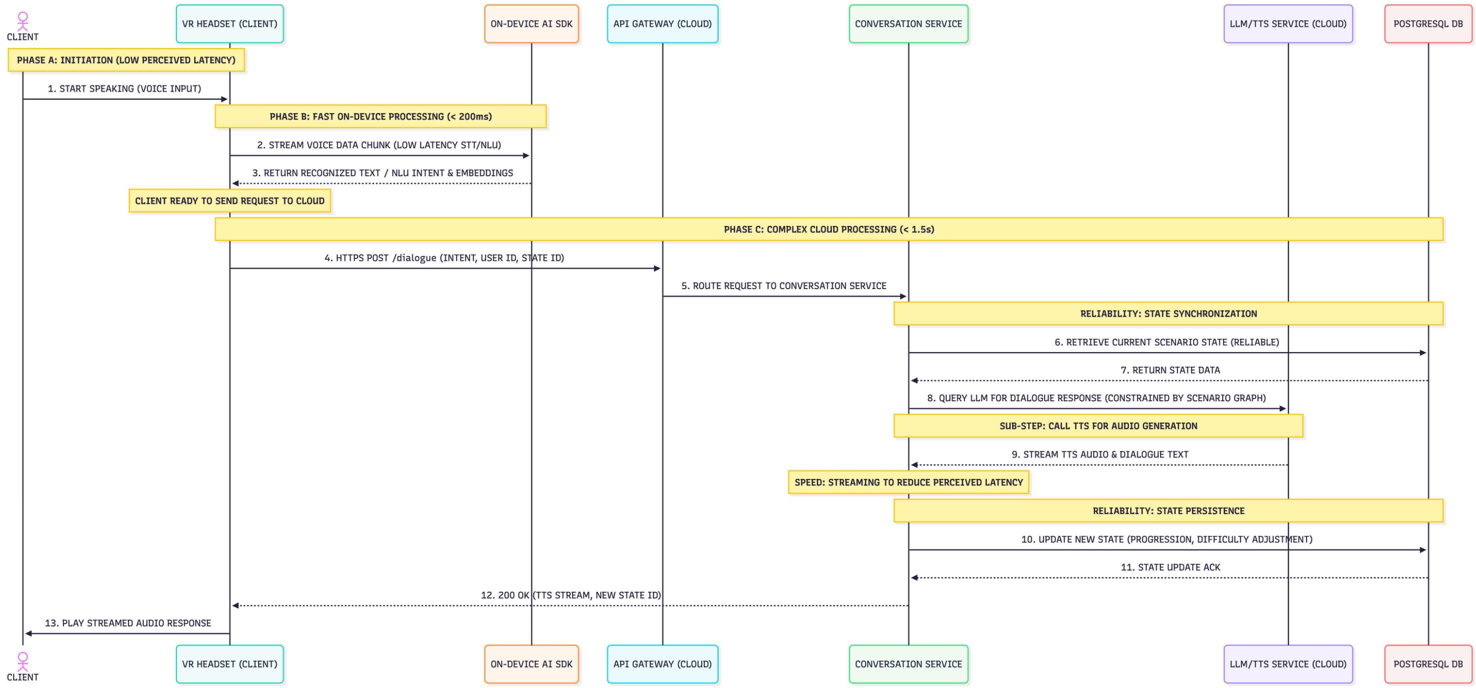
RQ3: How should the VR-backend interface minimize latency while ensuring reliable state synchronization?

1. Formal Specification: Sequence Diagram

The architectural solution proposed to address the requirements of RQ3 is based on the choice of a **Hybrid Edge-Cloud microservices architecture**. This approach is the key decision for simultaneously optimizing real-time performance and data consistency.

This diagram highlights the **task breakdown** (Phases B and C) designed to meet both latency (NFR1:< 1.5s) and reliability objective.



2. Detailed Architectural Justification and Strategies

The design of the VR-backend interface is structured around two fundamental goals: minimizing latency (speed) and ensuring reliable state synchronization (reliability).

A. Strategies for Minimizing Latency (Speed)

The primary goal is to adhere to the non-functional requirement (NFR1) for the cloud loop latency of less than 1.5 seconds.

