NodeSpark

WORKFLOW ENGINE

Brilliant Automation

Designed for iOS

Version 1.0

NodeSpark User Guide

Section 1 – Getting Started & Core Concepts

What NodeSpark Is

NodeSpark is a **visual automation builder for iPhone**.

You design "workflows" made of blocks called **nodes**:

- **Trigger nodes** – decide *when* a workflow runs
- **AI nodes** – let large language models transform text and call tools
- **Logic nodes** – modify or branch your data
- **Action nodes** – send notifications, call APIs, run Shortcuts, etc.

You wire these nodes together on a canvas, then let NodeSpark run them automatically.

Typical things you can do:

- Turn messy notes into clean task lists and send them somewhere
- Summarize daily journals into a notification at the end of the day
- Fire webhooks into tools like n8n, Notion integrations, or internal APIs
- Kick off workflows from **Shortcuts**, schedules, app events, or external HTTP calls

---

1. Installing & First Launch

1. **Download NodeSpark** from the iOS App Store on your iPhone.
2. Open the app. The first time you launch, you'll see:
    - A short **onboarding flow** explaining what NodeSpark does.
    - A prompt asking for **Notification permissions** (NodeSpark uses local notifications for scheduled workflows and alerts).
3. Allow notifications so scheduled and alert-style workflows can reach you.

You can always adjust notification permissions later in **Settings → Notifications → NodeSpark**.

---

2. Understanding the Main Screens

After onboarding, you land on the **Workflow List** – your "Control Room" for automations.

2.1 Workflow List Header

At the top you'll see:

- **NodeSpark title & tagline** – just branding and context.
- A small status chip showing **"X workflows"** – how many you've created.
- Three main controls under the header:
  - **AI Console** – opens global AI configuration and preferences.
  - **New Workflow** – start a blank workflow from scratch.
  - **Templates** – opens the Template Gallery with prebuilt workflows.

Below that you'll see:

- A **"Pro tip" banner** explaining that:
  - **Long-press** a workflow card for Quick Run, Integrations, Run History, Duplicate.
- The **Free plan status** (if you're on the free tier):
  - Shows currentCount/2 workflows used and an **Upgrade** button.
  - Free plan = up to **2 workflows**. Upgrading unlocks unlimited workflows.

2.2 Workflow Cards

Each workflow appears as a card with:

- **Title** – the name you gave it (e.g. "Daily AI Summary")
- **Description** – short explanation of what it does

- **Meta row** – last run time or “Never run”

Tapping a card:

- Opens the **Workflow Editor** (node canvas) for that workflow.

Long-pressing a card opens a context menu with:

- **Quick Run** – test the workflow with manual input
- **Integrations** – see how it hooks into Shortcuts, webhooks, APIs, subflows, AI tools
- **Run History** – view previous runs and logs
- **Duplicate Workflow** – clone the workflow (respects the free plan limit)

Swipe left on a card to **delete** it.

---

3. Creating Your First Workflow

Let’s build your first automation using a template so you can see the whole flow end-to-end.

3.1 Using a Template: “Daily AI Summary”

1. On the **Workflow List**, tap **Templates**.
2. In the **Template Gallery**, look for **“Daily AI Summary”**.
3. Tap the card and then select **“Add to my workflows”**.

What this template does:

- Input: You paste daily notes, a Slack thread, or a brain dump.
- Steps:
  - It **summarizes** your text with an AI node.
  - It sends you a **local notification** with the summary.
- Output: A 3–5 bullet summary of your day with key decisions & next steps.

After adding it:

4. Close the Template Gallery.
5. On the Workflow List, you'll now see a card called **"Daily AI Summary"**.
6. Tap it to open the **Workflow Editor**.

---

4. The Workflow Editor (Canvas) – High-Level Tour

When you open a workflow, you'll see a clean node canvas:

- **Top bar**
  - Back arrow – returns to Workflow List
  - Workflow name – e.g. "Daily AI Summary"
  - Search / integrations / run button (depending on your build)
- **Nodes on the canvas**
  - **Start** node – how the workflow is triggered
  - **Summarize** node – AI step that transforms your text
  - **Notify** node – sends a notification
- **Connection lines**
  - Curved lines showing the flow: Start → Summarize → Notify.
- **Bottom drawers**
  - **Test Input & Output** – for running and inspecting sample payloads
  - **Workflow Variables** – define reusable values (API keys, constants, etc.)
  - **Add Nodes** – button to insert new nodes into the flow
- **Mini-map (bottom-right area)**
  - A small overview of the entire canvas.
  - Helps you navigate big workflows.

We'll go deep on each node type in another section; for now, let's just run this example.

---

5. Running Your First Workflow (Quick Run)

There are two main ways to run a workflow while you're still designing it: **Quick Run** and the **Test Input drawer**. Quick Run is the easiest for new users.

5.1 Launching Quick Run

1. Go back to the **Workflow List**.
2. Long-press the **"Daily AI Summary"** workflow card.
3. Tap **"Quick Run"**.

This opens the **Quick Run Console** for that workflow.

5.2 Quick Run Layout

You'll see:

- **Header**
  - Name of the workflow
  - Short description of what Quick Run does
- **Input section**
  - A big text area for your test input
  - Optional **presets** such as:
    - "Summarize email"
    - "Extract tasks"
    - "JSON payload"
- **Run button**
  - "Run Workflow" / "Running…"
- **Output section**
  - Shows the final payload after the workflow finishes
- **Run Log**
  - A list of log messages for each step
- **Agent Scratchpad** (when you use AI Agent mode)
  - Internal notes from the AI while it calls tools or plans

5.3 Running the Daily AI Summary

1. In **Quick Run → Input**, paste a long note or email. Example:

   "Today I met with the team about Q2 marketing. We agreed to redesign the landing page, set up a webinar, and launch a small paid ads test next month…"

2. Tap **Run Workflow**.

3. Watch:
    - The **Run button** will show a spinner while it executes.
    - The **log** will fill with step messages.
4. When it finishes:
    - Look at the **Output** section.

      You should see a clean summary, e.g.:

      • Redesigned Q2 landing page approved

      • Webinar planned for next month

      • Paid ads test scheduled

5. Close Quick Run when you're done.

Tip: Quick Run does **not** change how the workflow is triggered in real life. It's just a safe sandbox to test the flow.

---

6. Understanding the Free Plan vs Unlimited

Before we move into more advanced topics, it's important to understand how the plan limit works inside the UI.

- **Free Plan**
    - Up to **2 workflows** total (regardless of complexity).
    - You can:
        - Create, edit, delete workflows
        - Use AI nodes, HTTP calls, etc. inside those workflows
    - If you try to:
        - Create a 3rd workflow
        - Or duplicate an existing one when you already have 2

          …NodeSpark will show the **Upgrade** paywall.

- **Unlimited (paid)**
    - No workflow count limit.
    - Same features, just more room to build as many flows as you want.

The app enforces this in a few places:

- Tapping **New Workflow**
  - If you're at the free limit, you'll see the Upgrade flow instead of the "New Workflow" sheet.
- Using **Templates**
  - If adding a template would push you over the limit, you'll also see the Upgrade sheet.
- **Duplicate Workflow**
  - Also checks the limit and can route you to the paywall.

We'll cover managing purchases and restoring them in its own section later.

---

7. Where We're Going Next

Now that you know:

- How the main screens work
- How to create a workflow from a template
- How to run and test it
- How the free vs unlimited plan behaves

…the next logical section is:

Section 2 – Nodes in Detail: Trigger, AI, Logic, and Action

In that section we'll go **node by node**, explain every control, and start showing step-by-step scenarios like:

- Schedule-based workflows (daily reminders, weekly digests)
- Webhook-triggered workflows for tools like **n8n**
- AI nodes with Agent Mode and tools (Slack/Email/Weather)

Section 2 – Nodes in Detail

Every workflow in NodeSpark is just **nodes + connections**.

There are four main node types:

1. **Trigger** – *When* the workflow starts
2. **AI** – *How text is transformed or analyzed*
3. **Logic** – *How data is cleaned, transformed, or branched*
4. **Action** – *What actually happens in the real world*

We'll walk through each one, with practical examples and step-by-steps.

2.1 Node Basics

2.1.1 Adding a node

1. Open a workflow.
2. At the bottom, tap **Add Nodes**.
3. Choose the category:
    - **Trigger**, **AI**, **Logic**, **Action**, etc.
4. Pick the node you want (for example: **Start**, **Summarize**, **HTTP Request**).
5. The node appears on the canvas. Drag to reposition if you want.

2.1.2 Connecting nodes

1. Tap the **output handle** of a node (small circle at the right/bottom edge).
2. Drag to the **input handle** of the next node.
3. Release to connect.

The flow always goes **left → right** / **top → bottom**.

At run time, NodeSpark will:

- Start at the **Trigger node**
- Follow the connections step by step
- Pass the **payload** (your text/data) along the chain

2.2 Trigger Nodes

Trigger nodes decide **how this workflow starts**.

You usually have **one main Start/Trigger node**, but you can mix trigger types in bigger workflows.

2.2.1 Trigger Types

Depending on the app build, you'll see options like:

- **Manual run** – You run it from the app or Quick Run.
- **Schedule** – Runs on a time schedule (daily, weekly, etc.).
- **Webhook / External call** – Starts when an external system (like n8n or your server) sends a HTTP request.
- **App Event** – Starts when something happens inside NodeSpark (app launch, foreground, notification tapped).

You choose the type in the **Trigger node detail sheet**.

A. Manual Run

- Best for: testing, "run on demand" flows.
- Shows up as **Start – Manual run** on the canvas.

How to configure:

1. Tap the **Trigger node**.
2. In **Trigger Type**, choose **Manual run**.
3. Give it a **Title** (optional) and **Subtitle** if you like.

How to use:

- Open Quick Run, or tap the **Run** button in the editor, and NodeSpark starts here.

B. Schedule Trigger

- Best for: **"every day at X"**, **weekly digest**, **end-of-day summary**.

How to set one up:

1. Tap the Trigger node.
2. Set **Trigger Type: Schedule**.
3. Fill in the schedule fields, for example:
    - Frequency: **Daily**
    - Time: **8:00 PM**
    - Optional description: "Summarize today's notes"
4. Save the node.

NodeSpark will:

- Register the schedule with its **TriggerScheduler** when:
    - You open the app
    - And anytime workflows change
- Fire the workflow at that time, even if you're on another screen (as long as the app is allowed to send notifications / run in background).

Example – Daily Summary

- Trigger: **Schedule – 8:00 PM every day**
- Next node: **AI – Summarize**
- Last node: **Action – Notify**

Result: At 8 PM, NodeSpark pulls your input, summarizes, and sends a notification.

C. Webhook / External Call Trigger

This is what you'll use with **n8n, custom servers, or Zapier-style tools**.

High level:

- NodeSpark exposes a **webhook entry point** for the workflow.
- You send a **JSON body** from your external tool.
- NodeSpark starts the workflow with that JSON (or text) as the input payload.

We'll do a dedicated n8n section later, but for now:

How to configure:

1. Tap the Trigger node.
2. Set **Trigger Type: Webhook** (or "On webhook / external call").
3. Add notes so you remember how it's used, e.g.
    - "Called by n8n when a lead is created"
4. Connect it to your next nodes (AI / Logic / Actions).

The **Integration Center** for this workflow will later show:

- That it has a **webhook trigger**
- Example JSON you can send

D. App Event Trigger

These are internal events from NodeSpark itself:

- **On app launch**
- **On app foreground**
- **On app background**
- **On notification tapped**, etc.

You set the type in the Trigger node:

1. Trigger Type: **App Event**
2. Choose the specific event, e.g. **App Launch**.

Then in your workflow:

- Wire that trigger into whatever you want to happen when you open NodeSpark.

Example – "On Launch: Daily Motivation"

- Trigger: **App Event – App Launch**
- AI Node: generates a daily motivational message.
- Action Node: notify or write to a log.

When you open NodeSpark, that event fires and the workflow runs.

We'll cover a full scenario for this in a later section.

2.3 AI Nodes

AI nodes let you **transform text, extract structure, or drive agent-style tools**.

2.3.1 Basic AI Node Layout

When you tap an AI node (like **Summarize**), you'll see:

- **Title** – what this step is called on the canvas (e.g. "Summarize")
- **Subtitle** – short description (e.g. "Summarize input text")
- **AI Mode** – presets:

    - Summarize
    - Extract tasks
    - Friendly rewrite
    - Custom
- **Instruction to AI** – the actual prompt you send to the model
- **Agent Mode (multi-step)** – optional switch for "tool-using" agent behavior
- Sometimes **tool settings** (subflows, webhooks, etc.) in advanced builds

2.3.2 Using AI Modes

The mode mostly pre-configures the **Instruction to AI** for you.

A. Summarize mode

- Goal: Turn long text into a clear, short summary.
- Sample instruction:

  "Summarize the payload in 1–2 concise, friendly sentences."

Use it for:

- Daily journals
- Long emails
- Meeting notes

B. Extract Tasks mode

- Goal: Extract bullet-point tasks from messy text.
- Instruction example:

  "Read the payload and output a clear task list with due dates, using bullet points."

Good for:

- Brain dumps you want to turn into a todo list
- Meeting notes with action items

C. Friendly Rewrite mode

- Goal: Rewrite text in a specific tone (kind, professional, casual, etc.).
- Use this if you're drafting replies to messages, emails, etc.

D. Custom mode

- Full manual control.
- You write the prompt exactly how you want:
    - "Transform this payload into a JSON object with fields title, dueDate, priority…"
    - "Answer the user's question concisely and do not include disclaimers…"

2.3.3 AI Node Example – Cleaning Up Tasks

Scenario: You have messy text from Apple Notes and want a clean list.

