A

Project Report

On

File transfer web application

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Harshit Upreti	**2262011**
Kanishka Nainwal	**2261303**
Muskesh Singh Rawat	**2261377**
Shobha Barti	**2261530**

Under the Guidance of

Mr.Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI, DISTRICT-

NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Harshit Upreti and group** hereby declare the work, which is being presented in the project, entitled '**File transfer web application** ' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of our work carried out under the supervision of **Mr. Anubhav Bewerwal**.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Harshit Upreti

Kanishka Nainwal

Mukesh Singh Rawat

Shobha Barti

CERTIFICATE

The project report entitled "File transfer web application" being submitted by Harshit Upreti (2262011) S/o R.B. Upreti, Kanishka Nainwal (2261303) D/o Mr. Harish Chandra Nainwal , Muskesh Singh Rawat (2261377) S/o Mr. Jai Singh Rawat, Shobha Barti (2261530) D/o Mr. S.S.Barti of B.Tech.(CSE)to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them.They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Anubhav Bewerwal **Dr. Ankur Singh Bisht**

(Project Guide) **(Head, CSE)**

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director**'Prof. (Col.)Anil Nair (Retd.)',** GEHU Bhimtal Campusto permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president**'Prof. (Dr.) Kamal Ghanshala'** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **'Dr. Ankur Singh Bisht'** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide**'Anubhav Bewerwal'**(Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus)and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Harshit Upreti 2262011

Kanishka Nainwal 2261303

Muskesh Singh Rawat 2261377

Shobha Barti 2261530

Abstract

The proposed project **"File transfer web application"** aims to develop a robust and userfriendly File Transfer Web Application utilizing Node.js and MongoDB, grounded in core operating system (OS) concepts such as file handling, memory management, process synchronization, and access control. This application is designed to facilitate secure, efficient, and scalable file sharing and storage for users, supporting features like public and private file sharing, real-time file management, cloud storage integration, and monetization through tiered storage plans.

The system leverages Node.js for non-blocking I/O operations and Express.js for API management, ensuring high performance and scalability. MongoDB serves as the NoSQL database for storing user data and file metadata, enabling fast access and search. OS-level principles are employed to implement effective file system management (using streams), access control (role-based authentication using JWT), and resource optimization (through caching, queuing, and concurrency control).

Key functionalities include user registration and login with hashed password security, file uploads with metadata generation, folder and subfolder hierarchy support, file sharing via unique links or email-based permissions, download tracking, and a recycle bin for safe deletions. The system also integrates Stripe and PayPal for premium storage upgrades and includes a content management feature for blog publishing.

The approach includes modular system design, RESTful API architecture, and optional Dockerbased deployment for scalability. Algorithms are designed for authentication, file upload/download, secure sharing, and payment handling. The proposed solution is highly feasible, both technically and economically, and can be deployed with minimal hardware while scaling efficiently with user demand.

This abstract outlines a complete end-to-end implementation plan suitable for academic, business, and real-world deployment, emphasizing security, usability, and extensibility.

.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

API : Application Programming Interface, Enables communication between different software components.
CSS: Cascading Style Sheets, Styles and formats the appearance of HTML elements.
CI/CD: Continuous Integration / Continuous Deployment, : Automates building, testing, and deployment processes.
DB: Database, Stores and manages structured and unstructured data.
EJS: Embedded JavaScript, Templating engine used to render dynamic HTML pages in Node.js.
GB: Gigabyte
HTML: HyperText Markup Language, Structures content on web pages.
HTTP: Hypertext Transfer Protocol, Protocol for data communication over the web.
I/O: Input/Output

IDE: Integrated Development Environment, Software for writing and debugging code efficiently.
JSON: JavaScript Object Notation, Format for exchanging data between client and server.
JS: JavaScript,
JWT:JSON Web Token, Used for securely authenticating users and sessions.
MFA: Multi-Factor Authentication, Security method requiring two or more verification steps.
MVC: Model-View-Controller, Design pattern for organizing application structure.
OS: Operating System,
RAM: Random Access Memory,
SMTP: Simple Mail Transfer Protocol, Protocol used for sending emails.
UI: User Interface, Visual components that users interact with in an application.
URL: Uniform Resource Locator, The web address used to access resources on the internet.
UX: User Experience, Overall usability and satisfaction of a user with the application.
VS Code: Visual Studio Code, code editor used for development.
VCS: Version Control System, Tracks and manages changes to code (e.g., Git).

INTRODUCTION

1.1 Prologue

In the digital age, seamless and secure data sharing is more critical than ever. With growing reliance on remote collaboration, cloud storage, and data-driven communication, individuals and organizations alike require a dependable platform to upload, manage, and share files efficiently. Traditional methods such as email attachments, USB drives, or third-party platforms often present limitations regarding file size, access control, and security.

This project aims to design and develop a **File Transfer Web Application** using Node.js and MongoDB that provides users with an intuitive and secure means to manage and share files. Leveraging key operating system (OS) concepts like file system handling, memory management, and access control, the system integrates performance optimization, secure authentication, and scalability as foundational principles. The application serves not only as a practical solution for file transfer but also as a demonstration of applying theoretical computing knowledge to realworld software engineering.

1.2 Background and Motivations

With the rapid increase in digital file exchange, there has been a growing demand for platforms that ensure secure file sharing while providing flexibility, scalability, and user control. Existing platforms such as Google Drive and Dropbox offer extensive features but often impose storage limits, require account creation for downloads, or involve complex pricing models.

