COLLEGE CODE : 8203

COLLEGE NAME : A.V.C. COLLEGE OF ENGINEERING, MANNAMPANDAL

DEPARTMENT : CSE

STUDENT NM-ID : C1E491D86B20DEA9A234F6AB87816599

ROLL NO : 820323104059

DATE : 22.09.2025

Completed the project named as Phase 3

TECHNOLOGY PROJECT NAME: FILE UPLOAD MANAGER

SUBMITTED BY,

NAME: S.MANGAIYARKARASI

MOBILE NO: 8122584133

PHASE 3 : MVP IMPLEMENTATION

Project setup

Runtime & framework

Backend built with Node.js and Express.js to provide RESTful APIs.

Key dependencies

- Multer(or multer - gridfs-storage)for handling multipart/form-data upload
- Mongoose (or native MongoDB driver) for database connectivity.
- GridFS for storing files in MongoDB when files may be large.
- Nodemon (dev) for auto-restart during development.

Environment configuration

Use environment variables for secrets and configuration: MONGO_URI, PORT, MAX_FILE_SIZE, ALLOWED_MIME_TYPES.

- Folder structure (recommended)
- index.js or app.js — express bootstrap
- routes/ — API routes (e.g., files.js)
- controllers/ — handlers for routes
- services/ — storage logic (GridFS adapters, local fs)

- models/ — metadata schemas (if storing metadata separate from GridFS)

- tests/ — unit & integration tests

Documentation

README with setup steps, how to run (dev & prod), example requests, and Postman collection.

Core Features Implementation

The following core features were implemented during this phase:

File Upload:
Users can upload files of multiple formats. Validation rules
such as file size and supported file types were added.

File Retrieval:
Files can be downloaded or previewed directly from the server.

File Deletion:
Users can delete files from the system when no longer required.

Metadata Management:
Along with the file itself, metadata such as
filename, file type, size, and upload date is stored for easy tracking.

Data Storage Approach

Two approaches were used for storing files:

Local Storage:

In the early stages of testing, files were stored directly on the server. This provided quick feedback and easier debugging. However, this approach is not suitable for long-term scalability.

Database Storage with GridFS:

MongoDB's GridFS was used to store larger files by splitting them into smaller chunks. This ensures the system can handle files exceeding

Testing Core Features

Testing was carried out to validate the reliability of the system.

Upload Testing:
Verified with text files, images, PDFs, and videos. Ensured that oversized and unsupported files were rejected.

Download Testing:
Confirmed that uploaded files could be retrieved correctly with the right format and metadata.

Deletion Testing:
Tested file removal from both local storage and GridFS database.

Error Handling:

Scenarios such as duplicate file names, invalid requests, and missing files were tested to ensure proper error messages and status codes.

Tools Used: Postman was used to send API requests and verify responses.

Version Control and Collaboration

- To ensure proper tracking and collaboration, the project was managed using GitHub.
- Frequent commits were made after implementing new features.
- Branching was used to separate development work from the main branch.
- Pull requests and code reviews ensured clean and maintainable code.
- The repository acted as a backup and reference for all team members.

GITHUB REPOSITORY LINK:

https://github.com/Mangai-11/NM-projects.git