RESUME BUILDER

A PROJECT REPORT

Submitted by

Vikas Parmar
(2100290140146)
Vidhi Sharma
(2100290140144)
Shivam Kumar
(2100290140128)
Ayush Sharma
(2100290140047)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the supervision of
Ms. Divya Singhal
(Assistant Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Delhi-NCR
Ghaziabad Uttar Pradesh-201206
December 2022

DECLARATION

I hereby declare that the work presented in this report entitled “Resume Builder", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name :Vikas Parmar
Roll. No.: 2100290140146
Branch: MCA
(Candidate Signature)

CERTIFICATE

Certified that **Vikas Parmar (2100290140146), Vidhi Sharma (2100290140144), Shivam Kumar (2100290140128) , Ayush Sharma (2100290140047)** have carried out the project work having "RESUME BUILDER" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date**:**

VIKAS PARMAR (2100290140146)

VIDHI SHARMA (2100290140144)

SHIVAM KUMAR (2100290140128)

AYUSH SHARMA (2100290140047)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:-

Ms. Divya Singhal
(Assistant Professor)
Department of Computer Applications
KIET Group of Institutions, Ghaziabad.

Signature of Internal Examiner

Signature of External Examiner

Dr. Arun Kr. Tripathi

Head, Department of Computer Applications

KIET Group of Institutions, Ghaziabad

ABSTRACT

Your resume is just a reflection of you on a page. It shows your skills, education, your past work experiences, achievements, courses you have pursued, etc. However, a resume is much more than that. The resume acts as a bridge between you and the prospective recruiter. Hence the importance of a resume can never be underestimated.

Let’s take an example: a candidate is having excellent skills but fails to structure them into a resume then the candidate might end up being rejected in the very first step of a job application. For grabbing a job offer it’s important to know that domain at the same time. For being able to move forward in this process you’ll always need a well-structured resume too.

Almost every time, the 1st round is Resume Shortlisting and many students are not selected here. The reason is simple, not having a well-structured resume. Recruiters don't have too much time to read line by line. That’s why an individual must have a Well-Structured Single Page Resume. When a job seeker wants to apply for a job then generally, he/she needs to attach his/her resume with the application and it is a must now a days.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, **Ms. Divya Singhal** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness**.**

VIKAS PARMAR

VIDHI SHARMA

SHIVAM KUMAR

AYUSH SHARMA

TABLE OF CONTENTS

CHAPTER 1

1.1 Introduction

The Resume Builder Project basically deals with Building a Resume Just by filling the required details and a Resume will be Obtained. Building a resume is a kind of tedious task, many times it has been observed that users (employees etc.) are not interested in building their own resume they consider it as a time taking and monotonous task. To resolve this problem, we are building a full-fledged resume just by filling the details as required by the user and user can download his/her resumein the form of PDF. It is based on React-Js technology which is one of the rising technology in the ITSector and is very Simple and Provides an Optimized Solution

1.2 Project Description

In this project we are designing a resume. It has three modules the first one is the Header module/section which describes the front page of the application. The second one is the Editor module in which all the required details are filled by the user. It consists of fields like achievements, projects etc. and subfields also. The final module is the resume itself. In the editor module all the details will be filled by the user and are saved. User can fill the fields of his/her own choice and thesedetails can be edited also. A save button is available through which the details will be saved and finally all these saved details will be reflected in the final resume module in the form of resume. A general template is provided by this application. Users can download the generated resume in the form of PDF with the help of download button and is saved in the local storage of the system. React-Js provides its library React-to-pdf to download the resume in the form of PDF.

1.3 Hardware and Software Requirements

Hardware Interfaces

- Minimum Hardware requirement
- Processor: P4 3.0 GHz
- RAM: 2 GB or Higher
- Monitor
- Mouse
- Hard disk: 512 GB

Software Interfaces

- Minimum Software requirement
- VS Code
- Chrome

1.4 Modules

There are three main modules of this application one is the Header and another one is the Editor and last one is Resume Template, let us see the responsibility.

1) **Header**
- ❖ It is the Front Interface or page of the Application.

2) **Editor**
- ❖ There are multiple section in this module like Basic info, workExp, project, education, achievement,summary, others.
- ❖ Users can fill the details. Users can add/delete/update the details. Users can save the entered details.

3) **Resume Template**

- ❖ Users can view their Resume.
- ❖ Users can download their resume in the form of PDF.

CHAPTER 2

2.1 Feasibility Study:

This project will be developed on computer, so first check whether the technology is technically available or not. Now a days computer interaction with any job becomes common for any kind of job or work. Andbecause of increasing usage of Computer, Computer is also available with a variety of hardware. Users can fulfill any type of hardware requirement. .
Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.
The main objective of this phase is to identify the current deficiencies in the user's environment and to determine which existing problem are going to be solve in proposed system and also which new functionneeds to be added in proposed system.
An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.
Thus, three tests are carried out on the system namely operation, technical and economical.

2.2 OPERATIONAL FEASIBILITY

Any project is beneficial if and only satisfies the organization's requirement. For any new system setup, it only meets to be communicated and work the other supporting system.
The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system.

2.3 TECHNICAL FEASIBILITY

Technical Feasibility examines whether the technology needed is available and if it is available then itfeasible to carry out all project activities.

The technical needs of a system include:

- The facility to produce outputs in a given time.
- Ability to process large number of transaction at a particular speed.
- Giving response to users under certain conditions.

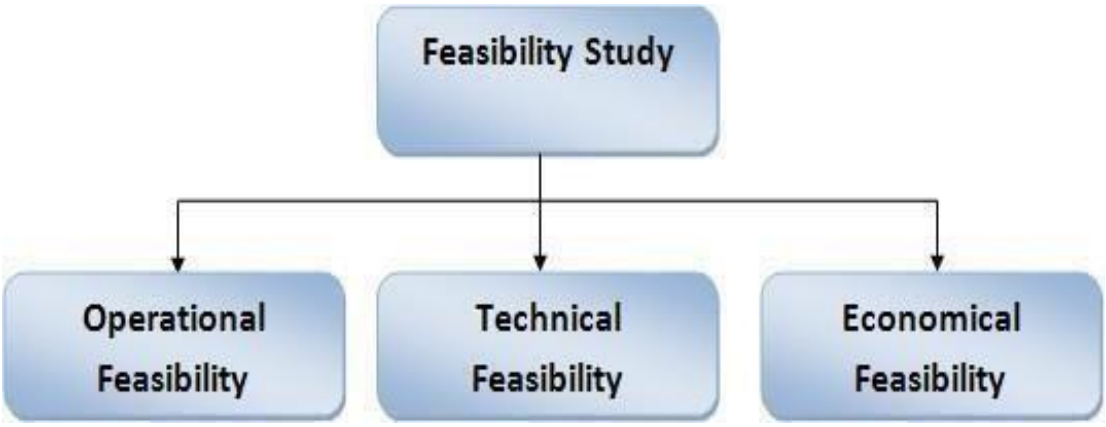
The technology needed for our system is mainly:

- Latest version of browsers.
- Any operating system.

These technologies are available which helps to carry out the system efficiently.

2.4 ECONOMICAL FEASIBLITY

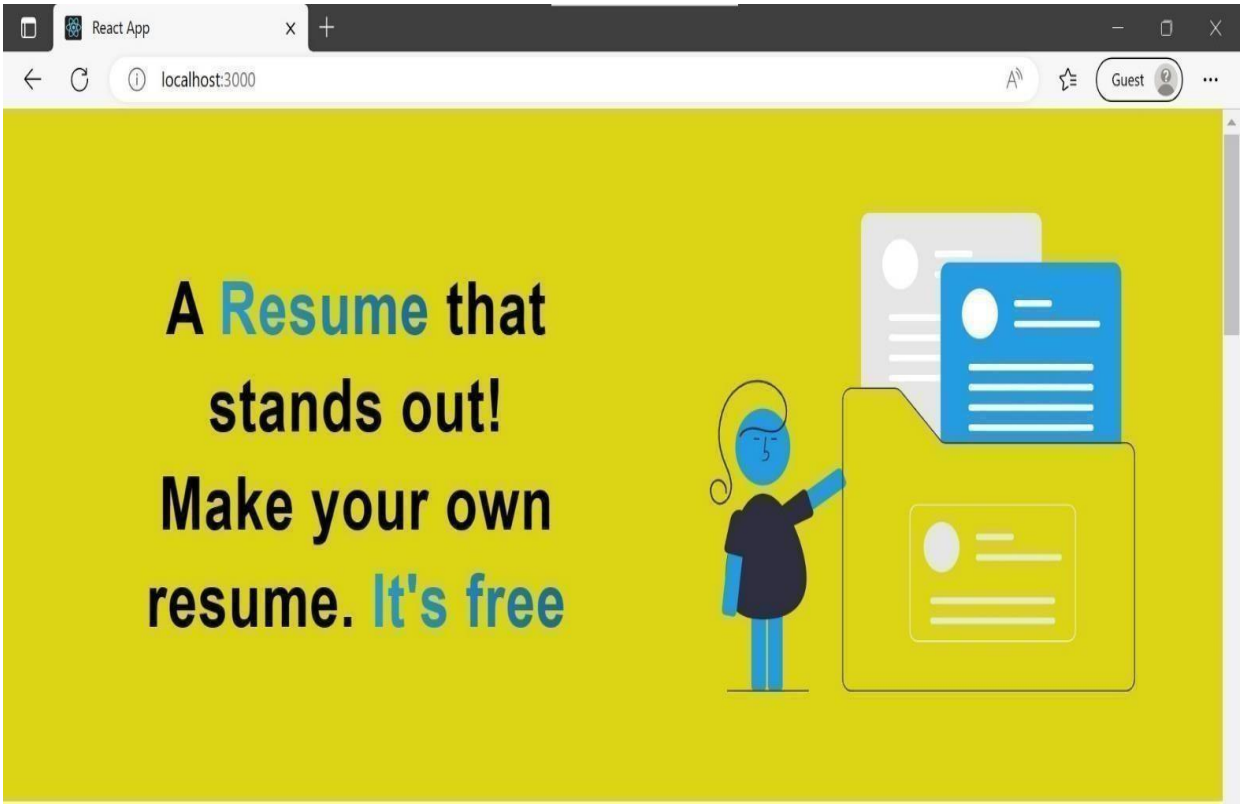
Economical feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction. The cost involves is in designing and developing a good investment for the organization. Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.
In the proposed system the finance is highly required for the installation of the software's which can also be recovered by implementing a better system.



