

Software Design Document

– A Template –

1. Introduction and Purpose

This SDD documents the design of a resume evaluation system powered by AI, an application that analyzes and rates users' resumes based on provided job descriptions. The system generates a rating based on how well the resume matches the job requirements. Refer to the SRD for the related requirements that drive this system design.

1.1 Acronyms, Terms and References

Acronym	Meaning
HDD	High-Level Design
LLD	Low-Level Design
SDD	Software Design Document
SRD	Software Requirements Document
SUD	System Under Development
SWEBOK	Software Engineering Body of Knowledge
OOP	Object-Oriented Programming
FP	Functional Programming
API	Application Programming Interface
CI/CD	Continuous integration and Continuous Deployment
LLM	Large Language Model
AI	Artificial Intelligence
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extension
IP	Internet Protocol
UI	User Interface

SE 300

CSS	Cascading Style Sheets
GPT	Generative Pre-trained Transformer

Term	Definition
Frontend	Web user interface program
Backend	Server side/database program
Strategy pattern	Encapsulates a family of algorithms/ behaviors in separate classes made interchangeable
Builder pattern	Separates construction of complex object from its representation providing a flexible solution
Docker	Platform that allows you to build, test, and deploy applications
Unit test	Testing individual components of source code to ensure functional
Integration test	Testing multiple components together to verify they work together
Parsing	Analyzing text or strings and converting to structured format
Environment Variable	System-level variable used to store configured data
Error Handling	Code used to detect and respond to errors

2 Design Considerations

2.1 Assumptions and Dependencies

- The SUD will operate in a Linux environment (Debian)
- The SUD will depend only on OpenAI's API for LLM integration.
[SRD_FuncReq_0019]
- The SUD will be deployable via Docker.

SE 300

- The SUD will work concurrently (when possible)

2.2 Design Constraints

This SUD should rely only on the OpenAI API [SRD_FuncReq_0019], but future support could implement connections to other, more affordable APIs. Furthermore, this application should be deployable to our own custom hardware [SRD_ExtSysReq_0001] and containerized via Docker [SRD_ExtSysReq_0002].

2.3 Goals and Guidelines

- Modularity via abstraction and interfacing (the strategy pattern) [SRD_DesignConstraint_0001].
- Easy to maintain and debug.
- Simple feature usage. I.e., a developer with little (or no) experience with Go should be able to understand what is going on [SRD_DesignConstraint_0002].
- Modules should be loosely coupled or totally independent when possible.
- Any “complex” object/class (types) will be created using the builder pattern.
- Code should be testable via unit tests as well as integration tests.
- Deployment should be automated via CI/CD.

2.4 Development Methods

During the development of the SUD, we will follow an iterative approach. Furthermore, this project will employ a mix of object-oriented design as well as some basic functional design. The SUD will be written in a language which does not conform to either style exclusively. Go [SRD_DesignConstraint_0002] is a “multi-paradigm” language with a strong focus on imperative design with small pieces inherited from OOP and FP.

2 High-Level Design (HLD)

3.1 System Overview

The software will be composed of 4 major components: The User, the AI Grading Inputs, the OpenAI engine, and the Output. Data will flow sequentially from the

SE 300

user with the grading inputs into the AI with the results returning to the user. This path is shown in Figure 1.

