Bilkent University

Department of Computer Science

Senior Design Project - CS491

Analysis and Requirements Report

PARSE: Presentation Analysis and
Real-time Semantic Extraction

Group T2501

Fall 2025/2026

Team Members:

22203783	Anıl Kılıç
22202680	Aybars Buğra Aksoy
22202981	Barış Yaycı
22203661	Bora Yetkin
22201772	Eren Berk Eraslan

Supervisor: Ayşegül Dündar Boral
Innovation Expert: Sezai Artun Özyeğin

Table of Contents

Introduction

Purpose

This document presents a comprehensive analysis of the PARSE (Presentation Analysis and Real-Time Semantic Extraction), an interactive AI assistant designed to enhance online presentations by bridging the gap between passive viewing and active understanding.

Problem Statement

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.

Scope

PARSE is designed to address these limitations by:

- Continuously fusing verbal explanations with on-screen visual materials (text, formulas, tables, and images)
- Maintaining a live, semantic model of the presentation
- Allowing participants to ask natural-language questions at any time
- Providing real-time, grounded answers with citations to specific slides or timestamps
- Generating automated summaries and analytics following the session
- Ensuring privacy through session-scoped data retention

Definitions and Abbreviations

Definitions and abbreviations

Term	Definition
STT	Speech-to-Text technology
VLM	Vision-Language Model

Term	Definition
RAG	Retrieval-Augmented Generation
CLIP	Contrastive Language-Image Pre-training
WebRTC	Web Real-Time Communication
DTLS/SRTP	Datagram Transport Layer Security / Secure Real-time Transport Protocol
DocLayout-YOLO	Document layout detection model based on YOLO architecture
Qwen-VL	Vision-Language Model by Alibaba for multimodal reasoning
LiveKit	Open-source WebRTC infrastructure for real-time communication

Current System

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

Identified Market Gap

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.

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

Limitations Summary

Limitations of existing systems

Limitation	Impact
No visual content understanding	Cannot answer questions about charts, diagrams, formulas
No layout parsing	Cannot identify titles, tables, figures on slides
Audio-only processing	Misses critical visual information in presentations
No cross-modal alignment	Cannot connect spoken words to visual elements
Generic summaries	Lack context from visual materials

Proposed System

Overview

System 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 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. Following the session, PARSE generates automated summaries and analytics dashboards.

Architecture Overview

The system architecture is split into four main components:

1. **Multimodal ML Services Backend**: GPU-managed models including DocLayout-YOLO, OpenCLIP, and Qwen-VL for visual and language understanding
2. **Central Application Backend**: FastAPI-based server for user/session management, LiveKit token generation, and API orchestration
3. **Temporary Vector Store**: Session-scoped embeddings for text and image patches
4. **Web Client**: LiveKit-based audio/video interface with interactive Q&A

Technology Stack

Technology stack

Layer	Technologies
Frontend	Next.js, React, TailwindCSS, LiveKit JS SDK
Backend	FastAPI, Python 3.9+, PostgreSQL
Real-time Communication	LiveKit Server, WebRTC, Redis
Audio Processing	Deepgram / OpenAI Whisper (STT)
Visual Processing	DocLayout-YOLO, OpenCLIP (ViT-L-14)
Reasoning & Generation	Qwen2.5-VL-7B (4-bit quantization)
Deployment	Docker, Docker Compose

Functional Requirements

Session Management & Connectivity

- **FR-01 WebRTC Connection:** The system shall utilize LiveKit to establish a secure, low-latency WebRTC connection for both audio and video tracks. *(High Priority)*
- **FR-02 Host Mode:** The system shall support a "host" mode allowing users to initiate and manage presentation sessions. *(High Priority)*
- **FR-03 Participant Mode:** The system shall allow participants to join sessions via a unique room token or shareable link. *(High Priority)*
- **FR-04 Slide Detection:** The system shall detect active presentation slides versus webcam feeds to prioritize high-resolution processing for slides. *(High Priority)*

Multimodal Data Ingestion & Processing

- **FR-05 Audio Pipeline:** The system shall stream audio buffers to Deepgram or OpenAI Whisper APIs for real-time Speech-to-Text (STT) conversion. *(High Priority)*
- **FR-06 Frame Sampling:** The system shall sample video frames at a configurable rate (e.g., 0.5 FPS) to minimize redundant processing while capturing slide transitions. *(High Priority)*
- **FR-07 Layout Analysis:** The system shall apply DocLayout-YOLO to extracted frames to identify and segment text blocks, headers, figures, tables, and formulas. *(High Priority)*

- **FR-08 Embedding Generation:** The system shall generate vector embeddings for both transcribed text and visual segments using OpenCLIP to enable semantic search. *(High Priority)*

Interactive Q&A & Reasoning

- **FR-09 Chat Interface:** The system shall provide a chat interface allowing users to submit natural language queries during the live session. *(High Priority)*
- **FR-10 Retrieval:** Upon receiving a query, the system shall retrieve the top-k most relevant audio transcripts and visual frames from the session history. *(High Priority)*
- **FR-11 Generation:** The system shall pass the retrieved context and user query to the Qwen-VL model to generate a context-aware answer. *(High Priority)*
- **FR-12 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. *(High Priority)*

Post-Session Operations

- **FR-13 Summary Generation:** The system shall generate a concise textual summary of the presentation immediately after the session ends. *(Medium Priority)*
- **FR-14 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"). *(High Priority)*

ML Services

- **FR-15 Layout Detection:** The system shall use DocLayout-YOLO for real-time slide analysis to detect and extract layout elements including titles, text blocks, figures, tables, and formulas. *(High Priority)*
- **FR-16 Visual Encoding:** The system shall use OpenCLIP (ViT-L-14) to create embeddings for visual frames, enabling efficient retrieval by matching user queries with relevant visual context. *(High Priority)*
- **FR-17 VLM Reasoning:** The system shall use Qwen2.5-VL-7B with 4-bit quantization as the primary Vision-Language Model to synthesize answers by reasoning over both transcribed text and retrieved visual context. *(High Priority)*
- **FR-18 GPU Management:** The system shall implement efficient VRAM management with model loading/unloading to operate within 12GB GPU memory constraints. *(High Priority)*

Nonfunctional Requirements

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 (DocLayout-YOLO + OpenCLIP) must process a slide frame within 3 seconds to ensure the bot's knowledge base is current.

Scalability & Concurrency

- **NFR-04 Stateless Architecture:** 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 Concurrent Users:** The system shall support at least 50 concurrent users in a single LiveKit room without degradation in audio/video quality.

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.

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.
- **NFR-10 Session-Scoped Retention:** All audio, video, transcriptions, visual embeddings, and generated answers shall be processed ephemerally and discarded at the end of the session.

Usability

- **NFR-11 Browser Compatibility:** The user interface shall be responsive and accessible via standard modern web browsers (Chrome, Firefox, Edge) without requiring additional software installation.
- **NFR-12 UI Layout:** The Q&A panel must not obscure more than 20% of the presentation screen area on desktop devices.

Pseudo Requirements

Pseudo requirements are constraints imposed by the client or the environment that restrict the implementation of the system.

- **PR-01 Programming Language:** The backend must be implemented in Python 3.9+ to ensure compatibility with modern ML libraries and async frameworks.
- **PR-02 Frontend Framework:** The web client must use Next.js with React for server-side rendering and optimal performance.
- **PR-03 ML Framework:** All machine learning models must be compatible with PyTorch for consistent inference pipelines.
- **PR-04 Real-time Protocol:** The system must use WebRTC via LiveKit for real-time audio/video communication.
- **PR-05 Database:** PostgreSQL shall be used for persistent data storage, and Redis for session-scoped temporary data.
- **PR-06 GPU Requirements:** The system must run on NVIDIA GPUs with a minimum of 12GB VRAM to support the ML pipeline.
- **PR-07 Quantization:** The Vision-Language Model must use 4-bit quantization (BitsAndBytes) for memory efficiency.
- **PR-08 Open Source:** All ML models must be open-source with no dependency on proprietary APIs to minimize costs.
- **PR-09 Containerization:** All services must be containerized using Docker for consistent deployment environments.
- **PR-10 Browser Support:** The web client must support Chrome, Firefox, and Edge browsers without additional plugins.

