```
Print("File-LLM");

import dotenv from "dotenv";
dotenv.config();
import { ChatOpenAI } from "@langchain/openai";
import { ChatPromptTemplate } from "@langchain/core/prompts";
import { StringOutputParser } from "@langchain/core/output_parsers";
import { createStuffDocumentsChain } from "langchain/chains/combine_documents";
import { Document } from "@langchain/core/documents";
import { MemoryVectorStore } from "langchain/vectorstores/memory";
import { CheerioWebBaseLoader } from "langchain/document_loaders/web/cheerio";
import { RecursiveCharacterTextSplitter } from "langchain/text_splitter";
import { OpenAIEmbeddings } from "@langchain/openai";
// import langchain from "langchain";
import { PDFLoader } from "langchain/document_loaders/fs/pdf";
import express from "express";
import { fileURLToPath } from "url";
import path, { dirname } from "path";
import multer from "multer";

const key = process.env.OPEN_AI_API_KEY;

async function main(inputPrompt, fileName) {
  const chatModel = new ChatOpenAI({
    openAIApiKey: key,
  });

  const filePath = path.join(
    fileURLToPath(import.meta.url),
    "../pdfs",
    fileName
  );

  const loader = new PDFLoader(filePath, { splitPages: true });
  const docs = await loader.load();

  const splitter = new RecursiveCharacterTextSplitter();
  const splitDocs = await splitter.splitDocuments(docs);

  Print(splitDocs);

  // const embeddings = new OpenAIEmbeddings({
  //   openAIApiKey: key,
  // });

  // const vectorstore = await MemoryVectorStore.fromDocuments(
  //   splitDocs,
  //   embeddings
  // );

  // const langchainInstance = new langchain();
  // langchainInstance.addVectorStore(vectorstore);

  const prompt =
    ChatPromptTemplate.fromTemplate(‘Answer the following question based only on the provided context
```

```
:
        <context>
        {context}
        </context>
    Question: {input}
‘);

  const documentChain = await createStuffDocumentsChain({
    llm: chatModel,
    prompt: prompt,
  });

  const response = await documentChain.invoke({
    input: inputPrompt,
    context: [
      new Document({
        pageContent: docs[0]?.pageContent,
      }),
    ],
  });

  return response;
}

function Print(arg) {
  console.log({ arg });
}

const app = express();
const port = 3000;

app.use(express.json());

app.get("/", async (req, res) => {
  return res.sendFile(
    path.join(dirname(fileURLToPath(import.meta.url)), "index.html")
  );
});

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "pdfs/");
  },
  filename: function (req, file, cb) {
    cb(null, file.originalname);
  },
});

const upload = multer({ storage: storage });

app.post("/upload", upload.single("file"), async (req, res) => {
  try {
    if (!req.file) {
      return res.json({
        message: "No file uploaded",
```

```
      });
    }
    if (req.file.mimetype !== "application/pdf") {
      return res.json({
        message: "currently only pdf files are supported",
      });
    }
    return res.json({
      status: true,
      message: "File is uploaded",
      data: {
        name: req.file?.originalname,
        mimetype: req.file?.mimetype,
        size: req.file?.size,
      },
    });
  } catch (error) {
    console.log(error);
    return res.send(‘Error when trying upload image: ${error}‘);
  }
});

app.post("/chat", async (req, res) => {
  const { input, fileName } = req.body;
  if (!input || !fileName) {
    return res.json("File is not uploaded!!!");
  }
  const response = await main(input, fileName);
  return res.json(response);
});

app.listen(port, () => {
  console.log(‘Example app listening at http://localhost:${port}‘);
});
```