Motivated by the need for a simplified, user-friendly solution with robust backend support, this project explores the development of a web application that allows users to share files of any type—publicly or privately—without sacrificing performance or security. Integrating features such as multi-level folders, download tracking, restoreable deletions, and monetized storage upgrades provides users with greater autonomy and data control. Furthermore, the project is structured to be cost-effective and easily deployable using common development tools and environments.

1.3 Problem Statement

In the current digital landscape, users frequently encounter challenges in managing and sharing files securely, efficiently, and with flexibility. Most existing cloud-based file transfer solutions impose limitations such as mandatory account creation for download access, restricted control over shared content, file compression, limited folder hierarchy support, and unclear monetization or storage expansion options. These constraints hinder seamless collaboration and user autonomy in academic, professional, and personal environments.

There is a clear need for a scalable, secure, and user-friendly web-based **file transfer web application** that overcomes these limitations. The system should allow users to upload files of any type and size, organize them in multi-level folders, and share them either publicly via links or privately via email. It should provide robust access control, download tracking, a recycle bin for recovery, backup options, and integration with payment gateways for premium storage upgrades. Moreover, the backend must leverage core operating system concepts such as file and memory

management, process handling, and access permissions to ensure high performance and data integrity.

This project addresses these needs by proposing the development of a full-stack File Transfer Web Application using Node.js and MongoDB, applying OS principles and modern web development practices to deliver a practical, cost-effective, and extensible solution.

1.4 Objectives and Research Methodology

The primary objective of this project is to design and implement a secure, scalable, and userfriendly file transfer web application using Node.js and MongoDB that enables users to upload, organize, share, and manage files efficiently, while integrating core OS concepts like file handling, memory management, and access control. Research Methodology :

- Literature Review: Studied existing file sharing platforms and OS concepts relevant to web development.
- Requirement Analysis: Defined system features, user needs, and technical constraints.
- System Design: Created architecture, database schema, and security flow using RESTful principles.
- Development: Implemented core features using Node.js, Express.js, and MongoDB with payment gateway and file operations.
- Testing: Conducted unit, integration, and performance testing to validate functionality.
- Deployment & Evaluation: Deployed on server, monitored performance, and refined based on feedback.

1.5 Project Organization

This report is organized to present a comprehensive overview of the File Transfer Web Application project in a structured manner. It begins with a literature survey covering existing file sharing platforms, cloud storage solutions, and relevant technologies such as Node.js, MongoDB, and core operating system concepts. The architecture section outlines the overall system design, including the client-server model, RESTful API structure, and folder/file management strategy. The implementation section describes the coding workflow, selected technology stack, key system modules such as upload, sharing, authentication, and payment integration. The testing and evaluation section provides insights into application performance, file handling efficiency, user access control, and system reliability. Finally, the conclusion summarizes key outcomes, discusses challenges encountered during development, and suggests possible improvements and future extensions for enhanced scalability and security.

PHASES OF SOFTWARE DEVELOPMENT CYCLE

2.1 Hardware Requirement

Specification	**Windows**	**macOS (OS X)**	**Linux**
Operating System	Microsoft Windows 8/10/11 (32 or 64 bit)	macOS 10.13 (High Sierra) or higher	Ubuntu (GNOME/KDE/Unity desktop) or similar
Processor	Minimum: Dual-core Intel i3 or equivalent Recommended: Quad-core Intel i5/i7 or Ryzen 5+	Minimum: Dual-core Intel i3 or equivalent Recommended: Quad-core Intel i5/i7 or Ryzen 5+	Minimum: Dual-core Intel i3 or equivalent Recommended: Quad-core Intel i5/i7 or Ryzen 5+
RAM	Minimum 4 GB, Recommended 8 GB	Minimum 4 GB, Recommended 8 GB	Minimum 4 GB, Recommended 8 GB
Storage	Minimum 10 GB free space	Minimum 10 GB free space	Minimum 10 GB free space
Development Tools	Node.js,MongoDB,npm, optionally VS Code	Node.js, MongoDB, npm, Xcode command line tools, optionally VS Code	Node.js, MongoDB, npm, GCC, optionally VS Code
Display	1280 x 800 resolution or higher	1280 x 800 resolution or higher	1280 x 800 resolution or higher
Network	Stable broadband internet connection	Stable broadband internet connection	Stable broadband internet connection
Notes	Supports 32/64 bit, requires latest updates	Requires macOS 10.13+ for best compatibility	Prefer Ubuntu 20.04+ or similar distros

2.2 Software Requirement

Sno.	**Name**	**Specifications**
1	Operating System	Windows/Linux/macOS
2	Programming Languages	JavaScript
3	Backend Framework	Node.js, Express.js
4	Frontend Framework	React.js / EJS / HTML/CSS
5	Database	MongoDB
6	File Upload Handling	Multer (Node.js middleware)
7	Authentication	JWT (JSON Web Tokens) / Passport.js
8	WebSocket Library	Socket.io
9	Version Control	Git and GitHub
10	Package Manager	npm or yarn
11	Deployment	Heroku / Vercel
12	CI/CD	Github Actions
13	Payment Integration	Stripe, PayPal APIs
14	Image Compression	Sharp (Node.js image processing)
15	Email Service	Nodemailer or any SMTP service

