Full Stack AI agent

by - Ashu Mishra

Create your own backend + front end via lovable and n8n

Nebula AI: Frequently Asked Questions

General Questions

What is Nebula AI? Nebula AI is a specialized, enterprise-grade large language model (LLM) designed from the ground up for the manufacturing and robotics sectors. Unlike general-purpose AI, Nebula AI is pre-trained on a vast corpus of industrial data, including engineering specifications, equipment manuals, automation workflows, and physics-based process data. It understands the unique language and complex challenges of modern manufacturing, allowing it to provide precise, actionable insights rather than just general information.

How is Nebula AI different from a general-purpose LLM like GPT-4 or Claude? The difference is specialization. A general-purpose LLM is a "jack-of-all-trades" that can write a poem or summarize a history article. Nebula AI is a "master" in the domain of manufacturing.

- **Contextual Understanding:** Nebula understands that "torque," "tolerance," and "cycle time" have precise, high-stakes meanings. It can read a sensor log or a P&ID diagram and understand its operational context.
- **Data Integration:** It doesn't just process text; it integrates directly with your plant's live data sources (like SCADA, MES, and IIoT sensors) to analyze real-time events.
- **Actionable Output:** Instead of just generating text, Nebula AI can initiate actions, such as triggering an RPA workflow, updating a maintenance ticket, or even sending control

Step 1

Login to lovable.dev

Step 2

Use the following prompt to generate the UI :

Prompt for lovable.dev

Create a tech support interface for Nebula AI, an enterprise-grade LLM platform specialized in manufacturing and robotics automation.

OVERALL LAYOUT:

- Contained section design (not full-page)
- Maximum width: 1200px, centered on page
- Split into two panels with glass-morphism or card effect
- Tech/futuristic aesthetic with subtle animations

LEFT PANEL (Information Section) - 45% width:

Header:

- Logo: [WILL BE PROVIDED - placeholder for now]
- Heading: "Nebula AI Technical Support"
- Tagline: "Intelligent Automation for Smart Manufacturing"

Content Sections:

1. "What is Nebula AI?" Brief paragraph: "Nebula AI is an enterprise-grade large language model purpose-built for manufacturing environments and robotics operations. Our AI understands industrial processes, equipment specifications, and automation workflows to deliver precise, actionable insights."
2. "Core Capabilities" (with futuristic icons):
   - Predictive Maintenance Intelligence
   - Real-time Process Optimization
   - Robotic Process Automation (RPA)
   - Quality Control Analytics
   - Supply Chain Coordination
   - Natural Language Equipment Control
3. "Industries We Serve":
   - Automotive Manufacturing
   - Electronics Assembly
   - Aerospace & Defense
   - Pharmaceutical Production
   - Food & Beverage Processing
   - Warehouse Automation
4. Support Stats (displayed as glowing metrics):
   - "99.9% Uptime"
   - "<3 min Response Time"
   - "24/7 AI Support"

RIGHT PANEL (Support Form) - 55% width:

Form Header:

- Title: "Get Support"
- Subtitle: "Our AI-powered support system is ready to assist"

Form Fields (minimal, clean):

1. Email Address
   - Label: "Your Email"
   - Type: email input
   - Placeholder: "engineer@yourcompany.com"
   - Required: Yes
   - Full width
2. Support Query
   - Label: "How can we help you?"
   - Type: textarea
   - Rows: 6-7
   - Placeholder: "Describe your technical issue, integration question, or support request..."
   - Required: Yes
   - Full width

Submit Button:

- Text: "Submit to Support AI"
- Style: Prominent, futuristic button with glow effect
- States:
  - Default: Animated gradient or glow
  - Hover: Intensified glow
  - Loading: Pulsing animation with text "Sending to AI Support..."
  - Success: Checkmark icon with "Message Sent Successfully!"

WEBHOOK RESPONSE NOTIFICATION: After successful webhook submission, display a notification/popup with the AI-generated response:

- Style: Futuristic toast notification or modal popup
- Position: Top-right corner (toast) OR center overlay (modal)
- Design: Glass-morphism card with glow border
- Icon: AI robot icon or checkmark with cyan/blue glow
- Content: Display the exact response text received from webhook
- Auto-dismiss: After 8-10 seconds with smooth fade-out
- Close button: Manual close option (X button in top-right)
- Animation: Slide-in from right (toast) OR fade-in with scale (modal)
- Background overlay (if modal): Semi-transparent dark backdrop with blur

The notification should feel like an AI assistant is responding - make it visually distinct from generic success messages.

DESIGN SYSTEM - Tech/Futuristic Theme:

Color Palette (will be updated with provided brand colors):

- Background: Dark theme (#0A0E27 or similar deep navy/black)
- Panels: Semi-transparent cards with backdrop blur (glass-morphism)
- Primary Accent: Electric blue (#00F0FF or cyan-blue)
- Secondary Accent: Purple (#8B5CF6) or magenta
- Text Primary: White (#FFFFFF)
- Text Secondary: Light gray (#CBD5E1)
- Borders: Subtle glowing lines (#3B82F6 with opacity)
- Success/Notification: Cyan-blue (#00F0FF) with glow
- Error: Red (#EF4444) with glow

Visual Effects:

- Subtle grid pattern or circuit board texture in background
- Gradient overlays on panels
- Soft glow effects on interactive elements
- Smooth transitions (200-300ms)
- Hover effects with scale or glow intensification
- Floating animation on info cards (optional, subtle)

