RESUME BUILDER WEB APPLICATION

Report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

Under the supervision of

Mrs. Ankita Jain

By

Pankaj Chaurasia (2200460100074)

Tushar Sharma (2200460100114)

Vivek Singh (2200460100124)

Ritesh pal (2200460100090)

Maharana Pratap Engineering College, Kanpur

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

Session: 2024-25

DECLARATION

This is to certify that Synopsis Report Entitled "**Resume builder web application"** which is submitted in partial fulfilment of the requirement for the award of degree B.Tech. in Computer Science and Engineering to MPEC Kanpur, Dr. A.P.J. Abdul Kalam Technical University, Lucknow comprises only original work and studies carried out by students himself. The matter embodied in this synopsis has not been submitted for the award of any other degree.

Date: .....

Pankaj Chaurasia (2200460100074)

Tushar Sharma (2200460100114)

Vivek Singh (2200460100124)

Ritesh Pal (2200460100090)

Approved By:

MS. Ankita Jain

Project Guide

CSE Department

Signature: .....

Dr. Pankaj Singh Yadav

HOD

CSE Department

Signature: .....

Table of contents

Table of figures

1. Abstract

Project - Rezume: A Web-based resume builder systemfor College students

Student’s Name - Pankaj Chaurasia

Supervising Professor: Prof.Ms Ankita Jain

A resume is created by a job applicant to showcase their qualifications for a position. It contains the summary of the applicant's experience and education making them a good fit in the company for the job opening. For new graduates, listing these qualifications becomes a strenuous job as the amount of practical knowledge they hold is limited. Research suggests that about 75% of the resumes don’t ever make the threshold of the company and these numbers are even higher for fresh graduates[5]. Other common issues with resume writing include the organization of the content, having the correct margins and spacing, knowing what sections to be included or excluded, etc. Creating and formatting different resumes for different job roles is a strenuous job. The proposed website aims to help the students draft and rework their resumes based on a simple profile-based setup which will help them save invaluable time and provide them with standard and well-designed resumes to help make a compelling impact. The website supports the concept of JSON Resume, an open-source initiative to create JSON-based standards for resumes, and uses the schema of data in JSON format to create a minimalistic resume for the user[13]. This further helps to create generalized resume checkers that can easily and efficiently extract data from the resumes.

2. Introduction

2.1 Background

Resumes are written with the intent to emphasize the candidate's strengths and to show the employer how they would be a good match for a job opening. Employers generally look for the candidate's job history and their advancements in the work environment. A candidate can get a job interview and make a good first impression solely based on an impressively designed resume. There are a lot of agencies as well as websites that help students and other candidates draft resumes for a significant price. The websites focused on providing templates for the resumes leave the hard work of designing the resumes for the candidates. The candidates end up spending hours updating and personalizing the resumes as per the job requirements instead of focusing on preparing themselves for the job. Larger companies use Applicant Tracking Systems (ATS) that screen a large number of applications and filter out the best candidates for the human recruiters to view[5]. This means when a student applies for a job their resumes don't reach the recruiter at all. Hence, designing an ATS- friendly resume is necessary, but this is seldom known to the candidates. The formatting of the resume plays an important role within the ATS systems as these softwares follow a standard set of rules to extract data from a resume. On other hand, a lot of students lack the practical knowledge in the process of designing a resume. This with the constant pressure to portray your set of required skills in the best possible way in a single piece of paper is a difficult process.

2.2 Motivation

To solve the above-mentioned problem and to enable the students to easily generate resumes using an automated process is a necessity. Students generally learn to format their resumes using word processors or take on learning LaTeX to code their resumes. This manual method of designing resumes requires brute force methodology and the students are generally left with unembellished resumes. There are multiple websites on the internet that provide similar functionalities but do always come at a price. Each resume costs about $100 on such portals which is something a student would rarely spend for designing resumes.

JSON Resumes is an initiative by a group of developers to bring standardization to the resume-making process. It uses a predefined format of the resume in JSON. JSON is a lightweight data interchangeable format that is easy to use, read and write by humans. Furthermore, JSON is built up of attribute-value pairs and arrays making them super easy to store and transmit in human-readable formats. Every developer working in the field are familiar with JSON formatted data and hence using it with an easy to use user interface for everyone helps achieve the goal of standardizing the CV fields to improve compatibility between tools like ATS checkers or resume parsers, ease of conversion, and above all ease of formatting the data into pre-designed formats. JSON when compared to the XML format is briefly encoded and thus JSON parsers require less space-time complexity. JSON also is built for faster transmission of data whereas XML does a lot more.
Intending to further spread the amazing initiative of JSON Resumes to non-developers who aren't familiar with JSON, a profile-based approach is preeminent. The user would fill a set of forms to create a profile and ultimately generate standardized resumes with a few clicks. This enables them to generate resumes compatible with ATS checkers using resume templates available with minimalist design, color, and logical layouts. This further makes it easier for students to recreate their resumes without having to format them repeatedly.

2.3 Contribution

The proposed project focuses on achieving a blend of all the above features with a focus on a user-friendly interface, a minimalistic design, and responsiveness. It would host a number of resume templates custom curated for new graduates. Each student will be able to build a free profile that holds data replicating their Social Media profiles. The stored data could be edited on the go and create quick and personalized resumes for the students efficiently. The generator allows the user to play around with the visibility and the order of the sections and subsections of the resume to make a one-page resume. The motive is to use the user-created profiles and store that data into a standard Resume format in JSON and generate resumes for the users within minutes. The profile will be editable and will be converted into a resume based on the templates selected by the user within seconds. This means the users will save a lot of precious time spent on aligning their texts, handling spacing and margins within MS Word, or even writing code on LaTeX.

A user will be able to add features like personal information including links to their Linkedin profiles or online portfolios, a summary of their qualifications and skills, their education, the experience they have in the respective field. The provided resume templates will follow several rules like the four-quadrant rule, which will use columns to represent information, and other design and typographic rules. Users would also have an option to select from a variety of colors and a set of standard fonts to portray their true selves via the resume.

The website aims at providing a professional-looking resume with minimal effort. To accomplish this further the user can also use data saved in JSON files from their previous resumes or even download the JSON format of the resume data to use them in other templates later on. This gives them the flexibility to generate resumes and customize them according to their specific needs.

3. LITERATURE REVIEW

3.1 Studying the existing systems

The resume builder systems on the web were explored and their best features have been added to the application. The biggest problem in most of these systems is that they endure a high cost every time a user builds a resume. The services do come with expert advice to improve the resume but do cost about hundreds of dollars. There are a few systems that are free of cost like the one I. Wu [1] describes the Pro Resume that focuses only on infographic resumes. Also, the website has only a single template but gives some design flexibility to the user. Similarly, Ingale [2] in her proposal of the Resume Portal focuses on an integrated system where the user can apply for job vacancies posted on the portal. This portal covers a lot of bases like building resumes and getting jobs with a lack of research and focusing on vastly used Applicant Tracking Systems that companies already use. It also assumes the companies will be posting their job requirements on the portal which makes the process of generating resumes coupled to the requirements. The one thing that can be useful is creating links to user resumes that can be shared by the user instead of uploading a PDF file everywhere. Other systems propose resume generators with multilingual capabilities or the use of AR markers to visualize various parts of printed resumes on the web.

