Backend Documentation (PHP)

Files:

1. Index.php

This is the home page of the website, where the user enters their data and uploads their resume.

2. sendData.php

This is the meat and potatoes of the backend code. All of the good stuff happens here – including but not limited to database updation, test case verification, file transfer, pdf parsing, and regular expression matching!

3. test.php

This is the test file designed to run parts of the code of sendData.php and return the results of those test cases. It works by checking if the correct entries are input into the database tables and if the IDs of the tables are correctly mapped to each other.

Variables:

1. sendData.php

- string **inputName** the name of the user including first and last names
- string **inputEmail** the email ID of the user
- int **inputJobTypeId** the ID key corresponding to the chosen job by the user
- string **target_file** the url of the pdf file to be uploaded
- array(int **resume_pass**, int **user_pass**) array of 2 integer values that indicate respectively whether the resume table code test cases and the user table code test cases have passed. 1-> pass, 0-> fail
- **conn** the connection variable through which we will run most of our queries.
- object **paramsFile**
- **params** associative array(string) extracts content from parameters.json and stores in itself.
- **servername** string and
- **username** string and
- **password** string and
- **db** string variables store the connection parameters for $conn
- **firstname** string stores first name
- **lastname** string stores last name
- **active** int and
- **created**_by int and
- **updated**_by int store placeholder values for updating tables

- **test_init_usermaster_userid** int and
- **test_init_resumemaster_resumeid** int store the initial max values
- **parser** Class Object pdfparser object
- **pdf** pdf Object
- **text** string stores the string data of the pdf file
- **max** int stores ID of latest resume
- **len** int stores length of the query result
- **test_fin_usermaster_userid** int and
- **test_fin_resumemaster_resumeid** int and
- **test_fin_userresume_resumeid** int and
- **test_fin_userresume_userid** store the values of the max IDs of their 4 respective tables to check for test pass/fail

2. index.php

- **skill_ids** array(int) stores the IDs of the job_master table entries
- **skill_array** array(string) stores the titles of the job_master table entries.

Code Segments:

1. sendData.php:

- *function executer($inputName, $inputEmail, $inputJobTypeId, $target_file){…*
    - The function executer runs the entire backend code after the submission of the form from index.php. Here, it uses the form data and the uploaded file to update the database tables with the user's information and matched skills.
- *include 'vendor/autoload.php';...*
    - Include the autoload.php file from the PdfParser application for reading the text from the pdf.
- *$conn = new mysqli($servername, $username, $password, $db);…*
    - This segment of the code establishes the first connection with the MySQL database.
- *if(preg_match('/\s/i', $inputName))…*
    - This code segment checks if the user has entered a full name or just the first name, splits the string accordingly and initializes variables
- *if(count($_POST)<=1){…*
    - This segment of the code is created for test cases. It fetches current max values of the user and resume IDs, so as to compare it with the max IDs after updating the tables.
- *$stmt = $conn->prepare…*
    - This segment of the code uses a prepared statement to insert the form data into the user_master table
- *$parser = new \Smalot\PdfParser\Parser();…*

    - This segment of the code invokes the Parser() function and reads text from the target directory file which was uploaded with the form.
- *$empty = "";…*
    - This segment of the code populates the resume_master table with matched skills from the resume text.
- *$sql = "SELECT MAX(resume_id) AS max FROM resume_master";…*
    - This segment of the code fetches the array of skills in the skill_master table, matches it to the pdf text and updates them in the resume_skills table
- *$sql = "SELECT MAX(user_id) AS user_max FROM user_master";…*
    - This segment of the code maps the user_master and resume_master tables by populating the user_resume table
- *$resume_pass = 0;…*
    - This segment of the code is created for test cases. It, again, fetches the current max values of the user and resume IDs, so as to compare it with the max IDs from earlier and check if all the tables have been updated and mapped correctly.
- *header("Location:index.php");…*
    - Here, the control is sent back to the index.php file
- *if(count($_POST)>1){…*
    - This segment of the code executes whenever this file is called using the website through the form, and not through the test files. It moves the uploaded file from the temp directory to the uploads/ folder and calls the executer function.

2. index.php

- *$paramsFile = file_get_contents("parameters.json");…*
    - This segment of code is responsible for loading entries from the job_master table and putting them into the drop-down in the form
- *<link rel="…*
    - We use Bootstrap CSS for styling purposes
- *<body>…*
    - Here, we create a simple form with the options to add name, email, job type and upload a .pdf file
- * …*
    - Here, we add a link to a useful resource for job searches

3. test.php

- *ob_start();…*
    - Here, the page output buffer is initialized and the results of the test cases are checked.
- *ob_end_clean();…*
    - Here, the page is refreshed and the results of the test cases are output to the webpage.

Functions:

1. sendData.php

- *function executer(string inputName, string inputEmail, int inputJobTypeId, string target_file):*
    - The function executer runs the entire backend code after the submission of the form from index.php. Here, it uses the form data and the uploaded file to update the database tables with the user's information and matched skills.
- *\Smalot\PdfParser\Parser(string target_file);*
    - This function taken from the pdfparser library parses the uploaded .pdf file and returns a string of text extracted from that file. It takes as input the directory of file to parse.

2. test.php

- *ob_start();*
    - This function begins the output buffer for displaying echoed results to the webpage
- *ob_end_clean();*
    - This function clears the output buffer page and anything echoed after is on a new page.

****