

node_modules
package-lock.json
Client

This project was generated using [Angular CLI](https://github.com/angular/angular-cli) version 19.0.4.

Development server

To start a local development server, run:

```
``bash
ng serve
``
```

Once the server is running, open your browser and navigate to `http://localhost:4200/`. The application will automatically reload whenever you modify any of the source files.

Code scaffolding

Angular CLI includes powerful code scaffolding tools. To generate a new component, run:

```
``bash
ng generate component component-name
``
```

For a complete list of available schematics (such as `components`, `directives`, or `pipes`), run:

```
``bash
ng generate --help
``
```

Building

To build the project run:

```
``bash
ng build
``
```

This will compile your project and store the build artifacts in the `dist/` directory. By default, the production build optimizes your application for performance and speed.

Running unit tests

To execute unit tests with the [Karma](https://karma-runner.github.io) test runner, use the following command:

```
```bash
ng test
```
```

Running end-to-end tests

For end-to-end (e2e) testing, run:

```
```bash
ng e2e
```
```

Angular CLI does not come with an end-to-end testing framework by default. You can choose one that suits your needs.

Additional Resources

For more information on using the Angular CLI, including detailed command references, visit the [Angular CLI Overview and Command Reference](https://angular.dev/tools/cli) page.

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "client": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:component": {
          "style": "scss"
        }
      },
    },
    "root": "",
    "sourceRoot": "src",
    "prefix": "app",
    "architect": {
      "build": {
        "builder": "@angular-devkit/build-angular:application",
        "options": {
          "outputPath": "dist/client",
          "index": "src/index.html",
          "browser": "src/main.ts",

```

```

    "polyfills": ["zone.js"],
    "tsConfig": "tsconfig.app.json",
    "inlineStyleLanguage": "scss",
    "assets": [
      {
        "glob": "**/*",
        "input": "public"
      }
    ],
    "styles": ["src/styles.scss"],
    "scripts": []
  },
  "configurations": {
    "production": {
      "budgets": [
        {
          "type": "initial",
          "maximumWarning": "500kB",
          "maximumError": "1MB"
        },
        {
          "type": "anyComponentStyle",
          "maximumWarning": "4kB",
          "maximumError": "8kB"
        }
      ],
      "outputHashing": "all"
    },
    "development": {
      "optimization": false,
      "extractLicenses": false,
      "sourceMap": true
    }
  },
  "defaultConfiguration": "production"
},
"serve": {
  "builder": "@angular-devkit/build-angular:dev-server",
  "configurations": {
    "production": {
      "buildTarget": "client:build:production"
    },
    "development": {
      "buildTarget": "client:build:development"
    }
  }
},

```

```

    "defaultConfiguration": "development"
  },
  "extract-i18n": {
    "builder": "@angular-devkit/build-angular:extract-i18n"
  },
  "test": {
    "builder": "@angular-devkit/build-angular:karma",
    "options": {
      "polyfills": ["zone.js", "zone.js/testing"],
      "tsConfig": "tsconfig.spec.json",
      "inlineStyleLanguage": "scss",
      "assets": [
        {
          "glob": "**/*",
          "input": "public"
        }
      ],
      "styles": ["src/styles.scss"],
      "scripts": []
    }
  }
}
}
}
}
{
  "name": "client",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^19.0.0",
    "@angular/common": "^19.0.0",
    "@angular/compiler": "^19.0.0",
    "@angular/core": "^19.0.0",
    "@angular/forms": "^19.0.0",
    "@angular/platform-browser": "^19.0.0",
    "@angular/platform-browser-dynamic": "^19.0.0",
    "@angular/router": "^19.0.0",
    "rxjs": "~7.8.0",

```