System Models

Scenarios

Scenario 1: Host Starts a Presentation Session

1. Professor Dr. Smith logs into PARSE using university credentials
2. Dr. Smith clicks "Create New Session" and enters session name "CS101 - Lecture 5"
3. System creates a LiveKit room and generates a shareable link
4. Dr. Smith shares the link with students via email/LMS
5. Dr. Smith starts screen sharing to display lecture slides
6. The PARSE bot automatically joins and begins processing the video stream
7. DocLayout-YOLO detects slide elements (titles, text, figures, formulas)
8. OpenCLIP generates embeddings for each detected element

Scenario 2: Participant Asks a Question

1. Student Alice joins the session via the shared link
2. Alice watches the presentation and sees a complex diagram on slide 15
3. Alice types in the Q&A panel: "Can you explain what the arrows in the architecture diagram mean?"
4. PARSE encodes Alice's question using OpenCLIP
5. System retrieves the most relevant slide patches (diagram from slide 15)
6. Qwen-VL generates an answer based on the visual context and transcript
7. Alice receives: "The arrows in the architecture diagram represent data flow between components. The blue arrows show the request path from client to server, while the red arrows indicate response paths. [Slide 15, 00:23:45]"

Scenario 3: Late Joiner Catches Up

1. Student Bob joins the session 20 minutes late
2. Bob asks: "What topics have been covered so far?"
3. PARSE retrieves transcript summaries and key slide titles
4. System generates a summary: "So far, the lecture covered: (1) Introduction to System Architecture [Slides 1-5], (2) Component Design Patterns [Slides 6-12], (3) Current topic: Data Flow Diagrams [Slide 15]"

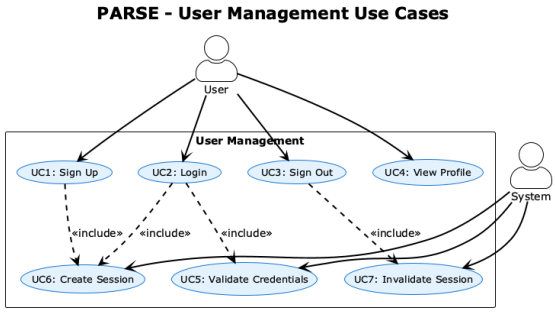
5. Bob can now follow along with context

Scenario 4: Session Ends with Summary

1. Dr. Smith clicks "End Session"
2. PARSE generates an automated summary of the lecture
3. Summary includes: key topics covered, frequently asked questions, and timestamps
4. All session data (embeddings, transcripts, screenshots) is permanently deleted
5. Only the summary (if opted-in) is retained

