**Tech Stack :-**

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

**BASIC WORKFLOW:-**

* The **app.js** is the main file from where execution begins. It creates an **express application** and listens on port 8080. Also, the required setup for the Document Generation SDK is done at the same time.
* If the application receives a **GET Request at ‘/’ endpoint**, it sends the **Index.html** file to load the **RESUME BUILDER UI.**

The UI follows Client-Side Rendering to Provide Smooth User Experience.

* On receiving **POST Request at ‘/resume’ endpoint**, the application calls the

**createResume** Middleware Function.

* This function first **verifies the Request Object** if it follows the expected format and all fields and subfields are present, their type is correct, they are non-empty and if there are no extra fields present.
* It then extracts all the fields data and **creates a valid JSON Object** to send to the Document Generation API.

After successful verification and extraction, the Document Merge Operation is instantized and executed. On receiving a success Response, the PDF file is sent in the response.

* Throughout the process, If the Application encounters any error or finds an invalid request object, It sends an error message in response.

**MAJOR COMPONENTS and their FUNCTIONALITIES:-**

* **app.js :-** This is the main file from where execution starts. It creates an express application, listen and serves the requests, imports all the necessary modules which sets up the SDK.
* **Controller :-** This Folder contains the JS Modules.
* **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.