Internship Summary

TripGoGo — AI & Prompt Engineering Intern

November 2025 – March 2026 — Part-time

Amsterdam Science Park, Netherlands

M.Sc. Electrical and Computer Engineering, Instituto Superior Técnico

Overview

During my internship at TripGoGo, I worked on improving the reliability and cost-efficiency of an LLM-based travel planning system. The product already generated complete itineraries from natural-language prompts, but the outputs still showed recurring factual inconsistencies. My work focused on evaluating those inconsistencies, understanding their root causes, and designing a practical path toward more grounded and reliable itinerary generation.

This experience gave me hands-on exposure to the difference between a system that looks convincing and a system that is actually robust. In particular, I learned how to think beyond prompt writing alone and treat an LLM application as a full pipeline: generation, validation, correction, and evaluation.

Project Context

The product goal was to generate travel itineraries that are both useful and believable. To achieve that, the system had to balance several requirements:

- **User alignment:** the itinerary should match the user's intent, mood, pace, and travel style;
- **Factual reliability:** suggested places should exist and be correctly referenced;
- **Operational quality:** itineraries should make practical sense in terms of locations and scheduling;
- **Efficiency:** generation quality should improve without making the system too expensive to run.

The challenge was that LLM outputs could be fluent and well-structured while still containing incorrect place names, wrong coordinates, or unrealistic sequencing.

My Role and Contributions

My internship was centred on reliability, evaluation, and grounded system design. More specifically, I contributed in four main ways:

- I tested generated itineraries and analysed recurring failure modes;
- I built a more structured framework for evaluating output quality;
- I identified where backend validation was needed instead of relying purely on the LLM;
- I proposed an incremental verification and RAG-oriented architecture for future development.

1. Identifying failure modes

A major part of my work was understanding where the system was failing. The most important error categories I found were:

- **Place naming errors:** generated places were sometimes too vague, slightly incorrect, or ambiguous;
- **Coordinate errors:** valid-looking places could still be linked to the wrong coordinates;
- **Feasibility issues:** some itineraries were unrealistic in terms of timing or visit sequence;
- **Context mismatch:** an itinerary could be factually acceptable but still not feel appropriate for the user request.

This helped turn quality assessment from subjective judgement into a more concrete engineering problem.

2. Rethinking evaluation

One of the most important lessons from the project was that itinerary quality has at least two dimensions:

- **Grounding quality:** whether names, places, and metadata are factually correct;
- **User-fit quality:** whether the result is actually a good answer for the request.

At first, I focused mostly on measurable correctness. Over time, I realised that this was not enough. A technically correct itinerary can still be a bad itinerary if it does not match the intended mood, traveller profile, or rhythm of the trip.

This changed how I thought about evaluation. Instead of asking only "Is this correct?", I started thinking in terms of:

- *Is this grounded?*
- *Is this useful for the specific user?*

3. Proposing a verification-based architecture

Based on the error patterns I found, I concluded that the system should not rely on the LLM alone for correctness-sensitive metadata. Instead, I proposed a hybrid workflow:

1. generate an itinerary draft with the LLM;
2. extract the places mentioned in the draft;
3. verify those places against a canonical source;
4. enrich or correct metadata such as names, coordinates, and opening hours;
5. optionally run a second refinement pass using the verified information.

This design was attractive because it was practical and incremental. It did not require a full system rewrite, but it directly addressed the most important failure modes I had observed.

4. Comparing quality and cost trade-offs

Another part of my work involved comparing model and correction strategies from both a quality and cost perspective. This meant thinking about:

- token usage and prompt efficiency;
- whether correction layers improved reliability enough to justify their cost;
- how to choose between cheaper and more expensive system configurations.

This gave me a more realistic understanding of AI engineering in a product setting: the technically strongest solution is not always the best solution if it does not scale economically.

What I Learned

This internship taught me much more than how to write better prompts. It helped me understand how to build and evaluate the full system around an LLM.

Technical learning

The most important technical things I learned were:

- how Retrieval-Augmented Generation fits into a real product pipeline;
- how to separate generative behaviour from deterministic backend logic;
- how to evaluate LLM systems using both quantitative and qualitative criteria;
- how external APIs and verified data sources can be used to reduce hallucinations;
- how token usage, correction steps, and model choice interact with cost.

Professional learning

Professionally, I learned how to work on an open-ended AI problem where the right solution is not obvious from the start. In particular, I improved in:

- structuring ambiguous problems into smaller, testable parts;
- documenting technical work clearly enough for future iterations;
- balancing exploration with practical product constraints;
- thinking iteratively rather than trying to over-design the system too early.

Outcome and Reflection

The result I am most proud of is that I helped make the system's weaknesses clearer and more actionable. Even without fully implementing the final architecture, I was able to:

- identify systematic reliability issues;
- connect those issues to concrete product risks;
- outline a more scalable and grounded direction for future development.

This internship confirmed that I enjoy solving open-ended engineering problems, especially when they involve system design, evaluation, and structured reasoning under uncertainty. It also

showed me that I am most motivated by building systems that are not only functional, but also logical, auditable, and maintainable.

Company website: https://www.tripgogo.ai/