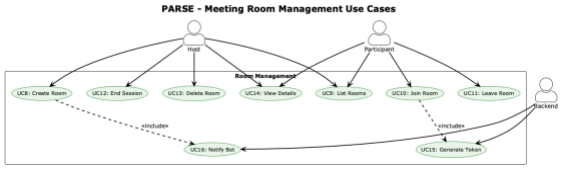
Use Case Model

User Management Use Cases



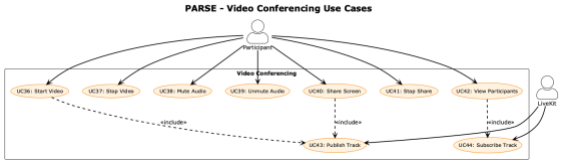
User Management Use Cases

Room Management Use Cases



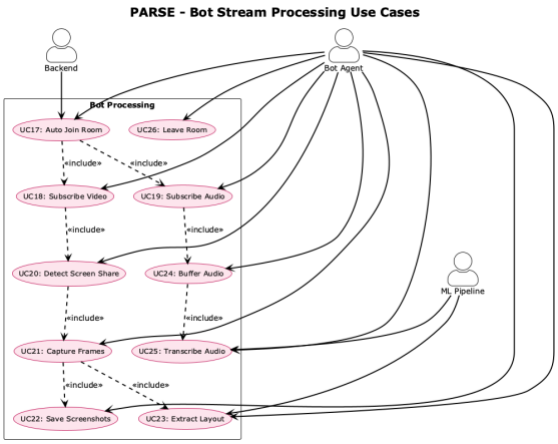
Room Management Use Cases

Video Conferencing Use Cases



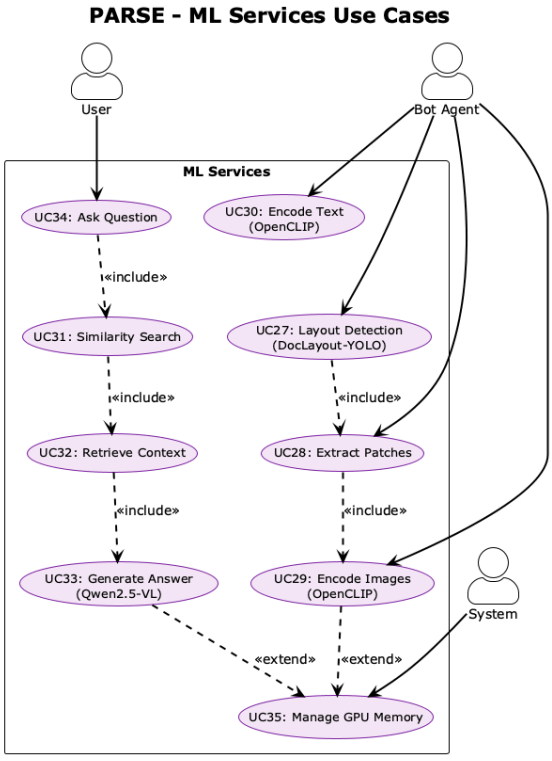
Video Conferencing Use Cases

Bot Processing Use Cases



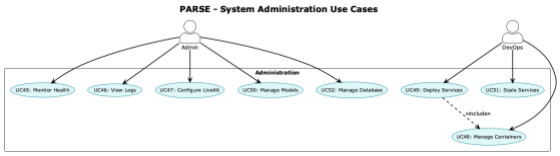
Bot Processing Use Cases

ML Services Use Cases



ML Services Use Cases

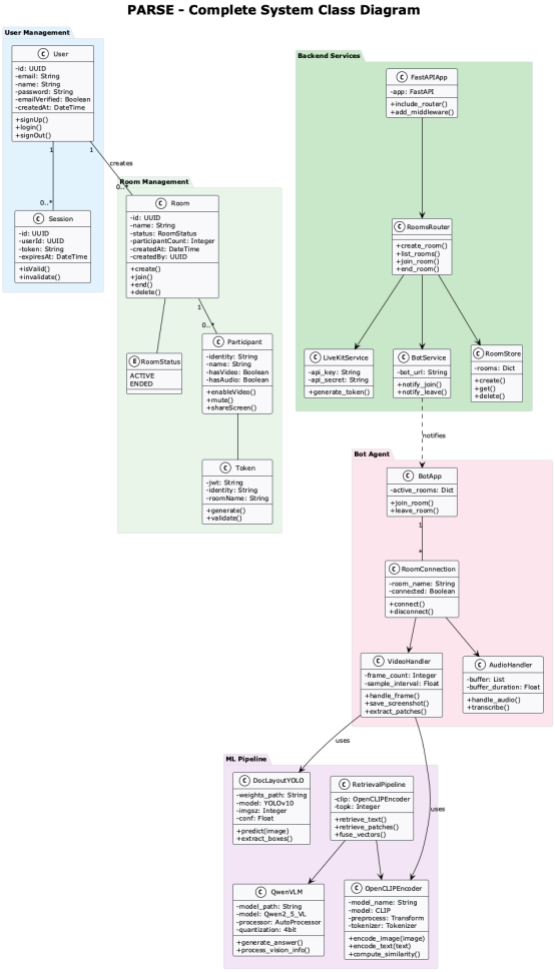
Administration Use Cases



System Administration Use Cases

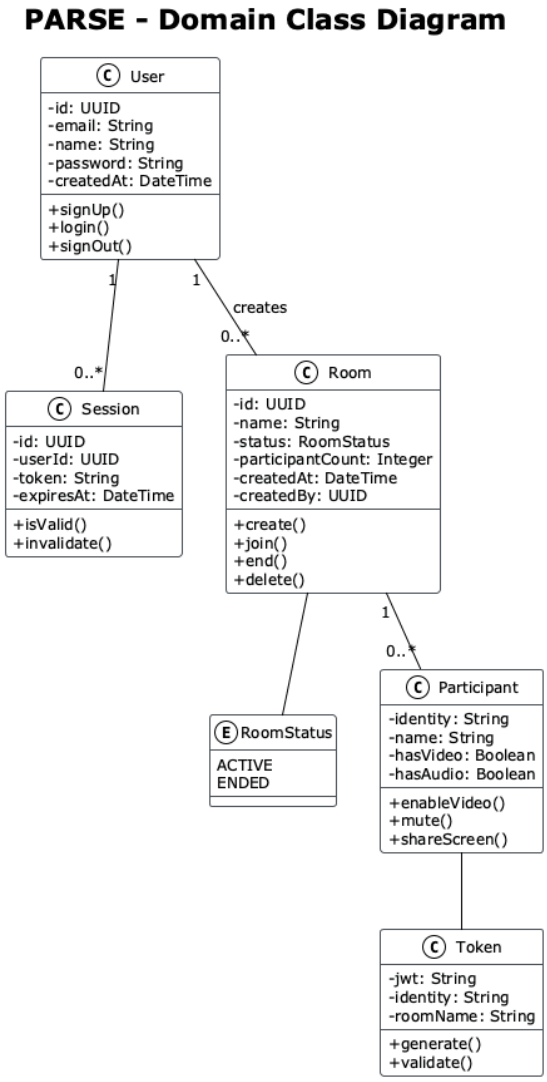
Class Model

Complete System Class Diagram



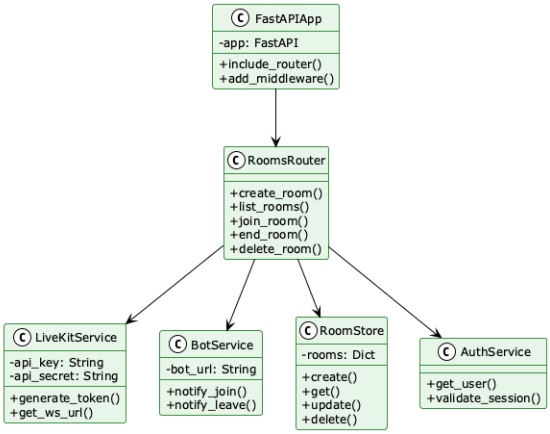
Complete System Class Diagram

Domain Classes



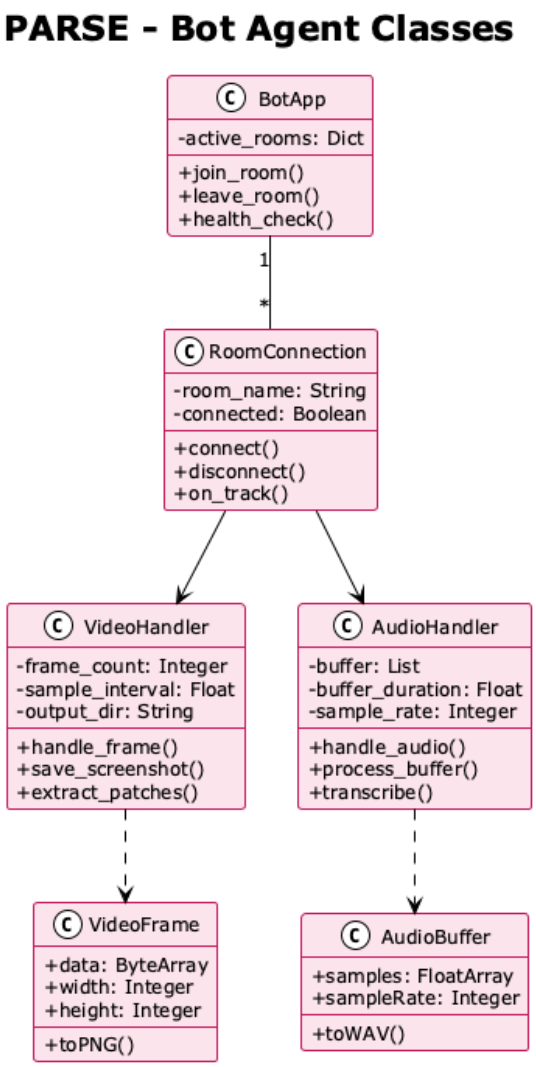
Domain Class Diagram

Backend Service Classes



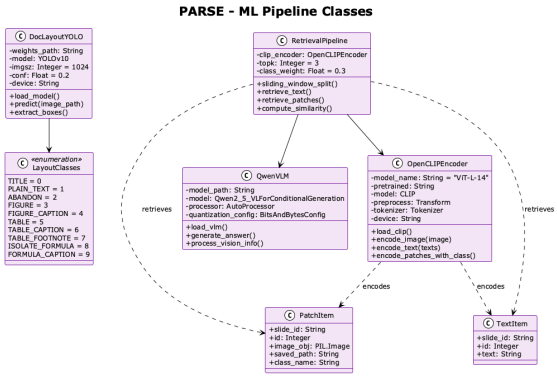
Backend Service Classes

Bot Agent Classes



Bot Agent Classes

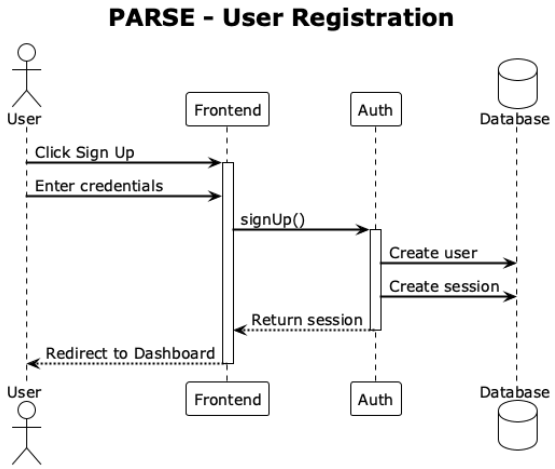
ML Pipeline Classes



ML Pipeline Classes (DocLayout-YOLO, OpenCLIP, Qwen-VL)

