Bilkent University

Senior Design Project - CS491

Fall 2025/2026

Project Specifications Report

Group “PARSE” - T2501

Supervisor: Ayşegül Dündar Boral

Innovation Expert: Sezai Artun Özyeğin

Team Members:

22203783 Anıl Kılıç

22202680 Aybars Buğra Aksoy

22202981 Barış Yaycı

22203661 Bora Yetkin

22201772 Eren Berk Eraslan

Project Specification Report: PARSE

1. Introduction

Online teaching, technical talks, and remote meetings increasingly rely on slide based presentations delivered over video conferencing platforms. While these tools make it easy to broadcast content, they do very little to ensure that participants understand and retain what is being presented. Attendees often join sessions part way through, miss verbal explanations, or fail to connect earlier slides to later ones. Existing “AI meeting assistants” mostly operate as transcription tools: they capture audio, generate a text log, and sometimes produce a generic summary. However, they rarely understand the visual content of slides such as formulas, charts, diagrams, or code, and therefore cannot answer questions that depend on slide graphics rather than speech alone. This gap is particularly problematic in visually dense domains such as engineering, finance, or scientific education, where key information is encoded in plots, equations, architectural diagrams, and tables. Most current assistants are unable to ground their answers in such content, because they do not parse layout, do not understand visual objects on slides, and cannot align these visuals with the presenter’s spoken narrative.

1.1 Description

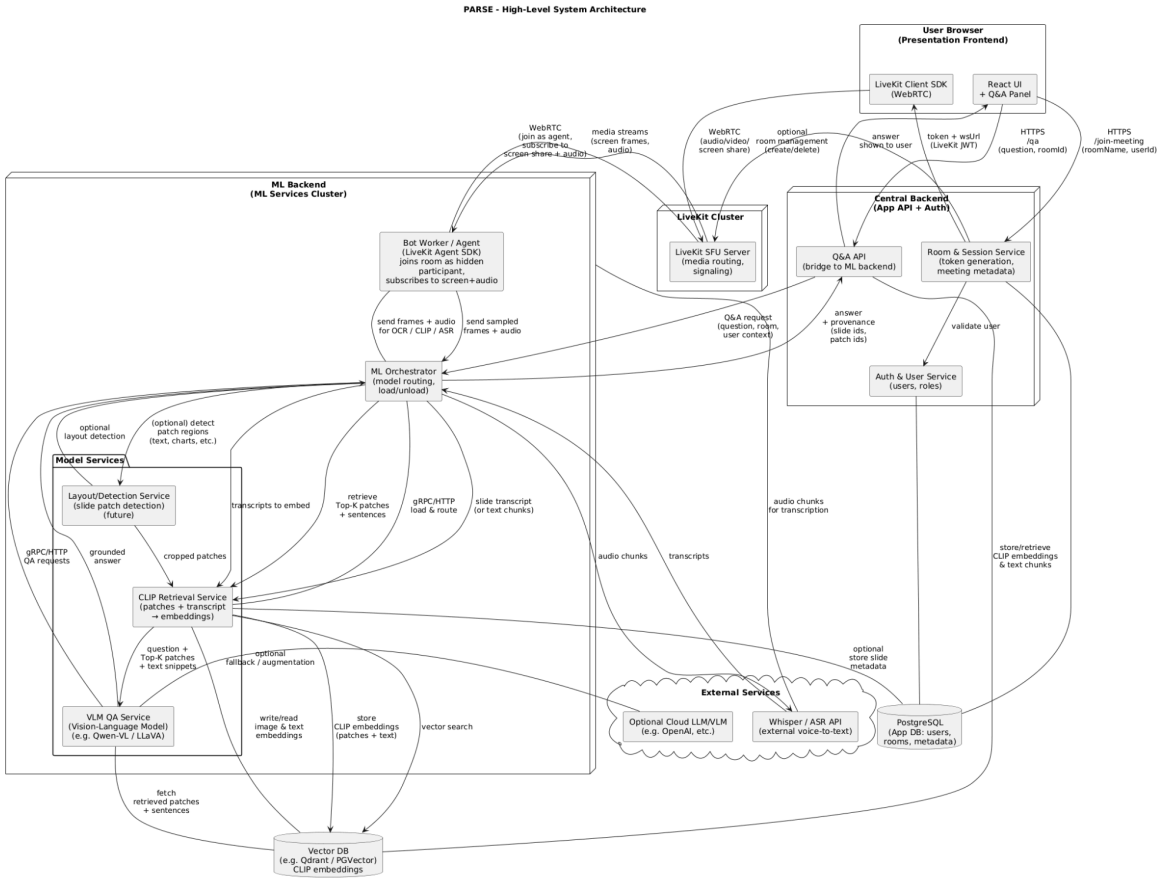
PARSE is an interactive AI assistant specifically designed to enhance online presentations by bridging the gap between passive viewing and active understanding. Unlike traditional meeting assistants that rely solely on audio transcription, PARSE continuously fuses verbal explanations with **on-screen visual materials** (text, formulas, tables, and images) to maintain a **live**, semantic model of the talk.