1. Add an **AI node** after your trigger.
2. Set **AI Mode: Extract tasks**.
3. Set **Instruction to AI** to something like:

   "From the payload, extract a clean list of tasks. For each task, include: title, optional due date, and priority (low/medium/high). Return them as bullet points."

4. Connect the AI node to:
    - A **Logic node** (to transform case, trim, etc.), or
    - An **Action node** that sends the result somewhere.

Run the workflow via Quick Run to see the transformation.

2.3.4 Agent Mode (multi-step)

Some builds of NodeSpark include **Agent Mode** on AI steps:

- When enabled, the AI can:
  - Take multiple reasoning steps
  - Call tools (subflows, webhooks, HTTP actions) you expose
  - Maintain an internal **scratchpad** that you can inspect in Quick Run

Basic idea:

1. Turn on **Agent Mode** in the AI node.
2. (Later sections) Configure which tools it can call (subflows, HTTP endpoints, etc.).
3. When you run the workflow, the **Agent Scratchpad** panel in Quick Run shows its thought process and tool calls.

We'll go deeper on Agent Mode when we talk about **AI + APIs + subflows**.

2.4 Logic Nodes

Logic nodes are your **text + control utilities**:

- Uppercase
- Lowercase
- Trim
- Condition (if/else behavior)

On the Logic node sheet, you'll see a **Logic Type picker**.

2.4.1 Simple Transformations

A. Uppercase

- Takes the payload and turns it into ALL CAPS.

- Good for quick formatting before a notification or API call.

Steps:

1. Add a **Logic Node** after an AI or trigger.
2. Set **Logic Type: Uppercase**.
3. Connect to your Action node.
4. Quick Run the workflow to confirm.

B. Lowercase

- Same thing, but all lowercased.

C. Trim

- Removes leading/trailing spaces, sometimes extra line breaks.
- Good hygiene step before sending to external APIs (JSON, URLs, etc.).

2.4.2 Condition Logic (if / else)

Depending on your build, the **Condition** type lets you check something about the payload and route differently.

Typical pattern:

- Input payload is text or JSON.
- Logic node in **Condition** mode has:
  - A condition expression (e.g., "contains 'error'" or length greater than X).
- Output:
  - Can set a flag, or
  - Send to different outputs (if your build has branch outputs).

We'll use Condition in later "scenario" sections for:

- "If the AI detects an error, send a notification, otherwise just log it"
- "If payload contains the word 'urgent', route to a different channel"

2.5 Action Nodes

Action nodes are where NodeSpark reaches outside itself.

Common modes:

- **Notify** – local notification
- **HTTP Request** – call web APIs, your server, or n8n webhook URLs
- **Run Shortcut** – call into the iOS Shortcuts app
- **Run Subflow** – run another NodeSpark workflow as a subroutine
- (Others, depending on what we wired in your project)

2.5.1 Notify (Local Notification)

Used in the "Daily AI Summary" template.

Fields you'll usually see:

- **Title** – notification title
- **Body** – uses the current payload (or a prefix + payload)
- Optional settings: sound, category, etc.

Typical pattern:

1. AI node generates text.
2. Action (Notify) sends that text as a notification.

Example:

- Title: “Daily Summary”
- Body: Use payload as the full message.

2.5.2 HTTP Request

This is your gateway to **n8n, internal APIs, Telegram bots, etc.**

Key fields:

- **URL** – endpoint (e.g. your n8n webhook URL)
- **Method** – GET, POST, PUT, PATCH, DELETE (usually POST)
- **Headers** – e.g. Authorization: Bearer <token>
- **Body** – text or JSON (often you send the payload as JSON)

Common usage:

- Send the AI’s result to n8n, which then:
    - Writes to Notion
    - Sends to Slack
    - Updates a database
    - Triggers another chain of automations

We’ll have a whole section just on **n8n + HTTP + auth**.

2.5.3 Run Shortcut

This action lets NodeSpark trigger a **Shortcut** by name and **pass the payload as input**.

Fields:

- **Shortcut Name** – must match the name in the Shortcuts app.
- **Input** – typically the workflow payload.

Classic pattern:

- NodeSpark prepares or transforms text.
- Shortcuts then:
  - Saves it to a file
  - Posts to a specific app
  - Uses any of Apple's native integrations

We'll build a complete "NodeSpark → Shortcuts → App" example in the Shortcuts section.

2.5.4 Run Subflow

This action calls **another NodeSpark workflow** as a subroutine, passing along the payload.

Use cases:

- Build small reusable flows:
  - "Clean & normalize text"
  - "Log run to storage"
  - "Send generic Slack message via HTTP"
- Call them from multiple parent workflows.

Fields:

- **Subflow Workflow Name** – must match exactly.
- Optional mapping of input/output (depending on how your project is wired).

In Integration Center, these subflows show up under **"Subflows & AI Tools"** so you can see the structure.

2.6 Node Detail Sheets – Common Controls

Whenever you tap a node, you're basically editing that node's **detail sheet**:

Common fields across types:

- **Node Title** – what appears on the canvas card
- **Subtitle** – a short description
- **Mode / Type** – (AI Mode, Trigger Type, Logic Type, Action Mode)
- Additional parameters:
    - Text instructions (AI)
    - URLs & methods (HTTP)
    - Schedule settings (Triggers)
    - Siri phrase hints, etc.

General tips:

- Keep titles short but descriptive:

  "Summarize meeting", "Push to n8n", "Condition: urgent?", etc.

- Use subtitles to remind your future self what this node is doing.
- If something is important to remember (like what API it hits), use the description / notes field so Integration Center can show it.

2.7 Putting It Together – A Simple Multi-Node Example

Let's build a small workflow using all four node types:

Goal:

Take a long note → clean it → extract tasks → send them to you via notification.

1. **Trigger Node**
    - Type: **Manual run** (for now).
    - Title: "Start"
    - Subtitle: "Manual run"
2. **AI Node – Extract Tasks**
    - Title: "Extract tasks"
    - Mode: **Extract tasks**
    - Instruction:

      "From the payload, extract a clear list of tasks with short titles. Output as bullet points."

3. **Logic Node – Trim**
    - Title: "Trim output"
    - Logic Type: **Trim**
4. **Action Node – Notify**
    - Title: "Notify"
    - Body: Use the payload as the notification body.

Connect them:

Trigger → AI (Extract tasks) → Logic (Trim) → Action (Notify)

Test it:

1. Long-press the workflow on the Workflow List → **Quick Run**.
2. Paste some messy text:

   "Today: follow up with Mike about the proposal, renew the domain name by Friday, prep slides for next Monday's meeting…"

3. Run the workflow.
4. Check:
    - Output: a clean bullet list of tasks.
    - Notification: you'll get the same list as a local notification.

You just used **Trigger + AI + Logic + Action** in one flow.

What's Next

Now that you know the nodes, next sections can go into:

- **Section 3 – AI + APIs + Webhooks:**

  Connecting to n8n, HTTP APIs, Telegram bots, auth patterns, etc.

Section 3 – AI + APIs + Webhooks

NodeSpark gets really powerful when you let AI talk to the outside world:

- Send AI-generated text into **n8n** or other automation servers
- Hit any **HTTP API** (Slack, Notion, internal tools)
- Push messages into **Telegram** or other chat apps
- Chain everything together with **AI + Logic + Actions**

This section walks through:

1. How NodeSpark's **payload** flows into HTTP APIs
2. How to configure an **HTTP Action node**
3. How to add **auth headers** and variables
4. A full **n8n integration example**
5. A full **Telegram bot example**
6. Error handling and best practices

3.1 Mental Model – AI + HTTP

Every workflow passes around a **payload** (usually text or JSON).

Typical pattern:

1. **Trigger** – gets some starting text or JSON.
2. **AI Node** – rewrites, summarizes, or structures it.
3. **Logic Node** – cleans or transforms (uppercase, trim, condition).
4. **Action – HTTP Request** – sends that final payload to an API.

You can think of the HTTP Action node as:

"Take whatever is in the payload right now and POST it to this URL."

You choose:

- Which **URL** to call (e.g. n8n webhook, Telegram API)
- Which **method** (GET / POST / etc.)
- Which **headers** (Authorization, Content-Type, etc.)
- What to send in the **body** (raw text or JSON)

3.2 Configuring an HTTP Action Node

1. Open a workflow.
2. Tap **Add Nodes → Action → HTTP Request** (or similar label, depending on your build).
3. Tap the node to open its detail sheet.

You'll see fields like:

- **Title / Subtitle**
  - For the canvas: e.g.
    - Title: Send to n8n
    - Subtitle: POST summary to webhook
- **Action Mode**
  - Set to **HTTP Request**.
- **URL**
  - The endpoint you want to call, e.g.

    https://your-n8n-instance.com/webhook/daily-summary

- **Method**
  - Usually **POST** for sending data.
- **Headers**
  - Key–value pairs, such as:
    - Content-Type: application/json
    - Authorization: Bearer {{n8nToken}}
- **Body**
  - Raw text or JSON. Often you'll send the **payload** here.

3.2.1 Sending the payload as JSON

A very common pattern is:

```json
{
  "workflowName": "{{workflowName}}",
  "payload": "{{payload}}",
  "timestamp": "{{now}}"
}
```

Where:

- {{payload}} = current workflow payload
- {{workflowName}} = name of this workflow
- {{now}} = current timestamp

(Your exact variable names depend on how you set up **Workflow Variables**; see below.)

3.3 Using Workflow Variables & API Keys

You don't want to hard-code secrets directly into nodes.

Use **Workflow Variables** (drawer at the bottom of the canvas):

1. Open the workflow.
2. Tap **Workflow Variables**.
3. Add variables, for example:
    - n8nToken
    - telegramToken
    - telegramChatId

4. Give each a value (API keys, chat IDs, etc.).

In your nodes, reference them with placeholders, for example:

- Header:
  - Authorization: Bearer {{n8nToken}}
- URL:
  - https://api.telegram.org/bot{{telegramToken}}/sendMessage
- JSON body:
  - "chat_id": "{{telegramChatId}}"

At run time, NodeSpark replaces {{variableName}} with the actual value.

Tips

- Keep secrets in variables; don't screenshot those screens.
- Use different variables per workflow if you have multiple services.

3.4 Example: AI → n8n Webhook (Daily Summary to Notion via n8n)

In this example:

- NodeSpark creates a **Daily AI Summary** of your notes.
- It sends that summary to **n8n** via an HTTP webhook.
- n8n then writes the summary into Notion (or anywhere else you wire it).

3.4.1 Step 1 – Create the workflow in NodeSpark

You can start from your existing **"Daily AI Summary"** template or build it fresh.

Nodes layout:

1. **Trigger – Schedule**
    - Trigger Type: Schedule
    - Example: **Every day at 8:00 PM**
    - Title: Daily summary trigger
2. **AI Node – Summarize**
    - Title: Summarize day
    - AI Mode: Summarize
    - Instruction:

      "Summarize the payload in 3–5 bullet points with key decisions and next steps."

3. **Action – HTTP Request**
    - Title: Send to n8n
    - Action Mode: HTTP Request
    - Method: POST
    - URL: your n8n webhook URL (we'll get this below)
    - Headers:
        - Content-Type: application/json
    - Body (JSON), for example:

```json
{
  "workflow": "{{workflowName}}",
  "summary": "{{payload}}",
  "timestamp": "{{now}}"
}
```

Connect them:

Trigger (Schedule) → AI (Summarize day) → Action (Send to n8n)

Connect them:

Trigger (Schedule) → AI (Summarize day) → Action (Send to n8n)

3.4.2 Step 2 – Create the n8n workflow

On your n8n server:

1. Create a new workflow.
2. Add a **Webhook** node:
    - HTTP Method: **POST**
    - Path: e.g. daily-node-spark-summary
    - Keep **"Response mode"** as default for now.
3. Copy the **Production Webhook URL** (you'll paste this into NodeSpark's HTTP URL).
4. Add your next nodes in n8n, for example:

- **Notion** node:
    - Create a new page in a specific database.
    - Title: something like Daily Summary – {{ $json.timestamp }}
    - Content: {{$json.summary}}

Or:

- **Slack** node:
    - Channel: #daily-updates
    - Text: {{$json.summary}}

4. Activate the n8n workflow (toggle switch to "Active").

3.4.3 Step 3 – Connect NodeSpark to the n8n webhook

Back in NodeSpark's **HTTP Request** node:

1. Paste the **Production Webhook URL** into the **URL** field.
2. Make sure:
    - Method: **POST**
    - Header: Content-Type: application/json
    - Body JSON includes summary and timestamp like the sample above.

3.4.4 Step 4 – Test with Quick Run

Before waiting for the schedule:

1. Long-press the workflow card in the list.
2. Tap **Quick Run**.
3. In the **Input** box, paste some sample notes:

   "Today: worked on new marketing plan, closed 3 deals, discussed launch timeline…"

4. Tap **Run Workflow**.

Check:

- **Quick Run Output**: you should see the AI summary.
- **Run Log**: should show a successful HTTP call.
- In **n8n / Notion / Slack**: your summary should appear.

If the HTTP call fails, see **3.7 – Troubleshooting** below.

Once it works, the scheduled trigger will fire the same flow every day at 8 PM.

3.5 Example: AI → Telegram Bot (Send yourself messages)

Telegram's Bot API is a classic HTTP integration.

Goal:

- NodeSpark formats a summary or notification.
- It hits https://api.telegram.org/bot<token>/sendMessage.
- You get the message in a Telegram chat.

3.5.1 Step 1 – Create your Telegram bot & get IDs

1. In Telegram, start a chat with **@BotFather**.
2. Send /newbot and follow the prompts.
3. BotFather gives you a **bot token**, for example:
    - 123456789:AAExampleTokenFromBotFather
4. Add the bot to a chat (yourself, a friend, or a group).
5. Use something like @RawDataBot or a simple n8n flow to obtain the **chat ID**, or:
    - Send a message, then check with your own API call:
        - https://api.telegram.org/bot<token>/getUpdates
        - Look for chat → id.

