Image File Uploader Application

To begin with, this application is split into 2 projects:

- **Server side** – This would comprise of the Web API which would handle the uploading images to Azure Blob Storage, storing of image to local server storage if internet is not available, and the storage of the Blob/Image details to a relational database
- **Client side** – This would allow the user to upload the images to the Web API endpoint.

Server side

Completing the server-side task was carried out and broken into the following steps:

Creating the '/ImageFileUploader' Endpoint

To complete the task, I have to create a post action with accept a file from a http request. The file is then validated to ensure that file is an image of type jpeg or png and the dimension is 1024*1024. If the file does not meet the requirement a bad request http status (400) is returned to the client with the appreciated message.

Adding the ability to store image in local Storage to Endpoint

To accomplish this task, local storage service was created. This service is mapped to its interface in the IService collection (dependency injection container), and injected into the ImageFileUploader controller. This ensures that our services are not tightly couple, and also makes service mocking in unit test from the endpoint possible. Furthermore, it allows for better maintenance of the code base. Calling the store async method on the services, the file to a Resources/Images of the current directory.

Storing Blob/Image file details to the SQL Server database table via endpoint

To achieve this task, entity framework code first approach is utilized to set up and interact with the database table. The dbcontext here is the ImageManagementContext. This stores the image id, name, url and store location (local or cloud). The details storage takes place after the image has been stored locally

Adding the ability to store image in Azure Blob Storage

To accomplish this task, Azure Storage Blob nugget package was used. The implementation was wrapped into the service class AzureStorage, similarly to Local Storage, and injected into the ImageFileUploader controller. Upon the implementation of this service, a number of changes were made to the endpoint:

- We attempt to upload the image to Azure storage, if that fails, we storage the image locally, and write details to database
- We attempt to upload the image to Azure storage, if the image already exists in Azure storage, we notify the user with a bad request http status (400)

Client Side

This is implemented using Angular, so that it is a SPA application, this stops the page having to reload every time we upload an image to the server '/ImageFileUploader' endpoint.