



COLLEGE CODE : 8203

COLLEGE NAME :A.V.C College Of Engineering

DEPARTMENT :B.Tech-Information Technology

STUDENTNM_ID : 50186AA220AC61191B545EC1FD3CC1DF

ROLL NO :23IT74

DATE : 15-09-2025

**Completed the project named as Phase II
technology project name :File Upload Manager**

**SUBMITTED BY,
NAME :Peramani B**

MOBILE NO:8610226879

Technical Stack

Frontend

- **HTML, CSS, JavaScript** → For building the basic user interface and interactivity.
- **Bootstrap / Tailwind CSS** → Provides modern, responsive styling for file upload forms and file lists.
- **Axios** → Handles API calls between frontend and backend for uploading, fetching, and deleting files.

Backend

- **Node.js** → Runtime environment for executing JavaScript on the server.
- **Express.js** → Web framework to handle API routes (upload, fetch, delete).
- **Multer** → Middleware to process multipart/form-data and manage file uploads.

Database & Storage

- **MongoDB** → Stores metadata like file name, type, size, and upload date.
- **GridFS** → Special MongoDB feature to store and retrieve large files (>16MB) in chunks.
- **Local Storage (Fallback)** → Stores smaller files directly on the server.

Security

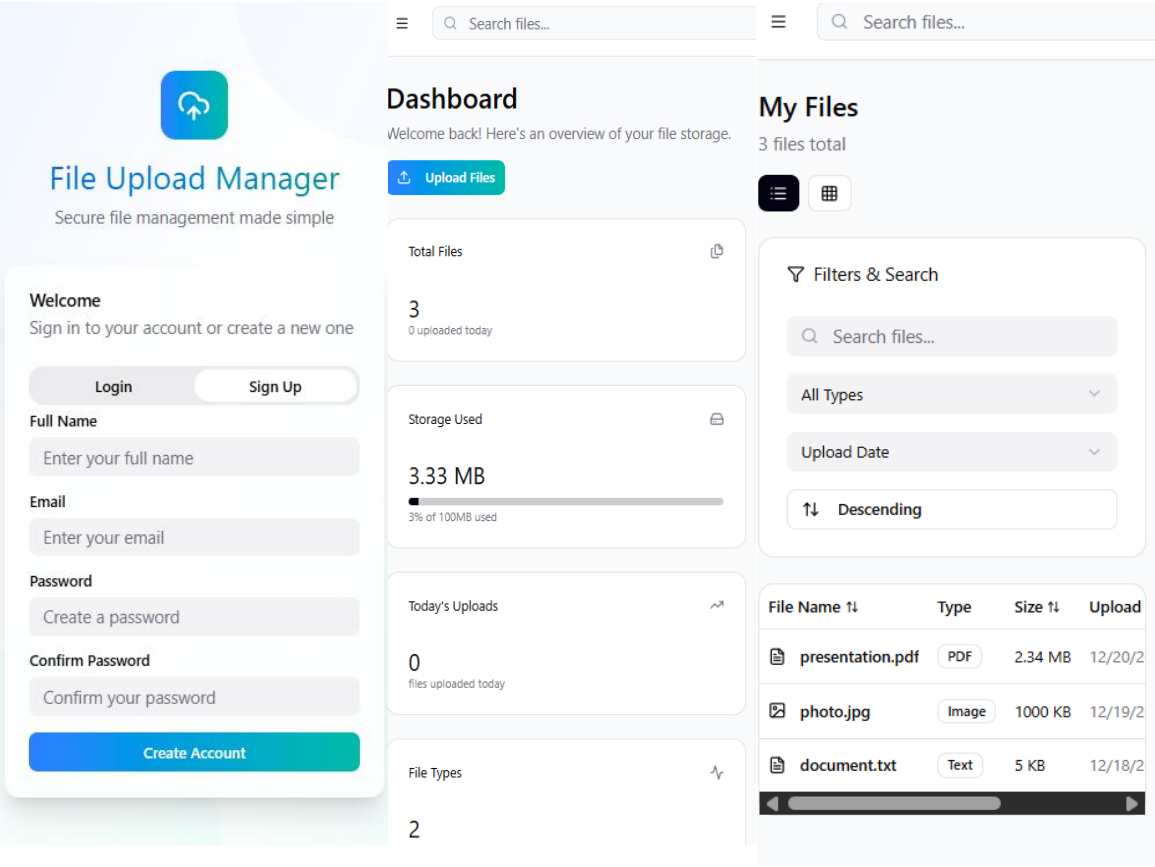
- **File Validation** → Ensures only allowed file types and sizes are uploaded.
- **Helmet.js** → Adds secure HTTP headers to protect the backend from common attacks.

Deployment & DevOps

- **GitHub** → Version control for source code.
- **MongoDB Atlas** → Cloud-hosted MongoDB database.
- **Heroku / Render** → Hosting platforms for deploying the backend and frontend.