3.5.2 Step 2 – Save secrets as Workflow Variables

In NodeSpark:

1. Open your workflow (or create a new "Telegram Alert" workflow).
2. Open **Workflow Variables**.
3. Add:
    - telegramToken → your bot token
    - telegramChatId → your chat ID

3.5.3 Step 3 – Build the workflow in NodeSpark

Nodes:

1. **Trigger – Manual run** (or schedule, or webhook if you want external events to trigger it).
2. Optional **AI Node – Format message**
    - Mode: Custom
    - Instruction example:

      "Take the payload and format a short, friendly message for Telegram. Keep it under 6 lines, use emojis sparingly."

3. **Action – HTTP Request**:
    - Title: Send Telegram message
    - Method: POST
    - URL:

https://api.telegram.org/bot{{telegramToken}}/sendMessage

3.
    - Headers:
        - Content-Type: application/json
    - Body:

```
json

{
  "chat_id": "{{telegramChatId}}",
  "text": "{{payload}}"
}
```

Connect them:

Trigger → (AI node, optional) → HTTP Request (Send Telegram message)

3.5.4 Step 4 – Test it

1. Long-press the workflow → **Quick Run**.

2. Input text:

   "New lead from website: John Doe, wants demo on Friday at 3 PM."

3. Run the workflow.

Result:

- AI formats a nice message (if you used the AI node).
- HTTP Request hits Telegram.
- You see the message in your Telegram chat almost instantly.

Now you can wire this same pattern behind:

- A schedule (daily summary to Telegram)
- A webhook (n8n calls NodeSpark or vice versa)
- A Shortcut (voice-triggered, see Shortcuts section)

3.6 AI + APIs Round-Trip Patterns (Advanced)

Once you're comfortable with HTTP, you can start doing more advanced patterns:

3.6.1 AI prepares, API executes

- **AI Node**:
  - Takes natural language, turns it into structured JSON.
- **Logic Node**:
  - Optionally validates or transforms.
- **HTTP Action**:
  - Sends that JSON into n8n / your server.

Example:

- AI converts free-form "today's sales" notes into a JSON object with keys:

    - date, totalSales, topCustomer, openIssues
- HTTP action sends that JSON to a server that writes it to a database or dashboard.

3.6.2 API responds, AI interprets

You can also design flows where:

1. HTTP Action calls an API that returns JSON (e.g. an external system).
2. That JSON becomes the new payload.
3. An AI Node then:
    - Explains the data in plain English,
    - Picks out anomalies,
    - Suggests next actions.

Depending on how your version of NodeSpark is wired, this may be a single or multiple steps. The pattern is the same: **API → payload → AI explanation.**

3.7 Troubleshooting & Logs

When something goes wrong with APIs, Quick Run is your friend.

3.7.1 Use Quick Run

1. Long-press the workflow in the list.
2. Tap **Quick Run**.
3. Provide test input and tap **Run Workflow**.

Watch:

- **Run Log** section:
    - If the HTTP call fails, you'll see an entry like:

        - HTTP 401 – Unauthorized
        - HTTP 404 – Not Found
        - Network error – ...
- **Output**:
    - Might contain error response text from the API.

3.7.2 Common issues

- **401 Unauthorized**
    - Check your Authorization header.
    - Make sure the token/secret is correct.
    - Confirm the variable name matches exactly (e.g. {{n8nToken}} vs {{n8ntoken}}).
- **404 Not Found**
    - Double-check the URL path.
    - For n8n, make sure the workflow is **active** and you're using the **Production** URL.
- **Timeout / Network error**
    - Check your phone's network connection.
    - Test the endpoint from another client (browser, Postman) to make sure the server is up.
- **Bad JSON**
    - If the API returns errors about malformed JSON:
        - Validate your request body with an online JSON validator.
        - Make sure you wrapped strings in quotes and commas are correct.

3.8 Security & Best Practices

A few things to keep in mind:

- **Never commit or screenshot secrets.**

  Keep API keys in **Workflow Variables**, not in screenshots or obvious text.

- **Use separate tokens per environment.**

  For example:

    - n8nTokenDev, n8nTokenProd
    - Then swap which variable you use.
- **Limit permissions.**

Use scopes/permissions on the API side so a leaked token can't wreck everything.

- **Test with throwaway data first.**

  Before pointing at your real Notion workspace, Slack channel, or production DB.

- **Use logs to understand behavior.**

  Quick Run's **Run Log** + Integration Center's overview help you see:

  - Which triggers are configured
  - Which APIs this workflow is contacting
  - How many HTTP endpoints are in play

That wraps up **Section 3 – AI + APIs + Webhooks**.

.

Section 4 – Shortcuts & Siri

One of the biggest superpowers of NodeSpark is how it plugs into **Shortcuts** and **Siri**:

- Trigger any NodeSpark workflow with your voice
- Chain NodeSpark into larger automations (open apps, send messages, log to Notes, etc.)
- Let Siri hand NodeSpark some text, and let NodeSpark hand results back to Shortcuts

This section covers:

1. How to **run a NodeSpark workflow from Shortcuts**
2. How to **assign a Siri phrase** ("Run Daily Summary in NodeSpark")
3. How to **pipe text into NodeSpark** from Shortcuts
4. How to **use NodeSpark's output** inside a Shortcut
5. How to **run Shortcuts from inside a NodeSpark workflow** (the other direction)
6. Real-world **example automations** you can set up in a few minutes

---

4.1 Where Shortcuts & Siri Fit In

Here's the mental model:

- **Shortcuts → NodeSpark**

  Use the "Run workflow in NodeSpark" action in Shortcuts to trigger any workflow you've built.

- **NodeSpark → Shortcuts**

  Use an **Action node** set to "Run Shortcut" in your workflow to call a specific Shortcut from inside NodeSpark.

With these two directions, you can build:

- "Hey Siri, **summarize my notes and text them to me**."
- "Take this voice note, **turn it into tasks with AI**, then **add them to Reminders**."

- "When I tap this Home Screen icon, **run my AI planner**, then **launch Calendar**."

4.2 Prerequisites

Before you start, make sure:

- NodeSpark is installed and you've **created at least one workflow**.
- The **Shortcuts** app is installed (it's built-in on iOS).
- You've opened NodeSpark at least once after installing/updating it (iOS uses that to discover its Shortcuts actions).

4.3 Running a NodeSpark Workflow from Shortcuts

This is the base move: one tap (or voice command) runs a workflow you built in NodeSpark.

4.3.1 Create the Shortcut

1. Open the **Shortcuts** app on your iPhone.
2. Tap **+** in the top-right to create a **New Shortcut**.
3. Tap **Add Action**.
4. In the search bar, type **"NodeSpark"**.
5. You should see an action like:
    - **"Run NodeSpark Workflow"**

      or

    - **"Run Workflow in NodeSpark"**

      (Exact wording may vary slightly, but look for NodeSpark's action.)
6. Tap that action to add it.

You'll now see an action with a parameter like:

- **Workflow**: *(Choose)*

4.3.2 Choose Which Workflow Runs

1. Tap the **Workflow** parameter inside the NodeSpark action.
2. A list of your NodeSpark workflows appears (same ones you see on the main screen).
3. Tap the workflow you want (e.g. Daily AI Summary, Email Draft Helper, Telegram Alert).

That's it: your Shortcut now calls that workflow.

4.3.3 Test the Shortcut

1. In Shortcuts, tap the **play (▶)** button at the bottom.
2. Watch NodeSpark run that workflow in the background.
3. Depending on the workflow, it may:
    - Send notifications
    - Hit webhooks / APIs
    - Copy content to clipboard
    - Trigger Shortcuts or other apps (if you wired those in)

4.4 Passing Text / Data Into NodeSpark from Shortcuts

By default, your NodeSpark workflow starts with its own trigger / default payload.

But you can also send **dynamic input** from Shortcuts.

Example: "Hey Siri, log this idea" → Siri asks what you want to log → NodeSpark processes it.

4.4.1 Add an "Ask for Input" step

1. In your Shortcut, above the NodeSpark action, tap + to add another action.
2. Search for **"Ask for Input"** and add it.
3. Configure it:
    - Prompt: What would you like to send to NodeSpark?
    - Input Type: Text.

This creates a variable called **Provided Input** (or similar).

4.4.2 Feed that input into NodeSpark

Depending on how the NodeSpark action exposes parameters in your build, you'll see something like:

- **Input Text**, **Payload**, or similar.

If the action supports input:

1. Tap the **Input / Text / Payload** field in the NodeSpark action.
2. Choose **"Provided Input"** (the variable from Ask for Input).

Now, when you run the Shortcut:

1. Siri (or Shortcuts) asks you: *"What would you like to send to NodeSpark?"*
2. You speak or type your text.
3. That text becomes the initial **payload** inside your NodeSpark workflow.

Inside NodeSpark, your workflow can:

- Feed that text into an **AI Node** (summarize, rewrite, extract tasks, etc.)
- Pass it through **Logic nodes**.
- Send it to your APIs via **HTTP Action nodes**.

4.5 Using NodeSpark's Output in Shortcuts

You can also pull the result back into Shortcuts.

If the NodeSpark Shortcuts action is configured to **return a result** (text):

- The NodeSpark action becomes a **block that outputs text**, which you can use in further steps.

Example: Display the result in a notification

1. After the NodeSpark action, tap **+** to add another action.
2. Search for **"Show Result"** or **"Show Notification"**.
3. For the text field, tap and select the **NodeSpark result** variable.

Now your Shortcut:

1. Sends the input into NodeSpark.
2. NodeSpark runs the full workflow (AI, logic, APIs).
3. Shortcuts shows you the final text result in a banner or full-screen alert.

You can also:

- **Copy to clipboard** (using "Copy to Clipboard").
- **Send as a message** (using "Send Message").
- **Add to Notes** (using "Create Note").
- **Add to Reminders / Calendar** (using those actions).

4.6 Assigning a Siri Phrase to a NodeSpark Workflow

Once your Shortcut is working, you can turn it into a voice command.

4.6.1 Add a Siri phrase

1. Open the **Shortcuts** app.
2. Tap the **"…"** on your NodeSpark Shortcut.
3. Find and tap **"Add to Siri"** (or **"Add to Home Screen / Details → Add to Siri"**, depending on iOS version).
4. Record a phrase, for example:
    - "Run Daily Summary"
    - "Log idea in NodeSpark"
    - "Plan my morning with NodeSpark"

Tap **Done** to save.

Now you can say:

"Hey Siri, Run Daily Summary."

Siri runs your Shortcut, which runs the NodeSpark workflow behind it.

Works from:

- iPhone lock screen
- Apple Watch (if Shortcuts sync is enabled)
- CarPlay (for supported actions)

4.7 Adding NodeSpark Shortcuts to the Home Screen

If you want one-tap access:

1. In the Shortcuts app, open your NodeSpark shortcut.

2. Tap the **"…"** button → then **"…"** again for details.
3. Tap **"Add to Home Screen"**.
4. Customize:
    - Name (e.g. "Daily Summary").
    - Icon (optional, you can pick a glyph or photo).
5. Tap **Add**.

Now tapping that icon on your Home Screen runs the Shortcut, which runs your NodeSpark workflow. Great for "Dashboard-style" workflows you open constantly.

4.8 Running Shortcuts "from Inside" NodeSpark

Now flip it around: NodeSpark → Shortcuts.

This is useful when NodeSpark:

- Prepares some text using AI.
- Then wants iOS to do something native:
    - Add Reminder/Calendar event
    - Open apps
    - Send a message
    - Append to a very specific Note
    - Run a more complex Shortcuts chain

4.8.1 Create the Shortcut you want NodeSpark to run

In Shortcuts:

1. Create a new Shortcut, for example: **"Create Tasks from Text"**.
2. Add actions such as:
    - **"Split Text"** (to break incoming text by line or separators)
    - **"Add New Reminder"** or **"Create Event"** for each line.
3. Make sure the Shortcut:
    - Accepts **Text** as input (check its **Details**).

- Is visible to other apps (Shortcuts usually handles this automatically).

This Shortcut will receive a big chunk of text from NodeSpark—like a list of tasks or a summary—and do something native with it.

4.8.2 Add a "Run Shortcut" node in NodeSpark

In your NodeSpark workflow:

1. Open the workflow.
2. Tap **Add Node → Action → Run Shortcut** (or similar label in your build).
3. Tap the node to edit its settings.
4. Configure:
    - **Action Mode**: Run Shortcut
    - **Shortcut Name**: type **exactly** the name of your Shortcut (e.g. Create Tasks from Text).
    - **Payload Strategy**:
        - Most flows will send the current **payload** (the AI result) as the input text to the Shortcut.

Connect it so the AI result flows into the Shortcut node:

Trigger → AI Node (creates tasks text) → Run Shortcut Action

When the workflow runs:

- NodeSpark builds a text payload (e.g. "1. Call supplier tomorrow… 2. Draft proposal…").
- The **Run Shortcut** node passes that payload into your Shortcut.
- The Shortcut converts it into Reminders, Calendar events, etc.

4.9 Example Automations

Here are some concrete flows you can set up and show off.

4.9.1 “Plan My Day” Siri Command

Goal: Say “Plan my day” and get a structured plan created by AI.

In NodeSpark:

1. Create a workflow named **“Plan My Day”**.
2. Nodes:
    - **Trigger**: Manual (Shortcut will trigger it).
    - **AI Node – Planner**:
        - Instruction:

          “You are a planning assistant. The input is a brain dump for today. Turn it into an ordered schedule from morning to evening, with times, tasks, and 1–2 priorities.”

