API Documentation: File Upload and Processed Results

1. Input Data Format

When uploading a file, the text file should follow this format:
File Format: Text file (.txt)

Content Format:
Each line should contain a data record with the following fields, separated by commas:

- <Parameter>,<Value>

Example Input:

```
Temperature,22.5°C
Humidity,55%
Pressure,1015hPa
WindSpeed,12km/h
WindDirection,NE
Precipitation,0.5mm
Temperature,18.3°C
Humidity,65%
Pressure,1012hPa
WindSpeed,8km/h
WindDirection,SW
Precipitation,1.2mm
…
```

Supported Parameters:

- **Temperature:** Measured in degrees Celsius (e.g., 22.5°C)
- **Humidity:** Measured in percentage (e.g., 55%)
- **Pressure:** Measured in hectopascals (e.g., 1015hPa)
- **WindSpeed:** Measured in kilometers per hour (e.g., 12km/h)
- **WindDirection:** Compass direction (e.g., NE)
- **Precipitation:** Measured in millimeters (e.g., 0.5mm)

2. Processed Results Response

After uploading a file and processing its data, the response will include aggregated results for each parameter.

Response Format:

```
{
  "fileUploadId": <ID>,
  "fileName": "<File Name>",
  "fileType": "<File Type>",
  "uploadDate": "<Upload Date>",
  "status": "<Status>",
  "processedData": [
    {
      "dataKey": "<Parameter>",
      "count": <Count>,
      "averageValue": "<Average Value>",
      "minValue": "<Minimum Value>",
      "maxValue": "<Maximum Value>"
    },
    ...
  ]
}
```

Example Response:

```
{
  "fileUploadId": 2,
  "fileName": "testing data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded",
  "processedData": [
    {
      "dataKey": "Precipitation",
      "count": 2,
      "averageValue": "0.50mm",
      "minValue": "0.5mm",
      "maxValue": "0.5mm"
    },
    {
      "dataKey": "Temperature",
      "count": 2,
      "averageValue": "22.50°C",
      "minValue": "22.5°C",
      "maxValue": "22.5°C"
    },
    {
      "dataKey": "Humidity",
      "count": 2,
      "averageValue": "55%",
      "minValue": "55%",
      "maxValue": "55%"
    },
    {
      "dataKey": "WindSpeed",
      "count": 2,
      "averageValue": "12.00km/h",
      "minValue": "12.0km/h",
      "maxValue": "12.0km/h"
    },
    {
      "dataKey": "WindDirection",
      "count": 5,
      "averageValue": "NE",
      "minValue": "N/A",
      "maxValue": "N/A"
    },
    {
      "dataKey": "Pressure",
      "count": 2,
      "averageValue": "1015.00hPa",
      "minValue": "1015.0hPa",
      "maxValue": "1015.0hPa"
    }
  ]
}
```

Fields:

- **fileUploadId:** ID assigned to the uploaded file.
- **fileName:** Name of the uploaded file.
- **fileType:** Type of the uploaded file (e.g., text/plain).
- **uploadDate:** Date and time when the file was uploaded.
- **status:** Status of the file upload (e.g., Uploaded).
- **processedData:** Array of processed results for each parameter.
    - **dataKey:** The type of data (e.g., Temperature, Humidity).
    - **count:** Number of records for the given parameter.
    - **averageValue:** Average value of the parameter.
    - **minValue:** Minimum value recorded for the parameter.
    - **maxValue:** Maximum value recorded for the parameter.

Note:

- For the "WindDirection" data key, the averageValue field represents the most frequent wind direction (mode) within the dataset, as it is not possible to calculate a numerical average for categorical data like wind direction.

3. API Endpoints

3.1 Upload File

Endpoint: /api/v1/files/upload
Method: POST
Description: Uploads a file for processing.
Request Parameters:

- file (required): The file to be uploaded, specified as a multipart/form-data request with the key file.

Request Example:

- POST /api/v1/files/upload
- Content-Type: multipart/form-data
- file=@/path/to/your/file.txt

Response Format:

```
{
  "fileUploadId": <ID>,
  "fileName": "<File Name>",
  "fileType": "<File Type>",
  "uploadDate": "<Upload Date>",
  "status": "<Status>"
}
```

Response Example:

```
{
  "fileUploadId": 2,
  "fileName": "testing data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded"
}
```
Success Response Code: 200 OK
Error Response Code: 400 Bad Request (if file upload fails or file is missing)

Error Response Body Example:
```
{
  "error": "File is missing"
}
```

3.2 Get File Upload by ID

Endpoint: /api/v1/files/{id}
Method: GET
Description: Retrieves details of a file upload by its ID.
Path Parameters:

- id (required): The ID of the file upload.

Request Example:

- GET /api/v1/files/2

Response Format:
```
{
  "fileUploadId": <ID>,
  "fileName": "<File Name>",
  "fileType": "<File Type>",
  "uploadDate": "<Upload Date>",
  "status": "<Status>"
}
```

Response Example:
```
{
  "fileUploadId": 2,
  "fileName": "testing data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded"
}
```
Success Response Code: 200 OK
Error Response Code: 404 Not Found (if file ID does not exist)

Error Response Body Example:
```
{
  "error": "File not found"
}
```

3.3 Update File Status

Endpoint: /api/v1/files/{id}/status
Method: PATCH
Description: Updates the status of a file upload.
Path Parameters:

- id (required): The ID of the file upload.