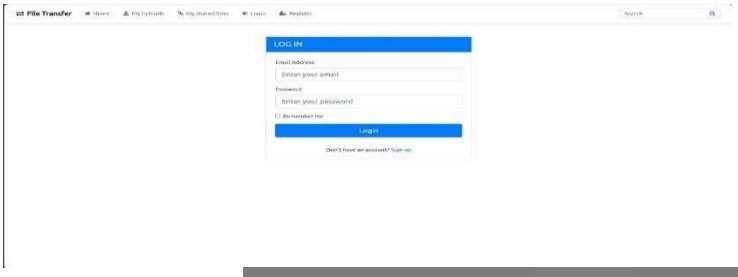
CODING OF FUNCTIONS

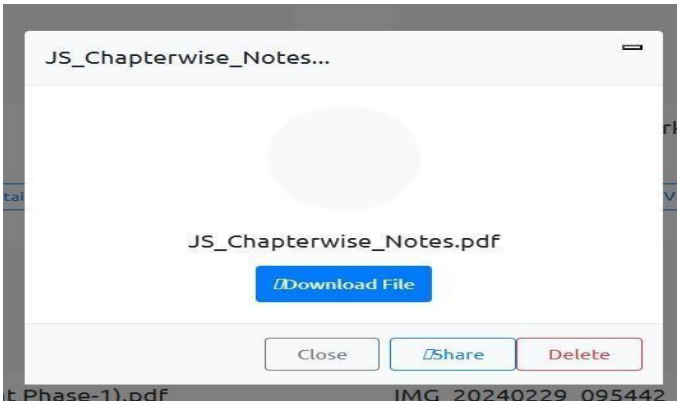
```
var express = require("express");
var app = express();

var formidable = require("express-formidable");
app.use(formidable());

var mongodb = require("mongodb");
var mongoClient = mongodb.MongoClient;
var ObjectId = mongodb.ObjectId;

var httpObj = require("http");
var http = httpObj.createServer(app);

var bcrypt = require("bcrypt");
var fileSystem = require("fs");

var session = require("express-session");
app.use(session({
    secret: 'secret key',
    resave: false,
    saveUninitialized: false
}));

app.use("/public/css", express.static(__dirname + "/public/css"));
app.use("/public/js", express.static(__dirname + "/public/js"));
app.use("/public/img", express.static(__dirname + "/public/img"));
app.use("/public/font-awesome-4.7.0", express.static(__dirname + "/public/font-awesome-
4.7.0"));
app.use("/public/fonts", express.static(__dirname + "/public/fonts"));

app.set("view engine", "ejs");
var mainURL = "http://localhost:3000";
var database = null;

app.use(function (request, result, next) {
    request.mainURL = mainURL;
    request.isLogin = (typeof request.session.user !== "undefined");
    request.user = request.session.user;
    next();
});

function recursiveGetFile (files, _id) {
    var singleFile = null;
    for (var a = 0; a < files.length; a++) {
        const file = files[a];
```

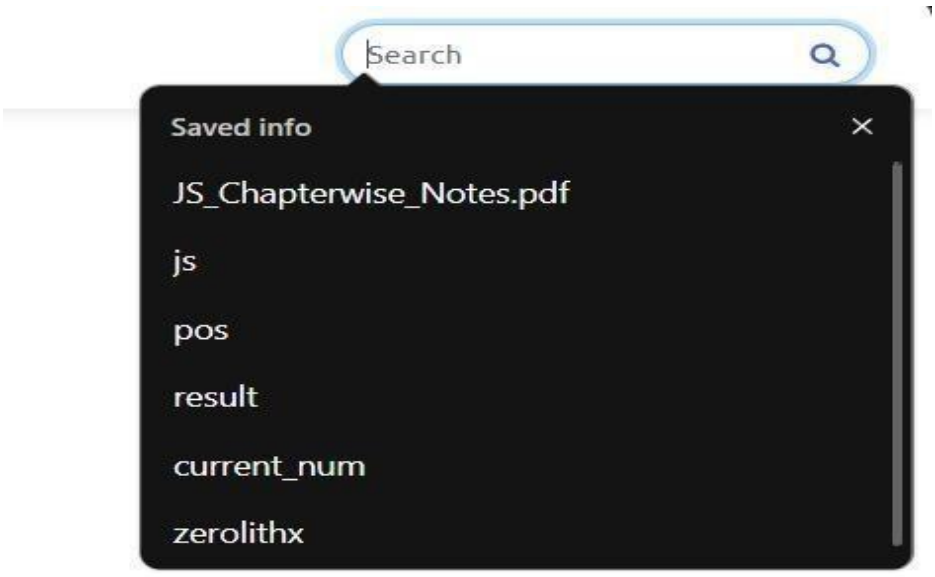
```
        if (file.type != "folder") {
            if (file._id == _id) {
                return file;
            }
        }
        if (file.type == "folder" && file.files.length > 0) {
            singleFile = recursiveGetFile(file.files, _id);
            if (singleFile != null) {
                return singleFile;
            }
        }
    }
}

function getUpdatedArray (arr, _id, uploadedObj) {
    for (var a = 0; a < arr.length; a++) {
        if (arr[a].type == "folder") {
            if (arr[a]._id == _id) {
                arr[a].files.push(uploadedObj);
                arr[a]._id = ObjectId(arr[a]._id);
            }
            if (arr[a].files.length > 0) {
                arr[a]._id = ObjectId(arr[a]._id);
                getUpdatedArray(arr[a].files, _id, uploadedObj);
            }
        }
    }
    return arr;
}

function removeFileReturnUpdated(arr, _id) {
    for (var a = 0; a < arr.length; a++) {
        if (arr[a].type != "folder" && arr[a]._id == _id) {
            try {
                fileSystem.unlinkSync(arr[a].filePath);
            } catch (exp) {}
            arr.splice(a, 1);
            break;
        }
        if (arr[a].type == "folder" && arr[a].files.length > 0) {
            arr[a]._id = ObjectId(arr[a]._id);
            removeFileReturnUpdated(arr[a].files, _id);
        }
    }
    return arr;
}
```

