```
node_modules
package-lock.json<app-nav></app-nav>
<router-outlet></router-outlet>import { Component, inject } from "@angular/core";
import { RouterOutlet } from "@angular/router";
import { HttpClient } from "@angular/common/http";
import { map, Observable, tap } from "rxjs";
import { AsyncPipe, NgForOf, NgIf } from "@angular/common";
import { DriveComponent } from "./components/drive/drive.component";
import { NavComponent } from "./components/nav/nav.component";
export interface Root {
  diskDetails: DiskDetails;
  allFilesAndDirs: AllFilesAndDir[];
}
export interface DiskDetails {
  diskPath: string;
  free: number;
  size: number;
}
export interface AllFilesAndDir {
  dirent: Dirent;
  name: string;
  path: string;
  stats: Stats;
  isFolder: boolean;
  isImage: boolean;
  isVideo: boolean;
  images?: Image[];
  buffer?: Buffer2;
}
export interface Dirent {
  name: string;
  path: string;
}
export interface Stats {
  dev: number;
  mode: number;
  nlink: number;
  uid: number;
  gid: number;
  rdev: number;
  blksize: number;
  ino: number;
  size: number;
  blocks: number;
  atimeMs: number;
  mtimeMs: number;
```

```
  ctimeMs: number;
  birthtimeMs: number;
  atime: string;
  mtime: string;
  ctime: string;
  birthtime: string;
}
export interface Image {
  dirent: Dirent2;
  name: string;
  path: string;
  stats: Stats2;
  isFolder: boolean;
  buffer: Buffer;
}
export interface Dirent2 {
  name: string;
  path: string;
}
export interface Stats2 {
  dev: number;
  mode: number;
  nlink: number;
  uid: number;
  gid: number;
  rdev: number;
  blksize: number;
  ino: number;
  size: number;
  blocks: number;
  atimeMs: number;
  mtimeMs: number;
  ctimeMs: number;
  birthtimeMs: number;
  atime: string;
  mtime: string;
  ctime: string;
  birthtime: string;
}
export interface Buffer {
  type: string;
  data: number[];
}
export interface Buffer2 {
  type: string;
  data: number[];
```

```
}
@Component({
  selector: "app-root",
  imports: [
    RouterOutlet,
    NgForOf,
    AsyncPipe,
    NgIf,
    DriveComponent,
    NavComponent,
  ],
  templateUrl: "./app.component.html",
  standalone: true,
  styleUrl: "./app.component.scss",
})
export class AppComponent {}import { ApplicationConfig,
provideZoneChangeDetection } from "@angular/core";
import { provideRouter } from "@angular/router";
import { routes } from "./app.routes";
import {
  provideHttpClient,
  withInterceptorsFromDi,
} from "@angular/common/http";
export const appConfig: ApplicationConfig = {
  providers: [
    provideHttpClient(withInterceptorsFromDi()),
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
  ],
};import { Routes } from "@angular/router";
import { HomeComponent } from "./components/home/home.component";
import { DriveComponent } from "./components/drive/drive.component";
export const routes: Routes = [
  { path: "", component: HomeComponent },
  { path: "drive", component: DriveComponent },
  { path: "**", redirectTo: "" },
];<!-- drive.component.html -->
<div class="h-full flex flex-col bg-gray-50">
  <!-- Header with search and controls -->
  <div class="bg-white border-b p-4 flex items-center justify-between">
    <div class="flex items-center space-x-4 flex-1">
      <!-- Search -->
      <div class="relative flex-1 max-w-xl">
        <input
          type="text"
          placeholder="Search files and folders..."