The system allows participants to ask natural-language questions at any time and receive real-time, grounded answers with citations linking back to specific slides or timestamps [1]. Following the session, PARSE generates automated summaries and analytics dashboards, highlighting frequently asked topics and unresolved queries. Designed with a "privacy-first" approach, the system ensures session-scoped data retention, meaning all data is permanently deleted upon session termination.

Architecturally, the system is split into:

- A **multimodal ML services backend** where an orchestration layer manages GPU resources, loads/unloads models like CLIP, DocLayout-YOLO, and Qwen-VL on demand, and routes requests to model containers [2][3][4][5].
- A **central application backend** responsible for user/session management, LiveKit token generation, API orchestration, and persistence of non-sensitive metadata in PostgreSQL [6].
- A **temporary vector store** holding session-scoped embeddings for text and image patches.
- A **web client** for presenters and participants, providing LiveKit-based audio/video, slide viewing, and an interactive Q&A interface.

1.2 High Level System Architecture & Components



The proposed solution utilizes a multimodal AI pipeline built on a microservices architecture. The core components are:

- **Communication Infrastructure: LiveKit** is used as the backbone for real-time video and audio streaming, ensuring low-latency data transmission without relying on external bot integrations or third-party meeting platforms [7].
- **Backend Framework:** The core application logic is developed using **FastAPI** (Python), chosen for its high performance and native support for asynchronous processing required for real-time AI tasks [6].
- **Audio Processing Pipeline:**
    - Incoming audio streams are processed using **Deepgram** or **OpenAI Whisper** for high-accuracy Speech-to-Text (STT) transcription [8].
- **Visual Processing Pipeline:**
    - **DocLayout-YOLO:** Utilized for real-time slide analysis, specifically to detect and patch layout elements (headers, text blocks, formulas, images) [3].
    - **CLIP (Contrastive Language-Image Pre-training):** Employed to create embeddings for visual frames, acting as a retrieval mechanism to match user queries with relevant visual context efficiently [4].
- **Reasoning & Generation (VLM):**
    - **Qwen-VL:** Serves as the primary Vision-Language Model to synthesize the final answer by reasoning over both the transcribed text and the retrieved visual context from the slides [5].
- **Client Side:** A web-based interface for the presenter and attendees to view the presentation and interact with the Q&A bot.

1.3 Constraints

1.3.1 Implementation Constraints

The implementation of PARSE involves several technical constraints stemming from its real-time, multimodal design. The system must process audio, video, and visual slide information with minimal latency to ensure a smooth user experience during live presentations. Heavy models such as **Qwen-VL**, **CLIP**, and **DocLayout-YOLO** introduce significant computational overhead, making inference optimization a critical requirement.

Communication Infrastructure:
PARSE relies on **LiveKit** as the backbone for real-time audiovisual streaming. Unlike platforms that depend on external meeting software (e.g., Zoom bots), PARSE must directly handle **WebRTC media streams**, including encoding, decoding, packet loss management, and synchronization. This introduces additional complexity compared to simple REST-based model interactions [7].

Backend Framework:
The backend is built using **FastAPI**, selected for its high throughput and asynchronous capabilities. FastAPI enables non-blocking processing of concurrent audio and video frames, which is essential for real-time AI inference workloads [6].

Audio Processing Pipeline:
Incoming audio is transcribed using **Deepgram** or **OpenAI Whisper**, chosen for their robustness to accents, noise, and domain-specific language. These systems must operate continuously under tight latency constraints to ensure that speech-to-text outputs are available to the reasoning module on time [8].