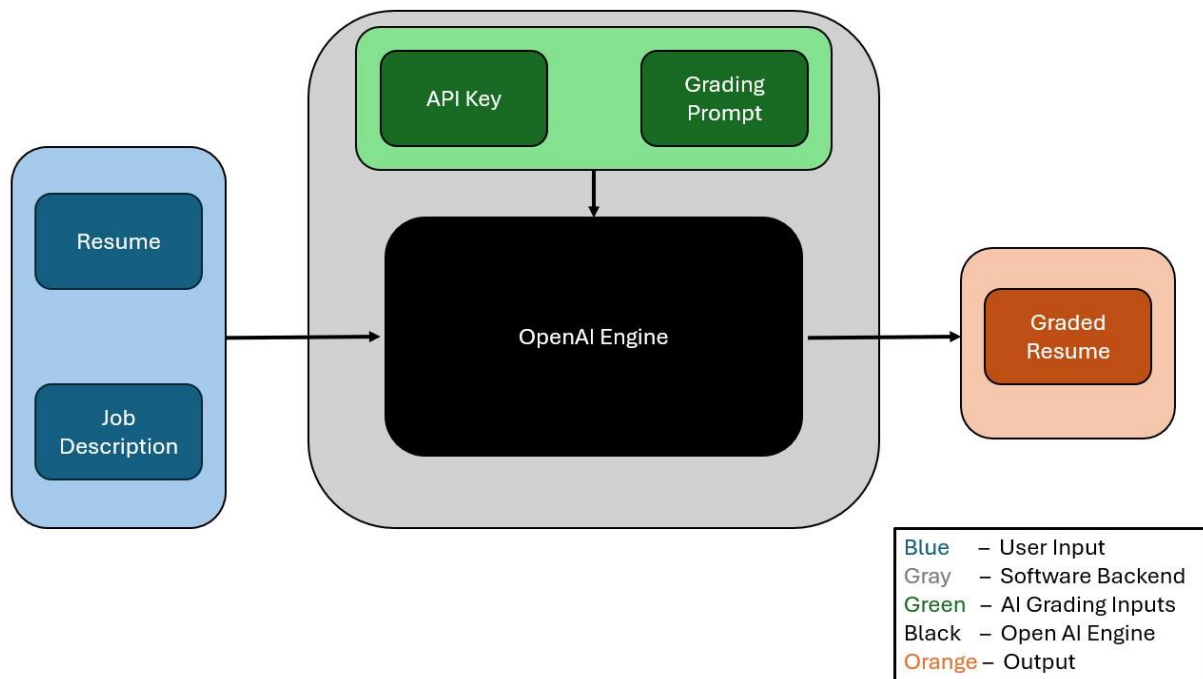


Figure 1. AI Resume Grade Top-Level Function View

Legend for Figure 1:

- Large boxes are each component with their respective subsections as described in the image.
- Only the inputs and outputs will be accessible by the user with the software backend being hidden.

SE 300

3.2 User Input

This object is composed of 2 sections: Resume and Job Description

Resume

This objective has three main responsibilities:

SDD_HLD_0001

- Accept only PDF formatting for input resumes.

SDD_HLD_0002

- Read all pages of the input PDF resume.

SDD_HLD_0003

- Read all words on the accepted pages of the PDF resume input transferring into string.

Job Description

This objective has one main responsibilities:

SDD_HLD_0004

- Read string of text from text box.

Note:

- All responsibility has respective errors handling if responsibility is not met and system will not proceed.
- Both inputs must be complete, correct, and accepted upon moving to the AI.

SE 300

3.3 AI Grading Inputs

This object is composed of 2 sections: API Key and Grading Prompt

API Key

This objective has three main responsibilities:

SDD_HLD_0015

- Authenticate requests to AI service through credential.

SDD_HLD_0006

- Function only within call rate limit of 10 calls per hour. Call limits are placed within specified time limit to minimize cost of system.

SDD_HLD_0007

- Remain hidden and never expose to the user.

Grading Prompt

This objective has one main responsibilities:

SDD_HLD_0016

- Read from embedded grading text string.

Note:

- All responsibility has respective errors handling if responsibility is not met and system will not proceed.
- Both inputs must be accepted before moving to the AI engine.
- Will be in software backend inaccessible to user.

SE 300

3.4 OpenAI Engine

This objective has four main responsibilities:

SDD_HLD_0015

- Accept API Key to run system.

SDD_HLD_0005

- Accept string inputs from Resume, Job Description, and Grading Prompt.

SDD_HLD_0008

- Produce string text output of graded resume to given job description following prompt.

SDD_HLD_0013

- Run until output string is produced so long as output is created within allotted run time of two minutes.

Note:

- All responsibility has respective error handling if responsibility is not met and system will not proceed.
- OpenAI Engine is a “black box” with only the known inputs being controlled variables.
- Will be in software backend inaccessible to user.

3.5 System Output

This objective has two main responsibilities:

SDD_HLD_0009

- Decompose the large output into job description criteria-based subsections.

SDD_HLD_0010

- Generate graded subsection categories on scale from 1-10.

SDD_HLD_0011

- Generate strengths and weaknesses found in resumes when compared to job description.

SDD_HLD_0012

- Generate grammar and spelling score in addition to correction for the resume.

SDD_HLD_0014

- Display results of output to the user through UI

Note:

- All responsibility has respective errors handling if responsibility is not met and system will not proceed.
- Final output of the system being accessible to the user after the input screen

3 Low-Level Design (LLD)

The low-level design refines the major components that are defined by the High-Level Design components into specific modules, structs, functions, data structures, and algorithms. It provides quality details to allow for direct implementation, unit testing, and integration testing while respecting the constraints that the design establishes. All of the modules prioritize loose coupling, testability, and deterministic behavior where it is allowed.