    - **(Optional)**: HTTP node to send the schedule to Notion / email / Telegram.
3. Make sure the AI node uses the **incoming payload** as the brain dump.

In Shortcuts:

1. New Shortcut → **Ask for Input**:
    - Prompt: What’s on your plate today?
2. NodeSpark action: **Run NodeSpark Workflow**:
    - Workflow: Plan My Day
    - Input: **Provided Input** from Ask for Input.
3. Add **Show Result** action to display the plan.

Add to Siri:

- Phrase: **“Plan my day”**

Now the flow is:

"Hey Siri, Plan my day" → you talk → NodeSpark uses AI to build a schedule → Shortcuts shows you your plan.

4.9.2 "Log Idea" Voice Inbox

Goal: Capture ideas hands-free, run them through AI, and log them somewhere via n8n or Telegram.

In NodeSpark:

1. Workflow name: **"Idea Inbox"**.
2. Nodes:
    - Trigger: Manual.
    - AI Node:
        - Instruction:

          "You are an idea formatter. Take the raw idea text and return a concise description, one-sentence summary, and 2–3 tags."

    - HTTP Action:
        - Sends that structured text to n8n, Notion, or Telegram (see Section 3).

In Shortcuts:

1. New Shortcut → **Ask for Input**:
    - Prompt: What's your idea?
2. NodeSpark action: Run Idea Inbox, passing the input text.
3. Optional: **Show Result** to see how NodeSpark cleaned the idea.

Add to Siri:

- Phrase: **"Log idea"**

Now it's:

"Hey Siri, Log idea…" → you say it → NodeSpark cleans it + logs it with your tools.

4.9.3 NodeSpark → Shortcut → Reminders

Goal: AI breaks your input into tasks, then Shortcut adds each one as a Reminder.

In Shortcuts (first):

1. Create a Shortcut called **"Create Reminders from Text Block"**.
2. Actions:
    - **Split Text**:
        - Input: the Shortcut's input.
        - Separator: newline (\n) or • depending on your style.
    - **Repeat with Each**:
        - Inside loop: **Add New Reminder** using the repeat item as the title.

In NodeSpark:

1. Workflow name: **"Task Extractor"**.
2. Nodes:
    - Trigger: Manual or Schedule or Webhook.
    - AI Node:
        - Instruction:

          "From the input text, extract a bullet list of actionable tasks, one per line."

    - Action Node – Run Shortcut:
        - Action mode: Run Shortcut
        - Shortcut Name: Create Reminders from Text Block

Now when this workflow runs, NodeSpark:

1. Takes your chaotic text.

2. AI converts it to a clean list of line-separated tasks.
3. NodeSpark runs the Shortcut.
4. Shortcut turns each line into a Reminder.

You can trigger this from:

- Quick Run inside NodeSpark
- A Siri Shortcut that sends text into Task Extractor
- A webhook or n8n call that forward notes into NodeSpark

4.10 Troubleshooting Shortcuts & Siri

If something doesn't show up or fails:

4.10.1 NodeSpark actions not appearing in Shortcuts

- Open NodeSpark once.
- Force-quit **Shortcuts** and open it again.
- Make sure you're not on an old iOS version that's blocking extensions.
- Check that the app is fully installed (not offloaded).

4.10.2 Workflow list is empty in the NodeSpark action

- Make sure you've created at least **one workflow** in NodeSpark.
- Open NodeSpark, confirm workflows exist.
- Re-open Shortcuts and tap the Workflow parameter again.

4.10.3 Siri phrase not triggering the right shortcut

- In Shortcuts, open the Shortcut and:
  - Update or re-record the Siri phrase under **Add to Siri**.
- If Siri says it doesn't know that phrase:
  - Make the phrase more unique.

- Avoid phrases that conflict with built-in system commands.

4.10.4 Shortcut runs but NodeSpark doesn't seem to do anything

- Double-check **which workflow** you selected in the NodeSpark action.
- Use **Run History** or **Quick Run** inside NodeSpark to confirm the workflow works on its own.
- If the workflow triggers APIs or Shortcuts, check:
  - API logs (n8n, Telegram, etc.).
  - The receiving Shortcut (if you're running one from inside NodeSpark).

That's **Section 4 – Shortcuts & Siri**.

Section 5 – Advanced Logic, Branching & Conditions

So far you've seen NodeSpark as:

- "Take some input → Run AI → Send it somewhere."

Now we're going to make it **smart**:

- Route based on **content** ("If the message mentions 'invoice', do X; if it mentions 'meeting', do Y")
- Route based on **variables** (flags, toggles, numbers, etc.)
- Route based on **AI decisions** ("Ask the AI what kind of request this is, then branch accordingly.")
- Combine multiple conditions with AND/OR logic.

This section covers:

1. The basic **Logic node types**
2. How to **build simple IF conditions**
3. How to create **multi-branch routers**
4. Using **workflow variables** in conditions
5. Using **AI to drive routing**
6. Full example flows you can copy

Note: the exact names of fields may differ slightly in your build, but I'll describe them in the way they show up conceptually in your app.

5.1 Logic Nodes: What They're For

When you tap **Add Node → Logic**, you'll see options like:

- **IF / Condition Node**
- **Router / Switch Node**
- (Depending on your build, you may also have things like "Filter", "Wait Until", etc.)

In general:

- **Condition / IF** = yes/no decision
  - One input, two possible paths: **True** or **False**
- **Router / Switch** = one-to-many decision
  - One input, many possible labeled paths (e.g. "Billing", "Support", "Other")

You drop these into your canvas like any other node and connect them with edges to other nodes.

5.2 Basic IF Condition: "If Text Contains…"

Let's start with the most common use case:

"If the payload contains certain words, go down path A, otherwise path B."

5.2.1 Add the IF node

1. Open your workflow in **Workflow Editor**.
2. Tap the **"+"** to add a node.
3. Choose **Logic → IF / Condition** (whatever your logic node is labeled as).
4. Tap the new IF node to open its settings.

You'll see fields similar to:

- **Condition Type**
- **Left-hand side / Source**
- **Operator**
- **Right-hand side / Value**

- Optional: description / notes

5.2.2 Configure “Text Contains” condition

Let’s say we want:

If the input text mentions “invoice”, treat it as billing.

In the IF node:

1. **Condition Type**: select something like Text / Payload
2. **Source**:
    - Choose **“Current Payload Text”** or similar.
3. **Operator**: contains (case-insensitive)
4. **Value**:
    - Type: invoice

So conceptually:

IF **payload_text** CONTAINS “invoice” → TRUE path
ELSE → FALSE path

5.2.3 Wire the branches

1. On the canvas, your IF node will show two output ports:
    - True (or Yes)
    - False (or No)
2. Connect:
    - True → a node that handles invoices (e.g. an AI node “Draft invoice reply” or a webhook to your billing system).
    - False → whatever should happen for non-invoice messages.

Run the workflow:

- If the text mentions “invoice”, the engine takes the **True** line.
- If not, the engine takes the **False** line.

5.3 Multi-Branch Routing with a Router / Switch Node

If you’re doing “If invoice, do X; if meeting, do Y; if support, do Z”, an IF chain gets messy fast. That’s where a **Router / Switch** node makes more sense.

5.3.1 Add the Router node

1. Tap **Add Node → Logic → Router / Switch**.
2. Tap the new node to configure it.

You’ll see something like:

- **Routing Source** (the thing we’re inspecting; usually payload or a variable)
- One or more **Cases**
- Optional **Default / Else** branch

5.3.2 Example: “Classify request by keywords”

We want:

- If text contains “invoice” → Billing path
- If text contains “meeting” or “call” → Scheduling path
- If text contains “bug” or “error” → Support path
- Else → General path

In the Router node:

1. **Routing Source**: Payload Text
2. Add Case 1:
    - **Name**: Billing
    - **Match Type**: contains any of
    - **Values**: invoice, billing, payment
3. Add Case 2:
    - **Name**: Scheduling
    - **Match Type**: contains any of
    - **Values**: meeting, call, schedule
4. Add Case 3:
    - **Name**: Support
    - **Match Type**: contains any of
    - **Values**: bug, error, issue, crash
5. **Default / Else**:
    - Name it General

On the canvas, you'll see multiple outputs from the Router:

- Billing
- Scheduling
- Support
- Default (or Else)

Connect each to a different branch of the workflow:

- Billing → AI node that drafts a response for invoices, plus optional webhook to accounting.
- Scheduling → AI node that suggests times / draft a scheduling reply.
- Support → AI node that triages bug reports, maybe POSTs to your ticket server.
- General → catch-all AI node or simple reply.

5.4 Using Variables in Conditions

NodeSpark already supports **workflow variables**. Logic nodes can use those as inputs.

This is powerful for:

- Feature flags (“If enable_debug_log is true, send extra info”)
- User preferences (“If daily_digest_enabled is true, send them a summary”)
- Counts / numeric thresholds (“If count > 10, trigger escalation”)

5.4.1 Example: “Send notification only when a flag is true”

Say you have a variable:

- **Name**: notify_user
- **Type**: Boolean / String
    - Values: "true" or "false" (depending on how you store it)

Add an IF node:

1. **Condition Type**: Variable
2. **Variable Name**: notify_user
3. **Operator**:
    - If boolean: is true
    - If string: equals (case-insensitive)
4. **Value** (if string-based): true

Wire it:

- True branch → Notification Action node (push, Telegram, etc.)
- False branch → either ends the workflow or goes to a different action.

Now: if another node earlier in the workflow sets notify_user to false, the alert never fires.

5.5 Combining Multiple Conditions (AND / OR Logic)

Sometimes you want something like:

If the payload contains “invoice” **AND** the user is premium → run advanced flow.
Or
If the text contains “bug” **OR** “error” **OR** “crash” → route to Support.

Depending on how your logic UI is wired, you’ll do this in one of two ways:

5.5.1 Single IF node with composite condition

Some versions of the logic node support multiple rows and a **Mode**:

- Condition Row 1: payload_text contains "invoice"
- Condition Row 2: variable customer_tier equals "premium"

Then choose:

- **Mode**: ALL (AND)
    - Both must be true

or

- **Mode**: ANY (OR)
    - At least one must be true

5.5.2 Chaining multiple IF nodes

If your build uses simple one-condition IFs, just chain them:

- First IF: payload_text contains "invoice"
    - True → Next IF: customer_tier == "premium"
    - False → route to “non-invoice” flow
- Second IF: customer_tier == "premium"
    - True → advanced invoice flow

- False → basic invoice flow

5.6 AI-Driven Routing (Let the Model Decide the Path)

This is where NodeSpark really gets fancy.

Instead of writing a ton of keyword rules, you can:

1. Use an **AI Node** to classify the request or produce a label.
2. Store that label in a variable.
3. Use a **Router** or IF node on that variable.

5.6.1 Example: AI as a router front-end

Goal: Let the AI decide if a message is **Billing**, **Scheduling**, **Support**, or **Other**.

Step 1 – AI Node: Classifier

- Instruction:

  "You will receive a user message. Your job is to classify it into exactly one of these categories: billing, scheduling, support, or other. Respond with only the category name, no explanation."

- Input: the full payload text.

Step 2 – Write result to variable

- Configure the AI node so that its **output** (the single word) is stored as:
  - Variable: category

Step 3 – Router by variable

Add a **Router** node:

1. **Routing Source**: Variable category
2. Cases:
    - billing
    - scheduling
    - support
    - other (or use Default branch)

Wire each case to different follow-up nodes (just like earlier).

Now your flow looks like:

Trigger → AI Classifier → Router by category → Branches

No brittle keyword lists. The agent handles weird language for you.

5.7 Time-Based Logic

You can also use logic based on **time** and **dates** if your nodes expose that:

Examples:

- "If current time is after 5pm, send an after-hours response."
- "If it's the weekend, queue the message instead of sending immediately."

Depending on how your engine exposes time, you might have:

- A Trigger with schedule params
- A variable like current_hour, day_of_week, set by a helper node

Example condition:

- **Source**: current_hour
- **Operator**: is greater than
- **Value**: 17

True → after-hours branch; False → normal-hours branch.

5.8 Scenario Walkthroughs

Here are some real-world flows using all of this in combination.

5.8.1 Scenario 1 – Smart Email Router with AI + conditions

Goal: A workflow that:

- Takes an email body as input.
- Uses AI to classify it (billing, scheduling, support).
- Uses a Router to branch.
- Sends different types of replies / webhooks for each type.

Nodes:

1. **Trigger**: Manual / Webhook (email text comes in as payload).
2. **AI Node – Classify request**:
    - Instruction:

"Classify this email as exactly one of: billing, scheduling, support, other. Respond with only the category."

- Output stored in variable: category

3. **Router Node – Route by category**:
   - Source: category
   - Cases:
     - billing → path A
     - scheduling → path B
     - support → path C
     - Default → path D
4. **Path A – Billing**:
   - AI Node: draft billing-style reply.
   - HTTP Node: send summary to n8n / CRM.
5. **Path B – Scheduling**:
   - AI Node: propose times.
   - Optional: Action node "Run Shortcut" to open a Calendar Shortcut.
6. **Path C – Support**:
   - AI Node: triage bug, extract steps to reproduce.
   - HTTP Node: POST to your ticket system.
7. **Path D – Other**:
   - AI Node: general polite reply.

Run it with different email payloads. The Router will send each one down the right branch, driven by the AI output.

5.8.2 Scenario 2 – Only escalate if AI says "urgent" AND count is above threshold

Goal: Don't spam alerts, but if AI says a message is **urgent** and you've seen too many in a short time, escalate.

Nodes:

1. **Trigger**: Webhook from n8n or your system logs.
2. **AI Node – Urgency check**:
   - Instruction:

"You are an incident classifier. Decide if this incident is urgent or normal. Respond with only urgent or normal."