Visual Processing Pipeline:

- **DocLayout-YOLO** is used to detect slide components titles, paragraphs, formulas, images enabling structured visual understanding [3].

- **CLIP embeddings** are generated for each captured frame so that user queries can be matched against visually relevant context efficiently. These embeddings must be generated and indexed in near real-time [4].

- **Reasoning and Generation (VLM):** The **Qwen-VL** model is employed as the core reasoning engine. It synthesizes responses using both user question, transcribed speech and visual embeddings, producing grounded, context-aware answers. Running a vision-language model at interactive speeds constrains the choice of model size and necessitates GPU-level optimization [5].

Client Interface:
A lightweight browser-based interface serves both presenters and attendees. The client must maintain live WebRTC streams, display slides, and support interactive Q&A while remaining responsive under unstable network conditions.

Additional Implementation Tools:

- **Python** is the primary development language for machine learning pipelines and inference integration.

- **Git** and **GitHub** support version control and collaborative development.

These requirements collectively restrict architectural freedom and necessitate systematic engineering decisions to maintain both performance and reliability.

1.3.2 Economic Constraints

PARSE is developed as a university capstone project, meaning the financial budget is strictly limited and must rely entirely on the team's own resources, without external sponsors or institutional funding. Consequently, all model choices, hosting decisions, and compute configurations emphasize cost efficiency. To minimize licensing and recurring costs, the system exclusively uses open-source models such as Qwen-VL, DocLayout-YOLO, Whisper, CLIP, and PyTorch-based pipelines. This eliminates dependency on costly proprietary APIs. However, the reliance on GPU-intensive models introduces constraints related to cloud compute costs. Running inference on cloud GPUs (e.g., Nvidia A10, L4, or T4 instances) requires careful scheduling, batching, and usage minimization. The architecture must therefore maximize inference efficiency to remain within the student budget.

1.3.3 Ethical Constraints

PARSE processes highly sensitive multimodal data—including users' voices, faces, presentation slides, and potentially confidential corporate content. Therefore, the system enforces strict ethical constraints centered on privacy, consent, and responsible data handling.

- **Consent:**
  Users must explicitly provide **informed consent** before audio or video streams are processed. The system's onboarding flow clearly explains what data is processed and how it is used.
- **Session-Scoped Retention:**
  PARSE follows a **zero persistent storage** policy. All audio, video, transcriptions, visual embeddings, and generated answers are processed **ephemerally** and discarded at the end of the session. No identifiable data is saved, logged, or transmitted to third parties.
- **Privacy Protections:**
  User data is never used to train models, improve system performance, or build datasets. All processing occurs only for the duration of the live session, and the system is architected to avoid accidental retention or leakage.

These ethical constraints shape the software architecture, model selection, database policies, and legal compliance mechanisms.

1.4 Professional and Ethical Issues

The development of PARSE adheres strictly to the Ethics and Professional Conduct, ensuring responsible engineering practices throughout the lifecycle of the project. The system is designed with an explicit commitment to transparency, accuracy, and confidentiality.

- **Transparency:**
  Users are informed clearly and proactively that they are engaging with an AI system. The interface states that PARSE processes audiovisual data in real time and provides AI-generated answers based on that input. No misleading or hidden automation is used.
- **Accuracy and Grounding:**
  To minimize hallucinations, one of the primary risks of large language models, the system enforces a grounding strategy where answers cite visual or verbal evidence extracted from the user's slides or spoken content [2]. This ensures that responses remain faithful to the presentation material rather than speculative or fabricated.
- **Confidentiality and Security:**
  The system is designed to prevent data leaks and protect sensitive information. Because PARSE operates entirely within the session window, no user content is stored, cached, exported, or used for training. By deleting all multimodal data once the session ends, the system eliminates long-term risks associated with data misuse, unauthorized access, or privacy violations.

PARSE incorporates secure communication protocols, encrypted transmission channels, and strict access controls to ensure that no unauthorized party can intercept or observe the audiovisual streams. Protecting user data is treated as a professional and ethical obligation; the system prioritizes privacy and confidentiality even at the cost of additional development complexity and reduced system analytics.

Overall, PARSE's design reflects a principled commitment to user safety, responsible AI usage, and ethical computational practices.