```

```
          [(ngModel)]="searchQuery"
          (ngModelChange)="onSearchChange()"
          class="w-full pl-10 pr-4 py-2 border rounded-lg focus:outline-none focus:ring-2
focus:ring-blue-500"
        />
        <i class="fas fa-search absolute left-3 top-2.5 text-gray-400"></i>
      </div>
      <!-- Upload buttons -->
      <div class="relative">
        <button
          (click)="isUploadMenuOpen = !isUploadMenuOpen"
          class="px-4 py-2 bg-blue-600 text-white rounded-lg flex items-center space-x-2
hover:bg-blue-700"
        >
          <i class="fas fa-upload"></i>
          <span>New</span>
          <i class="fas fa-chevron-down"></i>
        </button>
        <div
          *ngIf="isUploadMenuOpen"
          class="absolute top-full right-0 mt-2 bg-white rounded-lg shadow-lg border p-2
w-48"
        >
          <label class="block p-2 hover:bg-gray-100 rounded cursor-pointer">
            <input
              type="file"
              multiple
              (change)="handleFileUpload($event)"
              class="hidden"
            />
            <span class="flex items-center space-x-2">
              <i class="fas fa-upload"></i>
              <span>Upload files</span>
            </span>
          </label>
          <label class="block p-2 hover:bg-gray-100 rounded cursor-pointer">
            <input
              type="file"
              webkitdirectory
              directory
              (change)="handleFileUpload($event)"
              class="hidden"
            />
            <span class="flex items-center space-x-2">
              <i class="fas fa-folder-plus"></i>
              <span>Upload folder</span>
```

```
          </span>
        </label>
      </div>
    </div>
    <!-- View mode toggles -->
    <div class="flex items-center space-x-2">
      <button
        (click)="viewMode = 'grid'"
        [class.bg-gray-200]="viewMode === 'grid'"
        class="p-2 rounded hover:bg-gray-100"
      >
        <i class="fas fa-grid-2"></i>
      </button>
      <button
        (click)="viewMode = 'list'"
        [class.bg-gray-200]="viewMode === 'list'"
        class="p-2 rounded hover:bg-gray-100"
      >
        <i class="fas fa-list"></i>
      </button>
    </div>
  </div>
</div>
<!-- Sort controls -->
<div class="bg-white border-b px-4 py-2 flex items-center space-x-4">
  <button
    (click)="handleSort('name')"
    [class.bg-gray-200]="sortConfig.key === 'name'"
    class="px-3 py-1 rounded hover:bg-gray-100"
  >
    Name
    <i
      *ngIf="sortConfig.key === 'name'"
      [class.fa-sort-up]="sortConfig.direction === 'asc'"
      [class.fa-sort-down]="sortConfig.direction === 'desc'"
      class="fas"
    ></i>
  </button>
  <button
    (click)="handleSort('date')"
    [class.bg-gray-200]="sortConfig.key === 'date'"
    class="px-3 py-1 rounded hover:bg-gray-100"
  >
    Date
    <i
      *ngIf="sortConfig.key === 'date'"
```

```
        [class.fa-sort-up]="sortConfig.direction === 'asc'"
        [class.fa-sort-down]="sortConfig.direction === 'desc'"
        class="fas"
      ></i>
    </button>
    <button
      (click)="handleSort('size')"
      [class.bg-gray-200]="sortConfig.key === 'size'"
      class="px-3 py-1 rounded hover:bg-gray-100"
    >
      Size
      <i
        *ngIf="sortConfig.key === 'size'"
        [class.fa-sort-up]="sortConfig.direction === 'asc'"
        [class.fa-sort-down]="sortConfig.direction === 'desc'"
        class="fas"
      ></i>
    </button>
  </div>
  <!-- Storage info -->
  <div class="bg-white border-b px-4 py-3">
    <div class="flex items-center justify-between mb-2">
      <span class="text-sm text-gray-600">Storage</span>
      <span class="text-sm text-gray-600" *ngIf="diskDetails">
        {{ formatSize(diskDetails.free) }} free of {{
        formatSize(diskDetails.size) }}
      </span>
    </div>
    <div class="w-full bg-gray-200 rounded-full h-2">
      <div
        class="bg-blue-600 h-2 rounded-full"
        [style.width.%]="diskDetails ? ((diskDetails.size - diskDetails.free) /
diskDetails.size) * 100 : 0"
      ></div>
    </div>
  </div>
  <!-- Breadcrumb navigation -->
  <div class="bg-white border-b px-4 py-2 flex items-center space-x-2">
    <button