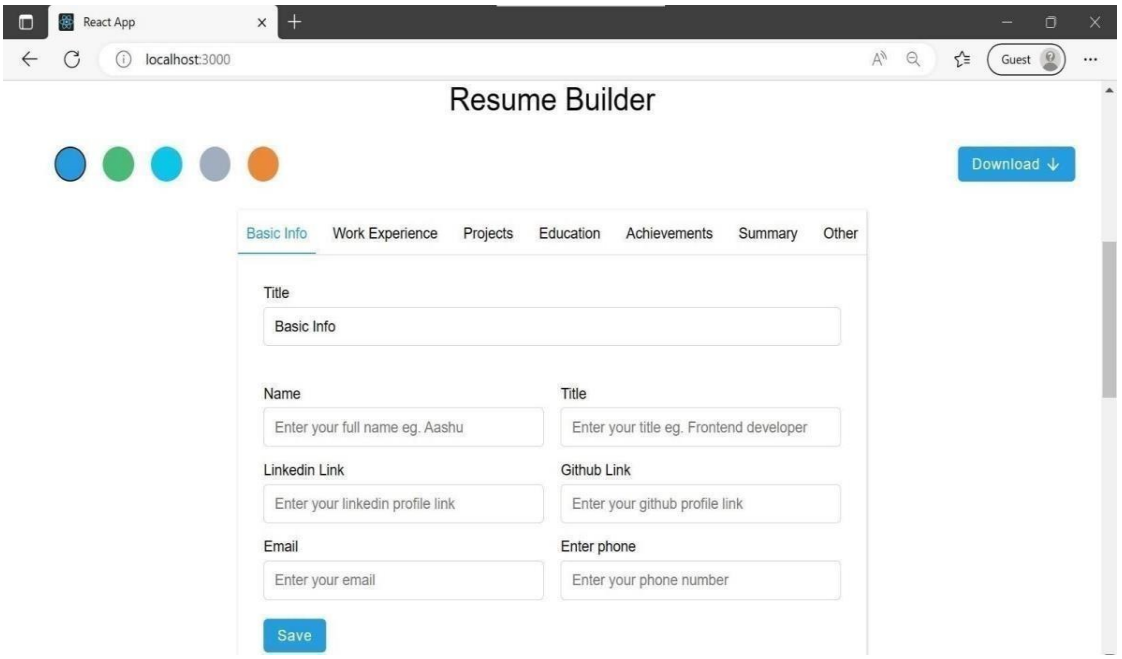
CHAPTER 3

FORM DESIGN

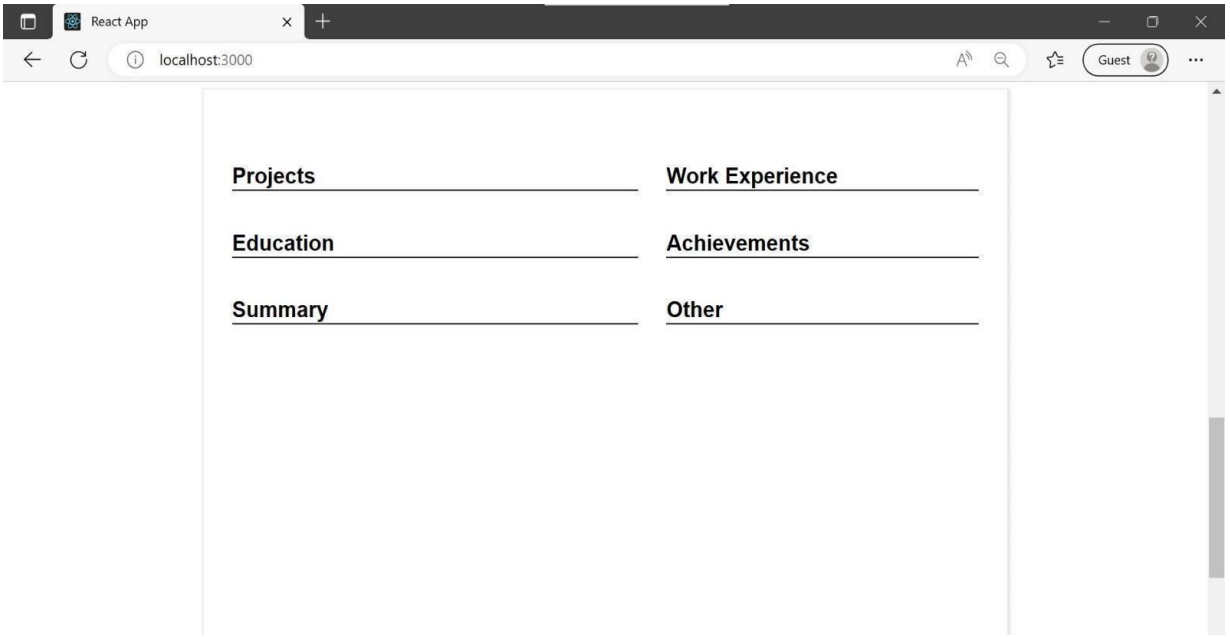
3.1 Front View (Screenshot)

