TAKSHSHILA INSTITUTE OF ENGINEERING & TECHNOLOGY,

JABALPUR (M.P.)

Session 2019

Department of Computer Science & Engineering

Major Project Report
On

"Sharify - File Sharing App"

Submitted For the partial fulfilment of the requirement

For the award of the degree of

Bachelor of Engineering

By

Amit Mehra (0207CS191008),

Danish beg (0207CS191022),

Aakash Kumar (0207CS191001)

And Abhay jhariya (0207CS191002),

Under the Guidance of

Prof. Swati Soni

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)
(UNIVERSITY OF TECHNOLOGY OF MADHYA PRADESH)

TAKSHSHILA INSTITUTE OF ENGINEERING & TECHNOLOGY, JABALPUR (M.P.)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled **"Sharify - File Sharing App"** is the bonafide record of the work done by **Amit Mehra, Danish beg, Aakash Kumar and Abhay jhariya** from Takshshila Institute of Engineering & Technology, Jabalpur, under the guidance of ***Prof. Swati Soni*** and submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal MP, in accordance with the requirement for the award of the Bachelor Engineering from Computer Science & Engineering.

I certify that the above declaration is true to the best of my knowledge and belief.

Prof.Deepak Agrawal
Department Coordinator

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)

(UNIVERSITY OF TECHNOLOGY OF MADHYA PRADESH)

APPROVAL CERTIFICATE

This is to certify that project entitled, **"Sharify - File Sharing App"** submitted by **Amit Mehra, Danish Beg, Aakash Kumar, and Abhay jhariya** is accepted for the award of degree of Bachelor Engineering in Computer Science & Engineering Department.

INTERNAL EXAMINER
Date:

EXTERNAL EXAMINER
Date:

ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.
I would like to express my sincere gratitude to my guide **Prof. *Swati Soni*** encouraging me throughout the research work and for providing me various opportunities at different stages of my studies.

I thank him especially for the help that he extended to me in the form of valuable discussions and reading materials**.**

I would like to thank to **Prof. Deepak Agrawal**, Department Coordinator. Takshshila Institute of engineering & Technology, Jabalpur, for his kind support and valuable suggestions wherever necessary.
I am always grateful to **Dr. B. K. Sahu,** Principal Takshshila Institute of Engineering &Technology, for permitting me to avail the facilities needed for this work.

I am grateful to all my colleagues at **Takshshila Institute of Engineering & Technology, Jabalpur** for encouraging me to complete the work.

Date……. **Students Name**

Amit Mehra, Danish Beg,

Aakash Kumar, Abhay jhariya

TABLE OF CONTENTS

ABOUT PROJECT

Sharify : File Sharing App

This is a file sharing app that allows users to share files with others quickly and easily. The app provides a secure platform to share files without any fear of data breaches or unauthorized access. With this app, users can share files of different types and sizes. Moreover, the app provides access to the shared files from anywhere in the world, making it an excellent choice for remote work or collaboration. Overall, this file sharing app is a convenient and secure tool for sharing files and collaborating with others.

HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements:

- **Operating System: Windows, Mac, or Linux**
- **Web Server: Apache or Nginx**
- **Programming Languages: PHP, JavaScript**
- **Database: MySQL**

Hardware Requirements:

- **Processor: 1 GHz or faster**
- **RAM: 1 GB minimum, 2 GB recommended**
- **Hard Disk Space: 50 MB for installation, additional space for file storage**
- **Network Interface: Ethernet or Wi-Fi for internet connectivity**

TECH STACK USED:

HTML

Hyper Text Markup Language is the computer language that facilitates website creation. The language, which has code words and syntax just like any other language, is relatively easy to comprehend and, as time goes on, increasingly powerful in what it allows someone to create. HTML continues to evolve to

meet the demands and requirements of the Internet under the guise of the World Wide Web Consortium, the organization that designs and maintains the language; for instance, with the transition to Web 2.0

CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web.

JAVASCRIPT

JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses Javascript to provide several forms of interactivity and simplicity.

EJS

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. No reinvention of iteration and control-flow. It is just plain JavaScript.

NodeJS

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

EXPRESSJS

Express.js is a small framework that works on top of Node.js web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middleware and

routing. It adds helpful utilities to Node.js HTTP objects and facilitates the rendering of dynamic HTTP objects.

MongoDB

MongoDB is a popular NoSQL document-based database management system that stores data in JSON-like documents with dynamic schemas, which makes it flexible and scalable. It is designed to handle large volumes of data and provides high availability and horizontal scaling with automatic sharding. MongoDB offers powerful indexing capabilities, allowing users to create indexes on any field or combination of fields, making queries faster and more efficient.

MongoDB also supports a range of programming languages and offers built-in support for data replication and backup. It also has a robust security system with features like authentication, authorization, encryption, and auditing to ensure data privacy and prevent unauthorized access.

One of the key benefits of MongoDB is its ability to handle unstructured and semi-structured data, making it a popular choice for big data and real-time analytics applications. Additionally, its flexible data model and horizontal scaling make it suitable for a wide range of use cases, including content management systems, social media platforms, e-commerce sites, and more.

DEVELOPMENT LIFE CYCLE

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in systems engineering and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. The concept generally refers to computer or information systems. In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process. Systems Development Life Cycle (SDLC) is a logical process used by a systems analyst to develop an information system, including requirements, validation, training, and user (stakeholder) ownership. Any SDLC should result in a high-quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

Computer systems are complex and often (especially with the recent rise of Service Oriented Architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, several SDLC models have been created: "waterfall"; "fountain"; "spiral"; "build and fix"; "rapid prototyping"; "incremental"; and "synchronize and stabilize". SDLC models can be described along a spectrum of agile to iterative to sequential. Agile methodologies, such as XP and Scrum, focus on light-weight processes which allow for rapid changes along the development cycle. Iterative methodologies, such as Rational Unified Process and Dynamic Systems Development Method, focus on limited project scopes and expanding or improving products by multiple iterations. Sequential or big-design-upfront (BDUF) models, such as Waterfall, focus on complete and correct planning to guide large projects and risks to successful and predictable results. Some agile and iterative proponents confuse the term SDLC with sequential or "more traditional" processes; however, SDLC is an umbrella term for all methodologies for the design, implementation, and release of software.

1. **Initiation/planning**
2. **Requirements gathering and analysis**
3. **Design**
4. **Build or coding**
5. **Testing**

· **Unit testing**: unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application

· **Integration testing**: The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware.

· **Black box testing**: Black-box testing uses external descriptions of the software, including specifications, requirements, and designs to derive test cases.

· **White box testing**: White box testing (a.k.a. clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure.

· **Regression testing**: It is any type of software testing that seeks to uncover software errors by partially retesting a modified program.

· Automation testing

· User acceptance testing & Performance testing

· **System testing**: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6. operations and maintenance:

Operations and maintenance (O&M) is a crucial phase in the development life cycle of a software system, as it ensures that the system remains functional and effective over time. O&M activities typically involve the ongoing monitoring, support, and maintenance of the system, as well as the identification and resolution of any issues that arise.