3.2 Research on " How to make ATS Friendly Resumes"

The Ladders eye-tracking study suggested that a recruiter spends about 7.4 seconds on a resume hence it is very important to focus on having a clean-looking resume, having short declarative statements, and most importantly having the standard headings and subheading without any extraordinary design experiments[10]. Thus, the details presented by Risavy in the Resume Research Literature[3] serve a very important role in the development of every template for the project. The valuable tips shared by Clift in the "How to beat ATS" [4] also list a checklist that should be followed while making resumes. The design tips shared on the [7] RIT's Career Services and Co- op and Purdue's Resume Design [8] article also provide important insight on the checklist created previously.

3.3 Exploration of features to improve the user experience

The overall functioning of an ATS is studied and the application proposed by Tiwari and et al in “Applicant Tracking System”[5] focuses on Natural Language Processing to only detect job-related words. Using the knowledge we have about the design of resume templates, a rubric of an improved system can be drafted. This would be done by including the factors listed and a higher preference would be given to the spellings and grammar of the Resume.

4. METHODOLOGY

The main idea behind the project is for the user to have a customizable resume-making experience with the ability to create, modify and delete the sections and subsections of the resume from the home screen itself. The JSON Resume standard is used to comply with a data format that can be used in the application by the users to create their resumes in a regulated manner. This format is discussed in detail in the upcoming sections. Further to make the visual aspect of the resume a number of functionalities are added so that users can generate the best possible resume for their needs. This is done by adding functionalities like changing the primary color in the template, updating the font family of the resume, size of the resume among A4, letter, and legal. Further, the user can either download the resume in the pdf format and also download the data in the JSON format that can be used elsewhere. A user will have to register to a secure portal to access all these features and then log in every time they want to use it. The user can either be a general user or an admin that will have more functionalities to add sections, new colors, and font faces to the application. The authentication system for the website is built on the principles of JSON web tokens and a role field within the user defines if they are an admin or not. The user authentication is handled in a highly secure way as the user data being stored on the cloud is sensitive. The user interface of the website is kept minimalistic with essential components combined together in an esthetic manner to reflect professionalism. The screen shows the resume the user is currently working on and all the components are stored within the slider menu. This reduces the number of unnecessary user interactions making the website easy to use.

4.1 Architecture

The application follows the Model View Controller architecture paradigm to promote organized programming; the functionality, logic, and user interface are all built separately to handle specific development aspects of the application.

Model - The model is the store of the application and handles the data and data-related logic of the application. This project uses a NoSQL database MongoDBs cloud implementation. The model is connected directly to the database so all the modifications to the data are done in the model component. The controller communicates with the model and requests the data and this request is further executed by the model to either retrieve, modify or add data from the database.

View - The view is what the user actually sees and how they interact with the application. The user can interact with the data and other business logic via the view. The view is built up using components using react and the dynamic data from the controller. The data actually comes from the Model but the controller acts as an interface between the view and the model. The user performs actions using buttons or other interactions on the view.

Controller - The controller is the main man that enables the interaction between the model and the view. At the backend, the controller just tells the model what needs to be done and then the data logic is handled by the model. Further the controller processes all the data received and instructs the view on how to represent the data. The controller also processes all the user inputs and interactions and then decides and informs both the view and the model what happens next. The controllers in the application are Express.js. Redux is also used as store management within JavaScript that acts as the data store for the UI layer.

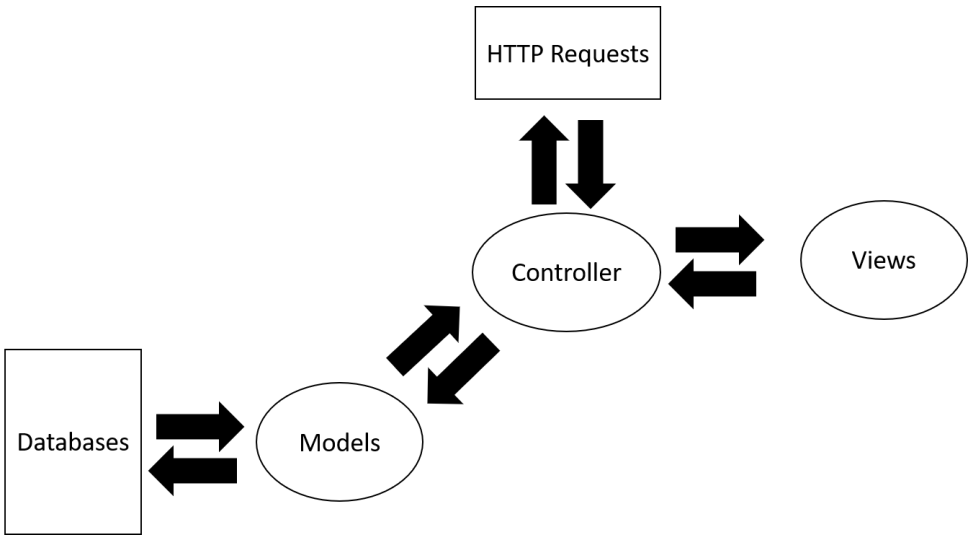

Figure 1: MVC Architecture

4.2 Tools and Technologies used

The project is built on the MERN stack which is used for faster and easier deployment of full-stack applications.

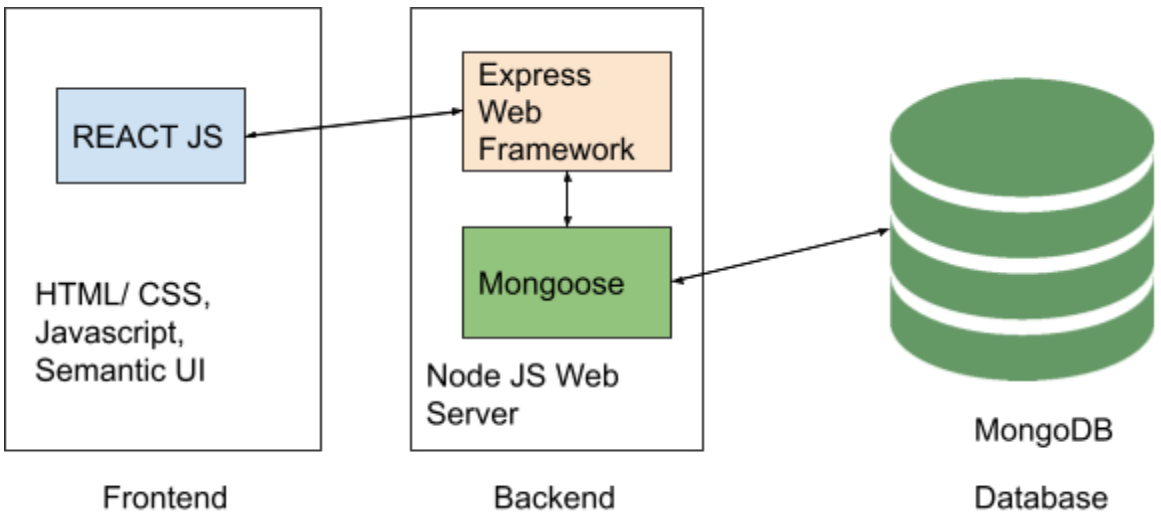

Figure 2: MERN Stack Architecture

MONGO DB - MongoDB is a NoSQL database management system that leverages a JSON-style storage format BSON. BSON is also known as Binary JSON is an extension of JSON that encodes type and length information thus making its parsing faster. This further allows MongoDB to store binary data and dates and to efficiently index, map, and nest data in

React allows unidirectional data flow so it becomes difficult to maintain all the states at one place within the application and to transfer it when and where needed. Thus the sharing of data across components is easily enabled by using state management that creates a physical data structure that stores the state of the application that the components can access. Thus Redux becomes a very important aspect for this application as a lot of data coming from the API is stored here. Additionally, multiple functionalities like visibility, the font face, order of the sections, etc are stored in the store as states. The actions on the current state of the application are performed by pure functions known as reducers returning the new state. An action is an object containing the payload of information that acts as the only source of information for the Redux store to be updated.

