**SKIPS University**

**School of Computer Science**

****

**MSc(Information Technology) - Batch 2024- 26**

**SOCS010202**

**Cloud Architecture Servicess**

**Project Title: Object Recognition using Azure Blob Storage and Custom Vision**

**Date**:  
16 June 2025

**Submitted By**: **Submitted To**:

2426047 Yash Khatod  
Vrut Talsaniya Faculty

Object Recognition using Azure Blob Storage and Custom Vision

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Content** | **Page No.** |
| **1.** | **Introduction** | **3** |
| **2.** | **Objective** | **4** |
| **3.** | **Scope** | **5** |
| **4.** | **Technology Stack** | **7** |
| **5.** | **System Architecture** | **8** |
| **6.** | **Work flow** | **10** |
| **7.** | **Output** | **11** |
| **8.** | **Conclusion** | **17** |

# 1. INTRODUCTION

In the modern technological era, Artificial Intelligence (AI) and Cloud Computing have revolutionized the way data is processed, stored, and analyzed. The increasing availability of cloud-based AI services allows developers and organizations to build intelligent, scalable, and cost-effective solutions without the need for extensive on-premises infrastructure. Among these, Microsoft Azure offers a rich suite of tools and services that enable seamless integration of machine learning and data storage capabilities into web applications.

This project focuses on developing a practical, end-to-end object recognition system that leverages Azure's capabilities. The system is designed as a web-based application where users can upload images through a locally hosted frontend interface. Upon receiving the image, the backend component uploads it to **Azure Blob Storage**, a highly scalable and secure storage service. The image is then accessed by **Azure Custom Vision**, a specialized service within Azure’s Cognitive Services that allows developers to build, train, and publish custom image classification and object detection models.

The **Custom Vision model**, trained with a domain-specific dataset, analyzes the uploaded image and returns structured prediction results—such as identified objects, bounding boxes (if detection is used), and confidence scores. These results are then processed by the backend and visually presented on the local website, providing instant feedback to the user.

This project demonstrates a real-world application of integrating:

* A **user-facing interface** for interaction,
* **Cloud storage** for managing image data, and
* **AI services** for intelligent analysis.

It serves as a foundational use case for a variety of industries including healthcare (e.g., medical image diagnostics), retail (e.g., product recognition), agriculture (e.g., crop health monitoring), and more.

In summary, the project offers an educational yet practical example of how to build intelligent applications by combining AI and cloud services, with a focus on modular design, clean architecture, and real-time user interaction.

# 2. OBJECTIVE

The primary objective of this project is to design and implement a complete, functional web-based image recognition system that utilizes Microsoft Azure’s cloud services to analyze user-submitted images in real-time. This project aims to bridge the gap between local application development and cloud-based artificial intelligence services, offering students and developers hands-on experience with modern tools and technologies.

**The specific objectives include:**

1. **Developing a Web Application Interface:**  
   To create a user-friendly web application hosted on localhost, enabling users to easily upload images through a clean and intuitive interface. This interface acts as the primary point of interaction between the user and the system.
2. **Integrating Cloud Storage (Azure Blob Storage):**  
   To securely store uploaded images in **Azure Blob Storage**, ensuring scalability, durability, and accessibility. Blob Storage serves as a reliable medium for storing image data that can be accessed later by the AI model.
3. **Leveraging AI for Image Analysis (Azure Custom Vision):**  
   To utilize **Azure Custom Vision**, a powerful AI service that allows training of custom image classification or object detection models. The objective is to analyze the stored image using a pre-trained custom model and return accurate prediction results.
4. **Displaying AI Results to End Users:**  
   To retrieve the prediction results from Custom Vision and render them back to the user via the web interface, thereby closing the feedback loop in the workflow. This provides users with a near real-time response based on their submitted image.
5. **Showcasing Full Cloud-Integrated Workflow:**  
   To build and demonstrate the **end-to-end pipeline** of cloud-based image recognition — from image input to cloud processing and output display — emphasizing modular architecture and real-world applicability.
6. **Educational Purpose and Hands-on Learning:**  
   To serve as a practical project for learning cloud computing, REST API integration, storage handling, and AI model deployment. It gives exposure to Microsoft Azure services in a development environment, preparing users for more advanced or production-scale implementations in the future.

# 3. SCOPE

This project is designed as a **proof-of-concept** to demonstrate how cloud-based AI services can be integrated into a traditional web development environment. The focus is on enabling learners and developers to understand the **end-to-end workflow** of image processing using **Azure Blob Storage** and **Azure Custom Vision** while maintaining a simple, manageable system architecture.

**Primary Scope**

1. **Educational and Learning Objectives:**
   * The project is tailored for **students, hobbyists, or developers** who are beginning their journey in **cloud computing** and **AI-powered applications**.
   * It provides hands-on exposure to configuring cloud services, training AI models, using REST APIs, and working with cloud storage.
   * It also helps build understanding of deploying real-time data pipelines using local web apps.
2. **System Integration:**
   * The system showcases how a **local web application** can communicate with remote cloud services.
   * It demonstrates the use of **Azure Blob Storage** for hosting image data and **Azure Custom Vision** for performing image analysis.
   * By doing so, it reflects a real-world cloud computing use case in a simplified, accessible format.
3. **Foundation for Scalable Applications:**
   * Though designed for demonstration purposes, the architecture can be extended to create large-scale systems such as:
     + **Surveillance and security solutions** (e.g., identifying intruders or anomalies in camera feeds),
     + **Retail and e-commerce platforms** (e.g., auto-tagging product photos),
     + **Agriculture or healthcare systems** (e.g., plant disease detection or medical image diagnostics),
     + **Quality inspection in manufacturing** (e.g., detecting defects in product images).

