

# 1. 引言

## 1.1 产品概述

我们构建了一套基于 Agent 的智能购物系统，通过多 Agent 协同工作，实现个性化购物需求。系统包含一个人机交互代理和多个功能代理，自动分解用户输入的购物需求（如预算、喜好、用途）为子任务，调用电商代理（如 Amazon 等）进行查询，并整合结果生成购物方案。其核心特色是模块化和可扩展性，支持动态调整需求和方案，实现快速迭代和优化。

## 1.2 编写目的

本手册旨在帮助用户全面了解和高效使用智能购物系统。通过详细的指导和说明，用户可以轻松上手，充分利用系统的功能来满足个性化购物需求。手册适用于所有希望了解系统功能及其操作的用户，包括首次使用者和需要深入了解系统特性的高级用户。

## 1.3 项目背景

本购物系统基于 Mofa(<https://github.com/moxin-org/mofa>)智能体软件框架，并由 SIMPLEST 团队开发的通过多代理（Agents）协同工作的智能购物系统。MoFa 是一个以组合（Composition）的方式构建 AI 智能体的软件框架。使用 MoFA，Agent 可以通过模版方式构建，堆叠的方式组合，形成更强大的超级智能体（Super Agent），从而实现更加强大的功能

# 2. 软件概述

## 2.1 目标

**自动化需求解析：**从用户文本输入中提取购物预算、产品偏好、用途场景等关键信息。

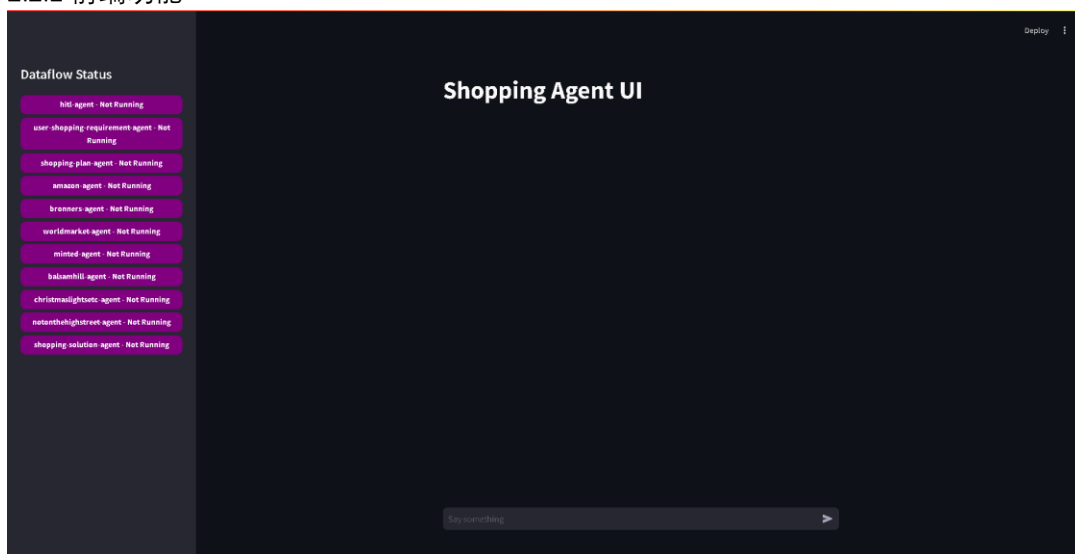
**任务分解与规划：**根据用户需求自动生成查询任务清单（如对多个电商平台的搜索指令）。

**多源信息聚合：**从不同电商代理获取产品信息（价格、规格、库存等），进行跨平台比较与筛选。

**方案生成与反馈：**为用户提供候选购物方案，并支持用户对方案的反馈与再次调整。

## 2.2 功能

### 2.2.1 前端功能



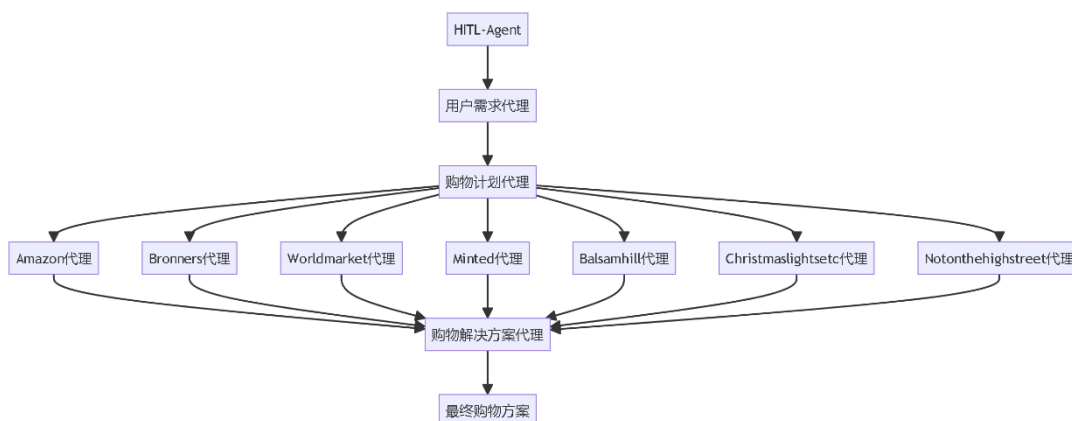
**Dataflow Status 面板：**显示各个代理的运行状态。用户可以查看每个代理（如 HITL、Amazon、Bronners 等）是否正在运行，方便监控系统的整体工作状态。注意：关于 Agent 的运行状态,由于比赛时间的要求，可能存在不准确或者 Agent 状态没有及时更新的问题，我们会在将来的版本上进行修复以及完善。

**聊天窗口：**用户可以在此输入购物需求和偏好，系统将通过人机交互代理进行响应，并提供个性化购物建议。

**实时反馈：**界面显示系统正在处理的任务（如“Bot is typing...”），让用户了解当前的操作进度。

**输入框：**用户可以在底部输入框中输入新的指令或随时调整需求偏好，系统将根据输入进行相应的回答策略调整和更加偏好的反馈。

### 2.2.2 后端功能



【简化，只说明作用】

#### a.HITL-Agent (hitl-agent)

角色：用户与系统交互的入口。

作用：接收用户初始需求 (user\_input)与接收用户对方案的反馈 (shopping\_solution\_user\_input、shopping\_plan\_user\_input)，如“更喜欢 NVIDIA 显卡”或“预算提高到 8500 元”等。

#### 用户购物需求代理 (user-shping-requirement-agent)

作用：将用户模糊、自然语言需求解析为结构化数据，为后续规划做准备。

#### **b.购物规划代理 (shopping-plan-agent)**

作用：根据用户需求，将任务分解为多个电商搜索指令，为各平台代理提供明确的查询参数（例如搜索价格区间、品牌、产品类型）。

**c.各电商代理**（如 amazon-agent、bronnars-agent、worldmarket-agent、minted-agent、balsamhill-agent、christmaslightsetc-agent、notonthehighstreet-agent）

作用：执行特定平台的产品查询与数据获取，为方案生成提供候选商品数据。

#### **d.购物解决方案代理 (shopping-solution-agent)**

作用：根据用户需求和各平台返回的数据，对候选产品进行筛选、组合与排序，生成最终购物方案。

**核心处理逻辑**在于模块化的任务分解与执行，通过多个代理并行处理，实现快速、个性化的购物方案生成。

## **3. 运行环境**

### **3.1 支持软件**

#### **a. 操作系统：**

Windows-Wsl 或 Linux (Ubuntu 20.04 及以上) 以及 mac【待补充】

#### **b. 编程语言与编译系统：**

Python 3.10 及以上。

#### **c. 其他必要的支持软件：**

##### **chrome 与 Chromedriver**

```
sudo apt-get update
```

```
sudo apt-get install -y curl unzip xfb libxi6 libgconf-2-4
```

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

```
sudo apt install ./google-chrome-stable_current_amd64.deb
```

