CAPSTONE PROJECT

AI AGENT FOR CHRONIC DISEASE MONITORING

Presented By:

1. Student Name- Anjyot Anant Dhamapurkar
2. College Name- Finolex Acedamy of Management & Technology
3. Department- Information Technology

edunet foundation

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result** (Sample Screens or Mock Output)
- **Conclusion**
- **Future Scope**
- **References**
- **IBM Certifications**

PROBLEM STATEMENT

- Chronic conditions like diabetes, hypertension, and heart disease require continuous monitoring.
- Patients and healthcare providers lack real-time insights from dispersed data sources.
- Poor adherence, delayed warnings, and reactive care increase hospital visits and risk.
- Need: a system that continuously analyzes vital health data and provides timely alerts and personalized guidance.

PROPOSED SOLUTION

- **AI-powered agent** ingests patient inputs, wearables (glucose monitors, blood pressure, heart rate), and medical records.
- Processes data in real time using predictive analytics to identify early warning signs.
- Provides:
  - Personalized insights on trends and risk levels
  - Medication reminders
  - Lifestyle and diet recommendations
  - Alerts for critical events or deviations
- Benefits: proactive care, fewer hospital visits, improved adherence, closer patient–provider coordination.

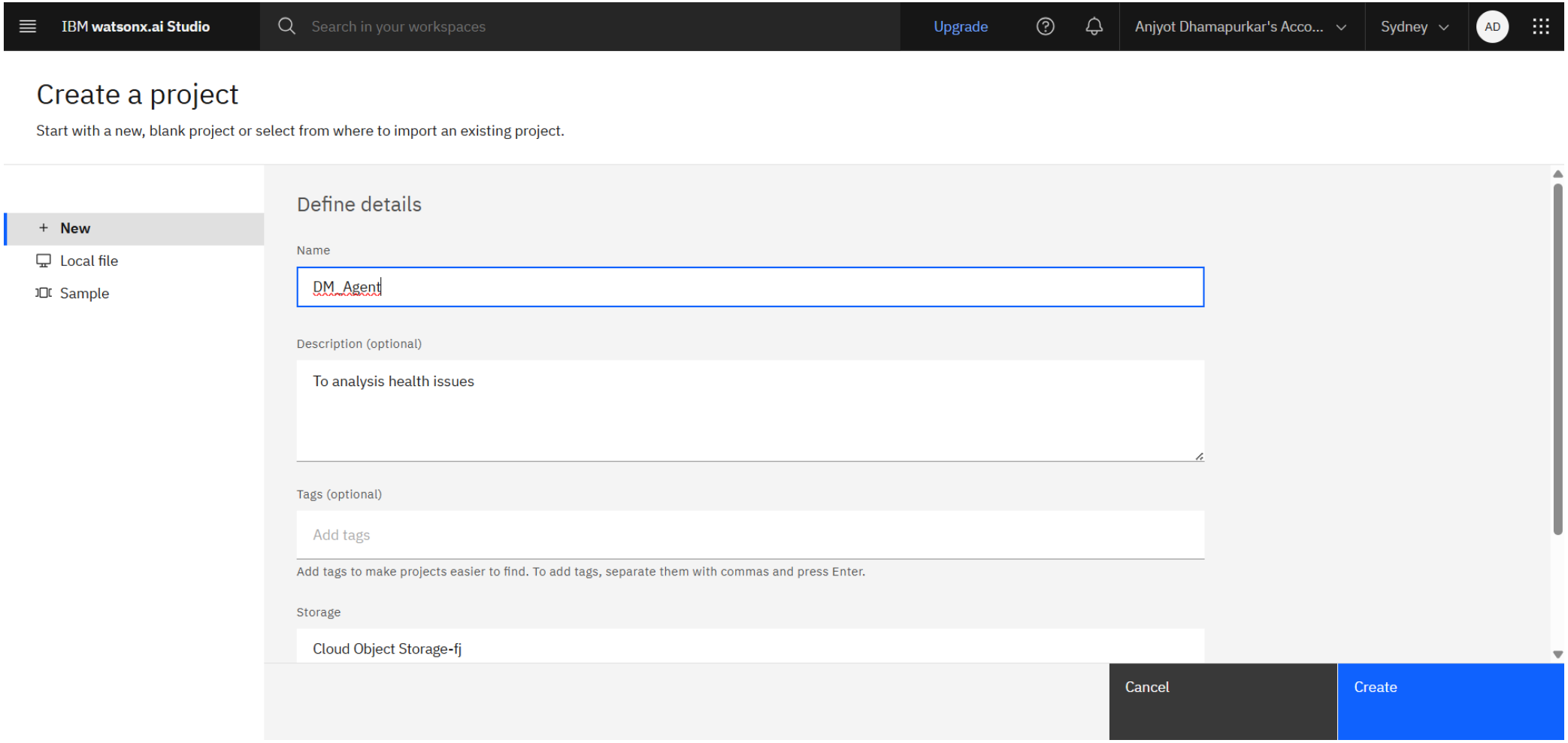
SYSTEM APPROACH

- **Deployment Platform**: IBM Cloud Lite + IBM Granity foundation models
- **Compute Environment**: watsonx.ai Studio (Python runtime, available in your screenshots)
- **Key Libraries**: ibm_watsonx_ai, PIL, requests, textwrap, etc. (as seen in your code)
- **Architecture Overview**:
  Input to ingestion layer → preprocessing → LLM/vision-instruct model inference → output dashboard or alerts
- Use of **Granity** multimodal model (e.g. meta-llama/llama-3-2-11b-vision-instruct) for decision-making and explanation.