      *ngIf="currentPath.length"
      (click)="navigateUp()"
      class="text-gray-600 hover:text-gray-800"
    >
      <i class="fas fa-arrow-left"></i>
    </button>
    <ng-container *ngFor="let path of currentPath; let last = last">
```

```
      <span [class.font-semibold]="last">{{ path }}</span>
      <span *ngIf="!last" class="text-gray-400">/</span>
    </ng-container>
  </div>
  <!-- File grid/list view -->
  <div class="flex-1 overflow-auto p-4">
    <!-- Grid View -->
    <div
      *ngIf="viewMode === 'grid'"
      class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-5
gap-4"
    >
      <div
        *ngFor="let item of currentFolder"
        (click)="item.isFolder ? navigateToFolder(item) : null"
        class="bg-white p-4 rounded-lg shadow hover:shadow-md transition-shadow
cursor-pointer"
      >
        <div class="aspect-square mb-2 flex items-center justify-center">
          <ng-container [ngSwitch]="true">
            <i
              *ngSwitchCase="item.isFolder"
              class="fas fa-folder text-yellow-500 text-4xl"
            ></i>
            <img
              *ngSwitchCase="item.isImage"
              [src]="item.imageData"
              [alt]="item.name"
              class="w-full h-full object-cover rounded"
            />
            <div
              *ngSwitchDefault
              class="w-16 h-16 bg-gray-200 rounded flex items-center justify-center text-sm"
            >
              {{ item.name.split('.').pop().toUpperCase() }}
            </div>
          </ng-container>
        </div>
        <p class="text-sm font-medium truncate">{{ item.name }}</p>
        <p class="text-xs text-gray-500">{{ formatSize(item.stats.size) }}</p>
      </div>
    </div>
    <!-- List View -->
    <div *ngIf="viewMode === 'list'" class="bg-white rounded-lg shadow">
      <div
        *ngFor="let item of currentFolder"
```

```
        (click)="item.isFolder ? navigateToFolder(item) : null"
        class="flex items-center px-4 py-3 border-b last:border-b-0 hover:bg-gray-50
cursor-pointer"
      >
        <div class="w-8">
          <ng-container [ngSwitch]="true">
            <i
              *ngSwitchCase="item.isFolder"
              class="fas fa-folder text-yellow-500"
            ></i>
            <img
              *ngSwitchCase="item.isImage"
              [src]="item.imageData"
              [alt]="item.name"
              class="w-5 h-5 object-cover rounded"
            />
            <div
              *ngSwitchDefault
              class="w-5 h-5 bg-gray-200 rounded flex items-center justify-center text-xs"
            >
              {{ item.name.split('.').pop().toUpperCase() }}
            </div>
          </ng-container>
        </div>
        <div class="flex-1 min-w-0">
          <p class="text-sm font-medium truncate">{{ item.name }}</p>
        </div>
        <div class="text-sm text-gray-500 w-32">
          {{ item.stats.mtime | date }}
        </div>
        <div class="text-sm text-gray-500 w-24 text-right">
          {{ formatSize(item.stats.size) }}
        </div>
      </div>
    </div>
  </div>
  <!-- Upload progress indicator -->
  <div
    *ngIf="isUploading"
    class="fixed bottom-4 right-4 bg-white rounded-lg shadow-lg p-4"
  >
    <div class="flex items-center space-x-3">
      <i class="fas fa-upload animate-bounce"></i>
      <span>Uploading...</span>
    </div>
  </div>
```

```
</div>import { Component, OnInit } from "@angular/core";
import { CommonModule } from "@angular/common";
import { FormsModule } from "@angular/forms";
import { FileService } from "../../services/file.service";
interface SortConfig {
  key: "name" | "date" | "size";
  direction: "asc" | "desc";
}
@Component({
  selector: "app-drive",
  standalone: true,
  imports: [CommonModule, FormsModule],
  templateUrl: "./drive.component.html",
  styleUrls: ["./drive.component.scss"],
})
export class DriveComponent implements OnInit {
  files: any[] = [];
  diskDetails?: { free: number; size: number };
  currentPath: string[] = [];
  currentFolder: any[] = [];
  searchQuery: string = "";
  sortConfig: SortConfig = { key: "name", direction: "asc" };
  viewMode: "grid" | "list" = "grid";
  isUploading = false;
  isUploadMenuOpen = false;