##### **确保 chrome 安装成功：**

```
google-chrome --version
```

获取安装的 google-chrome 版本对应的 chromedriver

<https://chromedriver.chromium.org/>

安装 · 解压 · 配置环境

# 中间的版本号对应着已经安装的浏览器的版本

```
wget https://chromedriver.storage.googleapis.com/86.0.4240.22/chromedriver_linux64.zip
```

```
unzip chromedriver_linux64.zip
```

```
sudo mv chromedriver /usr/bin/chromedriver
```

```
sudo chown root:root /usr/bin/chromedriver
```

```
sudo chmod +x /usr/bin/chromedriver
```

确保 chromedriver 安装成功:

```
chromedriver --version
```

如果你之前已经安装了 chromedriver 在 windows，并且它的路径加入到了 PATH，确保现在 chromedriver 指向的是刚刚解压好的 chromedriver

```
1 | which chromedriver
2 | # should be /usr/bin/chromedriver
```

更多详细步骤请参考源码仓库下的部署文档

## 4. 快速入门

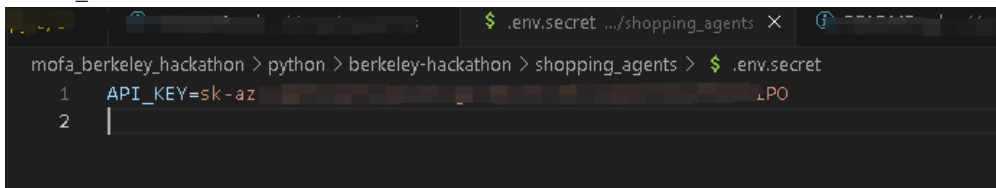
### 4.1 系统初始化

#### 4.1.1 启动后端

请一定按照下面所述顺序运行文件，错误的启动顺序将导致无法运行。

a.首先进入到 python/berkeley-hackathon/shopping\_agents 目录下  
在当前目录下创建一个文件 名字叫做.env.secret,结构如下

API\_KEY=



```
mofa_berkeley_hackathon > python > berkeley-hackathon > shopping_agents > $ .env.secret
1 | API_KEY=sk-az
2 |
```

b.在当前目录下运行命令

```
dora up && dora build shopping_dataflow.yml && dora start shopping_dataflow.yml - attach
```

出现下图红框中代表数据流的 id 号，即代表数据流启动成功。

```

(py310) root@DESKTOP-8MIQOKD:/home/testMofaEnv/mofa_berkeley_hack
○ athon/python/berkeley-hackathon/shopping_agents# dora up && dora
build shopping_dataflow.yml && dora start shopping_dataflow.yml -
-attach
Obtaining file:///home/testMofaEnv/mofa_berkeley_hackathon/python
/berkeley-hackathon/shopping_agents/hitl-agent
Installing build dependencies ... done
Checking if build backend supports build_editable ... done
Getting requirements to build editable ... done
Preparing editable metadata (pyproject.toml) ... done
Requirement already satisfied: numpy<2.0.0 in /root/miniconda3/en
vs/py310/lib/python3.10/site-packages (from hitl-agent==0.3.6) (1
.26.4)
Requirement already satisfied: pyarrow>=5.0.0 in /root/miniconda3
/envs/py310/lib/python3.10/site-packages (from hitl-agent==0.3.6)
(17.0.0)
Building wheels for collected packages: hitl-agent
Building editable for hitl-agent (pyproject.toml) ... done
Created wheel for hitl-agent: filename=hitl_agent-0.3.6-py2.py3
-none-any.whl size=1560 sha256=685a52bf5d7d31315097c9530eec3697ff
95841770b5b6e4b43c2741a2e8d4a4
Stored in directory: /tmp/pip-ephem-wheel-cache-sjm7tg30/wheels
/47/96/8b/59b293a744f9cb96dced43dad21511dc8bb8160a2612d8da40
Successfully built hitl-agent
Installing collected packages: hitl-agent
Attempting uninstall: hitl-agent
Found existing installation: hitl-agent 0.3.6
Uninstalling hitl-agent-0.3.6:
Successfully uninstalled hitl-agent-0.3.6
Successfully installed hitl-agent-0.3.6

```

0193da6e-bea4-713f-bedd-e3136798ce3d

c.在另外一个命令端下面运行 hitl-agent

```

(py310) root@DESKTOP-8MIQOKD:/home/testMofaEnv/mofa_berkeley
○ y_hackathon/python/berkeley-hackathon/shopping_agents# hitl
-agent
Server running on 127.0.0.1:12345...
Connected by ('127.0.0.1', 56752)

```

d.开启第三个命令端,在命令行中使用

```
cd /mofa_berkeley_hackathon/python/berkeley-hackathon/ui && streamlit run socket_client.py
```

可以看到你的页面打开了。

```
(py310) root@DESKTOP-8MIQOKD:/home/testMofaEnv/mofa
○ _berkeley_hackathon/python/berkeley-hackathon/ui# s
treamlit run socket_client.py

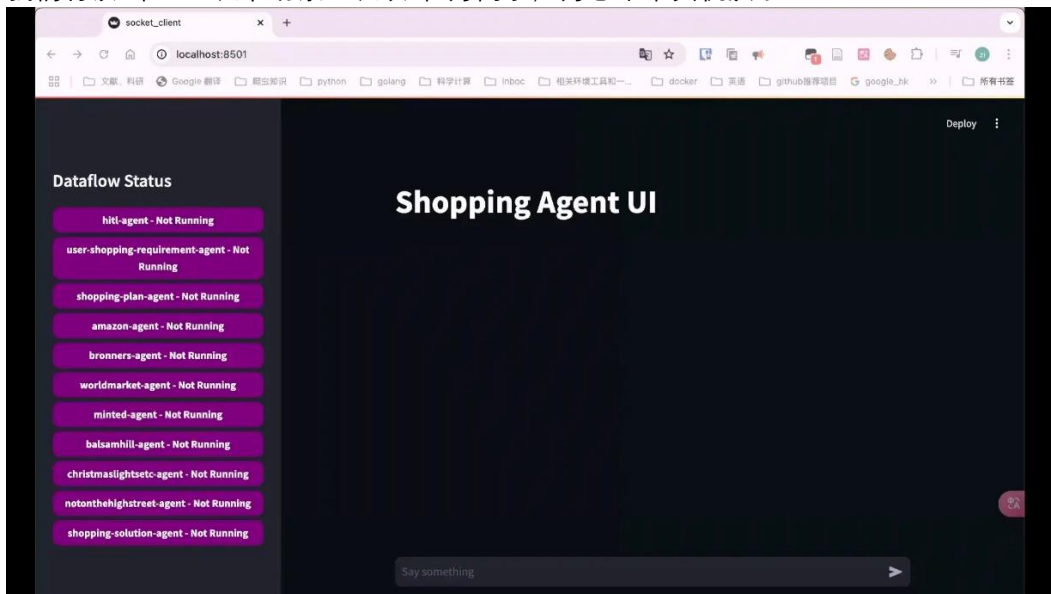
You can now view your Streamlit app in your brows
er.

Local URL: http://localhost:8501
Network URL: http://172.29.214.57:8501
```