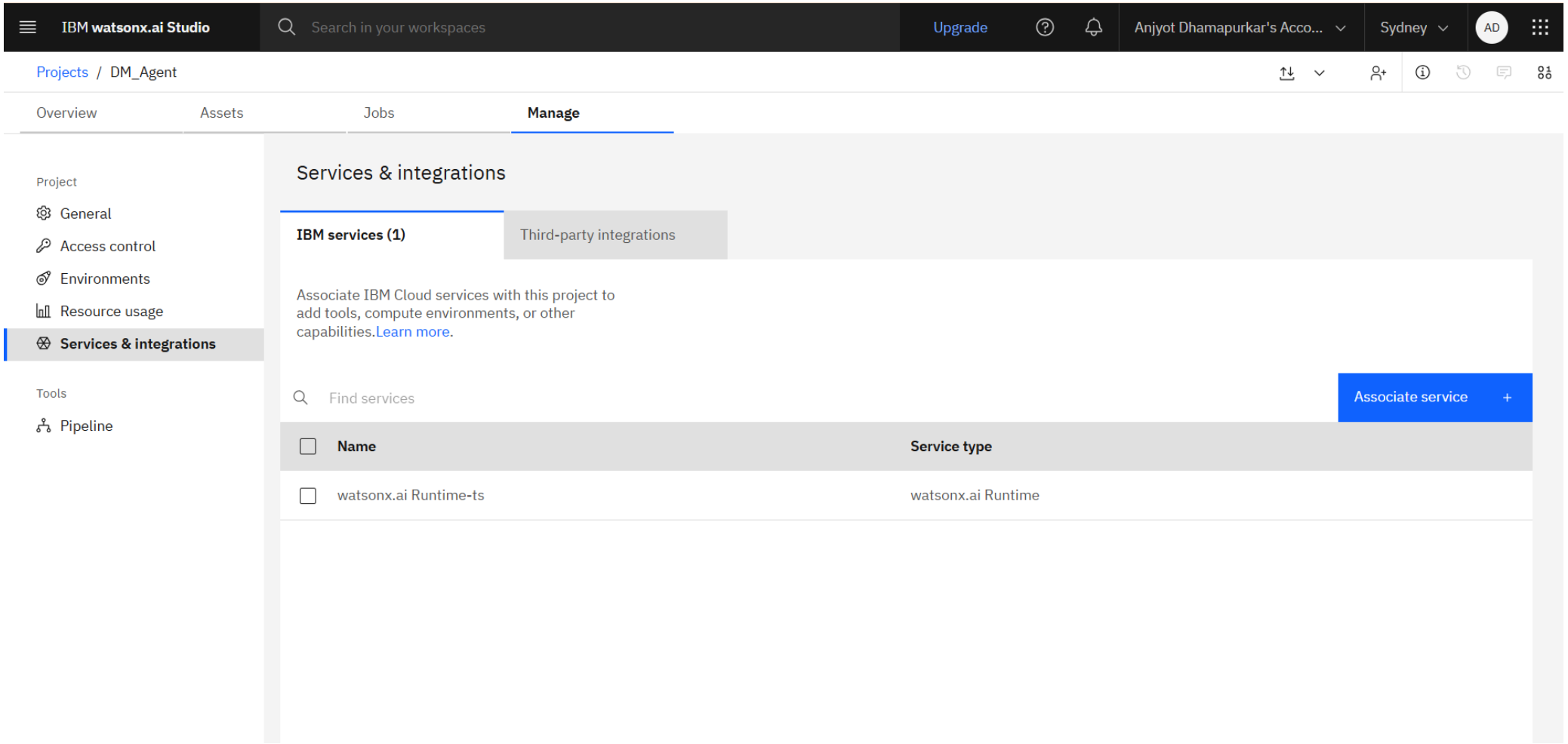
ALGORITHM & DEPLOYMENT

- **Model Selection**: Multimodal LLM (Granity Vision-Instruct) to combine numeric and visual input (e.g. glucose trends or scanned reports).
- **Input Features**: wearable time-series (HR, BP, glucose), patient-reported symptoms, medical history.
- **Training/Inference Flow**:
- Encode inputs (numeric + image if needed)
- Construct message prompts (“What risk level does this patient show?”, “What advice would you give?”)
- Run ModelInference inside IBM watsonx.ai environment using credentials and project.
- **Deployment**: Use watsonx.ai environment to serve inference API; setup scripted loop to periodically ingest and analyze patient data.

