**Smart resume builder**

|  |  |
| --- | --- |
| Name | Surname |
| Tebelelo | **Lekoana** |
| Hartton | **Malau** |
| Kagiso | **Monene** |
| Babyface | **Mokoena** |
| Morokolo | **Chueu** |
| Nelisiwe | **Kaziwa** |

**Technical Report: Smart Resume Builder**

# Table of Contents

1. Architecture and Technology Stack

2. User Experience (UX) and Design Philosophy

3. API Integration Methodology

4. Project Flow Diagram

5. Performance Optimization Techniques

6. Known Limitations and Future Enhancements

7. Sample Outputs

8. User Guide and Feature Walkthrough

# 1. Architecture and Technology Stack

The Smart Resume Builder is designed as a modern single-page web application (SPA) focused on speed and simplicity. The core architecture is client-side only, requiring no backend server for its core functionality, minimizing complexity and deployment overhead.

**Technology Stack:**

* HTML5 – Defines structure, interactive forms, and display areas.
* CSS3 (Tailwind CSS) – Utility-first styling for rapid, responsive UI development directly in markup.
* JavaScript (ES6+) – Handles DOM manipulation, event handling, and API communication in vanilla JS (no frameworks for minimal bundle size).
* CDN Libraries:
* Lucide Icons – Lightweight, customizable icon set.
* HTML-DOCX – Converts HTML content into .docx.
* html2pdf.js – Client-side PDF generation.

# 2. User Experience (UX) and Design Philosophy

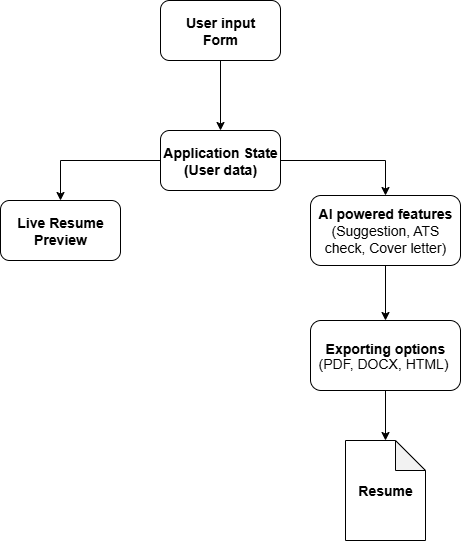
* **Real-time Feedback** – Resume preview updates instantly as the user types.
* **Modular Modals** – ATS check, cover letter generation, etc., appear in modal popups (no navigation away).
* **Responsive Layout** – Small screens: forms stack vertically; Large screens: side-by-side (forms + live preview).

# 3. API Integration Methodology

**The AI-powered features rely on the Gemini API.**

* **Client-Side Communication** – Fetch API sends requests directly (no server needed).
* **Request Payloads** – JSON includes Prompt (user data, job description), generationConfig (e.g., JSON output schema), Model name: **gemini-2.5-flash-preview-05-20.**
* **Async Handling** – async/await ensures smooth UI with loading indicators.
* **Error Handling** – try/catch with descriptive error modal to avoid crashes.

# 4. Project Flow Diagram



# 5. Performance Optimization Techniques

* **Minimalist Design** – Loads only essential assets.
* **CDN Libraries** – Faster load via caching/distribution.
* **Client-Side Logic** – Scales efficiently across many users.
* **Efficient DOM Updates** – Only modified elements re-render.
* **Debouncing** – Reduces unnecessary API calls while typing.

# 6. Known Limitations and Future Enhancements

* No Data Persistence – Refresh clears all input (future: cloud storage).
* Limited Templates – More can be added easily.
* No Offline Mode – Requires internet for AI features.
* Real-time Collaboration – Possible with Firestore real-time listeners.

# 7. Sample Outputs

A screenshot of a chat

AI-generated content may be incorrect.

* **Modern Template** – Two-column, clean layout with color accents. Large, centered name, icons for bullet points. Best for creative/tech roles.
* **Classic Template** – Single-column, text-heavy, formal design. Bold/underlined headings. Best for academic, legal, or formal industries.
* **Professional Template** – Balanced, modern single-column design. Strong headings + hierarchy. Versatile for corporate/professional fields.

# 8. User Guide and Feature Walkthrough

**Getting Started:**

1. Fill forms (Personal Info, Summary, Experience, etc.).
2. Preview updates live on the right.
3. Switch templates (Modern, Classic, Professional).

**AI Features:**

* **AI Suggestions** – Improved bullet points, summaries.
* **Cover Letter Generator** – Auto-drafts from your resume.
* **ATS Check** – Finds formatting/keyword issues.
* **ATS Job Match** – Analyzes resume vs. job description.

**Export Options:**

* PDF – Standard job application format.
* DOCX – Editable Word file.
* HTML – Shareable self-contained resume.