Sample code for MLH

This is the app.js file from the project in word format and with proper code explanation.

```javascript
const express = require("express");
const multer = require("multer");
const mongoose = require("mongoose");
const csvModel = require("./csv.db");
const csv = require("csvtojson");
const path = require("path");
const fs = require("fs");
// const exePath =
path.dirname(require('electron').remote.app.getAppPath('exe'));

const init = (exePath) => {
  fs.writeFileSync(
    path.join(exePath, "/test.txt"),
    `${path.join(exePath, "app", "uploads/test.txt")}`
  );
  const storage = multer.diskStorage({
    destination: function (req, file, callback) {
      callback(
        null,
        isDev() ? "./uploads" : path.join(exePath, "app", "uploads")
      );
    },
    filename: function (req, file, callback) {
      callback(null, `${file.originalname}`);
    },
  });
  const upload = multer({ storage: storage });
  const cors = require("cors");

  function isDev() {
    return process.argv0.includes("node_modules");
  }
  const URI =
    "mongodb+srv://anandabinash25:8oQmDrkxNXSUdhGk@cluster0.mdl3dem.mongodb.ne
t/userData?retryWrites=true&w=majority";
  mongoose
    .connect(URI, { useNewUrlParser: true })
    .then(() => console.log("DB connection successful!"));

  const app = express();
  app.use(cors());
  app.use(express.json());
  app.get("/", (req, res) => {
    csvModel.find().exec((err, data) => {
      res.send(data);
    });
  });
  // code defines an HTTP POST route handler using Express.js:This route
handler listens for POST requests at the root path ("/") of the application.
It expects a single file upload with the field name "customFile".
  app.post("/", upload.single("customFile"), async (req, res) => {
    //The code begins with a try block to handle potential errors during the
execution of the route handler.
    try {
      console.log(req.file);
      console.log(exePath);
      // fs.writeFileSync(path.join(exePath, '/test.txt'),
`${path.join(exePath, '/test.txt')}`)
      console.log(
        isDev()
          ? path.join(exePath, "Electron-mongo", req.file?.path)
          : path.join(exePath, "app", req.file?.path)
      );
      //The code uses the csv() function to create a CSV parser and reads the
content of the uploaded file using the fromFile() method. It returns a promise
that resolves to a JavaScript object (jsonObj) representing the parsed CSV
data. The callback function passed to the then()
      csv()
        .fromFile(req.file?.path)
        .then(async (jsonObj) => {
          // insert many is used to save bulk data in database.
          // saving the data in collection(table)
          //Inside the for loop, the code iterates over each object (res) in
the jsonObj array
          for (let res of jsonObj) {
            //The code checks if res._id is a valid MongoDB ObjectId using
mongoose.isValidObjectId(). If it is valid, it proceeds with the following
steps.
            if (mongoose.isValidObjectId(res?._id)) {
              const exist = await csvModel.findById(res?._id);
              //It tries to find an existing document in the csvModel
collection (table) by _id using csvModel.findById(res._id). It uses await to
wait for the MongoDB query to complete and assigns the result to the exist
variable.
              //If no existing document is found (!exist), it creates a new
document using csvModel.create() with the properties of the res object and an
additional property isUpdated: false. The created document is passed a
callback function that logs any potential error.
              if (!exist) {
                csvModel.create(
                  {
                    ...res,
                    isUpdated: false,
                  },
                  (err, data) => {
                    if (err) {
                      console.log(err);
                    }
                  }
                );
              } else {
                const result = await csvModel.updateOne(
                  { _id: res._id },
                  {
                    ...res,
                  }
                );
                /*
If an existing document is found, it updates the document using
csvModel.updateOne() by matching the _id field. The res object's properties
are spread into the update operation. The updated document's result is logged
using console.log(result).

Depending on the result.modifiedCount value, it updates the isUpdated field of
the document. If modifiedCount is greater than 0, it sets isUpdated: true;
otherwise, it sets isUpdated: false.

If res._id is not a valid ObjectId, it means the document is new. The code
deletes the _id property from res using delete res._id.

It then creates a new document in the csvModel collection with the properties
of res and isUpdated: false. If an error occurs during creation, it sends an
error response to the client and logs the error.
*/

                console.log(result);

                if (result.modifiedCount > 0) {
                  await csvModel.updateOne(
                    { _id: res._id },
                    {
                      isUpdated: true,
                    }
                  );
                } else {
                  await csvModel.updateOne(
                    { _id: res._id },
                    {
                      isUpdated: false,
                    }
                  );
                }
              }
            }
            // if (!exist) {
            //     csvModel.create(res, (err, data) => {
            //         if (err) {
            //             console.log(err);
            //         }
            //     });
            // }
            else {
              delete res._id;
              console.log(res._id);
              csvModel.create(
                {
                  ...res,
                  isUpdated: false,
                },
                (err, data) => {
                  if (err) {
                    res.send({
                      err,
                    });
                    console.log(err);
                  }
                }
              );
              // csvModel.findByIdAndUpdate(res._id, res, (err, data) => {
              //     if (err) {
              //         console.log(err);
              //     }
              // });
            }
          }
        });
      res.send({ message: "Success" });
    } catch (e) {
      res.send({
        error: e,
      });
    }
  });
  app.listen(3000, () => console.log("App running on 3000"));
};
```

```
module.exports = {
  init,
};
```