VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
Department of Computer Engineering

Project Report on

Intelligent Storage for Image Classification

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN **COMPUTER ENGINEERING**

By

Mansi Bellani(07)

Khushi Bhatia(08)

Muskan Chhabria(13)

Shubham Gupta(21)

Project Mentor
Mrs. Yugchhaya Galphat

University of Mumbai
(AY 2022-23)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
Department of Computer Engineering

CERTIFICATE

This is to certify that the Mini Project entitled **"Intelligent storage for image Classification"** is a bonafide work of **Mansi Bellani (07), Khushi Bhatia (08), Muskan Chhabria (13) & Shubham Gupta (21)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering".**

(Prof. Mrs.Yugchhaya Galphat)

(Dr. Nupur Giri)
Head of Department

(Dr. J. M. Nair)
Principal

Mini Project Approval

This Mini Project entitled **"Intelligent storage for image Classification"** is a bonafide work of **Mansi Bellani (07), Khushi Bhatia (08), Muskan Chhabria (13) & Shubham Gupta (21)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

1……
(Internal Examiner Name & Sign)

2……
(External Examiner name & Sign)

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

---------- ----------

(Signature) (Signature)

Mansi Bellani- 07 **Khushi Bhatia- 08**

---------- ----------

(Signature) (Signature)

Muskan Chhabria- 13 **Shubham Gupta-21**

Date:

Contents

Abstract

Intelligent cloud storage for image classification is a rapidly growing field, with a wide range of applications in various industries such as healthcare, agriculture, and retail. In this project, we propose a cloud-based image classification system using Microsoft Azure services. The system will leverage Azure Cognitive Services to perform image analysis and classification, while Azure Blob Storage will be used for storing and managing image data. The system will be scalable, secure, and reliable, enabling users to store, manage, and classify large volumes of image data in the cloud. The proposed system will have numerous benefits, including reducing storage costs, improving the accuracy of image classification, and increasing the efficiency of data management. The project will involve designing and developing the system using various Azure services, testing and optimizing the system performance, and evaluating the system's effectiveness in real-world scenarios. The results of this project will be useful for organizations that deal with large volumes of image data and need an efficient and scalable way to manage and classify their data in the cloud.

1. Introduction

1.1 Introduction

The growing volume and complexity of data generated in today's digital age have led to increasing demand for cloud-based solutions that can store, manage, and process data efficiently and reliably. Azure, Microsoft's cloud computing platform, offers a wide range of services that enable organizations to develop and deploy cloud-based solutions quickly and cost-effectively. In this project, we propose to develop a cloud-based image classification system using Azure Blob Storage for storage, Azure Machine Learning for machine learning, and Azure Web App Service for deployment.

1.2 Motivation

Image classification is a critical task in various industries, including healthcare, retail, and agriculture. However, managing and processing large volumes of image data can be challenging, requiring significant storage and computational resources. Cloud-based solutions offer a scalable, secure, and cost-effective way to store and process image data, enabling organizations to focus on their core business operations rather than managing IT infrastructure. Azure provides a comprehensive set of services that can be used to develop and deploy cloud-based image classification systems quickly and easily

1.3 Problem Statement & Objectives

The objective of this project is to develop a cloud-based image classification system that can effectively store, manage, and classify large volumes of image data. The system will leverage Azure Blob Storage for storage, Azure Machine Learning for machine learning, and Azure Web App Service for deployment. The project aims to address the challenges associated with managing and processing large volumes of image data by developing an efficient and scalable cloud-based solution that can be easily integrated into existing business workflows.

2. Literature Survey

2.1 Survey of Existing System

Title	Inference Drawn
Research on Cloud Data Storage Technology and Its Architecture Implementation	The paper discusses the architecture of cloud storage systems and various technologies used, such as distributed file systems, object storage, and block storage. The paper also highlights the advantages and disadvantages of these technologies and their respective use cases. Furthermore, the paper identifies the challenges faced by cloud storage systems and potential future research directions, such as improving data security and privacy, enhancing system scalability and reliability, and developing more efficient data management and analysis techniques. Overall, the paper provides a comprehensive overview of cloud storage technology and its implementation, making it a valuable resource for researchers and practitioners in the field of cloud computing and storage.
A Study on Information Classification and Storage in Cloud Computing	The paper proposes a group collaborative intelligent clustering approach to classify and store data in cloud computing data centers. The paper also discusses the benefits of this approach, including improved data security, accessibility, and reliability. Furthermore, the paper identifies the challenges faced by cloud computing data centers and suggests future research directions, such as optimizing the clustering algorithm and improving the performance of data storage systems. Overall, the paper provides insights into the classification and storage of information in cloud computing data centers, making it a valuable resource for researchers and practitioners in the field of cloud computing.

2.2 Limitations of Existing System or Research gap

- Limited Data Availability: Existing cloud storage systems may not have access to a large and diverse set of training data, which can limit their ability to classify images accurately.
- Limited Model Customization: Many cloud storage services offer pre-built models that cannot be customized or fine-tuned to meet specific user requirements. This can limit their usefulness for applications that require specialized models or tailored solutions.
- Limited Integration with Other Systems: Existing cloud storage services may not integrate seamlessly with other systems or tools, which can make it difficult for users to manage and analyze their image datasets.