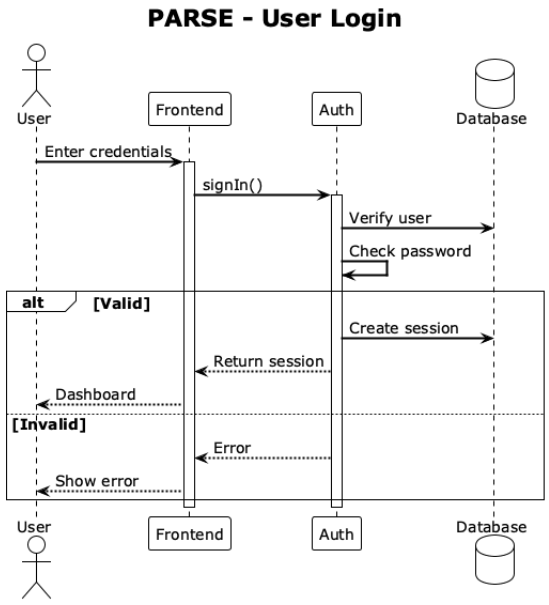
Dynamic Models

User Registration Sequence



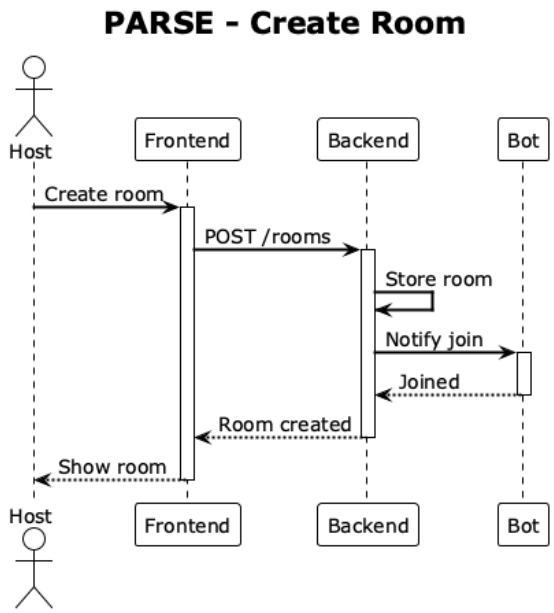
User Registration Sequence

User Login Sequence



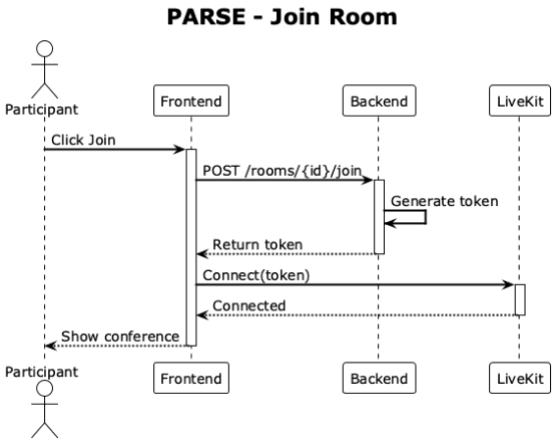
User Login Sequence

Create Room Sequence



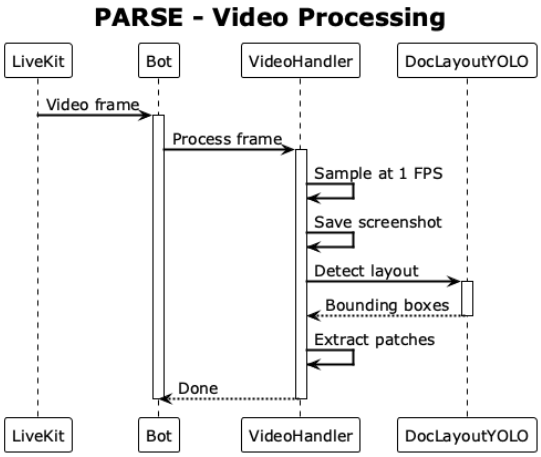
Create Room Sequence

Join Room Sequence



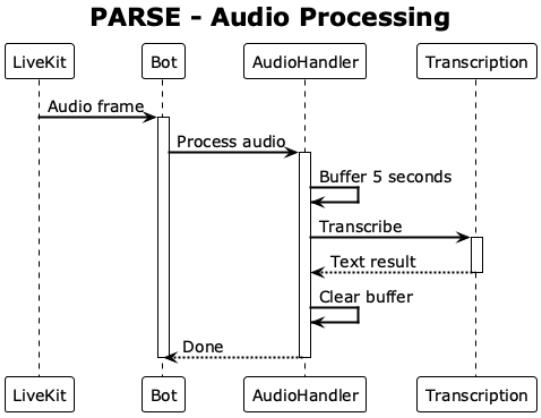
Join Room Sequence

Video Processing Sequence



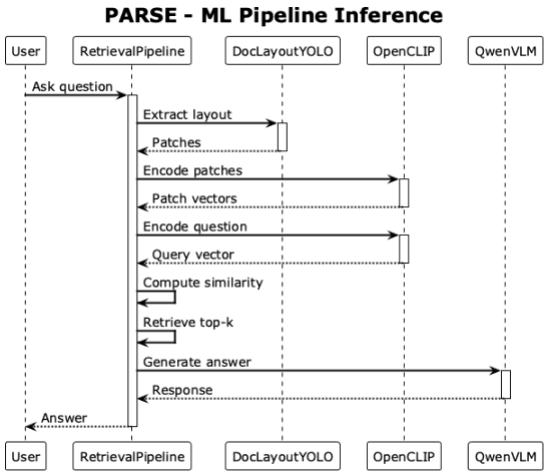
Video Processing Sequence

Audio Processing Sequence



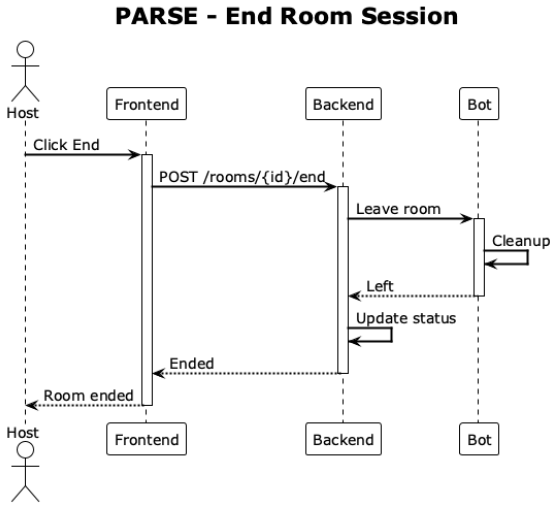
Audio Processing Sequence

ML Inference Sequence



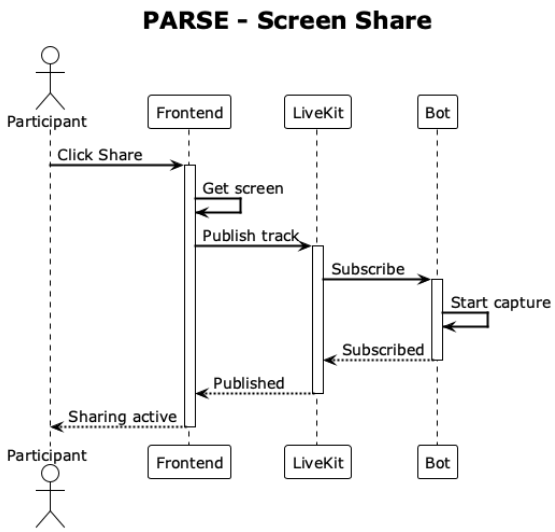
ML Pipeline Inference (YOLO → CLIP → Qwen-VL)