4.1 PDF Resume Processing Module

This object has three main responsibilities:

SDD_LLD_0001

- Confirm file extensions and MIME types to ensure that only PDF files are processed, rejecting all other formats.

SDD_LLD_0003

- Iteratively read through every page of the document that's uploaded, in order to extract and concatenate all readable text into a singular string.

SDD_LLD_0005

- Provide the HTTP handler and endpoint that are required to accept multipart/form data uploads from the frontend.

Note:

- Package: internal/pdf
- Uses a lightweight external library for extraction to help minimize system memory overload.
- Returns a "400 Bad Requests" status immediately if file validation or extraction fails.

4.2 Prompt Construction and Injection Defense Module

This object has three main responsibilities:

SDD_LLD_0006

SE 300

- Creates a high-priority system instruction set that then defines the AI's persona and core grading criteria.

SDD_LLD_0007

- Merges user-provided resume text and job rubric into a single query using structured delimiters.

SDD_LLD_0008

- Performs keyword and heuristic scans on input text to identify commands intended to override the system and instructions.

SDD_LLD_0009

- Include a specific security Boolean in the output object to alert the system and user of neutralized injection attempts.

Note:

- Package: internal/prompt
- Uses the Builder pattern to assemble immutable prompt objects.
- Employs the triple-hash (###) delimiters to help the AI distinguish between instructions and user-provided data.

4.3 OpenAI API Client and Rate Limiting Module

This object has five main responsibilities:

SDD_LLD_0010

- Verifies requests using credentials loaded solely from environment variables to prevent key exposure.

SDD_LLD_0011

- Restricts request frequency to ten calls per hour per unique source IP address using a token bucket algorithm.

SDD_LLD_0012

- Submit the final assembled prompt to the OpenAI completions endpoint via a secure HTTPS POST request.

SDD_LLD_0013

- Catches and translates external API errors (492, 500) into user friendly messages for the frontend.

SE 300

SDD_LLD_0014

- Cancels any pending request that exceeds a two minute processing duration using context timeouts.

Note:

- Package: internal/ai
- Rate limiting is implemented at the middleware level to reject unauthorized traffic before calling external services.
- Timeouts ensure system resources are released if the AI model becomes unresponsive.

4.4 Output Parsing and Structuring Module

This object has five main responsibilities:

SDD_LLD_0015

- Extracts the total score and summary of the feedback from the raw AI response string into a typed data object.

SDD_LLD_0016

- Maps individual criteria scores to specific data fields and clamps numeric ratings to the previously described 1-100 scale.

SDD_LLD_0017

- Converts AI-generated text blocks into formatted arrays for display in the feedback UI.

SDD_LLD_0018

- Identifies specific grammar feedback and spelling ratings to populate the writing quality report card.

SDD_LLD_0019

- Marshals the completed evaluation data structure into a formatted JSON file for local user storage.

Note:

- Package: internal/output
- Parsing logic includes data normalization to correct any AI responses that fall outside expected numeric bounds.
- Structured output is prioritized via JSON response formatting requested in the AI prompt.

SE 300

4.5 Frontend Component Structure

This object has three main responsibilities:

SDD_LLD_0020

- Provides a responsive file-picker and text input area for users to submit their evaluation materials.

SDD_LLD_0023

- Renders the final percentage score and criteria ratings using prominent, color-coded, visualization components.

SDD_LLD_0027

- Manages a global “loading” state to show a progress spinner and disable submission buttons during active API calls

Note:

- Built using React with state managed via standard hooks
- Uses vanilla CSS for layout to ensure the application remains lightweight and fast-loading.

4 Trace Tags

This section lists the trace tags in this document.

5.1 High-Level Design

High-Level Design Trace Tags

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_HLD_0001	Accept PDF resume input	SRD_FuncReq_0001
SDD_HLD_0002	Extract full textual content from resume PDF	SRD_FuncReq_0003
SDD_HLD_0003	Convert resume contents into string format for AI processing	SRD_FuncReq_0013
SDD_HLD_0004	Accept job description in textbox	SRD_FuncReq_0002
SDD_HLD_0005	Provide structured inputs to AI grader	SRD_FuncReq_0004
SDD_HLD_0006	Enforce AI request rate limit (10/hour)	SRD_FuncReq_0014
SDD_HLD_0007	Securely manage API credentials	SRD_SecReq_0001
SDD_HLD_0008	Generate graded evaluation output	SRD_FuncReq_0005