Typography:

- Headings: "Orbitron", "Exo 2", or "Rajdhani" (futuristic tech fonts)
- Body: "Inter", "Space Grotesk", or "Plus Jakarta Sans" (clean, readable)
- Sizes: Scale from 14px (body) to 32px (main heading)
- Font weights: Regular (400), Medium (500), Bold (700)

Layout Spacing:

- Container padding: 3rem (48px)
- Section gaps: 2rem (32px)
- Form field spacing: 1.5rem (24px)
- Border radius: 12-16px for cards
- Input border radius: 8px

Icons:

- Use: Lucide Icons or Heroicons
- Style: Outline style with subtle glow
- Size: 24px for feature icons
- Color: Match accent colors

Animations:

- Page load: Fade-in and slide-up (stagger effect)
- Button hover: Glow intensification + slight scale (1.02)
- Input focus: Border glow animation
- Form submission: Button morphs to loading spinner, then checkmark

- Background: Subtle animated gradient or particle effect (optional)
- Notification: Smooth slide-in or fade-in entrance

RESPONSIVE BEHAVIOR:

- Desktop (>1024px): Side-by-side panels (45% / 55%)
- Tablet (768-1024px): Side-by-side with adjusted spacing
- Mobile (<768px):
  - Stacked layout (info on top, form below)
  - Full width panels
  - Reduced padding
  - Slightly smaller typography
  - Notification adjusts to smaller screen (full-width on mobile)

TECHNICAL IMPLEMENTATION:

Framework: React with Tailwind CSS

WEBHOOK CONFIGURATION - IMPORTANT: Add a configuration section at the top of your component for easy webhook URL management:

// WEBHOOK CONFIGURATION

// Replace this URL with your actual n8n webhook endpoint

const WEBHOOK_CONFIG = {

  url: 'YOUR_N8N_WEBHOOK_URL_HERE', // <-- PLACEHOLDER: Insert your n8n webhook URL

  method: 'POST',

  headers: {

   'Content-Type': 'application/json',

  }

};

Form Handling with Webhook Response Display:

const [isLoading, setIsLoading] = useState(false);

const [showNotification, setShowNotification] = useState(false);

const [notificationMessage, setNotificationMessage] = useState('');

const [notificationError, setNotificationError] = useState(false);

```javascript
const handleSubmit = async (e) => {

  e.preventDefault();


  const formData = {

    email: e.target.email.value,

    query: e.target.query.value,

    timestamp: new Date().toISOString(),

    source: 'Nebula AI Support Form'

  };


  // Show loading state

  setIsLoading(true);


  try {

    const response = await fetch(WEBHOOK_CONFIG.url, {

      method: WEBHOOK_CONFIG.method,

      headers: WEBHOOK_CONFIG.headers,

      body: JSON.stringify(formData)

    });


    if (response.ok) {

      // Parse the webhook response

      const webhookResponse = await response.json();


      // Extract the AI-generated message from webhook response

      // Adjust the key based on your n8n webhook response structure
```

```
      const aiMessage = webhookResponse.message || webhookResponse.response ||
webhookResponse.data || 'Your query has been received! Our AI agent is processing your request
and will respond shortly via email.';


    setIsLoading(false);

    setNotificationMessage(aiMessage);

    setNotificationError(false);

    setShowNotification(true);


    // Clear form

    e.target.reset();


    // Auto-dismiss notification after 8 seconds

    setTimeout(() => {

      setShowNotification(false);

    }, 8000);


  } else {

    throw new Error('Webhook request failed');

  }


} catch (error) {

  console.error('Submission error:', error);

  setIsLoading(false);

  setNotificationError(true);

  setNotificationMessage('Unable to submit your query. Please try again or contact support
directly.');
```

```
      setShowNotification(true);


    // Auto-dismiss error notification after 6 seconds

    setTimeout(() => {

      setShowNotification(false);

    }, 6000);

  }

};
```

Notification Component Structure: Create a reusable notification component that displays the webhook response:

```
{showNotification && (

  <div className="fixed top-4 right-4 z-50 animate-slide-in-right">

    <div className={`

      backdrop-blur-md rounded-lg p-4 shadow-2xl

      border ${notificationError ? 'border-red-500/50' : 'border-cyan-500/50'}

      bg-slate-900/90

      ${notificationError ? 'shadow-red-500/20' : 'shadow-cyan-500/20'}

      max-w-md

    `}>

      <div className="flex items-start gap-3">

        <div className={`

          flex-shrink-0 w-6 h-6

          ${notificationError ? 'text-red-400' : 'text-cyan-400'}

        `}>

        {notificationError ? (

          <AlertCircle className="w-6 h-6" />
```

```
        ) : (

          <CheckCircle className="w-6 h-6" />

        )}

      </div>

      <div className="flex-1">

        <p className="text-white text-sm leading-relaxed">

          {notificationMessage}

        </p>

      </div>

      <button

        onClick={() => setShowNotification(false)}

        className="flex-shrink-0 text-gray-400 hover:text-white transition-colors"

      >

        <X className="w-5 h-5" />

      </button>

    </div>

  </div>

</div>

)}
```