UI Structure & API Schema Design

UI Structure:



Email

Enter your email

Password

Enter your password

Sign Up

Reset Password

Login

My Files

Your uploaded files

Document1.pdf

PDF

2.5 MB

Image1.jpg

JPG

1.2 MB

My Files

Settings

Upload File

Search files...

Upload Files

Drag and drop your files here or click to browse

Drop files here

or click to select files from your computer

Choose Files

Upload Guidelines:

Maximum file size: 10 MB. Allowed formats: JPG, PNG, GIF, PDF, TXT, ZIP

Recent Uploads

Your recently uploaded files

No recent uploads

API Schema Design:

1. Upload File

Endpoint:

POST /api/files/upload

Description: Upload a new file.

Request:

- multipart/form-data → file
 - Validation → check file type & size
- Response (Success):

```
{
  "status": "success",
  "fileId": "64b1a2f4c9a1",
  "fileName": "document.pdf",
  "fileType": "application/pdf",
  "fileSize": "2MB",
  "uploadDate": "2025-09-25",
  "fileUrl": "/api/files/64b1a2f4c9a1"
}
```

2. Fetch File (Download)

Endpoint:

GET /api/files/:id

Description: Retrieve/download a file by its ID.

Response (Success): Returns file stream (binary).

Error Response:

```
{
  "status": "error",
  "message": "File not found"
}
```

3. Get File Metadata

Endpoint:

GET /api/files/:id/metadata

Description: Get details of a specific file.

Response:

```
{
  "fileId": "64b1a2f4c9a1",
  "fileName": "image.png",
  "fileType": "image/png",
  "fileSize": "1.2MB",
  "uploadDate": "2025-09-25"
}
```

4. Delete File

Endpoint:

DELETE /api/files/:id

Description: Delete a file by its ID.

Response (Success):

```
{
  "status": "success",
  "message": "File deleted successfully"
}
```

Error Response:

```
{
  "status": "error",
  "message": "File not found"
}
```

5. List All Files (Optional)

Endpoint:

GET /api/files

Description: Get all uploaded files with metadata.

Response:

```
[
  {
    "fileId": "64b1a2f4c9a1",
    "fileName": "report.docx",
    "fileType": "application/msword",
    "fileSize": "450KB",
    "uploadDate": "2025-09-25"
  },
  {
    "fileId": "64b1a2f4c9b2",
    "fileName": "photo.jpg",
    "fileType": "image/jpeg",
    "fileSize": "3MB",
    "uploadDate": "2025-09-25"
  }
]
```

Components & Module Diagram

Components

1. Frontend (User Interface)

- **Upload Component**→ File selector + drag-and-drop zone + upload button.
- **File List Component**→ Displays uploaded files with metadata.
- **Action Buttons**→ Download, Delete, View Metadata.

- **Notification/Alert Component**→ Success & error messages.
- **Search/Filter Component** (optional) → Search files by name/type/date.

2. Backend (Server-Side)

- **API Controller**→ Handles requests for upload, fetch, delete, metadata.
- **Multer Middleware**→ Processes file uploads (multipart/form-data).
- **Validation Module**→ Checks file type & size before saving.
- **File Service Layer**→ Logic for storing, retrieving, deleting files.
- **Response Handler**→ Standardizes success/error JSON responses.

3. Database & Storage

- **MongoDB (Metadata Collection)**→ Stores file details (ID, name, type, size, upload date).
- **GridFS Storage**→ Stores large files in chunks (>16MB).
- **Local Storage (Optional)**→ Stores small files in server directory.

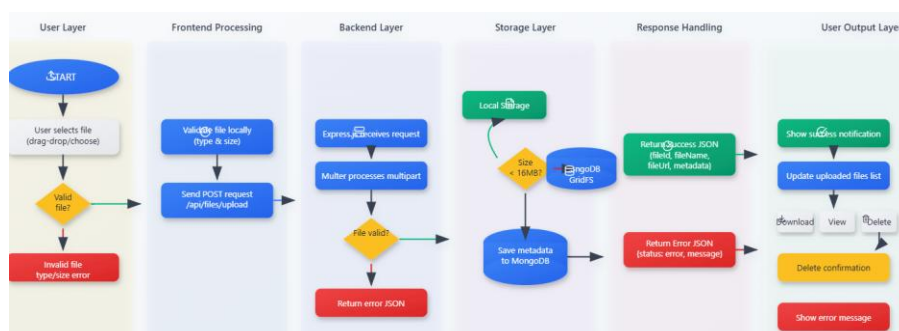
4. Security Layer

- **Validation & Sanitization**→ Prevent malicious uploads.
- **File Restrictions**→ Allowed types & size limits.
- **Access Control (Future Scope)**→ Role-based permissions for admin vs normal users.

5. Deployment & DevOps

- **Version Control (GitHub/GitLab)**→ Source code management.
- **Hosting (Heroku/Render/Vercel)**→ Backend & frontend deployment.
- **MongoDB Atlas**→ Cloud-hosted database.
- **CI/CD Pipeline (Optional)**→ Automated testing & deployment.

Module Diagram:



Basic FlowDiagram