End Room Sequence



End Room Sequence

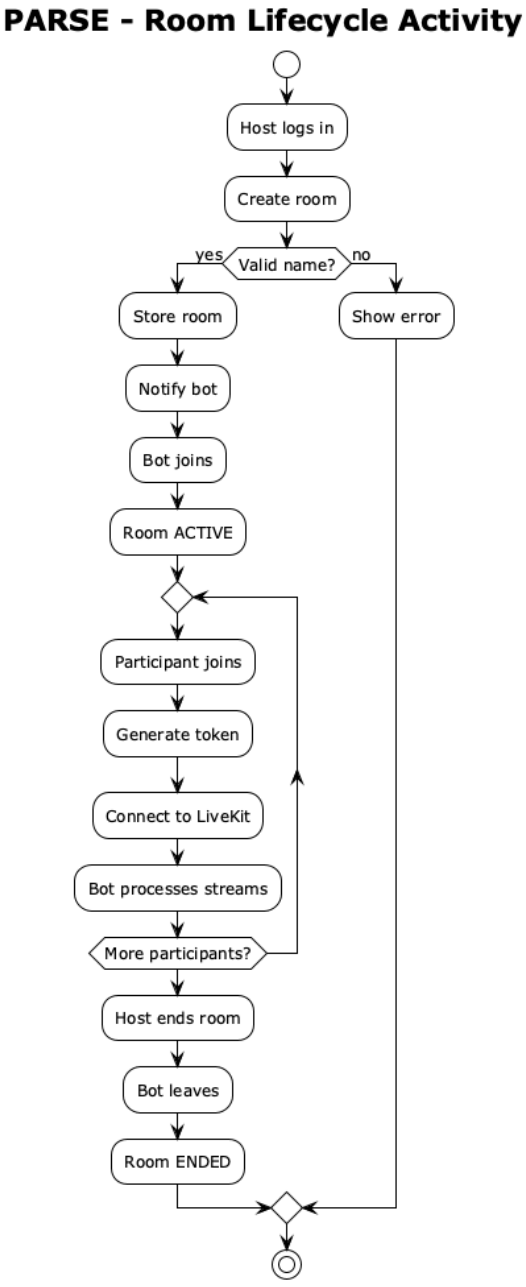
Screen Share Sequence



Screen Share Sequence

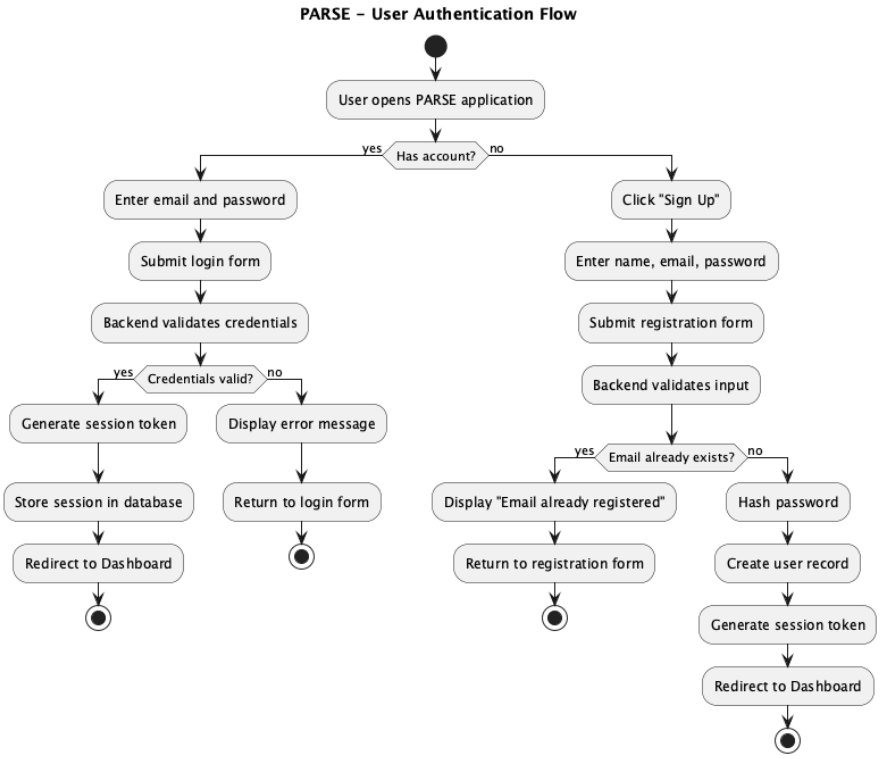
Behavioral Models

Room Lifecycle Activity Diagram



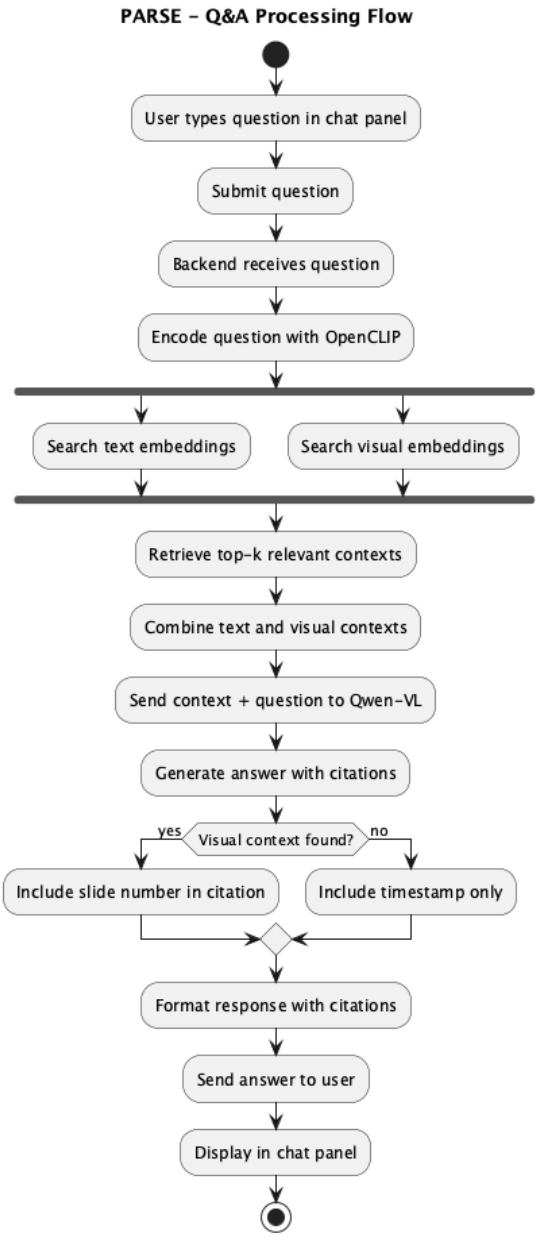
Room Lifecycle Activity Diagram

User Authentication Activity Diagram



User Authentication Flow

Q&A Processing Activity Diagram

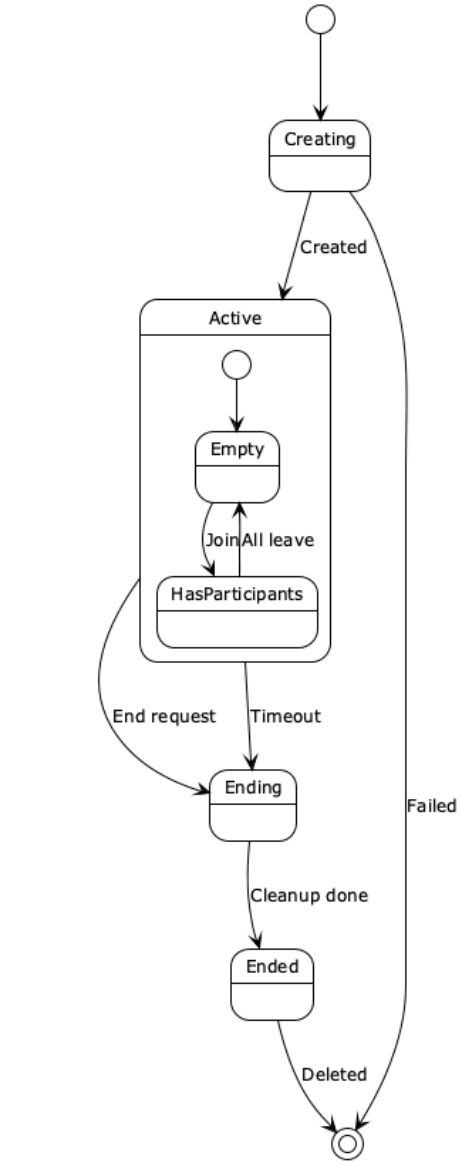


Q&A Processing Flow

ML Pipeline Activity Diagram

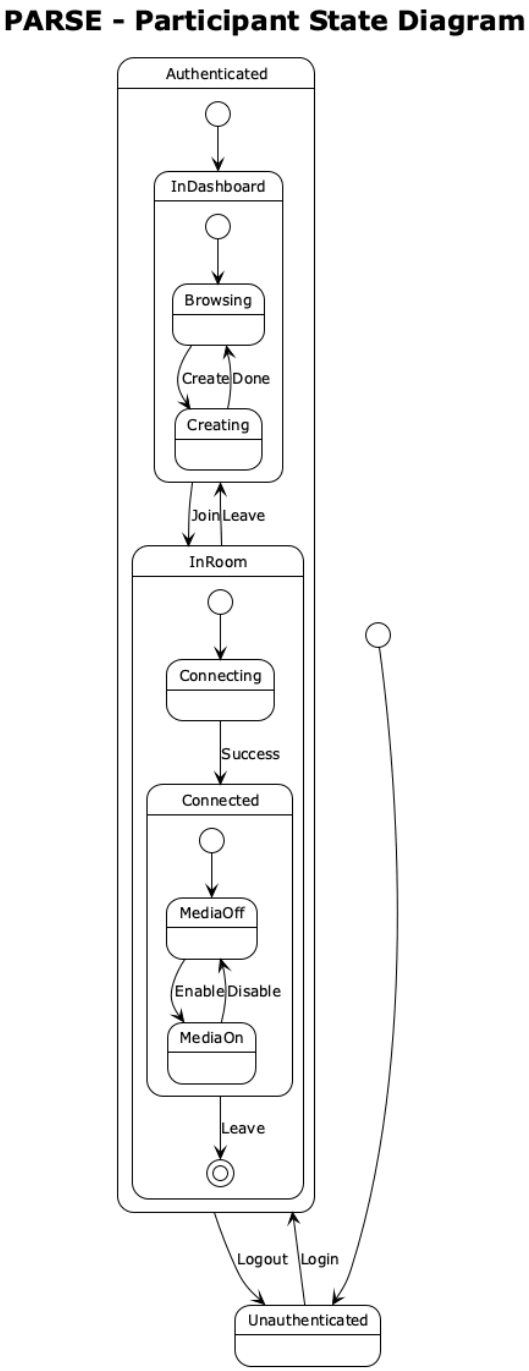
ML Pipeline Processing Flow

Room State Diagram



Room State Diagram

Participant State Diagram



Participant State Diagram

User Interface - Navigational Paths and Screen Mock-ups

Navigation Flow

The PARSE web application follows a simple navigation structure:

1. **Landing Page** → Login/Sign Up
2. **Dashboard** → Create Session / Join Session / View History