3. Proposed System

3.1 Introduction

In this project, We aim to develop a cloud-based image classification system using Azure Blob Storage, Azure Machine Learning, and Azure Functions. We will collect and preprocess a dataset of images, train a machine learning model, and deploy it using Azure Functions. The results of the image classification will be stored in Azure Blob Storage for easy access and analysis. This approach will enable us to efficiently manage and process large volumes of image data, making it a valuable tool for industries such as healthcare, retail, and agriculture.

3.2 Services Used

I. **Azure Blob Storage for Image Storage:** Blob Storage is optimized for storing massive amounts of unstructured data. Unstructured data is data that doesn't adhere to a particular data model or definition, such as text or binary data.

Blob Storage is designed for

- ➢ Serving images or documents directly to a browser.
- ➢ Storing files for distributed access.
- ➢ Streaming video and audio.
- ➢ Writing to log files.
- ➢ Storing data for backup and restore, disaster recovery, and archiving.
- ➢ Storing data for analysis by an on-premises or Azure-hosted service.

Blob Storage offers three types of resources:

- **The storage account:** Provides a unique namespace in Azure for our data.
- **A container in the storage account:** Organizes a set of blobs, similar to a directory in a file system, storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.
- **A blob in a container:** It has 3 types of blobs -
  - A. Block blobs store text and binary data.
  - B. Append blobs are more optimized for append operations, They are ideal for scenarios such as logging data from virtual machines.
  - C. Page blobs store random access files up to 8 TiB in size, It stores VHD files and serves as disks for Azure virtual machines.

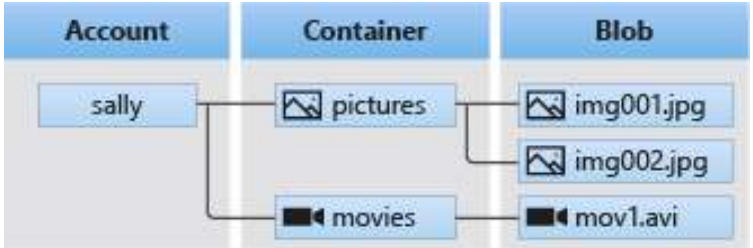

Figure 3.2.1
Resources of Blob Storage.

II. **Azure Machine Learning:** We can create a model in Azure Machine Learning or use a model built from an open-source platform, such as PyTorch, TensorFlow, or sci-kit-learn. MLOps tools help us monitor, retrain, and redeploy models. Typically models are developed as part of a project with an objective and goals. Projects often involve more than one person. When experimenting with data, algorithms, and models, development is iterative. The Azure ML project life cycle model is a framework that outlines the stages of building a machine learning model in Azure ML. The model consists of six stages:

1. **Business Understanding:** The first stage of the Azure ML project life cycle is to identify the business problem or opportunity that the machine learning model is intended to address. This stage involves defining the problem statement, the business objectives, and the success criteria.

2. **Data Acquisition and Understanding:** In this stage, data scientists gather and explore the data required to train the model. This includes acquiring, cleaning, and preprocessing data, as well as performing exploratory data analysis (EDA) to gain insights into the data and its distribution.

3. **Modeling:** The modeling stage involves selecting appropriate algorithms and building and training the machine learning model using the data prepared in the previous stage. This stage involves selecting a suitable algorithm, choosing hyperparameters, and tuning the model to achieve the best performance.

4. **Evaluation:** In this stage, the trained model is evaluated on a separate dataset that was not used for training. This helps to determine the effectiveness and accuracy of the model, and whether it meets the success criteria defined in the first stage.

5. **Deployment:** Once the model is trained and evaluated, it can be deployed to a production environment for use. This stage involves packaging the model and deploying it to an Azure environment, where it can be used by business applications.

6. **Monitoring and Maintenance:** The final stage of the Azure ML project life cycle is monitoring and maintenance. This involves tracking the performance of the deployed model and making any necessary adjustments or updates to ensure that it continues to perform effectively and accurately over time.