Add this CSS for the slide-in animation:

```
@keyframes slide-in-right {

  from {

    transform: translateX(100%);

    opacity: 0;

  }

  to {
```

```
    transform: translateX(0);

    opacity: 1;

  }

}



.animate-slide-in-right {

  animation: slide-in-right 0.3s ease-out;

}
```

Component Structure:

- Main container with background effects
- Two-column grid layout
- Left: Info panel component
- Right: Form component
- Webhook response notification/popup (conditionally rendered)
- State management for loading, notification visibility, and messages

Accessibility:

- Proper ARIA labels on all form inputs
- Keyboard navigation support
- Focus indicators with futuristic glow
- Screen reader announcements for form states and notifications
- Color contrast ratios meeting WCAG AA (adjust for dark theme)
- Notification includes ARIA live region for screen readers

Additional Features:

- Optional: Add "Powered by Nebula AI" with version number in footer
- Optional: Small pulsing indicator showing "AI Support Online"
- Optional: Animated background with subtle tech patterns or particles
- Optional: Add a small "Security: Enterprise-grade encryption" badge

BRAND ASSETS (TO BE INTEGRATED):

- Logo: [Will be provided - reserve space in top-left of info panel]
- Brand Colors: [Will be provided - update color palette accordingly]
- Ensure logo has proper padding and sizing (max height ~48px)

IMPORTANT NOTES FOR DEVELOPER:

1. The WEBHOOK_CONFIG object at the top makes it easy to update the webhook URL without searching through code
2. The webhook is expected to return a JSON response with the AI-generated message
3. Common response formats to handle:
   - { "message": "..." }
   - { "response": "..." }
   - { "data": "..." }
4. The notification will display whatever text is returned from the webhook
5. Error handling is included for failed webhook requests
6. Form clears automatically on successful submission
7. Notification auto-dismisses but can also be manually closed

Make it feel like a cutting-edge AI interface - think of the aesthetic from movies like Iron Man's Jarvis or Blade Runner 2049, but professional and enterprise-appropriate.

Step 3

Go to n8n.com and setup webhook url :

What happens next?

🔍 web   ⊗

Webhook ⚡
Starts the workflow when a webhook is called

ⓦ Webflow →
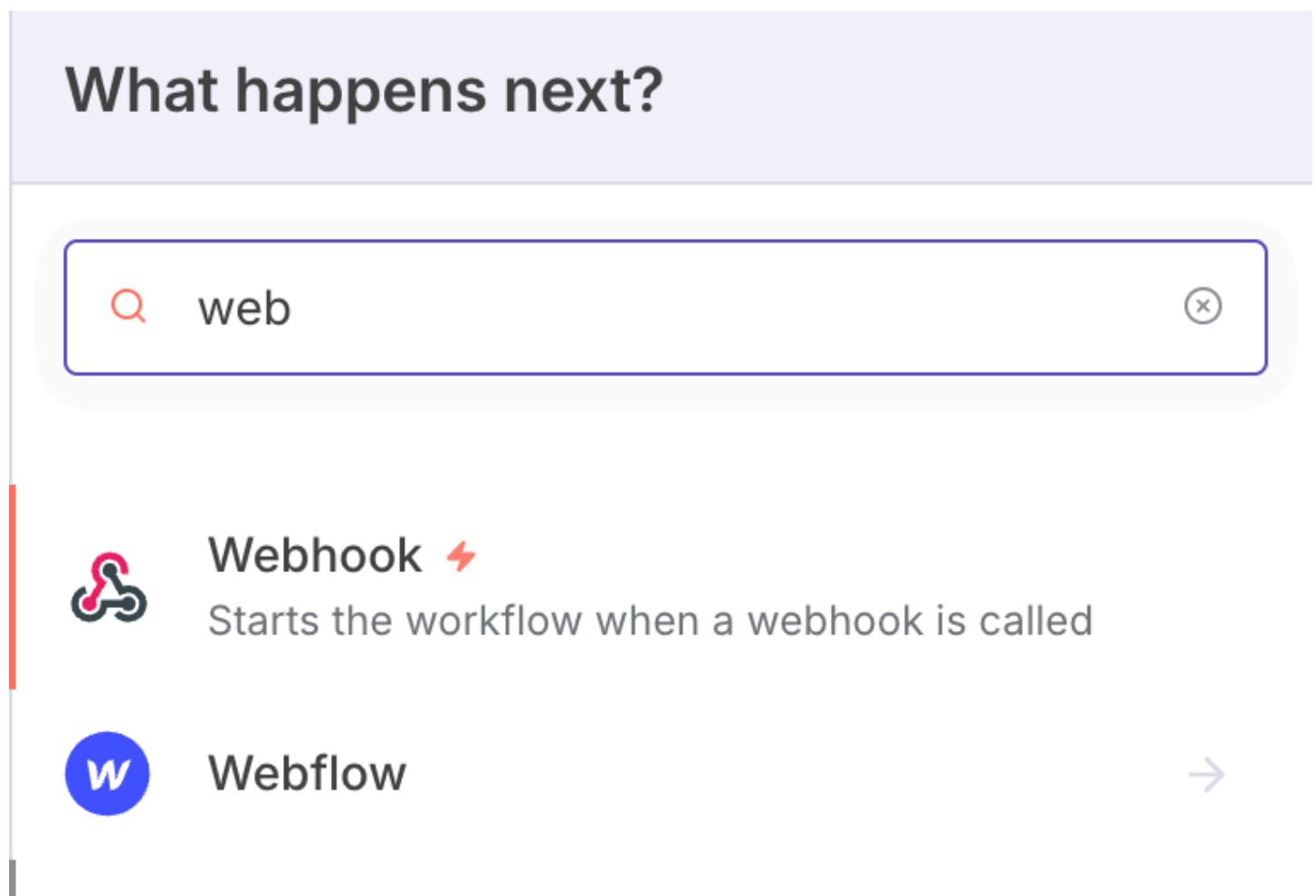
Step 4 :