**Limitations**

Despite its functional design, the project has the following limitations:

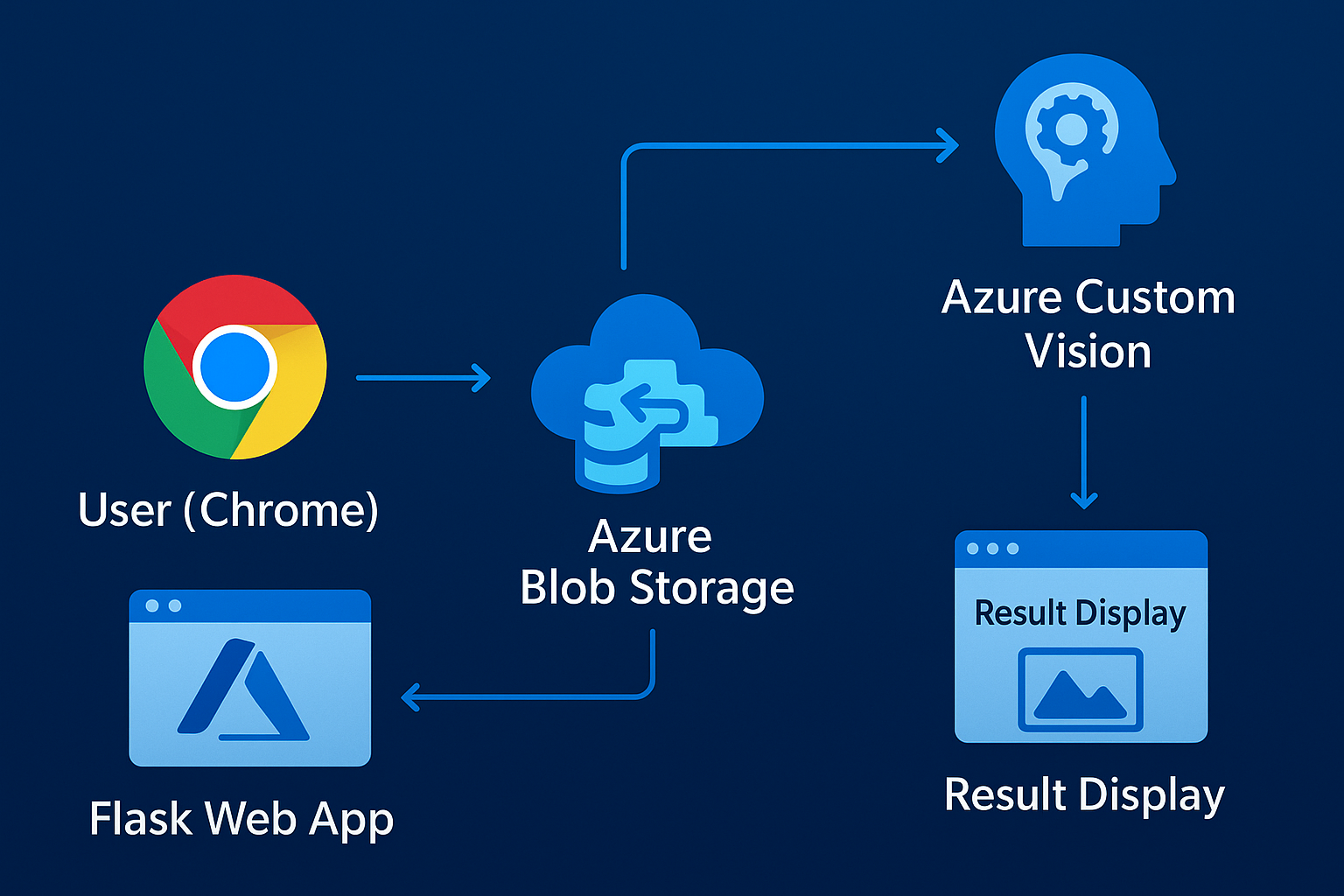
1. **Localhost Hosting Constraints:**
   * The web application runs locally on the developer's machine and is **not publicly accessible** unless deployed to a hosting platform like **Azure App Service**, **Heroku**, or **Netlify**.
   * This limits its usability for real-world, multi-user scenarios until further deployment steps are taken.
2. **Model Performance and Dataset Limitations:**
   * The **accuracy and reliability** of the object recognition depend entirely on the **quality, quantity, and diversity of the training dataset** used in Azure Custom Vision.
   * In most educational setups, only a limited dataset is used, which may not generalize well to real-world images.
3. **Security and Authentication:**
   * Basic implementations might not include security features like **authentication**, **authorization**, or **secure data transmission**, which are essential in production systems.
4. **Scalability and Performance:**
   * The project is not optimized for **high-volume usage** or **enterprise-scale performance**. It is ideal for **demo and testing purposes** only.

# 4. TECHNOLOGY STACK

This project utilizes a combination of frontend, backend, cloud storage, and AI technologies to build a robust and educational image recognition system. Each component in the stack plays a crucial role in enabling the end-to-end workflow, from user interaction to cloud-based image analysis and result display.

|  |  |  |
| --- | --- | --- |
| **Component** | **Technology/Tool** | **Detailed Description** |
| **Frontend** | HTML, CSS, JavaScript | The frontend is built using standard web technologies. **HTML** structures the web page, **CSS** styles the layout for usability, and **JavaScript** enables interactive behavior like image previews, form validation, and triggering backend API requests. It allows the user to upload images and view the recognition results in real time. |
| **Backend** | Python (Flask) or Node.js | The backend is responsible for handling HTTP requests, managing image uploads, and interacting with Azure services. You can choose between **Flask** (a lightweight Python web framework) or **Node.js** (a JavaScript runtime). This component processes the uploaded file, sends it to Azure Blob Storage, calls the Custom Vision API, and sends the response back to the frontend. |
| **Cloud Storage** | Azure Blob Storage | **Azure Blob Storage** is used to store uploaded images securely and efficiently. It provides scalable object storage and is integrated with Azure's authentication and access controls. Once an image is uploaded, a secure URL is generated, which can be passed to other services like Custom Vision for analysis. |
| **AI Model** | Azure Custom Vision | **Azure Custom Vision** is a cloud-based machine learning service that allows users to train and deploy custom image classification or object detection models. It takes the image URL from Blob Storage and returns predictions with tags, confidence scores, and bounding boxes (for object detection). You can train models with your own dataset and refine accuracy over time. |
| **Communication** | REST APIs | Communication between the backend and Azure services is handled using **RESTful APIs**. These APIs allow programmatic access to Blob Storage and Custom Vision. The backend makes secure HTTP requests (GET/POST) to perform operations like uploading files and retrieving AI predictions. |
| **Hosting** | Localhost | The entire application is hosted locally on the developer’s machine using **localhost**. This is suitable for development, testing, and demonstration purposes. For broader accessibility or production use, it can later be deployed to a cloud platform like **Azure App Service**. |

# 5. SYSTEM ARCHITECTURE



**Step-by-Step Process**

1. **Web App on Localhost (User Input):**  
   The user interacts with a simple and intuitive web interface hosted on localhost. Through this interface, the user selects and uploads an image for object recognition.
2. **Image Upload to Azure Blob Storage:**  
   Once the user submits the image, the backend server (Flask or Node.js) captures the file and uploads it to a designated container in **Azure Blob Storage**. The uploaded file is made accessible through a **public URL**, which is required for the next processing step.
3. **Image Processing by Azure Custom Vision:**  
   The image URL is sent to the **Azure Custom Vision Prediction API**. This service uses a **pre-trained custom model** to analyze the image. It detects and classifies objects and returns the results in the form of a structured **JSON response**. The response includes:
   * Object tags (e.g., "dog", "bottle", "car"),
   * Confidence scores (probability values), and
   * Bounding box coordinates (if using object detection mode).
4. **Result Display on Web Interface:**  
   The backend parses the response received from Custom Vision and extracts the relevant data. These results — including detected tags and their respective confidence percentages — are then displayed in a readable format on the web application. The user can see immediate feedback from the AI model.

