

Product Requirements Document (PRD)

Project Title: Description Optimizer for SEO

Date: July 28, 2025

1. Overview

1.1 Purpose

The SEO Description Optimizer is a Python-based solution that transforms raw product, service, or content descriptions from Excel or CSV files into SEO-friendly formats using OpenAI's language model. Its primary purpose is to improve the discoverability and relevance of content on search engines by automatically enhancing descriptions based on a user-defined prompt. This ensures that the resulting content is optimized for keywords, readable, structured in paragraph form, and aligned with domain-specific SEO strategies.

1.2 Goals

- Automatically optimize content descriptions to make them SEO-friendly, readable, and well-structured.
- Allow users to input a custom SEO prompt to guide the tone, style, and keyword targeting.
- Provide real-time feedback.

1.3 Scope

In Scope:

- Reading Excel (.xlsx) or CSV (.csv) files with columns: "Name," "Id," "Description."
- Extracting an SEO prompt from the first cell of the "Name" column (row 1).
- Optimizing descriptions (rows 2+) using Open AI's gpt-3.5-turbo model (or gpt-4 if specified).
- Saving optimized data to seo_optimized_data.xlsx.
- Handling errors (e.g., invalid file formats, missing columns, API failures).
- Supporting user-defined prompts for any domain.

Out of Scope:

- Generating charts or visualizations.
 - Supporting other file formats (e.g., .xls, JSON).
 - Real-time web scraping or external data integration.
 - Advanced SEO analytics (e.g., keyword density reports).
 - Multi-language SEO optimization.
 - UI/UX interface (script is command-line based).
-

2. User Stories

1. **As a user**, I want to upload an Excel or CSV file with descriptions so that I can automatically optimize them for SEO to improve online visibility.

2. **As a user**, I want to define an SEO prompt in the first cell of the file so that I can customize optimization (e.g., include domain-specific keywords).
 3. **As a user**, I want the script to handle both `.xlsx` and `.csv` files so that I can use my preferred format.
 4. **As a user**, I want clear error messages if the file is invalid or the API fails so that I can troubleshoot issues easily.
 5. **As a user**, I want the output saved as `seo_optimized_data.xlsx` without timestamps so that I can easily locate and share the file.
-

3. Functional Requirements

3.1 Input File Processing

- **F1:** The script must read an input file (`input_data.xlsx` or `input_data.csv`) with columns: "Name," "Id," "Description."
- **F2:** The script must extract the SEO prompt from the first cell of the "Name" column (row 1).
- **F3:** The script must validate:
 - Presence of required columns ("Name," "Id," "Description").
 - Valid SEO prompt (non-empty string in row 1, "Name" column).
 - Valid file format (`.xlsx` using `openpyxl`, `.csv` using `pandas`).
- **F4:** For `.xlsx` files, use the `openpyxl` engine to avoid format detection errors.
- **F5:** Handle errors:
 - `FileNotFoundError`: Print "Error: The file {input_file} was not found."
 - `BadZipFile`: Print "Error: The file is not a valid .xlsx file (not a ZIP archive)."
 - Missing columns: Print "Error: Excel file must contain 'Name', 'Id', and 'Description' columns."
 - Invalid prompt: Print "Error: The first cell in the 'Name' column must contain a valid SEO prompt."

3.2 SEO Optimization

- **F6:** Optimize descriptions (rows 2+) using Open AI's API (`gpt-3.5-turbo` by default).
- **F7:** Use the SEO prompt from the first cell to guide optimization, ensuring:
 - Inclusion of relevant keywords as specified in the prompt.
 - Avoidance of keyword stuffing.
 - Readability and full content retention (no length limits).
- **F8:** Clean descriptions before optimization by removing extra spaces (`re.sub(r'\s+', ' ', description.strip())`).
- **F9:** Capitalize optimized descriptions for readability.
- **F10:** Handle API errors by:
 - Printing "Error with Open AI API: {error}".
 - Falling back to the cleaned original description (capitalized).

3.3 Output File

- **F11:** Save the processed data to `seo_optimized_data.xlsx` using `openpyxl`.
- **F12:** Retain the original first row (prompt row) unchanged.
- **F13:** Overwrite `seo_optimized_data.xlsx` if it exists (no timestamp in filename).
- **F14:** Print confirmation: "SEO-friendly Excel file saved as {output_file}."

3.4 API Configuration

- **F15:** Use Open AI's `openai>=1.0.0` with `client.chat.completions.create`.
 - **F16:** Support API key via environment variable (`OPENAI_API_KEY`) or hardcoded (with warning to replace).
 - **F17:** Set `max_tokens=500` for longer descriptions and `temperature=0.7` for balanced output.
-

4. Non-Functional Requirements

4.1 Performance

- Process files with up to 1000 rows in under 5 minutes (dependent on API response time).
- Handle API rate limits gracefully with fallback to original descriptions.

4.2 Reliability

- Robust error handling for file I/O, invalid inputs, and API failures.
- Fallback mechanism ensures output file is generated even if API fails.

4.3 Security

- Warn users not to commit hardcoded API keys to version control.
- Encourage environment variable for API key (`OPENAI_API_KEY`).

4.4 Usability

- Clear error messages for debugging (e.g., invalid file format, missing columns).
- Simple command-line execution (`python seo_description_optimizer_with_openai.py`).

