modeldb.component.html

<div class="input-form">  
<h3>Upload File</h3>  
<input class="mat-raised-button" type="file" (change)="onFileSelected($event)" multiple/>  
<button class="btn-primary" type="button" (click)="onUpload()">Upload</button>  
  
<div class="row" *\*ngIf*="filesToUpload?.length > 0">  
 <div class="table">  
 <ul class="files" *\*ngFor*="**let** file of filesToUpload">  
 <li><a>{{file.name}}</a></li>  
 </ul>  
 </div>  
</div>  
</div>  
  
<mat-form-field class="project-form">  
 <mat-label>Project</mat-label>  
 <mat-select required (selectionChange)="projectSelect($event.value)">  
 <mat-option value="CNN">CNN</mat-option>  
 <mat-option value="FashionMnist">FashionMnist</mat-option>  
 <mat-option value="LSTM">LSTM</mat-option>  
 </mat-select>  
</mat-form-field>  
  
<mat-form-field *\*ngIf*="files" class="experiment-form">  
 <mat-select required (selectionChange)="experimentSelect($event.value)">  
 <mat-label>Experiments</mat-label>  
 <div *\*ngFor*="**let** file of files">  
 <mat-option *\*ngIf*="file.name.includes(event) > 0 && file.name.includes('files') > 0" value="{{file.name}}">{{file.name.substring(0, file.name.lastIndexOf('.'))}} </mat-option>  
 </div>  
 </mat-select>  
</mat-form-field>  
  
<div *\*ngIf*="files1" class="experiment-files">  
 <div *\*ngFor*="**let** fLinks of files1">  
 <a href="/api/file/{{event1}}/{{fLinks.filename}}">{{fLinks.filename}}</a>  
 </div>  
</div>

modeldb.component.css

h3 {  
 padding: 10px;  
 text-align: center;  
}  
  
.input-form {  
 text-align: center;  
}  
  
.table {  
 margin: 0 auto;  
 display: table;  
}  
  
li {  
 display: inline-block;  
 margin-top: 10px  
}  
  
.project-form, .experiment-form {  
 margin-top: 25px;  
 margin-left: 30%;  
 margin-right: 30%;  
 width: 40%;  
}  
  
.experiment-files{  
 margin-top: 25px;  
 margin-left: 30%;  
 margin-right: 30%;  
 width: 40%;  
 text-align: right;  
}

modeldb.component.ts

**import** { Component, OnInit } **from** '@angular/core';  
**import** { HttpClient } **from** '@angular/common/http';  
  
  
@Component({  
 selector: 'app-modeldb',  
 templateUrl: './modeldb.component.html',  
 styleUrls: ['./modeldb.component.css']  
})  
**export class** ModeldbComponent **implements** OnInit {  
  
 filesToUpload: Array<File> = [];  
 files: **any**;  
 files1: **any**;  
 event: **any**;  
 event1: **any**;  
  
 **constructor**(**private** http: HttpClient) {  
 }  
  
 ngOnInit() {  
 }  
  
 onFileSelected(fileInput: **any**) {  
 **this**.filesToUpload = fileInput.target.files **as** Array<File>;  
 }  
  
 onUpload() {  
 **const** fd = **new** FormData();  
 **const** files: Array<File> = **this**.filesToUpload;  
 console.log(files);  
  
 **for** (**let** i = 0; i < files.length; i++) {  
 fd.append('uploads[]', files[i], files[i].name);  
 }  
 **this**.http.post('/api/upload', fd)  
 .subscribe(res => {  
 console.log(res);  
 });  
 }  
  
 projectSelect(event) {  
 **this**.event = event;  
 **this**.files1 = '';  
 console.log('EVENT -> ' + **this**.event);  
 **this**.http.get('/api/all')  
 .subscribe(res => {  
 **this**.files = JSON.parse(JSON.stringify(res));  
 console.log(**this**.files);  
 });  
 }  
  
 experimentSelect(event) {  
 **this**.event1 = event.substring(0, event.lastIndexOf('.'));  
 console.log('EVENT 1 -->' + **this**.event1);  
 **this**.http.get('/api/collection/' + **this**.event1)  
 .subscribe(res => {  
 **this**.files1 = JSON.parse(JSON.stringify(res));  
 console.log(**this**.files1);  
 });  
 }  
  