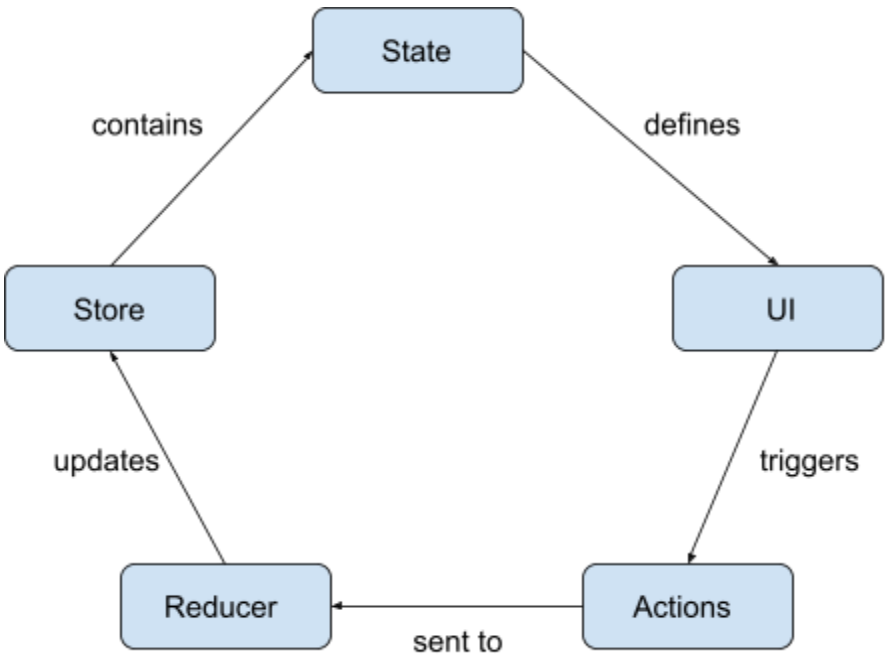

Figure 3: Working of React Redux

JSON - JavaScript Object Notation abbreviated as JSON is a lightweight schema-less, text-based representation of structured data. It supports the data-interchange format and is based on key-value pairs and ordered lists. It is parsed and generated by the machines easily and is supported in most programming languages either natively or by using libraries. JSON is a data-interchange format as it is a language-independent text format and two data structures

SEMANTIC UI - Semantic UI is a front-end development framework designed for theming that contains components that can be used to build responsive websites easily. It unpacks a variety of themes utilizing concise HTML, simplified CSS, and intuitive JavaScript and integrates perfectly with React, Angular, etc. It enables the users to design stunning designs by selecting their preferences from a ton of UI components. This helps reduce the file size and has lower load times with easy customizations that can be done on the go as it uses semantic class names. Semantic UI was chosen for this project because it works perfectly with React and provides the UI components needed with efficient and smaller code segments.

REST API - REST is an acronym for REpresentational State Transfer which is an architectural style commonly used in Web Services. RESTful API is built on six guiding principles consisting of a uniform interface, separation of the client and the server, statelessness, ability to reuse the response data, have a multi-layered system, and allow running the API code in form of applets or
scripts. The transaction within a RESTful API is broken down into small modules that address a part of the transaction. Further, the calls within the REST API are stateless making it compatible with cloud-based applications as they can be easily redeployed if something fails, and can scale to accommodate load changes. The database being used in the application is a cloud-based implementation of a No SQL database thus making RESTful architecture the best choice for the development of this application. The HTTP methodologies used by the REST architecture that is used in the project are
GET - This request is used to retrieve a resource from the server.

PUT - This request is used to update an existing resource on the server.

POST - This request is used to create a new resource on the server.

DELETE - This request is used to delete a resource on the server.