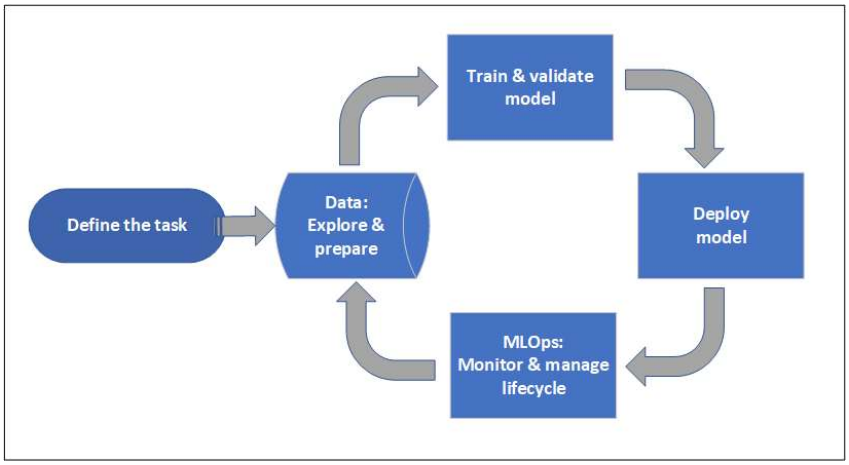

Figure 3.2.2
Project Lifecycle

III. Web App Services for Deployment:

We can create a web app using Azure portal, VS Code, Azure Tools extension pack, or Azure CLI.
Here are the steps involved in deploying a web app to Azure App Service:

1. **Create a web app in Azure:** Log in to the Azure portal and create a new web app using the App Service feature. Choose the appropriate settings such as the OS, runtime stack, and pricing tier based on your application's requirements.

2. **Deploy your application code to Azure:** There are several ways to deploy your code to Azure App Service, including using Git, FTP, or Azure DevOps. You can also use Azure CLI or Azure PowerShell to deploy your code programmatically.

3. **Browse to the app:** Once your code is deployed, you can browse to your web app using the URL provided by Azure App Service. You can also set up custom domain names for your app.

4. **Stream logs:** Azure App Service provides logs for your application, which can help you troubleshoot issues or monitor your app's performance. You can view these logs using the Azure portal or stream them using the Azure CLI or Azure PowerShell.

By following these steps, you can easily deploy your web app to Azure App Service and start running your application on the cloud.

When it comes to repository branches, web app slots can be used to deploy different branches of your code. For example, you could have a production slot that deploys your main branch and a staging slot that deploys a development or feature branch. This allows you to test new features in a separate environment before promoting them to production. You can configure your web app slots to automatically deploy from a specific branch in your repository whenever there is a new commit or pull request. Web app slots are a feature of Azure App Service that allows you to deploy multiple versions of your web app on a single instance. A slot is essentially a separate deployment environment for your web app, allowing you to test new features, perform staging, and deploy updates without affecting the production environment. Each slot has its own URL, and you can swap slots to promote a staging environment to production or roll back a production release.

Different types of deployment slots that you can use to manage different stages of your application's deployment and testing lifecycle:-

1. **Production Slot:** This is the primary slot for your application, and it is used to host the live version of your application that is accessible to end-users. This slot should always contain the latest stable version of your application, and you should minimize the amount of changes you make directly to this slot to reduce the risk of introducing bugs or downtime.

2. **Master Slot:** This is a special slot that is often used as a backup for the production slot. It is usually configured to automatically synchronize with the production slot so that if there is an issue with the production slot, you can quickly swap to the master slot to restore your application's availability.

3. **Stage Slot:** This is a separate deployment slot that is used to test new features or changes before they are promoted to the production slot. You can deploy a new version of your application to this slot and perform testing and validation before promoting it to the production slot. This helps to reduce the risk of introducing bugs or downtime in the production environment.

4. **Dev Slot:** This is a deployment slot that is used for development and testing purposes. You can deploy your latest code changes to this slot and test them in isolation before promoting them to the stage or production slots. This allows you to test and validate your changes without affecting the stability of your production or stage environments.

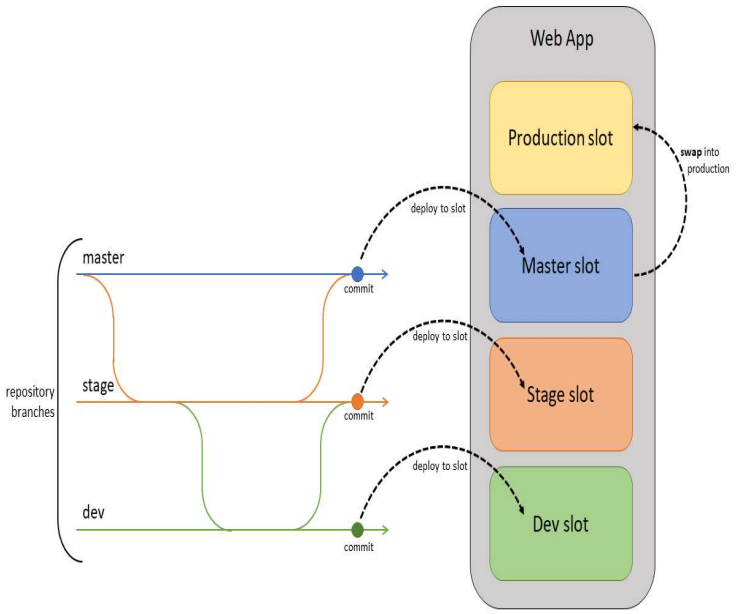

Figure 3.2.3
Life cycle of Web App Deployment.

Overall, these deployment slots provide a powerful way to manage the different stages of your application's lifecycle, from development and testing to production and backup. By using these slots, you can reduce the risk of introducing bugs or downtime in your application and ensure that you have a robust and reliable deployment strategy.