保证你的端口 12345 没有被占用，如果被占用了，使用 `lsof -i :12345` 来查看被占用的进程号，使用 `kill -9` 删除它。

#### 4.2.2 前端使用

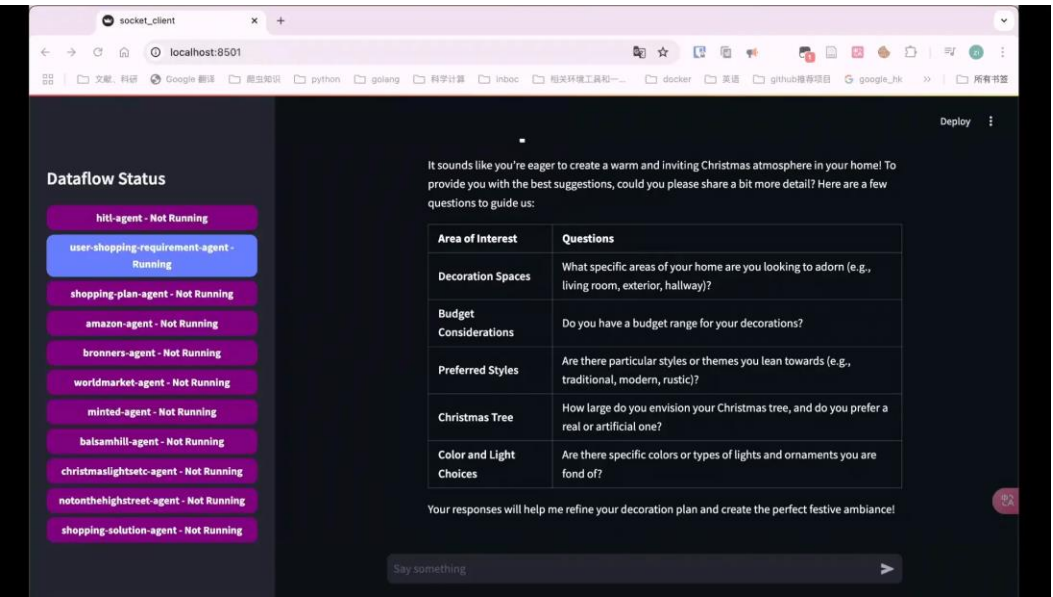
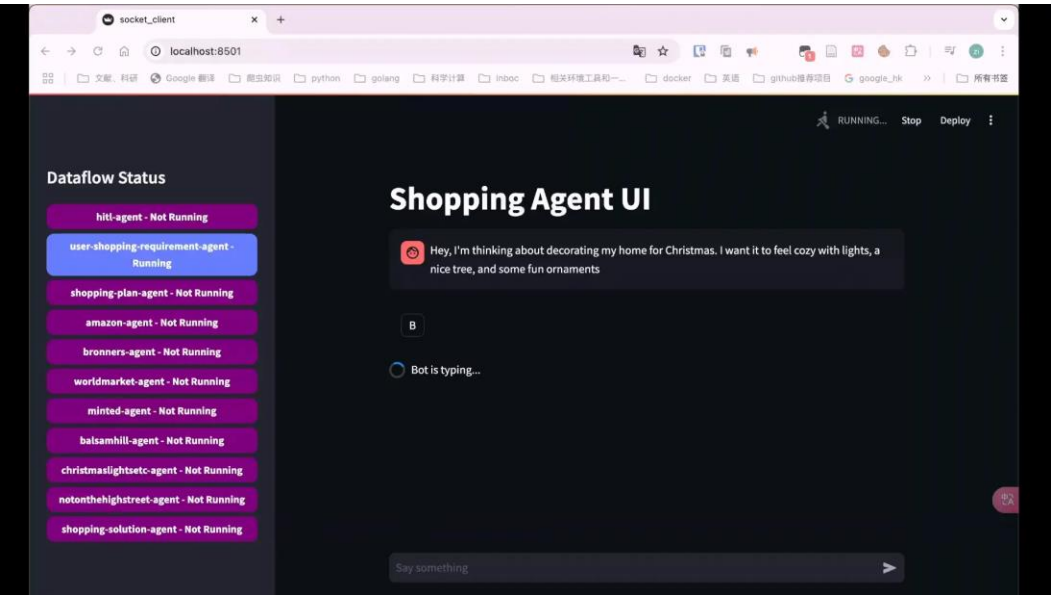
我们将以“布置圣诞节场景圣诞装饰”为例子，为您带来实机演示。



如果您顺利启动后端，那么将会自动跳转到上图这个网页。接下来在网页中的对话中开始我们的对话任务。

第一步：表达需求

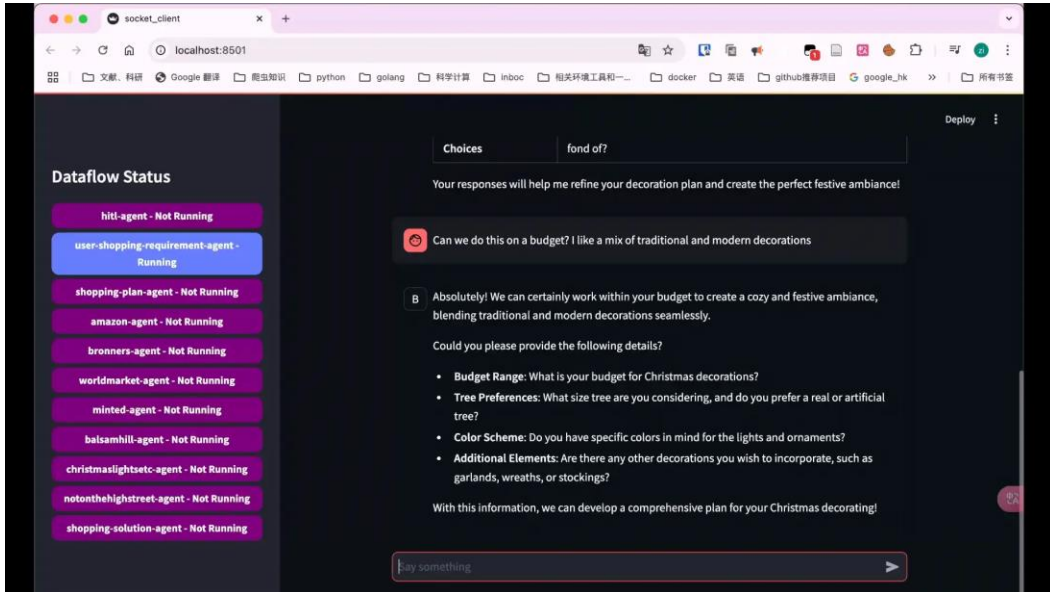
"Hey, I'm thinking about decorating my home for Christmas. I want it to feel cozy with lights, a nice tree, and some fun ornaments."



系统做出如上回答，系统期待您作出进一步的信息描述，以更好为您推荐。

## 第二步：额外要求

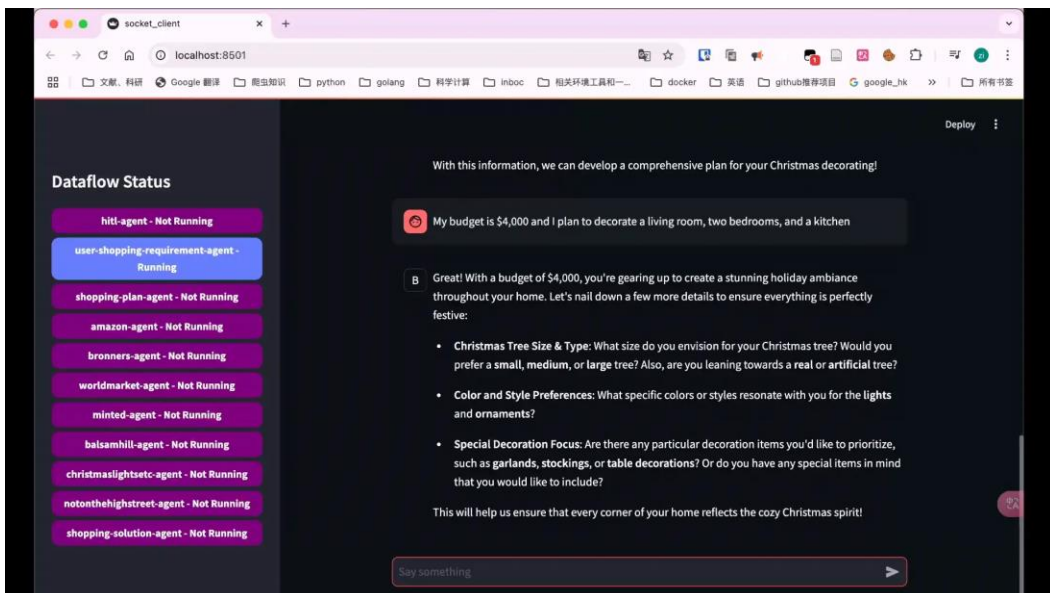
"Can we do this on a budget? I like a mix of traditional and modern decorations."



