

# AI Agents for Order Placement

## Problem Statement

As the business scales, the number of daily sourcing orders is expected to rise significantly. Currently, a small operations team handles all order placements, tracks deliveries, manages returns, and updates order details manually in the admin portal. This manual process is labor-intensive and prone to human error. Scaling up would require hiring more people, which is inefficient and increases the likelihood of mistakes. Additionally, to expand the share of in app transaction content on the User App, there is a need to process transactions across more websites, further straining manual operational capacity. The existing approach is not sustainable as order volume and platform complexity grow.

## Proposed Solution

To address the challenges of manual order processing and scaling inefficiencies, the proposed solution is to implement an **AI agent** that automates the purchasing workflow.


- The **AI agent** receives a list of **product URLs**, along with specific order details such as **quantity**, **size**, and **color**, directly from a **Google Sheet**.
- For each entry, the agent independently places the order on the respective website, following the provided specifications.
- After successfully completing each purchase, the agent updates the Google Sheet with the corresponding order ID.

## Technical Implementation

### n8n Workflow

Manual Triggers to execute the workflow. Using loops to iterate over the urls in google sheet, **RunTask** nodes to start the workflow(can be observed in the terminal which is used for server), Checking status of task every 3 seconds using **get task status** node and when status changes to **Finished**, the output of the task is fetched using **get task** node. The output is passed to an ai agent that updates the sheet. Prefer high context window models for “**Browser Use**” tasks and quicker model for ai agent node.

### Browser Use

Browser Use is the easiest way to connect your AI agents with the browser. It makes websites accessible for AI agents by providing a powerful, yet simple interface for browser automation. More information on this can be found at:  [Browser Use](#)

[se - The AI browser agent](#)

### Local Bridge

The n8n workflow cannot operate a browser through browser-use. For this we have utilised a local bridge we found here:

 [draphonix/browser-n8n-local](#)

In essence, it is a server to facilitate communication with “**Browser Use**” through a REST API.

Quite a few changes were done on the base local bridge code to handle our cases. They are mentioned below:

- Sensitive Data support like Passwords, Credit Card, etc without letting LLM know about this data.
- Agent Tools (Functions) to

- Access Messages on Pushbullet; ref: `fetch_otp_for_Credit_Card_login_for_Ibaeuty`
- Copy Paste; ref: `copy_to_clipboard`
- Session Recording
- Disable Security to run on sites with iframes.

Detailed implementation can be found in the repository code at: <https://github.com/wishlink-com/buyforme-aiagen>

**t RESTRICTED CON...**

## Installation & Usage

Pre-requisites:

- Docker
- OpenAI or Anthropic API Key
- Google Service Account Creds
- Python v13.x

High Level Steps:

1. Local setup of n8n on docker
2. Community node addition in locally setup n8n
3. Browser Use Local Bridge
4. n8n workflow setup

### Steps:

1. Setup n8n using docker (More information: [n8n-io/self-hosted-ai-starter-kit](https://github.com/n8n-io/self-hosted-ai-starter-kit)):

```
1 git clone https://github.com/n8n-io/self-hosted-ai-starter-kit.git
2 python3 -m venv venv # create virtual environment
3 cd self-hosted-ai-starter-kit
4 cp .env.example .env # you should update secrets and passwords inside
5 docker compose --profile cpu up
```

The n8n dashboard should be accessible here: <http://localhost:5678/>

2. Add `n8n-nodes-browser-use` community node in n8n. Visit: <http://localhost:5678/settings/community-nodes>. Click on Install and add `n8n-nodes-browser-use`.
3. Setup our custom Browser Use Local Bridge:
  - a. Clone the repository for the bridge service:

```
1 git clone https://github.com/wishlink-com/buyforme-aiagent
2 cd buyforme-aiagent
```

- b. Install the required dependencies:

```
1 pip install -r requirements.txt
```

