RESUME SHORTER

REACTJS

Done By : **SUMAN MANDAVA**

Introduction:

The Resume Shorter program efficiently filters resumes based on two key conditions: over 3 years of experience and proficiency in C++ and Java. It automates the initial screening process, saving time and resources. By analyzing the work experience, the program eliminates candidates with less than 3 years of relevant experience. It also scans for C++ and Java skills, giving priority to those who possess these proficiencies. Recruiters benefit from a streamlined process, as the program identifies qualified candidates objectively and quickly. The Resume Shorter program optimizes the recruitment process, ensuring that only the most suitable applicants proceed to subsequent stages.

Objective:

1. Automate resume screening: The primary objective of the Resume Shorter program is to automate the resume screening process, saving time and effort for recruiters by quickly filtering resumes based on predefined criteria.

2. Efficiently identify qualified candidates: The program aims to efficiently identify candidates with over 3 years of experience and proficiency in C++ and Java, ensuring that only qualified individuals move forward in the recruitment process.

3. Improve recruitment efficiency: By automating the initial screening process, the program improves the overall efficiency of the recruitment process, allowing recruiters to focus on evaluating the most promising candidates.

4. Enhance objectivity and reduce biases: The Resume Shorter program provides an objective assessment of applicant qualifications, reducing biases that can be present in manual resume screening and ensuring a fair evaluation of candidates.

5. Optimize resource allocation: By automating the resume screening process, the program optimizes the allocation of resources, enabling recruiters to allocate more time and effort to other critical aspects of the hiring process, such as interviewing and candidate evaluation.

Program:

ResumShorter.js :

```
  import React from 'react';

import * as Docxtemplater from 'docxtemplater'; import
PizZip from 'pizzip';
import { saveAs } from 'file-saver';

 const extractExperienceDetails = (text) => { const
 details = {
   name: '',
   experience: '',
   skills: [],
 };

 const namePattern = /NAME:\s*([\w\s]+)/i;
 const experiencePattern = /Experience:\s*([\w\s])/i;  const
 skillsPattern = /\b(C\+\+|JAVA)\b/gi;

 const nameMatch = text.match(namePattern);
 const experienceMatch = text.match(experiencePattern);  const
 skillsMatches = text.match(skillsPattern);

 if (nameMatch && nameMatch[1]) {
   details.name = nameMatch[1].trim();
 }

   if (experienceMatch && experienceMatch[1]) { details.experience
   = experienceMatch[1].trim();
 }

 if (skillsMatches) {
   details.skills = skillsMatches.map(skill => skill.trim());
 }

 return details;
};

const App = () => {
   const handleFileUpload = async (event) => { const file
   = event.target.files[0]; const fileReader = new
   FileReader();

   fileReader.onload = async (e) => {
     const arrayBuffer = e.target.result;
```

```
      const zip = new PizZip(arrayBuffer);
      const doc = new Docxtemplater().loadZip(zip);

      const extractedText = doc.getFullText();

      const experiencePattern = /\d+\s*(year|yr|years|yrs)/gi; const matches =
      extractedText.match(experiencePattern);

      if (matches) {
        let totalExperienceYears = 0;
        for (const match of matches) {
          const years = parseInt(match, 10);
          if (!isNaN(years)) {
            totalExperienceYears += years;
          }
        }

        if (totalExperienceYears > 3) {
          const details = extractExperienceDetails(extractedText);

if          (details.skills.includes('C++')           &&
details.skills.includes('JAVA')) {
          const jsonDetails = JSON.stringify(details, null, 2);
          const blob = new Blob([jsonDetails], { type: 'application/json'
});
          saveAs(blob, 'resume_details.json');
          console.log('Total      Experience      Years:',      totalExperienceYears);
          console.log('Details  saved  in  resume_details.json');  return;  // Exit the
          function if the condition is satisfied
        }
      }
    }

console.log('Experience is less than 3 years or required skills are not present.');
    console.log('Full Document:', extractedText); };

    fileReader.readAsArrayBuffer(file);
  };

  return (
    <div>
    <input type="file" onChange={handleFileUpload} /> </div>
  );
};
```

```
export default App;
```

This code is a React component that enables the upload and processing of resumes to extract relevant information such as experience details and skills. Let's break down the presentation of this code:

1. The code imports necessary **libraries**: React, Docxtemplater, PizZip, and file-saver. These libraries are used for handling file uploads, processing DOCX files, and saving extracted information.