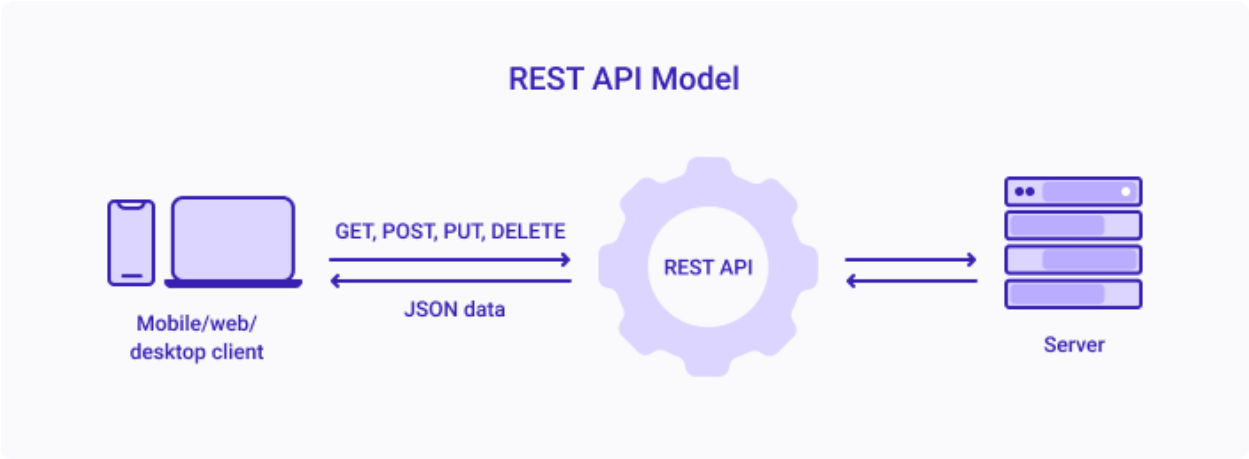

Figure 5: REST API Model

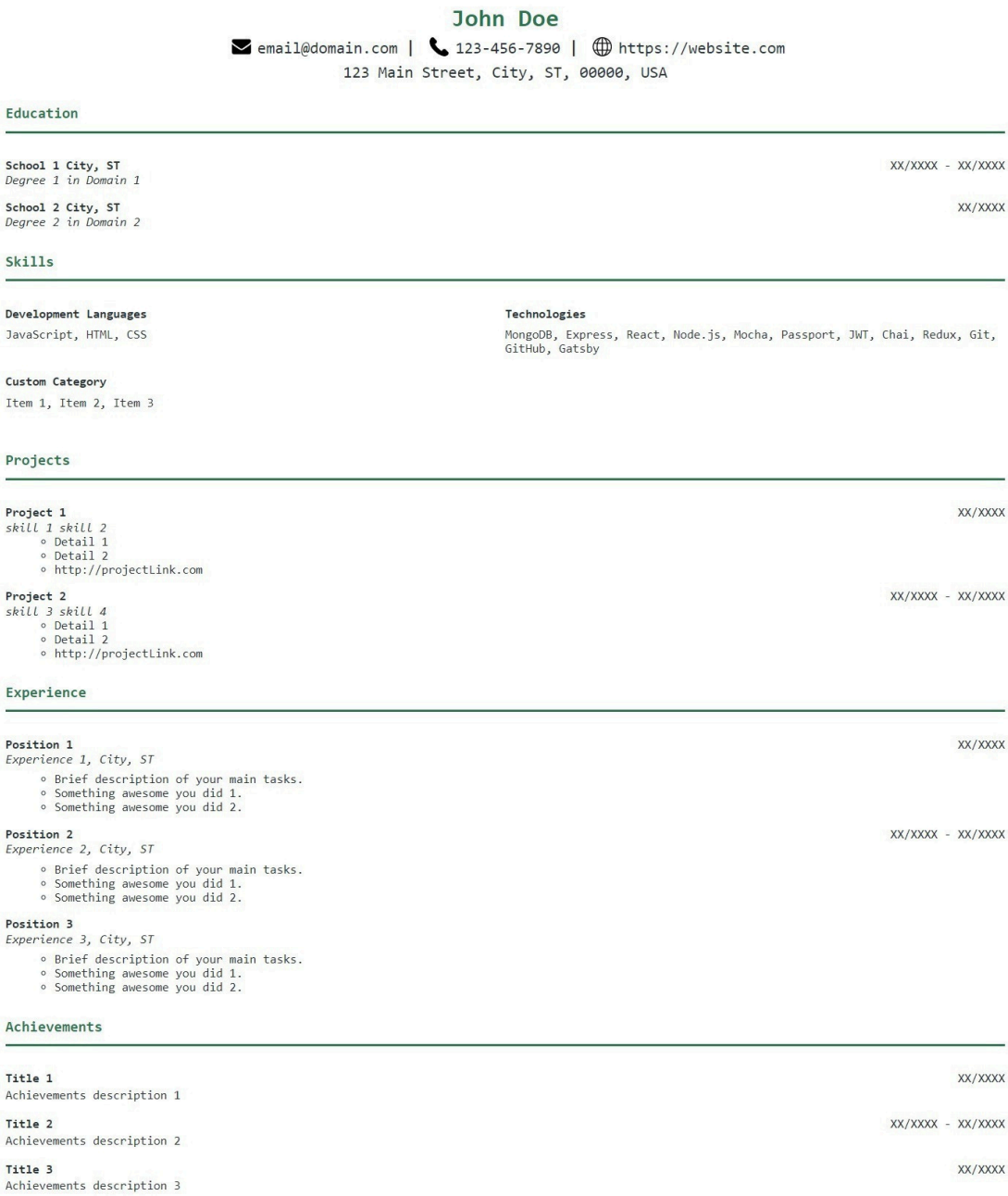

Figure 18: Resume Template 1

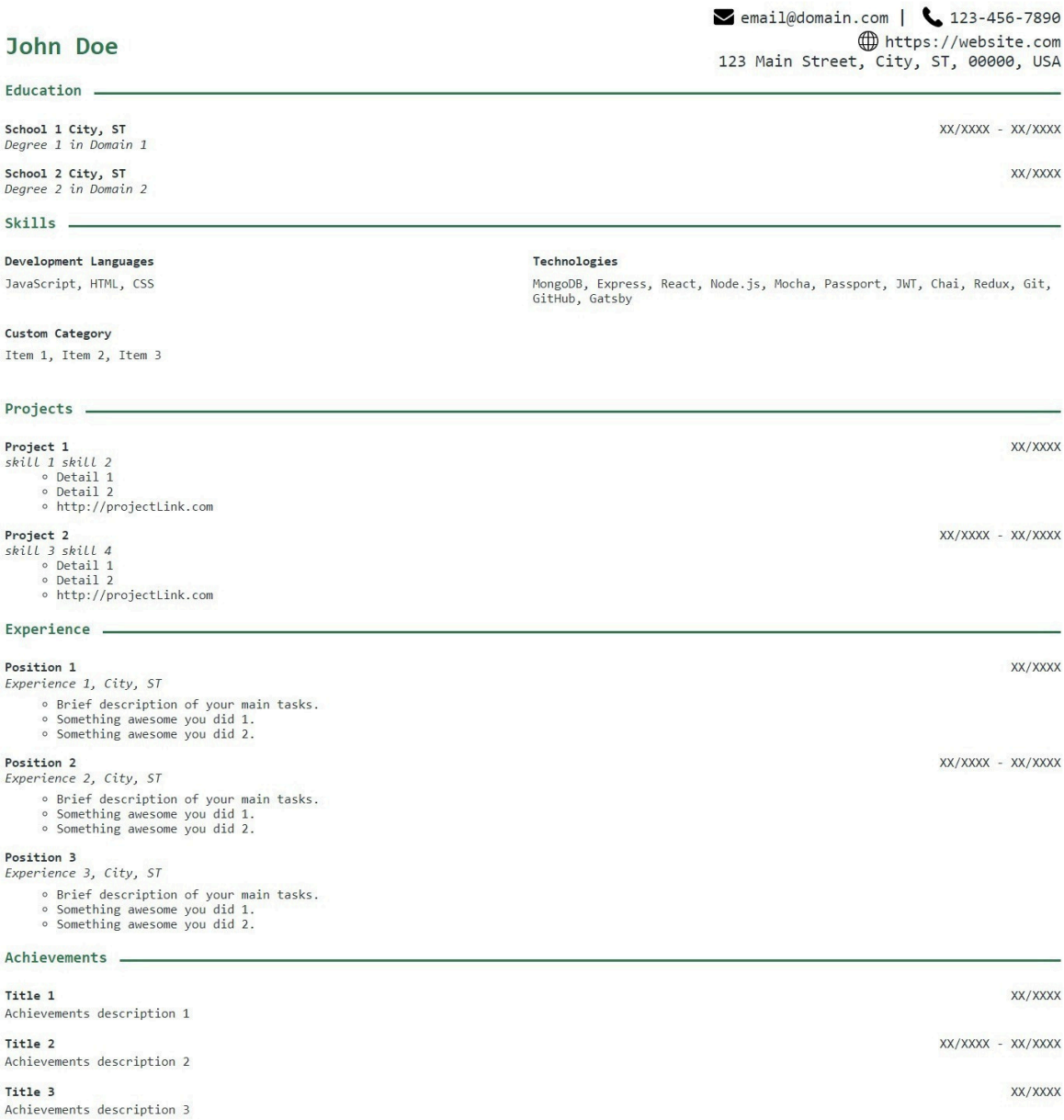

Figure 19: Resume Template 2

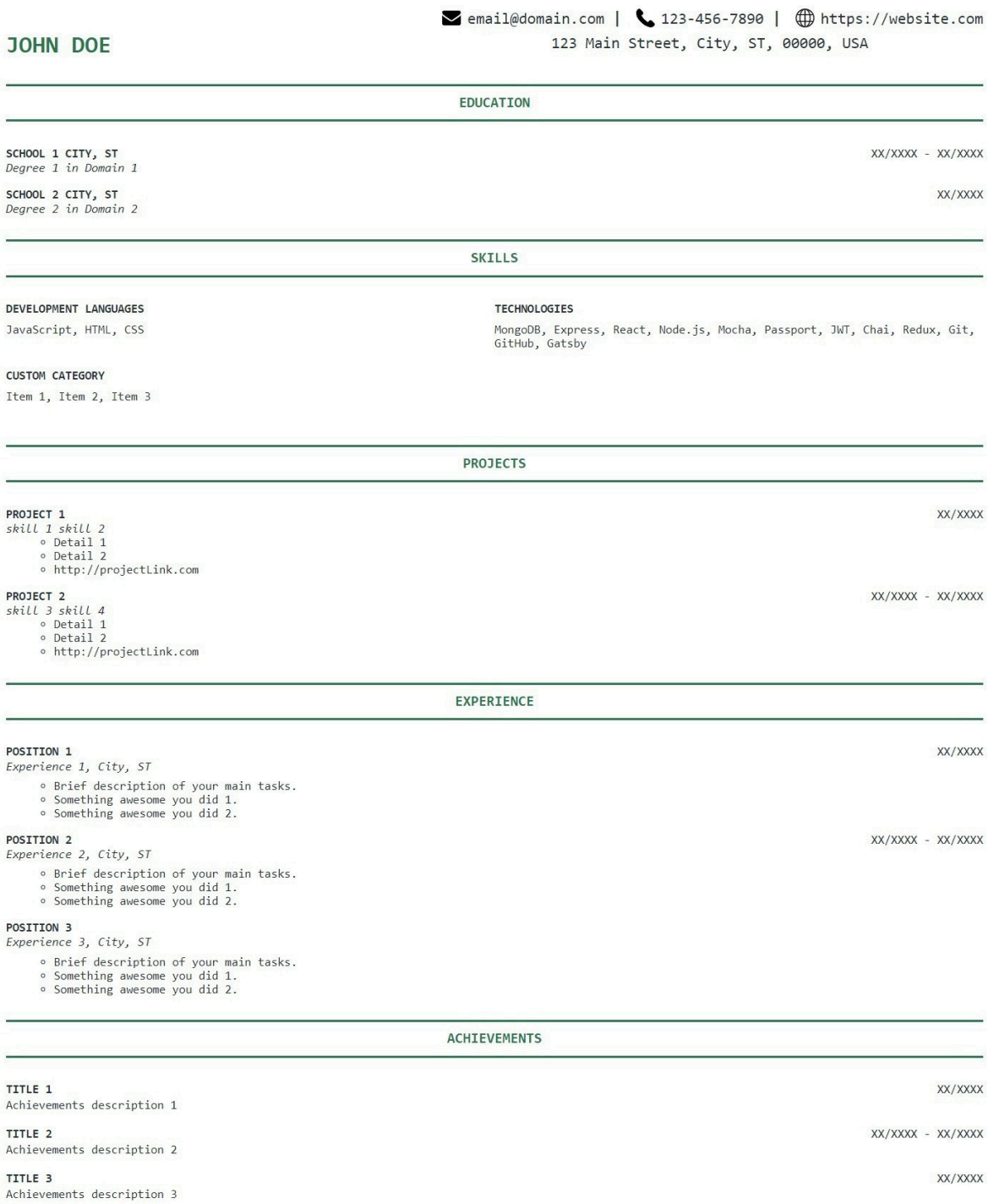

Figure 20: Resume Template 3

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Professional Resume Builder</title>
<link rel="stylesheet" href="style.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10.1/html2pdf.bundle.min.js"></script>
</head>
<body>
<div class="container">
<nav class="navbar">
<div class="logo">Resume Builder</div>
<div class="nav-buttons">
<button id="saveBtn" class="nav-btn"><i class="fas fa-save"></i> Save</button>
<button id="loadBtn" class="nav-btn"><i class="fas fa-folder-open"></i> Load</button>
<button id="downloadBtn" class="nav-btn"><i class="fas fa-download"></i> Download PDF</button>
</div>
</nav>

<div class="main-content">
<div class="form-section">
<!-- Personal Information -->
<div class="section">
<h2><i class="fas fa-user"></i> Personal Information</h2>
<div class="form-grid">
<div class="form-group">
<label>Full Name</label>
<input type="text" id="fullName" name="fullName" required>
</div>
<div class="form-group">
<label>Email</label>
<input type="email" id="email" name="email" required>
</div>
<div class="form-group">
<label>Phone</label>
<input type="tel" id="phone" name="phone">
</div>
<div class="form-group">
<label>Website</label>
<input type="url" id="website" name="website">
</div>
<div class="form-group full-width">
<label>Address</label>
<textarea id="address" name="address"></textarea>
</div>
</div>
```

```
<!-- Professional Summary -->
<div class="section">
<h2><i class="fas fa-briefcase"></i> Professional Summary</h2>
<textarea id="summary" name="summary" rows="4"></textarea>
</div>
<!-- Education -->
<div class="section">
<h2><i class="fas fa-graduation-cap"></i> Education</h2>
<div id="educationList">
<!-- Education entries will be added here -->
</div>
<button onclick="addEducation()" class="add-btn">
<i class="fas fa-plus"></i> Add Education
</button>
</div>
<!-- Experience -->
<div class="section">
<h2><i class="fas fa-building"></i> Work Experience</h2>
<div id="experienceList">
<!-- Experience entries will be added here -->
</div>
<button onclick="addExperience()" class="add-btn">
<i class="fas fa-plus"></i> Add Experience
</button>