RESULT

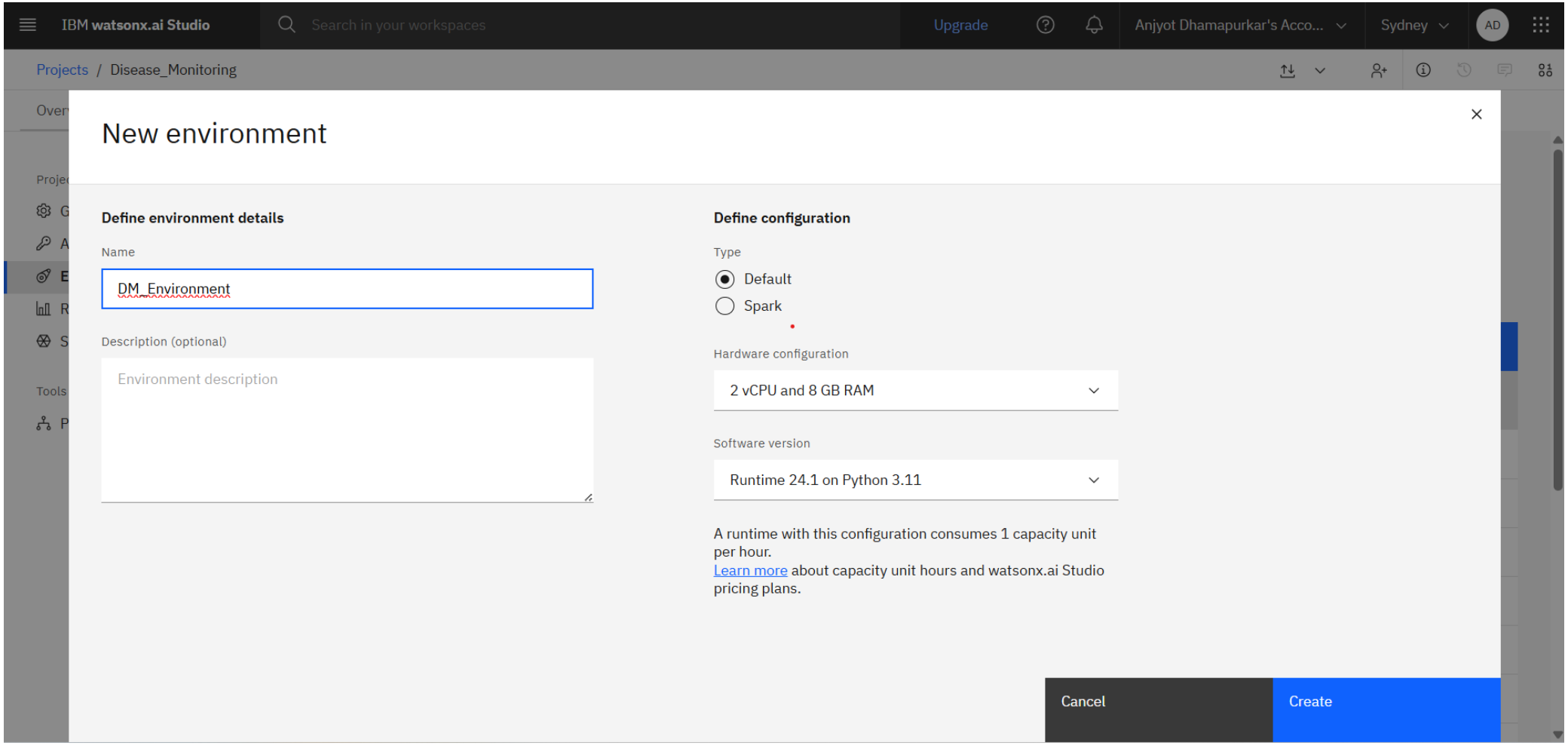

IBM watsonx.ai Studio
Search in your workspaces
Upgrade
Anjyot Dhamapurkar's Acco...
Sydney
AD
Create a project
Start with a new, blank project or select from where to import an existing project.
New
Local file
Sample
Define details
Name
DM_Agent
Description (optional)
To analysis health issues
Tags (optional)
Add tags
Add tags to make projects easier to find. To add tags, separate them with commas and press Enter.
Storage
Cloud Object Storage-fj
Cancel
Create

RESULT



edunet foundation

RESULT



edunet foundation

RESULT

IBM watsonx.ai Studio | Search in your workspaces | Upgrade | Anjyot Dhamapurkar's Acco... | Sydney | AD