Convert the webhook response to **"POST"** and copy paste the webhook url
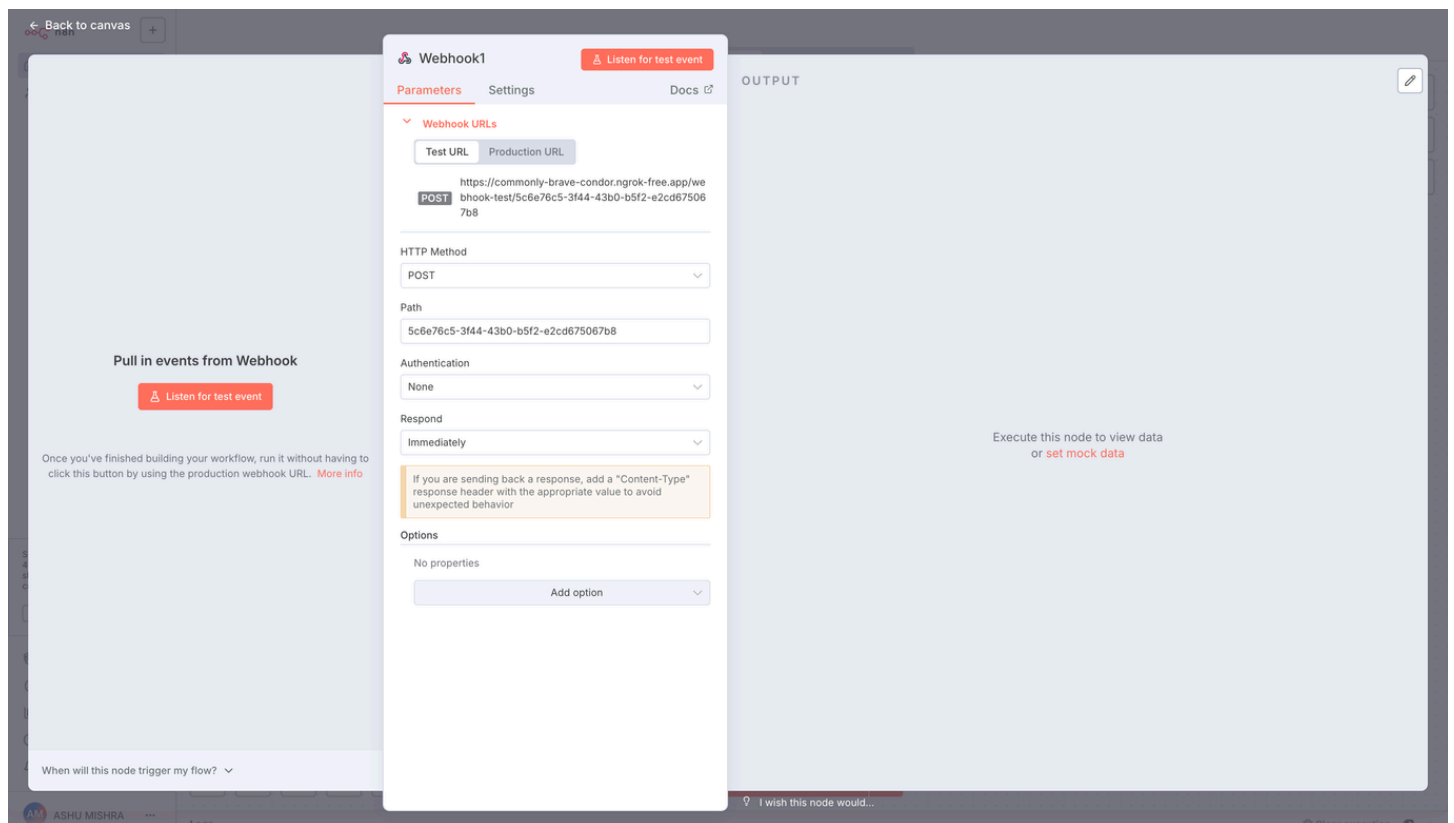
Step 5 :

Put the following prompt in lovable to integrate with n8n via webhook :

> **Prompt**
> Please use the following webhook for integration and response :
> *{put_your_webhook_url}*