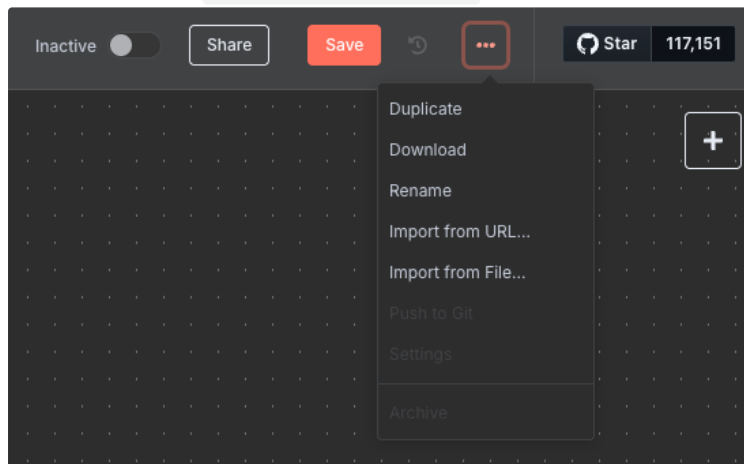
- c. Create `.env` & configure

```
1 cp .env.example .env
2 vim .env # Use IDE if you hate vim
```

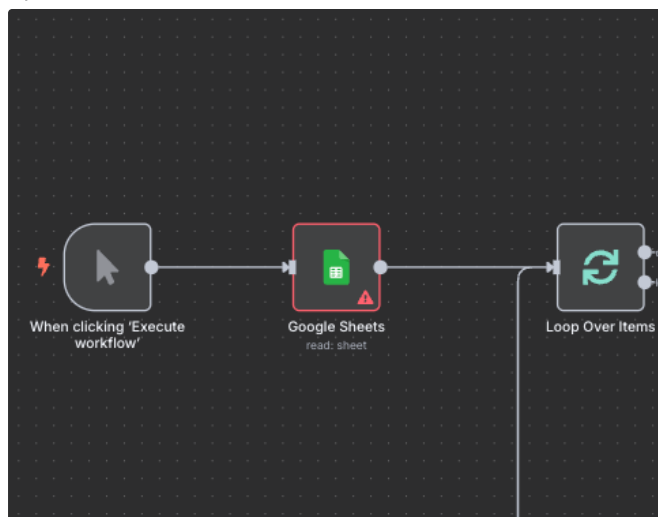
d. Run the bridge service:

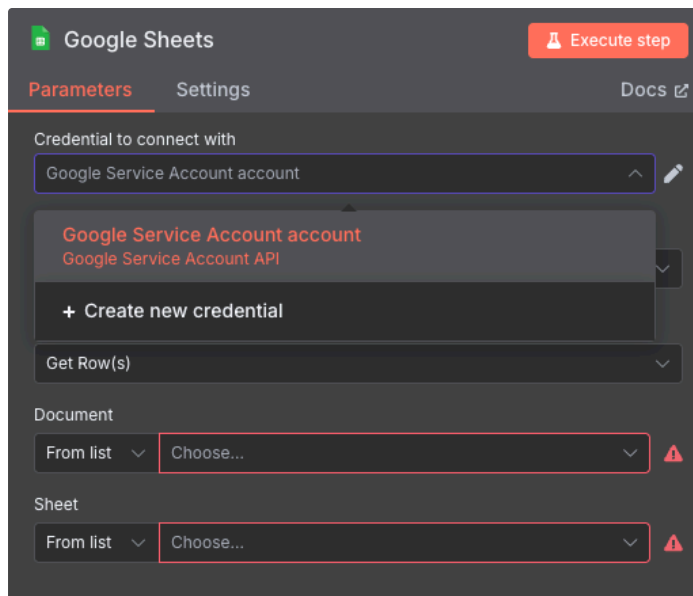
```
1 python app.py
```

4. Import n8n workflow inside n8n. Create a new Workflow inside n8n. Import the `buyformen8n.json` in the [repository](#) using Import from File... .