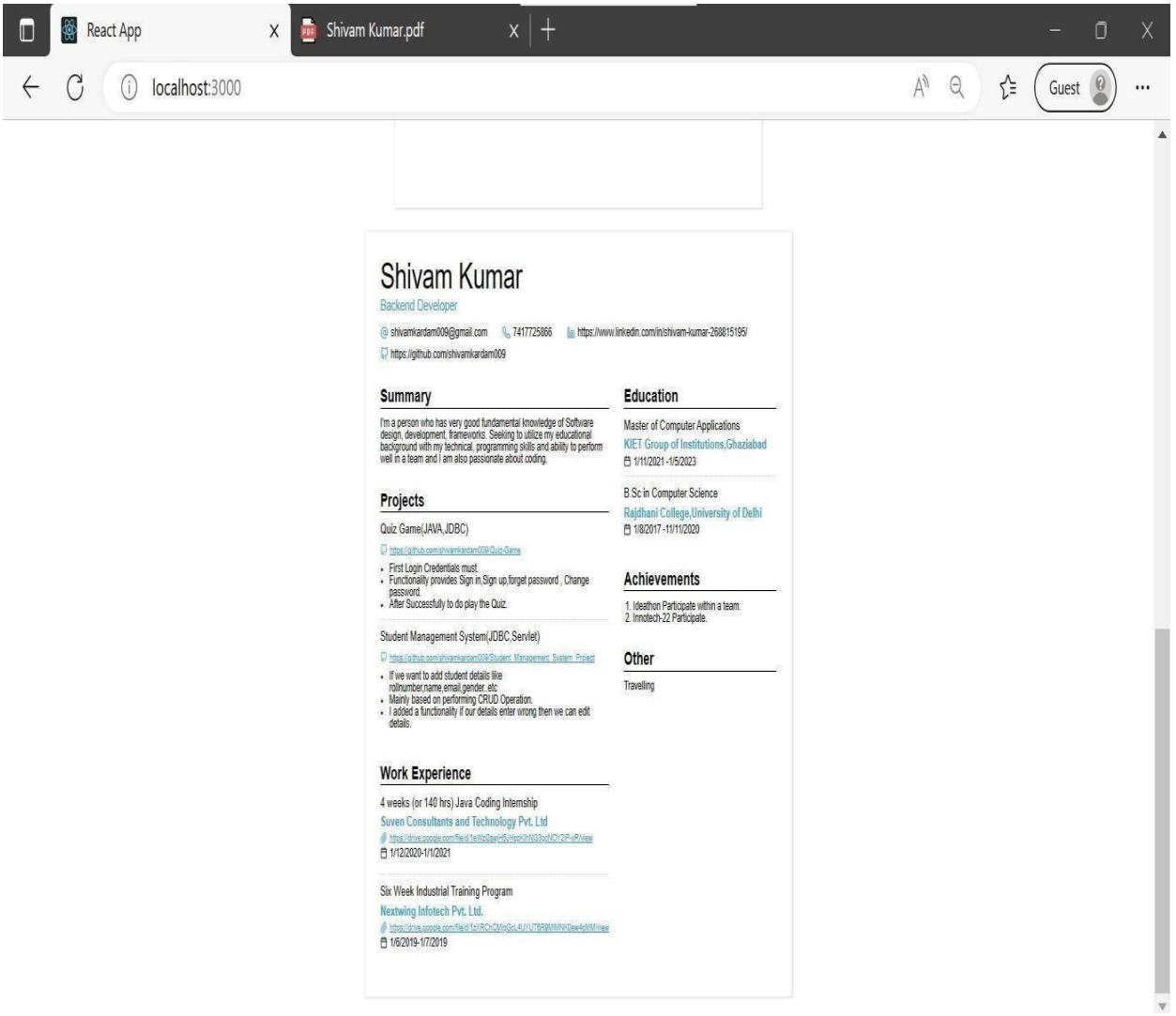


3.2 Input (Screenshot)



3.3 Output (Screenshot)

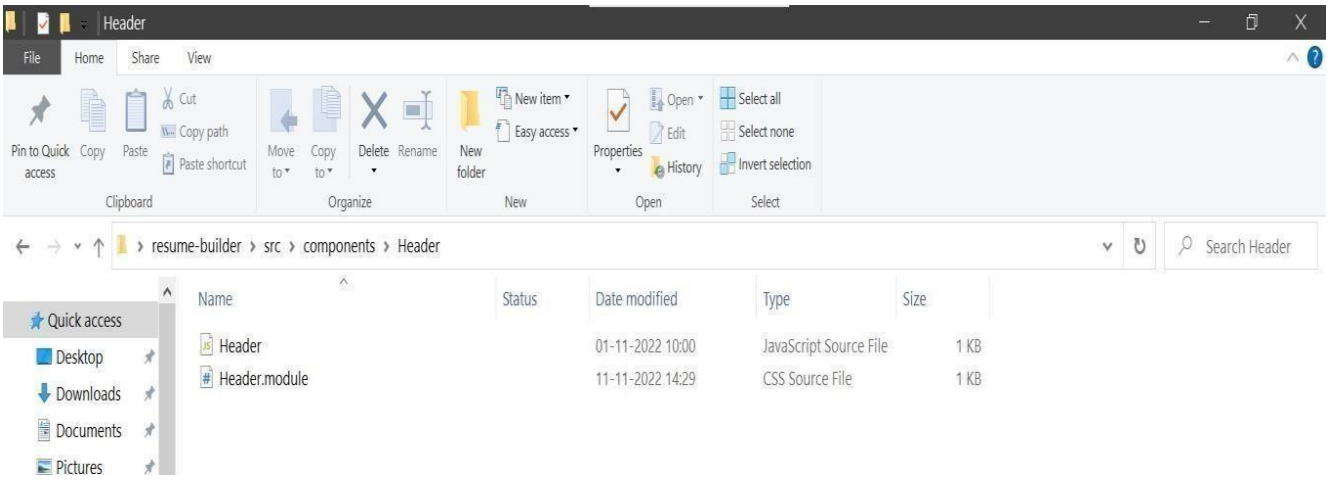




Chapter 4

Coding (Module Wise)

Heading Page



Header.js

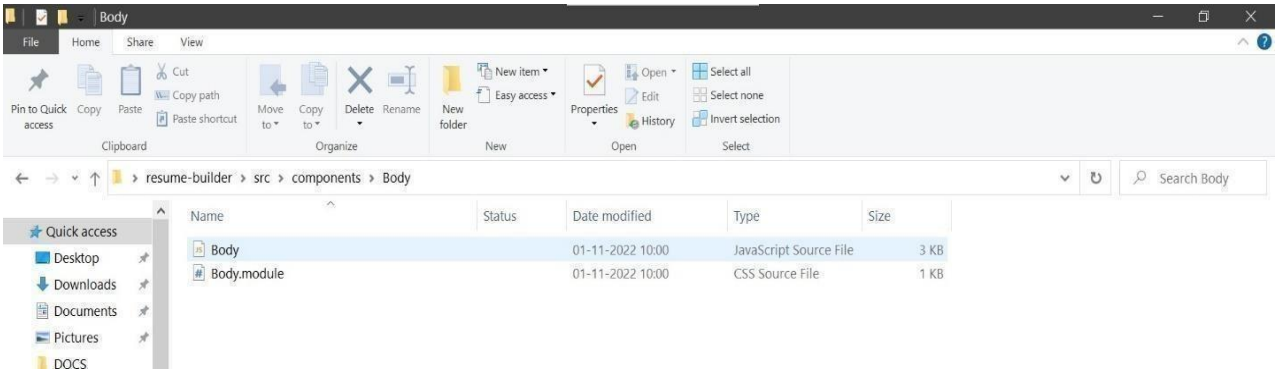
```
import React from "react";
import resumeSvg from "../../assets/resume.svg";
import styles from "./Header.module.css";
function Header() {
  return (
    <div className={styles.container}>
      <div className={styles.left}>
        <p className={styles.heading}>
          A <span>Resume</span> that stands out!
        </p>
        <p className={styles.heading}>
          Make your own resume. <span>It's free</span>
        </p>
      </div>
      <div className={styles.right}>
        <img src={resumeSvg} alt="Resume" />
      </div>
    </div>
  );
}
```

```
export default Header;
```

Header.css

```
.container  {
  padding: 50px30px;
  text-align: center;
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: space-evenly;
  gap: 30px;
  background-color: rgb(219, 213, 22);
  min-height: 85vh;
}

.heading {
  margin: 0 auto;
  max-width: 500px;
  color: #000;
  font-size: 3.1rem;
  line-height: 4.1rem;
  font-weight: bold;
  letter-spacing: 1px;
}

.heading span {
  background-clip: text;
  background-image: linear-gradient(to right, #239ce2, #1369b9);
  color: transparent;
}

.right img {
  width: 400px;
}
```

Body Page



Body.js

```
import React, { useRef, useState } from "react";
import ReactToPrint from "react-to-print";
import { ArrowDown } from "react-feather";

import Editor from "../Editor/Editor";
import Resume from "../Resume/Resume";

import styles from "./Body.module.css";

function Body() {
  const colors = ["#239ce2", "#48bb78", "#0bc5ea", "#a0aec0", "#ed8936"];
  const sections = {
    basicInfo: "Basic Info",
    workExp: "Work Experience",
    project:  "Projects",
    education: "Education",
    achievement: "Achievements",
    summary: "Summary",
    other: "Other",
  };
  const resumeRef = useRef();

  const [activeColor, setActiveColor] = useState(colors[0]);
  const [resumeInformation, setResumeInformation] = useState({
    [sections.basicInfo]: {
      id: sections.basicInfo,
```

```
      sectionTitle: sections.basicInfo,
      detail: {},
    },
    [sections.workExp]: {
      id: sections.workExp,
      sectionTitle: sections.workExp,
      details: [],
    },
    [sections.project]: {
      id: sections.project,
      sectionTitle: sections.project,
      details: [],
    },
    [sections.education]: {
      id: sections.education,
      sectionTitle: sections.education,
      details: [],
    },
    [sections.achievement]: {
      id: sections.achievement,
      sectionTitle: sections.achievement,
      points: [],
    },
    [sections.summary]: {
      id: sections.summary,
      sectionTitle: sections.summary,
      detail: "",
    },
    [sections.other]: {
      id: sections.other,
      sectionTitle: sections.other,
      detail: "",
    },
  });

  return (
    <div className={styles.container}>
      <p className={styles.heading}>Resume Builder</p>
      <div className={styles.toolbar}>
        <div className={styles.colors}>
          {colors.map((item) => (
            <span
              key={item}
              style={{ backgroundColor: item }}
              className={`${styles.color} ${
                activeColor === item ? styles.active : ""
              }`}
              onClick={() => setActiveColor(item)}
```

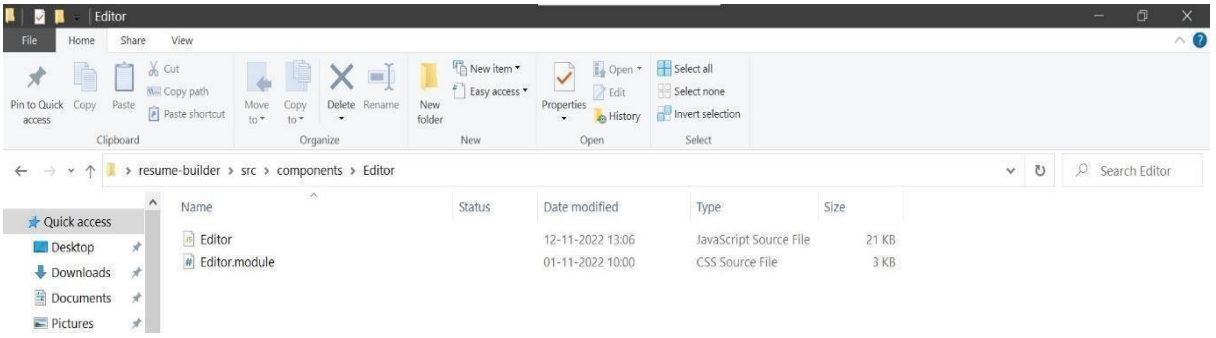
```
            />
          ))}
        </div>
        <ReactToPrint
          trigger={() => {
            return (
              <button>
                Download <ArrowDown />
              </button>
            );
          }}
          content={() => resumeRef.current}
        />
      </div>
      <div className={styles.main}>
        <Editor
          sections={sections}
          information={resumeInformation}
          setInformation={setResumeInformation}
        />
        <Resume
          ref={resumeRef}
          sections={sections}
          information={resumeInformation}
          activeColor={activeColor}
        />
      </div>
    </div>
  );
}
export default Body;
```