   - Output → variable urgency
3. **Helper Node – Count incidents**:
   - Some node that increments a recent_incident_count variable.
4. **IF Node – Urgent AND threshold**:
   - Condition 1: urgency == "urgent"
   - Condition 2: recent_incident_count > 5
   - Mode: ALL (AND)
5. **True branch**:
   - HTTP Node: hit your alert API.
   - Action Node: send Telegram, SMS, or Slack.
6. **False branch**:
   - Normal logging, no escalation.

This mixes AI routing + numeric logic + variables.

5.8.3 Scenario 3 – "Weekend mode" scheduler

Goal: During weekends, queue up actions instead of doing them. During weekdays, go full-speed.

Nodes:

1. **Trigger**: Schedule or app event.
2. **Helper Node – Set day_of_week**:
   - Produces a variable like day_of_week = "Mon", "Tue", etc.
3. **IF Node – Is weekend?**
   - Condition: day_of_week in ["Sat", "Sun"]

     (You can do this with a Router or string comparison.)

4. **True (weekend)**:
   - AI Node: create summary of what would've been done.
   - HTTP Node: send single digest to Telegram / email.
5. **False (weekday)**:
   - Regular workflow path: run full automation.

5.9 Debugging Logic & Branching

Logic bugs can be subtle. Here's how to sanity check:

5.9.1 Use Quick Run + Scratchpad + Logs

- Run your workflow from:
  - The **Quick Run** sheet
  - A trigger (schedule, webhook), then inspect **Run History**

Look for:

- Logs from each node ("condition evaluated to false", "category = billing").
- Scratchpad content from AI nodes (if you surfaced it).

5.9.2 Common gotchas

- **Whitespace / casing**:
  - Make sure you trim and/or lowercase inputs in conditions when possible.
- **Spelling between AI and Router**:
  - If the AI sometimes returns Billing vs billing, normalize values (lowercase them) or use case-insensitive matching.
- **Variables not set**:
  - If a variable doesn't exist yet, conditions can fail.
    - Add a default / fallback, or ensure some node always sets it before logic.

That's **Section 5 – Advanced Logic, Branching & Conditions**.

Section 6 – Templates, Quick Run & the "Automation Store"

NodeSpark isn't just a blank canvas where you have to build everything from scratch.

It also comes with a **Template Gallery** and a **Quick Run console** that make the app feel like an **automation store**:

- You **browse templates** like apps.
- Tap one → it installs as a workflow.
- Use **Quick Run** to test, tweak, and learn how it works.

This section covers:

1. Template Gallery overview
2. How to install a template as a workflow
3. Understanding template badges (Beginner / Shortcuts / API Key)
4. Editing and customizing a template
5. Quick Run: testing a workflow like a mini lab
6. Using Quick Run with templates (step-by-step example)
7. Recommended templates for different use cases

6.1 Template Gallery Overview

You can think of the Template Gallery as **NodeSpark's built-in "Automation Store"**.

How to open the Template Gallery

1. From the main **Workflow List** screen (the "Control room"):
    - At the top under the NodeSpark title, tap the **"Templates"** button (grid icon).
2. The **Template Gallery** sheet slides up.

You'll see:

- Templates grouped into **categories** (horizontal carousels):
  - **AI**
  - **Messaging**
  - **Webhooks**
  - **Productivity**
- Each template is shown as a **large, glossy card** with:
  - Title (what it's called)
  - Category pill (AI / Messaging / etc.)
  - Short **"What this does"** description
  - Example-style explanation built into the text
  - An action row: **"Use this template"**

The idea: you **pick a template** that looks like what you want, then tweak it instead of designing from zero.

6.2 Installing a Template as a Workflow

When you see a template you like:

1. Open **Template Gallery**.
2. Swipe horizontally in the category row (AI, Messaging, etc.) to browse.
3. Tap anywhere on a card to read the full description.
4. Tap **"Use this template"**.

What happens next:

- NodeSpark **creates a new workflow** based on that template:
  - The workflow appears in your **Workflow List**.
  - It comes with:
    - Pre-built nodes (Trigger, AI, Logic, HTTP, Shortcuts, etc.)
    - Example prompts
    - Example webhook / API structure (if applicable)
- You can tap that workflow in the list to open the **canvas** and see how it's wired.

Free vs Paid: Template install limits

- On the **Free plan**:
  - You can have **up to 2 workflows** (whether created from scratch or from templates).
  - If you try to install a template after you already have 2 workflows:
    - NodeSpark will show the **Upgrade** (paywall) screen.
- On **Paid (Unlimited)**:
  - You can install as many templates as you like.
  - Great for building a personal library of automations.

6.3 Template Badges & Requirements

Each template is more than a pretty card—it tells you **how advanced it is** and what it needs.

On each card you may see labels like:

- **Beginner-friendly**
  - Minimal setup.
  - Mostly uses AI + simple triggers.
  - Great for first-time users.
- **Requires Shortcuts**
  - This workflow includes nodes that **call Apple Shortcuts**.
  - You'll need:
    - The Shortcuts app installed (standard on iOS).
    - At least one Shortcut configured (e.g. "Add to Reminders", "Open Note", "Run Calendar Action").
  - The template's description will tell you **which Shortcut** to point it to.
- **Requires API key**
  - This workflow calls external APIs such as:
    - Webhook endpoints (n8n, Make, custom server)
    - Email services, Telegram bots, Notion API, etc.
  - You'll need to:
    - Go into the **AI Settings / API Settings** in NodeSpark.
    - Paste in the relevant API key or endpoint URL.
    - Update any placeholder URLs inside HTTP nodes.

There are **no "Pro Only" template badges**—templates themselves are available, but your free plan is limited to 2 total workflows. Upgrading just unlocks how many workflows you can keep, not special templates.

6.4 Customizing a Template After Install

Once you tap **"Use this template"**, it becomes **your workflow**. From there:

1. Go back to the **Workflow List**.
2. Tap the new workflow card to open the **canvas**.
3. You can now:
    - Rename the workflow (title & description).
    - Move nodes around.
    - Change AI prompts.
    - Change logic conditions.
    - Add/remove nodes as needed.

A good starter flow when customizing:

1. **Rename the workflow**
    - Tap the workflow name at the top of the editor.
    - Give it something meaningful:
        - "Daily AI Standup Digest"
        - "Telegram Inbox Triage"
        - "Notion Summary Bot"
2. **Review the Trigger node**
    - Is it manual? webhook? schedule? app event?
    - If it's schedule-based, update:
        - Time of day
        - Frequency (daily/weekly, etc.)
3. **Review AI nodes**
    - Open each AI node and check:
        - The **system / instruction prompt** (what the bot is asked to do).
        - Whether it uses **agent mode** (tools) or plain text.
    - Tweak the tone and behavior to match how *you* work.
4. **Update action nodes**
    - For HTTP nodes:
        - Set real URLs instead of sample ones.
        - Map payload fields to what your service expects.
    - For Shortcuts:
        - Select the actual Shortcut you want NodeSpark to call.

    - For messaging:
        - Confirm destination (Telegram chat ID, email, etc.).

Once you've adjusted the template, it's essentially **your custom automation**, built much faster than starting from a blank canvas.

6.5 Quick Run: Your Instant Test Console

Quick Run turns any workflow into a **mini test lab**:

- You give it input (text or JSON).
- NodeSpark runs the full workflow.
- You see:
    - Output
    - Run log (what happened step by step)
    - Agent Scratchpad (internal AI notes, tools used, decisions made)

How to open Quick Run

From the **Workflow List**:

1. **Long-press** a workflow card.
2. A context menu appears.
3. Tap **"Quick Run"**.

This opens the **Quick Run sheet** for that workflow.

What you see in Quick Run

The Quick Run screen is split into sections:

1. **Header**
    - Title: “Quick Run”
    - Shows which workflow you’re testing.
2. **Input section**
    - A large text area labeled **Input**.
    - **Preset chips** like:
        - “Summarize email”
        - “Extract tasks”
        - “JSON payload”
    - You can:
        - Tap a preset → auto-fill sample input.
        - Edit the text freely before running.
        - Tap **Clear** to wipe the input (red button).
3. **Run button**
    - Big button: **Run Workflow**
    - While running:
        - The button shows a spinner with “Running…”
        - It’s disabled until the run finishes.
4. **Output**
    - Large box labeled **Output**.
    - Shows:
        - Either “Run the workflow to see the result here.”
        - Or the final text result from your workflow.
    - Text is selectable so you can copy.
5. **Run Log**
    - Shows a timeline of messages like:
        - “Trigger node started…”
        - “AI node executed with 1024 tokens…”
        - “HTTP request returned 200…”
    - Let’s you verify what actually happened under the hood.
6. **Agent Scratchpad**
    - When using **Agent mode AI nodes**, you may see:
        - “Agent Scratchpad” box.
    - Contains:
        - The model’s internal planning notes.
        - What tools it tried.
        - Why it chose certain steps.
    - Perfect for debugging and explaining results.

Quick Run gives you confidence that a workflow is working **before** you hook it to schedules, webhooks, or notifications.

6.6 Using Quick Run with Templates – Example Walkthrough

Let's do a concrete example:

Template: **"Daily Standup / Task Summary Agent"**
(AI + Agent mode + Quick Run)

Step 1 – Install the template

1. Open **Template Gallery**.
2. Find the **AI** or **Productivity** category row.
3. Locate a template along the lines of:
    - "Daily Standup Summary"
    - "AI Task Digest"
4. Tap **"Use this template"**.
5. It appears in your **Workflow List**.

Step 2 – Open Quick Run for that template

1. In **Workflow List**, long-press the new workflow.
2. Tap **Quick Run**.

Step 3 – Provide input

1. In the **Input** section:
    - Either:
        - Tap the **"Extract tasks"** preset, or
        - Paste in your own messy notes / standup text, like:

          "Yesterday I fixed two bugs in the login page, today I need to review PR #42 and draft the new feature spec. Blocked on design sign-off for the dashboard."

2. Edit anything you want.

Step 4 – Run the workflow

1. Tap **Run Workflow**.
2. Watch while it runs:
    - The button shows loading.
    - Run Log fills in messages as nodes run.

Step 5 – Review the result

- **Output** section:
    - You might see:
        - A clean bullet-point task list
        - Or a formatted standup summary: "Yesterday / Today / Blockers"
- **Run Log**:
    - Confirms which nodes fired:
        - AI node: classification or summarization
        - Logic nodes: did it route correctly?
        - Any notification or webhook nodes
- **Agent Scratchpad** (if agent mode enabled):
    - Shows how the AI broke down your text:
        - Which tasks it detected
        - Any tools it considered using
        - Notes to itself about how to respond

If something's off:

- Edit the **AI prompt** in the AI node (via the main editor view).
- Re-open **Quick Run**, paste the same input, and run again.

This loop—**Template → Tweak → Quick Run → Repeat**—is how you quickly design real automations without going insane.

6.7 Recommended Templates for Different Users

To help you get traction fast, here's how to use the Automation Store side of NodeSpark depending on what you want.

1) "I just want to see it do something cool with AI"

- Use: **AI → Beginner-friendly templates**
  - Example types:
    - "Summarize text / notes"
    - "Rewrite / improve writing"
    - "Daily digest from copied text"
- Flow:

1. Install the template.
2. Use Quick Run.
3. Paste emails, notes, or web articles and watch it summarize.

2) "I want a simple bot that handles my messages"

- Use: **Messaging templates**
  - Example types:
    - "Inbox triage with AI"
    - "AI reply drafts for messages"
- Flow:

1. Install the template.
2. Configure any required Shortcuts / Telegram or API endpoints.
3. Use Quick Run with sample messages.
4. When happy, connect it to a **webhook** or **schedule**.

3) "I use n8n / Make / a custom server"

- Use: **Webhooks templates** with "Requires API key" badges.
  - Example types:
    - "Webhook → AI → JSON reply"
    - "Server event → AI Router → HTTP actions"
- Flow:

1. Install the template.
2. Update the **HTTP node URLs** to your n8n or server endpoints.
3. Copy sample JSON bodies from the template description or Integration Center.

4. Trigger via your external system and observe the result in NodeSpark's Run History.

4) "I want a personal assistant agent that uses tools"

- Use: **AI Agent mode templates**.
  - Example types:
    - "Research Agent"
    - "Notification Agent"
    - "Routing Agent" or "AI Router Bot"
- Flow:

1. Install the template and open it in the canvas.
2. Confirm the **AI node** is set to **Agent mode** and has tools enabled (subflows, HTTP, Shortcuts, etc.).
3. Use Quick Run with real-world prompts:
   - "Give me a summary of today's tasks, then ping me if anything is urgent."
   - "Figure out whether this belongs in Notion, Calendar, or as a notification."
4. Watch the **Agent Scratchpad** to see:
   - How it decides what to do
   - Which tools it uses

That's **Section 6 – Templates, Quick Run & the Automation Store**.

Section 7 – Run History, Debugging & Troubleshooting

Automations are great… *when they work.*

NodeSpark gives you tools to see **what actually happened** every time a workflow runs:

- **Run History** – a timeline of every execution of a workflow
- **Run Details** – the final payload, error status, and internal logs
- **Agent Scratchpad** – "how the AI thought" during the run
- **Quick Run Logs** – live test runs with full visibility

This section covers:

1. Where runs come from (ways a workflow can execute)
2. Viewing Run History for a workflow
3. Understanding the Run History entries
4. Opening a single run to debug it
5. Using Agent Scratchpad to understand AI behavior
6. Debugging common problems (HTTP errors, Shortcuts, webhooks, etc.)
7. Quick troubleshooting checklist

7.1 How a Workflow Can Run

First, it's helpful to know **what counts as a "run"** in NodeSpark.

