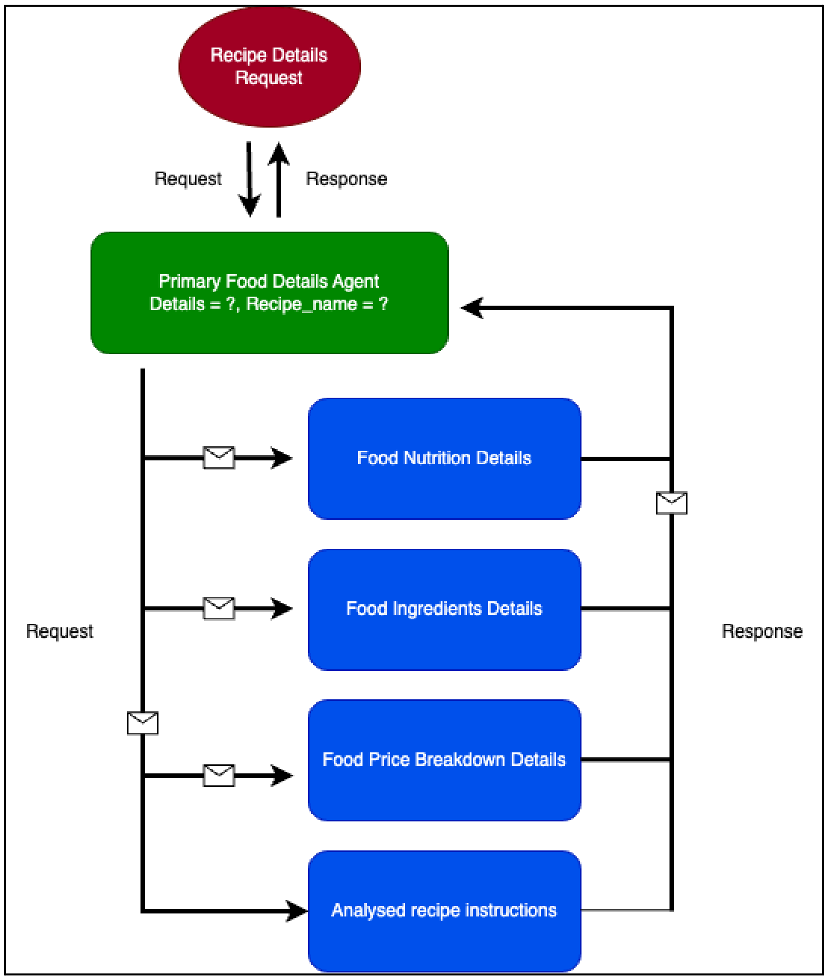
Recipe Details Functions DeltaV. (Function and Secondary functions user guide)

Introduction

In this demonstration, we will explore how to use Agentverse's primary and secondary functions and develop agents accordingly. We will create one primary agent that will take requests from the user and trigger the appropriate secondary agent/function to achieve the user's objective.

Our agent structure looks like as shown in below diagram:


Recipe Details Request
Request
Response
Primary Food Details Agent
Details = ?, Recipe_name = ?
Food Nutrition Details
Food Ingredients Details
Food Price Breakdown Details
Analysed recipe instructions
Request
Response

❖ Step I : Create Primary Agent

Use Agentverse to create an agent with the name ‘Recipe Details Agent’ and include the below script in your agent.

```python
# import required libraries
import requests
from ai_engine import UAgentResponse, UAgentResponseType

# Define a data model for getting recipe request
class RecipeRequest(Model):
   recipe_name : str
   details : str
   response : str

# Create a protocol make recipe requests
recipe_protocol = Protocol('Recipe details protocol')

# define message handler to process user's request
@recipe_protocol.on_message(model=RecipeRequest, replies=UAgentResponse)
async def on_nutrient_request(ctx: Context, sender: str, msg: RecipeRequest):
   ctx.logger.info(f"Received Recipe Nutrient request from {sender} with recipe
name: {msg.recipe_name} and choice: {msg.details}")
   ctx.logger.info(msg.response)
   # Check if the string contains numbered steps
   if re.search(r'\d+\.\s', msg.response):
       # Add line breaks after each numbered step
       message = re.sub(r'(\d+\.\s)', r'\n\1', msg.response)
   else:
       # Split the string using commas and join with newline characters
       message = "\n".join(msg.response.split(','))
      message = message.replace("[", "").replace("]", "").replace("'",
"").strip()
```

```
   await ctx.send(sender, UAgentResponse(message=str(message),
type=UAgentResponseType.FINAL))

agent.include(recipe_protocol)
```

