# Final Project Report — Shopping Chat Assistant

## Objective

The main goal of this project is to build an AI-powered interactive shopping assistant capable of:

- Searching for products from Google Shopping via the SerpApi interface.
- Providing intelligent product comparisons and recommendations using Large Language Models (LLMs).
- Evaluating the quality of the model’s responses to improve accuracy and user experience.

---

## Project Phases

### 1. Data Retrieval — Using SerpApi

In the first stage, a Python script was developed to:

- Receive the user’s query (e.g., “best wireless headphones”).
- Send a request to the Google Shopping API (SerpApi).
- Collect product-related data, including:
  - Title, price, store name, link, and rating (if available).
- Reformat the retrieved data into a structured format (JSON / DataFrame) for easier processing later.

**Skills gained:** working with RESTful APIs, managing API keys, and handling JSON data.

---

### 2. LLM Integration — Shopping Chat Assistant

An interactive chat interface was built that:

- Receives user queries about products.
- Fetches real-time product data from SerpApi.
- Integrates this data into the conversation context and sends it to an LLM (via Groq API).
- Returns an analytical and summarized response containing comparisons or product recommendations.

**Goal:** to combine factual data from online sources with LLM reasoning to deliver an intelligent shopping experience.

---

## 3. Accuracy Evaluation — Performance Scoring

An automated evaluation system was created to measure the quality of the generated responses using three key metrics:

| Metric | Description |
|---|---|
| Faithfulness | How accurately the answer reflects the retrieved product data |
| Relevance | How well the response addresses the user's question |
| Completeness | How comprehensive and detailed the response is |

**The total score is calculated as follows:**

```
Total = (0.4 × Faithfulness) + (0.3 × Relevance) + (0.3 × Completeness)
```

The results are then classified into three performance levels:

- **High:** ≥ 80
- **Medium:** 50–79
- **Low:** < 50

The updated evaluation results were saved in the file
`data_shopping_eval_checkpoint_clean.xlsx`
which includes the columns:
`new_faithfulness`, `new_relevance`, `new_completeness`, `new_total`, and `new_performance_class`.

---

## 4. Similarity & Correlation Analysis

A **similarity_score** was calculated between the model's response and the retrieved content to measure textual alignment.
Additionally, a **correlation analysis (corr())** was performed to explore how the three metrics (faithfulness, relevance, completeness) are interrelated and which has the greatest impact on total performance.

---

# Results Summary

- All dataset entries were successfully evaluated.
- The new scores showed significant improvement compared to the previous ones.
- Most responses were categorized as medium to high quality.
- Two main output files were generated:
    - `data_shopping_eval_comparison.xlsx` — containing all scores and metrics.
    - `data_shopping_eval_comparison_correlation.xlsx` — containing correlation matrices among metrics.

---

## Error Handling

- Added logic to handle Groq API connection failures and Rate Limit (429) errors.
- Implemented retry mechanisms with waiting intervals.
- When repeated failures occur, default values are assigned to ensure the system continues running without interruption.

---

## Tools and Technologies

| Area | Tool / Technology |
|---|---|
| API Search | SerpApi (Google Shopping) |
| LLM Integration | Groq API (OpenAI-Compatible) |
| Data Analysis | Python (Pandas, JSON, Requests) |
| Text Similarity | difflib.SequenceMatcher |
| Data Storage | Excel (.xlsx) |
| Environment Management | dotenv (.env) |

---

## Challenges

1. Handling dynamic JSON data structures from SerpApi.
2. Managing Groq API rate limits effectively.
3. Ensuring continuous and stable evaluation for all records.
4. Balancing execution speed and evaluation accuracy.
5. Improving prompt engineering to maintain full context understanding without word fragmentation.
6. Solving JSON formatting issues by developing a repair and validation function.
7. Detecting and adapting to the user’s language for better localized responses.
8. Handling inaccessible links or API response errors gracefully.
9. Designing a user interface that is clean, responsive, and user-friendly.
10. Managing chat session history efficiently.

11. Ensuring consistent and well-structured data presentation.

---

## Future Work / Planned Improvements

- **Add non-shopping question detection:** enable the bot to recognize off-topic questions and respond appropriately.
- **Enhance rate limit management:** implement smart request queuing or parallel processing to reduce delays.
- **Expand data sources:** integrate with additional e-commerce APIs such as Amazon and Noon.
- **Introduce a predictive quality model:** train a simple ML model to estimate response quality without manual scoring.
- **Improve the user interface:** make comparison and recommendation visualization faster and more intuitive.

---

## Conclusion

The Shopping Chat Assistant project was successfully completed — from the initial data retrieval and API integration to automated evaluation and analysis.
The current system performs efficiently, producing accurate, classified, and structured responses.
Planned future work aims to further improve model accuracy, stability, and user interactivity to deliver an even more advanced and reliable shopping assistant experience.

---