SE 300

SDD_HLD_0009	Produce criteria-based subdivision	SRD_FuncReq_0006
SDD_HLD_0010	Produce subdivision scoring (0–10 scale)	SRD_FuncReq_0007
SDD_HLD_0011	Generate strengths and weaknesses feedback	SRD_FuncReq_0008, SRD_FuncReq_0009
SDD_HLD_0012	Generate list of grammar/spelling score and suggestions	SRD_FuncReq_0010, SRD_FuncReq_0011
SDD_HLD_0013	Enforce runtime constraint (≤ 2 minutes)	SRD_FuncReq_0020
SDD_HLD_0014	Display results through UI interface	SRD_InterfaceReq_0005–0010
SDD_HLD_0015	Store an updated and acceptable API key	SRD_FuncReq_0021
SDD_HLD_0016	Store an readable string grading rubric for AI	SRD_FuncReq_0022

5.2 Low-Level Design

PDF Resume Processing Module

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_LLD_0001	Accept only PDF uploads	SRD_FuncReq_0001
SDD_LLD_0002	Reject non-PDF uploads	SRD_FuncReq_0012
SDD_LLD_0003	Extract full text from PDF pages	SRD_FuncReq_0003
SDD_LLD_0004	Ignore formatting/layout during extraction	SRD_FuncReq_0013
SDD_LLD_0005	Integrate with upload interface	SRD_InterfaceReq_0001

Prompt Construction and Injection Defense Module

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_LLD_0006	Construct system prompt for grading logic	SRD_FuncReq_0004
SDD_LLD_0007	Combine resume + job description inputs	SRD_FuncReq_0004
SDD_LLD_0008	Detect prompt injection attempts	SRD_NonFuncReq_0010
SDD_LLD_0009	Flag injection in structured output	SRD_QualAssurReq_0005

OpenAI API Client and Rate Limiting Module

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_LLD_0010	Securely load/store API key	SRD_SecReq_0001
SDD_LLD_0011	Enforce per-IP rate limit (10/hr)	SRD_SecReq_0004, SRD_SecReq_0006
SDD_LLD_0012	Execute OpenAI GPT-5 Mini requests	SRD_FuncReq_0019
SDD_LLD_0013	Handle API failures and error responses	SRD_FuncReq_0015
SDD_LLD_0014	Enforce execution timeout	SRD_FuncReq_0020

Output Parsing and Structuring Module

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_LLD_0015	Parse AI output into structured fields	SRD_FuncReq_0005
SDD_LLD_0016	Parse criteria subdivisions and ratings	SRD_FuncReq_0006, SRD_FuncReq_0007
SDD_LLD_0017	Parse strengths and weaknesses lists	SRD_FuncReq_0008, SRD_FuncReq_0009
SDD_LLD_0018	Parse grammar/spelling evaluation	SRD_FuncReq_0010, SRD_FuncReq_0011
SDD_LLD_0019	Support JSON download output	SRD_FuncReq_0018

Frontend Component Structure

SDD Trace Tag	Description	Traces To SRD Requirement
SDD_LLD_0020	Upload interface supports resume input	SRD_InterfaceReq_0001
SDD_LLD_0021	Accept textual job description input	SRD_InterfaceReq_0003
SDD_LLD_0022	Display resume preview	SRD_InterfaceReq_0004
SDD_LLD_0023	Display numeric score output	SRD_InterfaceReq_0005
SDD_LLD_0024	Display criteria breakdown results	SRD_InterfaceReq_0006
SDD_LLD_0025	Display summary, strengths, weaknesses	SRD_InterfaceReq_0007–0009
SDD_LLD_0026	Display grammar rating	SRD_InterfaceReq_0010
SDD_LLD_0027	Display loading indicator during processing	SRD_InterfaceReq_0011

SE 300

SDD_LLD_0028	Provide navigation across UI pages	SRD_InterfaceReq_0012, SRD_InterfaceReq_0013
SDD_LLD_0029	Display static demo output page	SRD_InterfaceReq_0014
SDD_LLD_0030	Display persistent error messages	SRD_InterfaceReq_0015

Appendix

Design Analysis & Review

Expanded high level design description of output to include specific scoring scale and some of the specific generated feedback given (strengths, weaknesses, grammar, spelling). Additionally, we wanted to especially note that the output will be displayed to the user with its own UI which gets later explained in LLD.

While developing the software design, we identified gaps in the SRD as many requirements lacked depth, while others did not fully cover all design aspects being considered. To ensure that all design elements have requirements and testable baselines, we created additional requirements in the SRD.

In the initial draft, section 2 did not contain any trace tags. During the review process, trace tags were added where necessary. Some of these were slightly repetitive, so they were only added in places where they made the most sense, not just places where keywords were used.