继续完善您的需求，系统会逐步引导您思考您的需求，您只需要畅所欲言。

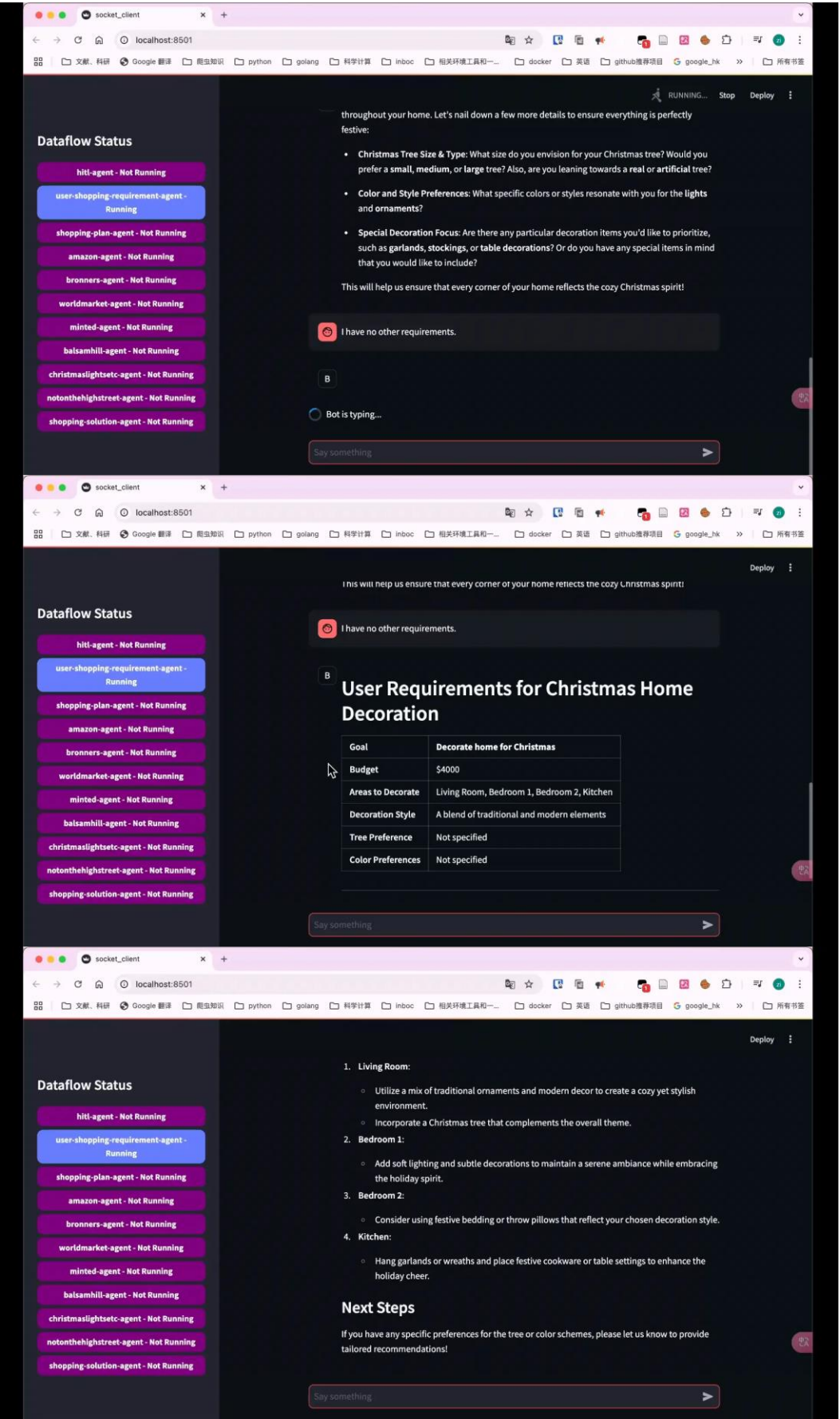
## 第三步：生成购物计划

"Can you put together a list of things I should get? If you have any cute DIY ideas, I'd love to see those too!"



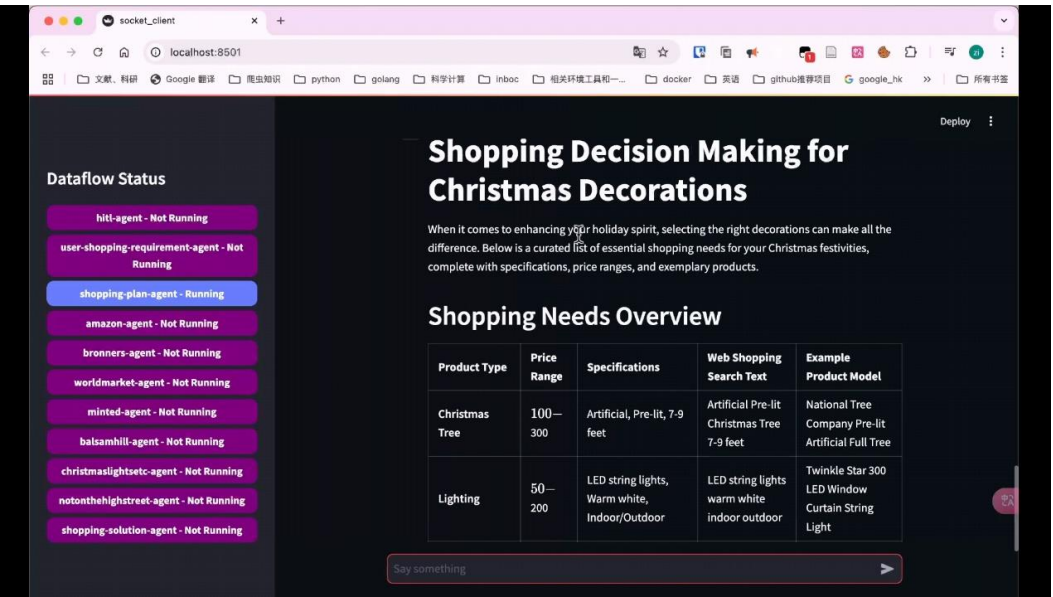
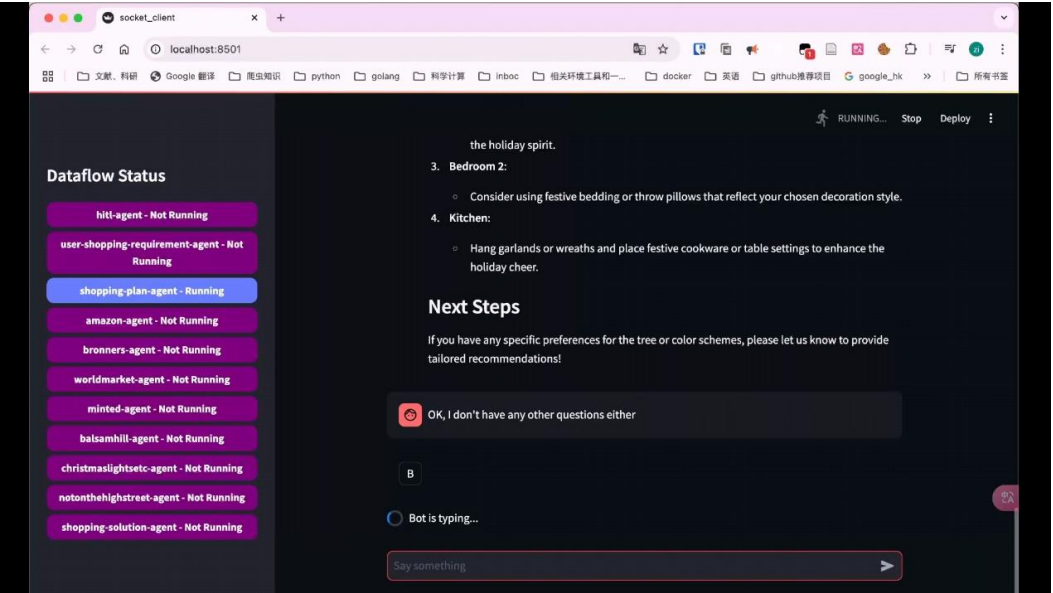