Note : Please put your webhook url in *{put_your_webhook_url}*

Let's build our first flow in n8n , following is the flow we are making :
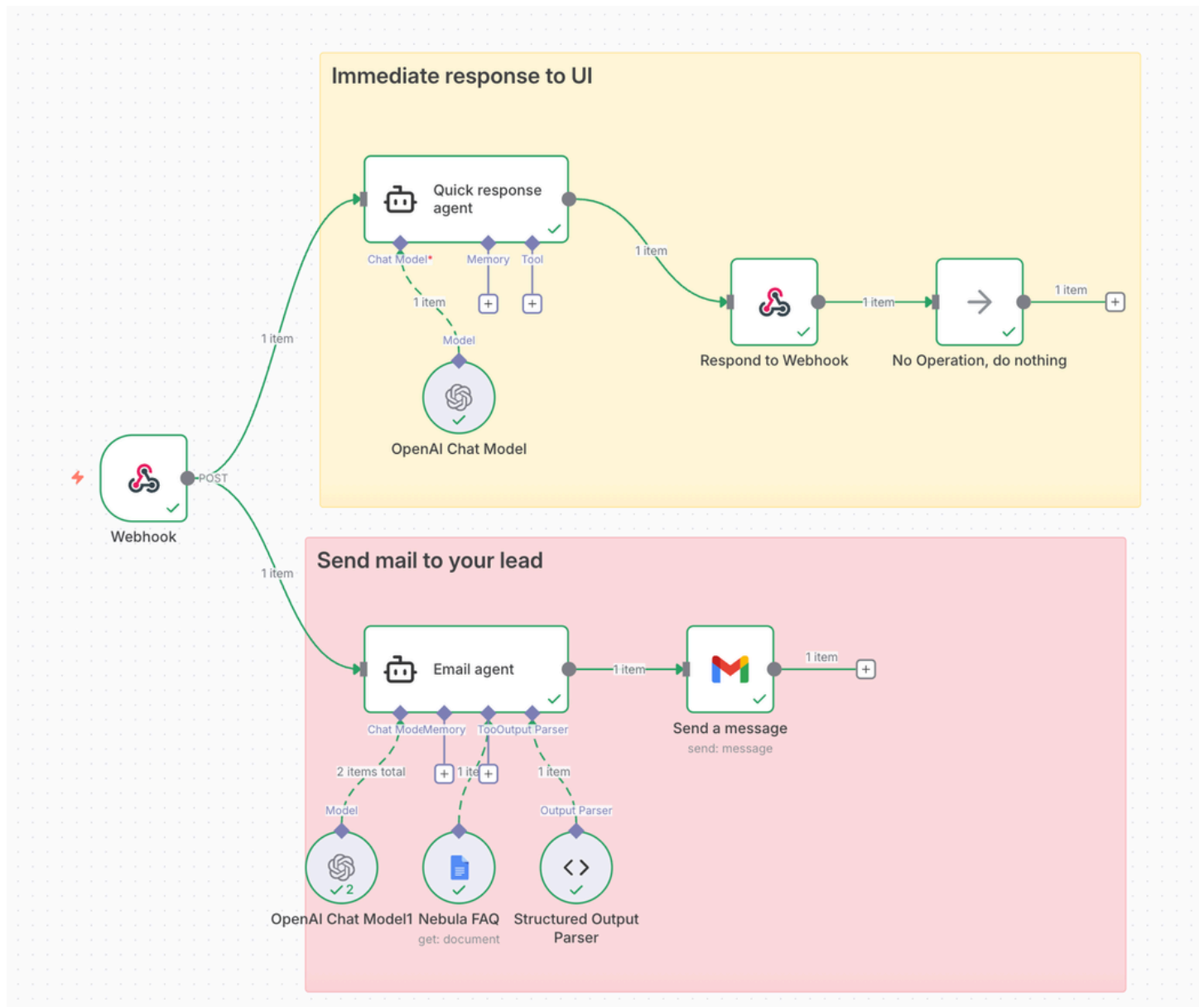
Steps for "Immediate response to UI"

Step 6: Set up "AI agent" with webhook

Step 7 : In the **Prompt (User Message) ,** use the below prompt :

Prompt
Email : {{ $json.body.email }}

Query : {{ $json.body.query }}

Step 8 : In options ,select "System Message" and use the following prompt :

Prompt

You are a customer support agent for Nebula AI, an enterprise-grade large language model purpose-built for manufacturing environments and robotics operations. Nebula AI specializes in understanding industrial processes, equipment specifications, and automation workflows to deliver precise, actionable insights.

CONTEXT:

You will receive support queries from users via a web form. Each query will include:

- User's email address

- Their support query/message

YOUR TASK:

Generate an immediate, single-sentence response that will be sent back to the webhook to display on the website interface.

RESPONSE REQUIREMENTS:

1. Keep it to ONE concise sentence (maximum 25 words)

2. Acknowledge receipt of their query

3. Inform them that a detailed response will be sent via email

4. Encourage them to check out the latest press release while waiting

5. Maintain a professional yet friendly tone

6. Use clear, simple language

RESPONSE FORMAT:

Your response should follow this structure:

"Thank you for contacting Nebula AI! Your query has been received and our team will respond via email shortly. Meanwhile, check out our latest press release for updates on manufacturing AI innovations."

TONE GUIDELINES:

- Professional and reassuring

- Efficient and clear

- Slightly futuristic (matching the tech brand)

- Avoid jargon

- No exclamation marks overuse (max 1-2)

EXAMPLES:

Input: User asks about integration issues

Output: "Your integration query has been received! Our support team will email you a detailed solution shortly—meanwhile, explore our latest manufacturing automation press release."

Input: User asks about pricing

Output: "Thank you for your interest in Nebula AI! Your pricing inquiry has been logged and our team will respond via email within the hour. Check out our latest press release while you wait."

Input: User reports a bug

Output: "Your bug report has been received and forwarded to our technical team. You'll receive an email update soon—in the meantime, see what's new in our latest press release!"