Body.css

```
.container {
  padding: 30px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 30px;
  padding-top: 0;
}

.heading {
  font-weight: 500;
  font-size: 2.1rem;
}

.toolbar  {
  width: 100%;
  display: flex;
  gap: 40px;
  justify-content: space-between;
  align-items: center;
}

.colors  {
  display: flex;
  gap: 20px;
  padding: 0 30px;
}

.colors .color {
  height: 36px;
  width: 36px;
  border-radius: 50%;
  background-color: #239ce2;
}
.colors  .active  {
  border: 2px solid #000;
}

.toolbar button {
  padding: 8px 16px;
  border-radius: 5px;
  background-color: #239ce2;
```

```
  color: #fff;
  outline: none;
  border: none;
  font-weight: 500;
  font-size: 1rem;
  letter-spacing: 1px;
  display: flex;
  gap: 5px;
  align-items: center;
  cursor: pointer;
}

.toolbar button svg {
  height: 20px;
  width: 20px;
}

.main {
  display: flex;
  flex-direction: column;
  gap: 30px;
  width: 100%;
}
```

Editor File



Editor.js

```
import React, { useEffect, useState } from "react";
import { X } from "react-feather";

import InputControl from "../InputControl/InputControl";

import styles from "./Editor.module.css";

function Editor(props) {
  const sections = props.sections;
  const information = props.information;

  const [activeSectionKey, setActiveSectionKey] = useState(
    Object.keys(sections)[0]
  );
  const [activeInformation, setActiveInformation] = useState(
    information[sections[Object.keys(sections)[0]]]
  );
  const [activeDetailIndex, setActiveDetailIndex] = useState(0);
  const [sectionTitle, setSectionTitle] = useState(
    sections[Object.keys(sections)[0]]
  );
  const [values, setValues] = useState({
    name: activeInformation?.detail?.name || "",
    title: activeInformation?.detail?.title || "",
    linkedin: activeInformation?.detail?.linkedin ||"",
    github: activeInformation?.detail?.github || "",
    phone: activeInformation?.detail?.phone || "",
```

```
    email: activeInformation?.detail?.email || "",
  });

  const handlePointUpdate = (value, index) => {
    const tempValues = { ...values };
    if (!Array.isArray(tempValues.points)) tempValues.points = [];
    tempValues.points[index] = value;
    setValues(tempValues);
  };

  const workExpBody = (
     <div className={styles.detail}>
       <div className={styles.row}>
         <InputControl
           label="Title"
           placeholder="Enter title eg. Frontend developer"
           value={values.title}
           onChange={(event) =>
             setValues((prev) => ({ ...prev, title: event.target.value }))
           }
         />
         <InputControl
           label="Company Name"
           placeholder="Enter company name eg. amazon"
           value={values.companyName}
           onChange={(event) =>
             setValues((prev) => ({ ...prev, companyName: event.target.value
}))

           }
         />
       </div>
       <div className={styles.row}>
         <InputControl
           label="Certificate Link"
           placeholder="Enter certificate link"
           value={values.certificationLink}
           onChange={(event) =>
             setValues((prev) => ({
               ...prev,
               certificationLink: event.target.value,
             }))
           }
         />
         <InputControl
           label="Location"
           placeholder="Enter location eg. Remote"
           value={values.location}
           onChange={(event) =>
```

```
            setValues((prev) => ({ ...prev, location: event.target.value }))
          }
        />
      </div>
      <div className={styles.row}>
        <InputControl
          label="Start Date"
          type="date"
          placeholder="Enter start date of work"
          value={values.startDate}
          onChange={(event) =>
            setValues((prev) => ({ ...prev, startDate: event.target.value }))
          }
        />
        <InputControl
          label="End Date"
          type="date"
          placeholder="Enter end date of work"
          value={values.endDate}
          onChange={(event) =>
            setValues((prev) => ({ ...prev, endDate: event.target.value }))
          }
        />
      </div>

      <div className={styles.column}>
        <label>Enter work description</label>
        <InputControl
          placeholder="Line 1"
          value={values.points ? values.points[0] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 0)}
        />
        <InputControl
          placeholder="Line 2"
          value={values.points ? values.points[1] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 1)}
        />
        <InputControl
          placeholder="Line 3"
          value={values.points ? values.points[2] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 2)}
        />
      </div>
    </div>
);
const projectBody = (
    <div className={styles.detail}>
      <div className={styles.row}>
```

```
    <InputControl
      label="Title"
      value={values.title}
      placeholder="Enter title eg. Chat app"
      onChange={(event) =>
        setValues((prev) => ({ ...prev, title: event.target.value }))
      }
    />
  </div>
  <InputControl
    label="Overview"
    value={values.overview}
    placeholder="Enter basic overview of project"
    onChange={(event) =>
      setValues((prev) => ({ ...prev, overview: event.target.value }))
    }
  />
  <div className={styles.row}>
    <InputControl
      label="Deployed Link"
      value={values.link}
      placeholder="Enter deployed link of project"
      onChange={(event) =>
        setValues((prev) => ({ ...prev, link: event.target.value }))
      }
    />
    <InputControl
      label="Github Link"
      value={values.github}
      placeholder="Enter github link of project"
      onChange={(event) =>
        setValues((prev) => ({ ...prev, github: event.target.value }))
      }
    />
  </div>
  <div className={styles.column}>
    <label>Enter project description</label>
    <InputControl
      placeholder="Line 1"
      value={values.points ? values.points[0] : ""}
      onChange={(event) => handlePointUpdate(event.target.value, 0)}
    />
    <InputControl
      placeholder="Line 2"
      value={values.points ? values.points[1] : ""}
      onChange={(event) => handlePointUpdate(event.target.value, 1)}
    />
    <InputControl
```

```
            placeholder="Line 3"
            value={values.points ? values.points[2] : ""}
            onChange={(event) => handlePointUpdate(event.target.value, 2)}
          />
          <InputControl
            placeholder="Line 4"
            value={values.points ? values.points[3] : ""}
            onChange={(event) => handlePointUpdate(event.target.value, 3)}
          />
        </div>
      </div>
    );
    const educationBody = (
      <div className={styles.detail}>
        <div className={styles.row}>
          <InputControl
            label="Title"
            value={values.title}
            placeholder="Enter title eg. B-tech"
            onChange={(event) =>
              setValues((prev) => ({ ...prev, title: event.target.value }))
            }
          />
        </div>
        <InputControl
          label="College/School Name"
          value={values.college}
          placeholder="Enter name of your college/school"
          onChange={(event) =>
            setValues((prev) => ({ ...prev, college: event.target.value }))
          }
        />
        <div className={styles.row}>
          <InputControl
            label="Start Date"
            type="date"
            placeholder="Enter start date of this education"
            value={values.startDate}
            onChange={(event) =>
              setValues((prev) => ({ ...prev, startDate: event.target.value }))
            }
          />
          <InputControl
            label="End Date"
            type="date"
            placeholder="Enter end date of this education"
            value={values.endDate}
            onChange={(event) =>
```

```
            setValues((prev) => ({ ...prev, endDate: event.target.value }))
          }
        />
      </div>
    </div>
);
const basicInfoBody = (
  <div className={styles.detail}>
    <div className={styles.row}>
      <InputControl
        label="Name"
        placeholder="Enter your full name eg. Aashu"
        value={values.name}
        onChange={(event) =>
          setValues((prev) => ({ ...prev, name: event.target.value }))
        }
      />
      <InputControl
        label="Title"
        value={values.title}
        placeholder="Enter your title eg. Frontend developer"
        onChange={(event) =>
          setValues((prev) => ({ ...prev, title: event.target.value }))
        }
      />
    </div>
    <div className={styles.row}>
      <InputControl
        label="Linkedin Link"
        value={values.linkedin}
        placeholder="Enter your linkedin profile link"
        onChange={(event) =>
          setValues((prev) => ({ ...prev, linkedin: event.target.value }))
        }
      />
      <InputControl
        label="Github Link"
        value={values.github}
        placeholder="Enter your github profile link"
        onChange={(event) =>
          setValues((prev) => ({ ...prev, github: event.target.value }))
        }
      />
    </div>
    <div className={styles.row}>
      <InputControl
        label="Email"
        value={values.email}
```

```
          placeholder="Enter your email"
          onChange={(event) =>
            setValues((prev) => ({ ...prev, email: event.target.value }))
          }
        />
        <InputControl
          label="Enter phone"
          value={values.phone}
          placeholder="Enter your phone number"
          onChange={(event) =>
            setValues((prev) => ({ ...prev, phone: event.target.value }))
          }
        />
      </div>
    </div>
  );
  const achievementsBody = (
    <div className={styles.detail}>
      <div className={styles.column}>
        <label>List your achievements</label>
        <InputControl
          placeholder="Line 1"
          value={values.points ? values.points[0] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 0)}
        />
        <InputControl
          placeholder="Line 2"
          value={values.points ? values.points[1] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 1)}
        />