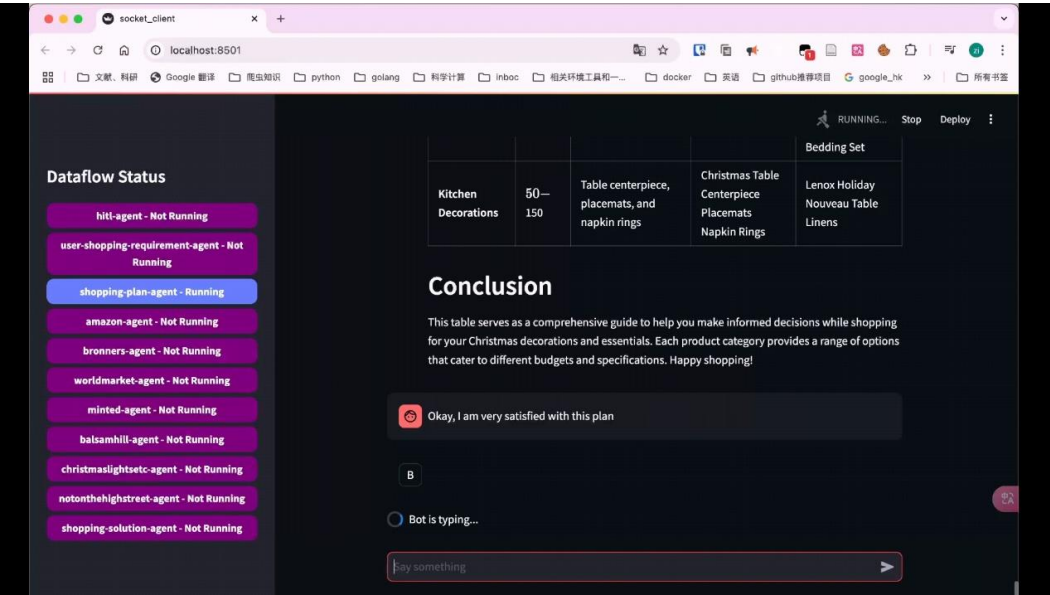
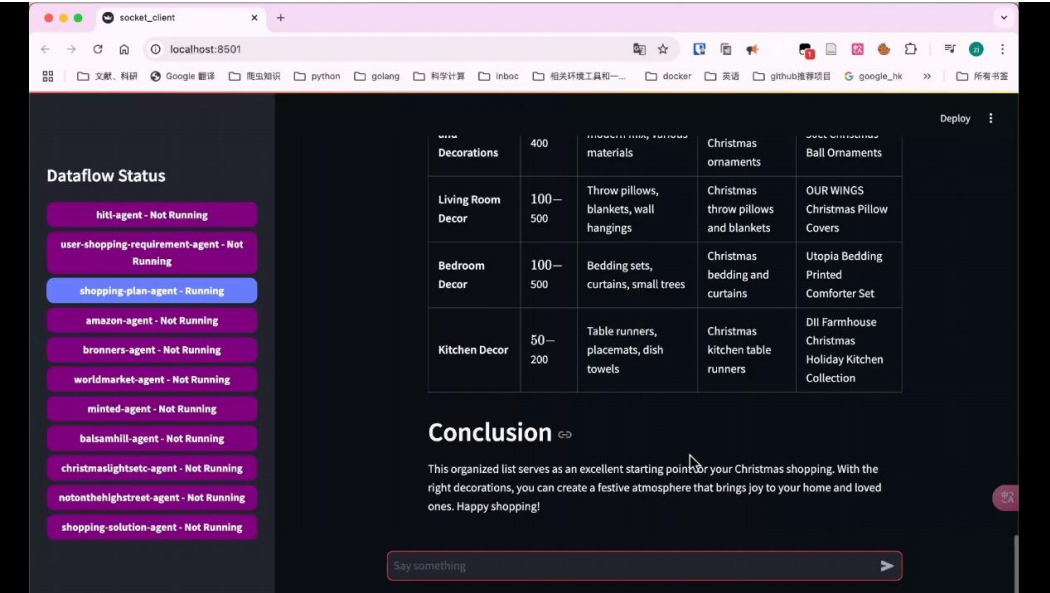
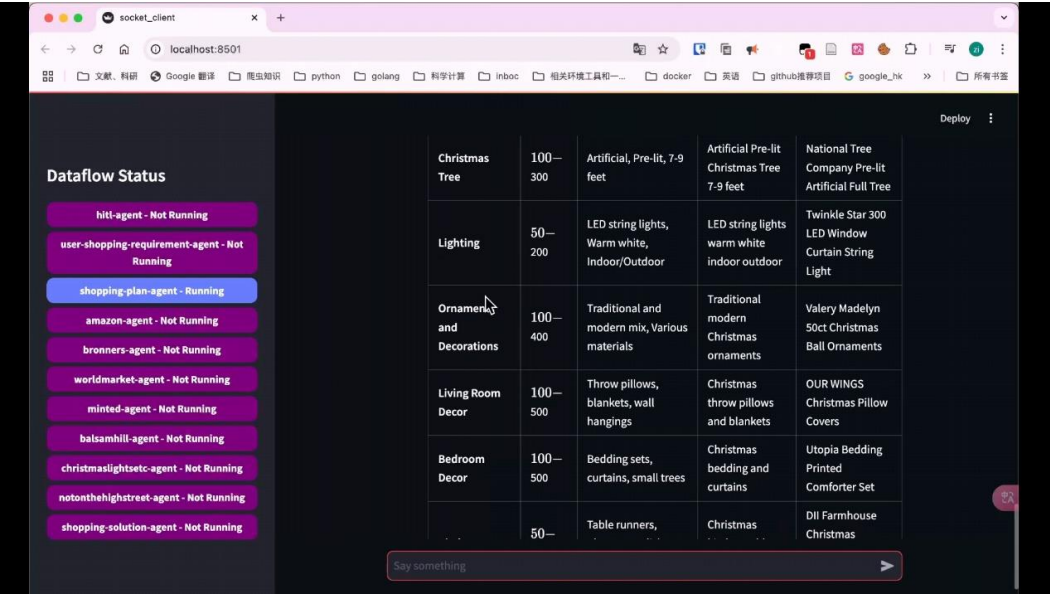
此时大致计划已经完成，系统会继续询问您进行微调。