IV. Azure Cosmos DB:

Azure Cosmos DB is a fully managed NoSQL and relational database for modern app development. Azure Cosmos DB offers single-digit millisecond response times, and automatic and instant scalability, along with guaranteeing speed at any scale. Business continuity is assured with SLA-backed availability and enterprise-grade security.As a fully managed service, Azure Cosmos DB takes database administration off your hands with automatic management, updates, and patching. It also handles capacity management with cost-effective serverless and automatic scaling options that respond to application needs to match capacity with demand. Cosmos DB is a globally distributed, multi-model database service provided by Microsoft Azure that is designed to handle large amounts of structured, semi-structured, and unstructured data.

The key features of Cosmos DB are:

- **Globally Distributed:** With Azure regions spread out globally, the data can be replicated globally.
- **Scalability:** Cosmos DB is horizontally scalable to support hundreds of millions of reads and writes per second.
- **Schema-Agnostic Indexing:** This enables the automatic indexing of data without schema and index management.
- **Multi-Model:** It can store data in Key-value Pairs, Document-based, Graph-based, Column Family-based databases. Global distribution, horizontal partitioning, and automatic indexing capabilities are the same irrespective of the data model.
- **High Availability:** It has 99.99 % availability for reads and writes for both multi-region and single-region Azure Cosmos DB accounts.
- **Low Latency:** The global availability of Azure regions allows for the global distribution of data, which further makes it available nearest to the customers. This reduces the latency in retrieving data.

Working of Cosmos DB:
If a website used by people all over the world writes its data into a primary database in one location (non-multi-master mode), the people near the location will be able to retrieve the data faster than the rest of the world due to network latency issues.

But Cosmos DB has multi-master support where the data can be simultaneously written into different databases spread out globally. In this way, the data is replicated onto the user's nearest region so that it can be accessed faster. But there can still be a difference of milliseconds between the data being replicated and this affects the consistency. Consistency indicates whether the data are in sync and are in the same state at any given point in time. Cosmos DB offers multiple levels of consistency with varying performances and availability. Cosmos DB allows us to set a default consistency while creating it and which can be changed from the application while retrieving data.

The various consistency levels offered are:

- **Eventual:** Here the data is written on the primary node and is propagated eventually to read-only secondary nodes. It might take some time for the users to get updated data.
- **Consistent Prefix:** Clients can read data in the same order as it is written.
- **Session:** Users who just committed the data will be able to see it but it will take some time for the others to get that data version.
- **Bounded Staleness:** Here a staleness period can be set, for which the data won't be replicated into the secondary nodes.
- **Strong:** This offers the latest copy of data for all the users but gives a relatively low performance.

The process for using Cosmos DB typically involves the following steps:

1. **Create a Cosmos DB account:** To use Cosmos DB, you first need to create a Cosmos DB account in Azure. This account will provide you with a URL endpoint and a set of keys that you can use to access your database.

2. **Choose a data model:** Cosmos DB supports multiple data models, including SQL, MongoDB, Cassandra, Azure Table Storage, and Gremlin. You will need to choose the data model that best fits your application's requirements.

3. **Create a database and containers:** Once you have chosen your data model, you can create a database and containers within your Cosmos DB account. A container is similar to a table in a traditional relational database and is used to store data for a specific partition key.

4. **Write data to your database:** Once your containers are set up, you can begin writing data to your Cosmos DB database. You can use the appropriate API for your chosen data model to insert, update, and query data.

5. **Monitor and optimize performance:** Cosmos DB provides various performance monitoring and optimization tools to help you optimize the performance of your database. You can monitor query metrics, analyze usage patterns, and configure features such as automatic indexing and partitioning to ensure that your database is performing optimally.