A workflow run is recorded when:

- You **tap the workflow** and run it from inside the app (via trigger or canvas)
- You use **Quick Run** from the long-press menu
- A **Schedule Trigger** fires (e.g. "Every day at 8AM")
- An **App Event Trigger** fires:
    - App launch
    - App foreground

    - Notification tapped, etc.
- A **Webhook / HTTP trigger** calls into NodeSpark (via your external tools like n8n/Make/custom server)
- A **Notification tap** launches the workflow (if configured)

Every execution gets:

- A **timestamp**
- A **final payload preview** (output)
- A **success / error flag**
- Optional:
    - Log lines
    - Agent scratchpad notes (if AI Agent mode is used)

All of that is what you see in **Run History**.

7.2 How to View Run History for a Workflow

You can open Run History right from the **Workflow List**.

Method 1 – Long-press the workflow

1. Go to the **Workflow List** ("Control room").
2. **Long-press** on a workflow card.
3. A context menu appears with options like:
    - Quick Run
    - Integrations
    - **Run History**
    - Duplicate Workflow
4. Tap **Run History**.

This opens a **Run History screen** just for that workflow.

Tip: If a workflow card shows:

- "Last run 2 hours ago – check logs"

  …that means the last run had an error, and Run History is where you'll see why.

7.3 Understanding the Run History Screen

The Run History screen is effectively a **logbook** for one workflow.

You'll typically see:

- A **list of runs**, most recent at the top
- Each row shows:
  - **Date / time** of the run
  - **Status**:
    - ✅ Success (no error)
    - ⚠️ Error (something went wrong)
  - **Preview**:
    - A short clip of the final payload text

      (e.g., "Summary: You need to finish 3 tasks…")

- Optional:
  - A label or icon if this run came from **Quick Run** vs. schedule vs. webhook (depending on your build)

Think of it like "call history", but for automations.

How many runs are stored?

- NodeSpark keeps a **recent history per workflow**.
- Older entries may be trimmed over time to keep things fast and small.
- For day-to-day debugging and audit, you'll almost always have the last several runs available.

7.4 Opening a Single Run to Debug It

From the **Run History** list:

1. Tap on a run entry you want to inspect.
2. You'll see a **Run Detail** view (exact layout may vary, but conceptually it includes):
    - **Timestamp & status**
        - When it ran
        - Whether it succeeded or errored
    - **Final payload**
        - The output that was passed to the last node
        - This is what gets used for notifications, HTTP bodies, clipboard, etc.
    - **Run Log**
        - A list of steps with messages like:
            - "Trigger: Schedule at 08:00 fired."
            - "AI node started – prompt: Daily summary of tasks."
            - "HTTP request to https://example.com/api returned 200."
            - "Tool X failed with 500; agent attempted to retry once."
        - This is your **"flight recorder"** of what happened.
    - **Agent Scratchpad** *(for AI Agent runs)*
        - Internal reasoning from the AI
        - Notes about tools it tried
        - Why it made certain decisions

This view is where you answer: **"What actually happened?"**

7.5 Agent Scratchpad – Seeing How the AI Thought

If your workflow uses an **AI Agent node** (tool-enabled mode), NodeSpark can capture an **agent scratchpad** during the run:

- This is **not** normally shown to your end users.
- It's a **debug feature** for you, the workflow designer.

You'll see things like:

- "User asked for a summary of tasks → decide to call 'GetTasks' tool."
- "Got HTTP 500 from API → decide to surface an error to user."
- "Intent classification: 'schedule_event' → route to calendar subflow."

Why this is useful:

- Helps you understand **why** the AI chose a particular path.
- Shows which **tools it actually used**.
- Let's you tune prompts:
    - If it picked the wrong route, adjust your instructions.
    - If it called a tool too often, clarify when it should use that tool.

How to use it for debugging

1. Find a run that **behaved strangely** (wrong output, wrong route).
2. Open the run's detail view.
3. Scroll down to **Agent Scratchpad** (if present).
4. Read through its thought process:
    - Did it misunderstand the user's intent?
    - Did it call the wrong tool?
    - Did it bail out due to an HTTP error?

Then:

- Go back to the **AI node** in your workflow.
- Update:
    - System prompt / instructions.
    - Tool descriptions.
    - Routing logic hints.
- Re-test via **Quick Run**.

7.6 Debugging Common Issues

Here are some typical problems and how to debug them using Run History and Quick Run.

A) The workflow shows “Last run – check logs” and nothing happened

Symptoms:

- Workflow card shows:
  - “Last run 5 minutes ago – check logs”
- You didn’t get the result you expected (no notification, no message, etc.).

Steps to debug:

1. **Open Run History** for that workflow.
2. Tap the most recent run marked with an **error / warning icon**.
3. Check the **Run Log**:
   - Did it stop at the AI node?
   - Did an HTTP request fail?
   - Did a Shortcut fail to run?
4. Common log messages to look for:
   - “Tool X failed with 500; here’s what the agent tried to do.”
   - “HTTP request to [URL] responded with status 401 (Unauthorized).”
   - “Shortcut named ‘XYZ’ not found.”
   - “Schedule triggered, but initial payload was empty.”
5. Fix the root cause:
   - For HTTP 401 / 403:
     - Update your API key, token, or credentials.
   - For HTTP 404:
     - Check the URL path.
   - For HTTP 500:
     - The other service is failing; try again later or simplify payload.
   - For missing Shortcuts:
     - Open the Shortcuts app and confirm the Shortcut exists with that name.
   - For empty payload:
     - Update the trigger or Quick Run input so the workflow receives actual data.
6. Re-test with **Quick Run**:
   - Long-press the workflow → Quick Run.
   - Use test input similar to the problematic run.
   - Confirm it now succeeds.

B) Webhook / n8n / Server Trigger Isn't Firing

Symptoms:

- Your external tool (n8n, Make, server) claims it sent a request.
- NodeSpark doesn't seem to do anything.

Steps:

1. Confirm the **Trigger node** is set correctly:
    - Open the workflow in the editor.
    - Check the Trigger node:
        - Should be set to **"On webhook / external call"**.
    - Ensure:
        - It's not disabled.
        - It has the right description / configuration text for your use case.
2. Check if **any run appeared**:
    - Open **Run History** for that workflow.
    - If no run shows:
        - NodeSpark never saw the trigger.
        - Re-check your integration:
            - n8n/Make scenario matches the correct endpoint.
            - Device is online.
            - Your external system is actually sending requests.
3. Use **Quick Run** to validate workflow logic:
    - If Run History is empty, your external trigger might be miswired, but the workflow itself could still be fine.
    - Long-press → Quick Run.
    - Paste sample payload that mimics what your server sends.
    - Run it and observe logs.
4. When Quick Run works but webhook doesn't:
    - Re-check your external system configuration:
        - URL, method (POST vs GET), headers (Content-Type: application/json), body structure.

C) HTTP Node Returning Errors

Symptoms:

- Run Log shows HTTP errors (400, 401, 404, 500, etc.).
- Output is empty or error-y.

Steps:

1. Open **Run History** → tap the specific run.
2. Look at the **Run Log**:
    - Find the line where the HTTP node is logged.
    - Look for:
        - URL
        - Status code
        - Any error message.
3. Typical issues & fixes:
    - **401 / 403 (Unauthorized/Forbidden)**
        - API key is missing or wrong.
        - Fix:
            - Check the HTTP node headers.
            - Make sure the Authorization header or token is correct.
            - Update API keys in the NodeSpark settings or node parameters.
    - **404 (Not Found)**
        - Endpoint URL is wrong.
        - Fix:
            - Copy-paste the URL into a browser / API test tool.
            - Compare the path to the API docs.
    - **400 (Bad Request)**
        - Payload shape doesn't match what the service expects.
        - Fix:
            - Check the JSON body you're sending.
            - Compare field names and nesting with the API documentation.
    - **500 (Server Error)**
        - The server you're calling is having an internal error.
        - Fix:
            - Try again later.
            - Simplify the payload to see if specific fields trigger the issue.
            - If the Run Log mentions "Tool X failed with 500; here's what the agent tried to do", inspect that log to see what it sent.
4. Re-run using **Quick Run**:
    - Adjust input or configuration.
    - Run again and watch logs for status changes.

D) Shortcut Didn’t Run / Didn’t Do Anything

Symptoms:

- Workflow runs, but your expected Shortcut-based action never happens:
  - No Reminders created.
  - No Calendar event.
  - No Shortcut UI.

Steps:

1. Open the **editor** for the workflow.
2. Find the **Action node** configured to **Run Shortcut**.
3. Confirm:
   - The chosen Shortcut name matches exactly the Shortcut in the Shortcuts app.
   - Any input you intend to pass into the Shortcut is mapped correctly.
4. Open **Run History**:
   - Tap the failed/suspicious run.
   - Look for log messages like:
     - “Shortcut ‘XYZ’ not found.”
     - “Run Shortcut action skipped due to missing configuration.”
5. Fix:
   - Rename the Shortcut in the Shortcuts app or update the node to match.
   - Ensure the Shortcut exists on **this device**.
6. Test:
   - Use **Quick Run** with test input and verify:
     - You see evidence in your Shortcuts history or the created reminders/events.

E) Agent Makes the Wrong Decision (Bad Routing / Wrong Tool)

Symptoms:

- The AI sends the wrong message, uses the wrong tool, or calls a subflow when it shouldn’t.

Steps:

1. Find the problematic run in **Run History**.
2. Open the run and scroll down to **Agent Scratchpad**.
3. Read what the agent thought:
    - Did it misinterpret the user's intent?
    - Did it think the message was "high priority" when it wasn't?
    - Did it think it should call a webhook instead of Shortcuts?
4. Fix in the **AI node**:
    - Clarify instructions:
        - "Only call the Slack webhook if urgency is 'high'."
        - "If the text mentions 'calendar' or 'schedule', use the Calendar subflow."
    - Tighten tool descriptions:
        - Explain what each tool is for, in plain language.
5. Retest with **Quick Run**:
    - Use the same input text.
    - Check the scratchpad again to see if the reasoning improved.

7.7 Quick Troubleshooting Checklist

When something's not right, run through this list:

1. **Did the workflow run at all?**
    - Check the **Workflow card**:
        - "Last run …" text
    - Open **Run History**:
        - If there are **no runs**, your trigger (schedule, webhook, event) may not be firing.
2. **If it did run, was there an error?**
    - Look for warnings like:
        - "Last run – check logs"
    - In **Run History**, check if the latest entry is marked as error / warning.
3. **Open the latest run's details**
    - Look at:
        - Final payload preview
        - Run Log messages
        - Agent Scratchpad (for AI Agent flows)
4. **Check integration points**
    - HTTP nodes:
        - Any non-2xx status codes?
    - Shortcuts:
        - Does the Shortcut exist with that exact name?

    - Webhooks:
        - Is your external tool actually sending the request?
5. **Reproduce with Quick Run**
    - Long-press workflow → **Quick Run**
    - Paste the same or similar input.
    - Watch the output + logs live.
6. **Adjust and test in small steps**
    - Change one thing at a time:
        - AI prompt text
        - Endpoint URL
        - Header or payload key
    - Re-run and confirm improvement.

That's **Section 7 – Run History, Debugging & Troubleshooting**

Section 8 – Real-World Recipes & Example Automations

This section is all about **copy-and-paste ideas**.

You'll see complete, step-by-step recipes that show how to:

- Turn NodeSpark into a **daily AI assistant**
- Use **Shortcuts + Siri** to capture tasks and route them through AI
- Wire NodeSpark together with **n8n** and **Telegram** for superpowers
- Build reusable **AI writing helpers** with Prompt Profiles + Quick Run

You don't have to build these *exactly* as written—but they're great starting points.

8.1 Daily AI Standup – "What Do I Need to Do Today?"

Goal: Every morning at 8 AM, NodeSpark:

1. Takes a block of text (your scratchpad/tasks)
2. Uses AI to:
    - Summarize your day
    - Extract key tasks and priorities
3. Sends a **local notification** with a clean summary

You can start simple (paste in your own text), or later wire it up with Shortcuts/Reminders.

8.1.1 Build the Workflow

1. Open **NodeSpark** → you're on the **Workflow List**.
2. Tap **New Workflow**.
3. Name it something like:

   "Daily AI Standup"

4. Optional description:

   "Summarizes my notes and tasks into a clear daily plan."

Tap **Create Workflow**.

8.1.2 Add a Schedule Trigger

1. In the canvas, add a **Trigger** node.
2. Set:
   - **Trigger Type**: Schedule
   - Time: 8:00 AM (or your preferred time)
   - Frequency: Every day
3. In the Trigger description/notes field, write something like:
   - *"Runs every morning at 8 AM to summarize my day."*

Now, whenever the schedule fires, the workflow starts.

8.1.3 Add an AI Node to Summarize

1. Add an **AI node**.
2. Connect the **Trigger → AI node**.
3. In the AI node configuration:
   - **Mode**: Basic AI or Single call (not Agent mode yet, to keep it simple)
   - **Prompt** (system or main instruction), for example:

*"You are a concise planning assistant. Given the user's notes or tasks, produce:

1. A 2–3 sentence summary of the day
2. A bulleted list of top 5 priorities

3. A ‘Next Action’ line at the bottom.”*

4. Initial version of this recipe:
    - Let the AI treat whatever **text payload** it receives as “today’s notes.”
    - Later you can feed real data in via Shortcuts, Reminders, etc.

8.1.4 Add a Notification Node

1. Add an **Action node** configured to send a **local notification**.
2. Connect **AI node → Notification node**.
3. In the Action node parameters:
    - **Action Mode**: Notification
    - **Title**:

      “Your daily standup is ready”

    - **Body**:

      Use the payload from the AI node (the summary)

      (Depending on your UI, you may have a field like notification_body = {{payload}})

Result: Each morning, the AI summary becomes the notification body.