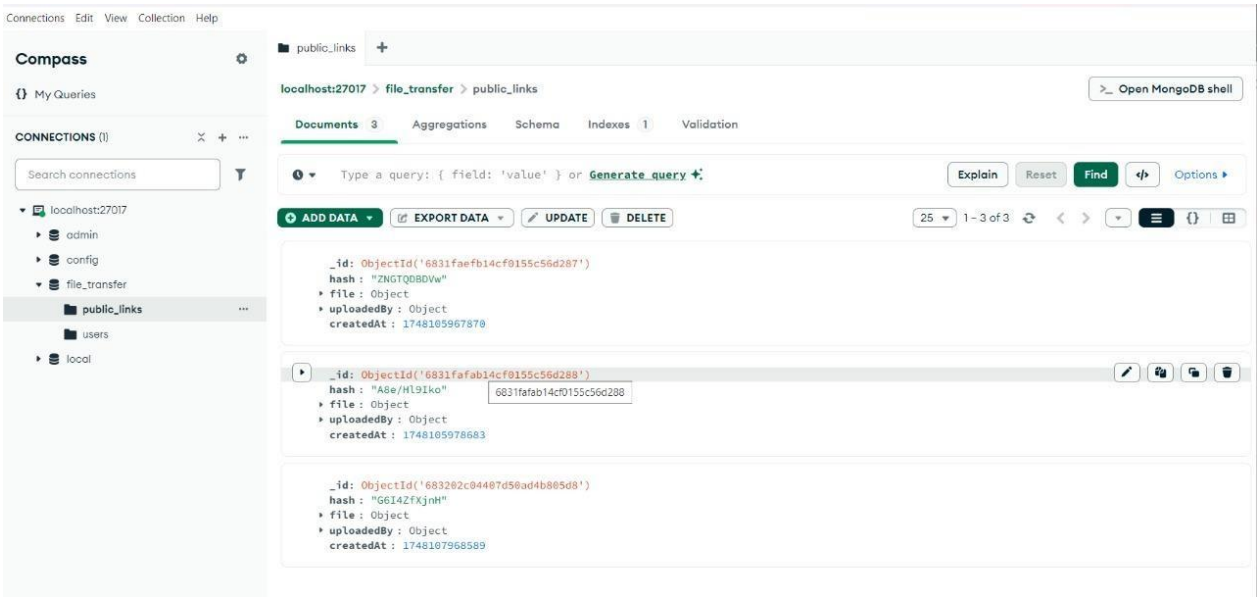
```
function recursiveSearch (files, query) {
    var singleFile = null;
    for (var a = 0; a < files.length; a++) {
        const file = files[a];
        if (file.type == "folder") {
            if (file.folderName.toLowerCase().search(query.toLowerCase()) > -1) {
                return file;
            }
            if (file.files.length > 0) {
                singleFile = recursiveSearch(file.files, query);
                if (singleFile != null) {
                    if (singleFile.type != "folder") {
                        singleFile.parent = file;
                    }
                    return singleFile;
                }
            }
        } else {
            if (file.name.toLowerCase().search(query.toLowerCase()) > -1) {
                return file;
            }
        }
    }
}
```

SNAPSHOTS

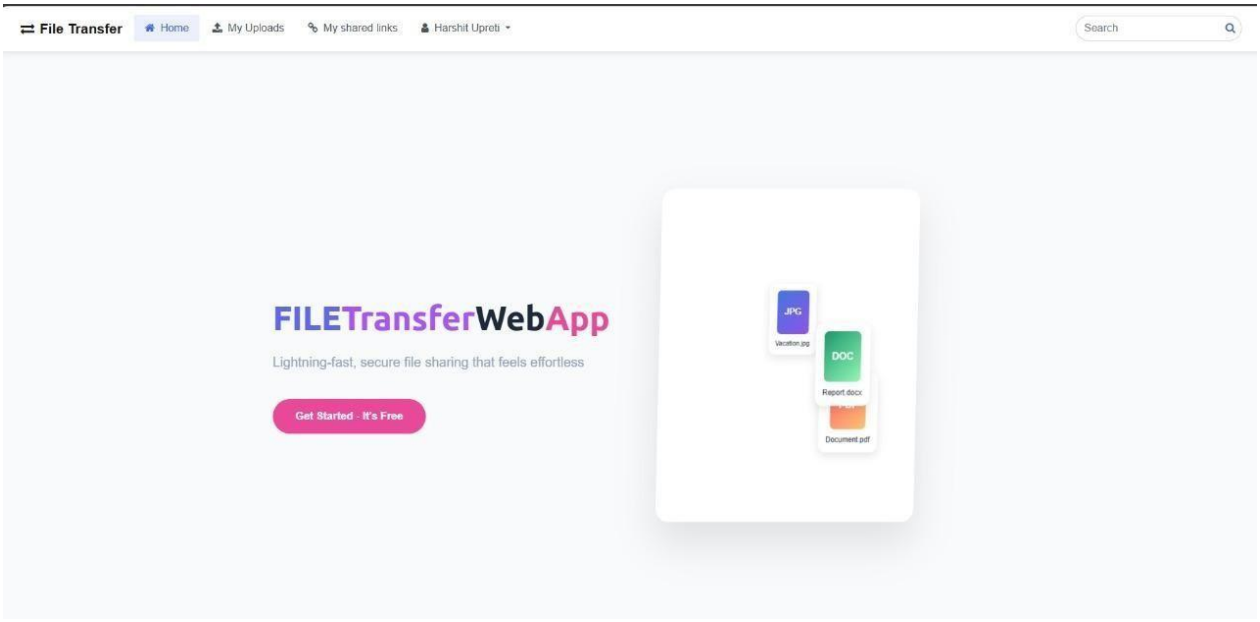

File Transfer
LOG IN
Email Address
Enter your email
Password
Enter your password
Remember me
Login
Don't have an account? Sign up