# 6. WORKFLOW

**Azure Configuration Steps**

To enable the cloud-side services, the following configuration steps are performed:

**A. Azure Blob Storage**

1. **Create a Storage Account:**
   * Go to [Azure Portal](https://portal.azure.com)
   * Create a new Storage Account with a unique name.
2. **Create a Container:**
   * Inside the storage account, create a **container** (e.g., images) with public access set to **Blob**.
3. **Generate Access Credentials:**
   * Use the **connection string** or **Shared Access Signature (SAS)** key for backend authentication.

**B. Azure Custom Vision**

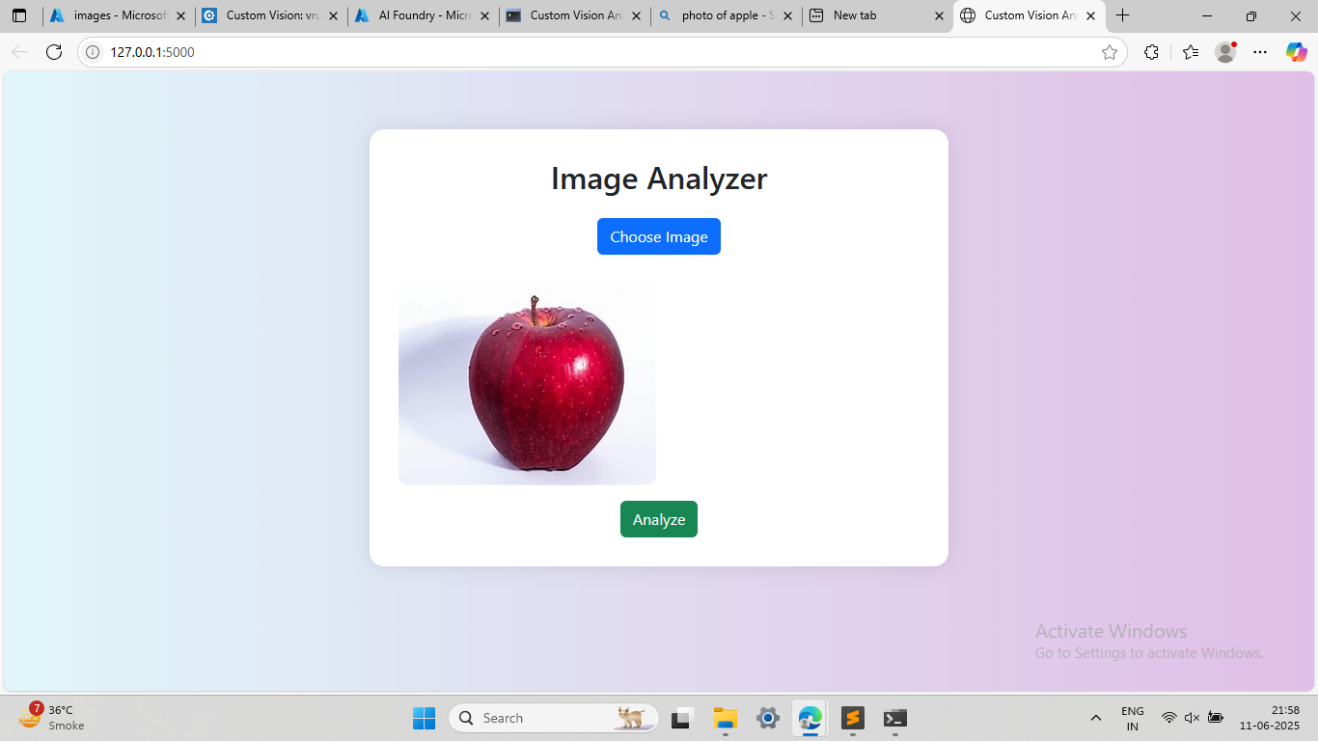
1. **Create a Custom Vision Project:**
   * Go to [Custom Vision Portal](https://www.customvision.ai)
   * Create a new **Classification** or **Object Detection** project.
2. **Upload & Tag Images:**
   * Upload training images and tag them (e.g., "apple", "tomato").
3. **Train the Model:**
   * Train the model using uploaded images. The more varied and well-labeled your dataset, the better the results.
4. **Publish an Iteration:**
   * Once the model is trained, publish the latest iteration.
   * Retrieve the **Prediction Endpoint URL** and **Prediction Key** from the **Performance tab**.

