing/Warning). This status must be updated via WebSocket from the Monitoring Service.
FR-REG-5.1.3	**Key Metadata Display**	Each entry SHALL display key metadata: Agent Name, Endpoint URL, Last Active Time, and a summary of its capabilities (e.g., "Vision," "Data Analysis") derived from the MCP Manifest.
FR-REG-5.1.4	**Action Buttons**	Each entry SHALL include action buttons for "View Details," "Edit Configuration," and "De-register."

5.2. Agent Configuration and CRUD

ID	Requirement	Description
FR-REG-5.2.1	**Registration Form**	The "Add New MCP Agent" form SHALL include fields for: Agent Name, Endpoint URL, Manifest Upload/Paste, and a secure input for API Key/Credentials.
FR-REG-5.2.2	**Secure Credential Input**	The API Key/Credentials field SHALL mask input and SHALL NOT store the value in the browser's local storage.
FR-REG-5.2.3	**Manifest Validation**	The UI SHALL perform client-side validation of the MCP Manifest structure before submission to the backend.
FR-REG-5.2.4	**Connection Test**	The UI SHALL provide a "Test Connection" button on the registration/edit form to trigger an immediate health check against the provided endpoint.
FR-REG-5.2.5	**Confirmation Dialog**	The "De-register" action SHALL require a confirmation dialog to prevent accidental deletion.

5.3. Agent Details View

ID	Requirement	Description
FR-REG-5.3.1	**Full Manifest Display**	The details view SHALL display the full, formatted MCP Manifest (JSON/YAML) in a read-only code block.
FR-REG-5.3.2	**Activity Log**	The details view SHALL include a filtered, paginated log of recent activity and error messages related to the

		specific agent, pulled from the Monitoring Service.
FR-REG-5.3.3	**Metrics Visualization**	The view SHALL display simple charts for key performance metrics, such as "Response Latency (Avg/P95)" and "Uptime Percentage" over the last 24 hours.

6. Functional Requirements: Multimodal Chat Interface

This section details the requirements for the primary user interaction screen, designed for seamless multimodal communication.

6.1. Core Chat Interaction

ID	Requirement	Description
FR-CHAT-6.1.1	**Text Input**	Standard text input field with support for multi-line entry and the ability to submit via the Enter key.
FR-CHAT-6.1.2	**Conversation History**	The chat window SHALL display the full conversation history, clearly distinguishing between user input and MCP responses.
FR-CHAT-6.1.3	**Agent Selection**	The UI SHALL provide a prominent dropdown or selector allowing the user to explicitly choose a registered MCP agent for the next query (defaulting to the Routing Service).
FR-CHAT-6.1.4	**Routing Status**	When the Routing Service is selected, the UI SHALL display a brief status message (e.g., "Routing to Vision Agent...") during the agent selection process.

6.2. Voice Input and Output

ID	Requirement	Description
FR-CHAT-6.2.1	**Microphone Button**	A dedicated, easily accessible microphone button SHALL initiate the voice recording session.
FR-CHAT-6.2.2	**Visual Feedback (Input)**	During recording, the UI SHALL display a real-time visual indicator (e.g., a waveform or volume meter) to confirm the microphone is active and receiving input.
FR-CHAT-6.2.3	**STT Display**	Upon completion of the voice input, the transcribed text (STT result) SHALL be displayed in the chat window as the user's message.
FR-CHAT-6.2.4	**TTS Playback**	Every MCP text response SHALL include a small "Play" button to trigger the TTS audio playback.
FR-CHAT-6.2.5	**Audio Playback Control**	The UI SHALL provide basic controls (play/pause, progress bar) for the TTS audio playback.

6.3. Vision and Document Input

ID	Requirement	Description
FR-CHAT-6.3.1	**File Upload Button**	A dedicated button (e.g., a paperclip icon) SHALL allow the user to upload files (images, documents).
FR-CHAT-6.3.2	**Supported Formats**	The UI SHALL clearly indicate support for image formats

		(JPEG, PNG) and document formats (PDF, DOCX, TXT).
FR-CHAT-6.3.3	**Upload Progress**	For large documents, the UI SHALL display a progress bar for the file upload and a status indicator for the backend processing (e.g., "Processing Document for Context...").
FR-CHAT-6.3.4	**Glazyr Context Request**	The UI SHALL include a dedicated button or command to request the current visual context from Glazyr (e.g., "Send Current Screen"). The resulting image/context summary SHALL be displayed in the chat history.
FR-CHAT-6.3.5	**Context Display**	When a query is sent with an attached file or Glazyr context, the UI SHALL visually link the query to the context (e.g., a thumbnail of the image or document).

7. Non-Functional Requirements (NFR)

These requirements ensure the quality, performance, and accessibility of the user interface.

7.1. User Experience and Aesthetics

ID	Requirement	Description
NFR-UX-7.1.1	**Dark Mode**	The UI SHALL support a user-selectable dark mode theme, with a high-contrast color palette that is easy on the eyes for extended use.
NFR-UX-7.1.2	**Responsiveness**	The UI SHALL be fully responsive and functional across all major screen sizes,

		including desktop, tablet, and mobile devices.
NFR-UX-7.1.3	**State Persistence**	The application SHALL persist user preferences (e.g., dark mode setting, last active chat session) across browser sessions.
NFR-UX-7.1.4	**Loading Feedback**	All asynchronous operations (e.g., sending a message, fetching status, uploading a file) SHALL provide clear, non-intrusive visual feedback (e.g., spinners, progress bars) to the user.

7.2. Performance

ID	Requirement	Description
NFR-PER-7.2.1	**Initial Load Time**	The application SHALL achieve a Largest Contentful Paint (LCP) of under 2.5 seconds on a standard broadband connection.
NFR-PER-7.2.2	**Chat Latency**	The UI SHALL display the start of the MCP response within 500ms of receiving the first data chunk from the API Gateway (streaming response).
NFR-PER-7.2.3	**Bundle Size**	The JavaScript bundle size SHALL be optimized to minimize download time, leveraging code splitting and tree-shaking.

7.3. Accessibility

ID	Requirement	Description

NFR-ACC-7.3.1	**WCAG Compliance**	The UI SHALL aim for WCAG 2.1 AA compliance, focusing on keyboard navigation, sufficient color contrast, and screen reader compatibility.
NFR-ACC-7.3.2	**Keyboard Navigation**	All interactive elements (buttons, links, form fields) SHALL be fully navigable and operable using only the keyboard.
NFR-ACC-7.3.3	**Semantic HTML**	The application SHALL use appropriate semantic HTML elements to ensure screen readers can correctly interpret the content structure.