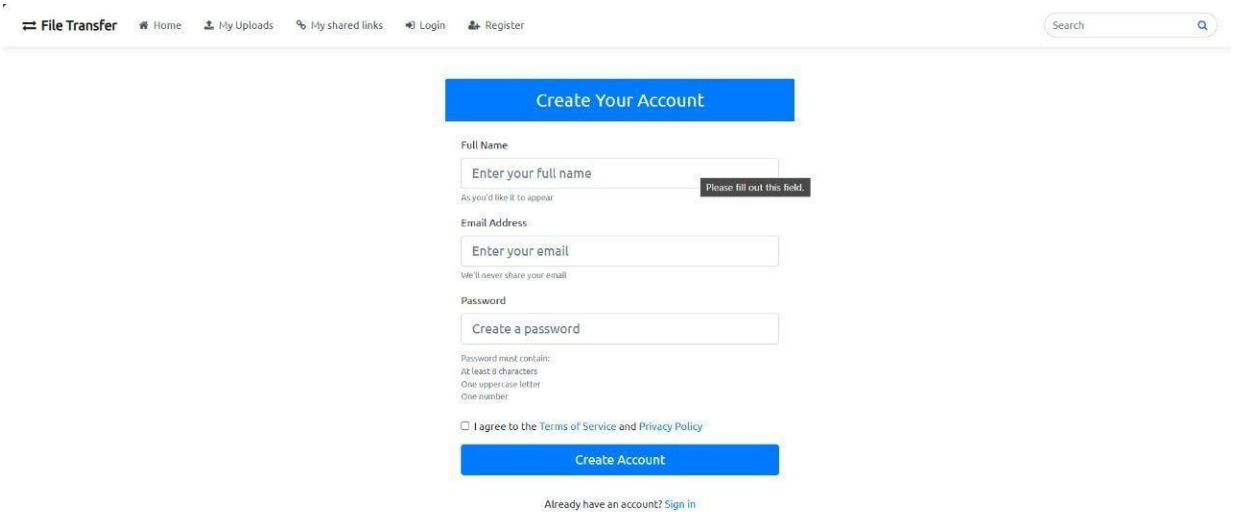

JS_Chapterwise_Notes...
JS_Chapterwise_Notes.pdf
Download File
Close
Share
Delete