3. **Session View** → Video Panel + Q&A Panel + Participants List
4. **Q&A Panel** → Ask Question / View Answers / See Citations

Main Screen Components

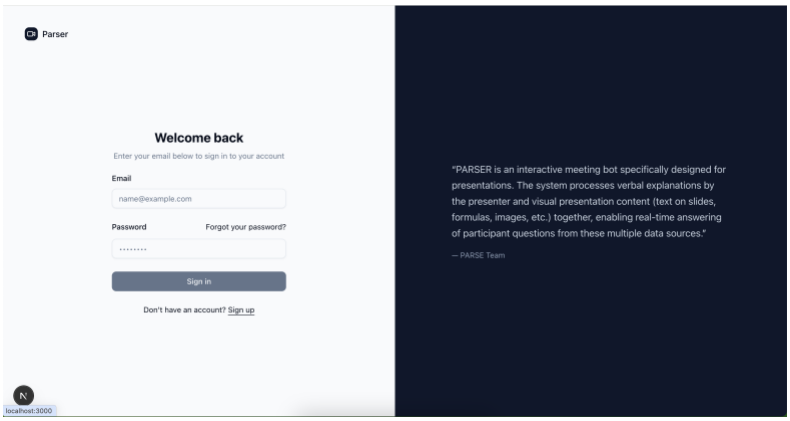
Screen components

Screen	Components
Login Page	Email input, Password input, Sign In button, Sign Up link
Dashboard	Session list, Create button, User profile dropdown
Session View	Video area (70%), Q&A sidebar (30%), Control bar (bottom)
Q&A Panel	Question input, Answer cards with citations, Scroll history
Control Bar	Mute, Video on/off, Screen share, End session, Settings

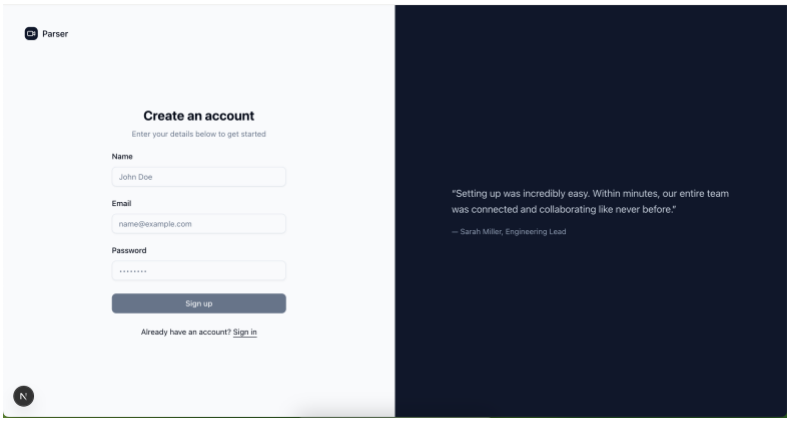
UI Design Principles

- **Minimal Obstruction**: Q&A panel occupies ≤30% of screen width
- **Clear Citations**: Each answer shows slide number and timestamp
- **Responsive Design**: Adapts to desktop and tablet screens
- **Accessibility**: Keyboard navigation, screen reader support
- **Real-time Feedback**: Loading indicators during ML inference