        <InputControl
          placeholder="Line 3"
          value={values.points ? values.points[2] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 2)}
        />
        <InputControl
          placeholder="Line 4"
          value={values.points ? values.points[3] : ""}
          onChange={(event) => handlePointUpdate(event.target.value, 3)}
        />
      </div>
    </div>
  );
  const summaryBody = (
    <div className={styles.detail}>
      <InputControl
        label="Summary"
        value={values.summary}
```

```
        placeholder="Enter your objective/summary"
        onChange={(event) =>
          setValues((prev) => ({ ...prev, summary: event.target.value }))
        }
      />
    </div>
);
const otherBody = (
    <div className={styles.detail}>
      <InputControl
        label="Other"
        value={values.other}
        placeholder="Enter something"
        onChange={(event) =>
          setValues((prev) => ({ ...prev, other: event.target.value }))
        }
      />
    </div>
);

const generateBody = () =>{
    switch (sections[activeSectionKey]) {
      case sections.basicInfo:
        return basicInfoBody;
      case sections.workExp:
        return workExpBody;
      case sections.project:
        return projectBody;
      case sections.education:
        return educationBody;
      case sections.achievement:
        return achievementsBody;
      case sections.summary:
        return summaryBody;
      case sections.other:
        return otherBody;
      default:
        return null;
    }
};

const handleSubmission = () => {
    switch (sections[activeSectionKey]) {
      case sections.basicInfo: {
        const tempDetail = {
          name: values.name,
          title: values.title,
          linkedin: values.linkedin,
```

```
      github: values.github,
      email: values.email,
      phone: values.phone,
    };

    props.setInformation((prev) => ({
      ...prev,
      [sections.basicInfo]: {
        ...prev[sections.basicInfo],
        detail: tempDetail,
        sectionTitle,
      },
    }));
    break;
  }
  case sections.workExp: {
    const tempDetail = {
      certificationLink: values.certificationLink,
      title: values.title,
      startDate: values.startDate,
      endDate: values.endDate,
      companyName: values.companyName,
      location: values.location,
      points: values.points,
    };
    const tempDetails = [...information[sections.workExp]?.details];
    tempDetails[activeDetailIndex] = tempDetail;

    props.setInformation((prev) => ({
      ...prev,
      [sections.workExp]: {
        ...prev[sections.workExp],
        details: tempDetails,
        sectionTitle,
      },
    }));
    break;
  }
  case sections.project: {
    const tempDetail = {
      link: values.link,
      title: values.title,
      overview: values.overview,
      github: values.github,
      points: values.points,
    };
    const tempDetails = [...information[sections.project]?.details];
    tempDetails[activeDetailIndex] = tempDetail;
```

```
    props.setInformation((prev) => ({
      ...prev,
      [sections.project]: {
        ...prev[sections.project],
        details: tempDetails,
        sectionTitle,
      },
    }));
    break;
  }
  case sections.education: {
    const tempDetail = {
      title:      values.title,
      college: values.college,
      startDate: values.startDate,
      endDate: values.endDate,
    };
    const tempDetails = [...information[sections.education]?.details];
    tempDetails[activeDetailIndex] = tempDetail;

    props.setInformation((prev) => ({
      ...prev,
      [sections.education]: {
        ...prev[sections.education],
        details: tempDetails,
        sectionTitle,
      },
    }));
    break;
  }
  case sections.achievement: {
    const tempPoints = values.points;

    props.setInformation((prev) => ({
      ...prev,
      [sections.achievement]: {
        ...prev[sections.achievement],
        points: tempPoints,
        sectionTitle,
      },
    }));
    break;
  }
  case sections.summary: {
    const tempDetail = values.summary;

    props.setInformation((prev) => ({
```

```
          ...prev,
          [sections.summary]: {
            ...prev[sections.summary],
            detail: tempDetail,
            sectionTitle,
          },
        }));
        break;
      }
      case sections.other: {
        const tempDetail = values.other;

        props.setInformation((prev) => ({
          ...prev,
          [sections.other]: {
            ...prev[sections.other],
            detail: tempDetail,
            sectionTitle,
          },
        }));
        break;
      }
    }
  };

  const handleAddNew = () => {
    const details = activeInformation?.details;
    if (!details) return;
    const lastDetail = details.slice(-1)[0];
    if (!Object.keys(lastDetail).length) return;
    details?.push({});

    props.setInformation((prev) => ({
      ...prev,
      [sections[activeSectionKey]]: {
        ...information[sections[activeSectionKey]],
        details: details,
      },
    }));
    setActiveDetailIndex(details?.length - 1);
  };

  const handleDeleteDetail = (index) => {
    const details = activeInformation?.details
      ? [...activeInformation?.details]
      : "";
    if (!details) return;
    details.splice(index, 1);
```

```
      props.setInformation((prev) => ({
        ...prev,
        [sections[activeSectionKey]]: {
          ...information[sections[activeSectionKey]],
          details: details,
        },
      }));

      setActiveDetailIndex((prev) => (prev === index ? 0 : prev - 1));
    };

    useEffect(() => {
      const activeInfo = information[sections[activeSectionKey]];
      setActiveInformation(activeInfo);
      setSectionTitle(sections[activeSectionKey]);
      setActiveDetailIndex(0);
      setValues({
        name: activeInfo?.detail?.name || "",
        overview: activeInfo?.details
          ? activeInfo.details[0]?.overview || ""
          : "",
        link: activeInfo?.details ? activeInfo.details[0]?.link || "" : "",
        certificationLink: activeInfo?.details
          ? activeInfo.details[0]?.certificationLink || ""
          : "",
        companyName: activeInfo?.details
          ? activeInfo.details[0]?.companyName || ""
          : "",
        college: activeInfo?.details
          ? activeInfo.details[0]?.college || ""
          : "",
        location: activeInfo?.details
          ? activeInfo.details[0]?.location || ""
          : "",
        startDate: activeInfo?.details
          ? activeInfo.details[0]?.startDate || ""
          : "",
        endDate: activeInfo?.details ? activeInfo.details[0]?.endDate || "" :
"",
        points: activeInfo?.details
          ? activeInfo.details[0]?.points
            ? [...activeInfo.details[0]?.points]
            : ""
          : activeInfo?.points
          ? [...activeInfo.points]
          : "",
        title: activeInfo?.details
          ? activeInfo.details[0]?.title || ""
```

```
        : activeInfo?.detail?.title || "",
      linkedin: activeInfo?.detail?.linkedin || "",
      github: activeInfo?.details
        ? activeInfo.details[0]?.github || ""
        : activeInfo?.detail?.github || "",
      phone: activeInfo?.detail?.phone || "",
      email: activeInfo?.detail?.email || "",
      summary: typeof activeInfo?.detail !== "object" ? activeInfo.detail :
"",
      other: typeof activeInfo?.detail !== "object" ? activeInfo.detail : "",
    });
  }, [activeSectionKey]);

  useEffect(() => {
    setActiveInformation(information[sections[activeSectionKey]]);
  }, [information]);

  useEffect(() => {
    const details = activeInformation?.details;
    if (!details) return;

    const activeInfo = information[sections[activeSectionKey]];
    setValues({
      overview: activeInfo.details[activeDetailIndex]?.overview || "",
      link: activeInfo.details[activeDetailIndex]?.link || "",
      certificationLink:
        activeInfo.details[activeDetailIndex]?.certificationLink || "",
      companyName: activeInfo.details[activeDetailIndex]?.companyName || "",
      location: activeInfo.details[activeDetailIndex]?.location || "",
      startDate: activeInfo.details[activeDetailIndex]?.startDate || "",
      endDate: activeInfo.details[activeDetailIndex]?.endDate || "",
      points: activeInfo.details[activeDetailIndex]?.points || "",
      title: activeInfo.details[activeDetailIndex]?.title || "",
      linkedin: activeInfo.details[activeDetailIndex]?.linkedin || "",
      github: activeInfo.details[activeDetailIndex]?.github || "",
      college: activeInfo.details[activeDetailIndex]?.college || "",
    });
  }, [activeDetailIndex]);

  return (
    <div className={styles.container}>
      <div className={styles.header}>
        {Object.keys(sections)?.map((key) => (
          <div
            className={`${styles.section} ${
              activeSectionKey === key ? styles.active : ""
            }`}
            key={key}
```

```
        onClick={() => setActiveSectionKey(key)}
      >
        {sections[key]}
      </div>
    ))}
</div>

<div className={styles.body}>
  <InputControl
    label="Title"
    placeholder="Enter section title"
    value={sectionTitle}
    onChange={(event) => setSectionTitle(event.target.value)}
  />

  <div className={styles.chips}>
    {activeInformation?.details
      ? activeInformation?.details?.map((item, index) => (
          <div
            className={`${styles.chip} ${
              activeDetailIndex === index ? styles.active : ""
            }`}
            key={item.title + index}
            onClick={() => setActiveDetailIndex(index)}
          >
            <p>
              {sections[activeSectionKey]} {index + 1}
            </p>
            <X
              onClick={(event) => {
                event.stopPropagation();
                handleDeleteDetail(index);
              }}
            />
          </div>
        ))
      : ""}
    {activeInformation?.details &&
    activeInformation?.details?.length > 0 ? (
      <div className={styles.new} onClick={handleAddNew}>
        +New
      </div>
    ) : (
      ""
    )}
  </div>
  {generateBody()}
  <button onClick={handleSubmission}>Save</button>
```

```
        </div>
      </div>
    );}
export default Editor;
```