  constructor(private fileService: FileService) {}
  ngOnInit() {
    this.loadFiles();
  }
  loadFiles() {
    this.fileService.getAllFiles().subscribe((response) => {
      this.files = response.allFilesAndDirs;
      this.diskDetails = response.diskDetails;
      this.updateCurrentFolder();
    });
  }
  updateCurrentFolder() {
    let current = this.files;
    for (const path of this.currentPath) {
      const folder = current.find((f) => f.name === path && f.isFolder);
      current = folder?.files || [];
    }
    this.currentFolder = this.sortFiles(this.filterFiles(current));
  }
  formatSize(bytes: number): string {
    const sizes = ["Bytes", "KB", "MB", "GB", "TB"];
```

```
  if (bytes === 0) return "0 Bytes";
  const i = Math.floor(Math.log(bytes) / Math.log(1024));
  return `${Math.round(bytes / Math.pow(1024, i))} ${sizes[i]}`;
}
filterFiles(items: any[]): any[] {
  if (!this.searchQuery) return items;
  return items.filter((item) =>
    item.name.toLowerCase().includes(this.searchQuery.toLowerCase()),
  );
}
sortFiles(items: any[]): any[] {
  return [...items].sort((a, b) => {
    if (this.sortConfig.key === "name") {
      return this.sortConfig.direction === "asc"
        ? a.name.localeCompare(b.name)
        : b.name.localeCompare(a.name);
    } else if (this.sortConfig.key === "date") {
      return this.sortConfig.direction === "asc"
        ? new Date(a.stats.mtime).getTime() -
            new Date(b.stats.mtime).getTime()
        : new Date(b.stats.mtime).getTime() -
            new Date(a.stats.mtime).getTime();
    } else if (this.sortConfig.key === "size") {
      return this.sortConfig.direction === "asc"
        ? a.stats.size - b.stats.size
        : b.stats.size - a.stats.size;
    }
    return 0;
  });
}
handleSort(key: "name" | "date" | "size") {
  if (this.sortConfig.key === key) {
    this.sortConfig.direction =
      this.sortConfig.direction === "asc" ? "desc" : "asc";
  } else {
    this.sortConfig = { key, direction: "asc" };
  }
  this.updateCurrentFolder();
}
async handleFileUpload(event: any) {
  const files = event.target.files;
  if (files.length === 0) return;
  this.isUploading = true;
  const formData = new FormData();
  Array.from(files).forEach((file) => {
    formData.append("files", file);
```

```
    });
    formData.append(
      "folderName",
      this.currentPath[this.currentPath.length - 1] || "root",
    );
    try {
      await this.fileService
        .uploadFiles(
          Array.from(files),
          this.currentPath[this.currentPath.length - 1] || "root",
        )
        .toPromise();
      this.loadFiles();
    } catch (error) {
      console.error("Upload failed:", error);
    } finally {
      this.isUploading = false;
      this.isUploadMenuOpen = false;
    }
  }
  navigateToFolder(folder: any) {
    this.currentPath.push(folder.name);
    this.updateCurrentFolder();
  }
  navigateUp() {
    this.currentPath.pop();
    this.updateCurrentFolder();
  }
  onSearchChange() {
    this.updateCurrentFolder();
  }
}<div class="min-h-screen bg-gray-100">
  <div class="max-w-7xl mx-auto py-12 px-4 sm:px-6 lg:px-8">
    <div class="text-center">
      <h1 class="text-4xl font-bold text-gray-900 mb-8">Welcome to My Drive</h1>
      <p class="text-xl text-gray-600 mb-8">
        Your personal file management system
      </p>
      <a
        routerLink="/drive"
        class="inline-flex items-center px-6 py-3 border border-transparent text-base font-
medium rounded-md text-white bg-blue-600 hover:bg-blue-700"
      >
        Go to Drive
        <svg
          class="ml-2 -mr-1 w-5 h-5"
```

```
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M14 5l7 7m0 0l-7 7m7-7H3"
          />
        </svg>
      </a>
    </div>
  </div>
</div>import { Component } from "@angular/core";
import { RouterLink } from "@angular/router";
@Component({
  selector: "app-home",
  imports: [RouterLink],
  templateUrl: "./home.component.html",
  styleUrl: "./home.component.scss",
  standalone: true,
})
export class HomeComponent {}<nav class="bg-white shadow-lg">
  <div class="max-w-7xl mx-auto px-4">
    <div class="flex justify-between h-16">
      <div class="flex">
        <div class="flex-shrink-0 flex items-center">
