Prompt:

Suggest 3 or more significant and interesting CV analysis features. The features suggested should have the following attribues where Python will be the programming language for the implementation:

\* The implementation of the feature should be well suited for depending heavily on feature engineering with Gemini API free version.

\* The implementation of the feature should be well suited for using at least one well known third party library for a significant part of the implementation.

\* The implementation should have its complexity focused in prompt engineering.

\* The implementation should not include API services that are not free or demand registration in a any form except for Gemini API free version.

Response:

I'll suggest features that focus on prompt engineering with Gemini while leveraging free libraries, avoiding any paid/registration services:

1. CV-to-Role Match Analyzer

* Core Feature: Analyze how well a CV matches a target role description
* Prompt Engineering Focus:
  + Multi-step prompting to first extract CV components
  + Chain-of-thought prompts to analyze each section against role requirements
  + Structured output prompts for generating specific improvement recommendations
* Key Third-Party Libraries:
  + PyPDF2 for PDF extraction
  + pandas for structured data organization
* Complex Prompt Aspects:
  + Role requirement identification
  + Skill mapping and gap analysis
  + Experience relevance scoring
  + Custom rubric application through prompting

1. Achievement Statement Enhancer

* Core Feature: Analyze and improve the impact statements in a CV
* Prompt Engineering Focus:
  + Iterative prompting to identify achievement statements
  + Context-aware prompts for suggesting quantifiable metrics
  + Comparative analysis prompts to evaluate before/after impact
* Key Third-Party Libraries:
  + PyPDF2 for PDF handling
  + rich for formatted console output
* Complex Prompt Aspects:
  + Achievement pattern recognition
  + Impact metric extraction
  + Industry-specific language adaptation
  + STAR format verification and enhancement

1. Experience Progression Analyzer

* Core Feature: Analyze career progression logic and identify potential red flags
* Prompt Engineering Focus:
  + Sequential prompting for timeline analysis
  + Role comparison prompts for progression logic
  + Gap analysis through structured prompting
* Key Third-Party Libraries:
  + PyPDF2 for PDF processing
  + matplotlib for visualization
* Complex Prompt Aspects:
  + Career trajectory pattern recognition
  + Skill evolution tracking
  + Role transition analysis
  + Industry-standard progression comparison

Each feature emphasizes:

1. Heavy prompt engineering focus
2. Uses only Gemini's free API
3. Leverages common, free Python libraries
4. Requires no external API services
5. No registration requirements beyond Gemini
6. Clear scope for implementation
7. Produces meaningful output

Would you like me to detail the prompt engineering complexity for any of these features? I can break down the prompt chain structure and explain how it would work with Gemini's capabilities.

Prompt:

Define the requirements for the following feature: CV-to-Role Match Analyzer with the following details:

\* Core Feature: Analyze how well a CV matches a target role description

\* Prompt Engineering Focus:

\* Multi-step prompting to first extract CV components

\* Chain-of-thought prompts to analyze each section against role requirements

\* Structured output prompts for generating specific improvement recommendations

\* Key Third-Party Libraries:

\* PyPDF2 for PDF extraction

\* pandas for structured data organization

\* Complex Prompt Aspects:

\* Role requirement identification

\* Skill mapping and gap analysis

\* Experience relevance scoring

\* Custom rubric application through prompting

The requirements should adhere to the following principles:

\* The requirements should include both functional and non-functional requirements

\* Include a wide range of the ISO 25010 software quality model, stating for each requirement the class it belongs to and the category inside the class.

\* All requiremnts must be clearly testable.

\* Completeness - all services required by the user should be defined.

\* Consistency - requirements should not have contradictory definitions.

\* Include the acceptance criterea, make sure they are also clearly testable.

Further instructions:

\* Explain why each requirement is testable.

\* Explain why the completeness principle is fulfilled.

\* Provide a thorough explanation as to why the consistency principle is fulfilled.

\* Explain why the acceptance critera are clearly testable.