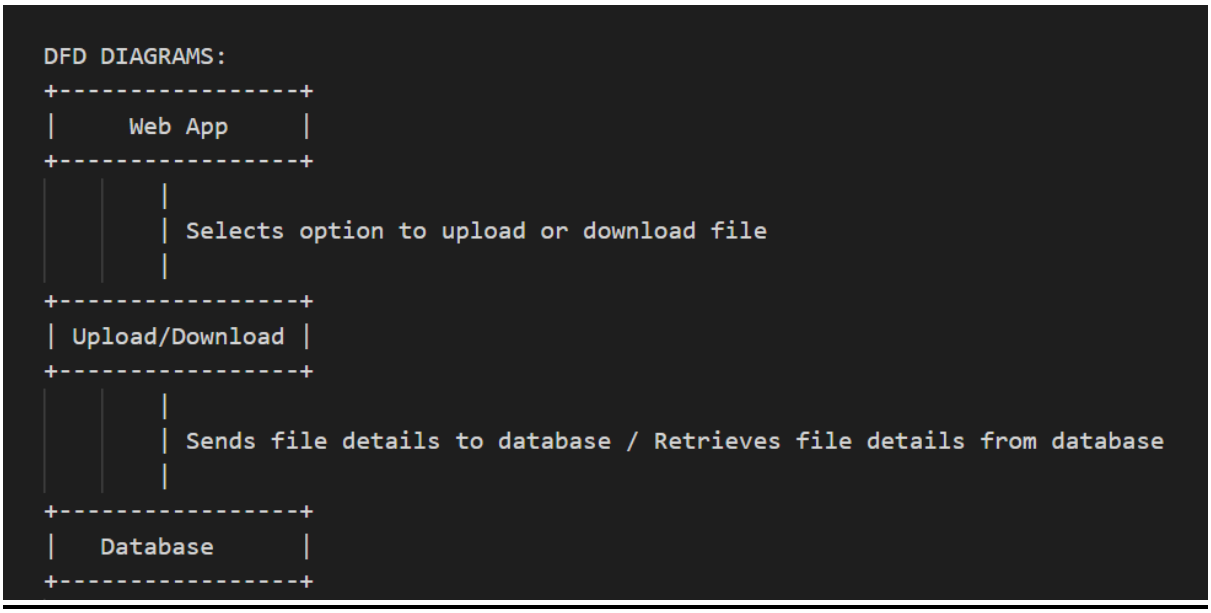
During the O&M phase, regular backups and updates are performed to ensure that the system remains up-to-date and secure. This includes maintaining the system's hardware and software components, as well as managing any necessary software upgrades or patches. In addition, system administrators and developers may need to provide technical support to end-users to troubleshoot issues and provide training as needed.

Another critical aspect of O&M is the monitoring and analysis of system performance metrics, such as response times and system availability. This helps to identify any potential issues before they become major problems, and can inform future updates or improvements to the system.

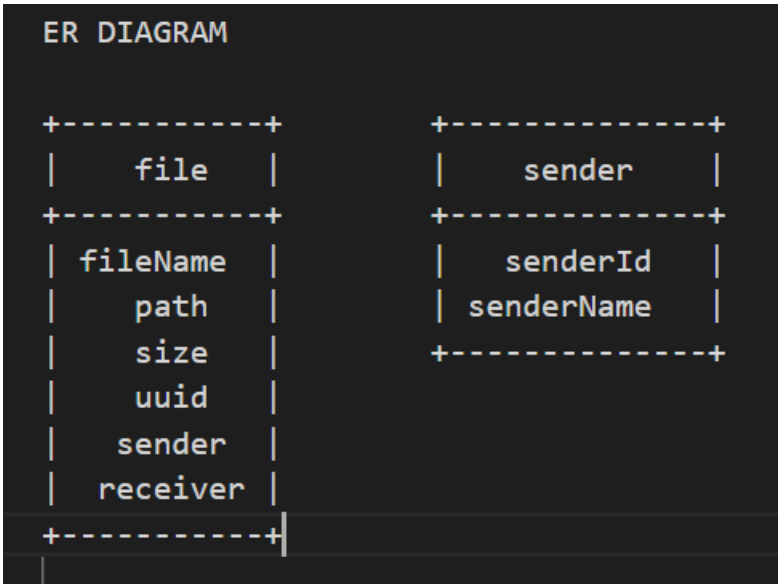
Overall, effective O&M helps to ensure that a software system remains reliable, efficient, and cost-effective over its entire lifecycle, and can contribute to improved user satisfaction and productivity.

DATABASE DESIGN:

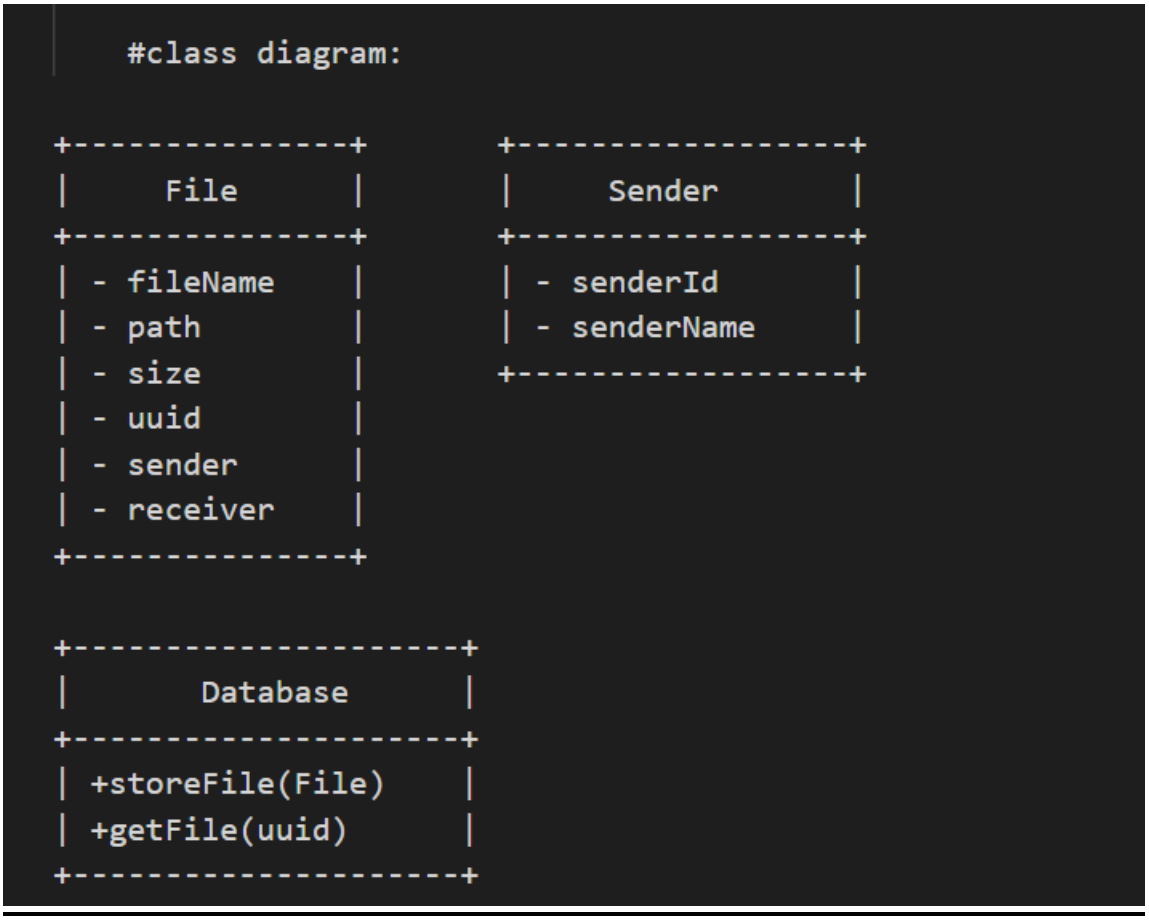
DFD DIAGRAMS:



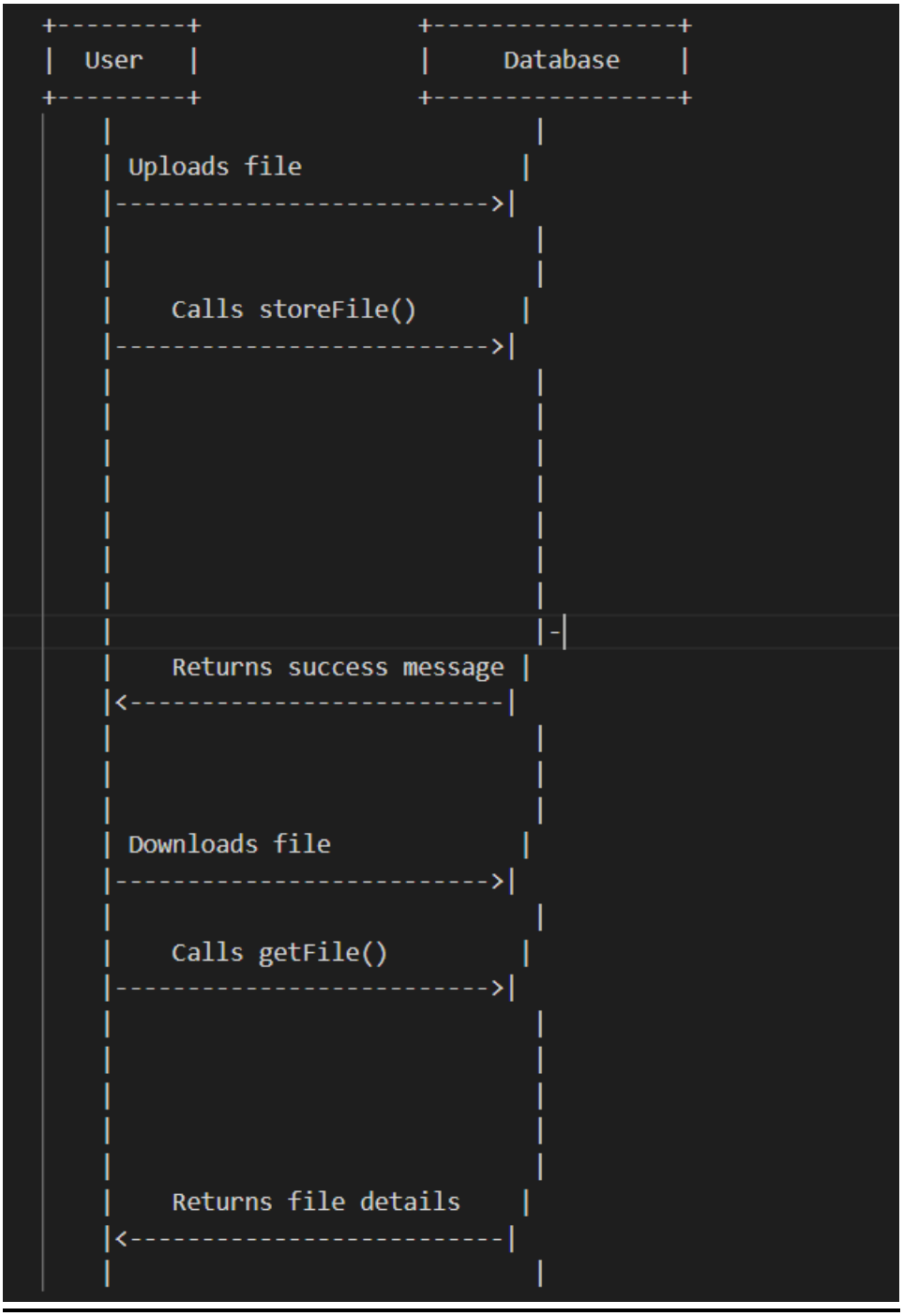
ER DIAGRAM:



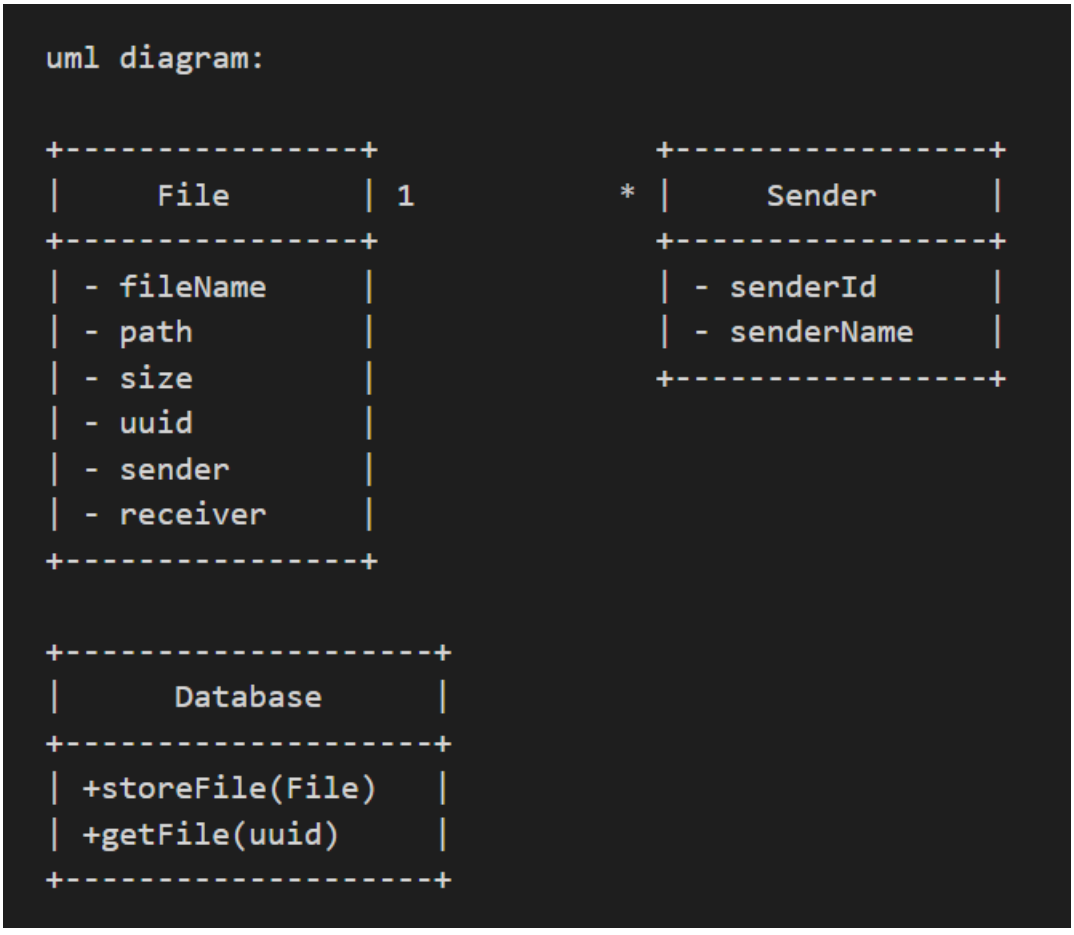
Class diagram:



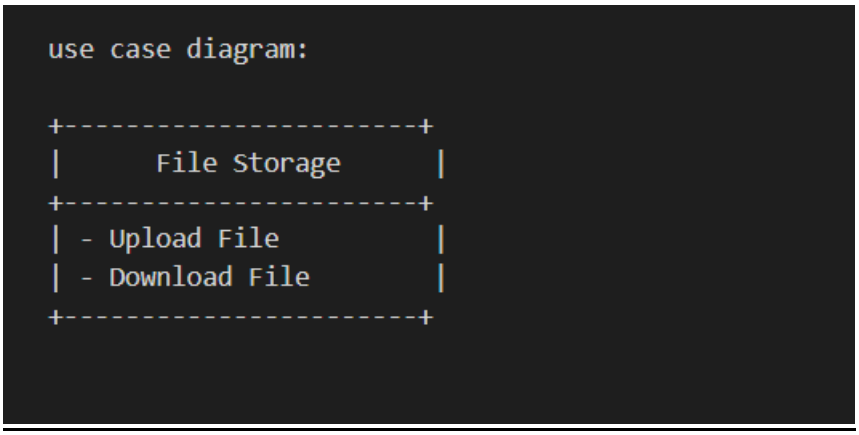
Sequence diagram:



UML diagram:

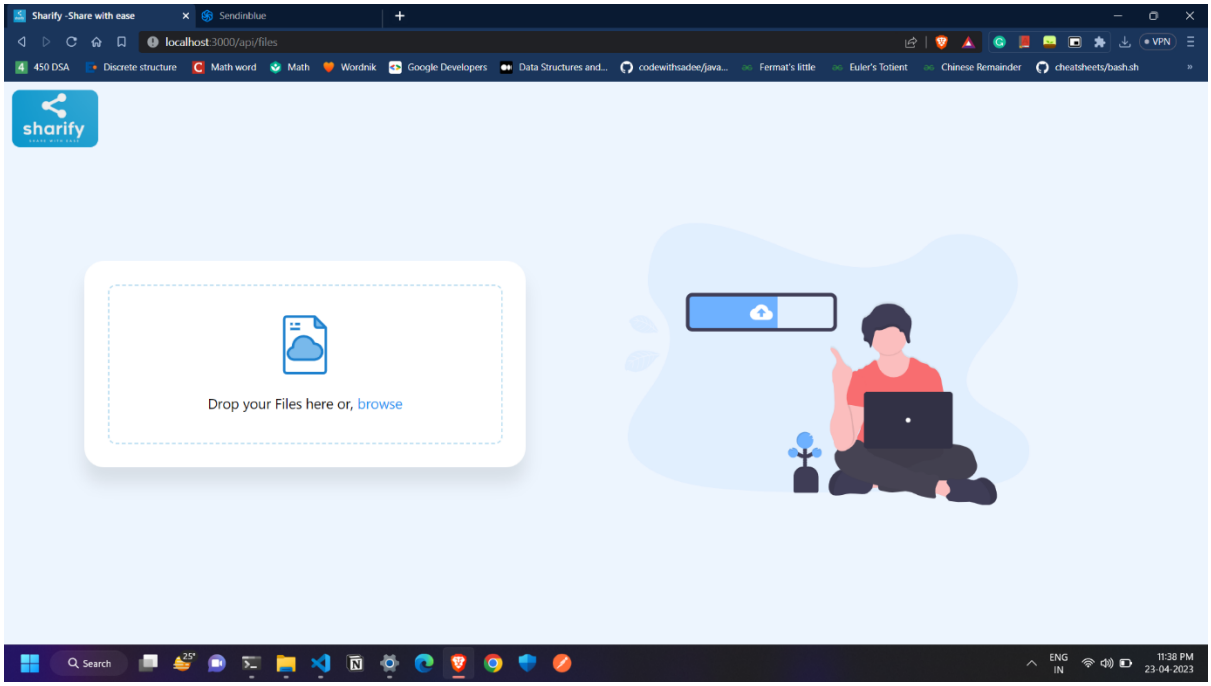


Use case diagram:

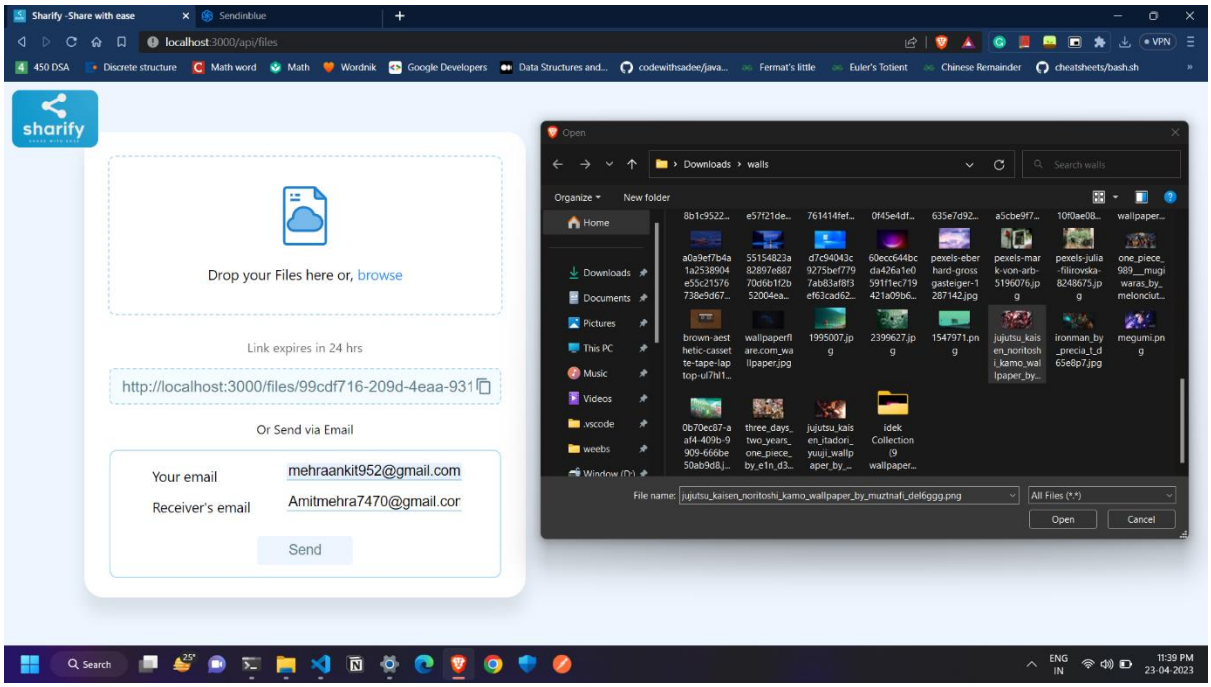


SCREEN LAYOUT

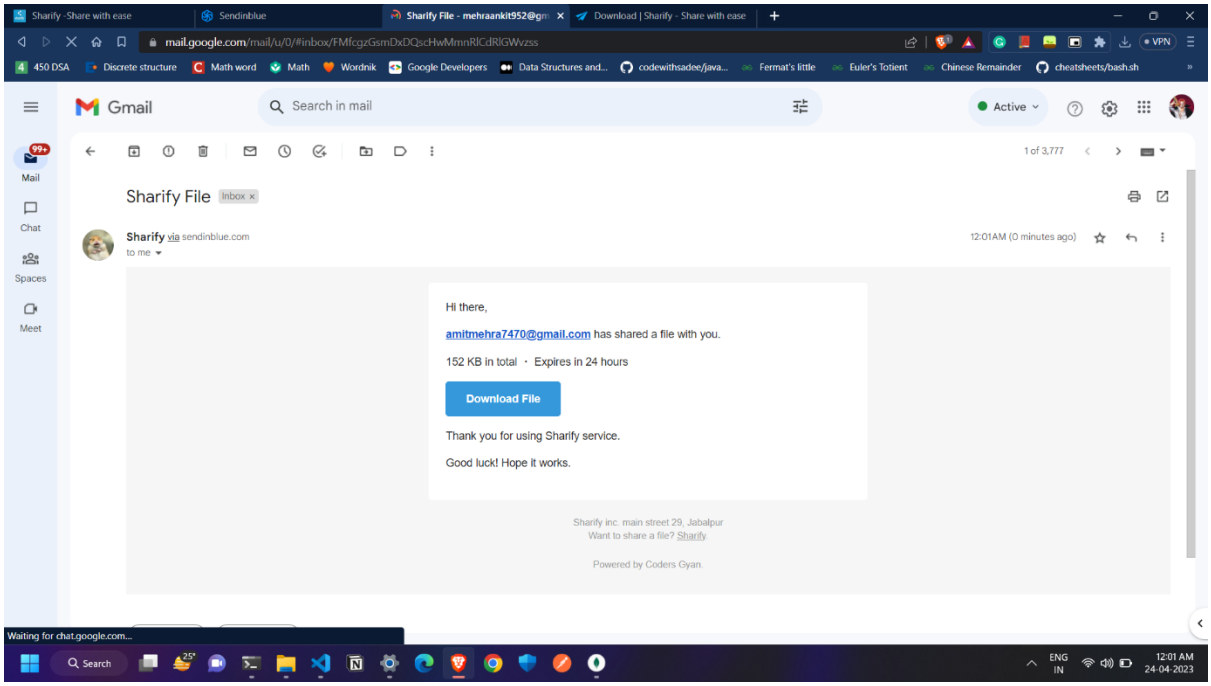
Home page:



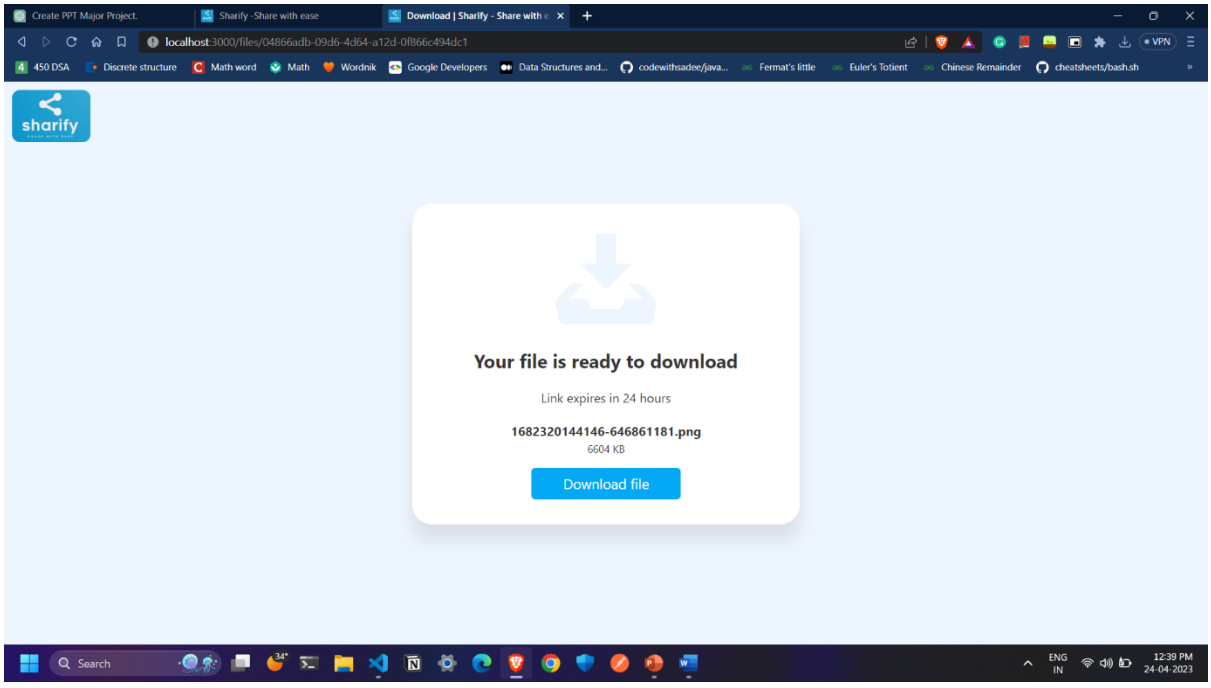
Uploading media and Generating link:



Received mail:



Download media



CODING

Server.js

```
require('dotenv').config();
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;
const path = require('path');
const scheduler = require("node-cron");
const { fetchAndDeleteData } = require("./services/fileCleaner");

const connectDB = require('./config/db');
connectDB();

//cleaning all the older files
scheduler.schedule("00 12 * * *", () => fetchAndDeleteData());

// middleware
app.use(express.json());
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.static(path.join(__dirname, 'public')));

// Routes
app.get('/', (req, res)=>{
    res.redirect("/api/files");
})

app.use('/api/files', require('./routes/files'));
app.use('/files', require('./routes/show'));
app.use('/files/download', require('./routes/download'));

// listening on port
app.listen(PORT, console.log(`Listening on port ${PORT}.`));
```

package.json

```
{
  "name": "sharify",
  "version": "1.0.0",
  "description": "An app that provides you the functionality to share files
with anyone with ease.",
  "main": "server.js",
  "scripts": {
    "dev": "nodemon server",
    "serve": "node server"
```

```
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "ejs": "^3.1.5",
    "express": "^4.17.1",
    "mongoose": "^5.10.0",
    "multer": "^1.4.4-lts.1",
    "node-cron": "^3.0.2",
    "nodemailer": "^6.4.11",
    "uuid": "^8.3.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.4"
  },
  "author": "Amit mehra",
  "license": "ISC"
}
```

Routes/files.js

```
const router = require('express').Router();
const multer = require('multer');
const path = require('path');
const File = require('../models/file');
const { v4: uuidv4 } = require('uuid');

let storage = multer.diskStorage({
    destination: (req, file, cb) => cb(null, 'uploads/') ,
    filename: (req, file, cb) => {
        const uniqueName = `${Date.now()}-${Math.round(Math.random() *
1E9)}${path.extname(file.originalname)}`;
              cb(null, uniqueName)
    } ,
});

let upload = multer({ storage, limits:{ fileSize: 1000000 * 100 },
}).single('myfile'); //100mb

router.get('/', (req, res) => {
  res.render("index");
})

/****/

router.post('/', (req, res) => {
```

```
    upload(req, res, async (err) =>
    {
      if (err)
      {
        return res.status(500).send({ error: err.message });
      }
      const file = new File({
          filename: req.file.filename,
          uuid: uuidv4(),
          path: req.file.path,
          size: req.file.size
      });
      const response = await file.save();
      console.log("file saved")
      res.json({ file: `${process.env.APP_BASE_URL}/files/${response.uuid}`
});
    });
});

router.post('/send', async (req, res) => {
  const { uuid, emailTo, emailFrom, expiresIn } = req.body;
  if(!uuid || !emailTo || !emailFrom) {
      return res.status(422).send({ error: 'All fields are required except
expiry.'});
  }
  // Get data from db
  try {
    const file = await File.findOne({ uuid: uuid });
    if(file.sender) {
      return res.status(422).send({ error: 'Email already sent once.'});
    }
    file.sender = emailFrom;
    file.receiver = emailTo;
    const response = await file.save();
    // send mail
    const sendMail = require('../services/mailService');
    sendMail({
      from: emailFrom,
      to: emailTo,
      subject: 'Sharify file',
      text: `${emailFrom} shared a file with you.`,
      html: require('../services/emailTemplate')({
                emailFrom,
                downloadLink:
`${process.env.APP_BASE_URL}/files/${file.uuid}?source=email` ,
                size: parseInt(file.size/1000) + ' KB',
                expires: '24 hours'
            })
```

```
    }).then(() => {
      return res.json({success: true});
    }).catch(err => {
      return res.status(500).json({error: 'Error in email sending.'});
    });
} catch(err) {
  return res.status(500).send({ error: 'Something went wrong.'});
}

});

module.exports = router;
```

routes/download.js

```
const router = require('express').Router();
const File = require('../models/file');

router.get('/:uuid', async (req, res) => {
   // Extract link and get file from storage send download stream
   const file = await File.findOne({ uuid: req.params.uuid });
   // Link expired
   if(!file) {
        return res.render('download', { error: 'Link has been expired.'});
   }
   const response = await file.save();
   const filePath = `${__dirname}/../${file.path}`;
   res.download(filePath);
});


module.exports = router;
```

config/db.js

```
require('dotenv').config();
const mongoose = require('mongoose');
function connectDB() {
    // Database connection 😬
    mongoose.connect(process.env.MONGO_CONNECTION_URL ,{ useNewUrlParser:
true, useCreateIndex:true, useUnifiedTopology: true, useFindAndModify : true
});
    const connection = mongoose.connection;
    connection.once('open', () => {
        console.log('Database connected 😬😬😬😬');
    }).catch(err => {
```

```
        console.log('Connection failed ☹☹☹☹');
    });
}

// mIAY0a6u1ByJsWWZ

module.exports = connectDB;
```

models/file.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const fileSchema = new Schema({
    filename: { type: String, required: true },
    path: { type: String, required: true },
    size: { type: Number, required: true },
    uuid: { type: String, required: true },
    sender: { type: String, required: false },
    receiver: { type: String, required: false },
}, { timestamps: true });

module.exports = mongoose.model('File', fileSchema);
```

services/mailService.js

```
const nodemailer = require("nodemailer");
module.exports = async ({ from, to, subject, text, html}) => {
        let transporter = nodemailer.createTransport({
            host: process.env.SMTP_HOST,
            port: process.env.SMTP_PORT,
            secure: false, // true for 465, false for other ports
            auth: {
                user: process.env.MAIL_USER, // generated ethereal user
                pass: process.env.MAIL_PASSWORD, // generated ethereal
password
            },
        });

        // send mail with defined transport object
    let info = await transporter.sendMail({
        from: `Sharify <${from}>`, // sender address
        to: to, // list of receivers
        subject: subject, // Subject line
        text: text, // plain text body
        html: html, // html body
```