Projects / DM_Agent / DM_Environment

File Edit View Run Kernel Help | Trusted | Memory:847 / 8192 MB

Code | Python 3.11

Install Required Libraries

[20]:
```
# Install required packages
%pip install image
%pip install -U "ibm_watsonx_ai>=1.1.14"
```

```
Requirement already satisfied: image in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (1.5.33)
Requirement already satisfied: pillow in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from image) (10.3.0)
Requirement already satisfied: django in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from image) (5.2.4)
Requirement already satisfied: six in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from image) (1.16.0)
Requirement already satisfied: asgiref>=3.8.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from django->image) (3.9.1)
Requirement already satisfied: sqlparse>=0.3.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from django->image) (0.5.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: ibm_watsonx_ai>=1.1.14 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (1.3.32)
Requirement already satisfied: requests in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (2.32.4)
Requirement already satisfied: httpx<0.29,>=0.27 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (0.28.1)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (1.26.19)
Requirement already satisfied: pandas<2.3.0,>=0.24.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (2.1.4)
Requirement already satisfied: certifi in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (2025.6.15)
Requirement already satisfied: lomond in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (0.8.10)
Requirement already satisfied: packaging in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (23.2)
Requirement already satisfied: ibm-cos-sdk<2.15.0,>=2.12.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (2.14.2)
Requirement already satisfied: cachetools in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm_watsonx_ai>=1.1.14) (5.3.3)
Requirement already satisfied: anyio in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (4.7.0)
Requirement already satisfied: httpcore==1.* in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (1.0.9)
Requirement already satisfied: idna in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (3.7)
Requirement already satisfied: h11>=0.16 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from httpcore==1.*->httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (0.16.0)
Requirement already satisfied: ibm-cos-sdk-core==2.14.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm-cos-sdk<2.15.0,>=2.12.0->ibm_watsonx_ai>=1.1.14) (2.14.2)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.14.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm-cos-sdk<2.15.0,>=2.12.0->ibm_watsonx_ai>=1.1.14) (2.14.2)
Requirement already satisfied: jmespath<=1.0.1,>=0.10.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm-cos-sdk<2.15.0,>=2.12.0->ibm_watsonx_ai>=1.1.14) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.9.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from ibm-cos-sdk-core==2.14.2->ibm-cos-sdk<2.15.0,>=2.12.0->ibm_watsonx_ai>=1.1.14) (2.9.0.post0)
Requirement already satisfied: numpy<2,>=1.23.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from pandas<2.3.0,>=0.24.2->ibm_watsonx_ai>=1.1.14) (1.26.4)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from pandas<2.3.0,>=0.24.2->ibm_watsonx_ai>=1.1.14) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from pandas<2.3.0,>=0.24.2->ibm_watsonx_ai>=1.1.14) (2023.3)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from requests->ibm_watsonx_ai>=1.1.14) (2.0.4)
Requirement already satisfied: six>=1.10.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from lomond->ibm_watsonx_ai>=1.1.14) (1.16.0)
Requirement already satisfied: sniffio>=1.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from anyio->httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (1.3.0)
Requirement already satisfied: typing_extensions>=4.5 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from anyio->httpx<0.29,>=0.27->ibm_watsonx_ai>=1.1.14) (4.12.2)
Note: you may need to restart the kernel to use updated packages.
```

edunet foundation

RESULT

IBM watsonx.ai Studio

Projects / DM_Agent / DM_Environment

File Edit View Run Kernel Help

Trusted Memory:847 / 8192 MB

Python 3.11

Import Libraries

[21]:
```python
import requests
import base64
import textwrap
from PIL import Image
from ibm_watsonx_ai import Credentials
from ibm_watsonx_ai.foundation_models import ModelInference
```

Input your `WATSONX_EU_APIKEY` and `WATSONX_EU_PROJECT_ID` that you created in steps 1 and 2 upon running the following cell. We will also set the URL serving as the API endpoint.

Set Up Credentials

[22]:
```python
WATSONX_EU_APIKEY = "G68ZGtAh61SLOc89XQDvxZBBeq961RSZ8AiSDzn9CPRz"
WATSONX_EU_PROJECT_ID = "33186dcf-12de-435b-a117-fc6fa1beefb1"
URL = "https://au-syd.ml.cloud.ibm.com"

credentials = Credentials(
    url=URL,
    api_key=WATSONX_EU_APIKEY
)
```

We can use the `Credentials` class to encapsulate our passed credentials.

Load & Encode Images (from URLs)

[23]:
```python
url_image_1 = 'https://hsc.unm.edu/medicine/departments/dermatology/_images/skin-atlas/acne/acne-type-iv.jpg'
url_image_2 = 'https://hsc.unm.edu/medicine/departments/dermatology/_images/skin-atlas/acne/acne-type-i.jpg'
image_urls = [url_image_1, url_image_2]

# Encode images in base64
encoded_images = []
for url in image_urls:
    img_data = requests.get(url).content
    encoded = base64.b64encode(img_data).decode("utf-8")
    encoded_images.append(encoded)
```

RESULT

IBM watsonx.ai Studio

Projects / DM_Agent / DM_Environment

File Edit View Run Kernel Help

Trusted Memory:847 / 8192 MB

Markdown

Python 3.11

Define Function to Prepare API Request

[24]:
```
def augment_api_request_body(user_query, image):
    return [{
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "You are a helpful assistant. Answer the following user query in 1 or 2 sentences: " + user_query
            },
            {
                "type": "image_url",
                "image_url": {
                    "url": f"{image}"
                }
            }
        ]
    }]
```

