Image Comparator

technical specification document

murale karthiCk arumugam

Contents

[Application Overview 2](#_Toc27843298)

[Document Terms/Abbreviations used 2](#_Toc27843299)

[Key Functional Requirements 2](#_Toc27843300)

[Service Design – Functional Overview 3](#_Toc27843301)

[Technology Decisions 3](#_Toc27843302)

[Image Comparison Method – SSIM vs MSE 3](#_Toc27843303)

[Service Design – Architectural Overview 4](#_Toc27843304)

[Detailed Design 4](#_Toc27843305)

[Telemetry Design 5](#_Toc27843306)

[Validations 6](#_Toc27843307)

[Secrets Management 6](#_Toc27843308)

[Service Level Objective 6](#_Toc27843309)

[Monitoring 7](#_Toc27843310)

[Security & Privacy 7](#_Toc27843311)

[Bibliography 7](#_Toc27843312)

[Version Control 7](#_Toc27843313)

# Application Overview

The purpose of the application is to allow the users to compare images. The tool performs a simple comparison of the images provided and returns a value on a scale of 0 to 1. The tool accepts the input in the form of a csv and allows the user to perform one or multiple comparisons at the same time.

# Document Terms/Abbreviations used

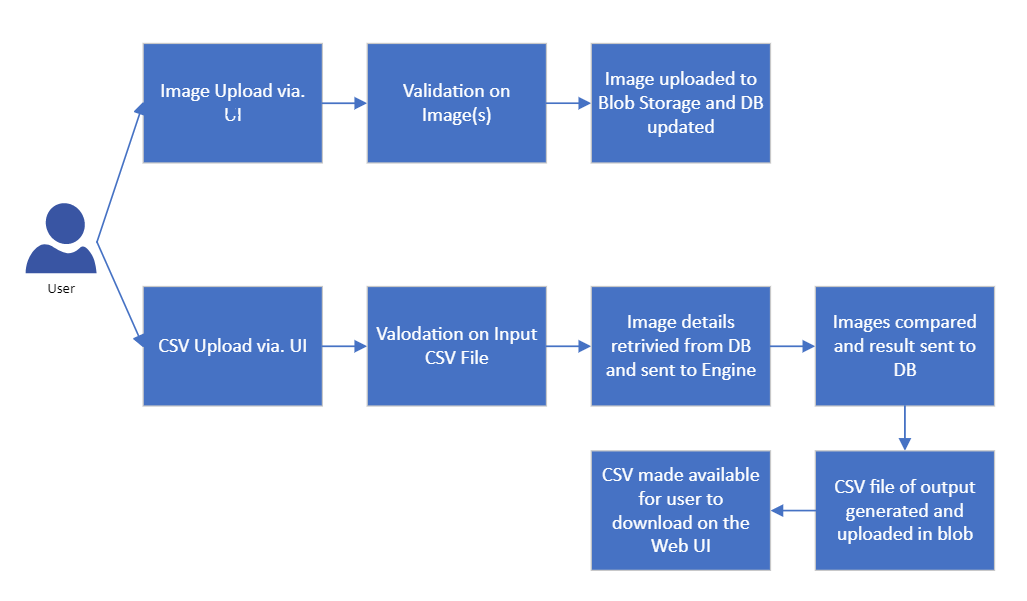
|  |  |
| --- | --- |
| **Term** | **Definition** |
| CSV | Comma Separated File |
| MB | Mega Bytes |
| SLO | Service Level Objective |
| SLI | Service Level Indicator |
| DB | Database |
| GDPR | General Data Privacy Requirement |

# Key Functional Requirements

Some of the key functional requirements are:

* Input and Output should be in CSV format
* Image comparison should be performed across different formats
* Comparison result is returned on a scale of 0 to 1
* Time taken (cost) of comparison is logged and returned together with the result

# Service Design – Functional Overview



# Technology Decisions

* Azure functions will be used to create the automation using C# language and .Net Framework 4.5 and .Net Core 2.1. All the components will be hosted in Azure cloud.
* Functions will be created as individual microservices which can be exposed to external system when needed using function proxies.
* Azure Functions are serverless and are capable of Autoscaling when the load increases.
* Costing model is Pay per use.
* Existing Azure SQL Database will be used to store the transactional data.
* All the Telemetry will be logged in Application Insights.