1.5 Standards

- **IEEE 830-1998:** Recommended Practice for Software Requirements Specifications [9].
- **UML 2.5.1:** Used for system modeling (Sequence and Component diagrams) [10].
- **RESTful API Standards:** For communication between the Frontend and FastAPI backend.
- **WebRTC:** Standard for real-time communication via LiveKit [11].

2. Design Requirements

2.1 Functional Requirements

The functional requirements are categorized into four core modules: Session Management, Multimodal Ingestion, Query Processing, and Post-Session Analysis.

2.1.1 Session Management & Connectivity

- **FR-01:** The system shall utilize **LiveKit** to establish a secure, low-latency WebRTC connection for both audio and video tracks.
- **FR-02:** The system shall support a "host" mode to initiate sessions and a "participant" mode to join via a unique room token.
- **FR-03:** The system shall detect active presentation slides versus webcam feeds to prioritize high-resolution processing for slides.

2.1.2 Multimodal Data Ingestion & Processing

- **FR-04 (Audio Pipeline):** The system shall stream audio buffers to **Deepgram/Whisper** APIs for real-time Speech-to-Text (STT) conversion.
- **FR-05 (Visual Pipeline):** The system shall sample video frames at a configurable rate (e.g., 0.5 FPS) to minimize redundant processing while capturing slide transitions.
- **FR-06 (Layout Analysis):** The system shall apply **DocLayout-YOLO** to extracted frames to identify and segment text blocks, headers, figures, and tables.
- **FR-07 (Embedding Generation):** The system shall generate vector embeddings for both transcribed text and visual segments using **CLIP** to enable semantic search.

2.1.3 Interactive Q&A & Reasoning

- **FR-08:** The system shall provide a chat interface allowing users to submit natural language queries during the live session.
- **FR-09 (Retrieval):** Upon receiving a query, the system shall retrieve the top-k most relevant audio transcripts and visual frames from the session history.
- **FR-10 (Generation):** The system shall pass the retrieved context and user query to the **Qwen-VL** model to generate a context-aware answer.
- **FR-11 (Grounded Citations):** Every generated answer must include a citation, explicitly linking to the specific timestamp of the audio or the slide number where the information was presented.

2.1.4 Post-Session Operations

- **FR-12:** The system shall generate a concise textual summary of the presentation immediately after the session ends.
- **FR-13 (Data Purging):** The system shall execute a hard-delete operation on all temporary vector stores and media files upon session termination to ensure privacy ("Session-Scoped Retention").

2.2 Non-Functional Requirements

These requirements define the quality attributes, performance constraints, and operating standards of the PARSE system.

2.2.1 Performance & Latency

- **NFR-01 (End-to-End Latency):** The total time from a user submitting a question to receiving an answer shall not exceed **10 seconds** under normal network conditions.
- **NFR-02 (Transcription Lag):** The delay between spoken words and the availability of their text transcript (STT latency) shall be less than **5 seconds**.
- **NFR-03 (Frame Processing):** The visual processing pipeline (YOLO + CLIP) must process a slide frame within **3 second** to ensure the bot's knowledge base is current.

2.2.2 Scalability & Concurrency

- **NFR-04:** The backend architecture (FastAPI) shall be stateless to allow for horizontal scaling via container orchestration (e.g., Docker Swarm or Kubernetes) if resources permit.

- **NFR-05:** The system shall support at least **50 concurrent users** in a single LiveKit room without degradation in audio/video quality.

2.2.3 Reliability & Availability

- **NFR-06 (Reconnection):** In the event of a client-side network drop, the system shall automatically attempt to reconnect to the LiveKit room for up to 30 seconds before timing out.
- **NFR-07 (Fallback):** If the visual analysis model (Qwen-VL) fails or times out, the system shall gracefully degrade to answer based solely on the textual transcript, notifying the user of the limitation.

2.2.4 Security & Privacy

- **NFR-08 (Encryption):** All media streams transmitted via LiveKit must be encrypted using **DTLS/SRTP** (Datagram Transport Layer Security / Secure Real-time Transport Protocol).
- **NFR-09 (Volatile Storage):** The system shall use in-memory databases (e.g., Redis or temporary vector indices) for session data to prevent accidental long-term persistence on disk.

2.2.5 Usability

- **NFR-10:** The user interface shall be responsive and accessible via standard modern web browsers (Chrome, Firefox, Edge) without requiring additional software installation.
- **NFR-11:** The Q&A panel must not obscure more than 20% of the presentation screen area on desktop devices.