2. The `extractExperienceDetails` function extracts the name, experience, and skills from the uploaded resume text using regular expressions. It returns an object containing these details.

3. The `App` component is defined as a functional component. It renders a file input field that triggers the `handleFileUpload` function when a file is selected.

4. The `handleFileUpload` function is responsible for processing the uploaded resume file. It reads the file content as an array buffer, loads it into a `Docxtemplater` instance, and extracts the full text of the document.

5. The function then searches for matches of experience patterns to calculate the total years of experience. If the total experience is greater than 3 years, it calls the `extractExperienceDetails` function to retrieve the name, experience, and skills details.

6. If the extracted details meet the conditions of having both C++ and Java skills, it converts the details object to **JSON format** and saves it as a file named **"resume_details.json"** using the `saveAs` function. It also logs the total experience years and a success message.

7. If the conditions are not met, it logs a message indicating that the experience is less than 3 years or the required skills are not present. It also logs the full text of the document.

8.   Finally, the `App` component is exported as the default export.

In summary, this code provides a React-based interface for uploading resumes, extracting information, and determining if the uploaded resume meets the criteria of having more than 3 years of experience and skills in both C++ and Java. It showcases the usage of various libraries to handle DOCX files and manipulate data, enhancing the efficiency of the resume screening process.

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './ResumeShoter';
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

This code serves as the entry point for a React application that utilizes the Resume Shorter component. Let's provide a brief overview of what this code does:

1.   The code imports the necessary modules: React and ReactDOM. React is the JavaScript library used for building user interfaces, while ReactDOM is responsible for rendering those interfaces in the browser.

2.   The './index.css' import is used to import a CSS file for styling the application. The specific CSS file being imported is not shown in this code snippet.

3.   The './App' import refers to the Resume Shorter component, which is the main component of the application. It handles the file upload and processing logic for extracting information from resumes.

4.   The ReactDOM.render() function is called to render the application. It takes two arguments: the component to render (in this case, the App component), and the DOM element where the component will be rendered (specified by the 'root' element ID).

