-> Tech Stack :-

* Resume Builder UI :- HTML, CSS, JavaScript, Bootstrap, JQuery
* Resume Builder API :- Node JS, Express JS

**APPROACH:-**

* The PDFs are converted to JSON Object using Extract API and Parsed to JS Object from which different fields are filtered.
* The PDFs Given in Test Dataset have a fixed Format. The Alignment from left margin is fixed for every Text element. Also, most of the elements are extracted in the same order everytime.
* So, for the elements that are easily and accurately detected by the Extract API, the code just iterates over the parsed Object and keeps on separating the respective fields as they appear in it. Fields Like Company Name, Company Address, Company Description, Invoice No, Invoice Issue Date, Item Details are Filtered in this way.
* For Rest of the elements, that are sometimes not detected distinctly and properly, The code traverses the parsed Object and based on the left Bound, groups the Text Elements. Then separate respective fields from these groups of data created.
* Sometimes, elements close to each other and similar in style are being extracted as a group. So, to avoid mistakes in filtering the fields, All the Elements close to each other are grouped as a string/array and then the individual fields are separated from it.

**CODE BREAKDOWN AND WORKFLOW:-**

* **app.js :-** This is the main file from where execution starts. It first sets up the SDK using the credentials and then traverses each PDF. For Every PDF, It sends the request to Extract API and then stores the result in an array. Once all the PDFs are extracted, the filtering of fields starts.
* **extractBillDetails.js :-** This Module contains the code for filtering the fields from the object array of all PDFs. It traverses over the respective object of each PDF and first filters out the fields based on the Left Bounds and after that, filters out the fields based on their order in the Object.
* **Create\_CSV :-** This folder contains the modules to create the CSV File. The column.js file contains the detail about the columns Title in the CSV File and the create\_CSV.js file uses csv-writer node module to set up the csv writer and contains a function to add array of record in the file.
* **Credentials Folder** contains the SDK credentials files( credentials.json file and private key) .
* config folder is for managing the default SDK logging.
* zip\_Files is for zip files created by the extract API for every PDF.
* InvoicesData Folder is the folder provided in question.
* **ExtractedData Folder** contains the output CSV file.

🡪 Make Sure the CSV file is not open anywhere while executing the code, else

it will land to a Read Write OS Conflict and **Give Error.** Also, Make Sure to

**install node modules** before running the code. Can execute the code using **npm start** or **node app.js** command.

**MUST READ KEY POINTS :-**

* The API can Handle Multiple Requests at the Same Time. The Rate Limit is dependent on the Rate Limit of Document Generation API. The API Response Time is primarily dependent on the response of Document Generation API.
* **Error Handling :-** If Some PDF is not able to be extracted by the API due to any server side issue, time out, etc. The request is sent again in order to not miss any PDF in our final CSV File.

So, The Code gives an **accurate result** and has been verified several times.