```
require('dotenv').config();
const { Client } = require("@octoai/client");
// Access the API Token

// Selecting the prompt for different requests from the user
const prompts = require('prompts'); // Installed prompts
//! https://www.npmjs.com/package/prompts

// Integrate yargs if needed
// const yargs = require('yargs');

const pdf = require('pdf-parse'); // Installed pdf-parse

// Since PDF will be located in the 'files' folder
const fs = require('fs/promises'); // Using promises for async operations
const path = require('path');

const client = new Client(process.env.OCTOAI_TOKEN); // Request to access the large language model

(async () => {
 // Now access the OctoAI models and map them for prompts
 let models = await client.chat.listAllModels().map(model => ({
  title: model,
  value: model
 }));

 // Give the choice to access different model types
 const modelSelected = await prompts({
  type: 'select',
  name: 'model',
```

```
  message: 'Which model would you like to use?',
  choices: models,
  initial: 7 // Initial model (adjust as needed)
});

const allfiles = await fs.readdir('./files');

// Filter all PDFs in the folder
let pdfs = allfiles.filter(file => path.extname(file).toLowerCase() === '.pdf');

// Map and create choices for prompts
let choices = pdfs.map(pdf => ({
  title: pdf,
  value: pdf
}));

const pdfselected = await prompts([
  {
    type: 'select',
    name: 'pdf',
    message: 'Which PDF would you like to use?',
    choices: choices
  }
]);

// Read the selected PDF
const dataBuffer = await fs.readFile(`./files/${pdfselected.pdf}`);



let text ; //  declere  as due to try and catch it very unaccessablethe text
// Extract text using pdf-parse (assuming async behavior)
```

```
 try {
   text = await pdf(dataBuffer).then(data => data.text);
  console.log(text);
} catch (error) {
  console.error("Error extracting text:", error.message);
  // Handle the error gracefully (e.g., inform the user, retry, etc.)
}

 // Your remaining code (e.g., interaction with OctoAI)


//this to pull out the text from the file

  //This is the promt of the prompts


 // this is the logic for the AI text prpm the user see
   const completion = await client.chat.completions.create({
      "model": "llama-2-13b-chat-fp16", //this came a varabile option for  choseing the model
      "messages": [//passing an array of meesage how the ai should function
         {
            "role": "system",
            "content": "Summrise the following pdf ",
         },
         {
            "role": "user",
            "content": "pdf content:\n"+text,
         }
      ],
   })
     console.log(completion.choices[0].message.content);
```

```
})();


//!https://www.npmjs.com/package/pdf-parse   sue pre javascript to extarct data form the pdf


//!   this model's context length (4096)
```