

AI Resume Generator – Project

Documentation

Project Overview

The AI Resume Generator is a web-based application designed to help job seekers to generate or edit resumes using artificial intelligence. By leveraging user input and OpenAI language processing capabilities, the tool provides formatted resumes suited to numerous job roles and markets.

Team Members

- Blake Schneringer
- Semen Kolesnykov
- Deris A Leon Carrillo
- Archer Amon
- Enrique Dominguez

Problem Statement

- Job seekers often struggle to create polished, tailored resumes efficiently.

- o Translating work experience into resume suited language is often difficult.
 - o Traditional resume tools lack automation, intelligent feedback, or adaptability to different job markets and user profiles.
- Our approach: create a smooth user interface for resume input integrated with AI to polish language

Goals and Features

- Smooth, guided resume creation process.
- AI-generated, professional, and job-tailored content.
- Multiple generation modes: from scratch from existing resume
- Real-time preview and editing.
- Export as PDF for easy sharing.

System Architecture:

Frontend

- **Technology:** React.js
- **Components:**
 - o Input Forms

- o Resume Preview Panel
- o PDF export Button

Backend

- o **AWS Lambda:** Handles logic for receiving request from frontend and sending requests to OpenAI API.
- o **API Gateway:** Exposes RESTful endpoints to the frontend. Handles routing to Lambda and request validation.
- o **AI integration:** Receives structured user data as a string and returns the resume in HTML content.

User Stories:

1. **Resume input:** as a user, I want to answer a few questions that the AI asks me, such as personal details, education, experience, and skills, so that the AI can accurately generate a detailed resume about me.
2. **Resume generation:** as a user, I want to choose between generating a resume from scratch, generating a resume from a previous resume, or from a job post.
3. **AI-Enhanced Content:** as a user, I want the AI to write role-specific resume content, so that my resume is professional and tailored to the job.
4. **Export and sharing:** as a user, I want to export my resume to PDF, so that I can easily share it with recruiters or upload it to job sites.

