Summary:

1.Demonstration of good SDLC practices –This solution is developed using ASP.NET CORE WEB API (C#) and azure cloud. Classes are organised as Models. Implemented a separate test file(.html) to test the API.

2.If cloud services are used, the reasons they were used : Azure storage blobs are used to upload the file and to read the file. This will help us to store and read larger files and to calculate the aggregate from the uploaded files.

3.Show a capability to create a service based solution : Implemented separate services for blob management and data processing.

4.Testing of the solution – Separate HTML page is crated to test the API end points.

5.Build and deployment tools used – Used Visual studio 2022 to build and publish the code. The deployment is done in Azure. Files are uploaded to GIT

**URL to access the API** <https://dataprocessingcsv-hyb5h7hugpd0cng0.australiaeast-01.azurewebsites.net/htmlpage.html>

**GIThub path**: <https://github.com/sarigathoovat/ShellCodingTask>

Ouputs:

1.Create a simple diagram to describe the solution and indicate the languages used

**Languages Used:**

* **HTML & JavaScript** : For the user interface. For handling API calls and dynamic content.
* **C#**: In the ASP.NET Core Web API for backend logic.
* **CSS** (inline) For styling the HTML UI

2.Demonstrate the invoking services used.

**1. File Upload**:

* **Method**: POST
* **Endpoint**: /api/data/UploadFileToBlobStorage
* **Functionality**: Uploads a file to Azure Blob Storage.
* **Example**: Invoked when the user selects a file and clicks an upload button in the UI.
* **Response**: Returns the URL of the uploaded blob once file uploaded to blob.

**2.List Blobs**:

* **Method**: GET
* **Endpoint**: /api/data/GetAllBlobFiles
* **Functionality**: Lists all files available in the Azure Blob Storage.
* **Example**: Invoked when the user wants to see available files.
* **Response**: Returns a list of blob URLs.

**3. Get Status**:

* **Method**: GET
* **Endpoint**: /api/data/GetAggregatedResult
* **Parameters**: fileName ,date, meter, dataType (as query parameters)
* **Functionality**: Retrieves aggregates (min, max, median) from the specified file, when the user submits the date, meter, and dataType in the UI.
* **Response**: Returns the aggregates.

**Simple Diagram of the Solution**

ASP.NET CORE WEB API (C#)

Services:

-Upload files to blob (POST)

-Process the CSV data by reading blob content (GET)

| |

+---------------------------------+

|

|

+---------------------------------+

HTML UI(Cloud)

Azure Blob Storage

Uploaded CSV files

HTTP Requests (e.g. POST & GET) to the API endpoints

Blob Container