```
    });
}
```

Services/emailTemplate.js

```
module.exports = ({emailFrom, downloadLink, size, expires}) => {
    return `
        <!doctype html>
            <html>
            <head>
                <meta name="viewport" content="width=device-width">
                <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
                <title>Simple Transactional Email</title>
                <style>
                /* -------------------------------------
                    INLINED WITH htmlemail.io/inline
                ------------------------------------- */
                /* -------------------------------------
                    RESPONSIVE AND MOBILE FRIENDLY STYLES
                ------------------------------------- */
                @media only screen and (max-width: 620px) {
                table[class=body] h1 {
                    font-size: 28px !important;
                    margin-bottom: 10px !important;
                }
                table[class=body] p,
                        table[class=body] ul,
                        table[class=body] ol,
                        table[class=body] td,
                        table[class=body] span,
                        table[class=body] a {
                    font-size: 16px !important;
                }
                table[class=body] .wrapper,
                        table[class=body] .article {
                    padding: 10px !important;
                }
                table[class=body] .content {
                    padding: 0 !important;
                }
                table[class=body] .container {
                    padding: 0 !important;
                    width: 100% !important;
                }
                table[class=body] .main {
                    border-left-width: 0 !important;
```

```
    border-radius: 0 !important;
    border-right-width: 0 !important;
}
table[class=body] .btn table {
    width: 100% !important;
}
table[class=body] .btn a {
    width: 100% !important;
}
table[class=body] .img-responsive {
    height: auto !important;
    max-width: 100% !important;
    width: auto !important;
}
}

/* -----------------------------------
    PRESERVE THESE STYLES IN THE HEAD
----------------------------------- */
@media all {
.ExternalClass {
    width: 100%;
}
.ExternalClass,
        .ExternalClass p,
        .ExternalClass span,
        .ExternalClass font,
        .ExternalClass td,
        .ExternalClass div {
    line-height: 100%;
}
.apple-link a {
    color: inherit !important;
    font-family: inherit !important;
    font-size: inherit !important;
    font-weight: inherit !important;
    line-height: inherit !important;
    text-decoration: none !important;
}
#MessageViewBody a {
    color: inherit;
    text-decoration: none;
    font-size: inherit;
    font-family: inherit;
    font-weight: inherit;
    line-height: inherit;
}
.btn-primary table td:hover {
```

```
                    background-color: #34495e !important;
                }
                .btn-primary a:hover {
                    background-color: #34495e !important;
                    border-color: #34495e !important;
                }
                }
                </style>
            </head>
            <body class="" style="background-color: #f6f6f6; font-family:
sans-serif; -webkit-font-smoothing: antialiased; font-size: 14px; line-height:
1.4; margin: 0; padding: 0; -ms-text-size-adjust: 100%; -webkit-text-size-
adjust: 100%;">
                <table border="0" cellpadding="0" cellspacing="0" class="body"
style="border-collapse: separate; mso-table-lspace: 0pt; mso-table-rspace:
0pt; width: 100%; background-color: #f6f6f6;">
                <tr>
                    <td style="font-family: sans-serif; font-size: 14px;
vertical-align: top;"> </td>
                    <td class="container" style="font-family: sans-serif;
font-size: 14px; vertical-align: top; display: block; Margin: 0 auto; max-
width: 580px; padding: 10px; width: 580px;">
                    <div class="content" style="box-sizing: border-box;
display: block; Margin: 0 auto; max-width: 580px; padding: 10px;">

                        <!-- START CENTERED WHITE CONTAINER -->
                        <span class="preheader" style="color: transparent;
display: none; height: 0; max-height: 0; max-width: 0; opacity: 0; overflow:
hidden; mso-hide: all; visibility: hidden; width: 0;">This is preheader text.
Some clients will show this text as a preview.</span>
                        <table class="main" style="border-collapse: separate;
mso-table-lspace: 0pt; mso-table-rspace: 0pt; width: 100%; background:
#ffffff; border-radius: 3px;">

                        <!-- START MAIN CONTENT AREA -->
                        <tr>
                            <td class="wrapper" style="font-family: sans-
serif; font-size: 14px; vertical-align: top; box-sizing: border-box; padding:
20px;">
                            <table border="0" cellpadding="0" cellspacing="0"
style="border-collapse: separate; mso-table-lspace: 0pt; mso-table-rspace:
0pt; width: 100%;">
                                <tr>
                                <td style="font-family: sans-serif; font-size:
14px; vertical-align: top;">
                                    <p style="font-family: sans-serif; font-
size: 14px; font-weight: normal; margin: 0; Margin-bottom: 15px;">Hi
there,</p>
```

```
                                    <p style="font-family: sans-serif; font-
size: 14px; font-weight: normal; margin: 0; Margin-bottom:
15px;"><b>${emailFrom}</b> has shared a file with you.</p>
                                    <p style="font-family: sans-serif; font-
size: 14px; font-weight: normal; margin: 0; Margin-bottom: 15px;">${size} in
total  ·  Expires in ${expires}</p>
                                    <table border="0" cellpadding="0"
cellspacing="0" class="btn btn-primary" style="border-collapse: separate; mso-
table-lspace: 0pt; mso-table-rspace: 0pt; width: 100%; box-sizing: border-
box;">
                                    <tbody>
                                        <tr>
                                        <td align="left" style="font-family:
sans-serif; font-size: 14px; vertical-align: top; padding-bottom: 15px;">
                                            <table border="0" cellpadding="0"
cellspacing="0" style="border-collapse: separate; mso-table-lspace: 0pt; mso-
table-rspace: 0pt; width: auto;">
                                            <tbody>
                                                <tr>
                                                <td style="font-family: sans-
serif; font-size: 14px; vertical-align: top; background-color: #3498db;
border-radius: 5px; text-align: center;"> <a href="${downloadLink}"
target="_blank" style="display: inline-block; color: #ffffff; background-
color: #3498db; border: solid 1px #3498db; border-radius: 5px; box-sizing:
border-box; cursor: pointer; text-decoration: none; font-size: 14px; font-
weight: bold; margin: 0; padding: 12px 25px; text-transform: capitalize;
border-color: #3498db;">Download file</a> </td>
                                                </tr>
                                            </tbody>
                                            </table>
                                        </td>
                                        </tr>
                                    </tbody>
                                    </table>
                                    <p style="font-family: sans-serif; font-
size: 14px; font-weight: normal; margin: 0; Margin-bottom: 15px;">Thank you
for using Sharify service.</p>
                                    <p style="font-family: sans-serif; font-
size: 14px; font-weight: normal; margin: 0; Margin-bottom: 15px;">Good luck!
Hope it works.</p>
                                </td>
                                </tr>
                            </table>
                            </td>
                        </tr>

                        <!-- END MAIN CONTENT AREA -->
                        </table>
```

```
                        <!-- START FOOTER -->
                        <div class="footer" style="clear: both; Margin-top:
10px; text-align: center; width: 100%;">
                        <table border="0" cellpadding="0" cellspacing="0"
style="border-collapse: separate; mso-table-lspace: 0pt; mso-table-rspace:
0pt; width: 100%;">
                            <tr>
                            <td class="content-block" style="font-family:
sans-serif; vertical-align: top; padding-bottom: 10px; padding-top: 10px;
font-size: 12px; color: #999999; text-align: center;">
                                <span class="apple-link" style="color:
#999999; font-size: 12px; text-align: center;">Sharify inc. main street 29,
Jabalpur</span>
                                <br> Want to share a file? <a
href="http://localhost:3000" style="text-decoration: underline; color:
#999999; font-size: 12px; text-align: center;">Sharify</a>.
                            </td>
                            </tr>
                            <tr>
                            <td class="content-block powered-by" style="font-
family: sans-serif; vertical-align: top; padding-bottom: 10px; padding-top:
10px; font-size: 12px; color: #999999; text-align: center;">
                                Powered by <a
href="https://www.youtube.com/channel/UCo9xTRmg1SqQ5JSsA2fAgJw" style="color:
#999999; font-size: 12px; text-align: center; text-decoration: none;">Coders
Gyan</a>.
                            </td>
                            </tr>
                        </table>
                        </div>
                        <!-- END FOOTER -->

                    <!-- END CENTERED WHITE CONTAINER -->
                    </div>
                    </td>
                    <td style="font-family: sans-serif; font-size: 14px;
vertical-align: top;"> </td>
                </tr>
                </table>
            </body>
        </html>
    `;
}
```

Services/fileCleaner.js