Use the Functions guide to create a primary function for your agent. Please include below details in your function.

```
Unset
{ "FunctionTitle": "Food details Analysis",

  "DescriptionForAIEngine": "This service helps user to get different kinds of
details for the recipe name provided.",

  "Application": "Primary function",

  "Recipe": "Recipe details protocol",

  "Model": "RecipeRequest",

  "Fields": {

    "details": {

      "description": "This describes the type of details user wants to get.",

      "prompt": "Ask this to user from options: Nutritions, Ingredients, Price
Breakdown, Analysed Recipe Instructions."

    },

    "recipe_name": {

      "description": "This describes the recipe for which user wants to get
details."

    },

    "response": {

      "description": "You MUST use subtask to get the value for this field.",

      "subtask": "Search for 'get <details> details for <recipe_name>'."

    }

  }

}
```

❖ Step II : Create secondary agents

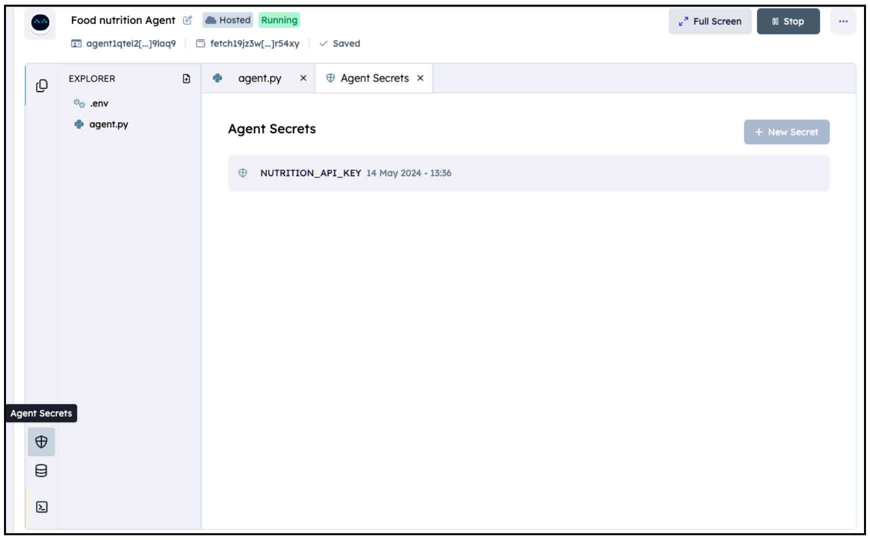
Use Agentverse to create the agents with the name ‘Recipe Nutritions Agent’,‘Recipe Ingredients Agent’, ‘Recipe price breakdown Agent’, ‘Analyzed Recipe Instructions Agent’ and include the below script in your agent.