第四步：微调购物计划

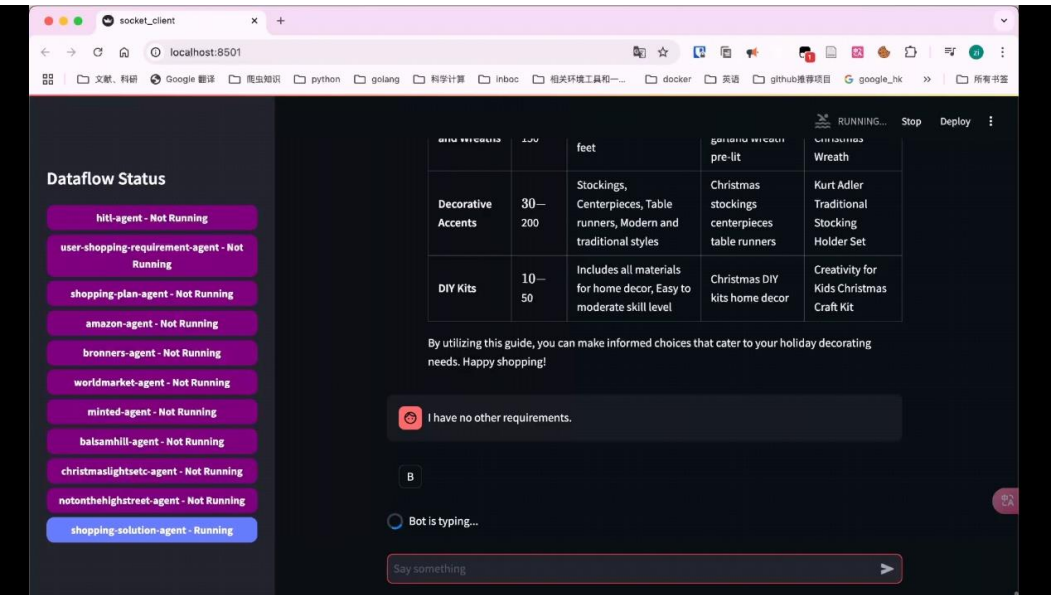
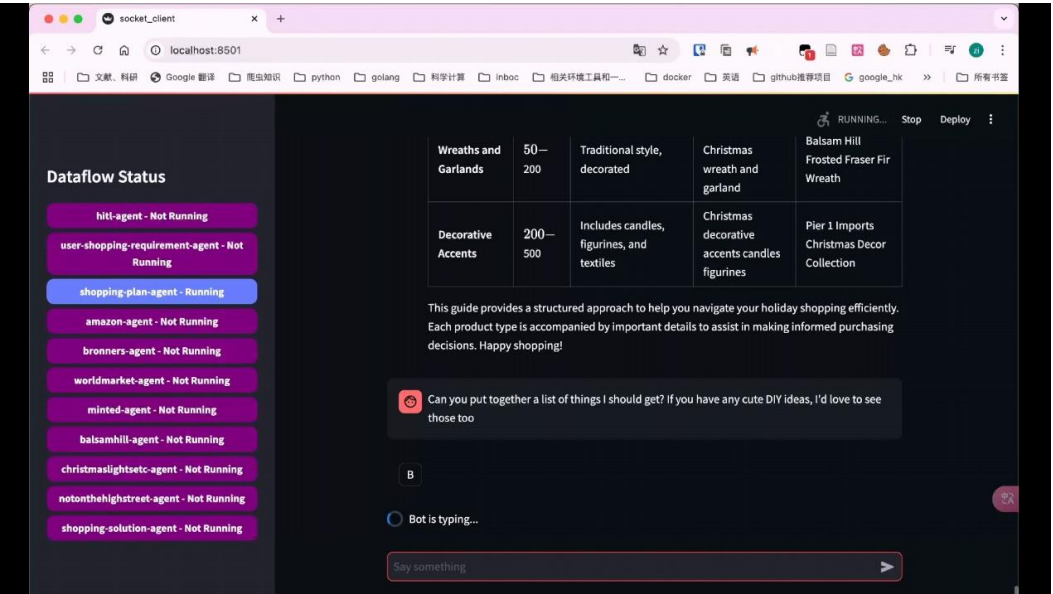
"These ideas are awesome! I love the wreath suggestion. Can you also add some nice dinnerware for our Christmas dinner to the list?"



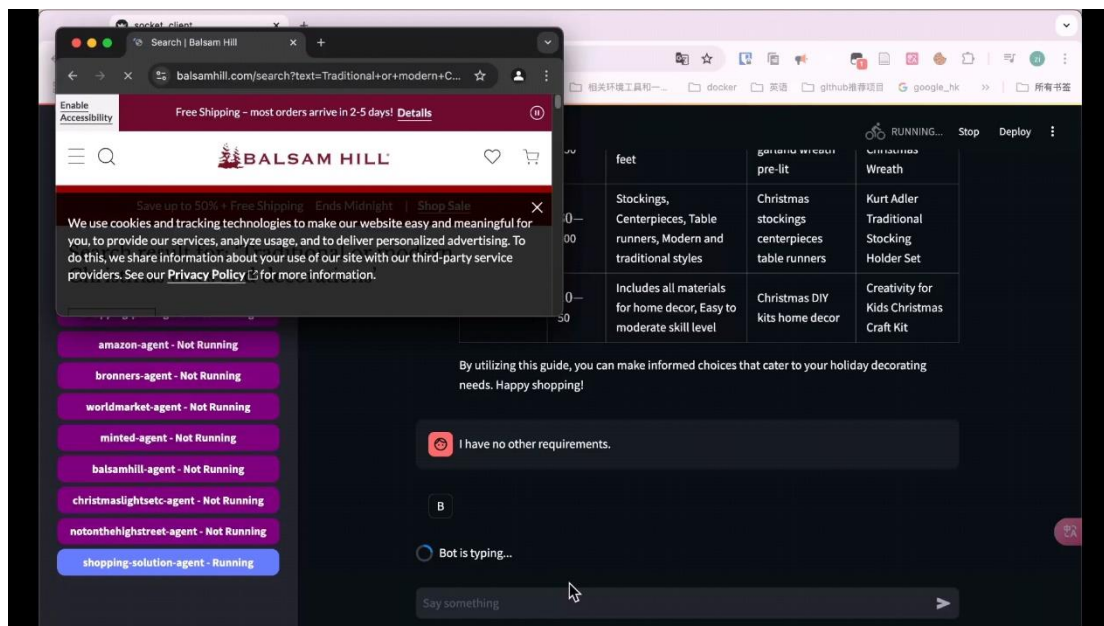


第五步：给出商品推荐

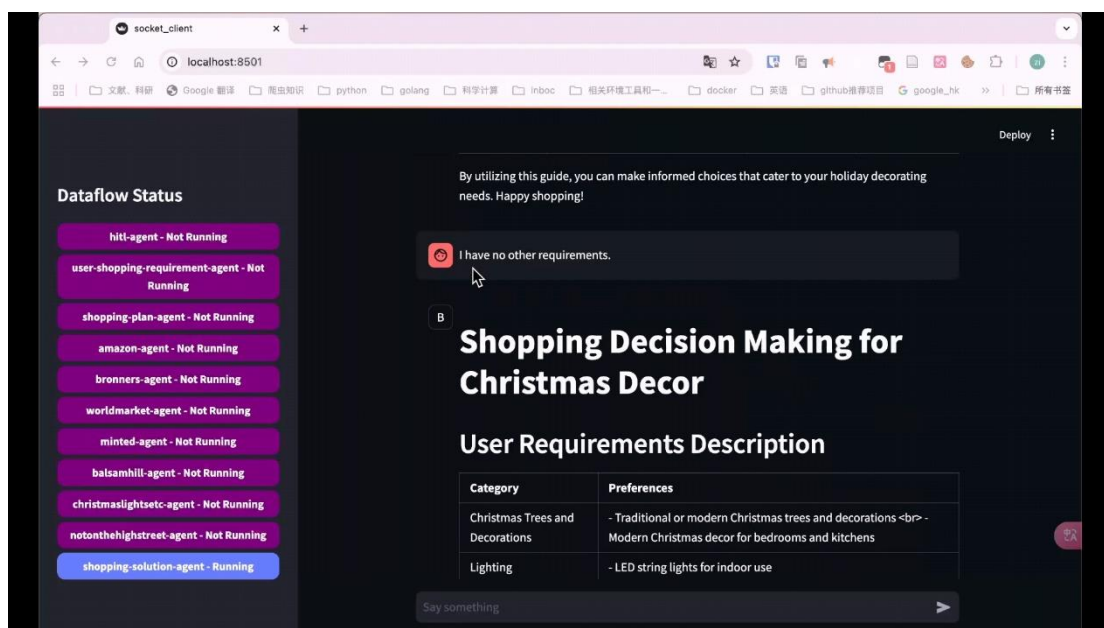
"Okay, I am very satisfied with this plan." I have no other requirements.