```
require("dotenv").config({ path: "../.env" });
const File = require("../models/file");
const fs = require("fs");

let fetchAndDeleteData = async () => {
  const files = await File.find({
    createdAt: { $lt: new Date(Date.now() - 24 * 60 * 60 * 1000) },
  }); // Get all documents older than 24 hours

  if (files.length) {
    for (const file of files) {
      try {
        fs.unlinkSync("../" + file.path);
        await file.remove();
        console.log(`Successfully deleted: ${file.filename}`);
      } catch (err) {
        console.log(`Error while deleting file: ${err} `);
      }
    }
  }

  console.log("All files older than 24 hours are deleted now");
};

module.exports = { fetchAndDeleteData };
```

views/index.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sharify -Share with ease</title>
    <link rel="stylesheet" href="/css/styles.css">
    <link rel="shortcut icon" href="/img/sharify_logo.png" type="image/x-icon"
style="border-radius: 10px;">

</head>

<body>
    <img src="/img/sharify_logo.png" alt="Inshare logo" class="logo">
```

```
    <section class="upload-container">
        <form action="">
            <div class="drop-zone">
                <div class="icon-container">
                    <img src="/img/file.svg" draggable="false" class="center"
alt="File Icon">
                    <img src="/img/file.svg" draggable="false" class="left"
alt="File Icon">
                    <img src="/img/file.svg" draggable="false" class="right"
alt="File Icon">
                </div>
                <input type="file" id="fileInput">
                <div class="title">Drop your Files here or, <span
id="browseBtn">browse</span></div>
            </div>
        </form>
        <div class="progress-container">
            <div class="bg-progress"></div>

            <div class="inner-container">
                <div class="status">Uploading...</div>
                <div class="percent-container">
                    <span class="percentage" id="progressPercent">0</span>%
                </div>
                <div class="progress-bar"></div>
            </div>

        </div>
        <div class="sharing-container">
            <p class="expire">Link expires in 24 hrs</p>


            <div class="input-container">
                <input type="text" id="fileURL" readonly>
                <img src="/img/copy-icon.svg" id="copyURLBtn" alt="copy to
clipboard icon">
            </div>


            <p class="email-info">Or Send via Email</p>
            <div class="email-container">
                <form id="emailForm">
                    <div class="filed">
                        <label for="fromEmail">Your email</label>
                        <input type="email" autocomplete="email" required
name="from-email" id="fromEmail">
                    </div>

                    <div class="filed">
```

```
                        <label for="toEmail">Receiver's email</label>
                        <input type="email" required autocomplete="receiver"
name="to-email" id="toEmail">
                    </div>
                    <div class="send-btn-container">
                        <button type="submit">Send</button>
                    </div>
                </form>
            </div>
        </div>
    </section>

    <div class="image-vector"></div>
    <div class="toast">Sample message</div>

    <script src="/index.js"></script>
</body>

</html>
```

Views/download.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Download | Sharify - Share with ease</title>
    <link rel="stylesheet" href="/css/style.css">
    <link rel="shortcut icon" href="/img/sharify_logo.png" type="image/x-icon"
style="border-radius: 10px;">
</head>

<body>
    <img src="/img/sharify_logo.png" alt="Sharify logo" class="logo">

    <section class="download">
        <img class="download__icon" src="/img/download-sd.svg" alt="inshare-
download">
        <% if(locals.error){ %>
        <h4><%= locals.error %> 😅</h4>
        <% } else { %>
        <h2>Your file is ready to download</h2>
        <p>Link expires in 24 hours</p>
        <div class="download__meta">
```

```
            <h4><%= fileName %></h4>
            <small><%= parseInt(fileSize/1000) %> KB</small>
        </div>
        <div class="send-btn-container">
            <a href="<%= downloadLink %>">Download file</a>
        </div>
        <% } %>
    </section>
</body>

</html>
```

Public/index.js

```
const dropZone = document.querySelector(".drop-zone");
const fileInput = document.querySelector("#fileInput");
const browseBtn = document.querySelector("#browseBtn");

const bgProgress = document.querySelector(".bg-progress");
const progressPercent = document.querySelector("#progressPercent");
const progressContainer = document.querySelector(".progress-container");
const progressBar = document.querySelector(".progress-bar");
const status = document.querySelector(".status");

const sharingContainer = document.querySelector(".sharing-container");
const copyURLBtn = document.querySelector("#copyURLBtn");
const fileURL = document.querySelector("#fileURL");
const emailForm = document.querySelector("#emailForm");

const toast = document.querySelector(".toast");

const baseURL = "http://localhost:3000";
const uploadURL = `${baseURL}/api/files`;
const emailURL = `${baseURL}/api/files/send`;

const maxAllowedSize = 100 * 1024 * 1024; //100mb


browseBtn.addEventListener("click", () => {
  fileInput.click();
});

dropZone.addEventListener("drop", (e) => {
  e.preventDefault();
  //   console.log("dropped", e.dataTransfer.files[0].name);
  const files = e.dataTransfer.files;
  if (files.length === 1) {
```

```
    if (files[0].size < maxAllowedSize) {
      fileInput.files = files;
      uploadFile();
    } else {
      showToast("Max file size is 100MB");
    }
  } else if (files.length > 1) {
    showToast("You can't upload multiple files");
  }
  dropZone.classList.remove("dragged");
});

dropZone.addEventListener("dragover", (e) => {
  e.preventDefault();
  dropZone.classList.add("dragged");

  // console.log("dropping file");
});

dropZone.addEventListener("dragleave", (e) => {
  dropZone.classList.remove("dragged");

  console.log("drag ended");
});

// file input change and uploader
fileInput.addEventListener("change", () => {
  if (fileInput.files[0].size > maxAllowedSize) {
    showToast("Max file size is 100MB");
    fileInput.value = ""; // reset the input
    return;
  }
  uploadFile();
});

// sharing container listenrs
copyURLBtn.addEventListener("click", () => {
  fileURL.select();
  document.execCommand("copy");
  showToast("Copied to clipboard");
});

fileURL.addEventListener("click", () => {
  fileURL.select();
});

const uploadFile = () => {
  console.log("file added uploading");
```

```
  files = fileInput.files;
  const formData = new FormData();
  formData.append("myfile", files[0]);

  //show the uploader
  progressContainer.style.display = "block";

  // upload file
  const xhr = new XMLHttpRequest();

  // listen for upload progress
  xhr.upload.onprogress = function (event) {
    // find the percentage of uploaded
    let percent = Math.round((100 * event.loaded) / event.total);
    progressPercent.innerText = percent;
    const scaleX = `scaleX(${percent / 100})`;
    bgProgress.style.transform = scaleX;
    progressBar.style.transform = scaleX;
  };

  // handle error
  xhr.upload.onerror = function () {
    showToast(`Error in upload: ${xhr.status}.`);
    fileInput.value = ""; // reset the input
  };

  // listen for response which will give the link
  xhr.onreadystatechange = function () {
    if (xhr.readyState == XMLHttpRequest.DONE) {
      console.log(xhr.responseText);
      onFileUploadSuccess(xhr.responseText);
    }
  };
  console.log("hello");
  xhr.open("POST", uploadURL);
  xhr.send(formData);
};

const onFileUploadSuccess = (res) => {
  fileInput.value = ""; // reset the input
  status.innerHTML = "Uploaded";

  // remove the disabled attribute from form btn & make text send
  emailForm[2].removeAttribute("disabled");
  emailForm[2].innerText = "Send";
  progressContainer.style.display = "none"; // hide the box
```

```
  const { file: url } = JSON.parse(res);
  console.log(url);
  sharingContainer.style.display = "block";
  fileURL.value = url;
};

emailForm.addEventListener("submit", (e) => {
  e.preventDefault(); // stop submission