[25]:
```
for idx, url in enumerate(image_urls):
    print(f'url_image_{idx}')
    display(Image.open(requests.get(url, stream=True).raw))
```

```
url_image_0
```

edunet foundation

RESULT

IBM watsonx.ai Studio

Projects / DM_Agent / DM_Environment

File Edit View Run Kernel Help

Trusted Memory:847 / 8192 MB

Markdown

Python 3.11

url_image_1

Load Supported Models

[26]:
```
# Text model (replaces Pixtral 12B)
model = ModelInference(
    model_id="ibm/granite-3-8b-instruct",  # ✅ Supported text model
    credentials=credentials,
    project_id=WATSONX_EU_PROJECT_ID,
    params={"max_tokens": 200}
)

# Vision + text model (for multimodal input)
model1 = ModelInference(
    model_id="meta-llama/llama-3-2-11b-vision-instruct",
    credentials=credentials,
    project_id=WATSONX_EU_PROJECT_ID,
    params={"max_tokens": 200}
)
```

edunet foundation

RESULT

IBM watsonx.ai Studio

Projects / DM_Agent / DM_Environment

File Edit View Run Kernel Help

Trusted Memory:847 / 8192 MB

Python 3.11

```
    params={"max_tokens": 200}
)
```

Run Model on Each Image

[28]:
```python
from ibm_watsonx_ai.foundation_models import ModelInference

model1 = ModelInference(
    model_id="meta-llama/llama-3-2-11b-vision-instruct",  # ✅ Supports image + text
    credentials=credentials,
    project_id=WATSONX_EU_PROJECT_ID,
    params={"max_tokens": 200}
)

# Now loop through images and ask query
for i in range(len(encoded_images)):
    image = encoded_images[i]
    user_query = "What kind of disease this person is having?"
    messages = augment_api_request_body(user_query, image)

    response1 = model1.chat(messages=messages)

    print("<== Response by LLaMA 11B Vision-Instruct model ==>")
    print(textwrap.fill(response1['choices'][0]['message']['content'], width=100), "\n")
```

```
<== Response by LLaMA 11B Vision-Instruct model ==>
The person in the image appears to have a skin condition characterized by multiple small, red, and
inflamed bumps or lesions. Without further information or a medical diagnosis, it is difficult to
determine the exact nature of the disease, but it could be acne, eczema, or another type of skin
condition.

<== Response by LLaMA 11B Vision-Instruct model ==>
The person in the image appears to be experiencing a skin condition characterized by small, red
bumps or pimples, which is commonly known as acne. Acne is a common skin condition that occurs when
the pores on the skin become clogged with oil and dead skin cells, leading to inflammation and
breakouts.
```

CONCLUSION

- The AI agent enables early detection and personalized intervention for chronic disease patients.
- Real-time monitoring and reminders support better adherence and reduce the need for emergency care.
- Integration with wearables and electronic health data improves patient–provider collaboration.
- Leveraging IBM technology (watsonx.ai + Granity) provides scalable, secure, low-cost infrastructure.

