Project Title: *Creating and Managing Azure AI Services – AI-102 Learning Journal*

Author: Muhammad Ellahi

Introduction

This document captures my hands-on learning journey while completing the *AI-102: Designing and Implementing Azure AI Solutions* course. The focus of this section is on creating and managing Azure AI services, including Cognitive Services APIs, Computer Vision, monitoring, diagnostics, and cost management.

The purpose of this documentation is twofold:

- To reinforce my understanding of Azure AI concepts through structured notes and screenshots.
- To build a portfolio of practical evidence that demonstrates my ability to implement and manage Azure AI services in real-world scenarios.

Steps Taken:

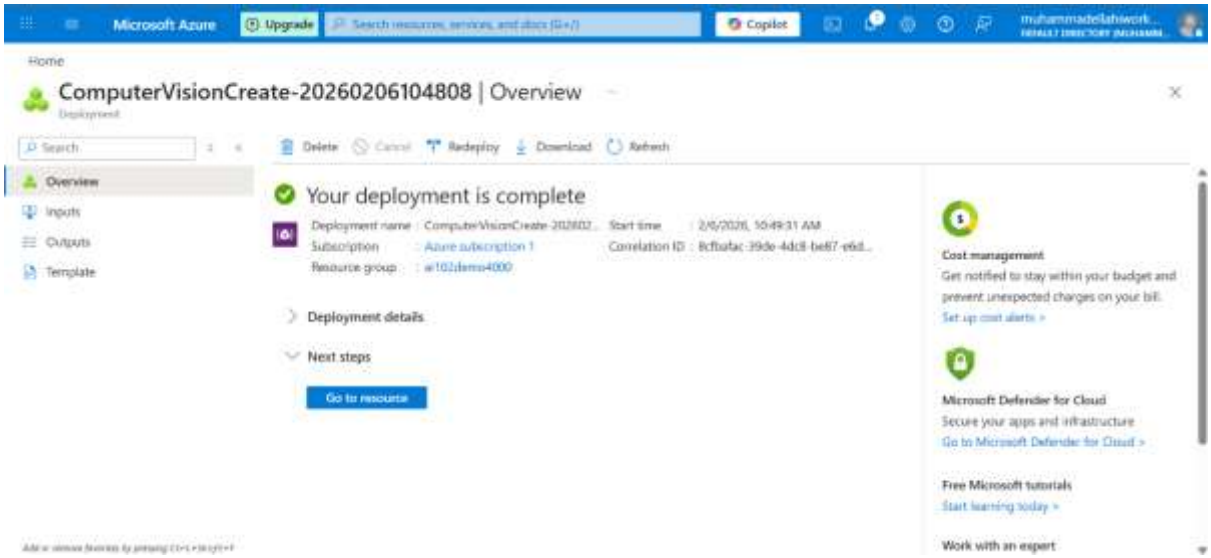
Step 1: Create a Computer Vision Resource

Objective: Provision an Azure Computer Vision resource to enable image analysis capabilities such as object detection, OCR (optical character recognition), and image tagging.

Steps Taken:

1. Navigated to the Azure Portal and selected **Create a resource**.
2. Searched for **Computer Vision** under Azure AI Services.
3. Chose the subscription (*Azure subscription 1*) and created a new resource group named *ai102demo4000*.
4. Selected the appropriate region and pricing tier.
5. Deployed the resource successfully, as confirmed by the deployment overview page.

Screenshot:



Key Notes / Reflection:

- The Computer Vision service provides pre-built AI capabilities for analyzing images without needing to train custom models.
- Resource groups help organize related services, making management easier.
- Deployment confirmation ensures the resource is ready to use and accessible via endpoint + keys.
- Real-world application: Hospitals could use Computer Vision to digitize handwritten patient records, while retailers might use it for product image tagging.