5.   The <React.StrictMode> component is wrapped around the App component. Itenables additional checks and warnings during development to ensure that the application follows best practices and avoids potential issues.
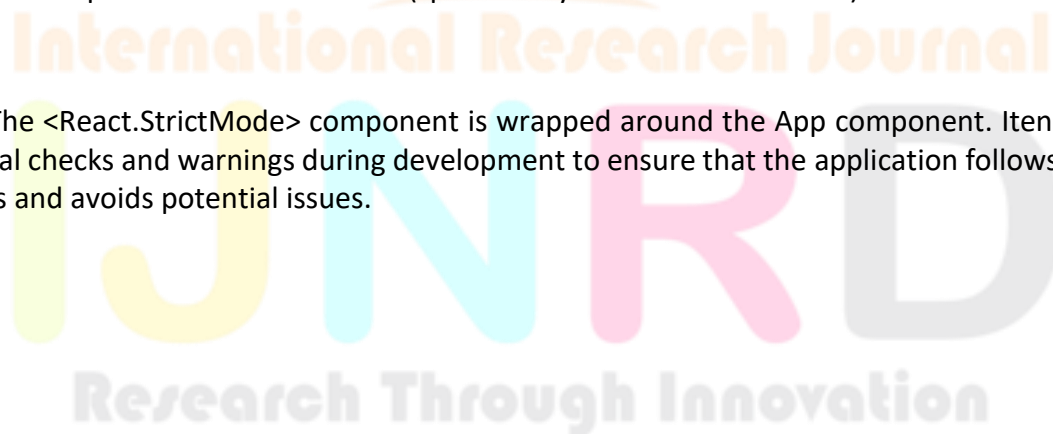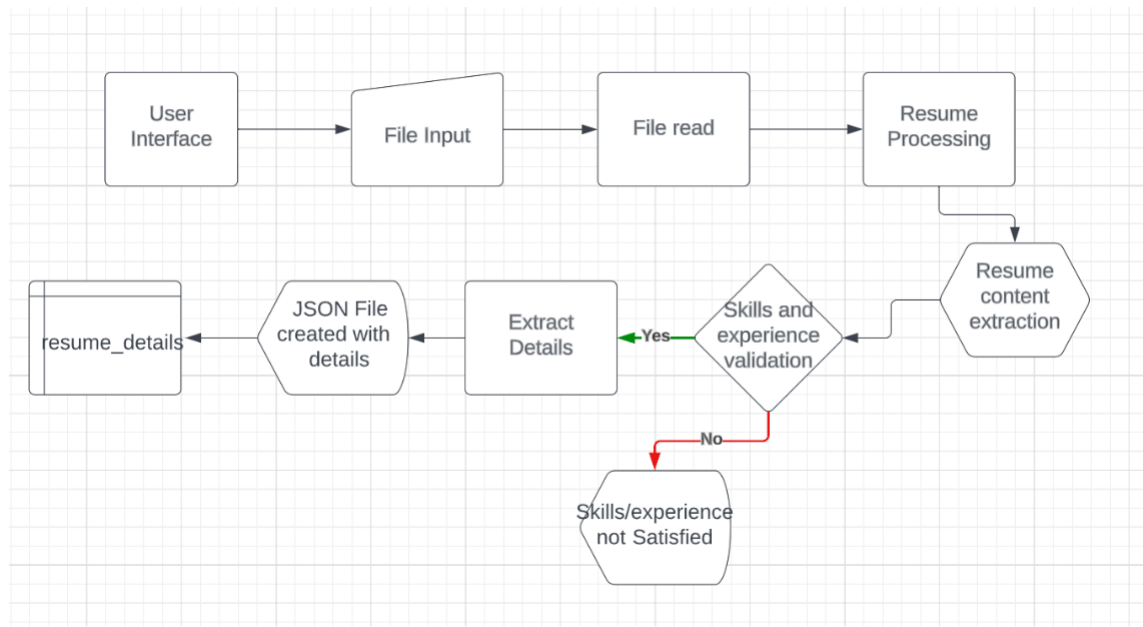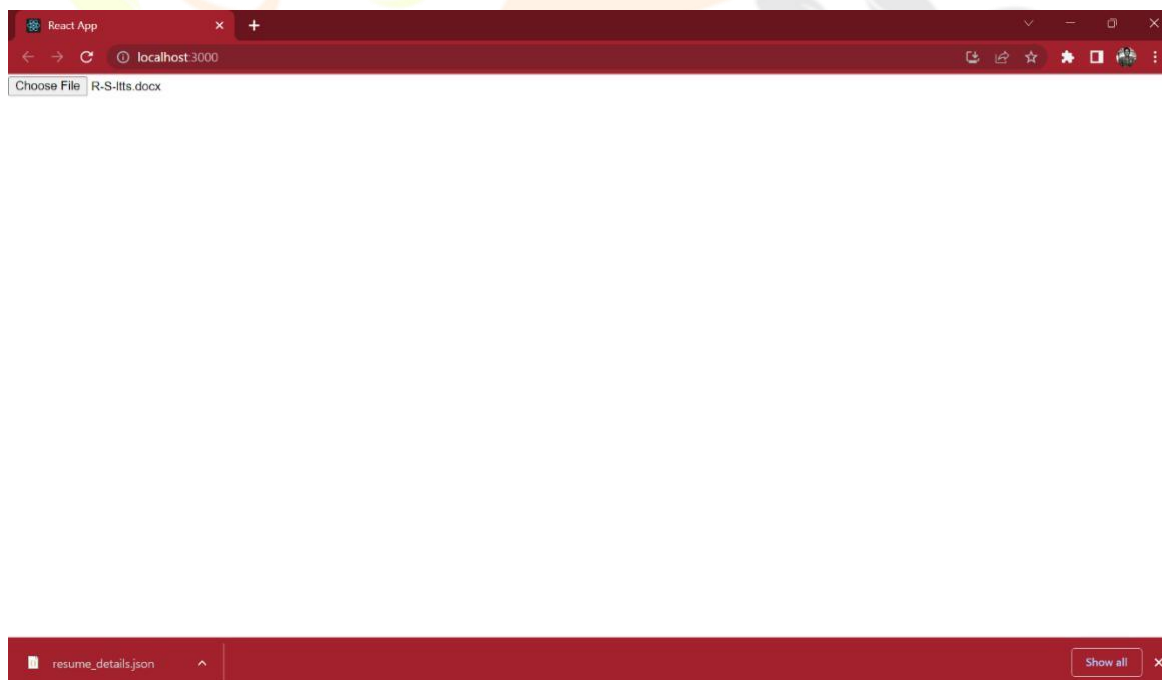
Block Diagram:

Fig-0.0

OUTPUT:

Fig-0.1.

Here, we need to load the Docx File to check the resume and print it in JSON File.
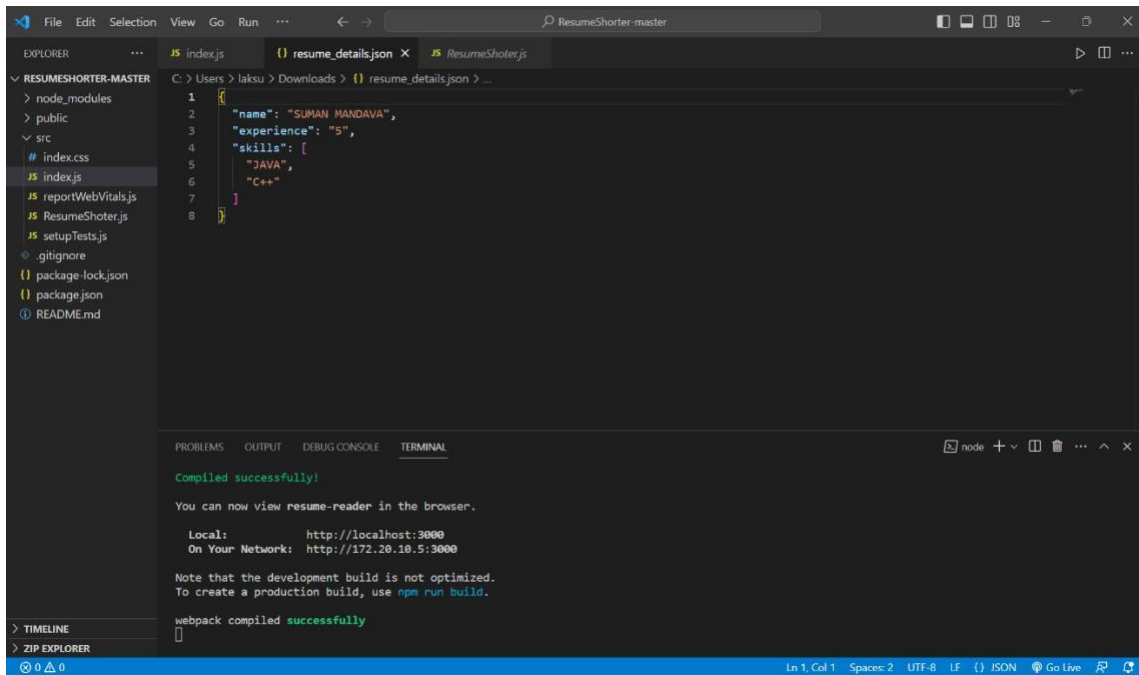
Resume details.json:

Fig-0.2.

Here, We can see the output in JSON file, with necessary details.