**Final Workflow Overview**

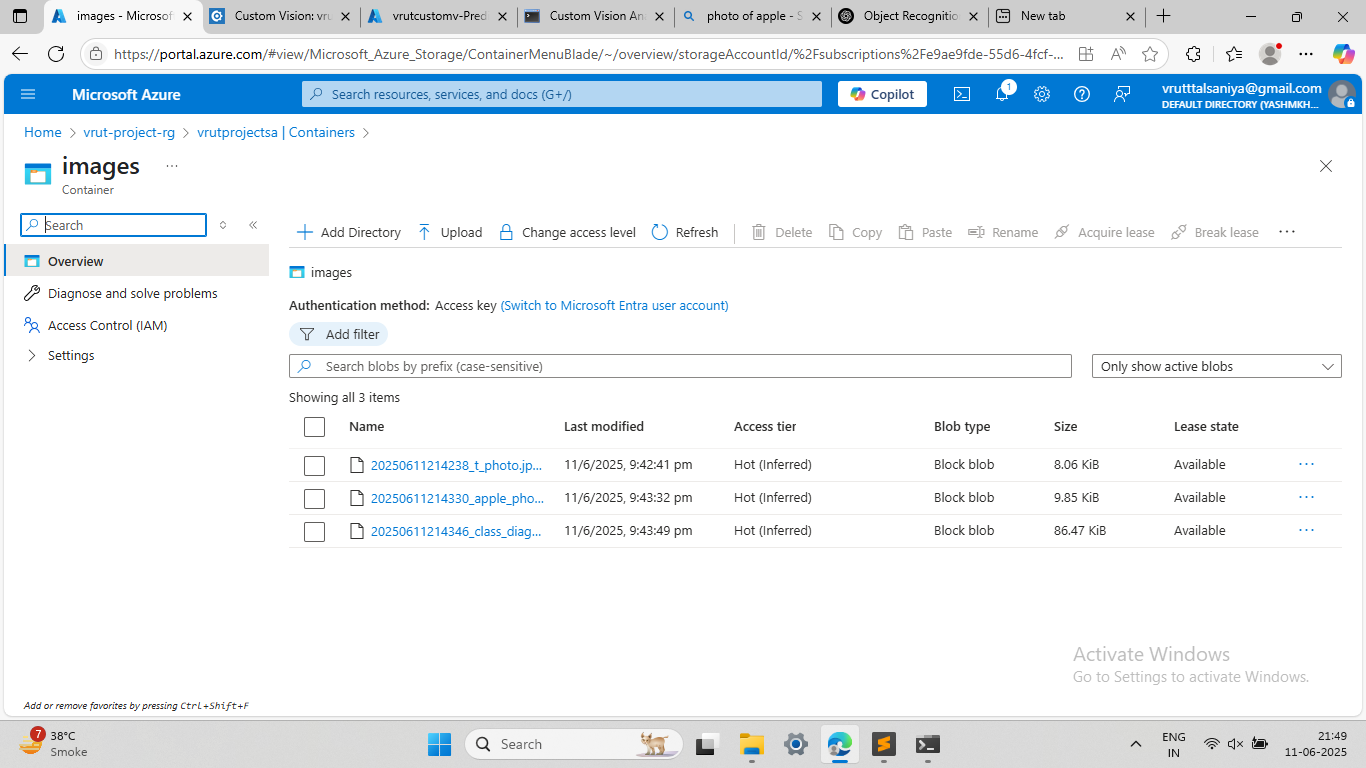
1. **User** uploads an image on the **local web app**.
2. Flask **uploads image to Azure Blob Storage** and gets the public URL.
3. Flask **calls Azure Custom Vision API** using the image URL.
4. Custom Vision returns **prediction results** (tags).
5. Flask **displays results** to the user on the web page.

# 7. OUTPUTS (SCREENSHOTS)

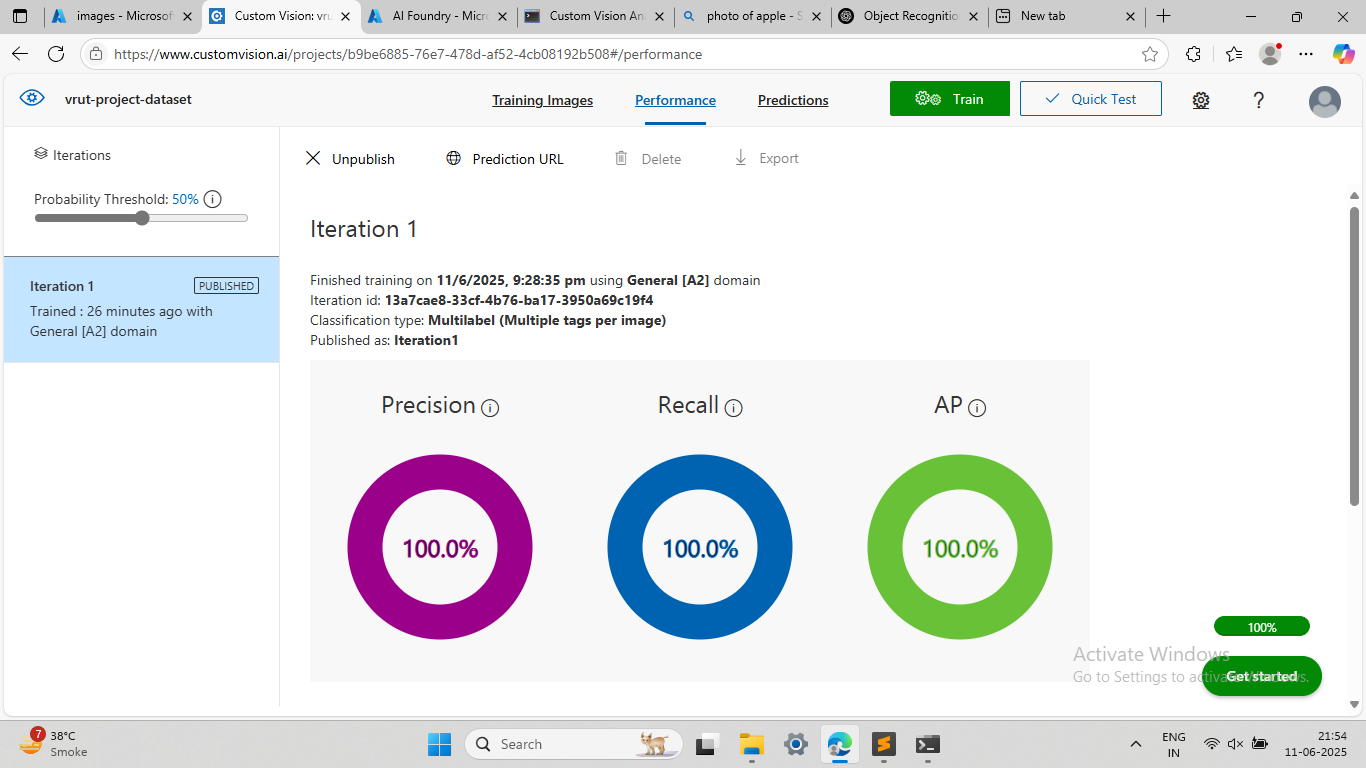
Add screenshots of:  
- Web Interface (upload form)