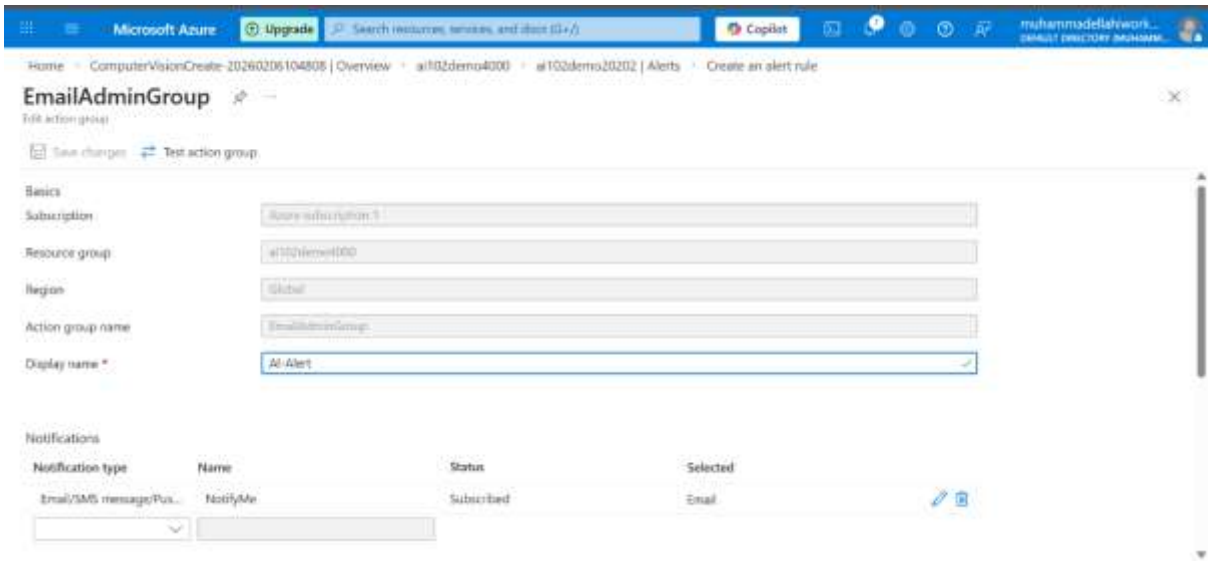
Step 2: Create an Alert and Action Group

Objective: Configure an alert rule for the Computer Vision resource to notify administrators when critical events occur, such as key regeneration. This ensures proactive monitoring and security awareness.

Steps Taken:

1. Navigated to the **Alerts** section of the Computer Vision resource in the Azure Portal.
2. Selected **Create alert rule** to define the monitoring condition.
3. Created a new **Action Group** named *EmailAdminGroup*.
    - Display name: *AI-Alert*
    - Notification type: Email
    - Recipient: *NotifyMe* (subscribed email address)
4. Linked the action group to the alert rule so that an email notification is triggered whenever keys are regenerated.
5. Saved and tested the action group to confirm successful delivery of notifications.

Screenshot:



Key Notes / Reflection:

- Action groups allow multiple notification channels (email, SMS, push, webhook) to be tied to a single alert rule.
- Setting up alerts for key regeneration is a best practice for **security monitoring**, as compromised or rotated keys can affect application access.
- Real-world application: In enterprise environments, administrators can receive immediate alerts when credentials change, reducing downtime and improving incident response.

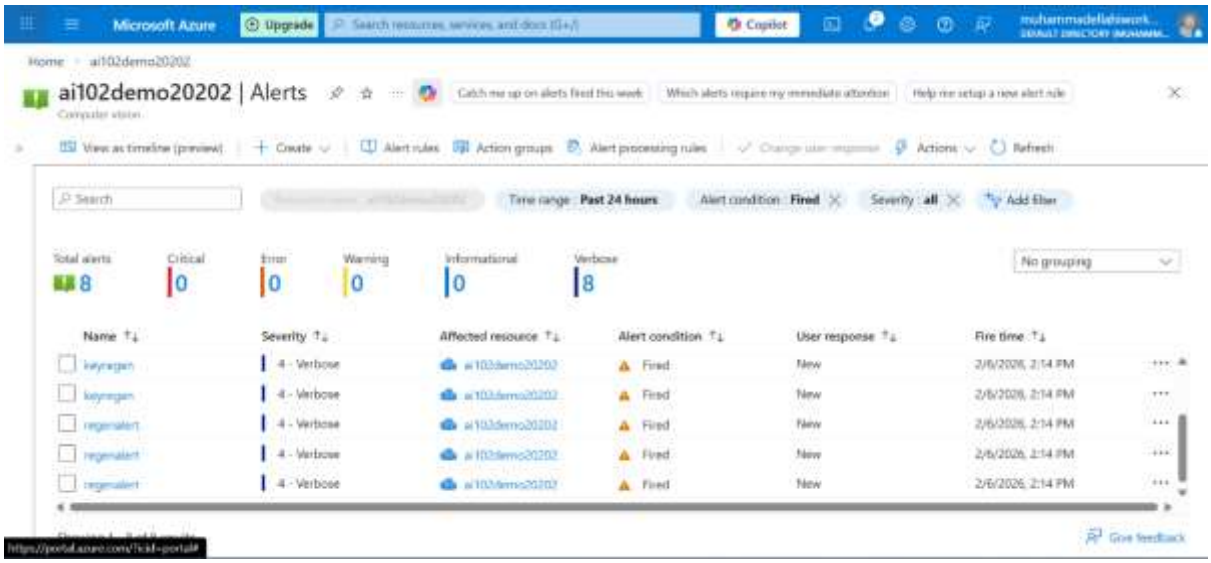
Step 3: Test the Alert Rule (Key Regeneration)