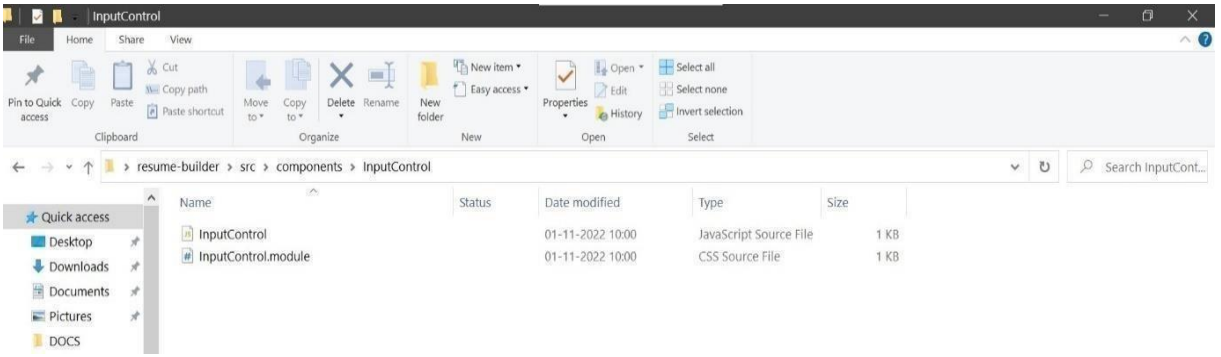
Editor.css

```
.container {
  min-width: 550px;
  min-height: 450px;
  max-width: 750px;
  display: flex;
  flex-direction: column;
  gap: 30px;
  box-shadow: 1px 1px 4px rgba(0, 0, 0, 0.2);
  padding-top: 5px;
  height: fit-content;
  margin: 0 auto;
}

.header {
  display: flex;
  gap: 10px;
  overflow-x: auto;
  border-bottom: 1px solid rgba(0, 0, 0, 0.1);
}

.header::-webkit-scrollbar {
  height: 7px;
}

.header::-webkit-scrollbar-thumb {
  border-radius: 20px;
  background-color: #c7c7c7;
}

.header .section {
  padding: 10px;
  border-bottom: 2px solid transparent;
  font-weight: 500;
  font-size: 1rem;
  white-space: nowrap;
  cursor: pointer;
}

.header .active {
  border-bottom: 2px solid #239ce2;
  color: #239ce2;
```

```
}

.body  {
  padding: 30px;
  display: flex;
  flex-direction: column;
  gap: 20px;
  padding-top: 0;
}

.detail {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

.detail .row {
  display: flex;
  gap: 20px;
}
.detail.column {
  display: flex;
  flex-direction: column;
  gap: 10px;
}
.detail .row > div {
  flex: 1;
}

.chips {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  gap: 20px;
}

.chips  .new {
  color: #239ce2;
  font-weight: bold;
  letter-spacing: 1px;
  cursor: pointer;
}

.chip  {
  display: flex;
  gap: 5px;
  align-items: center;
  padding: 4px 8px;
```

```
  background-color: #808080;
  border-radius: 20px;
  cursor: default;
}

.chip p {
  font-weight: 500;
  line-height: 1.25rem;
  color: #fff;
}
.chip svg  {
  color: #fff;
  height: 18px;
  width: 18px;
  cursor: pointer;
}

.chips .active {
  background-color: #239ce2;
}

.body button  {
  width: fit-content;
  padding: 8px 16px;
  border-radius: 5px;
  background-color: #239ce2;
  color: #fff;
  outline: none;
  border: none;
  font-weight: 500;
  font-size: 1rem;
  letter-spacing: 1px;
  display: flex;
  gap: 5px;
  align-items: center;
  cursor: pointer;
  transition: 200ms;
}

.body button:active {
  transform: translateY(2px);
}
```

InputControl File



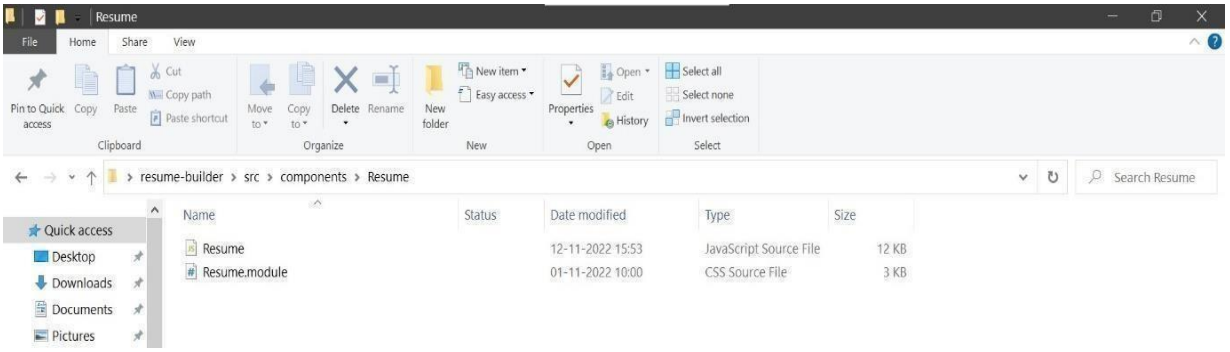
InputControl.js

```
import React from "react";

import styles from "./InputControl.module.css";

function InputControl({ label, ...props }) {
  return (
    <div className={styles.container}>
      {label && <label>{label}</label>}
      <input type="text" {...props} />
    </div>
  );
}

export default InputControl;
```

InputControl.css

```css
.container {
  display: flex;
  flex-direction: column;
  gap: 5px;
}

label {
  font-weight: 500;
  font-size: 1rem;
}

input {
  border: 1px solid #cccccc;
  outline: none;
  padding: 10px12px;
  font-size: 1rem;
  border-radius: 5px;
}

input:hover {
  border: 1px solid #adadad;
}

input:focus {
  border: 1px solid #239ce2;
}
```

