"A photorealistic image of solar photovoltaic panels on a rooftop with blue silicon cells and metal frames, under bright sunlight"

## **1. User Input Handling**

**File: main.py**

* Entry point for user commands
* Calls Nova to process the prompt

python

*# In main.py (interactive\_async\_main function)*

prompts = []

while True:

user\_input = input("> ")

if user\_input.lower() in ['done', 'finished']:

break

prompts.append(user\_input)

*# Creates tasks for all prompts*

task\_creation\_coroutines = [

nova.create\_task\_list\_from\_prompt\_async(prompt) for prompt in prompts

]

all\_task\_lists = await asyncio.gather(\*task\_creation\_coroutines)

## **2. Intent Detection & Categorization**

**File: agents/nova.py**

* Identifies the intent
* Categorizes and detects image generation requests

python

*# In nova.py (create\_task\_for\_category function)*

*# Text pattern matching to catch image requests*

intent\_lower = intent\_text.lower()

image\_keywords = ["image", "picture", "drawing", "sketch", "illustration", "photo"]

creation\_keywords = ["create", "generate", "make", "design", "produce"]

is\_image\_request = category.lower() == "uncategorized" and (

any(word in intent\_lower for word in image\_keywords) and

any(word in intent\_lower for word in creation\_keywords)

)

if is\_image\_request:

print(f"Detected image generation request: '{intent\_text}'")

*# Route to Ivan for image generation*

return Task(

name=f"Generate image: {intent\_text[:30]}...",

description=f"Generate image based on prompt",

agent="Ivan",

function\_name="generate\_image",

args={"prompt": intent\_text}

)

## **3. Task Routing**

**File: main.py**

* Routes the image generation task to Ivan

python

*# In main.py (process\_prompt\_tasks function)*

for i, task in enumerate(task\_list, 1):

print(f"[Prompt {prompt\_idx + 1}] Task {i}/{len(task\_list)}: {task.name}")

print(f"[Prompt {prompt\_idx + 1}] Delegating to {task.agent}")

if task.agent == "Ivan":

result = await agents["Ivan"].handle\_task\_async(task)

## **4. Image Generation**

**File: agents/ivan.py**

* Processes the image prompt
* Calls DALL-E API
* Saves and displays the image

python

*# In ivan.py (handle\_task\_async method)*

if task.function\_name == "generate\_image":

result = await asyncio.to\_thread(self.generate\_image, self.kb, \*\*task.args)

task.result = result

return result

*# In ivan.py (generate\_image method)*

def generate\_image(self, kb, prompt):

*# Process the prompt*

if "solar" in subject or "pv" in subject:

enhanced\_prompt = "A photorealistic image of solar photovoltaic panels installed on a rooftop..."

*# Log the prompt for debugging*

print(f"DALL-E PROMPT: '{enhanced\_prompt}'")

*# Call OpenAI's DALL-E API*

response = requests.post(

"https://api.openai.com/v1/images/generations",

headers=headers,

json=payload

)

*# Save and display the image*

with open(save\_path, "wb") as f:

f.write(image\_data)

*# Open the image in the default viewer*

os.startfile(save\_path) *# Windows*

## **5. API Integration**

**File: utils/get\_api\_keys.py**

* Retrieves API keys securely

python

*# In get\_api\_keys.py*

def get\_api\_key(service):

*# Return API key for the specified service*

if service == 'openai':

return os.environ.get('OPENAI\_API\_KEY')

## **6. Knowledge Base Integration**

**File: core/knowledge\_base.py**

* Stores the results and image metadata

python

*# In ivan.py (generate\_image method)*

*# Store results*

kb.set\_item("image\_result", result)

kb.set\_item("final\_report", result)

kb.set\_item("last\_dalle\_prompt", enhanced\_prompt)

## **System Flow Diagram**

User Input → main.py → Nova (nova.py) → Intent Detection → Task Creation

↓

Task Routing → Ivan (ivan.py) → Prompt Processing → DALL-E API Call

↓

Image Generation → Save Image → Display Image → Return Results → Knowledge Base

## **Key Innovations**

1. **Smart Routing**: Nova detects image requests even when categorized as "Uncategorized"
2. **Specialized Prompts**: Custom prompts for energy topics like solar and wind
3. **Prompt Visibility**: Both original and enhanced prompts are shown to the user
4. **Fallback Mechanism**: ASCII art generation when API calls fail
5. **Integration**: Seamless integration with the existing agent system

The new solar panel image shows the system is working perfectly, creating a realistic representation of photovoltaic panels with blue silicon cells and white grid lines, exactly as described in the prompt.