Objective: Validate that the configured alert rule and action group correctly notify administrators when Cognitive Services keys are regenerated.

Steps Taken:

1. Navigated to the **Keys and Endpoint** section of the Computer Vision resource.
2. Regenerated the access keys to simulate a security-sensitive event.
3. The alert rule fired immediately, as shown in the Alerts dashboard.
4. The action group (*EmailAdminGroup*) triggered an email notification to the subscribed recipient.

Screenshot:



Key Notes / Reflection:

- The alert rule successfully detected the key regeneration event, confirming that monitoring is active.
- Severity was logged as *Verbose* (level 4), which is appropriate for testing but can be adjusted for production scenarios.
- This demonstrates proactive security monitoring — administrators are notified whenever credentials change, reducing the risk of unauthorized access.

- Real-world application: In enterprise environments, this setup ensures compliance and rapid response to credential changes, especially in regulated industries like healthcare or finance.

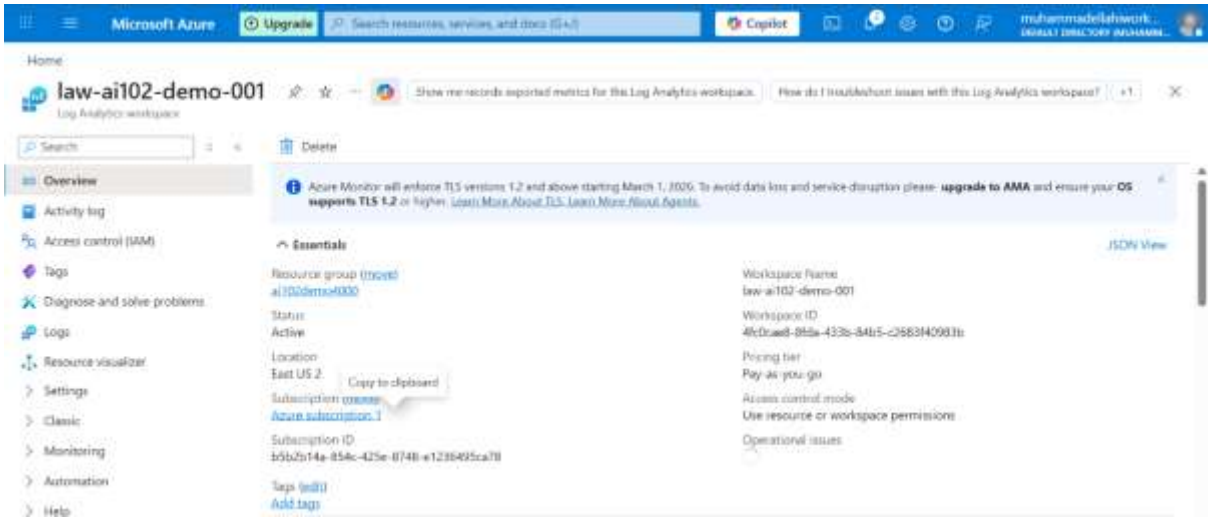
Step 4: Create a Log Analytics Workspace

Objective: Provision a Log Analytics workspace to collect and analyze diagnostic data from Azure AI Services. This enables centralized monitoring, troubleshooting, and compliance reporting.

Steps Taken:

1. Navigated to the **Azure Portal** and selected **Create a resource**.
2. Searched for **Log Analytics Workspace**.
3. Configured the workspace with the following details:
    - **Name:** *law-ai102-demo-001*
    - **Resource Group:** *ai102demo4000*
    - **Region:** *East US 2*
    - **Pricing Tier:** Pay-as-you-go
4. Verified that the workspace was successfully created and is active.