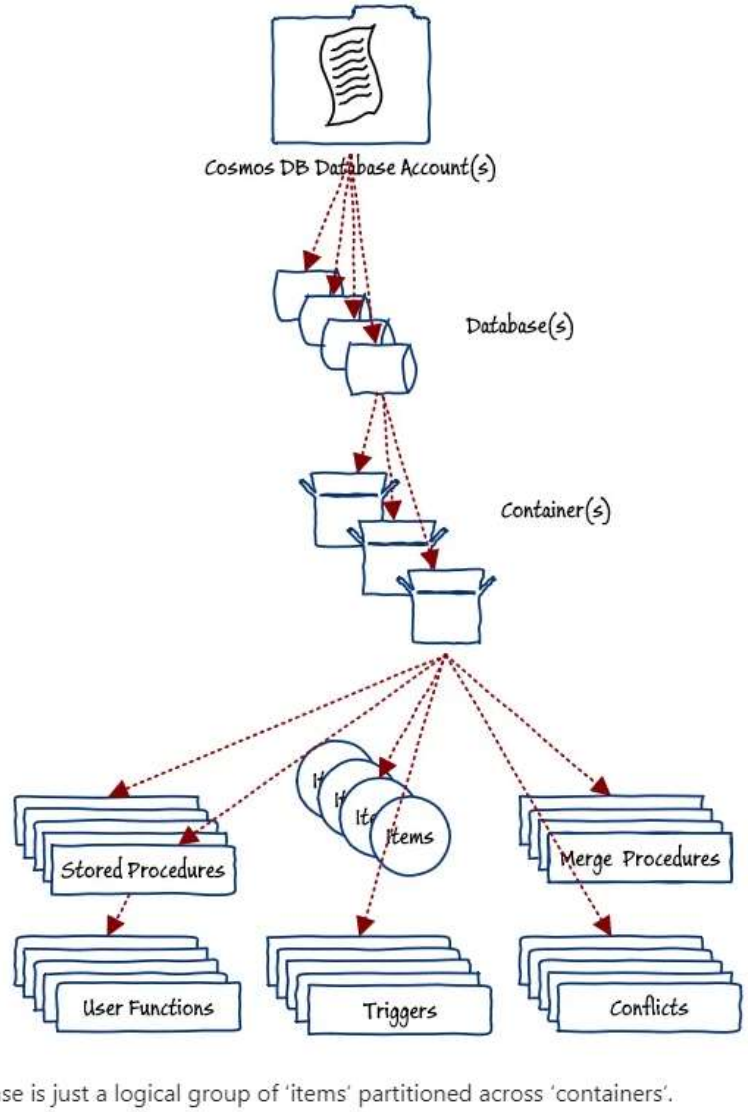

Figure 3.2.4
Flow of Cosmo DB

Overall, the process of using Cosmos DB involves creating an account, choosing a data model, setting up containers, writing data to your database, and optimizing performance. By following these steps, you can take advantage of the powerful capabilities of Cosmos DB to build high-performance, globally distributed applications that can handle a wide variety of data types and workloads.

3.3 Methodology Applied

1. **Data Collection:** We collected a dataset of images for training and testing our machine-learning model. We used publicly available datasets and also created our own custom dataset. The images were stored in Azure Blob Storage.
2. **Preprocessing:** The images were preprocessed to ensure that they are in a format that can be used by our machine learning model. We used Azure Functions to perform this step, which involved resizing, cropping, and converting the images to grayscale.
3. **Machine Learning Model Training:** We trained a machine learning model to classify the images. We used Azure Machine Learning to train the model. We used transfer learning to leverage pre-trained models and fine-tuned them to suit our specific use case.
4. **Deployment:** Once the model was trained, we deployed it using Azure Functions. This allowed us to run our code without managing the underlying infrastructure. We used the Azure Function App service to host our code and deploy it as a serverless application.
5. **Result Storage:** We have used Azure Blob Storage to store the results of the image classification. This enabled us to easily access and analyze the results. We have used Azure Cosmos DB as a NoSQL and relational database to store metadata of images, timestamps, and results.
6. **Testing and Evaluation:** We tested the model using a separate dataset and evaluated its performance using standard metrics such as accuracy, precision, and recall. We also performed real-world testing to ensure that the system performed well in a practical scenario.

Overall, this methodology allowed us to develop a scalable, reliable, and efficient cloud-based image classification system using Azure services. By leveraging the power of cloud computing, we were able to process large volumes of image data quickly and accurately, making it a valuable tool for a range of industries.