4.5 Compatibility

- Python 3.8+.
 - Libraries: `pandas`, `openpyxl`, `openai>=1.0.0`.
 - OS: Windows, macOS, Linux.
-

5. Technical Specifications

5.1 Architecture

- **Language:** Python 3.8+
- **Libraries:**
 - `pandas`: For reading/writing Excel and CSV files.
 - `openpyxl`: For `.xlsx` file handling.
 - `openai`: For SEO optimization via Open AI API.
 - `re`: For text cleaning.
 - `os`: For environment variable access.
 - `zipfile`: For `BadZipFile` error handling.
- **API:** Open AI `gpt-3.5-turbo` (configurable to `gpt-4` if user has access).
- **Input File:**
 - Format: `.xlsx` (preferred) or `.csv`.
 - Columns: "Name" (prompt in row 1), "Id," "Description."

- **Output File:** .xlsx (seo_optimized_data.xlsx).

5.2 Sample Input File

```
Name,Id,Description
"Optimize descriptions for SEO with relevant keywords; avoid keyword stuffing; keep full content",N/A,N/A
"Wireless Headphones",001,"High-quality headphones with noise cancellation and long battery life."
"Portable Charger",002,"A compact charger for devices with fast charging capabilities."
```

5.3 Sample Output File

Name	Id	Description
Optimize descriptions for SEO with relevant keywords; avoid keyword stuffing; keep full content	N/A N/A	
Wireless Headphones	001	Premium wireless headphones with advanced noise cancellation and extended battery life for immersive audio.
Portable Charger	002	Compact, high-speed portable charger for reliable device charging on the go.

5.4 Code Structure

```
# Initialize Open AI client
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY", "your-openai-api-key-here"))

# SEO optimization function
def make_seo_friendly(description, seo_prompt):
    # Clean description, call Open AI API, return optimized text

# File processing function
def process_excel_file(input_file, output_file):
    # Read file, validate, extract prompt, optimize descriptions, save output
```

6. Acceptance Criteria

- Input File:**
 - Script reads .xlsx or .csv files with required columns.
 - Extracts SEO prompt from row 1, "Name" column.
 - Fails gracefully with clear error messages for invalid files or columns.
- SEO Optimization:**
 - Optimizes descriptions (rows 2+) using Open AI API with user-defined prompt.
 - Retains full description length (no truncation).
 - Includes relevant keywords as per prompt.
 - Falls back to cleaned original description on API failure.
- Output File:**
 - Saves to seo_optimized_data.xlsx with unchanged first row.
 - Uses openpyxl engine for reliable .xlsx writing.
 - Confirms save with success message.
- Error Handling:**
 - Handles file not found, invalid .xlsx, missing columns, invalid prompt, and API errors.
 - Provides specific error messages (e.g., "The file is not a valid .xlsx file").
- API:**

- Uses `openai>=1.0.0` with `client.chat.completions.create`.
- Supports valid API key via environment variable or script.

7. Risks and Mitigations

Risk	Impact	Mitigation
Invalid <code>.xlsx</code> file (e.g., "File is not a zip file")	Script fails to read input	Provide clear error message; guide user to convert <code>.csv</code> to <code>.xlsx</code> correctly.
Open AI API failure (e.g., rate limits, invalid key)	Descriptions not optimized	Fallback to cleaned original descriptions; validate API key before running.
Cost of Open AI API usage	Unexpected expenses	Warn user to monitor token usage; use <code>gpt-3.5-turbo</code> for cost efficiency.
Overwriting <code>seo_optimized_data.xlsx</code>	Data loss	Document overwrite behavior; suggest manual backup or future file existence check.
Library version conflicts	Script errors	Specify <code>openai>=1.0.0</code> ; recommend virtual environment.

8. Implementation Plan

8.1 Development

- **Phase 1:** Script development (complete as of July 28, 2025).
 - Implemented file reading, prompt extraction, SEO optimization, and output saving.
 - Addressed errors (file format, API syntax, columns, prompt).
- **Phase 2:** Testing (recommended next step).
 - Test with various `.xlsx` and `.csv` files across domains (e.g., e-commerce, education).
 - Validate API performance with different prompts and description lengths.
 - Confirm error handling for edge cases.
- **Phase 3:** Deployment.
 - Share script with user (via file or repository).
 - Provide setup instructions (dependencies, API key).

8.2 Timeline

- Development: Complete (iterative updates from user feedback).
 - Testing: 1-2 days (user-driven).
 - Deployment: Immediate upon user confirmation.
-

9. Future Enhancements

- Add file existence check to prevent overwriting `seo_optimized_data.xlsx`.
- Support custom output file names via command-line arguments.
- Allow multiple prompts (e.g., per category in Excel).
- Integrate keyword density analysis for SEO validation.
- Add support for other file formats (e.g., JSON, `.xls`).
- Develop a simple GUI for non-technical users.

10. Setup Instructions

1. **Install Dependencies:**
2. `pip install pandas openpyxl openai`
3. **Set Open AI API Key:**
4. `export OPENAI_API_KEY="your-actual-api-key"`

Or update script with key (avoid committing to version control).

5. **Prepare Input File:**
 - Use provided `input_data.csv` or convert to `input_data.xlsx`.
 - Ensure columns: "Name," "Id," "Description" with prompt in row 1, "Name."
 6. **Run Script:**
 7. `python seo_description_optimizer_with_openai.py`
-

11. Notes

- **General Applicability:** The script is designed for any user needing SEO-optimized descriptions (e.g., e-commerce, content marketing). Users can customize prompts for their domain.
- **Cost Awareness:** Open AI API usage (with `max_tokens=500`) may incur costs. Monitor via Open AI dashboard.
- **Support:** For issues, provide error messages, input file sample, and environment details (e.g., Python version, library versions).