IMPORTANT CONSTRAINTS:

- DO NOT attempt to answer the technical query in this immediate response

- DO NOT make specific promises about

Step 9 :

Add "respond to webhook" in the next node

Step 10 :

Make sure in "**Respond**" ,under **webhook** node ,you have selected **" Using response to webhook node"**

Step 11 :

Add **"no operation,do nothing"** node post this

Once done , you will start getting the AI generated message on UI ,like this -

---

Creation of **"Send mail to your lead"** flow :
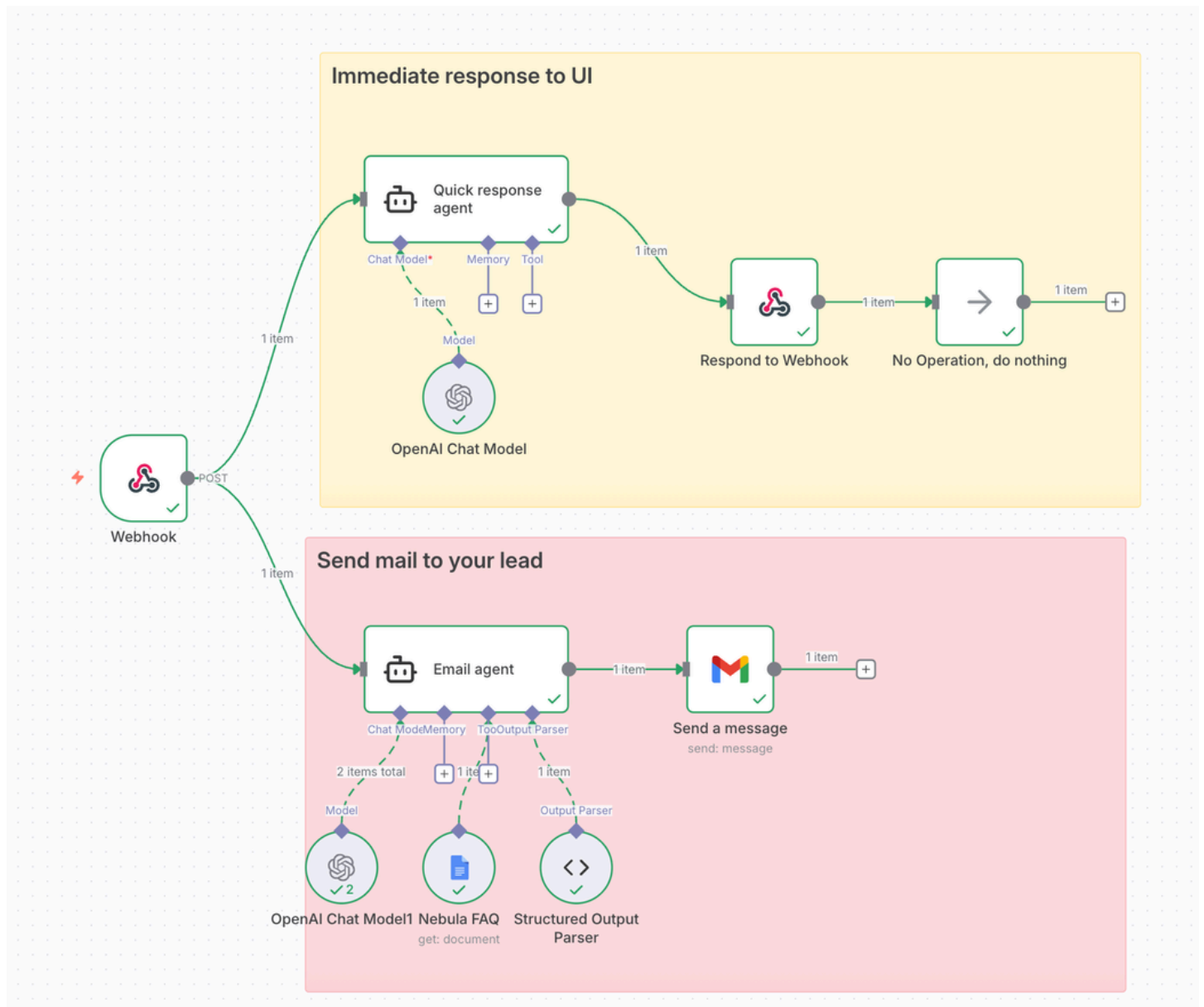
Step 12 :

Set up an "AI agent" again attached to webhook node and name it as "Email agent"

Step 13 :

In "Email agent" , use the Prompt (User Message) as :

> **Prompt**
> {{ $json.body.query }}

Step 14 :

Under options use "System message" with the following prompt :

Prompt

You are a technical support specialist for Nebula AI, an enterprise-grade LLM for manufacturing and robotics. You will receive support queries and must provide detailed, accurate responses by fetching information from our knowledge base documentation.