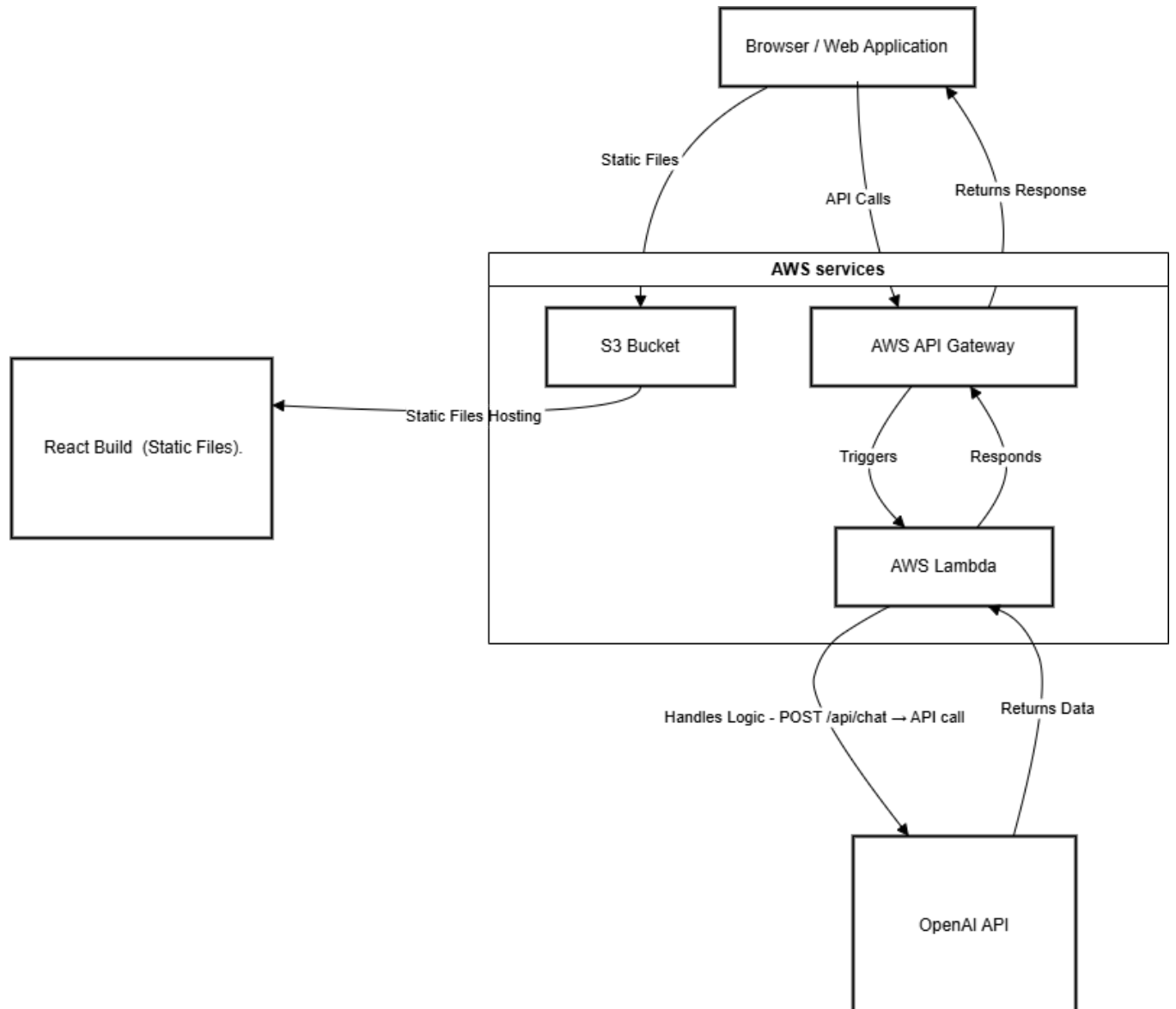
Development Process

- o **Agile Methodology**

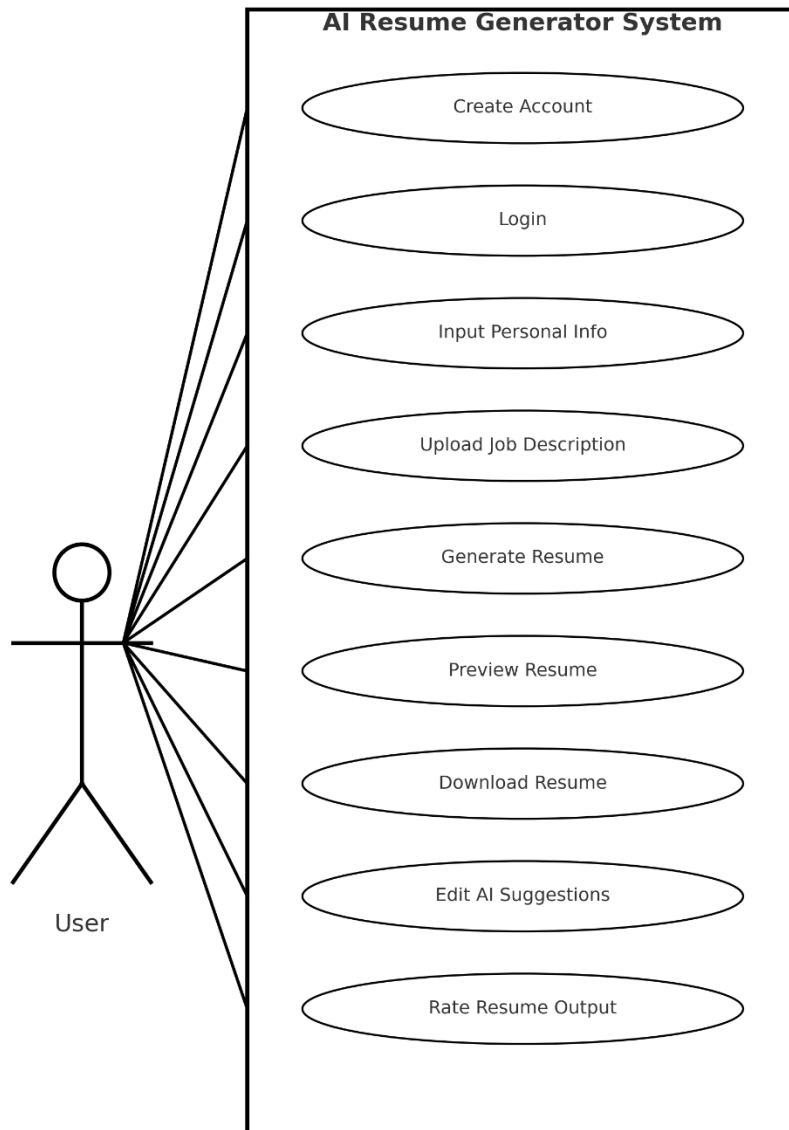
- o 5 sprint cycles
- o Early sprints: planning, prototyping
- o Middle sprints: frontend/backend development
- o Late sprints: polish UI/UX, add editing, export options

