Thoutha AI Chatbot — Development Challenges and Solutions

1. Model Availability and API Compatibility

Problem:
During early testing, the chatbot frequently failed to generate responses due to a `404 models/gemini-1.5-flash not found` error. This occurred because the model name used in the code (`gemini-1.5-flash`) was deprecated or unsupported under the `v1beta` API.

Impact:
The application would crash during response generation, making it impossible for the chatbot to reply to user queries.

Solution:
We updated the Gemini SDK (`google-generativeai`) to the latest version and changed the model to a supported one — `gemini-1.5-flash-latest`. This ensured full compatibility with the stable `v1` API and restored normal chatbot operations. A version check function was also added to detect and prevent future mismatches between the SDK and the API version.

---

2. Domain Restriction Logic (Dental-only Responses)

Problem:
Initially, the chatbot could answer any kind of question, which was not aligned with the project's goal of focusing solely on dental-related queries. Users could ask unrelated questions (e.g., about sports or politics), and the system would still respond.

Impact:
This reduced the chatbot's professional reliability and could cause confusion for elderly users, who are the primary target audience.

Solution:
A domain classifier prompt was implemented using a lightweight Gemini model to filter questions. The model checks if a user's query relates to dental health, teeth, or oral hygiene. If not, the chatbot politely declines and redirects the user back to the intended

topic. Later, a hybrid approach was added — combining keyword detection (a predefined dental-related word list) with AI-based classification for higher accuracy and lower cost.

---

3. Handling User Typos

Problem:
Users, especially elderly ones, often make spelling mistakes or typing errors (e.g., typing “اسناتي” instead of “أسناني”). These errors caused the chatbot to misclassify queries as non-dental, leading to poor user experience.

Impact:
Legitimate questions were rejected because of simple typos, making the system feel rigid and unhelpful.

Solution:
We integrated a lightweight text preprocessing module using fuzzy string matching to correct common spelling mistakes in both Arabic and English before classification. This significantly improved understanding accuracy without affecting response time.

---

4. Arabic Language Support

Problem:
Detecting Arabic text and ensuring it was processed correctly was challenging, as some libraries struggled with UTF-8 encoding or partial Arabic input mixed with English.

Impact:
The chatbot occasionally failed to detect Arabic messages, leading to responses in the wrong language.

Solution:
A custom regex-based `is_arabic()` function was implemented to accurately detect Arabic characters. Additionally, the model prompt was adjusted to maintain consistent language output (responding in Arabic if the query was Arabic, and in English otherwise).

---

5. Stability and Error Handling

Problem:
Uncaught exceptions from the Gemini API (timeouts, invalid responses, or connectivity errors) caused the application to terminate abruptly.

Impact:
This affected reliability, especially during live testing or demonstrations.

Solution:
We implemented robust try-except blocks in critical sections of the code (notably `generate_response()` and `is_query_dental()`) to catch exceptions gracefully. Instead of crashing, the system now provides a friendly fallback message to the user, ensuring continuous operation.

---

6. Performance Optimization

Problem:
Each classification and generation call initially used the same large model, which increased latency and cost.

Impact:
Responses were noticeably slow, especially during classification.

Solution:
We split the architecture into two model instances:

- A **lightweight model** (`gemini-1.5-flash-latest`) for classification and fast dental detection.
- A **full conversational model** for generating detailed responses.
 This division reduced response time and API token consumption while maintaining quality.

Summary

Through systematic debugging, testing, and iteration, we transformed the initial unstable chatbot prototype into **Thoutha**, a reliable, domain-specific assistant capable of understanding Arabic and English dental-related queries, correcting typos, and providing clear, medically relevant information for elderly users.