API Documentation

Overview

This API handles CSV file uploads with authentication, dynamic throttling, and circuit breaker protection for external services.

Base URL

```
http://localhost:3000
```

Authentication

This API uses **Basic Authentication**. To make requests, provide a username and password:

- **Username**: Defined in `BASIC_AUTH_USERNAME` (default: `default-username`)
- **Password**: Defined in `BASIC_AUTH_PASSWORD` (default: `default-password`)

Use the following authorization header:

```
Authorization: Basic <base64_encoded_username:password>
```

Endpoints

1. Upload a CSV File

POST `/upload`

Uploads a CSV file. The file is processed asynchronously, and the request is queued to respect the concurrency limit. This endpoint requires **Basic Authentication**.

Request

- **Headers**:
  - `Authorization`: Basic Auth credentials
- **Body** (multipart form-data):
  - `file`: CSV file

Responses

- **200 OK**: File uploaded successfully

  ```
  {
    "message": "File uploaded successfully",
    "filename": "uploaded-file.csv"
  }
  ```

- **400 Bad Request**: No file uploaded

  ```
  {
    "message": "No file uploaded"
  }
  ```

- **500 Internal Server Error**: Invalid file type (non-CSV file)

  ```
  {
    "message": "Only CSV files are allowed!"
  }
  ```

---

2. Health Check

GET `/health`

Returns the system's health status, including CPU usage, free memory, and the health status of external services.

Response

- **200 OK**:

  ```
  {
    "status": "healthy",
    "cpuUsage": "15.2%",
    "freeMemory": "45.3%",
    "externalServices": {
      "database": "healthy",
      "anotherAPI": "healthy"
    }
  }
  ```

- **500 Internal Server Error**: Health check failure

  ```
  {
    "status": "unhealthy",
    "error": "An issue occurred while checking the system status."
  }
  ```

---

3. External Service Check

GET `/health-check`

Checks the health of an external service using a **Circuit Breaker** mechanism, which retries requests and prevents excessive failures.

Response

- **200 OK**:

  ```
  {
    "status": "healthy",
    "externalServiceData": "some external data"
  }
  ```

- **500 Internal Server Error**:

  ```
  {
    "status": "unhealthy",
    "error": "External service is unavailable"
  }
  ```

Middleware & Features

1. Basic Authentication Middleware

All endpoints (except health checks) require **Basic Authentication** using the credentials defined in the environment variables (`BASIC_AUTH_USERNAME` and `BASIC_AUTH_PASSWORD`). If authentication fails, the response will be:

- **401 Unauthorized**:

  ```
  {
    "message": "Unauthorized"
  }
  ```

2. Dynamic Throttling

Requests may be throttled based on system resources:

- If **CPU usage exceeds 80%** or **free memory is less than 20%**, the API will return:
  - **429 Too Many Requests**:

    ```
    {
      "message": "System is under high load. Please try again later."
    }
    ```

3. Request Concurrency Limit

- The system limits concurrent file uploads to **5 requests** at a time. Additional requests will be queued and processed once the system is ready to handle them.
- If too many requests are made simultaneously, the server will delay processing to ensure that the system remains stable.

Performance & Load Handling

- **File Size Limit**: 250MB per upload
- **Concurrency Limit**: Maximum of 5 concurrent requests for `/upload`
- **Dynamic Throttling**: Limits requests if CPU usage > 80% or free memory < 20%
- **Retry Mechanism**: External API calls are retried up to 5 times with exponential backoff.
- **Circuit Breaker**: Prevents excessive failures by blocking requests to external services if failure rate exceeds 50%

Error Handling

- **400 Bad Request**: Missing required data (e.g., no file uploaded)

```
{
  "message": "No file uploaded"
}
```

- **401 Unauthorized**: Invalid authentication credentials

```
{
  "message": "Unauthorized"
}
```

- **429 Too Many Requests**: System under high load, throttling in place

```
{
  "message": "System is under high load. Please try again later."
}
```

- **500 Internal Server Error**: Unexpected system errors

```
{
  "message": "Unknown error: <error_message>"
}
```