8.1.5 Test with Quick Run

Before relying on the schedule, test manually.

1. Go back to the **Workflow List**.
2. Long-press **“Daily AI Standup”**.
3. Tap **Quick Run**.
4. In the Quick Run input box, paste some sample text, e.g.:

   “Today I need to finish the landing page copy, review the Q4 budget spreadsheet, reply to 5 customer emails, and schedule a call with the design team for Thursday.”

5. Tap **Run Workflow**.

6. Check:
    - **Output** section – do you get a nice summary + priorities?
    - A **local notification** should appear with that same text.
7. If something looks off:
    - Adjust the AI node prompt.
    - Run again using Quick Run until you like the output.

Now your daily AI standup runs automatically every morning.

8.2 "Hey Siri, Capture This" → AI Task Extractor via Shortcuts

Goal: Use your voice or the Share Sheet to dump messy thoughts into NodeSpark, let AI extract clean tasks, and feed them back into Shortcuts to create reminders or todos.

High-level flow:

1. Use **Shortcuts** to capture text (dictation, clipboard, or share).
2. Shortcuts calls **NodeSpark** with that text via **Run Workflow**.
3. NodeSpark AI turns it into a **clean task list**.
4. Shortcuts reads NodeSpark's output and makes **Reminders**.

8.2.1 Create the Workflow in NodeSpark

1. Open **NodeSpark → New Workflow**.
2. Name: **"AI Task Extractor"**.
3. Description:

    "Takes messy notes and returns a clean task list for Shortcuts to process."

4. Tap **Create Workflow**.

8.2.2 Configure the AI Node

1. In the workflow editor, add an **AI node** (no Trigger node needed if you'll always call it from Shortcuts).
2. Configure the AI node:
    - **Mode**: Basic or Structured output if you have that option.
    - **Prompt**, something like:

    *"You are a task extraction assistant. The input is a messy note the user dictated or pasted.

    Return ONLY a numbered list of tasks, one per line.

    No extra commentary.

    Example:

    1. Email Sarah about Q4 report
    2. Book dentist appointment
    3. Upload receipts to accounting folder"*

3. Set it so the AI node **outputs the final payload** as that formatted list.

You can keep this workflow as just a single AI node for now.

8.2.3 Add a Simple "Pass-through" Action (Optional)

If your node system requires an Action node to "finish":

1. Add an **Action node** after the AI node.
2. Choose a simple action like:
    - Do nothing / Pass-through

      …or just keep the payload unchanged.

3. The important part: the **final payload** of the workflow remains the AI's numbered list.

8.2.4 Build the Shortcut: "Capture with NodeSpark"

On your iPhone:

1. Open the **Shortcuts** app.
2. Tap + to create a new shortcut.
3. Name it: **"Capture with NodeSpark"**.
4. Add actions:

Step 1 – Get the input text

- Use one of:
  - **Dictate Text**

    (speak your thoughts, convert to text)

  - **Get Clipboard**

    (if you copy text from Mail/Notes)

  - **Ask for Input**

    (type something manually)

Step 2 – Run NodeSpark workflow

- Tap **Add Action**
- Search: **NodeSpark**
- Choose: **Run Workflow** (NodeSpark action)
- Set:
  - Workflow: **"AI Task Extractor"**
  - Input: the text from Step 1

Step 3 – Parse the output

- After the NodeSpark action, add:
  - **Split Text** → separator: New Lines
  - This turns the numbered list into a list of lines.

Step 4 – Create reminders from each line

- Add **Repeat with Each**
  - Repeat over the list from Split Text.
- Inside the loop:
  - Add **Create Reminder**
    - Title: use the current repeat item
    - Optional: set list / due date.

Now:

- When you run the Shortcut, it:
  - Captures text
  - Sends it to NodeSpark
  - Gets cleaned tasks
  - Creates reminders for each

You can trigger this with **"Hey Siri, Capture with NodeSpark"** or add it to your Home Screen.

8.3 Telegram → n8n → NodeSpark – AI Inbox Triage

This one's more advanced, but powerful.

Goal:

Use Telegram as an inbox, **n8n** as the router, and **NodeSpark** as the AI brain.

Flow:

1. Messages arrive in a **Telegram bot**.
2. **n8n** picks up new messages from the bot.
3. n8n sends the text to a NodeSpark workflow (via HTTP/webhook or your chosen integration bridge).
4. NodeSpark AI:
   - Classifies the message (e.g., "todo", "idea", "question")
   - Generates a short summary + label
5. n8n receives the AI output and routes the message:
   - To Notion, Trello, email, etc.

Note: Technically, the iPhone must be able to receive these calls or you use a bridging approach (like n8n calling a companion that triggers NodeSpark). This recipe focuses on the **concept and node wiring** in NodeSpark and n8n.

8.3.1 Create the AI Triage Workflow in NodeSpark

1. In **NodeSpark**, create a new workflow:
    - Name: **"AI Inbox Triage"**
    - Description: *"Takes a text message and returns a JSON classification + summary."*
2. Add a **Trigger** node:
    - Trigger Type: On webhook / external call
    - Notes:

      "Called from n8n with { payload: 'incoming message text' }."

3. Add an **AI node**:
    - Connect **Trigger → AI node**.
    - Configure:
        - **Mode**: Agent or Structured if available.
        - **Prompt**, for example:

      "You receive a single message from the user.

      Classify it into one of: todo, idea, question, info.

      Then create a short summary (max 120 characters).

      *Return **ONLY** JSON in this format:*

      {

      "category": "todo | idea | question | info",

      "summary": "short summary",

      "raw": "original message"

      }"

    - This way, n8n can parse the JSON directly.
4. Optional: Add a tiny **Action node** that simply passes the JSON along unchanged as final payload.

8.3.2 n8n – Get Telegram Messages

In **n8n**:

1. Create a new workflow.
2. Add a **Telegram Trigger** node:
    - Connect to your bot.
    - Trigger on new messages.
3. Add any optional preprocessing nodes (e.g., strip usernames, sanitize text).

8.3.3 n8n – Send to NodeSpark

This part depends on how you expose NodeSpark as a webhook/bridge, but conceptually:

1. Add an **HTTP Request** node in n8n:
    - Method: POST
    - URL: the endpoint that your NodeSpark integration listens to.
    - Headers:
        - Content-Type: application/json
    - Body JSON:

```json
{
  "workflowName": "AI Inbox Triage",
  "trigger": "webhook",
  "payload": "<<< Telegram message text here >>>"
}
```

    1.
    - In n8n, map the Telegram text field into "payload".
2. The HTTP node receives a **response** from your NodeSpark bridge containing the AI's JSON.
3. n8n parses that JSON and branches:
    - If category == "todo" → send to your task system
    - If category == "idea" → append to Notion "Ideas" page
    - If category == "question" → email you or push into another queue

You now have a Telegram → AI → n8n router using NodeSpark as the brain.

8.4 AI Writing Buddy – Prompt Profiles + Quick Run

Goal: Turn NodeSpark into a reusable **"mini writing assistant"**:

- Blog intro generator
- Email rewriter
- Social caption helper

Using **Prompt Profiles** and **Quick Run**, you can have multiple "agents" ready to go.

8.4.1 Create Prompt Profiles

1. In NodeSpark, open the **Prompt Profiles** screen (from Settings or wherever it's exposed).
2. Create profiles like:
    - **"Friendly Email Reply"**
        - Tone: friendly, concise, slightly informal.
        - Instructions:

          "Take the user's draft or an incoming email and write a friendly, concise reply.

          Use plain language, 3–6 sentences max.

          Always include a clear next step or confirmation."

    - **"Blog Intro Generator"**
        - Instructions:

          "You write engaging blog post introductions. Given a topic or rough notes, write a 2–3 paragraph intro with a hook in the first sentence and a smooth lead-in to the main content."

    - **"Social Caption (Short)"**
        - Instructions:

          "Write 3 alternative social captions under 180 characters each, separated by line breaks. Tone: casual, positive, modern. Include at most one emoji per caption."

3. Save each profile.

These profiles can be referenced by AI nodes or used in dedicated "writer" workflows.

8.4.2 Build a "Writing Workbench" Workflow

1. Create a new workflow:
    - Name: **"AI Writing Workbench"**
    - Description: *"Use Quick Run + Prompt Profiles to draft emails, intros, and captions."*
2. Add an **AI node**:
    - Set **Mode**: Basic AI or Profile-based if your UI supports selecting a profile.
    - Configure it to:
        - Accept **input text** (what you paste into Quick Run).
        - Use a selected **Prompt Profile** (e.g., chosen via a dropdown in the node).
3. Optional:
    - Add an Action node that copies the output to clipboard automatically.

8.4.3 Use Quick Run as Your Writing Console

1. From the **Workflow List**, long-press **"AI Writing Workbench"**.
2. Tap **Quick Run**.

   In Quick Run:

    - Paste:
        - An email you received
        - Some bullet points about a blog topic
        - A rough idea for a social post
    - Tap **Run Workflow**.
3. Output:
    - Copy your favorite version.
    - Paste into Mail, Notes, Twitter, etc.
4. For debugging:
    - Use **Run History** and **Agent Scratchpad** if the tone isn't right.
    - Go back to the Prompt Profile and tighten the instructions.

Over time, you can have a **gallery of profiles** for different writing styles—all powered by one workflow.

8.5 Putting It All Together – A Sample "Automation Stack"

To wrap up, here's a blueprint for combining everything:

- **Daily AI Standup** (Section 8.1)
  - Runs every morning at 8 AM
  - Sends a summary notification
- **AI Task Extractor Shortcut** (Section 8.2)
  - "Hey Siri, Capture with NodeSpark"
  - Creates reminders from messy thoughts
- **AI Inbox Triage (Telegram + n8n)** (Section 8.3)
  - Telegram → n8n → NodeSpark → n8n → Notion/Todo list
- **AI Writing Workbench** (Section 8.4)
  - Quick Run console for drafting, rewriting, and captioning

Once these are in place, you're not just running a few automations—you've built a **personal automation system** around NodeSpark, AI, Shortcuts, and your favorite tools.

Section 9 – NodeSpark Cheat Sheet

A quick reference you can skim when you're building or debugging workflows.

9.1 Common Node Types (At a Glance)

Trigger Node

What it does: Decides *when* your workflow starts.

Typical trigger modes:

- **Manual / Canvas run**
  - Fired when you tap **Run** inside the editor or use **Quick Run**.
- **Schedule**
  - Runs at specific times (e.g., every morning at 8 AM, weekdays at 6 PM).
- **On webhook / external call**
  - Used when an external tool (like n8n, a server, or another automation) calls into NodeSpark.
  - Usually expects a JSON payload like:

```
{ "workflowName": "My Flow", "trigger": "webhook", "payload": "some text or data" }
```

- **App event**
  - Examples: app launch, app foreground/background, notification tapped (depending on how you wired triggers).
- **Shortcut / Siri**
  - Triggered when the iOS Shortcuts app calls NodeSpark via the **Run Workflow** action.

Remember:

Every workflow should have **at least one Trigger node** so NodeSpark knows when to fire it.

AI Node

What it does: Calls your AI model with prompts, profiles, and (optionally) tools.

Typical modes:

- **Basic / Single-call AI**
  - One prompt in, one answer out.
  - Great for summaries, rewrites, and simple transformations.
- **Agent Mode**
  - The AI acts like an agent.
  - Can call tools: HTTP requests, subflows, notifications, etc. (based on how you configured that node).
  - Generates an **Agent Scratchpad** so you can see its internal reasoning and tool calls.

Key settings to think about:

- **Prompt / Instructions**
  - The "job description" for the AI.
- **Prompt Profile**
  - Reuses pre-saved instructions (e.g., "Task Extractor", "Email Rewriter").
- **Output format**
  - Plain text / markdown
  - Or structured JSON (recommended when other services need to parse it).

Action Node

What it does: Actually *does* something in the real world.

Common action modes:

- **HTTP Request**
  - Talk to APIs like Slack, Notion, n8n, your server, etc.
  - Key fields:
    - URL
    - Method (GET, POST, PUT, etc.)
    - Headers (e.g., Authorization: Bearer <token>)
    - Body / JSON template (often includes {{payload}} from previous nodes)
- **Notification**
  - Send a **local notification** on your iPhone.
  - Title + body; usually body is the AI's output.
- **Run Shortcut**
  - Call into the iOS Shortcuts app.
  - Good for tasks like:
    - Create Reminders
    - Move files
    - Open apps
- **Run Subflow**
  - Run another NodeSpark workflow as a subroutine.
  - Keeps big automations modular.
- **Clipboard / Open URL** (if you wired those)
  - Copy text to clipboard
  - Open links in Safari or other apps

Logic / Router Node

What it does: Branches your flow based on conditions.

Typical uses:

- **If / Else**
  - Example: If payload contains "urgent" → send notification; else → log only.
- **Switch / Match**
  - Example: Check a category field (todo, idea, question) and send each to different paths.
- **Filter / Guard**
  - Stop runs if the payload doesn't match your criteria.

Variables & State

Even if your UI doesn't show a literal "Variable node," the engine supports:

- **Workflow variables**
  - Key/value pairs that live for the duration of a run.
  - Great for:
    - Storing intermediate responses
    - Flags (e.g., isUrgent = true)
    - Aggregating text before sending to AI or HTTP.
- **Run History / Agent Scratchpad**
  - Not variables you edit directly, but:
    - **Run History** keeps a list of runs with success/failure.
    - **Agent Scratchpad** holds internal notes from an Agent AI node.

9.2 Quick Patterns (You'll Use These a Lot)

Think of these as "lego shapes" you can reuse.

Pattern 1 – Classic AI Notification

Use when: You want a daily or weekly AI summary.