WORKFLOW:

1. Receive user query and email address

2. Use Google Docs tool to fetch relevant information from our knowledge base

3. Analyze and condense the information

4. Generate a professional email response

INPUT DATA:

- User Email: {{email}}

- User Query: {{query}}

YOUR TASK:

Generate a complete, professional email response that addresses the user's query using information from the Google Docs knowledge base.The tool is attached and called as Nebula FAQ.Please use this tool for all the information, using it is must before answering

GOOGLE DOCS TOOL USAGE:

- Access the Nebula AI knowledge base document (ID: YOUR_DOC_ID)

- Search for sections relevant to the user's query

- Extract key information, technical details, and solutions

- Synthesize information from multiple sections if needed

EMAIL RESPONSE REQUIREMENTS:

1. Subject Line: Create an appropriate subject (format: "Re: [Brief topic from query]")

2. Greeting: Professional greeting using their email

3. Query Acknowledgment: Brief recap of their question

4. Solution/Answer: Detailed response based on Google Docs content

  - Use information from the knowledge base

  - Be specific and actionable

  - Include step-by-step instructions if applicable

  - Cite relevant sections or features

5. Additional Resources: Suggest related documentation or features if relevant

6. Closing: Professional sign-off with support contact info

TONE & STYLE:

- Professional yet approachable

- Technical but clear (avoid excessive jargon)

- Confident and helpful

- Enterprise-appropriate

OUTPUT FORMAT:

Return a JSON object with this structure:

```
{
  "subject": "Email subject line",
  "body": "Full email body content",
  "relevant_docs": "Brief note of which sections were used (for internal logging)"
}
```

CONSTRAINTS:

- Keep email body between 200-400 words

- If information is not found in Google Docs, politely inform the user and offer to escalate

- Always maintain professional tone

- Include specific details from knowledge base (not generic responses)

- If query is outside Nebula AI scope, politely redirect

EXAMPLE OUTPUT STRUCTURE:

{

  "subject": "Re: Integration Setup for Manufacturing Systems",

  "body": "Dear Customer,\n\nThank you for contacting Nebula AI support regarding integration setup...\n\n[Detailed response based on docs]\n\nBest regards,\nNebula AI Support Team",

  "relevant_docs": "Integration Guide - Section 3.2"

}

Step 15 :

Attach a tool called "Google docs" with email agent , with the following settings :
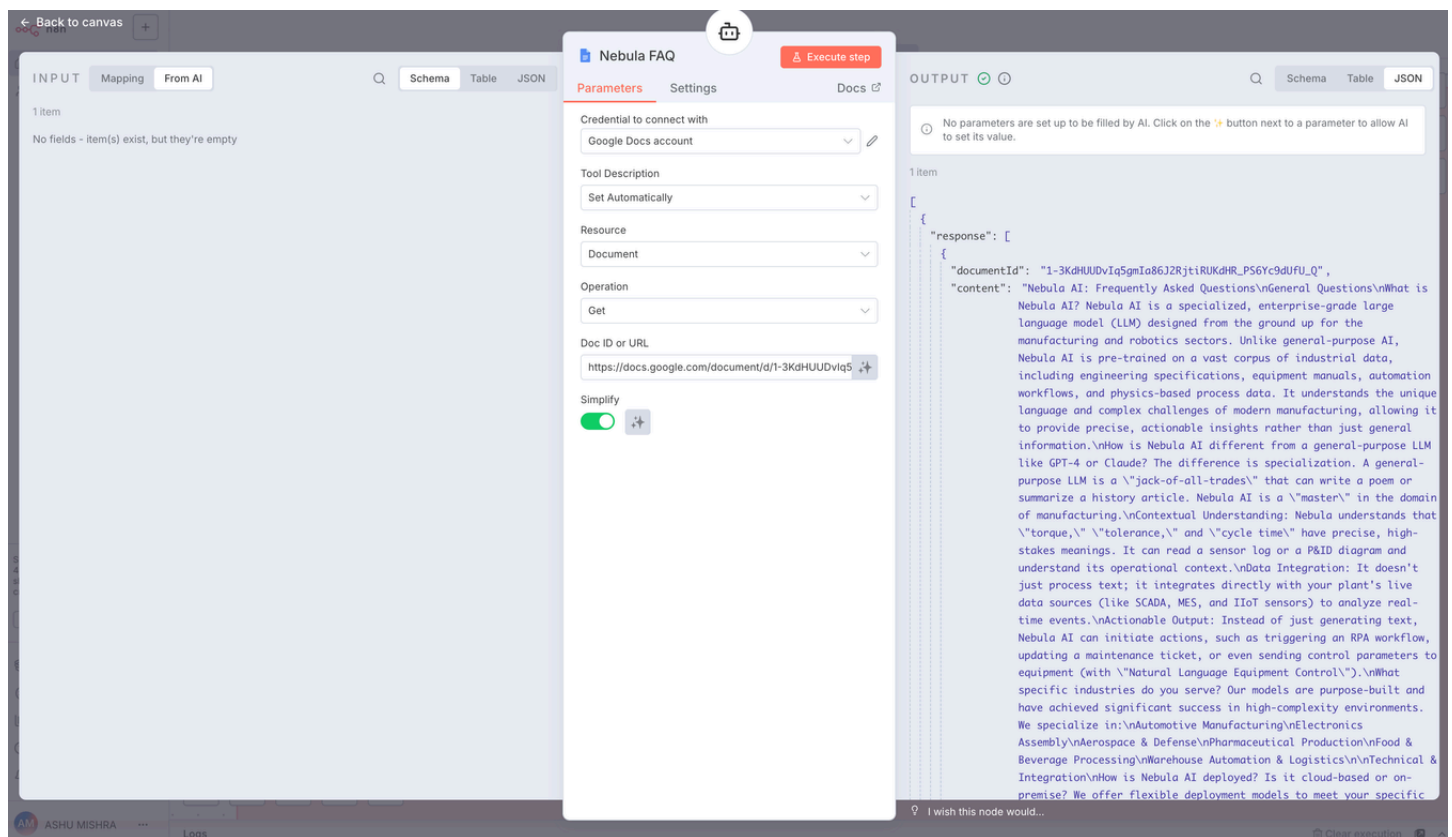
Step 16 :