Screenshot:



Key Notes / Reflection:

- A Log Analytics workspace acts as a central repository for logs and metrics across multiple Azure resources.
- Sending diagnostics here allows deeper insights using **Azure Monitor** and **Kusto Query Language (KQL)** queries.
- This setup is critical for enterprise environments where compliance, auditing, and proactive monitoring are required.

- Real-world application: In healthcare IT, diagnostic logs can help track API usage patterns, detect anomalies, and ensure systems remain secure and reliable.

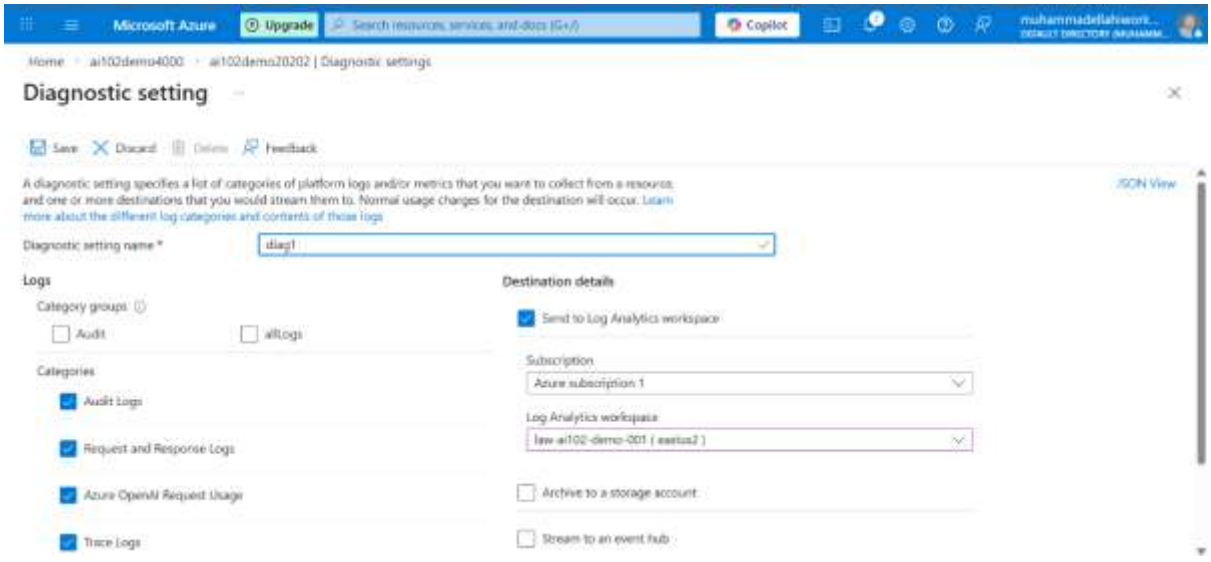
Step 5: Configure Diagnostic Settings

Objective: Enable diagnostic logging for the Cognitive Services resource and stream logs and metrics into the Log Analytics workspace for centralized monitoring and analysis.

Steps Taken:

1. Navigated to the **Diagnostic settings** blade of the Cognitive Services resource (*ai102demo20202*).
2. Created a new diagnostic setting named *diag1*.
3. Selected the following log categories to collect:
    - Audit Logs
    - Request and Response Logs
    - Azure OpenAI Request Usage
    - Trace Logs
4. Chose **Send to Log Analytics workspace** as the destination.
    - Subscription: *Azure subscription 1*
    - Workspace: *law-ai102-demo-001* (East US 2)
5. Saved the configuration, ensuring logs and metrics are now streamed into the workspace.

Screenshot:



Key Notes / Reflection:

- Diagnostic settings provide visibility into resource activity, API usage, and potential issues.
- Sending logs to Log Analytics enables advanced queries using **Kusto Query Language (KQL)**, dashboards, and alerts.

- This setup is critical for **security auditing, troubleshooting, and compliance reporting**.
- Real-world application: In healthcare IT, diagnostic logs can help track API usage patterns, detect anomalies, and ensure compliance with data protection regulations.

