COLLEGE CODE :9222

COLLEGE NAME :Theni Kammavar Sangam College Of Technology

DEPARTMENT :INFORMATION TECHNOLOGY

STUDENT NM-ID :18B6A90903386AF88EE47B5C70D5FIES

ROLL NO :24LIT01

DATE :3/10/2025

Completed the project named as Phase 3

TECHNOLOGY PROJECT NAME :

FILE UPLOAD MANAGER

SUBMITTED BY,

NAME : RAJALAXMI.P

MOBILE NO : 8072622588

MVP Implementation

Project Setup

The initial step involves preparing the development environment and defining the project architecture.

- **Technology Stack:** Node.js (server-side), Express.js (API handling), and a cloud service (AWS S3 / Google Cloud Storage / Azure Blob).
- **Environment Setup:** Configuring Node.js, package manager (npm/yarn), and essential dependencies for file upload and cloud integration.
- **Folder Structure:** Organizing the project into modules (routes, controllers, services, config, and tests) for maintainability.
- **Configuration:** Setting up environment variables for API keys, cloud storage credentials, and local testing.

Project Structure

```
FileUploadManager/
├── node_modules/          # Installed dependencies (auto-created)
├── uploads/               # Local file storage (if not using cloud)
├── src/
│   ├── routes/            # API route files
│   │   └── upload.js      # File upload routes (Multer/Cloud upload logic)
│   │
│   ├── config/            # Configuration files
│   │   └── cloudConfig.js # AWS/GCP/Azure setup
│   │
│   ├── controllers/       # Business logic
│   │   └── uploadController.js
│   │
│   ├── middlewares/       # Middleware (e.g., authentication, error handling)
│   │   └── progressMiddleware.js
│   │
│   ├── utils/             # Helper functions
│   │   └── logger.js
│   │
│   ├── app.js             # Main Express app setup
│   └── server.js          # Entry point (starts the server)
│
├── .env                   # Environment variables (keys, secrets)
├── .gitignore             # Files to ignore in GitHub (node_modules, .env, etc.)
├── package.json           # Project metadata & dependencies
└── README.md              # Project documentation (for GitHub)
```

Core Features Implementation

The central part of the MVP focuses on enabling file uploads and progress tracking.

- **File Upload Handling:** Using middleware (like `multer` or `busboy`) to manage incoming files.
- **Progress Tracking:** Implementing real-time upload status using event streams or WebSockets (percentage, upload speed, estimated completion).
- **Cloud Integration:** Uploading files directly to cloud storage services for reliability and scalability.
- **Error Handling:** Graceful handling of failed uploads with retry/pause/resume support.
- **User Interface (if included):** A simple dashboard or API response showing upload status and cloud storage link.

Data Storage (Local State / Database)

To manage uploaded files and track metadata:

- **Local State:** Temporary storage for active uploads, including file name, size, user ID, and progress.
- **Database (SQL/NoSQL):** Persistent storage for metadata such as file path/URL, upload timestamp, status, and user ownership.
- **Cloud Metadata Sync:** Ensuring cloud storage details (like object key or public URL) are stored for retrieval.

Testing Core Features

Testing ensures stability and reliability of the MVP.

- **Unit Testing:** Verify file upload logic, progress calculation, and database operations.
- **Integration Testing:** Test API endpoints for end-to-end upload workflow.
- **Cloud Upload Testing:** Validate that files are correctly stored and retrieved from the cloud service.
- **Performance Testing:** Assess handling of large files and concurrent uploads.
- **Error Simulation:** Test interruptions (e.g., network failure) to ensure resume/retry features work.

Version Control (GitHub)

To maintain collaboration and version history:

- **Repository Setup:** A dedicated GitHub repository for source code and documentation.
- **Branching Strategy:** Use main branch for stable releases and feature branches for development.
- **Commit Practices:** Frequent commits with meaningful messages to track development progress.
- **Collaboration:** Pull requests and code reviews for quality assurance.
- **Documentation:** Using README and Wiki to explain project setup, usage, and contribution guidelines.