Request Parameters:

- status (required): The new status of the file upload.

Request Example:

- PATCH /api/v1/files/2/status?status=Processed

Response Format:
```
{
  "fileUploadId": <ID>,
  "fileName": "<File Name>",
  "fileType": "<File Type>",
  "uploadDate": "<Upload Date>",
  "status": "<Updated Status>"
}
```

Response Example:
```
{
  "fileUploadId": 2,
  "fileName": "testing data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Processed"
}
```
Success Response Code: 200 OK
Error Response Code: 400 Bad Request (if status is invalid)

Error Response Body Example:
```
{
  "error": "Invalid status"
}
```

3.4 **Get All File Uploads**

Endpoint: /api/v1/files
Method: GET
Description: Retrieves a list of all file uploads.

Request Example:

- GET /api/v1/files

Response Format:
```
[
  {
    "fileUploadId": <ID>,
    "fileName": "<File Name>",
    "fileType": "<File Type>",
    "uploadDate": "<Upload Date>",
    "status": "<Status>"
  },
  ...
]
```

Response Example:

```
[
  {
    "fileUploadId": 1,
    "fileName": "example1.txt",
    "fileType": "text/plain",
    "uploadDate": "2024-08-17T14:22:00.000000000",
    "status": "Uploaded"
  },
  {
    "fileUploadId": 2,
    "fileName": "testing data.txt",
    "fileType": "text/plain",
    "uploadDate": "2024-08-18T16:42:04.956895100",
    "status": "Uploaded"
  }
]
```

Success Response Code: 200 OK
Error Response Code: 500 Internal Server Error (if there is an issue retrieving data)

3.5 Get Processed Results by File Upload ID

Endpoint: /api/v1/processed-results/file/{fileUploadId}
Method: GET
Description: Retrieves the processed results for a given file upload ID.
Path Parameters:

- fileUploadId (required): The ID of the file upload.

Request Example:

- GET /api/v1/processed-results/file/2

Response Format:

```
{
  "fileUploadId": <ID>,
  "fileName": "<File Name>",
  "fileType": "<File Type>",
  "uploadDate": "<Upload Date>",
  "status": "<Status>",
  "processedData": [
    {
      "dataKey": "<Parameter>",
      "count": <Count>,
      "averageValue": "<Average Value>",
      "minValue": "<Minimum Value>",
      "maxValue": "<Maximum Value>"
    },
    ...
  ]
}
```

Response Example:

```
{
  "fileUploadId": 2,
  "fileName": "testing data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded",
  "processedData": [
    {
      "dataKey": "Precipitation",
      "count": 2,
      "averageValue": "0.50mm",
      "minValue": "0.5mm",
      "maxValue": "0.5mm"
    },
    {
      "dataKey": "Temperature",
      "count": 2,
      "averageValue": "22.50°C",
      "minValue": "22.5°C",
      "maxValue": "22.5°C"
    },
    {
      "dataKey": "Humidity",
      "count": 2,
      "averageValue": "55%",
      "minValue": "55%",
      "maxValue": "55%"
    },
    {
      "dataKey": "WindSpeed",
      "count": 2,
      "averageValue": "12.00km/h",
      "minValue": "12.0km/h",
      "maxValue": "12.0km/h"
    },
    {
      "dataKey": "WindDirection",
      "count": 5,
      "averageValue": "NE",
      "minValue": "N/A",
      "maxValue": "N/A"
    },
    {
      "dataKey": "Pressure",
      "count": 2,
      "averageValue": "1015.00hPa",
      "minValue": "1015.0hPa",
      "maxValue": "1015.0hPa"
    }
  ]
}
```

Success Response Code: 200 OK

Error Response Code: 404 Not Found (if file upload ID does not exist)

Error Response Body Example:
```
{
  "error": "Processed results not found"
}
```

4. Error Handling Procedures

4.1 Common Error Response Format

Format:
```
{
  "error": "<Error Message>"
}
```

Example:
```
{
  "error": "File is missing"
}
```

4.2 Specific Error Cases

1. File Upload Errors
Missing File
HTTP Status Code: 400 Bad Request

Response Body Example:
```
{
  "error": "File is missing"
}
```

Invalid File Format
HTTP Status Code: 415 Unsupported Media Type
Response Body Example:
```
{
  "error": "Unsupported file format"
}
```

2. File Retrieval Errors
File Not Found
HTTP Status Code: 404 Not Found

Response Body Example:
```
{
  "error": "File not found"
}
```

```
3. Status Update Errors
Invalid Status
HTTP Status Code: 400 Bad Request

Response Body Example:
{
  "error": "Invalid status"
}
```