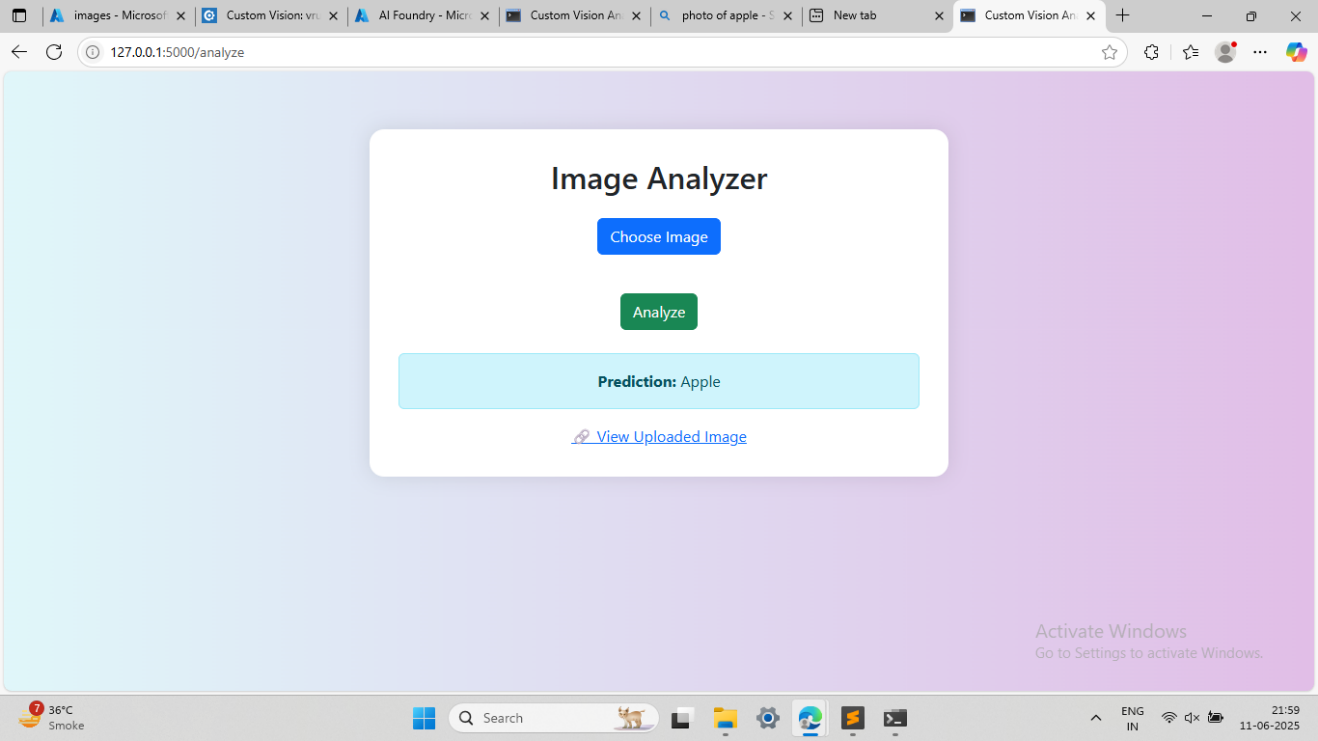
- Uploaded image in Azure Blob container



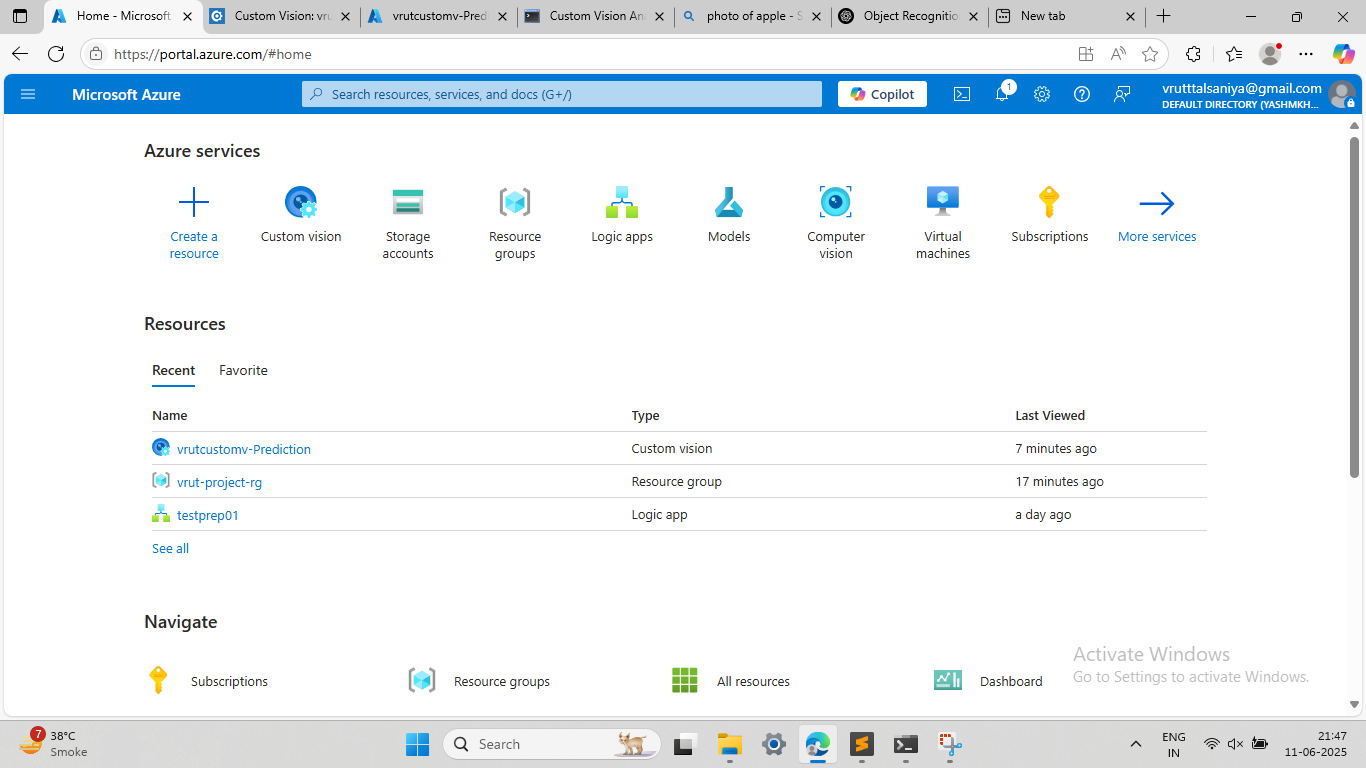
- Azure Custom Vision Portal showing predictions