- ‘Recipe Nutritions Agent’

```python
# import required libraries
import requests
from ai_engine import UAgentResponse, UAgentResponseType
# Define a data model for getting nutrition request
class NutrientRequest(Model):
   recipe_name : str
# Create a protocol for Recipe Nutrients
nutrients_protocol = Protocol("Recipe Nutrients Protocol")


# Define function to get nutritions list
async def nutrition_analysis(recipe_name):
   headers = {
       "X-RapidAPI-Key": NUTRITION_API_KEY,
       "X-RapidAPI-Host": "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"
   }
   url_recipe =
"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/search"
   querystring = {"query": recipe_name}
   response_recipe = requests.get(url_recipe, headers=headers,
params=querystring)
   data_recipe = response_recipe.json()
   recipe_id = data_recipe['results'][0]['id']
   url_nutrients =
f"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/{recipe_i
d}/nutritionWidget.json"
```

```
    response_nutrients = requests.get(url_nutrients, headers=headers)
    nutrients = response_nutrients.json()
    # Collecting basic nutrition facts
    nutrition_list = []
    for nutrient in nutrients["nutrients"]:
        nutrition_list.append(f"{nutrient['name']}:
{nutrient['amount']}{nutrient['unit']}")
    return nutrition_list
# Define message handler to process nutrients request
@nutrients_protocol.on_message(model=NutrientRequest, replies=UAgentResponse)
async def on_nutrient_request(ctx: Context, sender: str, msg: NutrientRequest):
    ctx.logger.info(f"Received Recipe Nutrient request with recipe name:
{msg.recipe_name}")
    nutrients = await nutrition_analysis(msg.recipe_name)
    ctx.logger.info(nutrients)
    await ctx.send(sender, UAgentResponse(message=str(nutrients),
type=UAgentResponseType.FINAL))
agent.include(nutrients_protocol)
```

Please create a secret `'NUTRITION_API_KEY'` which you can get from [RapidAPI](). Agent Secrets can be added using the shield icon in agentverse IDE.



NUTRITION_API_KEY 14 May 2024 - 13:36

Similarly create other agents using below codes.

- ‘Recipe Ingredients Agent’

```python
# import required libraries
import requests
from ai_engine import UAgentResponse, UAgentResponseType

# Define a data model for getting ingrdient request
class IngredientRequest(Model):
   recipe_name : str

# Create a protocol for the Recipe ingrdients
ingredient_protocol = Protocol("Recipe IngredientRequest Protocol")
# Define function to get ingredients list
async def ingredient_analysis(recipe_name):
   headers = {
       "X-RapidAPI-Key": NUTRITION_API_KEY,
       "X-RapidAPI-Host": "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"
   }
   url_recipe =
"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/search"
   querystring = {"query": recipe_name}
   response_recipe = requests.get(url_recipe, headers=headers,
params=querystring)
   data_recipe = response_recipe.json()
   recipe_id = data_recipe['results'][0]['id']
   url_ingredients =
f"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/{recipe_i
d}/ingredientWidget.json"
```

```
   ingredient_list = []


   response_ingredients = requests.get(url_ingredients, headers=headers)
   data_ingredients = response_ingredients.json()
   for ingredient in data_ingredients["ingredients"]:
       name = ingredient["name"]
       amount_metric = ingredient["amount"]["metric"]
       ingredient_list.append(f"{name}: {amount_metric['value']}
{amount_metric['unit']}")
   return ingredient_list


# Define message handler to process nutrients request
@ingredient_protocol.on_message(model=IngredientRequest,
replies=UAgentResponse)
async def on_nutrient_request(ctx: Context, sender: str, msg:
IngredientRequest):
   ctx.logger.info(f"Received Recipe Ingredients request with recipe name:
{msg.recipe_name}")
   ingredient = await ingredient_analysis(msg.recipe_name)
   ctx.logger.info(ingredient)
   await ctx.send(sender, UAgentResponse(message=str(ingredient),
type=UAgentResponseType.FINAL))
agent.include(ingredient_protocol)
```

- Recipe price breakdown Agent

Python
```
# import required libraries
import requests
from ai_engine import UAgentResponse, UAgentResponseType
```

```
# Define a data model for getting price breakdown request
class PriceRequest(Model):
   recipe_name : str


# Create a protocol for recipe price breakdown
price_protocol = Protocol("Recipe Price Protocol")
# Define function to get price breakdown list
async def price_analysis(recipe_name):
   headers = {
       "X-RapidAPI-Key": NUTRITION_API_KEY,
       "X-RapidAPI-Host": "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"
   }
   url_recipe =
"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/search"
   querystring = {"query": recipe_name}
   response_recipe = requests.get(url_recipe, headers=headers,
params=querystring)
   data_recipe = response_recipe.json()
   recipe_id = data_recipe['results'][0]['id']
   url_price =
f"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/{recipe_i
d}/priceBreakdownWidget.json"
   ingredient_price_list = []
   response_price = requests.get(url_price, headers=headers)
   data_price = response_price.json()
   for ingredient in data_price["ingredients"]:
       name = ingredient["name"]
       price = ingredient["price"]
       ingredient_price_list.append(f"{name}: ₹{price:.2f}")


   ingredient_price_list.append(f"totalCost: ₹{data_price['totalCost']:.2f}")
   ingredient_price_list.append(f"totalCostPerServing:
₹{data_price['totalCostPerServing']:.2f}")

   return ingredient_price_list


# Define message handler to process price breakdown request

@price_protocol.on_message(model=PriceRequest, replies=UAgentResponse)

async def on_price_request(ctx: Context, sender: str, msg: PriceRequest):

   ctx.logger.info(f"Received Recipe Price request with recipe name:
{msg.recipe_name}")

   price = await price_analysis(msg.recipe_name)

   ctx.logger.info(price)

   await ctx.send(sender, UAgentResponse(message=str(price),
type=UAgentResponseType.FINAL))

agent.include(price_protocol)
```