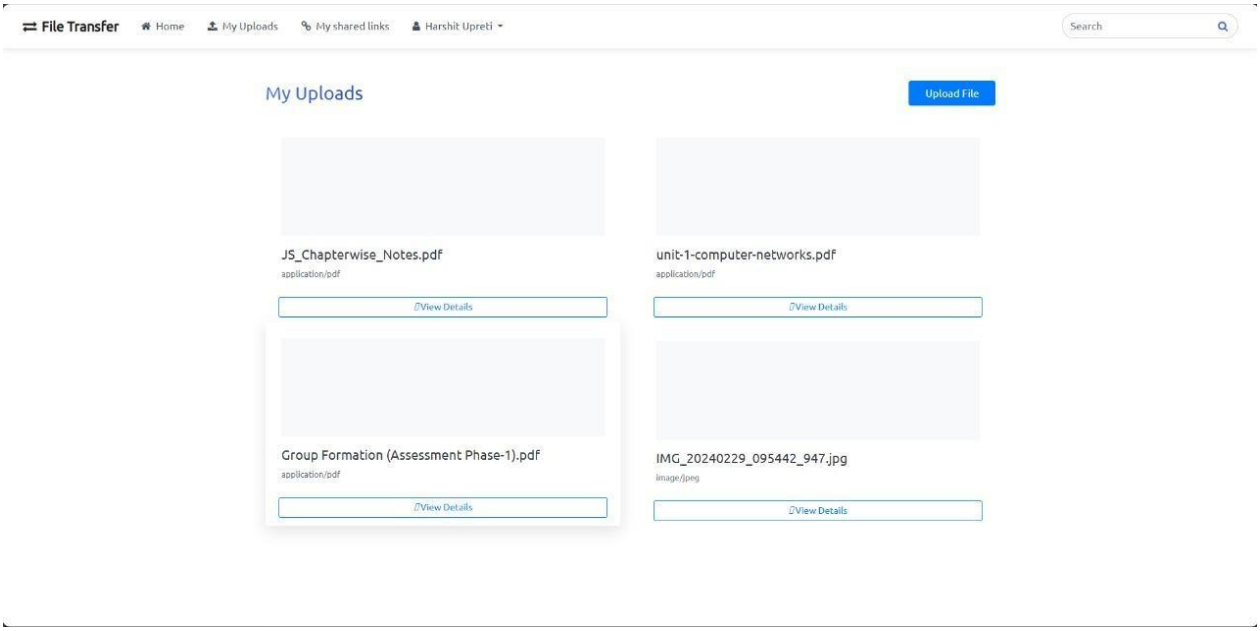

Search
Saved info
JS_Chapterwise_Notes.pdf
js
pos
result
current_num
zerolithx

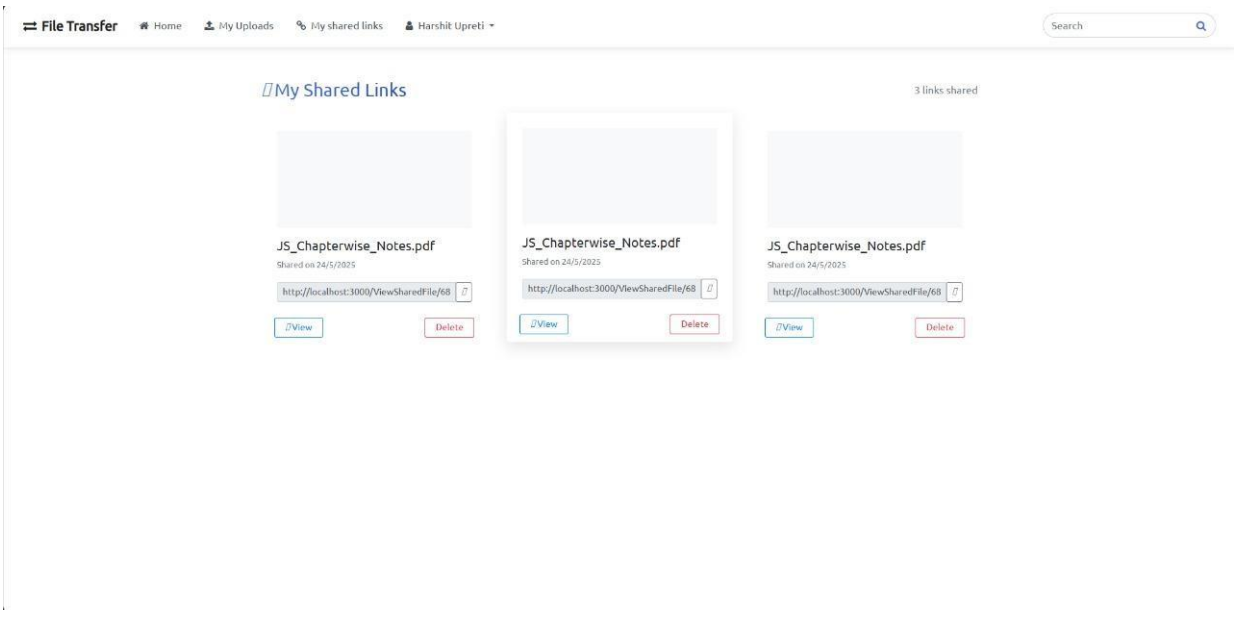

Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
file_transfer
public_links
users
local
public_links
localhost:27017 > file_transfer > public_links
Open MongoDB shell
Documents 3
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 3 of 3
_id: ObjectId('6831faefb14cf0155c56d287')
hash : "ZNGTQDBDVw"
file : Object
uploadedBy : Object
createdAt : 1748105967870
_id: ObjectId('6831fafab14cf0155c56d288')
hash : "A8e/Hl9Iko"
6831fafab14cf0155c56d288
file : Object
uploadedBy : Object
createdAt : 1748105978683
_id: ObjectId('683202c04407d50ad4b805d8')
hash : "G6I4ZfXjnH"
file : Object
uploadedBy : Object
createdAt : 1748107968589

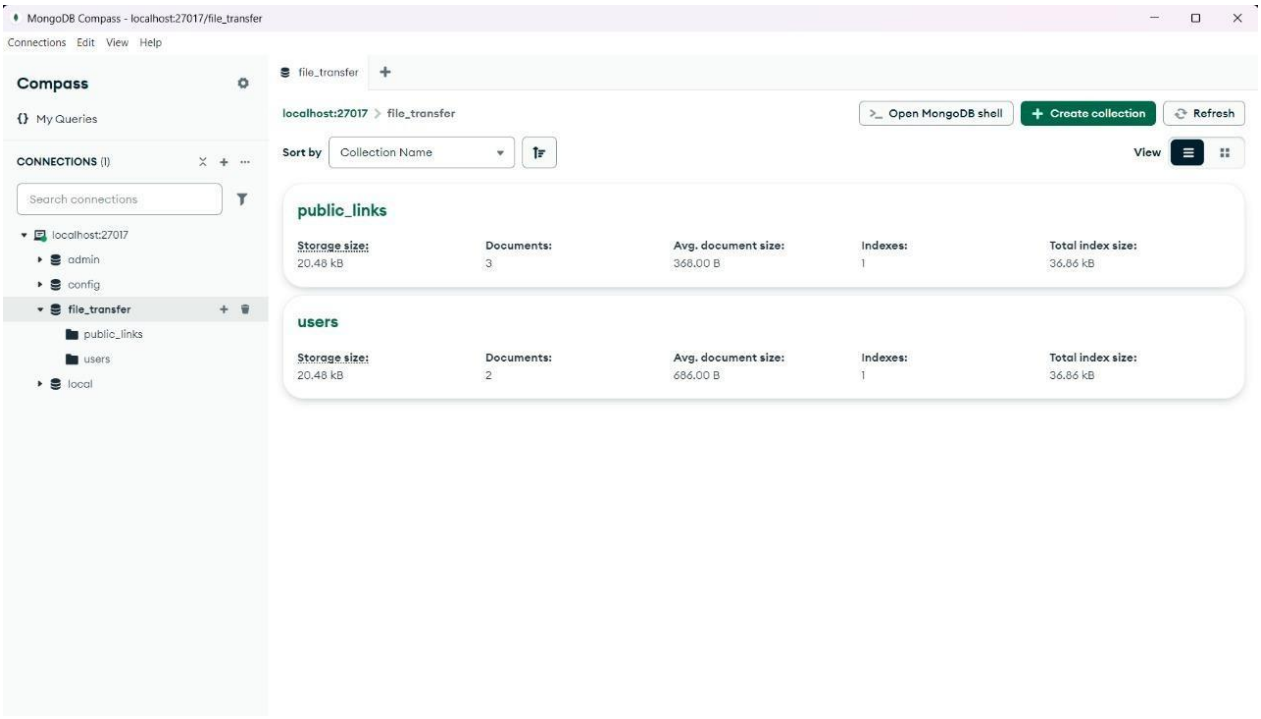

Generate Shareable Link?
This will create a public link that anyone can access.
Cancel
Generate Link