# Image Comparison Method – SSIM vs MSE

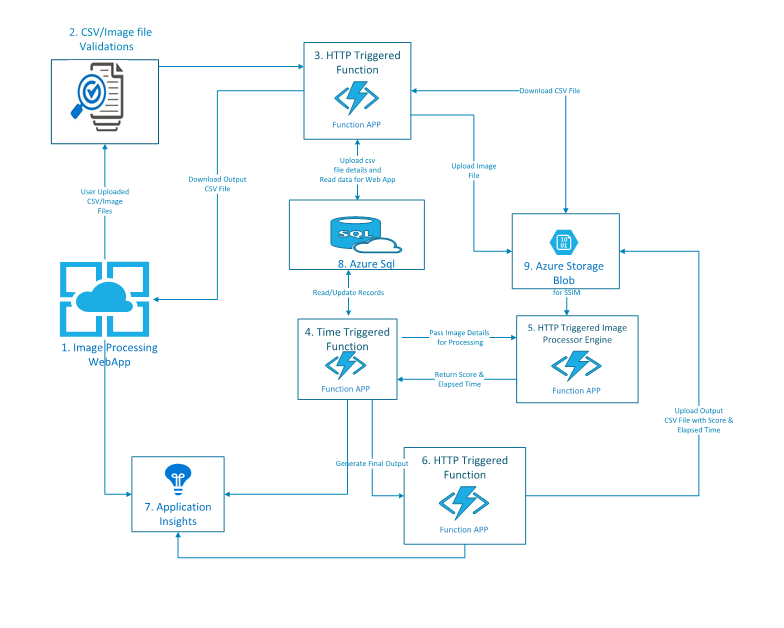
The SSIM method is clearly more involved than the MSE method when it comes to Image Comparison. SSIM attempts to model the perceived change in the structural information of the image, whereas MSE estimates the perceived errors. There is a subtle difference between the two, but the results are dramatic [1].

While the MSE is substantially faster to compute, it has the major drawback of (1) being applied globally and (2) only estimating the perceived errors of the image.

On the other hand, SSIM, while slower, can perceive the change in structural information of the image by comparing local regions of the image instead of globally.

For our implementation, this tool will utilize the SSIM scoring algorithm open source package by Andrew Johnson [2].