Strategy	Mechanism in the Interface	Impact on Latency
Hybrid Edge Processing	Lightweight components like basic Speech-to-Text (STT) and initial Natural Language Understanding (NLU, embeddings) are executed by the **On-Device AI SDK** (using Whisper Tiny + NLU).	Reduces **perceived latency** to **less than 200ms** for simple interactions by handling latency-critical tasks on the device.
TTS Streaming	The LLM/TTS Service generates the audio response via **streaming**.	The VR client can begin playing the audio (TTS) before the full message is received, effectively masking network and complex LLM generation latency.
Optimized Backend Stack	The backend is implemented using **Golang** microservices orchestrated by **Kubernetes**.	Golang offers high performance, and Kubernetes enables **rapid scaling** [12] to maintain performance under load.
API Gateway Pattern	Utilizes an **API Gateway** as a single entry point for the VR clients.	The gateway ensures efficient request routing, authentication, and rate limiting, reducing the overhead before reaching the Conversation Service.

B. Strategies for Ensuring Reliable State Synchronization (Reliability) .

Reliability is secured by the microservices model, ensuring that the user's pedagogical progress is consistently maintained (FR6).

Strategy	Mechanism in the Interface	Impact on Reliability
Persistent Database	**PostgreSQL** is used as the **single source of truth** for critical user progress and state data.	Ensures reliable (ACID) transactions for state retrieval (Message 6/7) and update (Message 10/11), guaranteeing persistent user profiles and progress tracking (FR6).
Scenario Control	LLM dialogue generation is **constrained by scenario graphs** (FR2), guided by the retrieved state.	Ensures the AI's response is aligned with pedagogical objectives (adaptive difficulty adjustment, FR5) and prevents inconsistent dialogues, enhancing pedagogical reliability.
State Isolation	State logic is separated into functionally isolated microservices (**Conversation Service**, **User Service**).	Reduces coupling and the risk of concurrent write errors.
Asynchronous Updates	**Redis Pub/Sub** is used for asynchronous processing (analytics, progress tracking).	Non-critical updates are processed in the background, avoiding blocking the synchronous LLM conversation loop, thus improving performance without compromising the reliability of the immediate interaction.

3. Conclusion

The answer to Research Question 3 is delivered by the adoption of a **Hybrid Edge-Cloud microservices architecture**. This solution successfully achieves both critical system objectives:

1. **Latency Minimization:** Achieved through **Edge offloading** of the initial speech processing and **TTS streaming**, which together reduce perceived latency and mask the Cloud delay.

2. **State Reliability:** Guaranteed by the **PostgreSQL** database as the *single source of truth* and the **Conversation Service** which ensures consistent state updates and dialogue control strictly aligned with pedagogical scenarios.

This design ensures a fluid and consistent VR experience while maintaining reliable user progress and traceability.

4. Bibliographic References

The strategies and architectural choices presented in this document are supported by the following scientific and technical literature:

[1]Rod Ellis. 2008. Principles of Instructed Second Language Acquisition. CAL Digest (2008).

[2]Adithya T. G., Abhinavaram N., and Gowri Srinivasa. 2024. Leveraging Virtual Reality and AI Tutoring for Language Learning... arXiv preprint arXiv:2411.12619 (2024).

[3]Bob Godwin-Jones. 2024. AI and VR Converge: The Future of Language Learning in an Emerging XR Ecosystem. Self-Published Research Report (2024).

[4]Frontiers in Virtual Reality. 2025. Analyzing and Comparing Augmented and Virtual Reality in Vocabulary Learning. Frontiers in Virtual Reality (2025).

[5]ISO/IEC/IEEE. 2011. Systems and software engineering Architecture description. Standard ISO/IEC/IEEE 42010:2011.

[6]Stephen D. Krashen. 1982. Principles and Practice in Second Language Acquisition. Pergamon Press, Oxford.

[7]Michael H. Long. 1996. The Role of the Linguistic Environment in Second Language Acquisition. Academic Press, San Diego, CA.

[8]Oxford University Press. 2025, Virtual Reality for Language Learning. ELT Journal (2025).

[9]Isabel Schorr, David A. Plecher, Christian Eichhorn, and Gudrun Klinker. 2024. Foreign language learning using augmented reality environments: a systematic review. Frontiers in Virtual Reality 5 (2024), 1288824.

[10]Merrill Swain. 1995. Three Functions of Output in Second Language Learning. Principles and Practice in Applied Linguistics: Studies in Honour of H. G. Widdowson (1995), 125-144.

[11]Michael Vallance. 2024. Virtual Heritage & AI: Learning about 19th Century Japan with Isabella Bird. In ICEEL '23: Proceedings of the 2023 7th International Conference on Education and E-Learning.

[12]Zihao Zhu, Ao Yu, Xin Tong, and Pan Hui. 2025. Exploring LLM-Powered Role and Action-Switching Pedagogical Agents for History Education in Virtual Reality. In CHI 25...