          <a routerLink="/" class="text-xl font-bold text-gray-800">My Drive</a>
        </div>
        <div class="hidden sm:ml-6 sm:flex sm:space-x-8">
          <a
            routerLink="/"
            routerLinkActive="border-blue-500 text-gray-900"
            [routerLinkActiveOptions]="{exact: true}"
            class="border-transparent text-gray-500 hover:border-gray-300 hover:text-
gray-700 inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium"
          >
            Home
          </a>
          <a
            routerLink="/drive"
            routerLinkActive="border-blue-500 text-gray-900"
            class="border-transparent text-gray-500 hover:border-gray-300 hover:text-
gray-700 inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium"
          >
```

```
          Drive
        </a>
      </div>
    </div>
  </div>
</nav>import { Component } from "@angular/core";
import { RouterLink, RouterLinkActive } from "@angular/router";
@Component({
  selector: "app-nav",
  imports: [RouterLink, RouterLinkActive],
  templateUrl: "./nav.component.html",
  styleUrl: "./nav.component.scss",
  standalone: true,
})
export class NavComponent {}export interface FileStats {
  size: number;
  mtime: Date;
}
export interface FileItem {
  name: string;
  path: string;
  stats: FileStats;
  isFolder: boolean;
  isImage?: boolean;
  isVideo?: boolean;
  imageData?: string;
  videoId?: string;
  files?: FileItem[];
}
export interface DiskDetails {
  free: number;
  size: number;
}
export interface ServerResponse {
  diskDetails: DiskDetails;
  allFilesAndDirs: FileItem[];
}import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Observable } from "rxjs";
import { ServerResponse } from "../models/file.model";
@Injectable({
  providedIn: "root",
})
export class FileService {
  private apiUrl = "http://localhost:4002";
```

```
  constructor(private http: HttpClient) {}
  getAllFiles(): Observable<ServerResponse> {
    return this.http.get<ServerResponse>(`${this.apiUrl}/all`);
  }
  uploadFiles(files: File[], folderName: string = "uploads"): Observable<any> {
    const formData = new FormData();
    files.forEach((file) => {
      formData.append("files", file);
    });
    formData.append("folderName", folderName);
    return this.http.post(`${this.apiUrl}/upload`, formData);
  }
  getVideoUrl(videoId: string): string {
    return `${this.apiUrl}/video/${videoId}`;
  }
}<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Client</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>import { bootstrapApplication } from "@angular/platform-browser";
import { appConfig } from "./app/app.config";
import { AppComponent } from "./app/app.component";
bootstrapApplication(AppComponent, appConfig).catch((err) =>
  console.error(err),
);@tailwind base;
@tailwind components;
@tailwind utilities;module.exports = {
  content: ["./src/**/*.{html,ts}"],
  theme: {
    extend: {},
  },
  plugins: [],
};{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
```

```
  "files": ["src/main.ts"],
  "include": ["src/**/*.d.ts"]
}{
  "compileOnSave": false,
  "compilerOptions": {
    "outDir": "./dist/out-tsc",
    "strict": true,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "skipLibCheck": true,
    "isolatedModules": true,
    "esModuleInterop": true,
    "experimentalDecorators": true,
    "moduleResolution": "bundler",
    "importHelpers": true,
    "target": "ES2022",
    "module": "ES2022",
    "strictNullChecks": false
  },
  "angularCompilerOptions": {
    "enableI18nLegacyMessageIdFormat": false,
    "strictInjectionParameters": true,
    "strictInputAccessModifiers": true,
    "strictTemplates": true
  }
}{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/spec",
    "types": ["jasmine"]
  },
  "include": ["src/**/*.spec.ts", "src/**/*.d.ts"]
}import express from "express";
import bodyParser from "body-parser";
import cors from "cors";
import { globby } from "globby";
import fs from "fs";
import checkDiskSpace from "check-disk-space";
import multer from "multer";
import path from "path";
const app = express();
app.use(express.json({ limit: "200mb" }));
app.use(express.urlencoded({ limit: "200mb" }));
app.use(cors());
```

```
const directoryPath = "/Users/oshalmay/images";