</div>
<!-- Skills -->
<div class="section">
<h2><i class="fas fa-tools"></i> Skills</h2>
<div class="skills-input-container">
<input type="text" id="skillInput" placeholder="Type a skill and press Enter">
<div id="skillsList" class="skills-list">
<!-- Skills will be added here -->
</div>
</div>
</div>
<!-- Projects -->
<div class="section">
<h2><i class="fas fa-project-diagram"></i> Projects</h2>
<div id="projectsList">
<!-- Projects will be added here -->
</div>
<button onclick="addProject()" class="add-btn">
<i class="fas fa-plus"></i> Add Project
</button>
</div>
</div>
```

load_resume.php

```
<?php
session_start();
header('Content-Type: application/json');

try {
// Check if there's a resume file saved in session
if (!isset($_SESSION['last_resume'])) {
 throw new Exception('No saved resume found');
}

$filename = $_SESSION['last_resume'];

// Check if file exists
if (!file_exists($filename)) {
throw new Exception('Resume file not found');
}

// Read the file
$data = file_get_contents($filename);
if ($data === false) {
throw new Exception('Failed to read resume data');
}

// Validate JSON
$resume = json_decode($data, true);
if (json_last_error() !== JSON_ERROR_NONE) {
throw new Exception('Invalid resume data format');
}

echo json_encode(['success' => true, 'resume' => $resume]);
} catch (Exception $e) {
http_response_code(400);
echo json_encode(['success' => false, 'message' => $e->getMessage()]);
}
```

```
<?php
session_start();
header('Content-Type: application/json');

// Function to get AI suggestions
function getAISuggestions($content) {
$apiKey = 'YOUR_OPENAI_API_KEY'; // Replace with your actual API key
$url = 'https://api.openai.com/v1/engines/davinci-codex/completions';

$data = [
'prompt' => "Provide suggestions for improving the following resume content: \n$content",
'max_tokens' => 150,
'temperature' => 0.7
];

$options = [
'http' => [
'header'  => "Content-type: application/json\r\nAuthorization: Bearer $apiKey\r\n",
'method'  => 'POST',
'content' => json_encode($data),
],
];

$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);

if ($result === FALSE) {
throw new Exception('Error contacting AI service');
}

$response = json_decode($result, true);
return $response['choices'][0]['text'] ?? 'No suggestions available.';
}

try {
// Validate input
if (!isset($_POST['data'])) {
throw new Exception('No data received');
}

