**AI Resume Builder — Technical Report**

Group Project 4 - Week 4

25 June 2025

Page 1: Architecture Decisions and Technology Stack

## 1. Overview

The AI Resume Builder is a front-end-focused web application with integrated Open-AI API functionality that assists users in generating high-quality resume content dynamically. It enables users to input personal information, customize fonts and themes, and generate AI-enhanced resume text, which can be exported as a PDF.

## 2. Architecture Decisions

Hybrid Client-Side Architecture:

* - Uses static HTML/CSS/JavaScript for form rendering, content preview, and user input handling.  
  - Open-AI's API is invoked from the client using fetch or similar methods to dynamically generate professional content.

No Back-end Server (Optional):

* - The current implementation runs entirely on the front-end. However, a production deployment would ideally use a back-end proxy to protect API keys.

Separation of Concerns:

* - src/ for editable development files  
  - dist/ for deployed/compiled files

## 3. Technology Stack

|  |  |  |
| --- | --- | --- |
| Layer | Tool/Library | Purpose |
| Markup | HTML5 | Form layout and structure |
| Styling | CSS3 | Custom fonts, colors, and responsive UI |
| Scripting | JavaScript (ES6) | Event handling, DOM updates, API calls |
| AI Engine | OpenAI API | Generating dynamic, natural-language content |
| PDF Export | jsPDF | Exporting resume to downloadable format |

Page 2: OpenAI Integration, Template Design & Optimization

## 4. API Integration Methodology

The application uses Open-AI's text-davinci-003 or gpt-3.5-turbo models for natural language generation. Users provide minimal inputs (e.g., job title, skills), and the app sends a prompt to the Open-AI API such as:

"Write a professional summary for a Front-end Developer with experience in HTML, CSS, JavaScript, and React."

The integration is typically performed via a fetch() request:

fetch("https://api.openai.com/v1/completions", {  
 method: "POST",  
 headers: {  
 "Content-Type": "application/json",  
 "Authorization": "Bearer YOUR\_API\_KEY"  
 },  
 body: JSON.stringify({  
 prompt: userPrompt,  
 model: "text-davinci-003",  
 max\_tokens: 150,  
 temperature: 0.7  
 })  
});

## 5. Template Design Approach

* - Resume templates are generated using JavaScript template literals with embedded user or AI-provided data.  
  - Supports custom fonts via <select> input, theme colors, and dynamic sections.  
  - Preview is rendered inside a resume-preview container.  
  - Users can modify, regenerate, and export the resume easily.

## 6. Performance Optimization Techniques

* - Debounced Input Fields: Reduces unnecessary re-renders or API calls.  
  - A sync/Await + Try/Catch: Prevents UI crashes during failed requests.  
  - Single-Pass Rendering: The DOM is updated once per resume generation.  
  - Lightweight Libraries: Only PDF is used externally to keep load time minimal.  
  - Scalable Templates: Easy to add new styles or resume types without rewriting core logic.

# Page 3: Limitations, Enhancements & Sample Outputs

## 7. Known Limitations

* - API Key Exposure: If the key is stored in front-end JavaScript, it’s insecure for production.  
  - Limited Template Variety: Only one HTML structure used.  
  - No Login/Storage: Users cannot save drafts or access past resumes.  
  - Token Limits: Long prompts or resumes may be truncated by Open-AI.

## 8. Future Enhancements

* - Implement a Node.js or Flask back-end to securely manage the Open-AI API key.  
  - Add multiple visual templates and layout styles.  
  - Allow resume comparison based on different job descriptions.  
  - Enable cloud saving and user accounts.  
  - Introduce feedback rating for AI-generated content to improve accuracy.  
  - Add export to DOCX or integration with Google Docs.

## 9. Sample AI-Generated Outputs

Example 1: AI-Generated Summary

“Creative and detail-oriented Front-end Developer with 3+ years of experience building responsive web applications using React, HTML, CSS, and JavaScript. Adept at collaborating in agile teams and delivering user-centric solutions.”

Example 2: AI-Generated Experience Entry

“Led the redesign of a corporate website, improving page speed by 40% and increasing user engagement through responsive, accessible design principles.”

## Evaluation Criteria Table

|  |  |
| --- | --- |
| Evaluation Criterion | Implementation Status |
| User Experience Intuitiveness | Clear, interactive interface with real-time feedback |
| Output Quality & Customization | High: AI-generated summaries, theme/font options |
| Technical Implementation | Robust for MVP; security considerations noted |
| Documentation Completeness | README and this report provide full coverage |
| Innovation in AI Application | Smart prompt use, real-world use case |