Diagrams

Back-End Structure:



Use cases:



Test Cases:

- Test Case 1: Resume Generation via OpenAI API
 - Input: Valid user profile data
 - Expected: Lambda Code sends structured prompt to OpenAI, and React displays a well-formatted AI-generated resume
- Test Case 2: Frontend Form Validation
 - Input: Empty or malformed fields
 - Expected: React prevents submission and shows appropriate error messages and no API call is made
- Test Case 3: AI Prompt Injection Handling
 - Input: User tries injecting a prompt trick
 - Expected: Flask sanitizes the input or OpenAI's response is checked before displaying
- Test Case 4: Resume Template Rendering
 - Input: User selects a resume style and or template
 - Expected: React re-renders resume using the correct styling logic without a full page reload
- Test Case 5: Backend API Timeout or Failure
 - Input: OpenAI API is unresponsive or errors

- o Expected: Lambda handles the error and sends a clear message to React; React shows retry option
- Test Case 6: PDF Export Function
 - o Input: User clicks “Download as PDF”
 - o Expected: React exports current resume content to a correctly formatted downloadable PDF
- Test Case 7: Save & Retrieve Resume
 - o Input: User saves a resume draft and later returns
 - o Expected: Lambda stores resume in a session, and then react loads saved content properly on return
- Test Case 8: Mobile Responsiveness
 - o Input: Opens website/webapp on a mobile device
 - o Expected: React layout adapts properly, form inputs, and buttons are usable on smaller screens

Test Suites:

Test Suite 1: User Authentication

Test Case 1: Successful login

Steps: Enter valid credentials and log in

Expected Result: User successfully logs in and sees dashboard

Test Case 2: Failed login (wrong password)

Steps: Enter invalid credentials

Expected Result: Error message displayed

Test Case 3: Registration with valid data

Steps: Fill out registration form completely

Expected Result: Account created, redirected to dashboard

Test Case 4: Registration with missing fields

Steps: Leave required field empty

Expected Result: Validation error displayed

Test Suite 2: Resume Input & Generation

Test Case 1: Upload job description

Steps: Upload a valid job description file

Expected Result: File is accepted and processed

Test Case 2: Enter personal info

Steps: Fill out personal details (name, experience, etc.)

Expected Result: Data saved and confirmed

Test Case 3: Generate resume with complete data

Steps: Click 'Generate Resume' with all fields filled

Expected Result: AI-generated resume displayed

Test Case 4: Generate resume with missing data

Steps: Leave key fields empty and generate

Expected Result: Validation error or prompt to fill missing data

Test Suite 3: Resume Preview & Download

Test Case 1: Preview resume

Steps: Click 'Preview' after generating resume

Expected Result: Resume preview shown

Test Case 2: Download resume as PDF

Steps: Click 'Download PDF' button

Expected Result: PDF downloaded successfully

Test Case 3: Edit AI suggestions

Steps: Edit generated text and save changes

Expected Result: Changes saved and reflected

Test Suite 4: Feedback (if implemented)

Test Case 1: Submit feedback on generated resume

Steps: Leave feedback for AI suggestions

Expected Result: Feedback saved and acknowledged

Test Case 2: View feedback summary

Steps: Navigate to feedback summary

Expected Result: Feedback list displayed