3. Feasibility Discussions

This section evaluates PARSE from two perspectives:

(1) Market & competitive feasibility, which assesses how the system fits into the current technological landscape, and

(2) Academic feasibility, which examines its technical soundness, research alignment, and practicality within a university project.

3.1 Market & Competitive Analysis

3.1.1 Overview of Existing Solutions

The current market for meeting and presentation assistants is dominated by products such as **Otter.ai**, **Microsoft Copilot**, **Google Duet**, and **Zoom AI Companion**. These solutions primarily offer:

- Speech-to-text transcription

- Meeting summaries

- Keyword extraction
- Chat-based Q&A derived from text logs

While effective for note-taking and basic meeting automation, these systems share a critical limitation: they operate almost entirely on **audio-only understanding**. Their Q&A capabilities rely on transcripts and cannot interpret the **visual content** on presentation slides—formulas, diagrams, charts, code snippets, tables, or layout structure.

As modern presentations, especially in academic and technical fields, heavily depend on visual information, this creates a substantial unmet need in the market.

3.1.2 Identified Market Gap

In domains such as engineering, machine learning, finance, and science education, participants often ask questions directly about the *visual content* on slides:

- "What does this graph imply?"
- "Can you explain the equation on the slide?"
- "What is the meaning of this architecture diagram?"

Traditional assistants cannot answer these questions because they:

1. Do not parse slide layouts,
2. Do not understand visual objects,
3. Cannot relate visuals with spoken explanations.

This makes them unsuitable for visually rich presentations and technical lectures.

3.1.3 PARSE's Competitive Advantage

PARSE differentiates itself through **multimodal comprehension**, achieved by integrating:

- **DocLayout-YOLO** for detecting text blocks, titles, formulas, and figures.
- **CLIP embeddings** for cross-modal semantic retrieval.
- **Qwen-VL**, a state-of-the-art vision-language model capable of interpreting images, reading text, and answering questions grounded in visuals.

This enables PARSE to:

- Understand slide content frame-by-frame,
- Match user queries to relevant slide regions,
- Combine transcript information with extracted visual elements,
- Provide grounded answers with citations to specific timestamps or slide numbers.

No mainstream commercial tool currently provides this combination of **real-time multimodal retrieval + grounded reasoning**.

3.1.4 Target Market and Use Cases

PARSE naturally fits several domains where slide content is critical:

- **Online Education:** university lectures, MOOCs, recorded lessons
- **Technical Webinars:** machine learning, cybersecurity, engineering workshops
- **Corporate Training Sessions:** process explanations, internal workshops
- **Research Conferences:** presentations with heavy diagrams and equations

These audiences often request real-time clarifications about visual material; PARSE directly addresses this pain point.

3.1.5 Cost and Deployment Feasibility

Because PARSE uses **open-source** models (Qwen-VL, Whisper, DocLayout-YOLO, CLIP), the system avoids the recurring API costs associated with proprietary AI services. This aligns with the **economic constraints** of a student capstone project.

Computational requirements can be managed through:

- GPU-efficient model sizes
- Reduced frame sampling rates (e.g., 0.5 FPS for slides)
- Batch processing when possible
- Temporary in-memory data for session-scoped retention

This makes the project financially feasible using low-cost GPU instances or a single university-provided GPU workstation.

3.1.6 Market Feasibility Conclusion

Given the limited capabilities of existing meeting assistants and the strong demand for tools that understand **both spoken content and slides**, PARSE provides a unique and competitive solution. Its reliance on open-source technologies further strengthens feasibility within the capstone budget. The market feasibility is therefore **high**, with a clear niche that is currently underserved.

3.2 Academic Analysis

3.2.1 Alignment with Modern AI Research

PARSE aligns closely with several active research areas in artificial intelligence:

- **Vision-Language Models (VLMs)**
- **Document Layout Understanding**
- **Multimodal Retrieval-Augmented Generation (RAG)**
- **Real-time inference and streaming ML**
- **Cross-modal semantic alignment using CLIP embeddings**

These topics appear frequently in recent ML conferences (CVPR, NeurIPS, ACL, ICCV), indicating that PARSE sits at the intersection of modern academic interest and applied engineering.