- Analyzed recipe integration price breakdown Agent

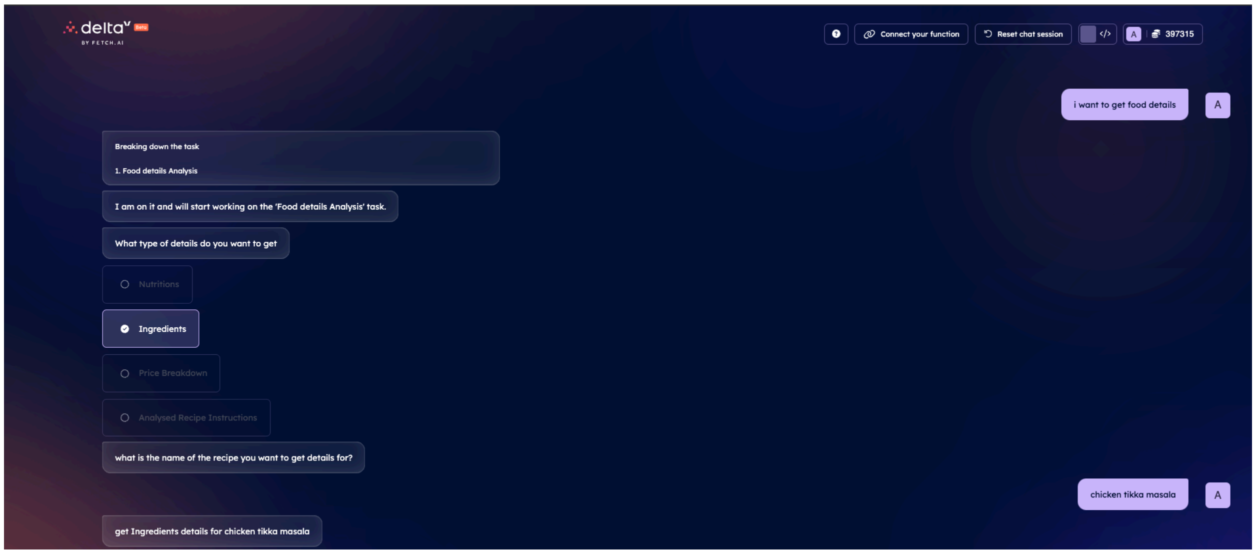
```Python
# import required libraries

import requests

from ai_engine import UAgentResponse, UAgentResponseType


# Define a data model for getting instructions request

class InstructionsRequest(Model):

   recipe_name : str


# Create a protocol for the Hugging Face finbert agent

instructions_protocol = Protocol("Recipe Instructions Protocol")


# Define function to get instructions to prepare recipe

async def instructions_analysis(recipe_name):
   headers = {
       "X-RapidAPI-Key": NUTRITION_API_KEY,
       "X-RapidAPI-Host": "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"
   }

   url_recipe =
"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/search"
   querystring = {"query": recipe_name}

   response_recipe = requests.get(url_recipe, headers=headers,
params=querystring)
   data_recipe = response_recipe.json()
   recipe_id = data_recipe['results'][0]['id']

   url_instructions =
f"https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/{recipe_i
d}/analyzedInstructions"
   steps_list = []

   response_instructions = requests.get(url_instructions, headers=headers)
   data_instructions = response_instructions.json()

   for recipe in data_instructions:
       for step in recipe["steps"]:
           steps_list.append(f"{step['number']}. {step['step']}")
   return steps_list

# Define handler to process recipe instructions request
@instructions_protocol.on_message(model=InstructionsRequest,
replies=UAgentResponse)
async def on_price_request(ctx: Context, sender: str, msg:
InstructionsRequest):
   ctx.logger.info(f"Received Recipe Instructions request with recipe name: {msg.recipe_name}")

   instructions = await instructions_analysis(msg.recipe_name)

   ctx.logger.info(instructions)

   await ctx.send(sender, UAgentResponse(message=str(instructions), type=UAgentResponseType.FINAL))

agent.include(instructions_protocol)
```