3.4 Architectural Framework / Conceptual Design

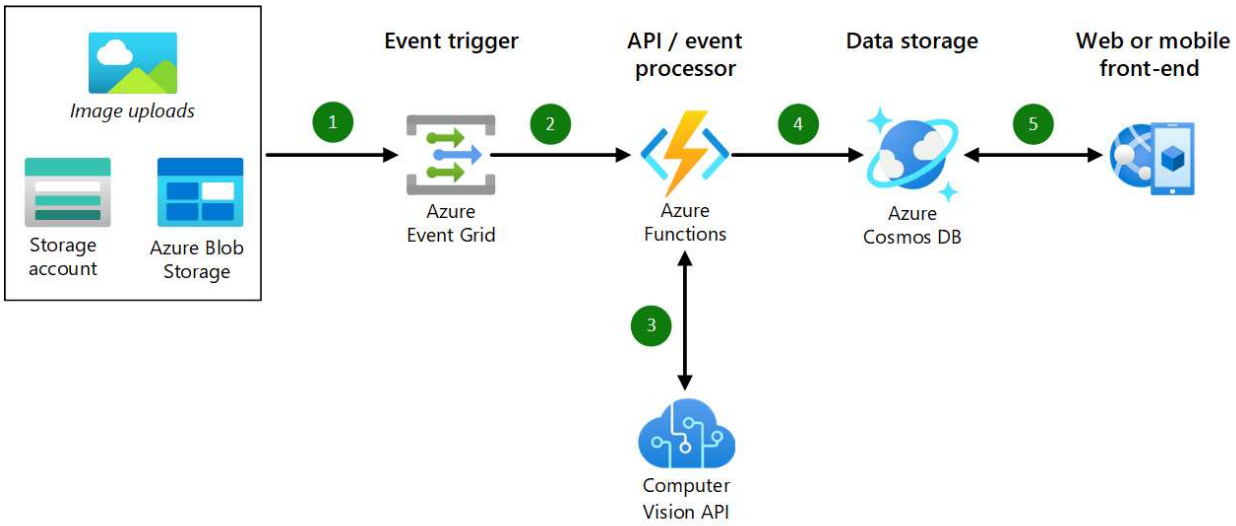

Figure 3.4.1
Conceptual design of proposed system

3.5 Azure and Other Software Specifications

1. Azure Storage Blob
2. Azure Machine Learning
3. Azure Web App Service
4. Python
5. Jupyter Notebook, Flask, Streamlit

3.6 Implementation & Result

In this project, we utilized Azure Blob Storage, Azure Machine Learning, Azure Web App Service, and Azure Cosmos DB to create an intelligent cloud storage system for image classification. Our goal was to develop a machine learning model that could accurately classify images stored in Azure Blob Storage.

Output screenshots of the preprocessing steps are shown:

Front-end of project Deployed on Azure web service

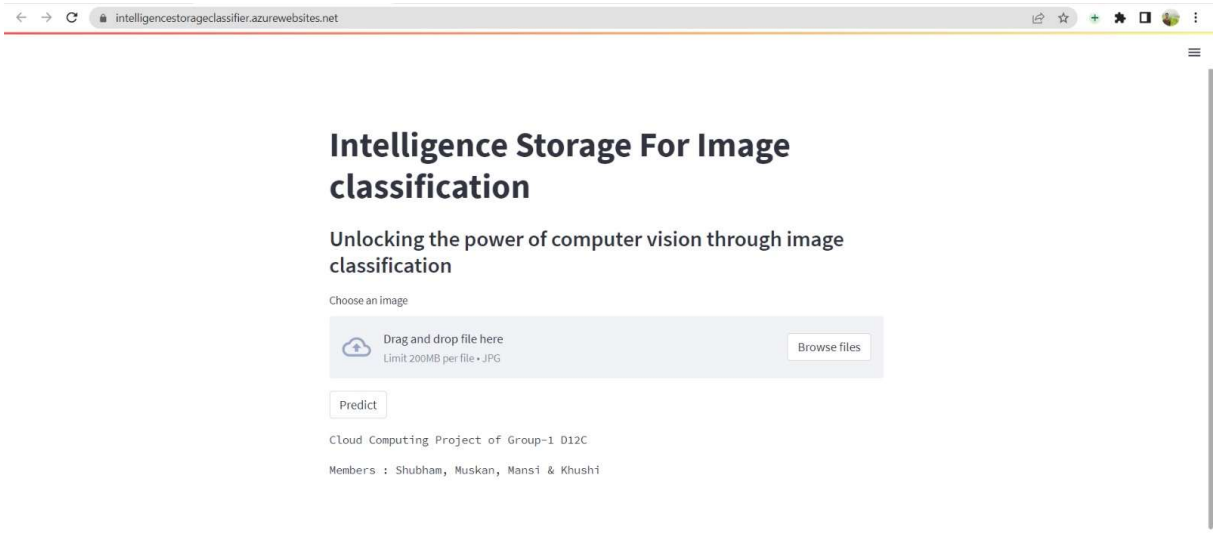

Figure 3.6.1
Landing page of the Intelligence Storage having,
Image upload section and button to predict