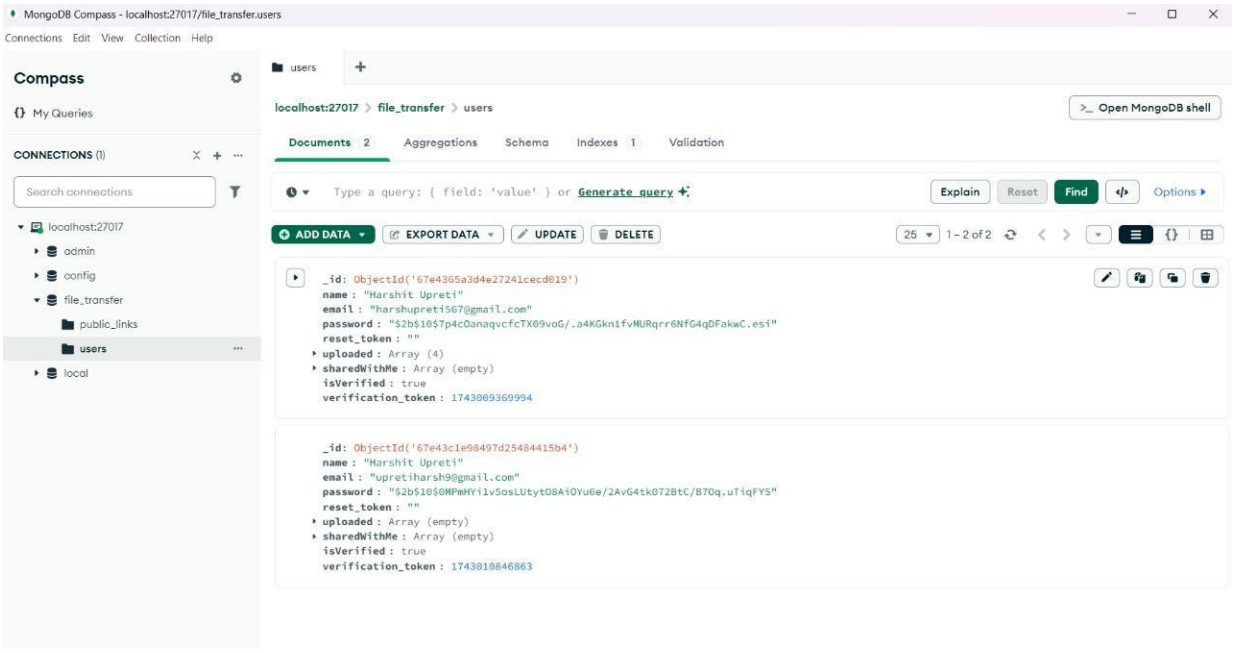

File Transfer
Home
My Uploads
My shared links
Harshit Uproti
Search
FILETransferWebApp
Lightning-fast, secure file sharing that feels effortless
Get Started - It's Free
JPG
Vacation.jpg
DOC
Report.docx
Document.pdf


File Transfer
Home
My Uploads
My shared links
Login
Register
Search
Create Your Account
Full Name
Enter your full name
Please fill out this field.
As you'd like it to appear
Email Address
Enter your email
We'll never share your email
Password
Create a password
Password must contain:
At least 8 characters
One uppercase letter
One number
I agree to the Terms of Service and Privacy Policy
Create Account
Already have an account? Sign in


File Transfer
Home
My Uploads
My shared links
Harshit Upreti
Search
My Uploads
Upload File
JS_Chapterwise_Notes.pdf
application/pdf
View Details
unit-1-computer-networks.pdf
application/pdf
View Details
Group Formation (Assessment Phase-1).pdf
application/pdf
View Details
IMG_20240229_095442_947.jpg
image/jpeg
View Details


File Transfer
Home
My Uploads
My shared links
Harshit Upreti
Search
My Shared Links
3 links shared
JS_Chapterwise_Notes.pdf
Shared on 24/5/2025
http://localhost:3000/ViewSharedFile/68
View
Delete
JS_Chapterwise_Notes.pdf
Shared on 24/5/2025
http://localhost:3000/ViewSharedFile/68
View
Delete
JS_Chapterwise_Notes.pdf
Shared on 24/5/2025
http://localhost:3000/ViewSharedFile/68
View
Delete


MongoDB Compass - localhost:27017/file_transfer
Connections Edit View Help
Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
file_transfer
public_links
users
local
file_transfer
localhost:27017 > file_transfer
Open MongoDB shell
Create collection
Refresh
Sort by
Collection Name
View
public_links
Storage size: 20.48 kB
Documents: 3
Avg. document size: 368.00 B
Indexes: 1
Total index size: 36.86 kB
users
Storage size: 20.48 kB
Documents: 2
Avg. document size: 686.00 B
Indexes: 1
Total index size: 36.86 kB