Note: Keep in mind that `'NUTRITION_API_KEY'` is added into the agent's secret for each of the agents.

For functions just follow the below template for each secondary agent. Update <detail_type> in below templates with Nutrients, Ingredients, Price breakdown, Analyzed recipe Instructions

```Unset
{
  "FunctionTitle": "Recipe <details_type> details",

  "DescriptionForAIEngine": "this helps to get <detail_type> details for given <recipe_name>.",

  "Application": "Secondary function",

  "Protocol": "Recipe <detail_type> Protocol",

  "Model": "<detail_type>Request",

  "Fields": {

    "recipe_name": {

      "description": "This describes the recipe for which user wants to get details."

    }

  }

}
```

Expected results: (DeltaV)



i want to get food details

Breaking down the task

1. Food details Analysis

I am on it and will start working on the 'Food details Analysis' task.

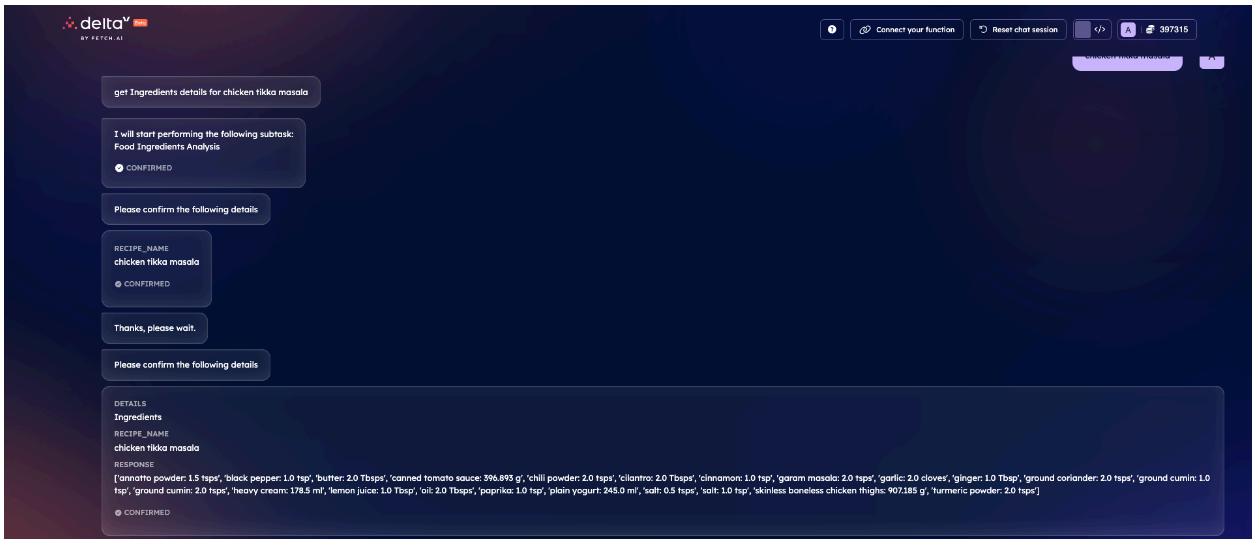
What type of details do you want to get

Nutritions [✓ Ingredients] Price Breakdown Analysed Recipe Instructions

what is the name of the recipe you want to get details for?