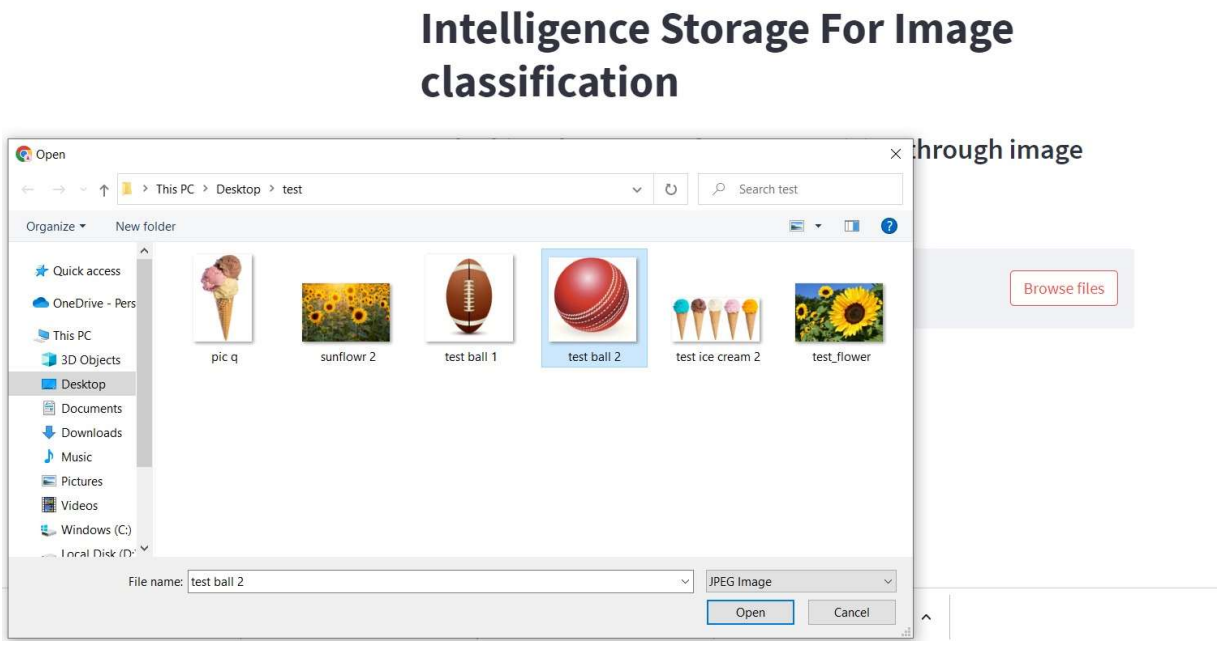

Figure 3.6.2
Uploading image for classification using Choose Image or Drag and Drop section

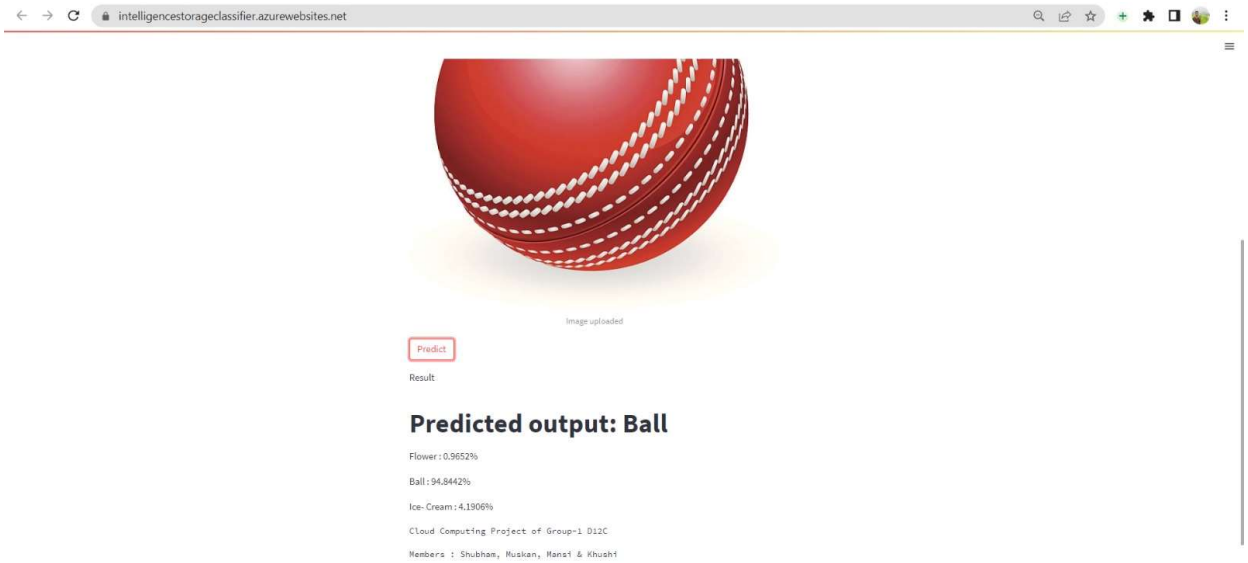

Figure 3.6.3
Output of uploaded image along with the prediction probabilities of other Target