# Service Design – Architectural Overview

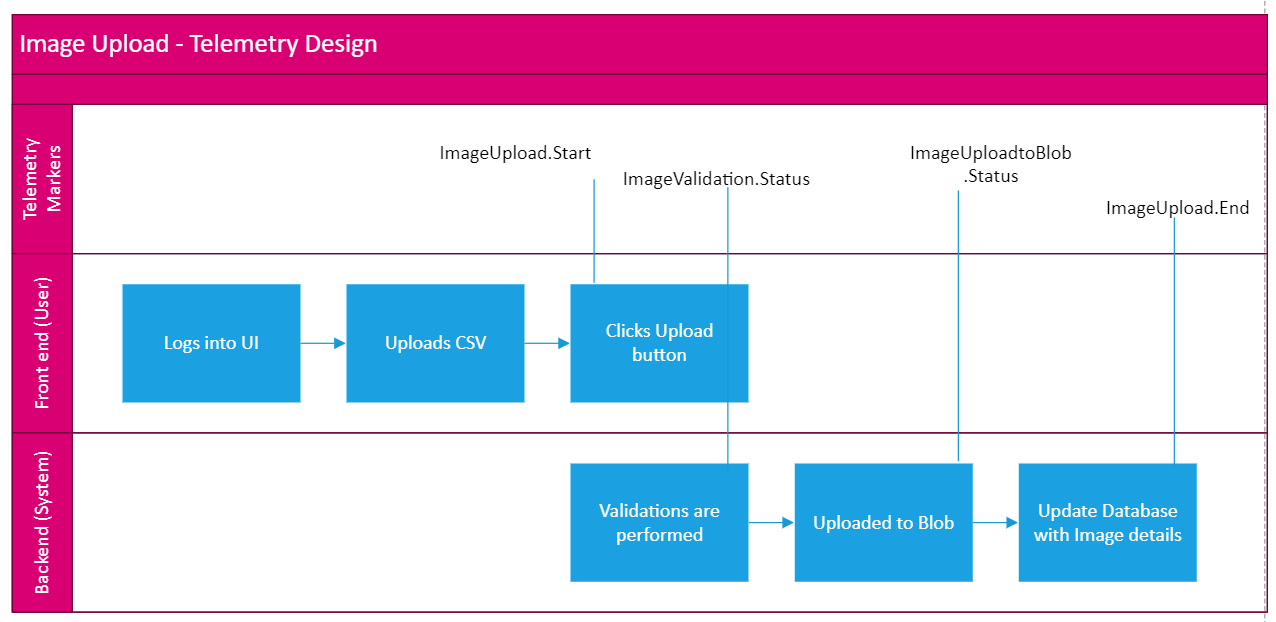


# Detailed Design

|  |  |  |
| --- | --- | --- |
| **S. No** | **Component** | **Description** |
| 1 | Web User Interface | The Web UI is available for the users to perform two functions 1) Upload the required images to blob storage for comparison 2) Upload the completed CSV for image comparison |
| 2 | Image/CSV Validations | This is a function app that performs validations on the image upload (*ex: file size of image should be greater than 0 bytes*) and validations on CSV input file (*ex: image 1 or image 2 details should not be blank*) |
| 3 | HTTP Triggered Function (Upload images) | This is a function app which uploads the images to the Blob storage once the user uploaded images are successfully validated |
| 4 | Time Triggered Function | This function reads the images to be compared from the Azure blob and makes it available for the Image Compare Engine |
| 5 | Image Processor Engine | The image process engine downloads the images from the blob storage and performs the actual comparison of the images and returns a value in the range of 0 to 1 |
| 6 | Time Triggered Function | This function app updates the output of the Image Processor Engine to the database. It also generates a CSV and uploads it to the Azure Blob for it to be available for the user. |
| 7 | Application Insights | All the telemetry from the various functional components are logged into Application Insights. This will in turn be used for monitoring and alerting on the health of the application |
| 8 | Key Vault | All the application specific constants are maintained in Key Vault. |
| 9 | Azure SQL | The Azure SQL DB stores transaction data including the details of the images compared, result of the comparison, time taken for comparison etc. |
| 10 | Azure Blob Storage | Azure blob storage is used to store the images uploaded by the user that will later be used for comparison |

# Telemetry Design

Here’s a sample telemetry design for logging the user activity and related details to Application Insights



# Validations

Below are the validations performed on user inputs:

Image upload

* Image is selected before performing an upload
* File uploaded is in an image format (??)
* File size of the image is greater than 0 bytes
* File size is not greater than 2 MB (??)

CSV upload

* Both Image 1 and Image 2 fields are provided to perform the comparison

# Secrets Management

The following are the key secrets that should be rotated at regular intervals:

* Storage Account Key – The key of the Azure Blob storage should be rotated, and the configuration should be updated
* Azure SQL Credentials – The Azure SQL server password should be updated, and the configuration should be updated

# Service Level Objective

SLO for file upload of images < 2 MB

|  |  |  |
| --- | --- | --- |
| SLO | | |
| SLI | Objective | |
| Upload Time in ms (p99) | Target | Attainment |
| <3000 ms | 99.90% |

SLO for Image Compare of images < 2 MB

|  |  |  |
| --- | --- | --- |
| SLO | | |
| SLI | Objective | |
| Elapsed Time in ms (p99) | Target | Attainment |
| <3000 ms | 99.90% |

# Monitoring

Infrastructure Monitoring – Availability, Reliability and Performance of the Azure Resources are monitored through out of the box Azure monitoring

Business Process Monitoring – End to end workflows are monitored based on the telemetry logged into Application Insights

# Security & Privacy

* There is no direct access to the users on this function. The Azure function is exposed only over the intranet and does not pose any security risk.
* Incoming data is parsed to be in standard format and is rejected if it does not meet the standard input data validation.
* Communication between the function, DB and Key vault is secured using MSI Authentication.
* Database firewall is open only for the Azure function calls. Database is set up for Auditing and threat detection.
* There is no customer data involved in the flow and the app is GDPR compliant.

# Bibliography

[1] <https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/>

[2] <https://github.com/helios2k6/MobileImageProcessor>

# Version Control

|  |  |  |
| --- | --- | --- |
| Version Number | Date Modified | Modified By |
| 1.0 | 12/20/2019 | Murale Karthick Arumugam |