chicken tikka masala

get Ingredients details for chicken tikka masala



get Ingredients details for chicken tikka masala

I will start performing the following subtask: Food Ingredients Analysis

CONFIRMED

Please confirm the following details

RECIPE_NAME
chicken tikka masala

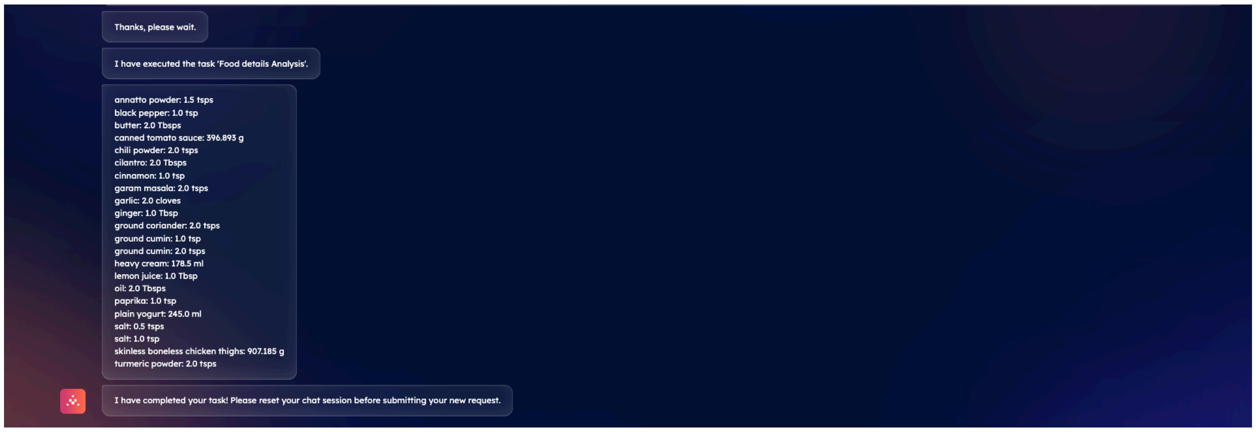
CONFIRMED

Thanks, please wait.

Please confirm the following details

DETAILS
Ingredients
RECIPE_NAME
chicken tikka masala
RESPONSE
['annatto powder: 1.5 tsps', 'black pepper: 1.0 tsp', 'butter: 2.0 Tbsps', 'canned tomato sauce: 396.893 g', 'chili powder: 2.0 tsps', 'cilantro: 2.0 Tbsps', 'cinnamon: 1.0 tsp', 'garam masala: 2.0 tsps', 'garlic: 2.0 cloves', 'ginger: 1.0 Tbsp', 'ground coriander: 2.0 tsps', 'ground cumin: 1.0 tsp', 'ground cumin: 2.0 tsps', 'heavy cream: 178.5 ml', 'lemon juice: 1.0 Tbsp', 'oil: 2.0 Tbsps', 'paprika: 1.0 tsp', 'plain yogurt: 245.0 ml', 'salt: 0.5 tsps', 'salt: 1.0 tsp', 'skinless boneless chicken thighs: 907.185 g', 'turmeric powder: 2.0 tsps']

CONFIRMED



Thanks, please wait.

I have executed the task 'Food details Analysis'.

annatto powder: 1.5 tsps
black pepper: 1.0 tsp
butter: 2.0 Tbsps
canned tomato sauce: 396.893 g
chili powder: 2.0 tsps
cilantro: 2.0 Tbsps
cinnamon: 1.0 tsp
garam masala: 2.0 tsps
garlic: 2.0 cloves
ginger: 1.0 Tbsp
ground coriander: 2.0 tsps
ground cumin: 1.0 tsp
ground cumin: 2.0 tsps
heavy cream: 178.5 ml
lemon juice: 1.0 Tbsp
oil: 2.0 Tbsps
paprika: 1.0 tsp
plain yogurt: 245.0 ml
salt: 0.5 tsps
salt: 1.0 tsp
skinless boneless chicken thighs: 907.185 g
turmeric powder: 2.0 tsps

I have completed your task! Please reset your chat session before submitting your new request.