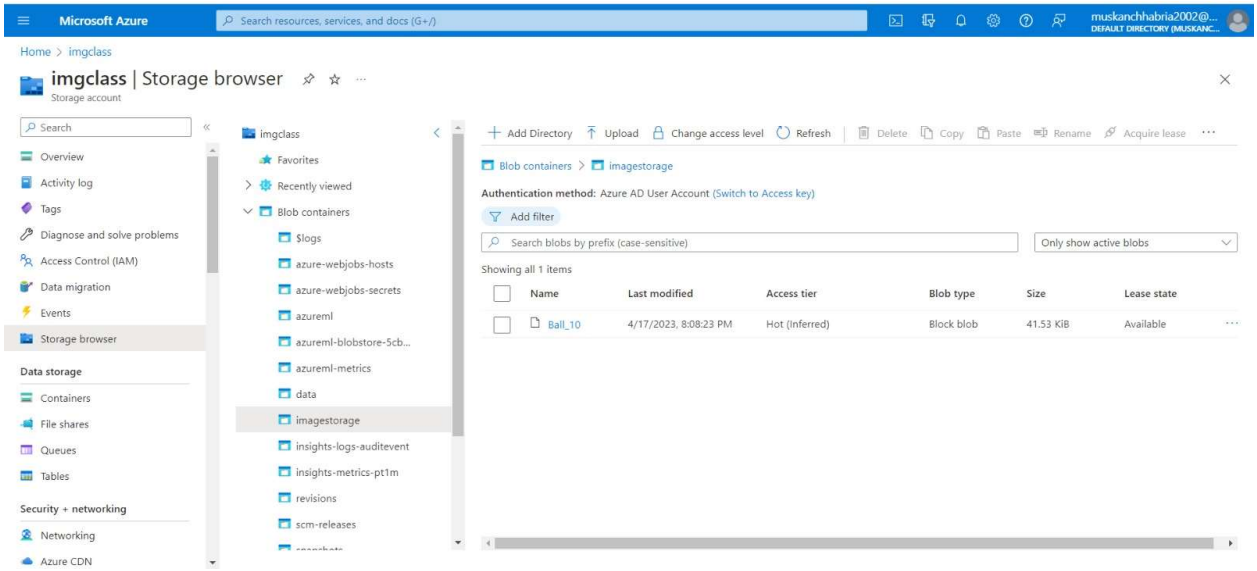

Figure 3.6.4
Image stored in Blob storage with is triggered while prediction

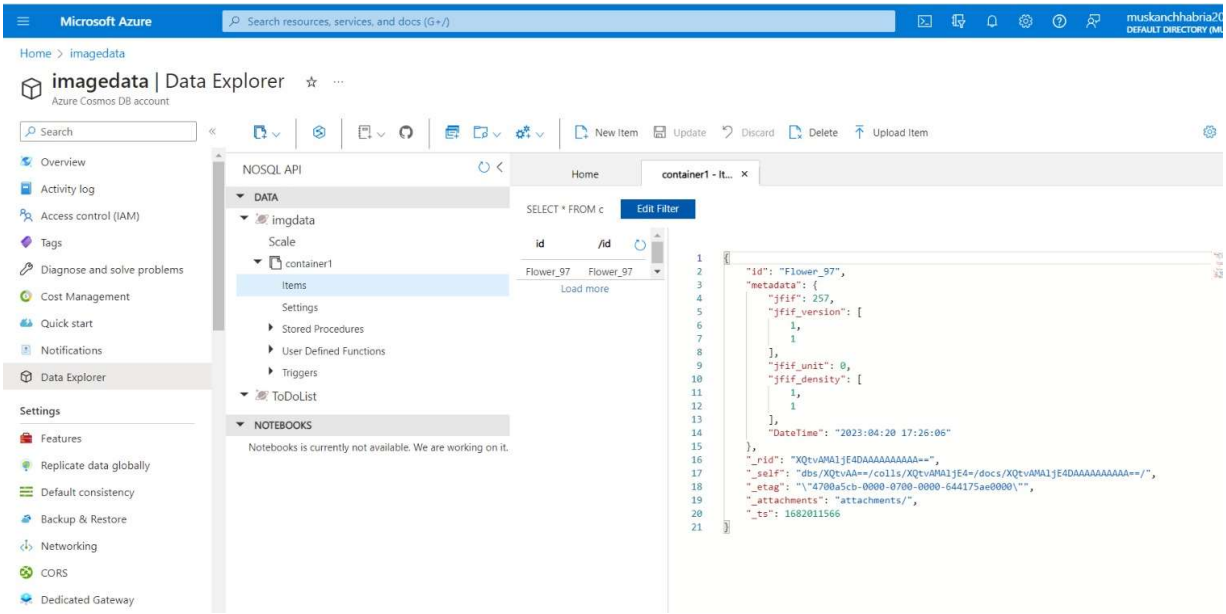

Figure 3.6.5
Metadata of Image, Timestamp and the Result is stored in Azure CosmoDB

3.7 Conclusion

In conclusion, we have successfully developed an intelligent cloud-based image classification system using Azure services. We collected and preprocessed a dataset of images, trained a machine learning model using Azure Machine Learning, and deployed it using Azure Functions. We used Azure Blob Storage to store the results of the image classification, making it easy to access and analyze the results. Our project has demonstrated the power and potential of cloud computing and machine learning in solving real-world problems. This image classification system can be applied in a wide range of industries such as healthcare, agriculture, and retail, where large volumes of image data need to be processed efficiently and accurately. In addition, we have explored the use of various Azure services, such as Azure Blob Storage, Azure Machine Learning, and Azure Functions. We have also discussed the benefits of using Azure Web App Service for deployment. Overall, this project has provided valuable insights into the process of developing a cloud-based image classification system using Azure services, and the potential of such systems in revolutionizing industries and solving real-world problems.

References

Research Papers:-

- Research on Cloud Data Storage Technology and Its Architecture Implementation
- A Study on Information Classification and Storage in Cloud Computing

Websites:-

- https://learn.microsoft.com/en-us/azure/machine-learning/v1/tutorial-train-deploy-notebook?view=azureml-api-1
- https://learn.microsoft.com/en-us/azure/event-grid/resize-images-on-storage-blob-upload-event?tabs=dotnet%2Cazure-powershell
- https://www.hindawi.com/journals/jece/2022/1476661/