Resume File



Resume.js

```
import React, { forwardRef, useEffect, useRef, useState } from "react";
import {
  AtSign,
  Calendar,
  GitHub,
  Linkedin,
  MapPin,
  Paperclip,
  Phone,
} from "react-feather";

import styles from "./Resume.module.css";

const Resume = forwardRef((props, ref) => {
  const information = props.information;
  const sections = props.sections;
  const containerRef = useRef();

  const [columns, setColumns] = useState([[], []]);
  const [source, setSource] = useState("");
  const [target, seTarget] = useState("");

  const info = {
    workExp: information[sections.workExp],
```

```
  project: information[sections.project],
  achievement: information[sections.achievement],
  education: information[sections.education],
  basicInfo: information[sections.basicInfo],
  summary: information[sections.summary],
  other: information[sections.other],
};

const getFormattedDate = (value) => {
  if (!value) return "";
  const date = new Date(value);

  return `${date.getDate()}/${date.getMonth() + 1}/${date.getFullYear()}`;
};

const sectionDiv = {
  [sections.workExp]: (
    <div
      key={"workexp"}
      draggable
      onDragOver={() => seTarget(info.workExp?.id)}
      onDragEnd={() => setSource(info.workExp?.id)}
      className={`${styles.section} ${
        info.workExp?.sectionTitle ? "" : styles.hidden
      }`}
    >
      <div className={styles.sectionTitle}>{info.workExp.sectionTitle}</div>
      <div className={styles.content}>
        {info.workExp?.details?.map((item) => (
          <div className={styles.item} key={item.title}>
            {item.title ? (
              <p className={styles.title}>{item.title}</p>
            ) : (
              <span />
            )}
            {item.companyName ? (
              <p   className={styles.subTitle}>{item.companyName}</p>
            ) : (
              <span />
            )}
            {item.certificationLink ? (
              <a className={styles.link} href={item.certificationLink}>
                <Paperclip />
                {item.certificationLink}
              </a>
            ) : (
              <span />
            )}
```

```
            {item.startDate && item.endDate ? (
              <div className={styles.date}>
                <Calendar /> {getFormattedDate(item.startDate)}-
                {getFormattedDate(item.endDate)}
              </div>
            ) : (
              <div />
            )}
            {item.location ? (
              <p className={styles.date}>
                <MapPin /> Remote
              </p>
            ) : (
              <span />
            )}
            {item.points?.length > 0 ? (
              <ul className={styles.points}>
                {item.points?.map((elem, index) => (
                  <li className={styles.point} key={elem + index}>
                    {elem}
                  </li>
                ))}
              </ul>
            ) : (
              <span />
            )}
          </div>
        ))}
      </div>
    </div>
  ),
  [sections.project]: (
    <div
      key={"project"}
      draggable
      onDragOver={() => seTarget(info.project?.id)}
      onDragEnd={() => setSource(info.project?.id)}
      className={`${styles.section} ${
        info.project?.sectionTitle ? "" : styles.hidden
      }`}
    >
      <div className={styles.sectionTitle}>{info.project.sectionTitle}</div>
      <div className={styles.content}>
        {info.project?.details?.map((item) => (
          <div className={styles.item}>
            {item.title ? (
              <p className={styles.title}>{item.title}</p>
            ) : (
```

```
            <span />
          )}
          {item.link ? (
            <a className={styles.link} href={item.link}>
              <Paperclip />
              {item.link}
            </a>
          ) : (
            <span />
          )}
          {item.github ? (
            <a className={styles.link} href={item.github}>
              <GitHub />
              {item.github}
            </a>
          ) : (
            <span />
          )}
          {item.overview ? (
            <p  className={styles.overview}>{item.overview}  </p>
          ) : (
            <span />
          )}
          {item.points?.length > 0 ? (
            <ul className={styles.points}>
              {item.points?.map((elem, index) => (
                <li  className={styles.point} key={elem + index}>
                  {elem}
                </li>
              ))}
            </ul>
          ) : (
            <span />
          )}
        </div>
      ))}
    </div>
  </div>
),
[sections.education]: (
  <div
    key={"education"}
    draggable
    onDragOver={()  =>  seTarget(info.education?.id)}
    onDragEnd={()  =>  setSource(info.education?.id)}
    className={`${styles.section}  ${
      info.education?.sectionTitle  ?  ""  :  styles.hidden
    }`}
```

```
  >
    <div className={styles.sectionTitle}>
      {info.education?.sectionTitle}
    </div>
    <div className={styles.content}>
      {info.education?.details?.map((item) => (
        <div className={styles.item}>
          {item.title ? (
            <p className={styles.title}>{item.title}</p>
          ) : (
            <span />
          )}
          {item.college ? (
            <p  className={styles.subTitle}>{item.college}</p>
          ) : (
            <span />
          )}
          {item.startDate && item.endDate ? (
            <div className={styles.date}>
              <Calendar /> {getFormattedDate(item.startDate)} -
              {getFormattedDate(item.endDate)}
            </div>
          ) : (
            ""
          )}
        </div>
      ))}
    </div>
  </div>
),
[sections.achievement]: (
  <div
    key={"achievement"}
    draggable
    onDragOver={() => seTarget(info.achievement?.id)}
    onDragEnd={() => setSource(info.achievement?.id)}
    className={`${styles.section} ${
      info.achievement?.sectionTitle ? "" : styles.hidden
    }`}
  >
    <div className={styles.sectionTitle}>
      {info.achievement?.sectionTitle}
    </div>
    <div className={styles.content}>
      {info.achievement?.points?.length > 0 ? (
        <ul  className={styles.numbered}>
          {info.achievement?.points?.map((elem, index) => (
            <li className={styles.point} key={elem + index}>
```

```
                    {elem}
                  </li>
                ))}
              </ul>
            ) : (
              <span />
            )}
          </div>
        </div>
      ),
      [sections.summary]: (
        <div
          key={"summary"}
          draggable
          onDragOver={() => seTarget(info.summary?.id)}
          onDragEnd={() => setSource(info.summary?.id)}
          className={`${styles.section} ${
            info.summary?.sectionTitle ? "" : styles.hidden
          }`}
        >
          <div
className={styles.sectionTitle}>{info.summary?.sectionTitle}</div>
          <div className={styles.content}>
            <p className={styles.overview}>{info.summary?.detail}</p>
          </div>
        </div>
      ),
      [sections.other]: (
        <div
          key={"other"}
          draggable
          onDragOver={() => seTarget(info.other?.id)}
          onDragEnd={() => setSource(info.other?.id)}
          className={`${styles.section} ${
            info.other?.sectionTitle ? "" : styles.hidden
          }`}
        >
          <div className={styles.sectionTitle}>{info.other?.sectionTitle}</div>
          <div className={styles.content}>
            <p className={styles.overview}>{info?.other?.detail}</p>
          </div>
        </div>
      ),
    };

    const swapSourceTarget = (source, target) => {
      if (!source || !target) return;
      const tempColumns = [[...columns[0]], [...columns[1]]];
```

```
  let sourceRowIndex = tempColumns[0].findIndex((item) => item === source);
  let sourceColumnIndex = 0;
  if (sourceRowIndex < 0) {
    sourceColumnIndex = 1;
    sourceRowIndex = tempColumns[1].findIndex((item) => item === source);
  }

  let targetRowIndex = tempColumns[0].findIndex((item) => item === target);
  let targetColumnIndex = 0;
  if (targetRowIndex < 0) {
    targetColumnIndex = 1;
    targetRowIndex = tempColumns[1].findIndex((item) => item === target);
  }

  const tempSource = tempColumns[sourceColumnIndex][sourceRowIndex];
  tempColumns[sourceColumnIndex][sourceRowIndex] =
    tempColumns[targetColumnIndex][targetRowIndex];

  tempColumns[targetColumnIndex][targetRowIndex] = tempSource;

  setColumns(tempColumns);
};

useEffect(() => {
  setColumns([
    [sections.project, sections.education, sections.summary],
    [sections.workExp, sections.achievement, sections.other],
  ]);
}, []);

useEffect(() => {
  swapSourceTarget(source, target);
}, [source]);

useEffect(() => {
  const container = containerRef.current;
  if (!props.activeColor || !container) return;

  container.style.setProperty("--color", props.activeColor);
}, [props.activeColor]);

return (
  <div ref={ref}>
    <div ref={containerRef} className={styles.container}>
      <div className={styles.header}>
        <p   className={styles.heading}>{info.basicInfo?.detail?.name}</p>
        <p   className={styles.subHeading}>{info.basicInfo?.detail?.title}</p>
```

```
            <div className={styles.links}>
              {info.basicInfo?.detail?.email ? (
                <a className={styles.link} type="email">
                  <AtSign /> {info.basicInfo?.detail?.email}
                </a>
              ) : (
                <span />
              )}
              {info.basicInfo?.detail?.phone ? (
                <a className={styles.link}>
                  <Phone /> {info.basicInfo?.detail?.phone}
                </a>
              ) : (
                <span />
              )}
              {info.basicInfo?.detail?.linkedin ? (
                <a className={styles.link}>
                  <Linkedin /> {info.basicInfo?.detail?.linkedin}
                </a>
              ) : (
                <span />
              )}
              {info.basicInfo?.detail?.github ? (
                <a className={styles.link}>
                  <GitHub /> {info.basicInfo?.detail?.github}
                </a>
              ) : (
                <span />
              )}
            </div>
          </div>

          <div className={styles.main}>
            <div className={styles.col1}>
              {columns[0].map((item) => sectionDiv[item])}
            </div>
            <div className={styles.col2}>
              {columns[1].map((item) => sectionDiv[item])}
            </div>
          </div>
        </div>
      </div>
    );
});

export default Resume;
```