```

    "tslib": "^2.3.0",
    "zone.js": "~0.15.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^19.0.4",
    "@angular/cli": "^19.0.4",
    "@angular/compiler-cli": "^19.0.0",
    "@types/jasmine": "~5.1.0",
    "autoprefixer": "^10.4.20",
    "jasmine-core": "~5.4.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.2.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.1.0",
    "postcss": "^8.4.49",
    "tailwindcss": "^3.4.16",
    "typescript": "~5.6.2"
  }
}
<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,  
  styleUrls: ["./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: "" }, // Redirect invalid routes to home
];
<!-- 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)="viewModel = 'grid'"
  [class.bg-gray-200]="viewModel === 'grid'"
  class="p-2 rounded hover:bg-gray-100"
>
  <i class="fas fa-grid-2"></i>
</button>
<button
  (click)="viewModel = 'list'"
  [class.bg-gray-200]="viewModel === '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>
// drive.component.ts
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) => {
      // @ts-ignore
      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",
  styleUrls: "./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>
  </div>
</nav>

```

```

import { Component } from "@angular/core";
import { RouterLink, RouterLinkActive } from "@angular/router";

```

```

@Component({
  selector: "app-nav",
  imports: [RouterLink, RouterLinkActive],
  templateUrl: "./nav.component.html",
  styleUrls: ["./nav.component.scss"],
  standalone: true,
})
export class NavComponent {}
// src/app/models/file.model.ts
export interface FileStats {
  size: number;
  mtime: Date;
  // Add other stats as needed
}

```

```

}

export interface FileItem {
  name: string;
  path: string;
  stats: FileStats;
  isFolder: boolean;
  isImage?: boolean;
  isVideo?: boolean;
  imageData?: string; // The base64 image data
  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;
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["/src/**/*.{html,ts}"],
  theme: {
    extend: {},
  },
  plugins: [],
};
/* To learn more about Typescript configuration file: https://www.typescriptlang.org/docs/handbook/tsconfig-json.html. */
/* To learn more about Angular compiler options: https://angular.dev/reference/configs/angular-compiler-options. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",

```

```

    "types": []
  },
  "files": ["src/main.ts"],
  "include": ["src/**/*.d.ts"]
}
/* To learn more about Typescript configuration file: https://www.typescriptlang.org/docs/
handbook/tsconfig-json.html. */
/* To learn more about Angular compiler options: https://angular.dev/reference/configs/
angular-compiler-options. */
{
  "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
  }
}
/* To learn more about Typescript configuration file: https://www.typescriptlang.org/docs/
handbook/tsconfig-json.html. */
/* To learn more about Angular compiler options: https://angular.dev/reference/configs/
angular-compiler-options. */
{
  "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(/\V(?:\V+)?\V(?:\V+)?$/)?.[1];

if (fileIsImage || fileIsVideo) {
  file.isFolder = false;
  file.isImage = fileIsImage;
  file.isVideo = fileIsVideo;

  // Read and convert image to base64
  if (fileIsImage) {
    const data = fs.readFileSync(file.path);
    // Get file extension for proper mime type
    const ext = path.extname(file.path).toLowerCase();
    const mimeType = ext === ".png" ? "image/png" : "image/jpeg";
    // Convert to base64 with proper data URI
    file.imageData = `data:${mimeType};base64,${data.toString(
      "base64",
    )}`;
  }

  // For videos, just send metadata and video ID
  if (fileIsVideo) {
    file.videoId = Buffer.from(file.path).toString("base64"); // Simple way to create a
unique ID
  }

  // Handle parent folder relationship
  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);
  });

  // New route for video streaming
  app.get("/video/:videoid", (req, res) => {
    try {
      const videoPath = Buffer.from(req.params.videoid, "base64").toString();

      // Verify the file exists and is within allowed directory
      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" });
});
// Error handling middleware
app.use((err, req, res, next) => {
  if (err instanceof multer.MulterError) {
    // Multer error occurred (e.g., file size exceeded)
    return res.status(400).json({ message: err.message });
  } else if (err) {
    // Other errors (e.g., unsupported file type)
    return res.status(400).json({ message: err.message });
  }
  next();
});
// app.post("/upload", async (req, res) => {
//   const {images, folderName} = req.body;
//   images.forEach(image => {
//     const {imageName, data} = image;
//     const path = `${directoryPath}/${folderName}/${imageName}`;
//     fs.writeFileSync(path, Buffer.from(data, 'base64'));
//   });
//   res.send({message: "Images uploaded successfully"});
// });

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"  
}  
}
```