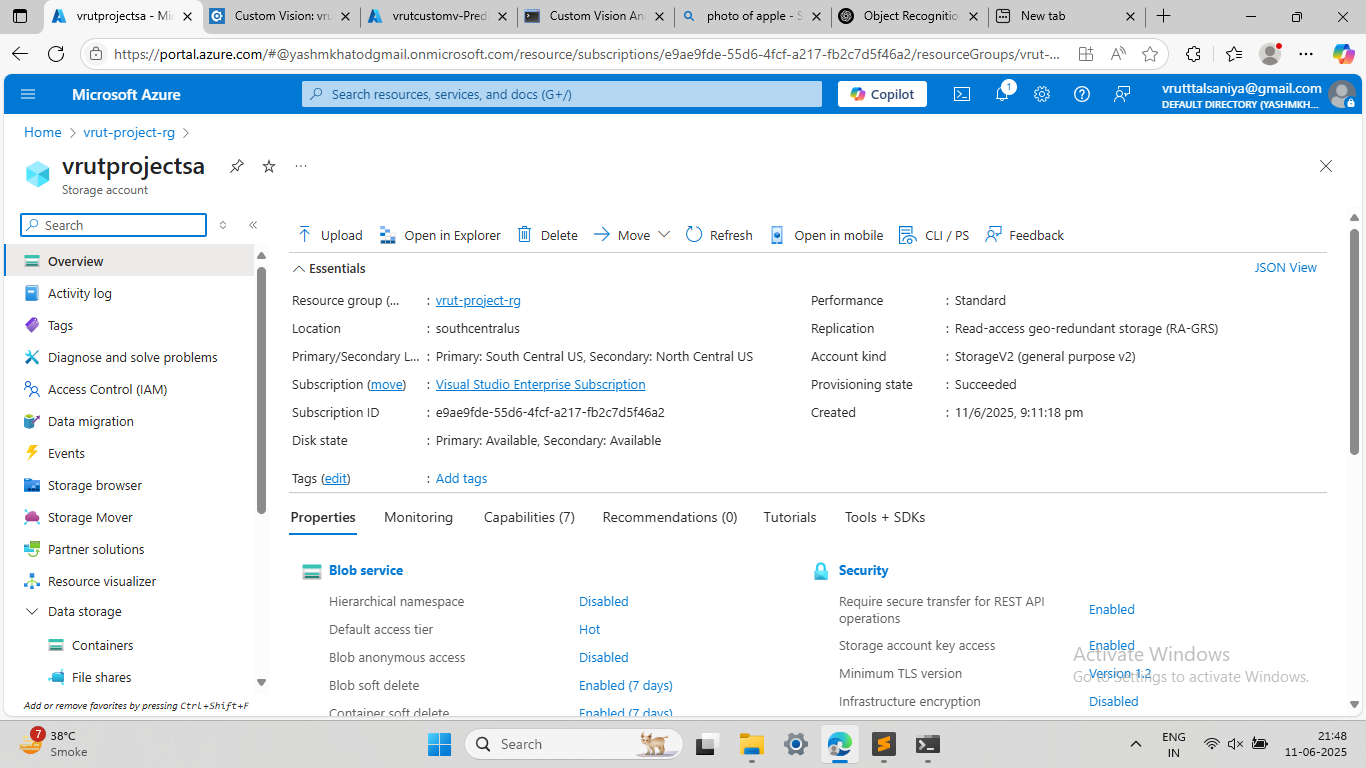
- Result displayed on the local web app



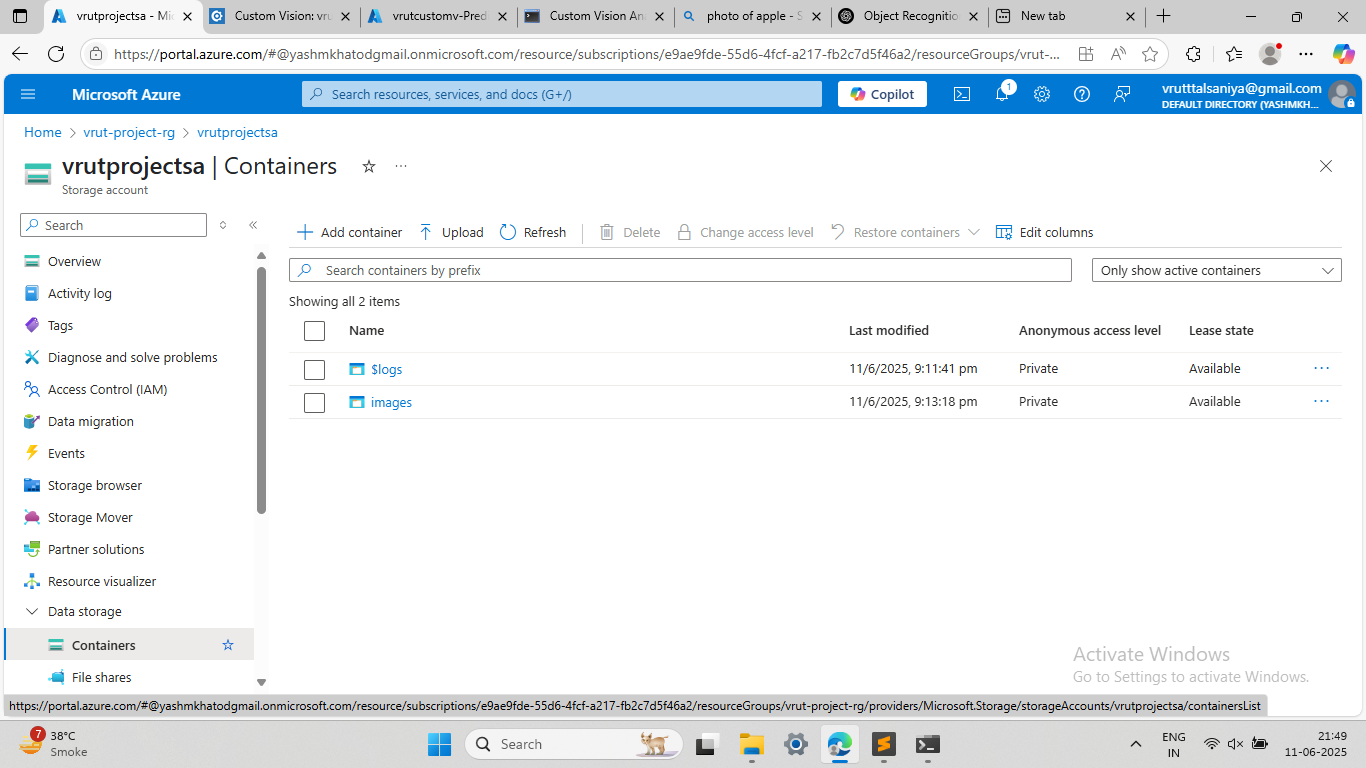
Services Used



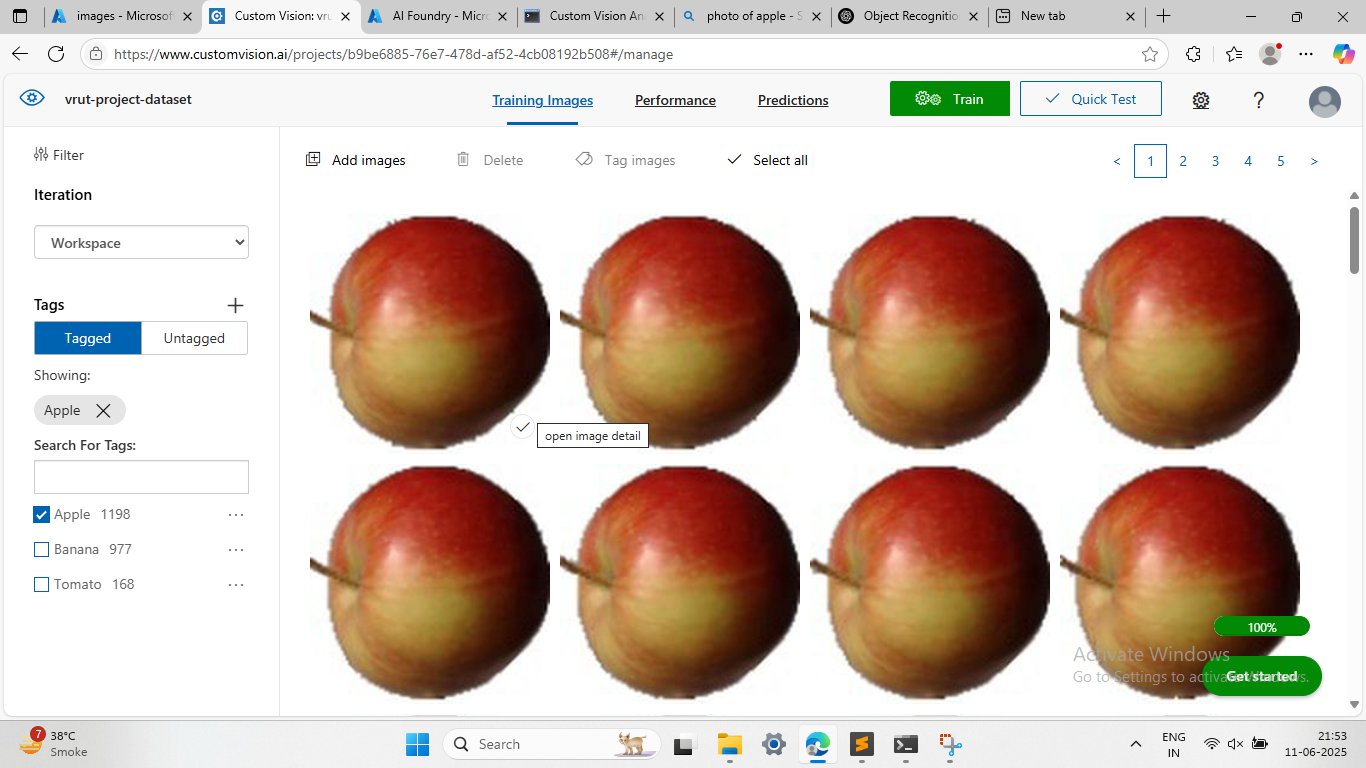
Configuration of Storage Account



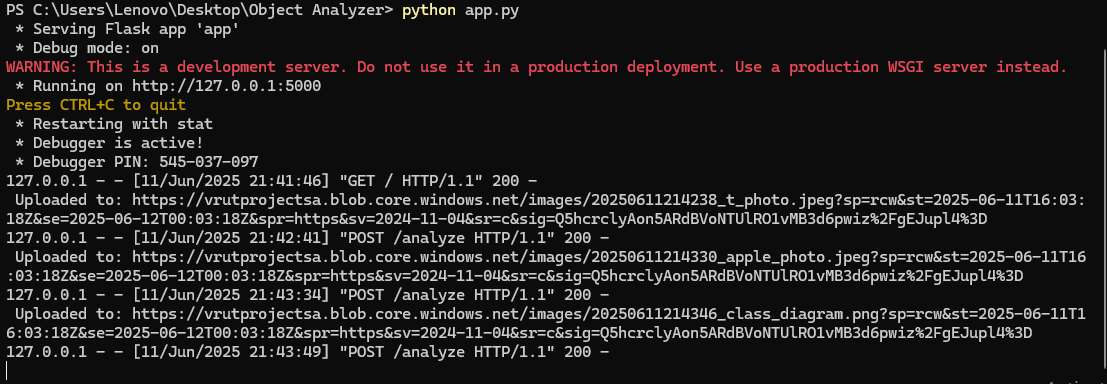
Containers:-



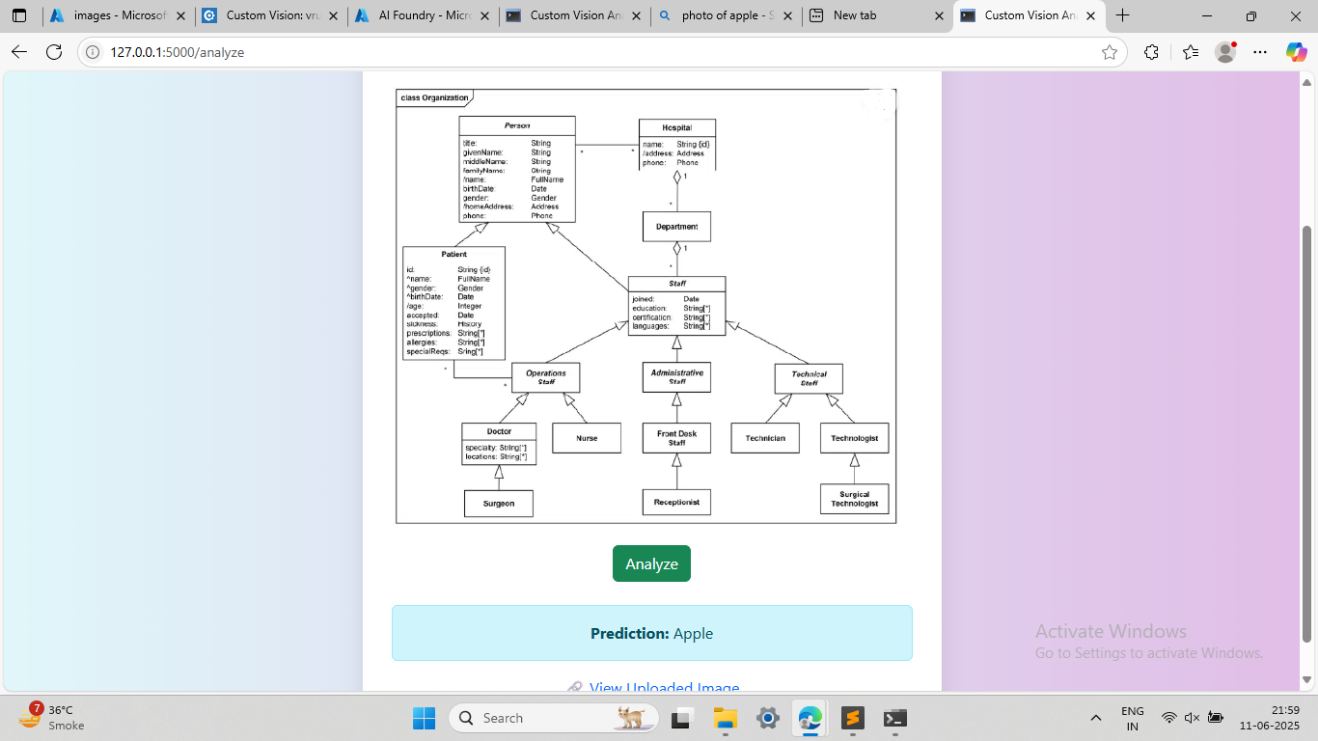
Trained model

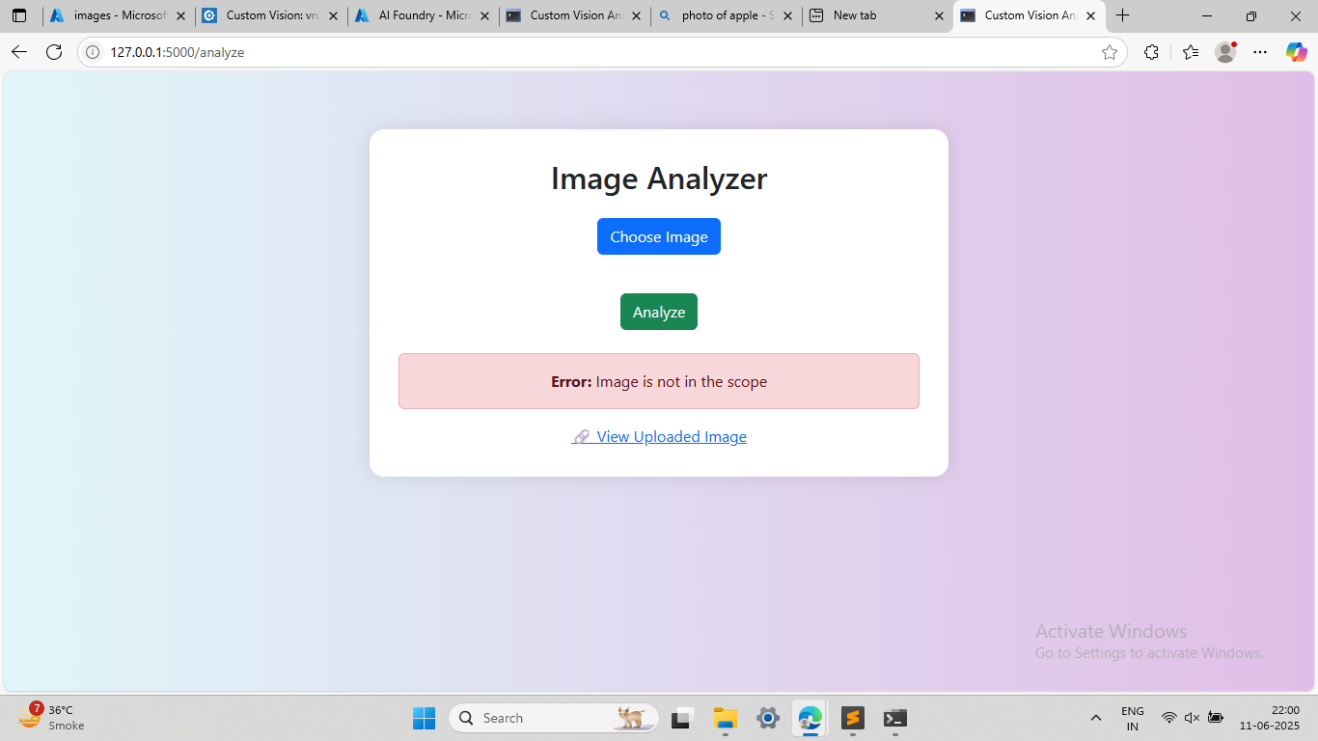


Commands To run:-



If user will Upload Wrong image





# 8. CONCLUSION

This project serves as a practical demonstration of how cloud computing and artificial intelligence can be combined to solve real-world problems using minimal infrastructure. The implementation of an image recognition system using **Azure Blob Storage**, **Azure Custom Vision**, and a **locally hosted web application** highlights the effectiveness and accessibility of modern cloud services in educational and prototype-level applications.

By enabling users to upload images through a simple web interface and leveraging Azure’s intelligent services to process and analyze those images, the system delivers a full-stack solution that