Resume.css

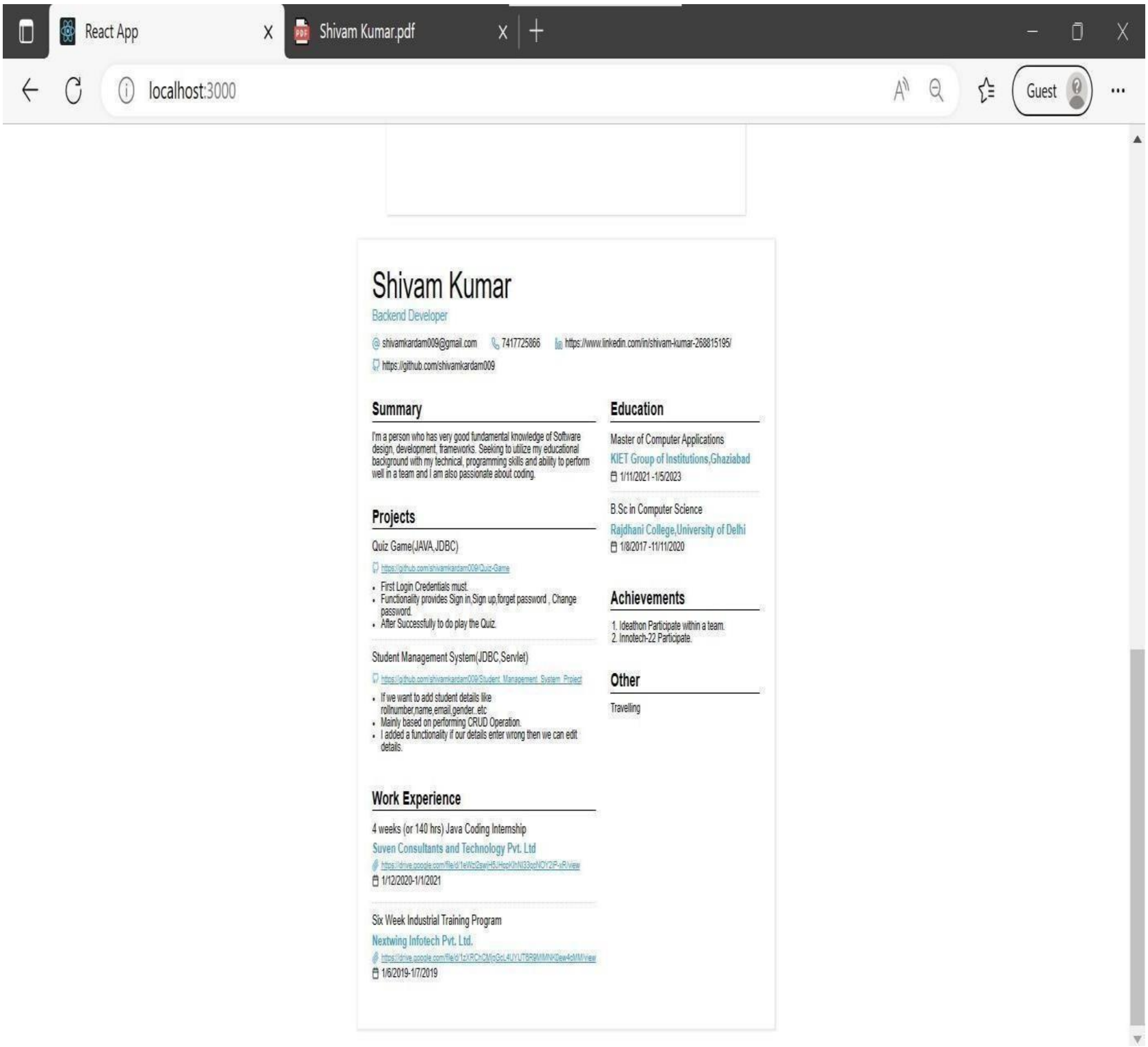
```
.container {
  --color: #239ce2;
  min-width: 700px;
  max-width: 850px;
  margin: 0 auto;
  flex: 1.2;
  height: fit-content;
  min-height: 900px;
  box-shadow: 1px 1px 3px 2px rgba(0, 0, 0, 0.1);
  display: flex;
  flex-direction: column;
  gap: 30px;
  padding: 30px;
}

.header {
  display: flex;
  flex-direction: column;
}

.header .heading {
  font-size: 2.7rem;
  font-weight: 500;
  text-transform: capitalize;
}

.header .subHeading {
  color: var(--color);
  font-weight: 500;
  font-size: 1.1rem;
}

.header .links {
  margin-top: 15px;
  display: flex;
  gap: 30px;
  flex-wrap: wrap;
  row-gap: 10px;
}
```

```
.header .link {
  font-size: 0.875rem;
  display: flex; align-
  items: center; gap:
  5px;
  cursor: pointer;
}

.header .link svg {
  color: var(--color);
  height: 16px;
  width: 16px;
}

.main {
  display: flex;
  gap: 30px;
}

.col1 {
  flex: 1.3;
  display: flex;
  flex-direction: column;
  gap: 20px;
}
.col2 {
  flex: 1;
  display: flex;
  flex-direction: column;
  gap: 20px;
}

.section .sectionTitle {
  font-size: 1.4rem;
  font-weight: bold;
  width: 100%;
  border-bottom: 2px solid #000;
}

.section .content {
  display: flex;
  flex-direction: column;
  gap: 10px;
  padding: 10px 0;
}
.section .content .item {
  border-bottom: 1px dotted lightgray;
```

```
  display: flex;
  flex-direction: column;
  gap: 4px;
  padding-bottom: 10px;
}

.section .content .item:last-child {
  border-bottom: none;
}

.content .title {
  font-weight: 500;
  font-size: 1rem;
  line-height: 1.3rem;
}

.content .subTitle {
  font-weight: bold;
  color: var(--color);
  font-size: 1rem;
}

.content .overview {
  font-size: 0.875rem;
}

.content .link {
  display: flex;
  gap: 5px;
  font-size: 0.75rem;
  cursor: pointer;
  color: var(--color);
}
.content .link svg {
  height: 14px;
  width: 14px;
  color: var(--color);
}

.content .date {
  display: flex;
  gap: 5px;
  align-items: center;
  font-size: 0.875rem;
}

.content .date svg {
  height: 14px;
```

```
  width: 14px;
}
.content .points,
.numbered {
  padding-left: 18px;
  font-size: 0.875rem;
}

.numbered li {
  list-style-type: decimal;
}

.hidden {
  display: none;
}
```

Output:


React App
Shivam Kumar.pdf
localhost:3000
Guest
Shivam Kumar
Backend Developer
shivamkardam009@gmail.com
7417725866
https://www.linkedin.com/in/shivam-kumar-268815195/
https://github.com/shivamkardam009
Summary
I'm a person who has very good fundamental knowledge of Software design, development, frameworks. Seeking to utilize my educational background with my technical, programming skills and ability to perform well in a team and I am also passionate about coding.
Projects
Quiz Game(JAVA,JDBC)
First Login Credentials must.
Functionality provides Sign in,Sign up,forget password , Change password.
After Successfully to do play the Quiz.
Student Management System(JDBC,Servlet)
If we want to add student details like rollnumber,name,email,gender, etc
Mainly based on performing CRUD Operation.
I added a functionality if our details enter wrong then we can edit details.
Work Experience
4 weeks (or 140 hrs) Java Coding Internship
Suven Consultants and Technology Pvt. Ltd
1/12/2020-1/1/2021
Six Week Industrial Training Program
Nextwing Infotech Pvt. Ltd.
1/6/2019-1/7/2019
Education
Master of Computer Applications
KIET Group of Institutions,Ghaziabad
1/11/2021 -1/5/2023
B.Sc in Computer Science
Rajdhani College,University of Delhi
1/8/2017 -11/11/2020
Achievements
1. Ideathon Participate within a team.
2. Innotech-22 Participate.
Other
Travelling

Chapter 5

5.1 Testing

5.1.1 INTRODUCTION

Testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- it is sufficiently usable,
- can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

5.1.2 Static vs. Dynamic testing

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing.

Static testing is often implicit, like proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for these are either using stubs/drivers or execution from a debugger environment.

5.1.3 White-box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

• API testing – testing of the application using public and private APIs (application programming interfaces)

• Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)

• Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies

• Mutation testing methods

• Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

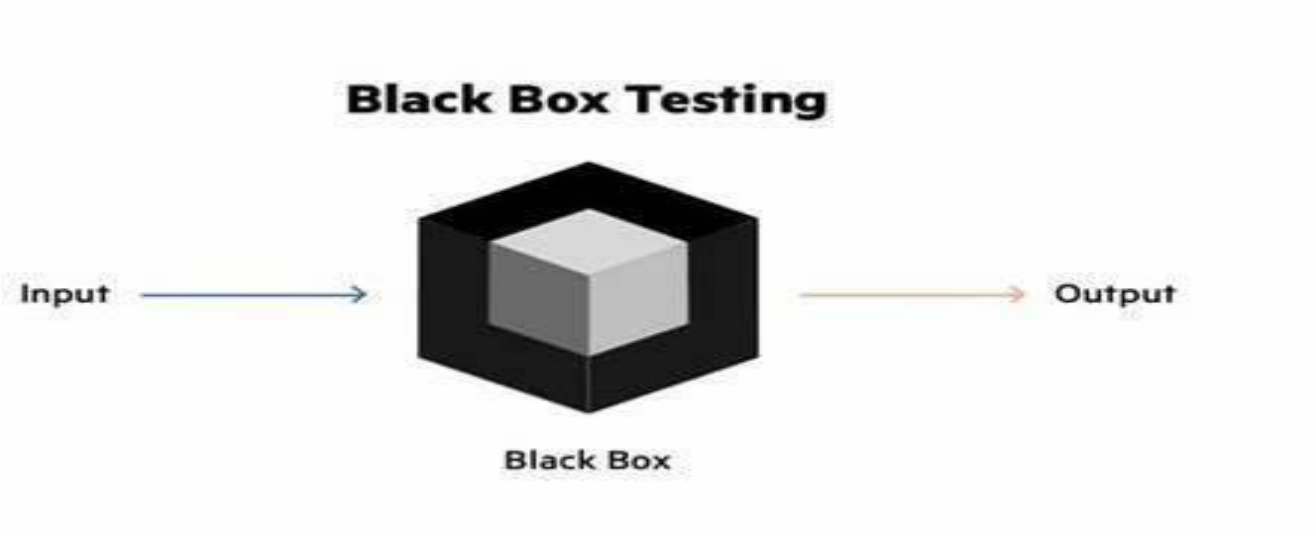
• Function coverage, which reports on functions executed

• Statement coverage, which reports on the number of lines executed to complete the test

• Decision coverage, which reports on whether both the True and the False branch of a given tens.

100% statement coverage ensures that all code paths or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly. Pseudo-tested functions and methods are those that are covered but not specified (it is possible to remove their body without breaking any test case).

5.1.4 Black-box testing



Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

5.2 TEST CASES

5.2.1 Testing using Input

Some Details of user was inserted into the editor, and then we proceeded to submit. When we checked out and tested with user entered information, Data stored successfully in their respected states. This indicates that the Editor works appropriately as it should. The "app.js" component file is responsible for handling the user data of the app. The user can manipulates his/her resume such as updating the details or, adding details to resume.

5.2.2 Resume Download

The user can only submit the resume details if he/she has entered the details correctly. Without correct details, he/she is unable to submit successfully.

CHAPTER 6

CONCLUSION

Building a strong resume ensures that you'll get through the first step of your dream job. A few years back people generally use MS Office or Google Docs for making a resume and it was a tedious task but nowadays there are a bunch of websites that provide templates for automatic resume generation either free or paid. Also, it doesn't matter whether you are using an Android application or Web application, you must require an Internet connection.

BIBLIOGRAPHY

- https://www.tutorialspoint.com/index.htm
- https://www.javatpoint.com
- https://www.w3schools.com
- https://html.com