$data = json_decode($_POST['data'], true);
if (json_last_error() !== JSON_ERROR_NONE) {
throw new Exception('Invalid JSON data');
}
```

```
// Create data directory if it doesn't exist
$dataDir = 'resumes';
if (!file_exists($dataDir)) {
if (!mkdir($dataDir, 0777, true)) {
throw new Exception('Failed to create directories...');
}
}

// Generate unique filename using timestamp
$filename = $dataDir . '/resume_' . time() . '.json';

// Save the data
if (file_put_contents($filename, json_encode($data)) === false) {
throw new Exception('Failed to save resume data');
}

// Store the filename in session for later retrieval
$_SESSION['last_resume'] = $filename;

// Integrate AI suggestions
$aiSuggestions = getAISuggestions($data['content']);

// Include AI suggestions in the response
echo json_encode(['success' => true, 'message' => 'Resume saved successfully', 'aiSuggestions' =>
$aiSuggestions]);
} catch (Exception $e) {
http_response_code(400);
echo json_encode(['success' => false, 'message' => $e->getMessage()]);
}
```

save_student.php

```
<?php
// Check if the form was submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
// Collect and sanitize form data
$name = htmlspecialchars($_POST['name']);
$age = htmlspecialchars($_POST['age']);
$email = htmlspecialchars($_POST['email']);
$course = htmlspecialchars($_POST['course']);

// Here you can add code to save the data to a database or file
// For demonstration, we'll just display the data
echo "<h2>Student Details Submitted:</h2>";
echo "<p>Name: $name</p>";
echo "<p>Age: $age</p>";
echo "<p>Email: $email</p>";
echo "<p>Course: $course</p>";
} else {
echo "<p>Invalid request method.</p>";
}
?>
```

script.js

```
// DOM Elements
document.addEventListener('DOMContentLoaded', () => {
// Initialize event listeners
document.getElementById('saveBtn').addEventListener('click', saveResume);
document.getElementById('loadBtn').addEventListener('click', loadResume);
document.getElementById('downloadBtn').addEventListener('click', downloadResume);
document.getElementById('skillInput').addEventListener('keypress', handleSkillInput);
document.getElementById('resumeForm').addEventListener('submit', validateForm);

// Initialize first entries
addEducation();
addExperience();
addProject();
updatePreview();
});

// Education Section
function addEducation() {
const educationList = document.getElementById('educationList');
const entry = document.createElement('div');
entry.className = 'education-entry form-grid';
entry.innerHTML = `
<div class="form-group">
<input type="text" placeholder="Degree/Course" class="education-degree">
</div>
<div class="form-group">
<input type="text" placeholder="Institution" class="education-institution">
</div>
<div class="form-group">
<input type="text" placeholder="Year" class="education-year">
</div>
<div class="form-group">
<input type="text" placeholder="Score/GPA" class="education-score">
</div>
<button onclick="removeEntry(this)" class="remove-btn"><i class="fas fa-trash"></i>
</button>
`;
educationList.appendChild(entry);
updatePreview();
}
```

```
// Experience Section
function addExperience() {
const experienceList = document.getElementById('experienceList');
const entry = document.createElement('div');
entry.className = 'experience-entry';
entry.innerHTML = `
<div class="form-grid">
<div class="form-group">
<input type="text" placeholder="Job Title" class="experience-title">
</div>
<div class="form-group">
<input type="text" placeholder="Company" class="experience-company">
</div>
<div class="form-group">
<input type="text" placeholder="Start Date" class="experience-start">
</div>
<div class="form-group">
<input type="text" placeholder="End Date" class="experience-end">
</div>
</div>
<div class="form-group">
<textarea placeholder="Job Description" class="experience-description"></textarea>
</div>
<button onclick="removeEntry(this)" class="remove-btn"><i class="fas fa-trash"></i>
</button>
`;
experienceList.appendChild(entry);
updatePreview();
}

// Project Section
function addProject() {
const projectsList = document.getElementById('projectsList');
const entry = document.createElement('div');
entry.className = 'project-entry';
entry.innerHTML = `
<div class="form-grid">
<div class="form-group">
<input type="text" placeholder="Project Name" class="project-name">
</div>
<div class="form-group">
<input type="text" placeholder="Technologies Used" class="project-tech">
</div>
</div>
<div class="form-group">
<textarea placeholder="Project Description" class="project-description"></textarea>
</div>
<div class="form-group">
<input type="url" placeholder="Project Link" class="project-link">
</div>
```

```
// Skills Section
function handleSkillInput(e) {
if (e.key === 'Enter') {
e.preventDefault();
const skillInput = document.getElementById('skillInput');
const skill = skillInput.value.trim();
if (skill) {
addSkill(skill);
skillInput.value = '';
updatePreview();
}
}
}

function addSkill(skillText) {
const skillsList = document.getElementById('skillsList');
const skill = document.createElement('div');
skill.className = 'skill-tag';
skill.innerHTML = `
${skillText}
<button onclick="removeEntry(this.parentElement)">×</button>
`;
skillsList.appendChild(skill);
}

// Remove Entry
function removeEntry(button) {
button.closest('.education-entry, .experience-entry, .project-entry, .skill-tag').remove();
updatePreview();
}

// Update Preview
function updatePreview() {
const preview = document.getElementById('resumePreview');
const data = collectFormData();

preview.innerHTML = `
<div class="resume">
<header class="resume-header">
<h1>${data.personalInfo.fullName || 'Your Name'}</h1>
<div class="contact-info">
${data.personalInfo.email ? `<p><i class="fas fa-envelope"></i>
${data.personalInfo.email}</p>` : ''}
${data.personalInfo.phone ? `<p><i class="fas fa-phone"></i>
${data.personalInfo.phone}</p>` : ''}
${data.personalInfo.website ? `<p><i class="fas fa-globe"></i>
${data.personalInfo.website}</p>` : ''}
${data.personalInfo.address ? `<p><i class="fas fa-map-marker-alt"></i>
${data.personalInfo.address}</p>` : ''}
</div>
```

```
// Collect Form Data
function collectFormData() {
return {
personalInfo: {
fullName: document.getElementById('fullName').value,
email: document.getElementById('email').value,
phone: document.getElementById('phone').value,
website: document.getElementById('website').value,
address: document.getElementById('address').value
},
summary: document.getElementById('summary').value,
education: Array.from(document.querySelectorAll('.education-entry')).map(entry => ({
degree: entry.querySelector('.education-degree').value,
institution: entry.querySelector('.education-institution').value,
year: entry.querySelector('.education-year').value,
score: entry.querySelector('.education-score').value
})),
experience: Array.from(document.querySelectorAll('.experience-entry')).map(entry => ({
title: entry.querySelector('.experience-title').value,
company: entry.querySelector('.experience-company').value,
startDate: entry.querySelector('.experience-start').value,
endDate: entry.querySelector('.experience-end').value,
description: entry.querySelector('.experience-description').value
})),
projects: Array.from(document.querySelectorAll('.project-entry')).map(entry => ({
name: entry.querySelector('.project-name').value,
technologies: entry.querySelector('.project-tech').value,
description: entry.querySelector('.project-description').value,
link: entry.querySelector('.project-link').value
})),
skills: Array.from(document.querySelectorAll('.skill-tag')).map(tag => tag.textContent.trim())
};
}