3.2.2 Technical Soundness

The system architecture is built around components that are extensively documented and validated in literature:

- **Qwen-VL** is recognized for strong OCR, chart reading, and multimodal reasoning.
- **Whisper / Deepgram** are industry-standard STT systems capable of handling noise, accents, and technical vocabulary.
- **CLIP** provides a unified embedding space for text and images, enabling efficient retrieval.
- **DocLayout-YOLO** is optimized for detecting structured elements within document-like layouts.
- **FastAPI** supports high-performance asynchronous data handling.

- **LiveKit** abstracts complex WebRTC operations like ICE negotiation, RTP streams, and synchronization.

This means that each subsystem is supported by mature academic or industrial work, reducing research risk and implementation uncertainty.

3.2.3 Engineering Feasibility

Though ambitious, the system is engineered with manageable complexity due to modularity:

1. **Audio Pipeline**

   – Whisper/Deepgram provide deterministic <5s latency, satisfying the non-functional latency requirements.

2. **Visual Pipeline**

   – Frames are captured at a low sampling rate (e.g., 0.5 FPS), ensuring DocLayout-YOLO and CLIP can run efficiently.

3. **Retrieval and Reasoning Pipeline**

   – Embeddings are indexed in memory, making lookup operations highly efficient [8].

   – Qwen-VL inference remains feasible with GPU-level optimization (FP16, model caching) [5].

4. **Real-time Communication**

   – LiveKit simplifies real-time media handling, eliminating the need to implement custom WebRTC logic.

5. **Privacy and Data Management**

   – Session-scoped retention is entirely feasible using Redis or in-memory vector stores.

Collectively, these components make the project's real-time requirements achievable within capstone constraints.

3.2.4 Academic Contribution

PARSE contributes academically in several ways:

- Demonstrates a working integration of **multimodal RAG** for real-time scenarios.

- Shows how VLMs can be grounded using timestamp-based and slide-based citations.
- Explores the use of layout detection to improve AI reasoning over slides.
- Provides a full end-to-end system that merges WebRTC, ML pipelines, and retrieval systems.

Such an implementation is uncommon in student projects and represents a meaningful academic contribution.

3.2.5 Academic Feasibility Conclusion

The system leverages modern, open-source AI research while remaining within manageable computational and implementation constraints. All core components are technically sound, well-documented, and compatible with real-time requirements. PARSE is therefore **academically feasible**, with strong potential for both practical impact and contribution to the field of multimodal AI systems.

5. Glossary

- **STT (Speech-to-Text):** Technology that converts spoken language into written text.
- **VLM (Vision-Language Model):** AI models designed to understand and generate content based on both visual and textual inputs (e.g., Qwen-VL).
- **RAG (Retrieval-Augmented Generation):** A technique to optimize LLM output by referencing an authoritative knowledge base outside its training data.
- **Latency:** The delay before a transfer of data begins following an instruction for its transfer.
- **LiveKit:** An open-source infrastructure for building real-time audio and video applications.

6. References

1. Lewis, P., et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 9459-9474.
2. Ji, Z., et al. (2023). "Survey of Hallucination in Natural Language Generation." *ACM Computing Surveys*, 55(12), 1-38.
3. Wang, L., et al. (2023). "DocLayout-YOLO: Enhancing Document Layout Analysis with Improved Feature Extraction.".
4. Radford, A., et al. (2021). "Learning Transferable Visual Models From Natural Language Supervision." *International Conference on Machine Learning (ICML)*, 8748-8763. (The CLIP Paper).
5. Bai, J., et al. (2023). "Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, and Generation." *arXiv preprint arXiv:2308.12966*.
6. Ramírez, S. (2023). *FastAPI: Modern Python Web Development*. O'Reilly Media. / *FastAPI Documentation*. https://fastapi.tiangolo.com/
7. LiveKit. (2024). *LiveKit Documentation: Real-time video and audio for developers*. https://docs.livekit.io/
8. Radford, A., et al. (2023). "Robust Speech Recognition via Large-Scale Weak Supervision." *International Conference on Machine Learning (ICML)*, 28492-28518. (The Whisper Paper).
9. IEEE. (1998). *IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)*. IEEE Computer Society.
10. Object Management Group (OMG). (2017). *OMG Unified Modeling Language (OMG UML), Version 2.5.1*.
11. Loreto, S., & Romano, S. P. (2014). *Real-Time Communication with WebRTC*. O'Reilly Media. / IETF RFC 7478: *Web Real-Time Communication Use Cases and Requirements*.