  // disable the button
  emailForm[2].setAttribute("disabled", "true");
  emailForm[2].innerText = "Sending";

  const url = fileURL.value;

  const formData = {
    uuid: url.split("/").splice(-1, 1)[0],
    emailTo: emailForm.elements["to-email"].value,
    emailFrom: emailForm.elements["from-email"].value,
  };
  console.log(formData);
  fetch(emailURL, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(formData),
  })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        showToast("Email Sent");
        sharingContainer.style.display = "none"; // hide the box
      }
    });
});

let toastTimer;
// the toast function
const showToast = (msg) => {
  clearTimeout(toastTimer);
  toast.innerText = msg;
  toast.classList.add("show");
  toastTimer = setTimeout(() => {
    toast.classList.remove("show");
  }, 2000);
};
```

Public/css/style.css

```
:root {
    --main-bg-color: #edf5fe;
    --light-blue: #03a9f4;
    --dark-blue: #028bca;
  }

  body {
    font-family: system-ui;
    background: var(--main-bg-color);
    height: 98vh;
    overflow: hidden;
    display: flex;
    align-items: center;
    justify-content: center;
    color: #333;
  }

  .logo {
    position: absolute;
    top: 10px;
    left: 10px;
    width: 100px;
    border-radius: 10px;
  }
section.download {
    background: #fff;
    width: 430px;
    max-width: 90%;
    border-radius: 25px;
    box-shadow: 0px 20px 20px 0px #00000017;
    padding: 2rem;
    text-align: center;
}
.download__icon {
    height: 8rem;
}
.download__meta h4 {
    margin-bottom: 0;
    line-height: 1.3;
}
.send-btn-container a {
    display: inline-block;
    font-size: 18px;
    padding: 8px 40px;
    margin-top: 15px;
    background: var(--light-blue);
```

```
    text-decoration: none;
    border: none;
    border-radius: 5px;
    color: #fff;
    cursor: pointer;
    transition: all .3s ease-in-out;
  }

  .send-btn-container a:hover {
      background: var(--dark-blue);
  }
```

Public/css/styles.css

```
:root {
  --main-bg-color: #edf5fe;
  --light-blue: #03a9f4;
  --border-color: #0288d147;
  --container-width: 500px;
}

body {
  font-family: system-ui;
  background: var(--main-bg-color);
  height: 98vh;
  overflow: hidden;
}

.logo {
  position: absolute;
  top: 10px;
  left: 10px;
  width: 100px;
  border-radius: 10px;
}

.image-vector {
  width: 50vw;
  height: 50vh;
  background: url(/img/undraw-upload.svg) no-repeat center;
  background-size: contain;
}

body,
.upload-container,
.drop-zone {
  display: flex;
```

```
  align-items: center;
  justify-content: center;
}
.upload-container,
.drop-zone {
  flex-direction: column;
}

.upload-container {
  background: #ffffff;
  border-radius: 25px;
  box-shadow: 0px 20px 20px 0px #00000017;
}

.drop-zone {
  width: var(--container-width);
  min-height: 200px;
  border: 2px dashed var(--border-color);
  border-radius: 10px;
  margin: 30px;
  transition: 0.2s all ease-in;
}

/* will be added when user drags */
.drop-zone.dragged {
  background: var(--main-bg-color);
  border-color: #0288d1;
}

.drop-zone input {
  display: none;
}

.icon-container {
  position: relative;
  width: 75px;
  height: 100px;
}

.icon-container img {
  width: 75px;
  position: absolute;
  transition: transform 0.25s ease-in-out;
  transform-origin: bottom;
}

.icon-container .center {
  z-index: 10;
```

```
}
.icon-container .right,
.icon-container .left {
  filter: grayscale(0.5);
  transform: scale(0.9);
}

.dragged .center {
  transform: translateY(-5px);
}
.dragged .right {
  transform: rotate(10deg) scale(0.9) translateX(20px);
}
.dragged .left {
  transform: rotate(-10deg) scale(0.9) translateX(-20px);
}

.title {
  font-size: large;
}

#browseBtn {
  color: #2196f3;
  cursor: pointer;
}

/* uploading progress styles */
.progress-container {
  border: 2px solid var(--main-bg-color);
  width: var(--container-width);
  height: 70px;
  border-radius: 10px;
  margin-bottom: 25px;
  position: relative;
  display: none;
}

.progress-container .inner-container {
  margin: 10px 15px;
  z-index: 2;
  position: absolute;
  width: calc(100% - 30px);
}

.progress-container .percent-container {
  font-size: 14px;
  margin: 5px;
  opacity: 0.7;
```

```css
}

.progress-container .bg-progress {
  position: absolute;
  background: var(--main-bg-color);
  width: 100%;
  height: 100%;
  z-index: 1;
  transition: transform 250ms linear;
  transform: scaleX(0);
  transform-origin: left;
}

.progress-container .progress-bar {
  width: 100%;
  height: 3px;
  border-radius: 2px;
  background: #03a9f4;
  transition: transform 200ms linear;
  transform: scaleX(0);
  transform-origin: left;
}

/* sharing container style */
.sharing-container {
  margin-bottom: 25px;
  width: var(--container-width);
  border-radius: 10px;
  display: none;
}

.sharing-container p {
  text-align: center;
}

.sharing-container .expire {
  font-size: 16px;
  opacity: 0.7;
  margin-top: 0;
}

.sharing-container .input-container {
  display: flex;
  position: relative;
  width: 100%;
  margin-bottom: 20px;
}
```

```css
.sharing-container .input-container input {
  width: 100%;
  border-radius: 3px;
  padding: 10px 15px;
  font-size: 20px;
  border: 2px dashed var(--border-color);
  border-radius: 6px;
  background: #f5fcff;
  color: #607d8b;
}

.sharing-container img {
  height: 22px;
  width: 30px;
  position: absolute;
  right: 7px;
  top: 12px;
  cursor: pointer;
  background: #f5fcff;
}

.email-container form {
  border: 2px solid var(--border-color);
  width: 100%;
  padding: 15px;
  box-sizing: border-box;
  border-radius: 10px;
  display: flex;
  flex-direction: column;
  align-items: center;
}
.email-container,
.send-btn-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
.email-container label {
  margin: 5px;
  font-size: 18px;
}

.email-container input {
  border: none;
  border-bottom: 2px solid var(--border-color);
  height: 19px;
  font-size: 18px;
  text-align: center;
```

```css
}

.email-container input:focus {
  outline: none;
}

.email-container .filed {
  margin-bottom: 5px;
  display: flex;
  justify-content: space-between;
  width: 400px;
}

.send-btn-container button {
  font-size: 18px;
  padding: 8px 40px;
  margin-top: 15px;
  background: var(--main-bg-color);
  border: none;
  border-radius: 5px;
  color: #607d8b;
  cursor: pointer;
}

.toast {
  position: absolute;
  bottom: 10px;
  right: 50%;
  transform: translate(50%, 60px);
  padding: 10px 20px;
  background: var(--light-blue);
  color: #fff;
  border-radius: 5px;
  font-size: 18px;
  box-shadow: 0px 10px 15px -3px rgba(0, 0, 0, 0.1),
    0px 4px 6px -2px rgba(0, 0, 0, 0.05);
  transition: transform ease-in-out 0.2s;
}

.show.toast {
  transform: translate(50%, 0);
}

@media screen and (max-width: 900px) {
  :root {
    --container-width: 320px;
  }
  .image-vector {
```

```
    display: none;
  }
  .email-container .filed {
    flex-direction: column;
  }
  .email-container .filed {
    width: 300px;
  }
}
```

TESTING

In computer programming, **unit testing** is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits. Unit tests find problems early in the development cycle.

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all the "integrated" software components that have successfully passed integration testing and the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter assemblages" and within the system as a whole.

The following examples are different types of testing that should be considered during System testing:

· GUI software testing
· Usability testing
· Performance testing
· Compatibility testing
· Error handling testing
· Load testing
· Volume testing
· Stress testing
· Security testing
· Scalability testing
· Smoke testing
· Exploratory testing
· Ad hoc testing
· Regression testing
· Reliability testing
· Installation testing
· Maintenance testing
· Recovery testing / failover
· Accessibility testing,

Bibliography

Website references:

https://javascript.org

https://fontawesome.com

https://ejs.com

https://npmjs.com

https://expressjs.com

https://nodejs.com

https://google.com

https://mongodb.com