}

app.js

**const** express = require('express');  
**const** bodyParser = require('body-parser');  
**const** path = require('path');  
**const** crypto = require('crypto');  
**const** mongoose = require('mongoose');  
**const** multer = require('multer');  
**const** GridFsStorage = require('multer-gridfs-storage');  
**const** Grid = require('gridfs-stream');  
**const** methodOverride = require('method-override');  
**const** cors = require('cors');  
  
**const** app = express();  
  
app.use(bodyParser.json());  
app.use(cors());  
app.use(methodOverride('\_method'));  
app.set('view engine', 'ejs');  
app.use(**function**(req, res, next) {  
 res.header("Access-Control-Allow-Origin", "\*");  
 res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");  
 next();  
});  
  
app.use(express.static(path.join(\_\_dirname, 'public')));  
  
**const** mongoURI = 'mongodb+srv://modelkb:modelkb@modelkb-0dva5.gcp.mongodb.net/uploads?retryWrites=true';  
  
**const** conn = mongoose.createConnection(mongoURI);  
  
**let** gfs;  
  
conn.once('open', () => {  
 gfs = Grid(conn.db, mongoose.mongo);  
});  
  
**const** storage = **new** GridFsStorage({  
 url: mongoURI,  
 file: (req, file) => {  
 **return new** Promise((resolve, reject) => {  
 crypto.randomBytes(16, (err, buf) => {  
 **if** (err) {  
 **return** reject(err);  
 }  
 **const** filename = file.originalname;  
 **const** bucketname = file.originalname.substring(0, file.originalname.lastIndexOf("\_"));  
 **const** fileInfo = {  
 filename: filename,  
 bucketName: bucketname  
 };  
 resolve(fileInfo);  
 });  
 });  
 },  
});  
**const** upload = multer({ storage });  
  
app.post('/api/upload', upload.array("uploads[]", 12),(req, res)=> {  
 //console.log(req.name.substr(0, req.name.indexOf('-')));  
 res.json({array: req.array});  
});  
  
app.get('/api/collection/:bucketName', (req, res) => {  
 gfs.collection(req.params.bucketName);  
 gfs.files.find().toArray((err, files) => {  
 **if**(!files || files.length === 0) {  
 **return** res.status(404).json({  
 err: 'No files exist'  
 });  
 }  
 **return** res.json(files);  
 });  
});  
  
app.get('/api/all', (req, res) => {  
 conn.db.listCollections().toArray((err, collInfos) => {  
 **if**(!collInfos) {  
 **return** res.status(404).json({  
 err: 'No Collections Exist'  
 });  
 }  
 **return** res.json(collInfos);  
 });  
});  
  
app.get('/api/collection/:bucketName/:filename', (req, res) => {  
 gfs.collection(req.params.bucketName);  
 gfs.files.findOne({filename: req.params.filename}, (err, file) => {  
 **if**(!file || file.length === 0) {  
 **return** res.status(404).json({  
 err: 'No file exist'  
 });  
 }  
 **return** res.json(file);  
 });  
});  
  
app.get('/api/file/:bucketName/:filename', (req, res) => {  
 gfs.collection(req.params.bucketName);  
 gfs.files.findOne({filename: req.params.filename}, (err, file) => {  
 **if**(!file || file.length === 0) {  
 **return** res.status(404).json({  
 err: 'No file exist'  
 });  
 }  
 **if**(file.contentType === 'image/jpeg' || file.contentType === 'img/png' || file.contentType === "text/plain" || file.contentType === "application/octet-stream") {  
 **const** readstream = gfs.createReadStream(file.filename);  
 console.log(readstream.pipe(res));  
 readstream.pipe(res);  
 } **else** {  
 res.status(404).json({  
 err: 'Not an image or plain text file'  
 });  
 }  
 });  
});  
  
**const** port = 5000;  
  
app.listen(port, () => console.log(`Server started on port ${port}`));