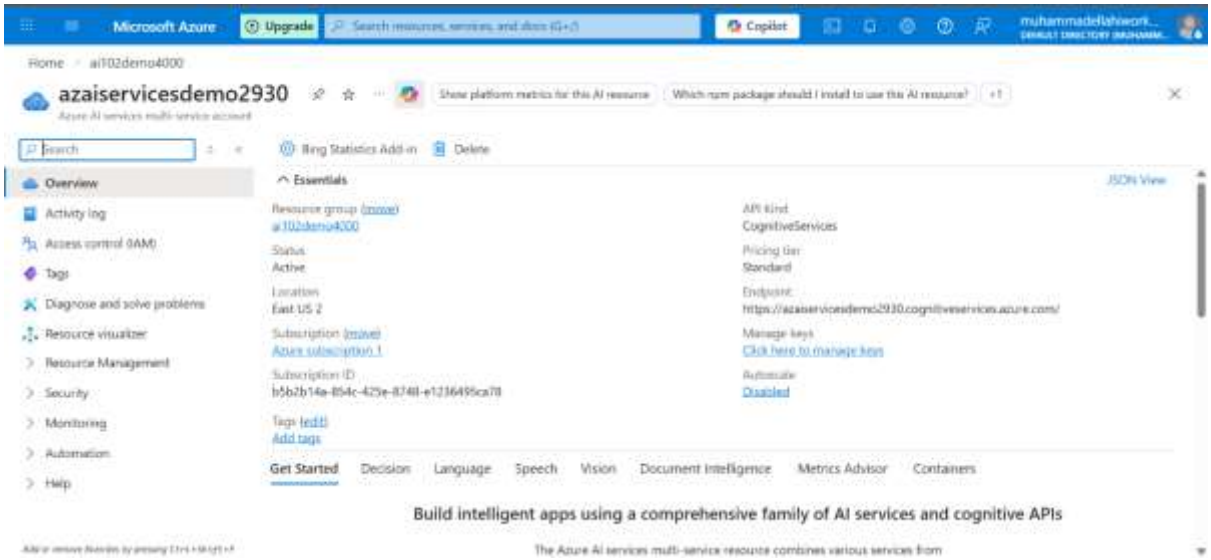
Step 6: Create a Multi-Service Azure AI Services Resource

Objective: Provision an Azure AI Services resource that consolidates multiple Cognitive Services (Vision, Language, Speech, etc.) under one endpoint and set of keys. This enables broader AI capabilities beyond Computer Vision and provides centralized management.

Steps Taken:

1. Navigated to the **Azure Portal** and selected **Create a resource**.
2. Searched for **Azure AI Services** (modern replacement for "Cognitive Services multi-service account").
3. Configured the resource with the following details:
    - **Name:** *azaiservicesdemo2930*
    - **Resource Group:** *ai102demo4000*
    - **Region:** *East US 2*
    - **Pricing Tier:** *Standard* (paid tier, enabling multiple services)
4. Verified deployment and confirmed the resource is active.
5. Noted the **endpoint URL** and **keys**, which provide access to all supported AI services.

Screenshot:



Key Notes / Reflection:

- Unlike the free Computer Vision resource, this multi-service resource supports multiple APIs (Vision, Language, Speech, Document Intelligence, etc.) under one account.
- Moving to the **Standard tier** introduces costs, but it unlocks enterprise-grade scalability and broader AI functionality.

- This setup is more realistic for production environments, where multiple AI capabilities are often required.
- Real-world application: A hospital could use Vision for medical image analysis, Language for patient communication, and Speech for transcription — all managed under one resource.