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    const folderName = req.body.folderName || "uploads";
    const uploadPath = path.join(directoryPath, folderName);
    if (!fs.existsSync(uploadPath)) {
      fs.mkdirSync(uploadPath, { recursive: true });
    }
    cb(null, uploadPath);
  },
  filename: (req, file, cb) => {
    cb(null, file.originalname);
  },
});
const listAllFilesAndDirs = (dir) =>
  globby(`${dir}/**/*`, {
    onlyFiles: false,
    expandDirectories: true,
    objectMode: true,
  });
app.get("/all", async (req, res) => {
  let retVal = {};
  retVal.diskDetails = await checkDiskSpace(directoryPath);
  retVal.allFilesAndDirs = await listAllFilesAndDirs(directoryPath).then(
    async (files) => {
      for await (const file of files) {
        file.stats = fs.statSync(file.path);
        file.isFolder = true;
        const fileIsImage = ["png", "jpg", "jpeg", "gif", "bmp", "tiff"].some(
          (ext) => file.path.endsWith(ext),
        );
        const fileIsVideo = ["mov", "mp4", "avi", "mkv", "flv", "wmv"].some(
          (ext) => file.path.endsWith(ext),
        );
        const parentFolder = file.path.match(/\/([^\/]+)\/[^\/]+$/)?.[1];
        if (fileIsImage || fileIsVideo) {
          file.isFolder = false;
          file.isImage = fileIsImage;
          file.isVideo = fileIsVideo;
          if (fileIsImage) {
            const data = fs.readFileSync(file.path);
            const ext = path.extname(file.path).toLowerCase();
            const mimeType = ext === ".png" ? "image/png" : "image/jpeg";
            file.imageData = `data:${mimeType};base64,${data.toString(
              "base64",
            )}`;
```

```
        }
        if (fileIsVideo) {
          file.videoId = Buffer.from(file.path).toString("base64");
        }
        if (parentFolder) {
          const parentFolderObject = files.find(
            (file) => file.name === parentFolder,
          );
          if (parentFolderObject) {
            parentFolderObject.files = [
              ...(parentFolderObject?.files || []),
              file,
            ];
            files = files.filter((f) => f.path !== file.path);
          }
        }
      }
    }
    return files;
  },
 );
 res.send(retVal);
});
app.get("/video/:videoId", (req, res) => {
 try {
   const videoPath = Buffer.from(req.params.videoId, "base64").toString();
   if (!fs.existsSync(videoPath) || !videoPath.startsWith(directoryPath)) {
     return res.status(404).send("Video not found");
   }
   const stat = fs.statSync(videoPath);
   const fileSize = stat.size;
   const range = req.headers.range;
   if (range) {
     const parts = range.replace(/bytes=/, "").split("-");
     const start = parseInt(parts[0], 10);
     const end = parts[1] ? parseInt(parts[1], 10) : fileSize - 1;
     const chunksize = end - start + 1;
     const file = fs.createReadStream(videoPath, { start, end });
     const head = {
       "Content-Range": `bytes ${start}-${end}/${fileSize}`,
       "Accept-Ranges": "bytes",
       "Content-Length": chunksize,
       "Content-Type": "video/mp4",
     };
     res.writeHead(206, head);
     file.pipe(res);
```

```
    } else {
      const head = {
        "Content-Length": fileSize,
        "Content-Type": "video/mp4",
      };
      res.writeHead(200, head);
      fs.createReadStream(videoPath).pipe(res);
    }
  } catch (error) {
    console.error("Error streaming video:", error);
    res.status(500).send("Error streaming video");
  }
});
const upload = multer({ storage: storage });
app.post("/upload", upload.array("files"), (req, res) => {
  res.send({ message: "Files uploaded successfully" });
});
app.use((err, req, res, next) => {
  if (err instanceof multer.MulterError) {
    return res.status(400).json({ message: err.message });
  } else if (err) {
    return res.status(400).json({ message: err.message });
  }
  next();
});
app.listen(4002, () => {
  console.log("Listening on 4002");
});{
  "name": "posts",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^1.3.4",
    "check-disk-space": "^3.4.0",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "globby": "^14.0.2",
    "multer": "^1.4.5-lts.1",
```

```
    "nodemon": "^2.0.22"
  }
}
```