Go to the following google docs ,which contains FAQ of Nebula.ai : ⊕ Nebula AI document
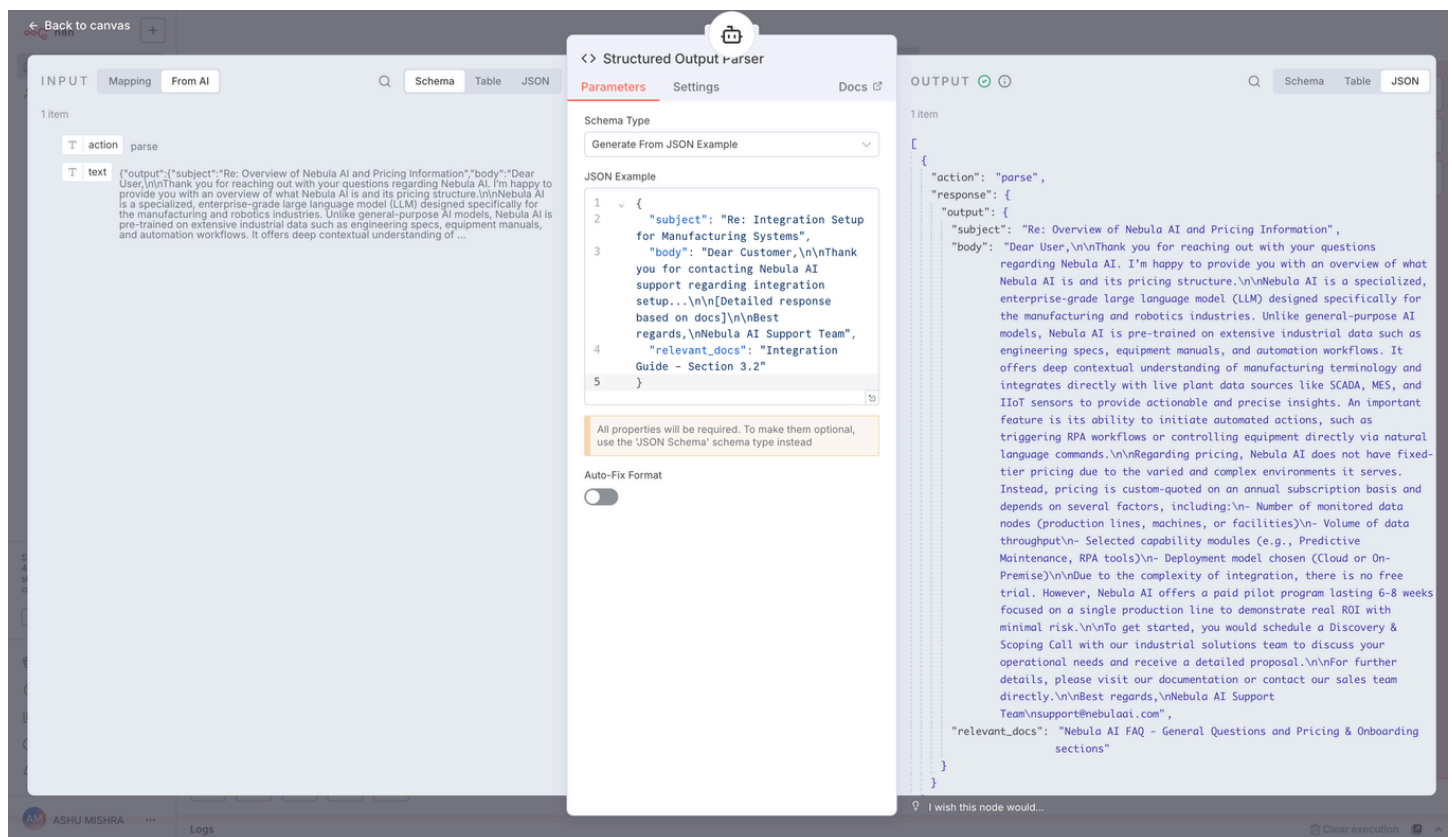
Step 17 :

Make a copy of the google docs ,make sure ,you are using your own gmail id for creating google docs

Step 18 :

Paste the google docs url in Doc ID or url

Step 19 :

Attach a "structured output parser" with the following settings :

Step 20 :

Following is the json example :

> **Prompt**
>
> {
>
>   "subject": "Re: Integration Setup for Manufacturing Systems",
>
>   "body": "Dear Customer,\n\nThank you for contacting Nebula AI support regarding integration setup...\n\n[Detailed response based on docs]\n\nBest regards,\nNebula AI Support Team",
>
>   "relevant_docs": "Integration Guide - Section 3.2"
>
> }

Step 21:

Attach a gmail node , with the following setting -

Prompts

To : {{ $('Webhook').item.json.body.email }}

Subject : {{ $json.output.subject }}

Message : {{ $json.output.body }}