```
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");
});
```