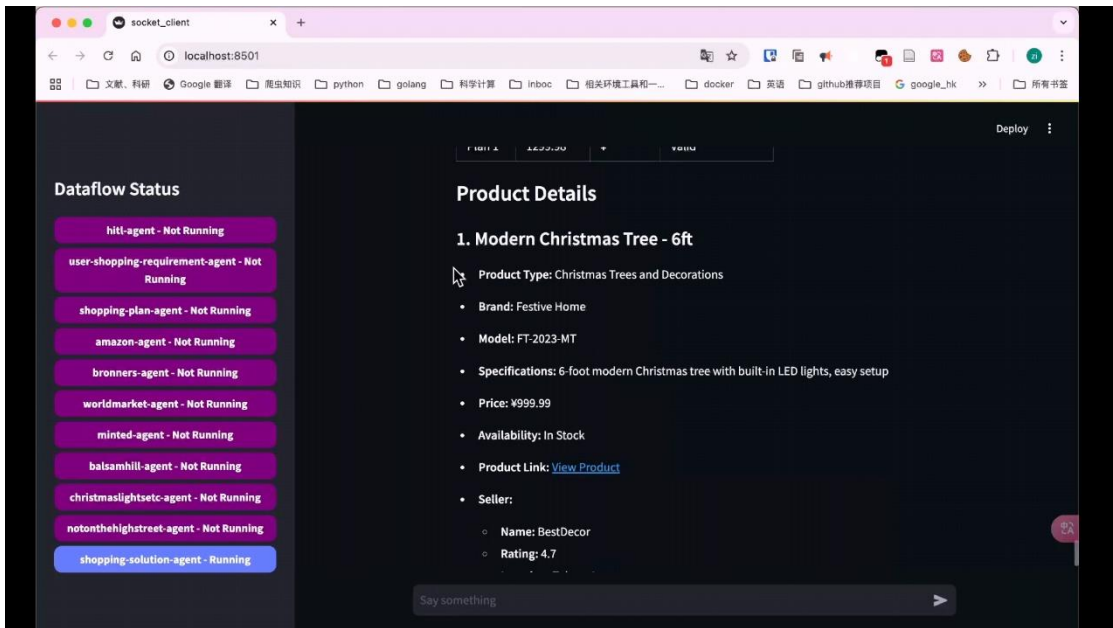
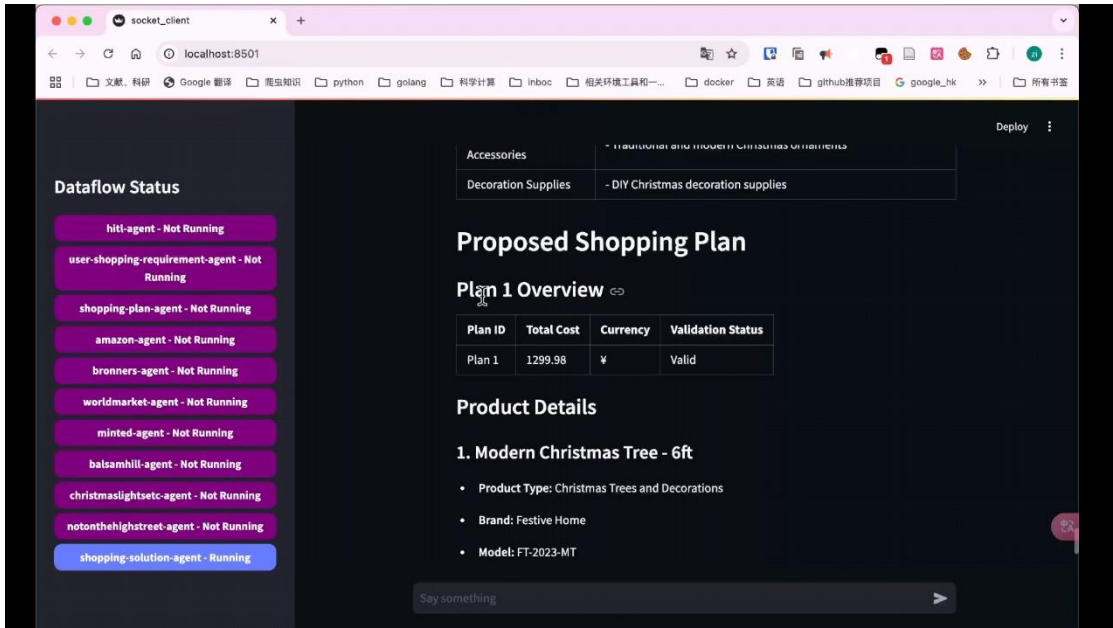


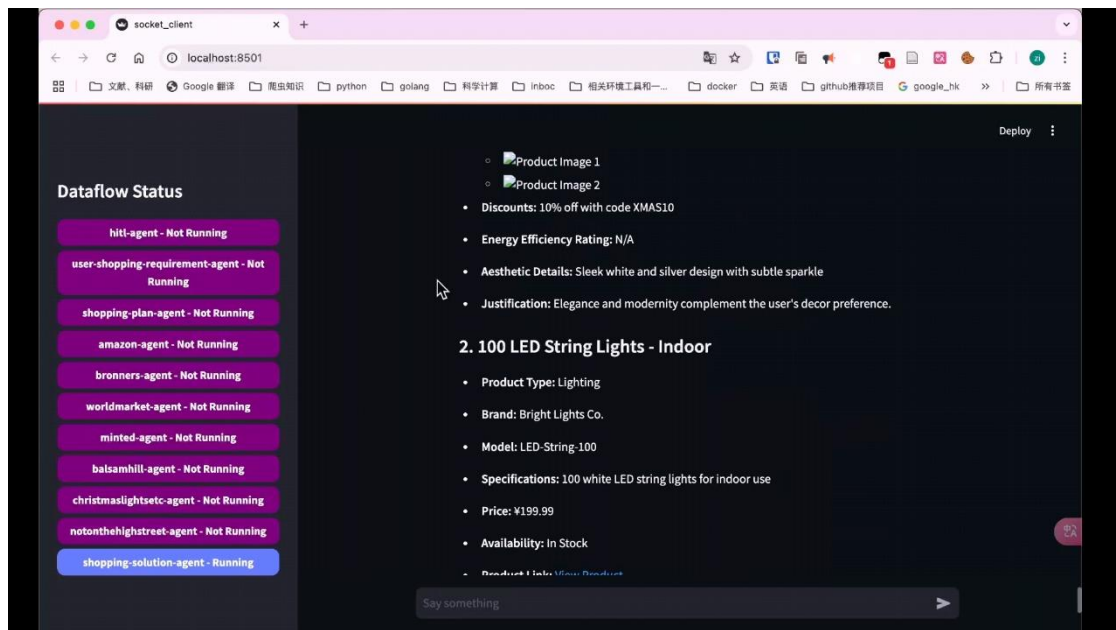
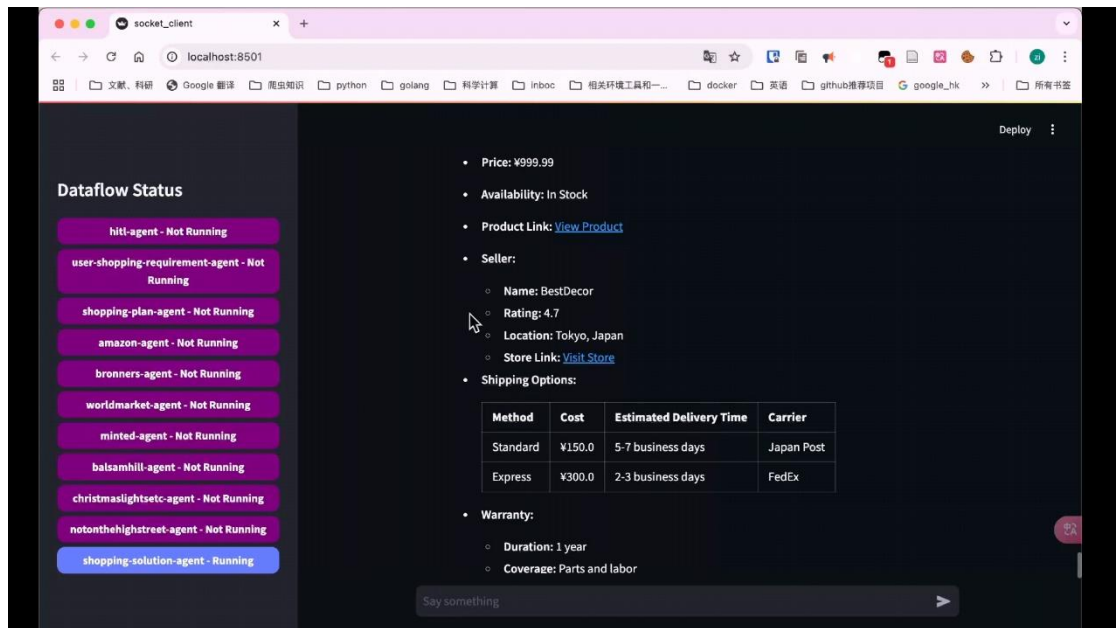


