Project Safety and Reflection Report

> A Real-Time Voice AI Consultant for TUM Applicants

Group: 5
Live Demo: https://voice-assistant-gilt.vercel.app/
GitHub Repository: https://github.com/tsaichen1o/voiceAssistant

---

---

1. Introduction

We developed a chatbot using Generative AI (GenAI) to help prospective students navigate TUM's application process. The chatbot is instructed to act as a professional academic advisor for the Technical University of Munich (TUM) and answer questions **only** based on the information available in the provided context from our TUM Knowledge Base. If the required information is not found in the context, it must clearly state that it cannot find the relevant information and must **not fabricate** an answer under any circumstances.

2. Safety of GenAI: Considerations and Mitigation Strategies

2.1 Safety Consideration

Safety is the most important factor in our chatbot design. The chatbot provides advice to applicants for TUM programs. If it provides incorrect information, it could prevent a program from reaching its intended audience or reduce the chances of attracting highly qualified applicants. Additionally, since the chatbot acts as a professional TUM academic advisor, any unintended behavior — such as responses triggered by prompt injection attacks — could potentially harm TUM's reputation.

2.2 Safety Tests

Our testing scripts and the test results are documented here and available for review.

- Collected Test Data and Results
- Test Automation Scripts

2.2.1 Accuracy Testing

Challenges
We encountered several challenges while designing the accuracy tests. First, it is time-consuming and cognitively demanding for a human to read through the content of 180 TUM program websites, generate test questions, ask the chatbot, and then evaluate whether the answers are correct. Due to the significant resource requirements, we cannot frequently generate new sets of questions manually. Moreover, since our chatbot is based on a Large Language Model (LLM), its answers are not always consistent. Therefore, traditional automated methods for validating correctness are not applicable.

Methodology
To address these challenges, we developed an LLM-based testing agent called the **Test Chatbot Agent**, which can automatically perform the following tasks: First, it generates TUM program-related questions along with their correct answers. These questions are designed to be objective and have only one correct answer. Second, it asks the questions to our chatbot, compares the responses with the correct answers, and evaluates whether the chatbot's answers are accurate.

We used the **Google Agent Development Kit (ADK)** to build this Test Chatbot Agent. ADK provides tools to develop AI agents capable of automatically calling functions with valid

parameters to complete assigned tasks. This is how our Test Chatbot Agent performs all testing steps without human involvement.

Test Results
According to the test report generated by the Test Chatbot Agent, our chatbot achieved an accuracy rate of **94.8%**, correctly answering **110 out of 116 questions**. Link to test report.

Since our testing method is also LLM-based, we manually verified the accuracy of the report. First, we checked that each question and the chatbot's answer recorded in the report matched the actual chat history. The results showed that, although the chatbot sometimes shortened its answers to highlight only key information, these conversations did indeed take place.

Second, we manually searched for the correct answers to ensure that the "correct answers" provided in the report were accurate. This verification confirmed their correctness.

Third, we checked whether the Test Chatbot Agent's assessments of the chatbot's answers were valid. We found **one** case where the chatbot's answer was actually correct, but it was mistakenly marked as incorrect. Additionally, in **two** cases, the chatbot responded with "I cannot find the relevant information," but the agent still marked those as incorrect.

2.2.2 Prompt Attacks Safety Testing

Challenges
Our initial approach to assess our chatbot's safety was by using a well-known benchmark. Specifically, we tested it using Microsoft's PromptBench. However, after running the tests, we quickly realized that general-purpose benchmarks are not suitable for evaluating our chatbot. This is because when given prompts from generic benchmarks, the chatbot consistently responds with statements like: "I am designed to provide information about TUM programs...".

Methodology
To test our chatbot's safety against prompt attacks, we needed to craft attack prompts related to TUM programs so that the chatbot would not immediately deny the request. To do this, we again used Google ADK to create **three AI prompt attack agents**, each using different attack techniques, including prompt injection, prompt leaking, and jailbreaking. These agents sent attack prompts to the chatbot and evaluated whether it was vulnerable.

Test Results
The test results show that 5 out of 13 prompt injection attacks succeeded, 8 out of 10 prompt leaking attacks succeeded, and all jailbreaking attacks failed. Link to successful attack reports.

2.3 Mitigation Strategies

2.3.1 Accuracy Mitigation Strategies

Although our chatbot's accuracy is high (94.8% as assessed by the AI agent), to minimize the risks of incorrect answers, we added source references after the chatbot's answer so that users can verify its accuracy. Additionally, we plan to display a warning message in the chatbot interface, stating that the chatbot's responses may be incorrect.

2.3.2 Prompt Attack Mitigation Strategies

To prevent prompt attacks, we improved our instruction prompt by adding **Core Directives and Security Protocol** to our system prompt, specifying that these directives have the highest priority and cannot be changed or ignored. Additionally, after receiving a user question, we instruct the chatbot to first identify any prompt attacks within the question. If any are found, it must refuse to answer. Finally, we re-ran the previously successful prompt attacks, and this time, all of them were correctly blocked.

3. Lessons Learned and Reflections

3.1 Lessons Learned

Through this project, we learned the importance of prompt engineering. Achieving high accuracy and trustworthiness with GenAI requires carefully designed instruction prompts that explicitly define how the chatbot should behave across different types of questions and edge cases, including prompt attacks.

Additionally, domain knowledge is critical. Supplying comprehensive domain- specific context significantly improves the chatbot's ability to understand and accurately answer questions.

3.2 Reflection

3.2.1 What worked well and what could be improved in the future

Our chatbot answers straightforward questions with high accuracy. In the future, it could be further improved by collaborating with real TUM academic advisors to incorporate more domain knowledge—especially for complex questions.

Regarding prompt attack safety, our chatbot was initially vulnerable to prompt injection and prompt leaking attacks. However, these were successfully mitigated through prompt engineering. Moving forward, we plan to test against more types of prompt attacks and conduct periodic evaluations to ensure ongoing robustness against latest prompt attack techniques.

3.2.2 Insights Gained

First, we learned how to practice the Waterfall software development lifecycle and collaborate effectively as a team while following software development best practices. Second, we learned how to develop and test a GenAI-based chatbot. Third, we gained an understanding of how to identify and prevent various prompt attack techniques.

3.2.3 How this project has influenced our understanding of GenAI and its applications

This project has significantly broadened our understanding of GenAI beyond that of casual users. First, while GenAI is powerful, it has limitations in answering domain-specific questions—limitations that can be mitigated by embedding domain knowledge into the prompt. Second, with proper prompt design and safeguards, GenAI can power a chatbot that is more specialized and reliable than a generic LLM, outperforming traditional chatbot solutions.