**RESUME BUILDER**

The Project Code is for a Resume Builder which allows users to create Resumes quickly and easily.

The users can fill the multi-step user-friendly **Form** through the interface and start the generation process. On successful verification of Data, the API generates the Resume using **Document Generation API** and sends it back to the UI. The UI displays the PDF using the **Adobe Embed API** allowing the user to download it too. In case of any error, the user can again start the process by just modifying the Form as the state of Form remains same.

**Tech Stack :-**

* Resume Builder UI :- HTML, CSS, JavaScript, Bootstrap, jQuery, Adobe Embed API.
* Resume Builder API :- Node JS, Express JS, npm packages, Adobe Document Generation API.

**API DESCRIPTION:-**

* Endpoints :-
  + GET / :- To load the UI.
  + POST /resume :- Accepts a request object and uses the Document Generation API to create response and send back.
* Request Object :- As mentioned in the description, the request object should strictly contain all the **response headers and fields**. In case any of them is in a wrong format, missing or contains an invalid data, request is rejected.

For **attaching Hyperlinks** to the text, it should be written in this format: **<a href=\"**URL**\">**Text**</a>** wherever required in the request object **except for LinkedIn URL.** For that, only URL is required.

* Response Object :- On successful generation of Resume, it is sent in response with proper headers. In case of errors, the respective error status code is sent with a Description Message in JSON Format.
* Expected Error Status Codes and causes:-
  + 400 :- The Request object is invalid. It either contains wrong headers or invalid data.
  + 401 :- If authorization fails with the Document Generation API.
  + 404 :- The Resume Template is not found.
  + 500 :- If some error occurs at the Server side.

**BASIC WORKFLOW OF API:-**

* The **server.js** is the main file from where execution begins. It imports the **app module** which is an **express application** and makes it listen on port 8080. Also, the required setup for the Document Generation SDK is done at the same time by other imported modules.
* 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.
* 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.

**RESUME BUILDER API CODE DESCRIPTION:-**

* **server.js :-** This is the main file from where execution starts. It imports an express app and makes it listen on port 8080.
* **app.js :-** It creates an express application and imports all the necessary modules which sets up the SDK.
* **Controller :-** This Folder contains all the JS Modules.
  + **verifyRequest.js :-** This module takes as input a Request Object and then verifies it. It verifies the request headers, Format of all the required fields and subfields and makes sure no extra fields are present. It also makes sure no field/sub field is empty.
  + **createJSON.js :-** This Module takes as input a Data Object and returns a JSON Object in the format to be sent to the Document Generation API.

It also trims the strings to remove unnecessary spaces at the beginning/end, **converts the LinkedIn URL to Hyperlink**. For rest of the Hyperlinks, they must be passed in the request Data directly.

* + **createResume.js :-** This Module first sets up the Document Generation SDK. It has a Middleware function that takes as input a request and response object, uses above Modules to verify the same and extract JSON Object.

It then sends a request to the Document Generation API and sends the PDF/Error response.

It uses a request count variable to help in uniquely naming output PDFs. As soon as the task finishes, the output PDF is removed to save space and the count variable resets when no active requests are present.

* **Tests :-** This folder contains the **Unit Testing File** for multiple test cases.
* **Credentials Folder** contains the SDK credentials files(credentials.json file and secret.js file which contains client ID and Secret Key) .
* **ResumeTemplates** Folder stores the Templates docx format and **TemporaryResumes** Folder stores the API output pdfs temporarily.
* **config folder** is for managing the default SDK logging.

**RESUME BUILDER UI CODE DESCRIPTION:-**

* **Views :-** This folder contains the **index.html** file that is the Homepage for the Resume Builder UI.
* **Public :-** This folder contains all the **static files** for the UI. It includes multiple **HTML components**, all **CSS files** and different **script files**. Also contains the sample PDFs to be shown when selecting Template.

**MUST READ KEY POINTS :-**

* The API can Handle **Multiple Requests** Independently at the Same Time. The Rate Limit and Response Time is primarily dependent on that of Document Generation API.
* The PDF files generated by the SDK are deleted just after response is sent and so, no Information is cached between requests as well as Space is optimized.
* The API has been **unit tested for multiple test cases**. Also, a lot of **manual testing** is done both for UI and API. Also, the code has been well commented, organized and modularized to for easier understanding.
* The UI has been **designed pretty well** in an **efficient**, **responsive** and **user friendly** manner and at the same time to provide a nice and smooth experience. It contains all the necessary **data checks** and also includes **tips** to help the user. It uses the Adobe PDF Embed API to display the Resume PDFs.

So, the code has been designed to be easy to use, maintain, accurate as well as provide an amazing experience throughout.