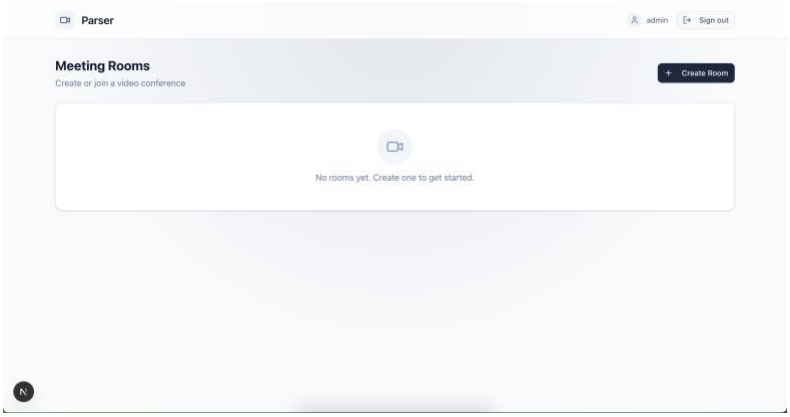
Screen Mock-ups



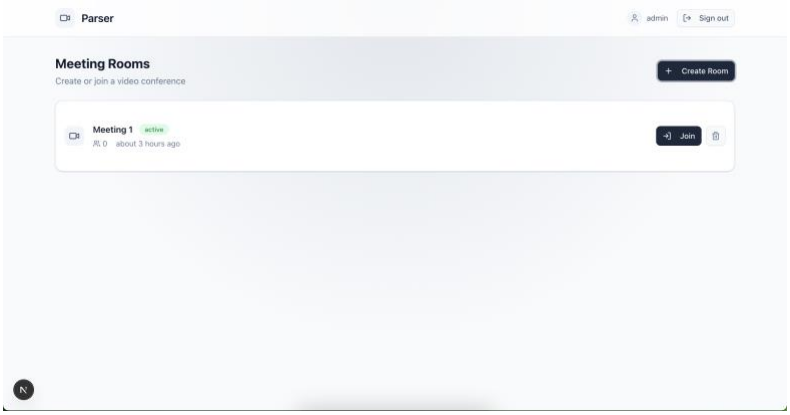
Login Screen



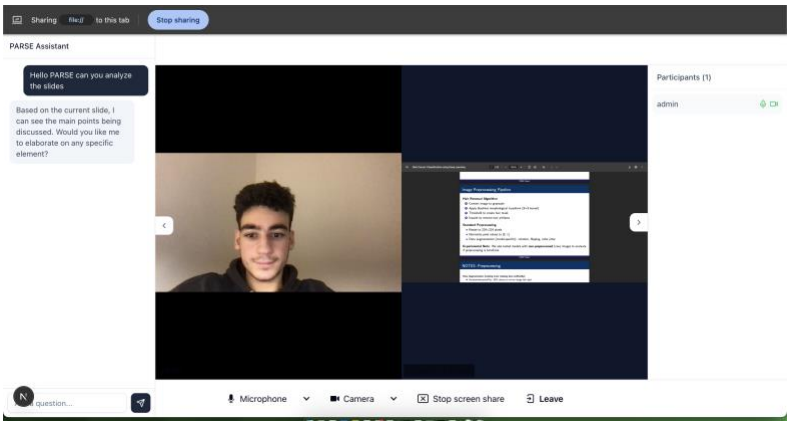
Registration Screen



Dashboard - Empty State



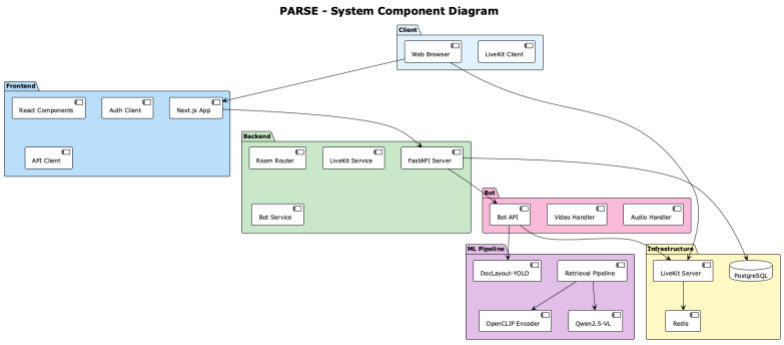
Dashboard - With Rooms



Meeting Room Screen

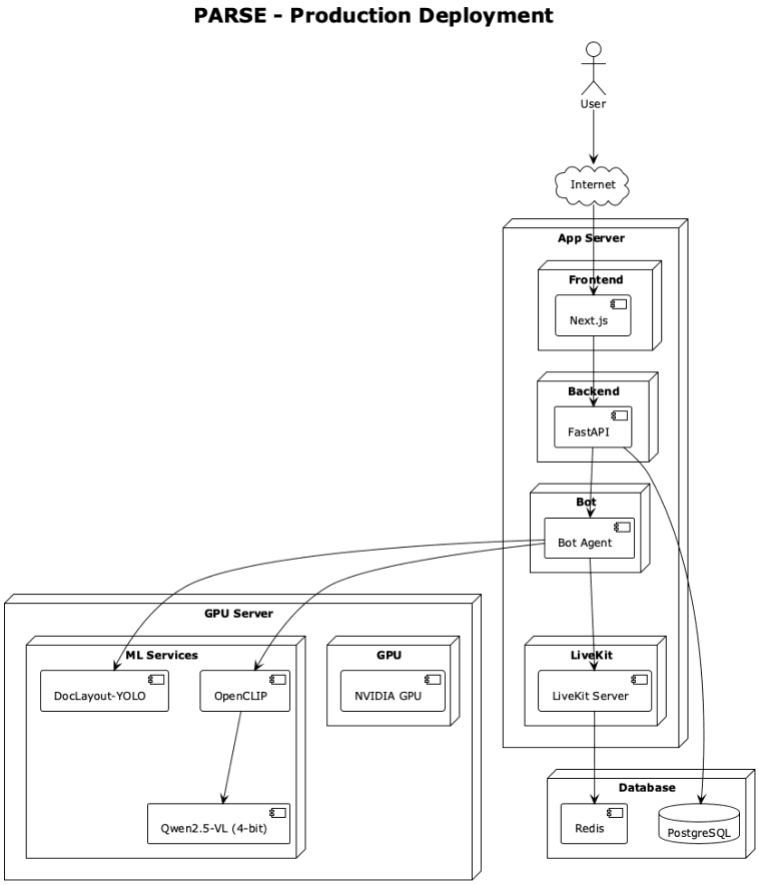
Component and Deployment Diagrams

Component Diagram



System Component Diagram

Production Deployment



Production Deployment Architecture

Other Analysis Elements

Consideration of Various Factors in Engineering Design

Constraints

Implementation Constraints

The implementation of PARSE involves several technical constraints:

- **Communication Infrastructure**: LiveKit handles WebRTC media streams including encoding, decoding, packet loss management, and synchronization
- **Backend Framework**: FastAPI enables non-blocking processing of concurrent audio and video frames
- **Audio Processing**: Deepgram/Whisper must operate under tight latency constraints (<5s)

- **Visual Processing**: DocLayout-YOLO and CLIP embeddings must be generated in near real-time
- **VLM Inference**: Qwen2.5-VL with 4-bit quantization for 12GB VRAM compatibility

Economic Constraints

PARSE is developed as a university capstone project with limited budget:

- Exclusively uses open-source models (Qwen-VL, DocLayout-YOLO, Whisper, CLIP)
- Eliminates dependency on costly proprietary APIs
- Requires careful GPU compute scheduling and batching

Ethical Constraints

PARSE enforces strict ethical constraints:

- **Consent**: Users must provide informed consent before processing
- **Session-Scoped Retention**: Zero persistent storage policy
- **Privacy Protections**: No data used for training or shared with third parties

Standards

Engineering standards

Standard	Application
IEEE 830-1998	Software Requirements Specifications
UML 2.5.1	System modeling (Sequence, Component diagrams)
RESTful API	Frontend-Backend communication
WebRTC	Real-time communication via LiveKit
DTLS/SRTP	Media stream encryption

Factors Impact Table

Factors impact assessment

Factor	Effect (0-10)	Justification
Public Health	4	Enables accessible remote education
Safety	5	Data security and privacy addressed
Welfare	6	Improves learning accessibility
Global	7	Cloud enables worldwide access
Cultural	5	Multi-language support possible
Social	8	Enables inclusive remote collaboration

Factor	Effect (0-10)	Justification
Environmental	6	Reduces travel carbon footprint
Economic	7	Cost-effective open-source solution