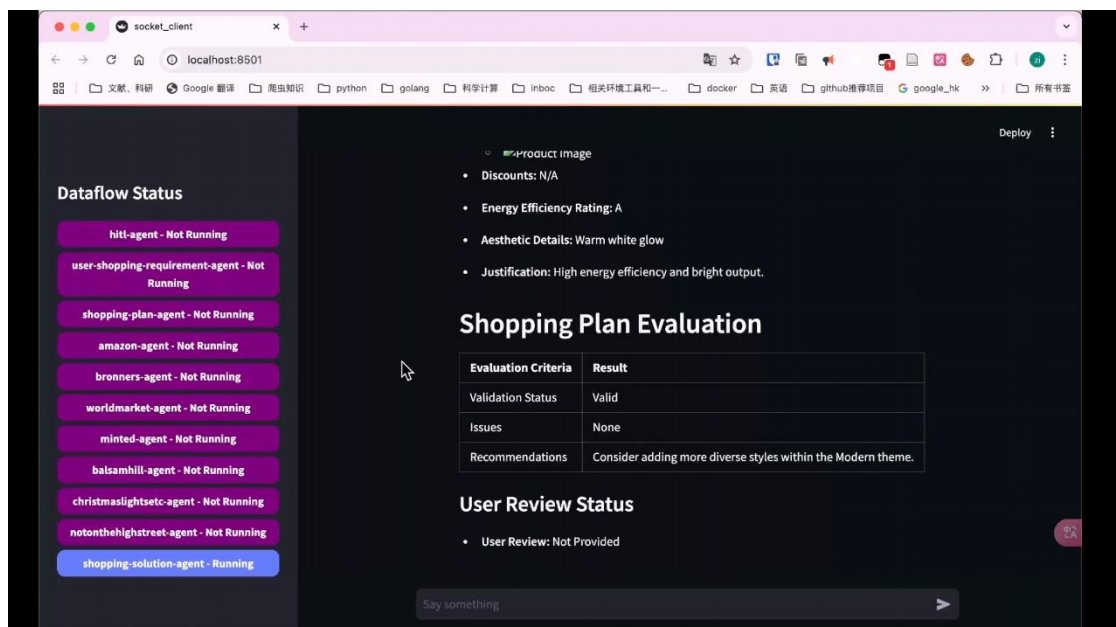
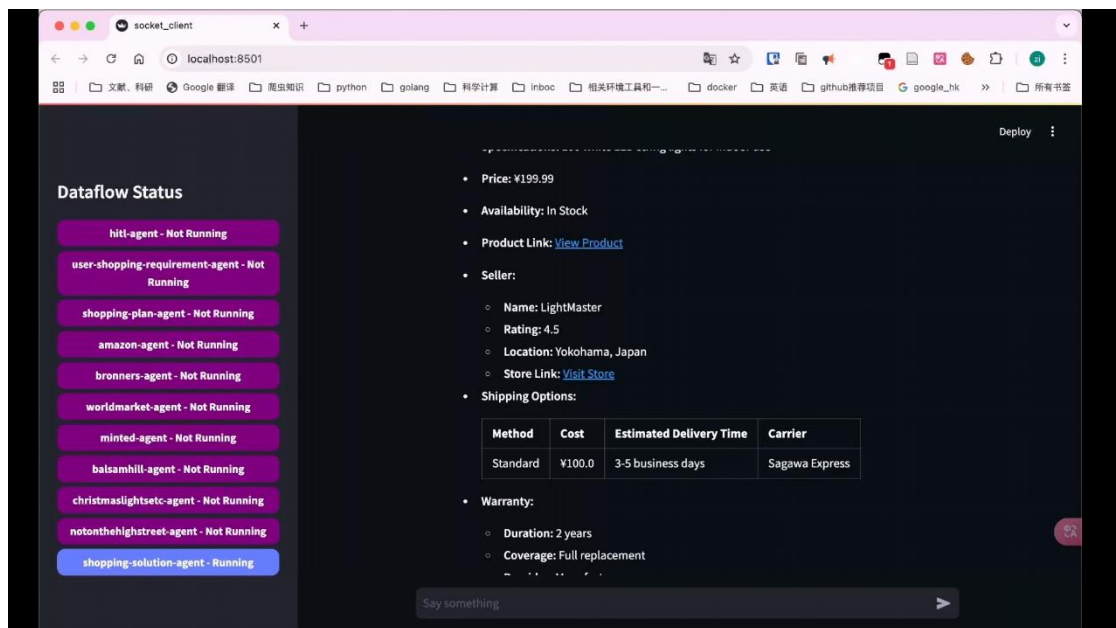
此时，系统开始从购物网站中获取数据，以搜索您计划中的商品，左边的数据状态也同步更新到结果搜索的环节即 shopping-solution-agent



限于不同地区的网络连接速度，此处的系统回答时长可能会在 1-2 分钟左右，搜索完成后，系统将给出如下图的最终报告。注意：我们的寻找物品的 agent 是基于 selenium 的，所以它的结果可能会受限于网页更改或者网络环境，会出现 agent 搜寻不到任何物品的情况，这是正常的，所以结果每次不一定会一致。







系统将最终给出完成的购物建议，包含多件商品，每件商品的类型，品牌，模型，特点，价格，折扣等等信息都会为您总结好，最后会给出购屋计划的评估，以帮助您更好的调整计划。

## 6.常见问题

### 6.1 问题排查【待补充】

a.代理节点设置错误导致的没有访问权限的报错/网速过慢或者节点错误导致的连接报错



```

[ERROR]
Dataflow 0193d905-8308-7a56-8142-fc7fe8660967 failed:

Node `user-shopping-requirement-agent` failed: exited with code 1
with stderr output:
-----
[...]      File "/root/miniconda3/envs/py310/lib/python3.10/site-
packages/openai/_base_client.py", line 957, in request
            return self._request(
                File "/root/miniconda3/envs/py310/lib/python3.10/site-pac
packages/openai/_base_client.py", line 1061, in _request
                    raise self._make_status_error_from_response(err.respons
e) from None

            PermissionDeniedError: Error code: 403 - {'error': {'code':
'unsupported_country_region_territory', 'message': 'Country, reg
ion, or territory not supported', 'param': None, 'type': 'request
_forbidden'}}

Location:
    binaries/runtime/src/operator/python.rs:28:9
-----

```

```

Traceback (most recent call last):
  File "/root/miniconda3/envs/py310/bin/hitl-agent", line 8
, in <module>
    sys.exit(main())
  File "/home/testMofaEnv/mofa_berkeley_hackathon/python/be
rkeley-hackathon/shopping_agents/hitl-agent/hitl_agent/main
.py", line 245, in main
    send_task_and_receive_data(node)
  File "/home/testMofaEnv/mofa_berkeley_hackathon/python/be
rkeley-hackathon/shopping_agents/hitl-agent/hitl_agent/main
.py", line 131, in send_task_and_receive_data
    click_log(event=event, click=click)
  File "/home/testMofaEnv/mofa_berkeley_hackathon/python/be
rkeley-hackathon/shopping_agents/hitl-agent/hitl_agent/main
.py", line 105, in click_log
    node_results = json.loads(event['value']).to_pylist()[0]
)
KeyError: 'value'

```

解决方法：更换代理中的节点，一定换到美国节点，即使延迟高于其余亚洲及欧洲节点。

b.前端数据流状态不跳转。。。[待补充]

## 7. 维护与更新

### 7.1 更新指南

```
git fetch origin  
git pull origin develop
```

在运行后端代码时，先进行最新仓库代码的拉取。此处我们的分支是 develop，如果更换分支请更换你想要拉去的分支的名称

## 8.支持与反馈

技术支持：

如果您有任何代码或其余的疑问，欢迎您随时与我们的技术团队联系：

联系方式：**【待补充】**

反馈渠道：

您可以通过在 github 中提交 pr 的方式或者按照上方所示联系方式与我们的技术团队取得沟通来提出您的意见或建议。