// Validate Form
function validateForm(event) {
let valid = true;
const fullName = document.getElementById('fullName').value.trim();
const email = document.getElementById('email').value.trim();
const phone = document.getElementById('phone').value.trim();
const address = document.getElementById('address').value.trim();

if (fullName === '') {
alert('Full Name is required.');
valid = false;
}

if (email === '' || !/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
alert('Please enter a valid email address.');
```

```
// Save Resume
function saveResume() {
const formData = new FormData(document.getElementById('resumeForm'));
fetch('save_resume.php', {
method: 'POST',
body: formData
})
.then(response => response.text())
.then(data => {
alert('Resume saved successfully!');
})
.catch(error => {
console.error('Error saving resume:', error);
});
}

// Load Resume
function loadResume() {
fetch('load_resume.php')
.then(response => response.json())
.then(data => {
document.getElementById('name').value = data.name;
document.getElementById('email').value = data.email;
document.getElementById('education').value = data.education;
document.getElementById('work').value = data.work;
document.getElementById('skills').value = data.skills;
})
.catch(error => {
console.error('Error loading resume:', error);
});
}

// Download Resume
function downloadResume() {
const element = document.getElementById('resumePreview');
const opt = {
margin: 1,
filename: 'resume.pdf',
image: { type: 'jpeg', quality: 0.98 },
html2canvas: { scale: 2 },
jsPDF: { unit: 'in', format: 'letter', orientation: 'portrait' }
};

html2pdf().set(opt).from(element).save();
}

import { validateStudentForm } from './studentFormLibrary.js';

validateStudentForm('studentForm');
```

style.css

```
:root {
--primary-color: #2c3e50;
--secondary-color: #3498db;
--accent-color: #e74c3c;
--background-color: #f5f5f5;
--text-color: #333;
--border-color: #ddd;
--success-color: #27ae60;
--shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
background-color: var(--background-color);
color: var(--text-color);
line-height: 1.6;
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
background-color: #f4f4f9;
}

.container {
max-width: 1400px;
margin: 0 auto;
}

/* Navbar Styles */
.navbar {
background-color: var(--primary-color);
padding: 1rem 2rem;
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 2rem;
box-shadow: var(--shadow);
}
```

```
.navbar {
background-color: var(--primary-color);
padding: 1rem 2rem;
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 2rem;
box-shadow: var(--shadow);
}

.logo {
color: white;
font-size: 1.5rem;
font-weight: bold;
}

.nav-buttons {
display: flex;
gap: 1rem;
}

.nav-btn {
background-color: transparent;
color: white;
border: 1px solid white;
padding: 0.5rem 1rem;
border-radius: 4px;
cursor: pointer;
transition: all 0.3s ease;
display: flex;
align-items: center;
gap: 0.5rem;
}

.nav-btn:hover {
background-color: white;
color: var(--primary-color);
}

/* Main Content Layout */
.main-content {
display: grid;
grid-template-columns: 1fr 1fr;
gap: 2rem;
padding: 0 2rem;
}

/* Form Styles */
.form-section {
background-color: white;
```

```
@media (max-width: 768px) {
    .form-grid {
        grid-template-columns: 1fr;
    }

    .navbar {
        flex-direction: column;
        gap: 1rem;
    }

    .container {
        padding: 1rem;
    }
}

/* Print Styles */
@media print {
    .form-section, .navbar {
        display: none;
    }

    .main-content {
        display: block;
    }

    .preview-section {
        position: static;
        height: auto;
    }

    .resume-preview {
        box-shadow: none;
    }
}

header {
    background-color: #4CAF50;
    color: white;
    text-align: center;
    padding: 1em 0;
}