Risks and Alternatives

Risk Analysis

Risk analysis

ID	Risk	Prob.	Impact	Mitigation
R-001	GPU memory exhaustion	Med	High	4-bit quantization, model unloading
R-002	LiveKit failure	Low	Critical	Redis persistence, reconnection
R-003	VLM accuracy issues	Med	Med	Grounding with citations
R-004	Network latency	Med	High	Frame sampling optimization
R-005	Privacy breach	Low	Critical	Session-scoped retention
R-006	Model loading time	Med	Med	Model caching, preloading

B Plan - Alternatives

Alternative 1: Cloud VLM APIs

- Use GPT-4V or Claude Vision APIs
- Higher cost but no GPU management required

Alternative 2: Lighter Models

- Use smaller Qwen-VL variants (3B instead of 7B)
- Lower accuracy but faster inference

Alternative 3: Text-Only Fallback

- Gracefully degrade to transcript-based Q&A
- If visual processing fails, still provide value

Project Plan

Project Goals

1. Develop functional multimodal presentation assistant
2. Integrate real-time visual and audio processing
3. Implement grounded Q&A with citations

4. Ensure privacy through session-scoped data retention
5. Deploy production-ready system

Work Packages

Work packages with leaders and members

WP	Title	Leader	Members	Deliverables
WP1	Frontend	Barış Y.	Anıl K., Bora Y.	Next.js app, LiveKit UI, Q&A panel
WP2	Backend	Aybars B.	Eren B., Barış Y.	FastAPI server, APIs, Auth
WP3	Bot Agent	Eren B.	Aybars B., Bora Y.	Stream processor, handlers
WP4	ML Pipeline	Anıl K.	Eren B., Barış Y.	YOLO, CLIP, Qwen-VL integration
WP5	Vector Store	Bora Y.	Anıl K., Aybars B.	Embeddings, retrieval, cleanup
WP6	DevOps	Aybars B.	Bora Y., Eren B.	Docker, CI/CD, deployment
WP7	Testing/Docs	Barış Y.	All members	Tests, reports, documentation

Work Package Schedule

Work package schedule

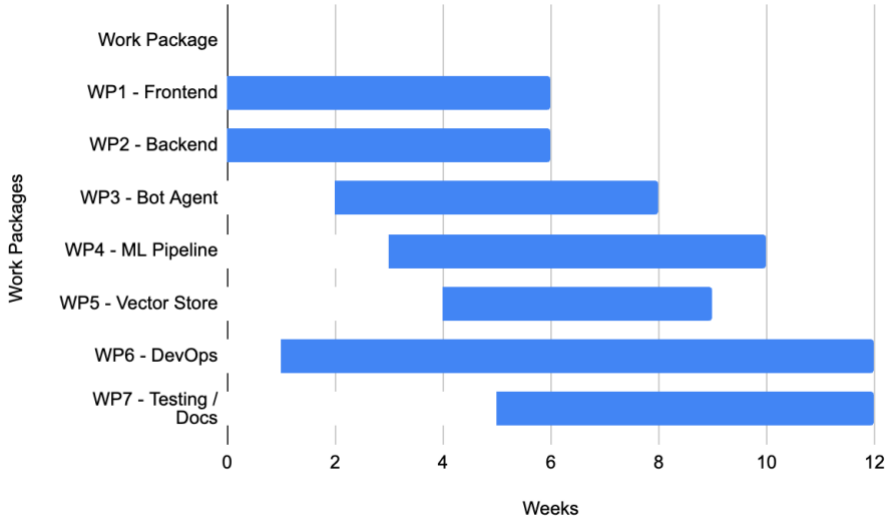
WP	Start	End	Duration
WP1 - Frontend	Week 1	Week 6	6 weeks
WP2 - Backend	Week 1	Week 6	6 weeks
WP3 - Bot Agent	Week 3	Week 8	6 weeks
WP4 - ML Pipeline	Week 4	Week 10	7 weeks
WP5 - Vector Store	Week 5	Week 9	5 weeks
WP6 - DevOps	Week 2	Week 12	11 weeks
WP7 - Testing/Docs	Week 6	Week 12	7 weeks

Milestones

Project milestones

ID	Milestone	Week	Deliverable
M1	Analysis Complete	Week 2	Analysis and Requirements Report
M2	High-Level Design	Week 4	High-Level Design Report
M3	Core Demo	Week 6	Working video conference + bot
M4	ML Integration	Week 8	Full ML pipeline operational
M5	Low-Level Design	Week 9	Low-Level Design Report
M6	Beta Release	Week 10	Feature-complete system
M7	Final Demo	Week 12	Production-ready deployment
M8	Final Report	Week 12	Final Project Report

Gantt Chart



The project schedule is managed using GitHub Projects, which provides a timeline view for tracking work packages and milestones. The Gantt chart is maintained in the GitHub repository and updated weekly during sprint planning meetings.

Key timeline highlights:

- **Weeks 1-4**: Foundation phase (Frontend, Backend, Analysis)
- **Weeks 5-8**: Integration phase (Bot, ML Pipeline, Vector Store)
- **Weeks 9-12**: Refinement phase (Testing, Documentation, Deployment)

Project Management Tools

Selected Tools:

1. **GitHub**: Source code repository, issues, CI/CD
2. **Notion**: Documentation, knowledge base, meeting notes
3. **Discord**: Team communication

Ensuring Proper Teamwork

Collaboration Strategy

1. Rotate work package leadership
2. Regular standup meetings
3. All PRs require peer review

4. Pair programming for critical features
5. Weekly progress presentations

Ethics and Professional Responsibilities

Ethical considerations

Concern	Mitigation
User Privacy	Session-scoped retention, no persistent storage
Data Security	DTLS/SRTP encryption, secure authentication
Transparency	Users informed they're interacting with AI
Accuracy	Grounded answers with citations to reduce hallucination
Consent	Clear consent before audio/video processing
Confidentiality	No data used for training or shared externally

Planning for New Knowledge and Learning Strategies

Learning plan

Area	Current	Target	Strategy
LiveKit SDK	None	Proficient	Documentation, tutorials
Vision-Language Models	Basic	Advanced	Research papers, hands-on
CLIP Embeddings	Basic	Proficient	OpenCLIP documentation
Document Layout	None	Intermediate	DocLayout-YOLO papers
4-bit Quantization	None	Proficient	BitsAndBytes documentation

Glossary

Term	Definition
CLIP	Contrastive Language-Image Pre-training model for cross-modal embeddings
DocLayout-YOLO	Document layout detection model for identifying slide elements
FastAPI	Modern Python web framework for building APIs
Grounding	Technique to base AI answers on source evidence
LiveKit	Open-source WebRTC infrastructure for real-time communication
OpenCLIP	Open-source implementation of CLIP
Qwen-VL	Vision-Language Model capable of understanding images and text
RAG	Retrieval-Augmented Generation
Session-Scoped	Data retention only for duration of active session

Term	Definition
STT	Speech-to-Text transcription
VLM	Vision-Language Model
WebRTC	Web Real-Time Communication protocol

References

1. Bruegge, B., & Dutoit, A. H. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall.
2. Radford, A., et al. (2021). "Learning Transferable Visual Models From Natural Language Supervision." *ICML 2021*. (CLIP)
3. Zhao, Y., et al. (2024). "DocLayout-YOLO: Enhancing Document Layout Analysis through Diverse Synthetic Data and Global-to-Local Adaptive Perception." *arXiv preprint*.
4. Bai, J., et al. (2023). "Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond." *arXiv preprint*.
5. LiveKit Documentation. https://docs.livekit.io/
6. FastAPI Documentation. https://fastapi.tiangolo.com/
7. OpenCLIP. https://github.com/mlfoundations/open_clip
8. IEEE 830-1998. *Recommended Practice for Software Requirements Specifications*.
9. UML 2.5.1 Specification. https://www.omg.org/spec/UML/2.5.1
10. WebRTC Standard. https://webrtc.org/