MongoDB Compass - localhost:27017/file_transfer.users
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
file_transfer
public_links
users
local
users
localhost:27017 > file_transfer > users
Open MongoDB shell
Documents 2
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 2 of 2
_id: ObjectId('67e4365a3d4e27241cecd019')
name : "Harshit Upreti"
email : "harshupreti567@gmail.com"
password : "$2b$10$7p4cOanaqvcfcTX09voG/.a4KGkn1fvMURqrr6NfG4qDFakwC.esi"
reset_token : ""
uploaded : Array (4)
sharedWithMe : Array (empty)
isVerified : true
verification_token : 1743009369994
_id: ObjectId('67e43c1e98497d25484415b4')
name : "Harshit Upreti"
email : "upretiharsh9@gmail.com"
password : "$2b$10$0MPmHYi1v5osLUtytO8AiOYu6e/2AvG4tk072BtC/B7Oq.uTiqFYS"
reset_token : ""
uploaded : Array (empty)
sharedWithMe : Array (empty)
isVerified : true
verification_token : 1743010846863

LIMITATIONS

1. **Limited Real-Time Collaboration**
   The system supports file sharing but lacks real-time collaborative editing or simultaneous access to shared documents like Google Docs.

2. **No In-Browser File Preview**
   Users cannot preview certain file types (e.g., videos, PDFs, Word documents) directly in the browser; they must download them first.

3. **File Size Dependent on Server Resources**
   The maximum file upload size is restricted by the server's RAM, disk space, and Node.js upload limits.

4. **Limited Image Compression Control**
   Image compression is implemented with basic configurations and may not allow users to adjust quality or format options before upload.

5. **Lack of Mobile App Support**
   The application is web-based and not optimized for mobile applications or native mobile device functionalities.

6. **Minimal Access Control Levels**
   Role-based access control is basic; advanced permission hierarchies (e.g., read-only, edit-only, link expiration) are not supported.

7. **No Version Control**
   If a file is updated or replaced, earlier versions are lost—there is no built-in version history or rollback option

   .

8. **Dependency on Internet Connectivity**
   Since the system is hosted on a remote server, any disruption in internet access will make the platform unavailable to users.

9. **No Offline Mode or Syncing**
   Unlike some cloud storage platforms, there is no feature to sync files offline or resume operations once connectivity is restored.

10. **Scalability Constraints on Shared Hosting**
    Performance may degrade with a high number of concurrent users or large file uploads, especially if deployed on a low-tier server.

ENHANCEMENTS

1. **User-Friendly Interface Improvements**
   Design a more intuitive and responsive UI with drag-and-drop file uploads, progress bars, and better folder navigation.

2. **Multi-File Uploads with Bulk Operations**
   Allow users to upload, move, rename, or delete multiple files and folders simultaneously for better efficiency.

3. **Customizable Notifications**
   Enable email or in-app notifications for file uploads, downloads, shares, and deletions to keep users informed.

4. **Integration with Cloud Storage Services**
   Provide options to connect and sync files with popular cloud storage platforms like Google Drive, Dropbox, or OneDrive.

5. **Advanced Search and Filtering**
   Enhance the search feature with filters by file type, date, size, and shared status to quickly locate files.

6. **Tagging and Metadata Management**
   Allow users to tag files and folders with keywords or categories for easier organization and retrieval.

7. **Scheduled File Sharing and Expiry Links**
   Introduce the ability to set automatic expiry dates on shareable links or schedule sharing for specific times.

8. **Collaborative Workspaces**
   Create shared workspaces or team folders with role-based permissions for group projects or departments.

9. **API Access**
   Develop RESTful APIs to allow third-party applications or automation tools to interact with the system.

10. **Multi-Factor Authentication (MFA)**
    Enhance account security by adding MFA options such as OTP or authenticator apps.

11. **File Tagging and Comments**
    Allow users to add comments or notes on files to facilitate feedback and collaboration.
12. **Custom Branding and Themes**
    Offer options for businesses to customize the UI with their branding colors, logos, and themes.

13. **Activity Logs and Audit Trails**
    Maintain detailed logs of user activities for security audits and accountability.

CONCLUSION

The File Transfer Web Application developed using Node.js and MongoDB offers a robust and scalable solution for seamless file management and sharing. By integrating core OS concepts such as file handling, access control, and directory management, the system efficiently supports uploading, organizing, and securely sharing files with both public links and private email-based sharing. The application incorporates essential features like folder creation, file renaming, move operations, recycle bin functionality, backup, and payment gateway integration, providing a comprehensive user experience.

Through careful system design and implementation, leveraging a modern technology stack including Express.js and MongoDB, the project achieves flexibility, security, and extensibility. While certain limitations exist, such as lack of real-time collaboration and mobile app support, proposed enhancements like in-browser preview, advanced access controls, file versioning, and API access pave the way for future growth.

Thorough testing and evaluation demonstrate the application's reliability, performance, and usability. Overall, this project establishes a strong foundation for a practical file transfer platform that can be expanded and adapted to meet evolving user needs and industry standards.

REFERENCES

1. Node.js Foundation. (n.d.). Node.js documentation. https://nodejs.org/en/docs/
2. Express.js. (n.d.). Express - Node.js web application framework. https://expressjs.com/
3. MongoDB, Inc. (n.d.). MongoDB documentation. https://www.mongodb.com/docs/
4. GitHub - expressjs/multer. (n.d.). Multer middleware for handling multipart/form-data. https://github.com/expressjs/multer
5. Stripe, Inc. (n.d.). Stripe API documentation. https://stripe.com/docs
6. Meta. (n.d.). React:A JavaScript library for building user interfaces. https://reactjs.org/docs/getting-started.html
7. GitHub, Inc. (n.d.). GitHub Docs. https://docs.github.com/