Project Proposal

Group Information:

Phillip O'Reggio, pno3. Skills: Java, C, C++, took OS and Systems programming recently
Xinyu Wu, xw586. Skills: Java, Machine learning, Deep learning, Python.
Shiyin Tang, st966. Skills: Java, Python, C/C++, Django, React, Object-oriented design.
We don't take the M.Eng Credits.

Project idea:

Our basic idea is to build a smart photo album web application using Azure Blob Storage with Django framework. Users can upload and share pictures, like those interactive picture frames, each owned by oneself. They can also join a group with whoever has similar interests. They can also search pictures with keywords, so this involves image classification that we will talk about in detail later.

Cloud Computing System Components:

For now, we think our project is going to involve these Components:

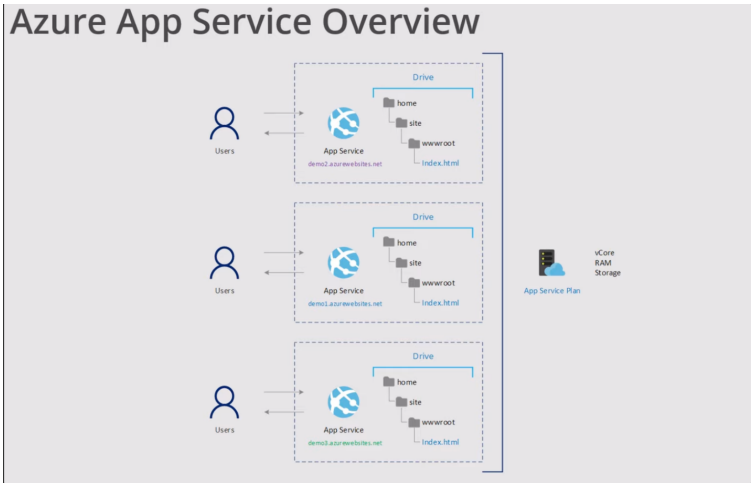
- **App Service**: for quickly build, deploy, and scale web apps and APIs
- **Computer Vision API**: is used to retrieve information about each image.
- **Azure Functions**: provides the back-end API for the web application, as well as the event processing for uploaded images.
- **Event Grid** : triggers an event when a new image is uploaded to blob storage. The image is then processed with Azure functions.
- **Blob storage**: stores all of the image files that are uploaded into the web application, as well any static files that the web application consumes.
- **CosmosDB**: stores metadata about each image that is uploaded, including the results of the processing from Computer Vision API.

Data source and other resources:

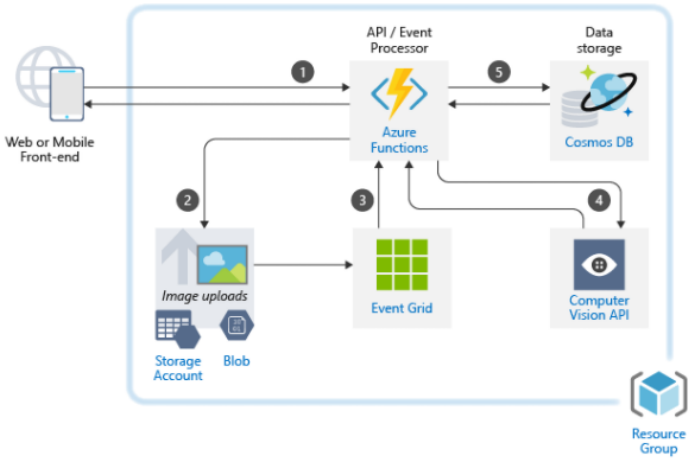
The data being analyzed and processed is provided by the users, so there is no external source or sensor input.
For the purpose of testing our project, we will create test users to use the artificial photo album system implemented on the backend.

Initial system design and architecture:

This is a high level system architecture for the web application service part. This part will be related to system logics like user login in/out, authentication, data management and picture sharing among users and so on.



This is the system architecture for the artificial photo classification part.



We basically divide the picture classification system into five primary parts as shown above:

1. The API layer is built using Azure Functions. These APIs enable the application to upload images and retrieve data from Cosmos DB.
2. When an image is uploaded via an API call, it's stored in Blob storage.
3. Adding new pictures to Blob storage triggers an Event Grid notification to be sent to an Azure Function.
4. Azure Functions sends a link to the newly uploaded pictures to the Computer Vision API to analyze.
5. Once the data has been returned from the Computer Vision API, Azure Functions makes an entry in Cosmos DB to persist the results of the analysis along with the image metadata.

Timeline:

3.21 - 3.27
Learn Azure Functions, App Services, Azure Storage and any related technical skills.
3.28 - 4.10
Everyone in our group will focus on different parts as described below, so work will be done concurrently, with each one focusing on some aspects. We will build a demo web application using App Service and deal with user login/out, user authentication.
Develop the application to upload images and save them into the Blob Storage system.
4.11 - 4.18
Deal with image uploading, sharing and group setting functions and so on.
Build Event Grid notification function to send new pictures to Computer Vision API part.
Implement Computer vision API to classify pictures data stream into different types.
4.19 - 4.26
Deal with smart search function with processed CosmosDB storage.
Return the classification results to Azure Functions and save the pictures along with different types tags into Cosmos DB.
4.26 - 5.5
System Testing and Documentation.

Workload Allocation:

Shiyin is going to focus on the back-end APIs and front-end visualization given those APIs.
Phillip is going to focus on the Cloud Storage aspect.
Xinyu is going to focus on the photo classification part.
It is likely some parts will be finished sooner than others, and others may take longer than others. In that case, we can further divide tasks among parts that were more complicated than we expected.