5. Setup Google Sheets Access. Click on the Google Sheets node. Click on Create new credential. Add the service account details. Contact the respective person for the service account Credentials for accessing Google Spreadsheet.

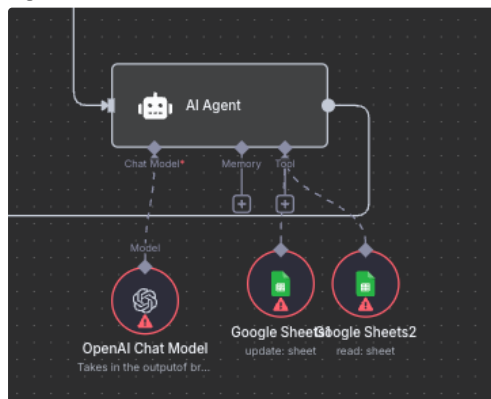




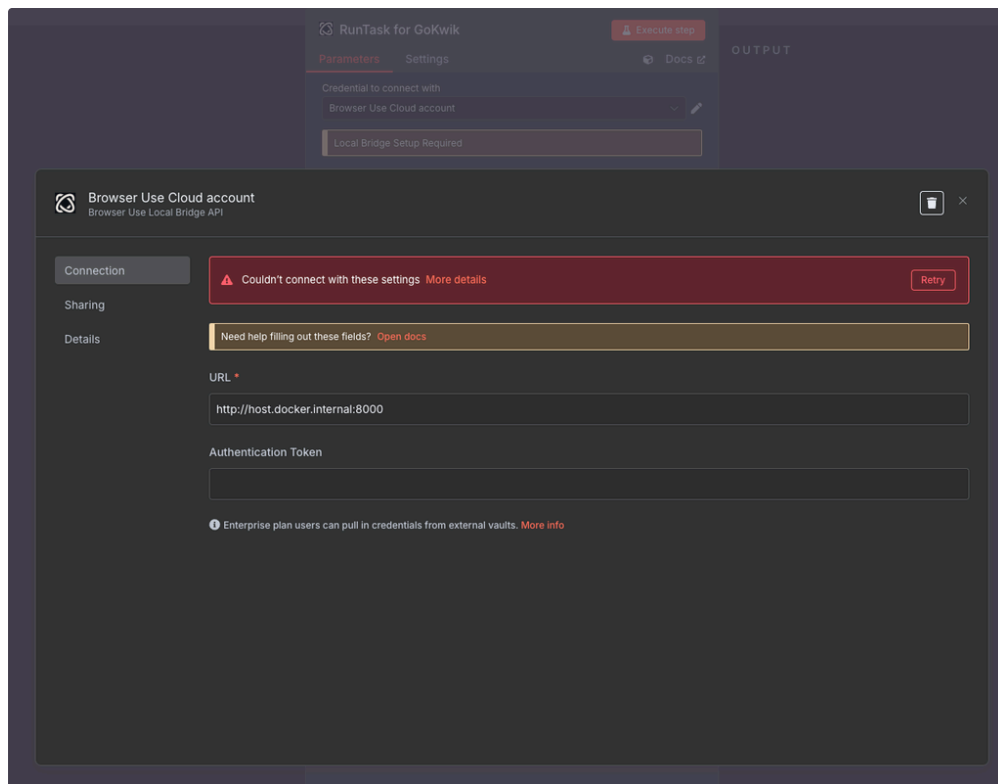
Click on Choose inside Document and select the Google Spreadsheet.

Click on Choose inside Sheet and select the respective Sheet inside the Google Spreadsheet.

6. Similar to above setup the below nodes as well. The only difference being that you don't have to setup Credentials again.



7. Setup OpenAI credentials inside OpenAI Chat Model node in the above screenshot.
8. Update all the browser use nodes with the credentials. When setting the credentials for the first time refer the below image. Make sure to fill in the url as shown as it the request will be made to the host from inside of the docker container. Here is the URL value: `http://host.docker.internal:8000`



9. After everything is setup Press on the **Execute** Button in the n8n workflow to run the entire workflow.