Trigger (Schedule) → AI Node (Summarize) → Notification

Example:

- Trigger: Every day at 8 AM
- AI Node: "Summarize this text and extract top 5 tasks"
- Action: Notification with AI output as body

Pattern 2 – Webhook to AI to API

Use when: Another system sends data to NodeSpark, and NodeSpark sends processed data out.

Trigger (Webhook) → AI Node → Action (HTTP Request)

Example:

- n8n posts JSON into NodeSpark
- AI turns messy notes into structured tasks/json
- HTTP Action sends that structured data into Notion/your server

---

Pattern 3 – AI Preprocessor for Shortcuts

Use when: You want Siri / Shortcuts to capture messy input and NodeSpark cleans it up.

Trigger (Shortcut / App Event) → AI Node (extract tasks) → Action (Notification / Clipboard / Return to Shortcut)

Real world:

- Shortcuts gets dictation or selected text
- Sends to NodeSpark
- AI returns clean task list
- Shortcuts then:
    - Creates reminders
    - Adds todos to a list
    - Sends email, etc.

Pattern 4 – Agent with Tools

Use when: You want a mini "copilot" that decides what to call.

Trigger → AI Node (Agent Mode with tools) → *(optional)* **Action**

Tools exposed to the agent might include:

- HTTP request tool
- Subflow runner
- Clipboard / notification / etc.

The agent:

- Plans steps
- Calls tools
- Writes mini notes in the **Agent Scratchpad**
- Returns a final answer for your workflow

Pattern 5 – Logic Router

Use when: You want different paths based on AI classification or raw text.

Trigger → AI Node (classify) → Logic Node →

- Branch A: todo → HTTP → task manager
- Branch B: idea → HTTP → Notion
- Branch C: question → Notification / email, etc.

Keeps all triage in one workflow instead of separate flows.

9.3 Prompt Tips (So the AI Actually Behaves)

Good prompts save you a **ton** of debugging time. Use these rules of thumb:

1. **Give the AI a job title**
   - "You are a task extraction assistant…"
   - "You are a writing assistant that drafts email replies…"
   - "You are a routing agent that classifies messages…"
2. **Tell it what the input is**
   - "The input is a messy note the user dictated."
   - "The input is an email the user received."
   - "The input is a JSON blob from another workflow."
3. **Define the output format clearly**
   - For text:
     - "Return 3–5 bullet points only."
     - "No intro, no extra commentary, just the steps."
   - For JSON:
     - "Return ONLY valid JSON in this exact shape: { "category": "...", "summary": "...", "raw": "..." }"
4. **Set limits**
   - "No more than 120 characters."
   - "Three bullet points maximum."
   - "One short paragraph."
5. **Give a mini example (optional but powerful)**
   - Show a tiny input → output example in the prompt.
   - The AI will often mimic that structure and tone.
6. **For Agent Mode: say when to use tools**
   - "Use tools only when needed."
   - "Only call the HTTP tool if you need fresh external data."
   - "If you can answer from the given context, do NOT call tools."

9.4 Logic & Debugging Tips

A few habits that make complex flows way easier to manage:

1. **One responsibility per workflow when possible**
    - Example:
        - Workflow A: classify + tag
        - Workflow B: send notifications
        - Workflow C: do API calls
    - Use **Run Subflow** to connect them instead of building giant monsters.
2. **Use variables to avoid recomputing**
    - Store important values (like category, summary) once.
    - Reuse them in later nodes (HTTP payloads, notifications).
3. **Test with Quick Run first**
    - Use different inputs:
        - "Happy path"
        - Edge cases (very short text, very long text)
    - Check:
        - Output
        - Run Log
        - Agent Scratchpad (for agent runs)
4. **Turn vague prompts into explicit ones**
    - If the AI keeps rambling:
        - Tighten the instructions ("2 sentences max.")
    - If the AI keeps changing format:
        - Tell it exactly what to output.
    - If something keeps breaking downstream (e.g., JSON parsing):
        - Force strict JSON, no commentary.
5. **Start simple, then add magic**
    - Get a minimal pattern working:
        - Trigger → AI → Notify
    - Once it's solid:
        - Add logging, agents, HTTP calls, short-term "scratchpad" behavior, etc.

Section 10 – FAQ & Best Practices

A grab-bag of answers to “why isn’t this working?” and “how should I design this?” questions you’ll run into once you’re deep into NodeSpark.

10.1 Frequently Asked Questions

Q1: My workflow didn’t run. How do I figure out why?

Check these in order:

1. **Does the workflow have a Trigger node?**
    - Open the workflow on the canvas.
    - Make sure you have at least **one Trigger node** connected at the start (Manual / Schedule / Webhook / Shortcut / App Event).
2. **Is the trigger actually firing?**
    - **Manual / Quick Run**
        - Use **Quick Run** from the workflow card’s long-press menu and see if it runs there.
    - **Schedule**
        - Confirm the schedule settings (time, days, etc.) in the Trigger node.
    - **Webhook**
        - Make sure your external tool is actually sending a request to NodeSpark (correct URL / payload).
    - **Shortcut**
        - Make sure your Shortcut is calling the correct workflow name.
3. **Check Run History**
    - Long-press the workflow card → **Run History**.
    - Look at the latest run:
        - Green/success vs red/error.
        - Tap into a run (if you build that) or look at the **Run Log** via **Quick Run** to see where it failed.
4. **Check the Run Log / Agent Scratchpad**
    - Open **Quick Run** on that workflow.

- Run with similar input.
- Look at:
  - **Run Log** messages.
  - **Agent Scratchpad** (for Agent mode) to see internal tool calls & steps.

Q2: The AI output looks messy or too long. How do I control it?

Tighten the prompt in your **AI node**:

- Add explicit constraints:
  - "Return *exactly* 3 bullet points."
  - "Reply with **one short paragraph** under 120 words."
  - "No intro text, no 'Here is your summary:' – just the summary."
- If you need structured data:
  - Tell it to return **only JSON**:

Return ONLY valid JSON in this shape:

```
text

Return ONLY valid JSON in this shape:
{
  "category": "...",
  "summary": "...",
  "raw": "..."
}
No extra commentary.
```

- If you're reusing this logic across flows:
  - Convert it into a **Prompt Profile** and reuse that profile in multiple AI nodes.

Q3: My JSON keeps breaking downstream. What am I doing wrong?

Common causes:

1. **AI is adding commentary**

    - Fix: In the prompt, say:
        - "Return ONLY valid JSON. No backticks, no explanation, no markdown."
2. **Malformed JSON syntax**
    - Trailing commas, unescaped quotes, etc.
    - Fix:
        - Ask the AI to **self-check**:
            - "Validate that your JSON is syntactically valid before returning it."
3. **Unexpected shape**
    - Your HTTP target or downstream tool expects different field names.
    - Fix:
        - Make sure your AI prompt describes the **exact** field names and types the API expects.

Q4: When should I use Agent Mode vs a simple AI node?

Use **Basic AI mode** when:

- You just need:
    - A summary
    - A rewrite
    - A classification
    - A simple transformation
- You don't need extra tool calls or planning.

Use **Agent Mode** when:

- You want the AI to:
    - Decide **which tools** to call (HTTP, subflows, notifications).
    - Execute multiple steps (plan → call tool → refine answer).
    - Use the **Agent Scratchpad** so you can see its reasoning.

Rule of thumb:

- Start with a **simple AI node**.
- Only turn on **Agent mode** when you truly need multi-step tool calling.

Q5: My webhook integration isn't doing anything. How do I check it?

Walk through:

1. **Confirm the Trigger**
    - Make sure your workflow has a **Trigger node** set to "On webhook / external call".
2. **Check Integration Center**
    - Long-press the workflow → **Integrations**.
    - In the Webhooks section, confirm:
        - You see your webhook trigger(s).
        - You're using the suggested JSON shape, or at least including the fields you expect.
3. **Send a test payload from your external tool**
    - Use your tool's "Test" button (n8n, Postman, whatever you're using).
    - Make sure:
        - URL and method are correct.
        - Body includes the expected fields (like payload, workflowName, etc. if you designed it that way).
4. **Check Run History**
    - If NodeSpark is receiving the call, you should see runs in **Run History**.
    - If not:
        - There might be an auth issue or a wrong endpoint.

Q6: How do I hook NodeSpark up to n8n?

Smart pattern:

1. **In NodeSpark**
    - Create a workflow with:
        - Trigger → "On webhook / external call"
        - AI node (transform text / structure JSON)
        - Optional HTTP action to call back into n8n or another service.
2. **In n8n**
    - Use an **HTTP Request** node:
        - Method: POST
        - URL: The endpoint or scheme that NodeSpark is expecting (depending on how you wired it).
        - Body: Send a JSON like:

```json
{
  "workflowName": "Daily Inbox Triage",
  "trigger": "webhook",
  "payload": "Your text or JSON here"
}
```

2.
   - Optionally wait for NodeSpark's output (if you've wired a response pattern) and route that back into an n8n chain.
3. **Test end-to-end**
   - Trigger the n8n workflow.
   - Check NodeSpark's **Run History** and **Run Log**.
   - Adjust payload shape until everything lines up.

Q7: How do I run workflows from Siri / Shortcuts?

High-level:

1. In NodeSpark
   - Build your workflow and note its **name**.
2. In **Shortcuts**
   - Create a new Shortcut.
   - Add the NodeSpark **"Run Workflow"** action.
   - Choose your workflow from the list (or pass the name as a parameter, depending on how you wired the intent).
   - Optionally pass text into it (e.g., from dictation or clipboard).
3. Siri
   - Give the Shortcut a **spoken name** or use the suggested "Run <workflow> in NodeSpark" phrase.
   - Say your phrase to Siri.
   - Siri → Shortcuts → NodeSpark runs your workflow.

Use **Integration Center → Shortcuts & Siri** (long-press workflow → Integrations) as a built-in guide and to **copy** suggested Siri phrases.

Q8: Why am I limited to 2 workflows?

That's by design:

- **Free plan**
  - You can create and use up to **2 workflows**.
- **Upgrade plan**
  - Unlocks **unlimited workflows**.

You'll see your plan status in the **header** of the Workflow list:

- "Free plan – X/2 workflows used"
- Tap **Upgrade** to go to the paywall and unlock unlimited automations.

Q9: What's the best way to organize a lot of workflows?

A few patterns that scale well:

1. **Prefix by area**
   - Inbox – Daily Triage
   - Inbox – High Priority Extractor
   - Content – Blog Outline Generator
   - Content – LinkedIn Summarizer
2. **Use descriptions**
   - Write a short 1-2 line description in each workflow:
     - What it does
     - Where it sends output
3. **Use one workflow as a "router"**
   - Have a central workflow that:
     - Classifies input
     - Calls **subflows** for each category
   - Keeps the high-level logic in one place, and each subflow simple.

Q10: When should I use Templates vs building from scratch?

Use **Templates** when:

- You're new to the app.
- You want a working starting point:
  - AI summarizer
  - Task extractor
  - Agent-style "copilot" flows
  - Webhook + AI + HTTP combos
- You want to see **best-practice prompts** and layout.

Use **New Workflow** when:

- You already know what you want.
- You're chaining multiple existing workflows and templates into something custom.

Remember: you can combine both. Start from a template, then tweak:

- Prompts
- Triggers
- HTTP endpoints
- Notifications

Q11: How do I debug an Agent run specifically?

When an Agent run feels weird or wrong:

1. **Open Quick Run**
   - Feed it the same input.
2. **Run it and watch:**
   - **Run Log**
     - Shows each step the engine took.
   - **Agent Scratchpad**

        - Shows what the agent was planning, which tools it called, what the tools returned.
3. Tweak:
    - Agent prompt:
        - "Use tools sparingly."
        - "Do not call HTTP unless strictly necessary."
    - Tool configuration:
        - Check URLs, headers, etc.
    - Limits:
        - Cap steps, timeouts if your agent is over-thinking.

10.2 Best Practices for Designing Solid Workflows

Think of these as rules of thumb once you're past "hello world."

1. **Start tiny, then layer complexity**
    - Get a simple chain working:
        - Trigger → AI → Notify
    - Then extend it:
        - Add HTTP call
        - Add logic router
        - Add subflow / agent mode
2. **Use Prompt Profiles for repeatable behavior**
    - If you reuse the same kind of prompt (summaries, task extraction, classification), put it in a **Prompt Profile**.
    - That way:
        - You only improve it in one place.
        - All AI nodes using that profile get better instantly.
3. **Keep flows modular**
    - Instead of one giant everything-flow:
        - Break it into smaller workflows.
        - Use **Run Subflow** actions from a "parent" workflow.
4. **Use Quick Run before wiring external tools**
    - Always test with **Quick Run** first.
    - Once the workflow behaves with sample input:
        - Then wire it to Shortcuts, APIs, webhooks.
5. **Document your flows**
    - Use:
        - Workflow description
        - Internal comments (e.g., in "notes" fields on nodes if you added them).
    - Future you (or your team) will thank you.
6. **Fail gracefully**

- In Agent prompts, mention what to do if a tool fails:
  - "If a tool returns an error, explain briefly what failed and still return the best possible answer."
- In HTTP nodes:
  - Handle failure cases in downstream logic (e.g., branch if error text is present).

7. **Respect rate limits and tokens**
   - Use clear, tight prompts.
   - Avoid unnecessary tool calls (especially in Agents).
   - Consider simplifying output where possible.

Section 11 – About

Thank you for supporting us by purchasing NodeSpark

Website: https.synryzen.com

Support Email: support@synryzen.com

Developer and Owner of NodeSpark & SynRyzen

Matthew C Elliott

Check out our other iOS apps on the Apple Store

GhostLedger: Your Personal Finance App

Lexora: AI Contract Lawyer

BookGlass: Epub and PDF Reader