4. Processed Results Retrieval Errors
Processed Results Not Found
HTTP Status Code: 404 Not Found

Response Body Example:
```
{
  "error": "Processed results not found"
}
```

```
5. General Server Errors
Internal Server Error
HTTP Status Code: 500 Internal Server Error

Response Body Example:
{
  "error": "An unexpected error occurred"
}
```

5. Example Usage

5.1 Upload File Example

Request:

- **URL:** /api/v1/files/upload
- **Method:** POST
- **Headers:**
    - Content-Type: multipart/form-data
- **Body:**
    - Form-data with field file (upload a .txt file)

Response:

- **HTTP Status Code:** 200 OK
- **Response Body Example:**

```
{
  "fileUploadId": 1,
  "fileName": "example_data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded",
  "processedData": [
    {
      "dataKey": "Temperature",
      "count": 5,
      "averageValue": "22.75°C",
      "minValue": "18.3°C",
      "maxValue": "25.1°C"
    },
    {
      "dataKey": "Humidity",
      "count": 5,
      "averageValue": "60%",
      "minValue": "50%",
      "maxValue": "70%"
    },
    {
      "dataKey": "Pressure",
      "count": 5,
      "averageValue": "1012.0hPa",
      "minValue": "1009hPa",
      "maxValue": "1016hPa"
    },
    {
      "dataKey": "WindSpeed",
      "count": 5,
      "averageValue": "11.0km/h",
      "minValue": "5km/h",
      "maxValue": "15km/h"
    },
    {
      "dataKey": "WindDirection",
      "count": 5,
      "averageValue": "NE",
      "minValue": "N/A",
      "maxValue": "N/A"
    },
    {
      "dataKey": "Precipitation",
      "count": 5,
      "averageValue": "0.50mm",
      "minValue": "0.0mm",
      "maxValue": "1.2mm"
    }
  ]
}
```

5.2 Get File Upload by ID Example

Request:

- **URL:** /api/v1/files/{id}
- **Method:** GET

Response:

- **HTTP Status Code:** 200 OK

- **Response Body Example:**

```
{
  "fileUploadId": 1,
  "fileName": "example_data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Uploaded"
}
```

5.3 Update File Status Example

Request:

- **URL:** /api/v1/files/{id}/status
- **Method:** PATCH
- **Headers:**
    - Content-Type: application/x-www-form-urlencoded
- **Body:**
    - Form-data with field status (e.g., Processed)

Response:

- **HTTP Status Code:** 200 OK
- **Response Body Example:**

```
{
  "message": "File status updated successfully"
}
```

5.4 Get Processed Results by File Upload ID Example

Request:

- **URL:** /api/v1/processed-results/file/{fileUploadId}
- **Method:** GET

Response:

- **HTTP Status Code:** 200 OK
- **Response Body Example:**

```
{
  "fileUploadId": 1,
  "fileName": "example_data.txt",
  "fileType": "text/plain",
  "uploadDate": "2024-08-18T16:42:04.956895100",
  "status": "Processed",
  "processedData": [
    {
      "dataKey": "Temperature",
      "count": 5,
      "averageValue": "22.75°C",
      "minValue": "18.3°C",
      "maxValue": "25.1°C"
    },
    {
      "dataKey": "Humidity",
      "count": 5,
      "averageValue": "60%",
      "minValue": "50%",
      "maxValue": "70%"
    },
    {
      "dataKey": "Pressure",
      "count": 5,
      "averageValue": "1012.0hPa",
      "minValue": "1009hPa",
      "maxValue": "1016hPa"
    },
    {
      "dataKey": "WindSpeed",
      "count": 5,
      "averageValue": "11.0km/h",
      "minValue": "5km/h",
      "maxValue": "15km/h"
    },
    {
      "dataKey": "WindDirection",
      "count": 5,
      "averageValue": "NE",
      "minValue": "N/A",
      "maxValue": "N/A"
    },
    {
      "dataKey": "Precipitation",
      "count": 5,
      "averageValue": "0.50mm",
      "minValue": "0.0mm",
      "maxValue": "1.2mm"
    }
  ]
}
```

6. Error Handling

6.1 Error Response Format

All error responses will follow the structure below:

Response Format:

```
{
  "error": {
    "status": <HTTP Status Code>,
    "message": "<Error Message>"
  }
}
```

Example Error Response:

- **HTTP Status Code:** 400 Bad Request
  ```
  {
    "error": {
  ```

```
    "status": 400,
    "message": "Invalid file format. Only text files are allowed."
  }
}
```

- **HTTP Status Code:** 404 Not Found
  ```
  {
    "error": {
      "status": 404,
      "message": "File with the given ID was not found."
    }
  }
  ```

- **HTTP Status Code:** 500 Internal Server Error
  ```
  {
    "error": {
      "status": 500,
      "message": "An unexpected error occurred. Please try again
  later."
    }
  }
  ```

6.2 Error Handling Procedures

- **Invalid File Format:** If the uploaded file is not a text file, return a 400 Bad Request status with an appropriate error message.
- **File Not Found:** If the file with the provided ID does not exist, return a 404 Not Found status with an appropriate error message.
- **Processing Errors:** If any error occurs during the processing of the file, return a 500 Internal Server Error status with a generic error message to avoid exposing internal details.