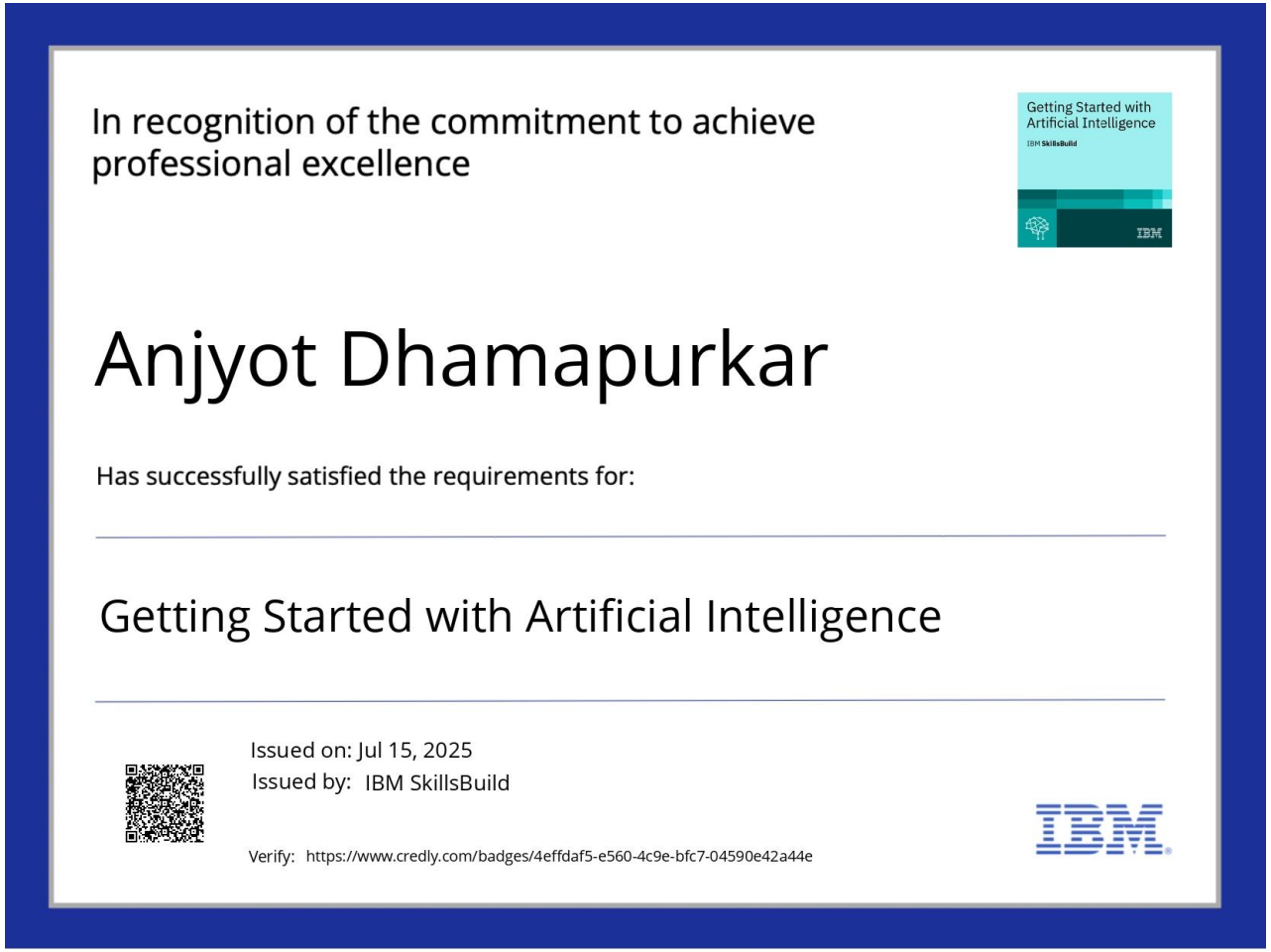
FUTURE SCOPE

- Expand to other chronic conditions (e.g. COPD, chronic kidney disease)
- Fine-tune or train custom predictive models tailored to patient demographic clusters
- Integrate with telehealth platforms and mobile apps for real-time feedback
- Incorporate more advanced wearables (ECG patches, continuous glucose monitors)
- Add edge computing for privacy-preserving, offline data capture

REFERENCES

- Cite relevant research papers on AI in chronic disease monitoring
- Documentation pages for IBM watsonx.ai Studio and Granity
- Medical guidelines and sources you used (e.g. diabetes management protocols)

IBM CERTIFICATIONS



In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence
IBM SkillsBuild

Anjyot Dhamapurkar

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 15, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/4effdaf5-e560-4c9e-bfc7-04590e42a44e

IBM

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution
IBM SkillsBuild

Anjyot Dhamapurkar

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 17, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/6ebce498-fb07-4ba2-b5b2-9ac0b45c22d9

IBM

IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Anjyot Dhamapurkar

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 23 Jul 2025 (GMT) **Learning hours:** 20 mins

edunet foundation

THANK YOU