**Project Documentation: Gemini-based Product Description Generator**

**Overview**  
This Flask-based API generates product descriptions using Google's Gemini 1.5 Flash model. It accepts structured input and returns a JSON-formatted output suitable for e-commerce listings. It also includes an evaluation endpoint to assess generated content.

**Features**

* Accepts product info (name, category, price, features, etc.)
* Uses Gemini model to generate descriptions
* Validates input and output format
* Cleans and parses JSON from model
* Includes retry logic for robustness
* RESTful API with health check and evaluation

**Project Structure**

├── app.py # Main Flask application

├── config.py # Configuration for API key

├── prompts/

│ └── prompt\_templates.py # Prompt builder

├── utils/

│ ├── validators.py # Input/output validation

│ └── evaluator.py # Output evaluation logic

**Environment & API Key**

* Flask runs in debug mode (app.run(debug=True))
* Gemini API key is set in config.py

**How It Works**

1. **Input: (POST /generate-description)**

Expected JSON:

{

"product\_name": "iPhone 15 Pro",

"category": "smartphone",

"price": 999,

"key\_features": ["A17 Pro chip", "48MP camera", "Titanium body"],

"target\_audience": "tech-savvy users",

"tone": "professional"

}

1. **Workflow**
   * Input validated using validate\_input()
   * Prompt generated with get\_product\_description\_prompt()
   * Content generated by Gemini 1.5 Flash
   * Response parsed using clean\_json\_response()
   * Output validated and returned as JSON
2. **Output Example:**

{

"short\_description": "...",

"detailed\_description": "...",

"bullet\_points": ["..."],

"seo\_keywords": ["..."],

"call\_to\_action": "..."

}

📊 **Evaluation Endpoint: (POST /evaluate)**  
Evaluates generated description against input.

Example input:

{

"input\_data": {...},

"generated\_output": {...}

}

Returns a structured evaluation report.

💡 **Health Check: (GET /health)**  
Returns status of the app and Gemini configuration.

**Running the App**

**Requirements:**

* Python 3.8+
* flask, google-generativeai

**Command:**

python app.py

The app will run at: http://127.0.0.1:5000/

**Sample cURL Commands**

Generate Description:

curl -X POST http://127.0.0.1:5000/generate-description \

-H "Content-Type: application/json" \

-d '{

"product\_name": "iPhone 15 Pro",

"category": "smartphone",

"price": 999,

"key\_features": ["A17 Pro chip", "48MP camera", "Titanium body"],

"target\_audience": "tech-savvy users",

"tone": "professional"

}'

Evaluate Output:

curl -X POST http://127.0.0.1:5000/evaluate \

-H "Content-Type: application/json" \

-d '{

"input\_data": {...},

"generated\_output": {...}

}'

Health Check:

curl http://127.0.0.1:5000/health