Logo

Description automatically generated

File Upload API

Version 1.0

File Upload API

Dated: 04-03-2025

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ver. No | Date of Release | Prepared By | Reviewed/ Approved By | List of changes from Previous Version |
| 1.0 | 04-03-2025 | MD. MEHEDI HASSAN |  | Document created. |

Table of Contents

[1 File Upload API 4](#_Toc192002826)

[1.1 Multi-File Type Support 4](#_Toc192002827)

[1.2 File Size Validation 5](#_Toc192002828)

[1.3 Secure File Storage 6](#_Toc192002829)

[1.4 Efficient File Upload Handling 7](#_Toc192002830)

[1.5 Error Handling and Feedback 8](#_Toc192002831)

[1.6 File Retrieval and Management 9](#_Toc192002832)

[1.7 Postman Integration for Testing 10](#_Toc192002833)

[1.8 API Documentation 11](#_Toc192002834)

# File Upload API

## Multi-File Type Support

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.1 Multi-File Type Support | | |
| Requirement Type | Functional | | |
| Description/Business Logic | | | |
| The API must accept and process multiple file types, including images **(JPEG, PNG, GIF)** and documents **(PDF, DOCX, TXT).**  When a file is uploaded, the backend should verify its MIME type and extension to ensure it meets the supported criteria.  Upon receiving a valid file, the system should proceed with the upload and return a success response containing file metadata (e.g., filename, file type, size, and a unique identifier).  If an unsupported file type is detected, the API should immediately return a descriptive error message indicating the allowed file formats | | | |
| API | |  |
| Clarification | | N/A |
| Dependency | | FastAPI/Flask, File Upload Libraries |
| Data Definition | | |
| **Accepted File Types:** image/jpeg, image/png, image/gif, application/pdf, application/vnd.openxmlformats-officedocument.wordprocessingml.document, text/plain  **Response:** JSON object with success status and file metadata | | |

## File Size Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.2. File Size Validation | | |
| Requirement Type | Functional | | |
| Description/Business Logic | | | |
| The API must validate the file size before storing it to ensure efficient resource usage and prevent overloading the system.  When a file is uploaded, the system should check its size against the maximum limit of **5MB**.  If the file size is within the allowed range, the upload process continues; otherwise, the system returns an error message specifying the file size limit.  This validation occurs at the initial upload stage to minimize processing time for invalid files. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | CORS Middleware, File Handling Modules |
| Data Definition | | |
| **Max File Size:** 5MB  **Response:** JSON object with error message if size exceeds the limit | | |

## Secure File Storage

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.3. Secure File Storage | | |
| Requirement Type | Non-Functional | | |
| Description/Business Logic | | | |
| All uploaded files must be securely stored in MongoDB Atlas using the GridFS system for optimal handling of large binary files.  Upon successful validation, each file is assigned a unique identifier to prevent naming conflicts and ensure accurate retrieval.  File metadata (e.g., upload time, MIME type, size) is stored alongside the file for easy reference and future audits.  Access to stored files should be restricted to authorized users, with additional security measures for sensitive data if required. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | MongoDB Atlas |
| Data Definition | | |
| **File Storage Path:** MongoDB GridFS  **Metadata:** Unique file ID, file type, upload timestamp | | |

## Efficient File Upload Handling

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.4. Efficient File Upload Handling | | |
| Requirement Type | Non-Functional | | |
| Description/Business Logic | | | |
| The API must efficiently handle multiple concurrent file uploads by leveraging asynchronous processing capabilities.  Large files should be processed in chunks to optimize memory usage and reduce the risk of server overload.  For each upload request, the API should support multiple files in a batch, with configurable limits (e.g., 5 files per request).  System performance should be monitored, and optimizations should be made to maintain responsiveness under high load. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | Asynchronous File Handling (FastAPI/Flask), MongoDB Atlas |
| Data Definition | | |
| **Concurrent Upload Limit:** Configurable (e.g., 5 files per request)  **Response:** JSON object with upload statuses for each file | | |

## Error Handling and Feedback

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.5. Error Handling and Feedback | | |
| Requirement Type | Functional | | |
| Description/Business Logic | | | |
| The API must provide detailed error messages and feedback when an upload fails due to issues such as:   * Unsupported file type * File size exceeding the limit * Internal server errors   Each error response should include an error code, a human-readable message, and a timestamp for troubleshooting.  The system should log all errors for monitoring and analysis, enabling quick identification of recurring issues.  Clear and consistent error messages enhance user experience by guiding users on how to resolve upload issues. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | Error Logging System, FastAPI/Flask Exception Handling |
| Data Definition | | |
| **Error Format:** JSON object with error code, message, and timestamp | | |

## File Retrieval and Management

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.6. File Retrieval and Management | | |
| Requirement Type | Functional | | |
| Description/Business Logic | | | |
| The API must allow users to retrieve uploaded files using their unique identifiers, ensuring accurate and fast access.  File retrieval should include both file content and associated metadata.  Users must be able to delete files when they are no longer needed, with safeguards against accidental deletions.  Additional endpoints should be available for querying file metadata, enabling search by upload date, type, or other criteria. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | MongoDB Atlas, RESTful Endpoint Implementation |
| Data Definition | | |
| **Query Parameters:** File ID, Metadata Fields  **Response:** File content, Metadata | | |

## Postman Integration for Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.5. Postman Integration for Testing | | |
| Requirement Type | Non-Functional | | |
| Description/Business Logic | | | |
| The API must be tested using Postman to validate all major functionalities, including:   * Successful and failed uploads * File retrieval and deletion * Edge cases (e.g., large files, concurrent uploads)   Postman test collections should be organized by endpoint and include various test cases to ensure thorough validation.  Regular testing must be conducted to identify regressions and verify performance optimizations. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | Postman, API Testing Framework |
| Data Definition | | |
| **Test Collection:** Organized by endpoint, request type, and expected response | | |

## API Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | 1.8. API Documentation | | |
| Requirement Type | Non-Functional | | |
| Description/Business Logic | | | |
| The API documentation must provide clear instructions on how to interact with each endpoint.  Each endpoint should include:   * URL structure * HTTP methods (GET, POST, DELETE) * Request parameters and body schema * Expected responses and error codes   The documentation should follow the OpenAPI (Swagger) format to facilitate automatic generation and maintenance. | | | |
| API | | Method:  URL:  Request: |
| Clarification | | N/A |
| Dependency | | OpenAPI Specification (Swagger), FastAPI/Flask AutoDoc Tools |
| Data Definition | | |
| **Document Sections:** Endpoint Details, Request/Response Schema, Error Codes | | |