Response:

Let me provide a comprehensive requirements analysis with detailed explanations:

FUNCTIONAL REQUIREMENTS

1. Input Processing (Functional Suitability - Functional Completeness) FR1.1: The system shall accept PDF format CVs as input files up to 10MB in size Testability: Can be verified by attempting to upload PDFs of various sizes and validating system response

FR1.2: The system shall accept role descriptions as text input up to 2000 characters Testability: Can be verified by providing inputs of varying lengths and checking system handling

FR1.3: The system shall successfully extract text from PDF CVs with 100% text content preservation Testability: Can be compared against manual text extraction for verification

1. Analysis Processing (Functional Suitability - Functional Appropriateness) FR2.1: The system shall extract and categorize CV components into:

* Education (degrees, institutions, dates)
* Work experience (roles, companies, dates, responsibilities)
* Skills (technical, soft skills)
* Certifications Testability: Can be verified by comparing system categorization against manual categorization

FR2.2: The system shall identify from role descriptions:

* Required skills (minimum 80% accuracy)
* Required experience level
* Required education
* Required certifications Testability: Can be verified by comparing against human-identified requirements

1. Matching Analysis (Functional Suitability - Functional Correctness) FR3.1: The system shall generate a numerical match score (0-100) Testability: Can be verified by comparing scores across multiple CVs for the same role

FR3.2: The system shall identify all skill gaps and categorize them as:

* Critical (required but missing)
* Important (preferred but missing)
* Nice-to-have (mentioned but missing) Testability: Can be verified by comparing against manual gap analysis

1. Output Generation (Functional Suitability - Functional Completeness) FR4.1: The system shall generate a JSON report containing:

* Overall match score
* Categorized skill gaps
* Specific improvement recommendations Testability: Can be verified by validating JSON structure and content completeness

NON-FUNCTIONAL REQUIREMENTS

1. Performance Efficiency (Time Behavior) NFR1.1: The system shall complete analysis within 30 seconds Testability: Can be measured using performance timing tools
2. Reliability (Maturity) NFR2.1: The system shall maintain 95% accuracy in requirement identification Testability: Can be verified through comparison with expert analysis
3. Security (Confidentiality) NFR3.1: The system shall not persist any CV data beyond the analysis session Testability: Can be verified by inspecting storage after session completion

ACCEPTANCE CRITERIA

AC1: PDF Processing

* Test: System successfully processes PDFs of sizes 1KB to 10MB
* Expected Result: Complete text extraction with preserved formatting
* Verification Method: Automated comparison with original text

AC2: Match Score Accuracy

* Test: Process 20 CVs against 5 different role descriptions
* Expected Result: Consistent scoring (±5% variance for same inputs)
* Verification Method: Statistical analysis of results

AC3: Skill Gap Identification

* Test: Process CVs with known skill gaps
* Expected Result: 90% accuracy in identifying critical gaps
* Verification Method: Compare with expert analysis

COMPLETENESS EXPLANATION The requirements fulfill completeness because they cover:

1. All input scenarios (PDF processing, text input)
2. All processing steps (extraction, analysis, matching)
3. All output requirements (scoring, recommendations)
4. All error conditions and handling
5. All user interaction points
6. All data management aspects

CONSISTENCY EXPLANATION The requirements maintain consistency through:

1. No contradictory requirements (verified by cross-referencing)
2. Aligned performance metrics (all percentage-based metrics use same scale)
3. Compatible time constraints (processing times account for each step)
4. Consistent data handling approaches
5. Uniform error handling requirements
6. Coherent security model

TESTABILITY EXPLANATION Each requirement is testable because:

1. Contains measurable metrics (time, accuracy percentages)
2. Defines clear expected outputs
3. Specifies verification methods
4. Includes acceptance thresholds
5. Details test conditions
6. Provides comparison baselines

This comprehensive requirements specification ensures:

* Clear verification methods for each requirement
* Complete coverage of system functionality
* Consistent definitions and metrics
* Measurable acceptance criteria
* Traceable implementation guidelines

Would you like me to elaborate on any specific aspect of these requirements or their testability?