mimics real-world AI workflows. The architecture illustrates the end-to-end pipeline from **data collection**, **cloud-based storage**, and **AI inference**, all the way back to **user feedback**, fostering a deeper understanding of how distributed systems and cloud APIs operate.

**Key Learnings**

* **Hands-on Experience with Cloud APIs:**  
  The project offered real, working experience using Azure Blob Storage for file management and Azure Custom Vision for AI-based image classification or object detection. This reinforced how REST APIs and SDKs can be incorporated in web applications.
* **Full Stack Integration:**  
  From frontend design and backend processing to cloud deployment, the system shows how all layers of a modern application can be tied together efficiently.
* **Cloud + AI Synergy:**  
  Using cloud-hosted models to analyze uploaded content demonstrates the power of offloading complex AI tasks to scalable platforms, saving local resources and increasing flexibility.
* **Rapid Prototyping of AI Solutions:**  
  The project mimics how startups and enterprises prototype intelligent features before full-scale deployment, validating business and technical feasibility with minimal investment.

**Future Enhancements**

While the current system provides a complete proof-of-concept, several improvements can be considered to make it production-ready:

1. **Cloud Deployment:**
   * Move from local hosting to **Azure App Service**, **Heroku**, or **Netlify** to make the web app accessible from anywhere.
2. **User Authentication:**
   * Add login and authentication mechanisms to control access and associate uploads with individual users.
3. **Batch Image Processing:**
   * Enable bulk image uploads and simultaneous predictions using asynchronous queues like **Azure Functions** or **Azure Logic Apps**.
4. **Improved AI Accuracy:**
   * Train the Custom Vision model with a larger and more diverse dataset.
   * Incorporate **model iteration management** to track performance over time.
5. **Rich UI Enhancements:**
   * Add dynamic image previews, loading animations, and detailed confidence visualizations using charts or bounding boxes.