main {
    padding: 20px;
    max-width: 800px;
    margin: 0 auto;
}
```

The resume templates in the images above are developed in conjugation with the logical rules of design and typography. The features such as changing the font size and font are provided to let the user customize the resumes according to their preferences while only allowing a set of standard fonts on the portal. The templates are designed in a single-column data representation with clear and bold headings. This enables the ATC checkers to easily parse the resume enabling more hits. The user can also download all the data linked to their resumes in the form of a JSON file and can be then uploaded on other templates that support JSON data to create awesome resumes. Similarly, JSON data manually coded by the users or from other JSON resume websites can also be uploaded on Rezume. The JSON format for the resume can be viewed by the user by clicking on the round file icon beside the Load from File button.

Figure 21: Download & Load Data Functionality

The user can also select the page size from the standard sizes they would prefer their resume to be generated in. Currently, Rezume supports Letter, A4, and Legal sizes.

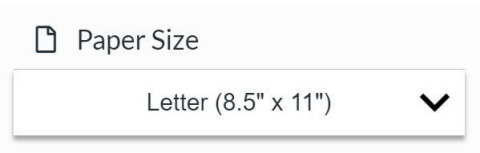

Figure 22: Paper Size Functionality

All the customizations made to the resume can be stored in the local storage and will be reloaded when the user visits the website next. This is optional for the user and can be toggled by the autosave switch. The delete icon next to it can be used to delete all the data stored in the local storage. This would also delete the tokens stored in order to keep the user signed in on their device. Finally, the user can directly sign out of the website and can keep their data stored by clicking on the Signout button.

Figure 23: Local Storage & Sign Out Functionality

The following picture shows the state chart of the tools used within the application. The tools state within Redux and react are responsible for handling all the above-mentioned functionalities.

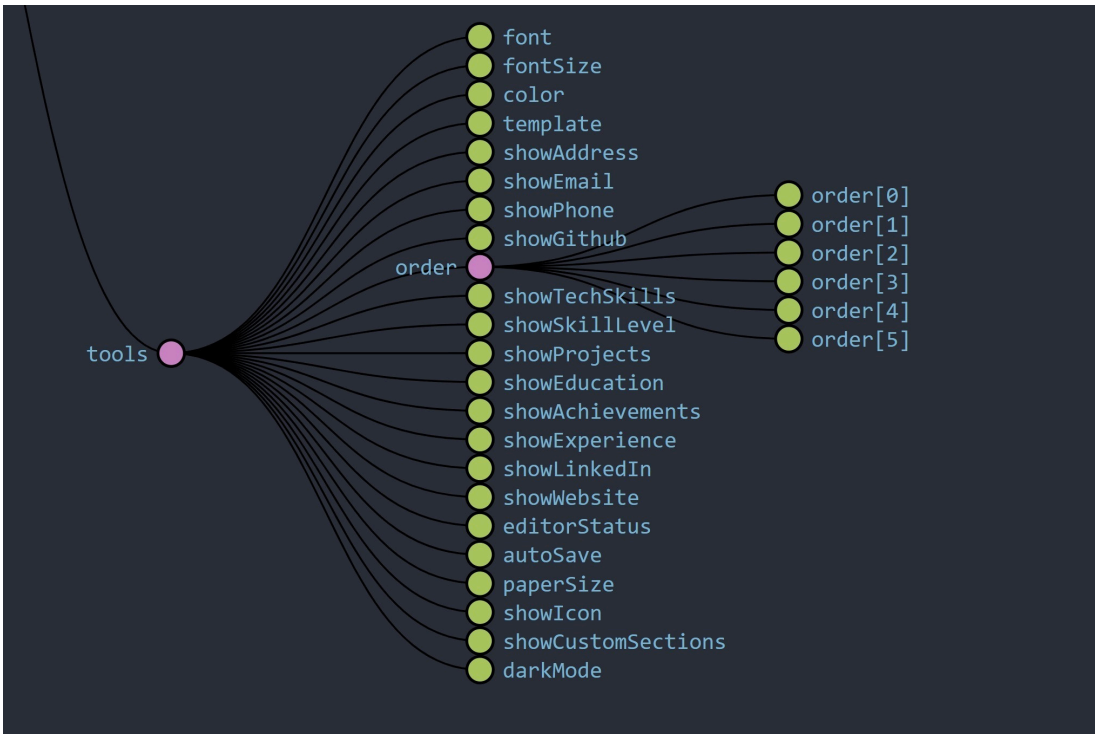

Figure 24: Functionality State Chart in Redux

The admin is directed to an admin dashboard directly and can use it to customize the Rezume portal for the users. The following image shows the Admin Dashboard. Accessing this state data within components directly wouldnt be possible on React without passing the data throughout the components. React only allows direct passing of data from parent to child component and an unconventional way to pass it in the reverse direction using callbacks. Hence using a Redux state helps accessing the data at any components directly.

6. Future Works

The aim of this application is to help students across fields generate Resumes easily. Significant effort has been put in to make the application visually appealing and smooth. But having experts in UI/ UX design wireframe templates for the websites using the current trends in the industry would help make the website much better and user friendly. Further, this website needs thorough usability testing where representative users can try it and give their advice on how to make it better. This would also help understand how easy it is for users to navigate through the website, the complexity of the interface, how usable are the functionalities, etc. Rigorous application testing is also required before it can be hosted so as to discover any major bugs. The requirements for this application were drawn out based on the research of the existing systems and by understanding the problems faced by the students during their resume creation process. There wasn't a large analysis to understand what features that the users might actually want with such a portal. Insight on that would be really helpful to expand the project further. There are certain features like letting the user save the resumes they built on the portal stored within their profile so that they can access them by easily signing in. Resume sharing functionality via email could also be a useful feature. Having the system generate cover letters in similar templates to the resume would be a useful feature that could be implemented in the future. The application is built upon REST API with routes being protected using authentication and authorization middlewares. Rigorous testing and update of the API would be needed to remove any security concerns like Broken Access Control, Sensitive Data Exposure. Action must be taken to remove injection attacks. As this application is highly dependent on data being properly stored and accessed, proper client-side and server validation is necessary. Injecting the wrong format of data within the application might lead to opening access to unauthorized users and would be a major security concern. There are no complexity rules present in the password fields to secure the passwords from hit and trial errors. Also in the future providing the users with the power to log in and register using their social media handles like Facebook, Linkedin, google, etc. Finally, a feature of forgot password is extremely important and can be implemented so that users can reassess their accounts. A lot of effort has been put into making the application fast and efficient by reducing the number of API calls being made and using the web browser's ability to store data within itself. This can

still be further improved by researching adequate ways to get the tasks done in the best way possible. Work can also be put in the React components to make them faster and efficient. Currently, no tests were performed to measure the complexity and render time of the components in the front end. This application gives a push towards making itself scalable by introducing custom sections that can be added by the admin. Currently, the functionalities linked to all the other sections weren't implemented because of the complexity of its design. More efforts into this module will mean it would be possible for the admins and then eventually users to add new sections to the resumes making it completely customizable. This application currently only includes three templates that are professional-looking. A great future addition would be to have a lot more options for the user to work with. This would include some templates for students in creative fields helping them build resumes customized for their needs and taste.

7. Conclusions

This application serves the purpose to help students at universities to design and create their resumes with ease using forms and profiles that we all are so used to in terms of social media. The development of this application has taught me a lot about the application development process ranging from planning, requirements analysis to the deployment of the project. Planning and understanding of the requirements were some of the most challenging aspects of the project after the build phase itself. Resumes are a crucial part of who a student is professionally and having them curated to their needs is difficult. Thus a standard format is considered for the sake of this project so as to perfectly fit the needs of the companies job postings. Further, a lot of effort was put into developing forms that can be multiplicated dynamically for the user to insert multiple entries within one section. A good amount of research and input from the Professors helped me tackle this problem easily. The application uses a lot of technologies that I had some expertise on and some of which I learned focusing on this project. Developing the front end and its integration with the amount of data that comes in was a difficult task. Further implementing functionalities like reorganizing the sections within the resume, visibility of the sections and individual sections was the most challenging part. This utilized the core concept of React states, Redux and Hooks, and made me understand these concepts in a much better way. Developing the User Interface I had in my mind for each component including the functionalities was a very complicated task. This was thus done using the Semantic UI framework that keeps a large set of components pre-built which can be smoothly and efficiently integrated within React. This project also helped me improve my research skills a lot as this is a relatively unexplored topic when it comes to scholarly articles and journals. Thus to better understand this project I had a lot of communication with the students around me to understand their point of view on this project and to plan the best way to help them out. The Internet became my best resource providing me with tonnes of information on the various aspects of the project from the design of the resume to how to make the website user-friendly. Further, this project made me realize that the journey of becoming a better software developer is a cycle of finding a solution to do a task, realizing you were wrong, and reiterating the whole process to bring out the best possible solution for the users. Overall this project has been a perfect portal for me to portray software developer.

8. References

[1] I. Wu, K. Wayne, S. Lakka, and T. Rai, “Pro-Resume: The Infographic Resume Builder,” [2] K. D. Ingale, “A Review Paper on Resume Portal,” p. 5, 2018. [3] S. Risavy, “The Resume Research Literature: Where Have We Been and Where Should We Go Next?,” *Journal of Educational and Developmental Psychology*, vol. 7, p. 169, Feb. 2017, DOI: 10.5539/jeep.v7n1p169. [4] “How to Beat an Applicant Tracking System (ATS) with a 100% Pass Rate.”

https://www.visualcv.com/blog/how-to-beat-the-applicant-tracking-system/(accessed dec. 08, 2024).

[5] “Resume Statistics [2024]: What Job Seekers Need To Know – Zippia.” https://www.zippia.com/advice/resume-statistics/. [6] A. Tiwari, S. Vaghela, R. Nagar, and M. Desai, “Applicant Tracking and Scoring System,” vol. 06, no. 04, p. 6, 2019. [7] C. Martin-Lacroux and A. Lacroux, “Do Employers Forgive Applicants’ Bad Spelling in Résumés?,” *Business and Professional Communication Quarterly*, vol. 80, no. 3, pp. 321–335, Sep. 2017, DOI:10.1177/2329490616671310.

[8] “Cover Letters and Resumes | Career Services and Co-op | RIT.” https://www.rit.edu/careerservices/students/job-search/cover-letters-and-resumes#resumes (accessed Jul. 21, 2024).

[9] P. W. Lab, “Résumé Design // Purdue Writing Lab,” *Purdue Writing Lab*. https://owl.purdue.edu/owl/job_search_writing/resumes_and_vitas/resume_design.html (accessed Jul. 21, 2024).

[10] *You have 7.4 seconds to make an impression: How recruiters see your resume*. (n.d.). Ladders | Business News & Career Advice. Retrieved December 5, 2024, from https://www.theladders.com:443/career-advice/you-only-get-6-seconds-of-fame-make-it-count

[11] Malewade, S. M., & Ekbote, A. (n.d.). Performance Optimization using MERN stack on Web Application.*International Journal of EngineeringResearch*,*10*(06), 5.

[12] *Getting Started with Redux | Redux*. (n.d.). Retrieved December 5, 2024, from https://redux.js.org/introduction/getting-started

[13] *Getting Started—JSON Resume*. (n.d.). Retrieved December 5, 2024, from https://jsonresume.org/getting-started