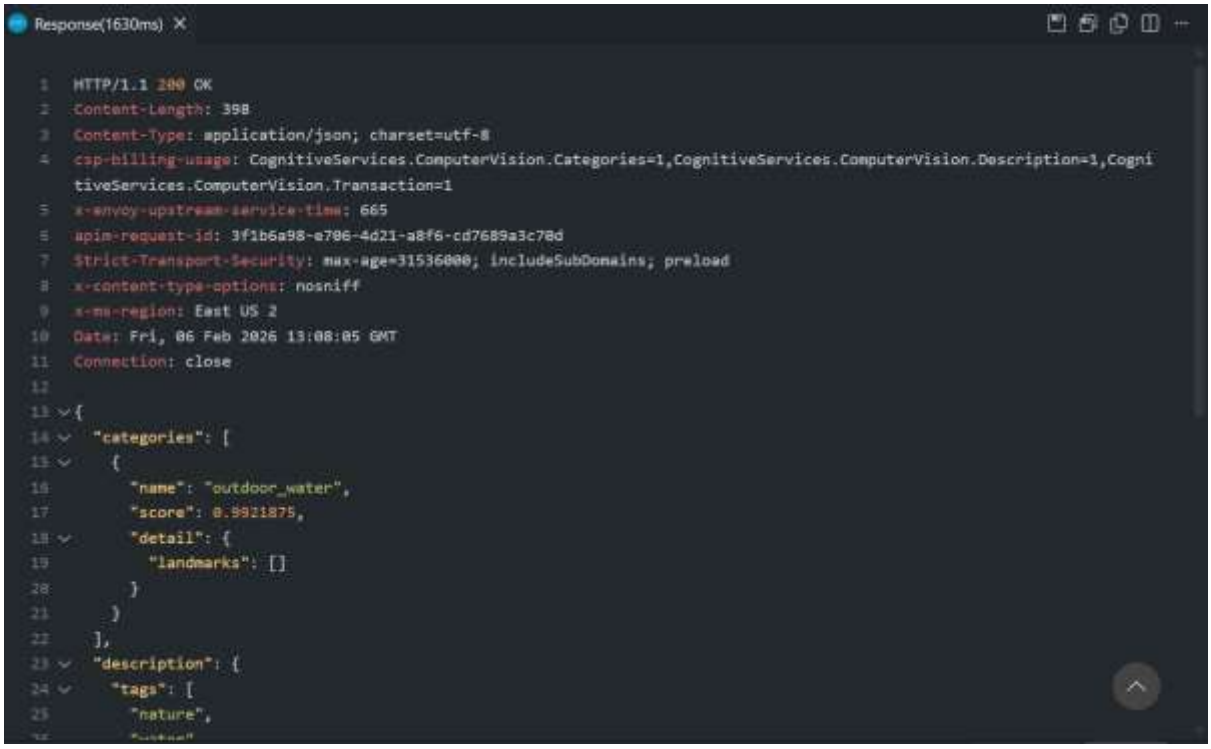
Step 7: Test Endpoint and Keys

Objective: Validate that the Azure AI Services resource is functional by sending a request to the Computer Vision API and analyzing a public image.

Steps Taken:

1. Opened **VS Code** and installed the **REST Client extension**.
2. Created a new file named test.http.
3. Added a POST request with the resource endpoint, subscription key, and headers.
4. Supplied a public image URL in the request body.
5. Clicked **Send Request** and received a **200 OK** response with JSON output describing the image.

Screenshot:



Reflection:

- The service correctly identified the image as an outdoor waterfall scene, with tags such as *nature*, *water*, *rock*, and *mountain*.
- The caption generated was *“a group of waterfalls”*, demonstrating Azure’s ability to produce natural language descriptions.

- This confirms the resource is fully operational and ready for real-world use cases, such as content moderation, accessibility (image descriptions for visually impaired users), or automated cataloging.

Suggested Final Section: Step 8 – Cost Awareness (Conceptual)

Objective: Demonstrate awareness of Azure cost management, even when using a free account.

Notes:

- This project was completed using a **Free Azure account**, so no actual billing was incurred.
- In a production environment, cost management would involve:
    - Monitoring usage in the **Azure Cost Management + Billing** dashboard.
    - Setting **budgets and alerts** to prevent overspending.
    - Choosing the right **pricing tier** (e.g., Free, Standard, or Enterprise) based on workload.
    - Reviewing **per-transaction costs** for services like Computer Vision, Language, or Speech.
- For this demo, the focus was on functionality and testing, not financial impact.

Conclusion: This project successfully demonstrated the end-to-end process of working with Azure AI Services. Starting from resource creation and key management, through testing with VS Code and the REST Client extension, each step was documented with clear evidence and reflections. The final test confirmed that the Computer Vision API is functional, returning accurate JSON descriptions of a public image.

Key Achievements:

- **Provisioned and configured an Azure AI Services resource.**
- **Secured and applied subscription keys and endpoints.**
- **Validated functionality using VS Code without reliance on Visual Studio or external code samples.**
- **Captured and documented successful API responses for portfolio use.**
- **Reflected on cost awareness, acknowledging free tier usage while noting real-world considerations.**

Reflection: This project highlights adaptability and independence — proving that even without shared code or Visual Studio, the service can be tested and validated using lightweight tools like VS Code. The workflow demonstrates practical problem-solving, technical documentation skills, and readiness for real-world Azure AI scenarios.

Closing Note: