PARSE

Presentation Analysis and Real-time Semantic Extraction

Detailed Design Report

Date: April 27, 2026

Team PARSE – T2501

22203783 Anıl Kılıç – DevOps and Infrastructure Lead
22202680 Aybars Buğra Aksoy – Backend Development Lead
22202981 Barış Yaycı – ML/AI Integration Lead
22203661 Bora Yetkin – Frontend Development Lead
22201772 Eren Berk Eraslan – Zoom and ML/AI Lead

Department of Computer Engineering
Bilkent University

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose of the System

PARSE (Presentation Analysis and Real-time Semantic Extraction) is a multi-platform intelligent meeting assistant designed to revolutionize how users interact with and learn from presentations. The system addresses the growing need for accessible, AI-powered tools that can process real-time video and audio streams during meetings, providing instant transcription, slide analysis, and context-aware question answering.

1.1.1 Problem Statement

In today's digital age, remote meetings and presentations have become ubiquitous. However, participants face several challenges:

- **Information Overload**: Difficulty in capturing and retaining key information during lengthy presentations
- **Accessibility Barriers**: Hearing-impaired individuals lack real-time transcription
- **Engagement Gaps**: Participants who miss parts of a presentation cannot easily catch up
- **Post-Meeting Review**: No intelligent way to query presentation content after the fact
- **Multi-Platform Fragmentation**: Different meeting platforms (Zoom, LiveKit) require separate solutions

1.1.2 Solution Overview

PARSE provides a unified solution that:

1. Integrates with multiple meeting platforms (LiveKit for self-hosted, Zoom for enterprise)
2. Provides real-time speech-to-text transcription using Deepgram Nova 3
3. Automatically detects and processes presentation slides using computer vision (YOLO, CLIP)
4. Enables AI-powered question answering about presentation content using Qwen2.5-VL
5. Stores slide embeddings for semantic search and retrieval

1.2 Design Goals

The following design goals guided the architectural decisions for PARSE:

Table 1.1: PARSE Design Goals and Priorities

Design Goal	Description	Priority
Usability	Intuitive interface requiring minimal setup; seamless integration with existing meeting workflows	High
Performance	Real-time processing with <3 second latency for transcription; <5 second response for Q&A	High
Reliability	99.5% uptime; graceful degradation when components fail	High
Scalability	Support concurrent meetings; horizontal scaling of processing nodes	Medium
Security	End-to-end encryption; secure authentication; GDPR compliance	High
Extensibility	Modular architecture allowing easy addition of new ML models and platforms	Medium
Maintainability	Clean separation of concerns; comprehensive logging; containerized deployment	Medium
Modularity	Independent microservices with well-defined APIs	High
Flexibility	Support for both cloud and on-premises deployment	Medium
Aesthetics	Modern, professional UI consistent with enterprise standards	Low

1.3 Definitions, Acronyms, and Abbreviations

Table 1.2: Definitions and Acronyms

Term	**Definition**
PARSE	Presentation Analysis and Real-time Semantic Extraction
LiveKit	Open-source, self-hosted WebRTC platform for real-time communication
RTMS	Real-Time Media Streams (Zoom's API for accessing meeting media)
WebRTC	Web Real-Time Communication protocol for peer-to-peer audio/video
STT	Speech-to-Text conversion
VLM	Vision Language Model (AI model that processes both images and text)
YOLO	You Only Look Once – real-time object detection algorithm
CLIP	Contrastive Language-Image Pre-training (OpenAI's vision-language model)
pgvector	PostgreSQL extension for vector similarity search
JWT	JSON Web Token for secure authentication
OAuth	Open Authorization protocol for secure delegated access
API	Application Programming Interface
CUDA	Compute Unified Device Architecture (NVIDIA GPU computing)

1.4 Overview

This Detailed Design Report is organized as follows:

- **Chapter 2**: Analyzes current software architecture including competitors and alternative solutions
- **Chapter 3**: Presents the proposed software architecture with detailed subsystem decomposition
- **Chapter 4**: Documents the services provided by each subsystem
- **Chapter 5**: Defines functional and non-functional test cases
- **Chapter 6**: Discusses engineering design considerations including constraints and standards
- **Chapter 7**: Details teamwork organization and collaboration
- **Chapter 8**: Provides a glossary of terms
- **Chapter 9**: Lists all references used

Chapter 2

Current Software Architecture

2.1 Market Analysis

The meeting assistant market has grown significantly, with several competitors offering various features. This section analyzes existing solutions to position PARSE's unique value proposition.

2.2 Competitor Analysis

2.2.1 Otter.ai

Overview: Cloud-based AI meeting assistant focusing on transcription and note-taking.

Features:

- Real-time transcription
- Speaker identification
- Meeting summaries
- Integration with Zoom, Google Meet, Microsoft Teams

Limitations:

- No slide analysis capability
- No context-aware Q&A
- Cloud-only (no self-hosted option)
- Subscription-based pricing

2.2.2 Microsoft Copilot in Teams

Overview: AI assistant integrated into Microsoft Teams.

Features:

- Meeting transcription
- Action item extraction
- Post-meeting summaries
- Integration with Microsoft 365

Limitations:

- Limited to Microsoft ecosystem
- No real-time slide analysis
- Enterprise-only pricing
- No open-source option

2.2.3 Zoom AI Companion

Overview: Native AI features within Zoom platform.

Features:

- Meeting summaries
- Smart recording highlights
- Chat compose assistance

Limitations:

- Limited to Zoom platform
- No slide content analysis
- No semantic Q&A capability
- Requires Zoom paid plans

2.3 Comparative Analysis

Table 2.1: Feature Comparison Matrix

Feature	PARSE	Otter.ai	MS Copilot	Zoom AI
Real-time Transcription	✓	✓	✓	✓
Slide Detection	✓	×	×	×
Slide Content Analysis	✓	×	×	×
Context-aware Q&A	✓	×	Partial	×
Multi-platform Support	✓	✓	×	×
Self-hosted Option	✓	×	×	×
Open Source	✓	×	×	×
Vector Semantic Search	✓	×	×	×

2.4 PARSE Differentiators

PARSE's unique value proposition includes:

1. **Intelligent Slide Processing**: Using DocLayout YOLO and OpenCLIP for semantic understanding of presentation content

2. **Vision Language Model Q&A**: Qwen2.5-VL enables contextual questions about slide content
3. **Multi-Platform Architecture**: Single backend supports both LiveKit (self-hosted) and Zoom (enterprise)
4. **Privacy-First Design**: Self-hosted option keeps all data on-premises
5. **Vector-Based Retrieval**: pgvector enables semantic search across all processed slides

Chapter 3

Proposed Software Architecture

3.1 Overview

PARSE follows a microservices architecture with clear separation of concerns. The system is composed of six primary subsystems that communicate via well-defined REST APIs and WebSocket connections.

3.1.1 Architectural Style

The system employs a **layered microservices architecture** with the following characteristics:

- **Presentation Layer**: Next.js frontend and Zoom sidebar app
- **API Gateway Layer**: FastAPI backend handling authentication and routing
- **Processing Layer**: Bot service for media stream processing
- **Intelligence Layer**: GPU server running ML models
- **Data Layer**: PostgreSQL with pgvector extension

3.1.2 System Architecture Diagram

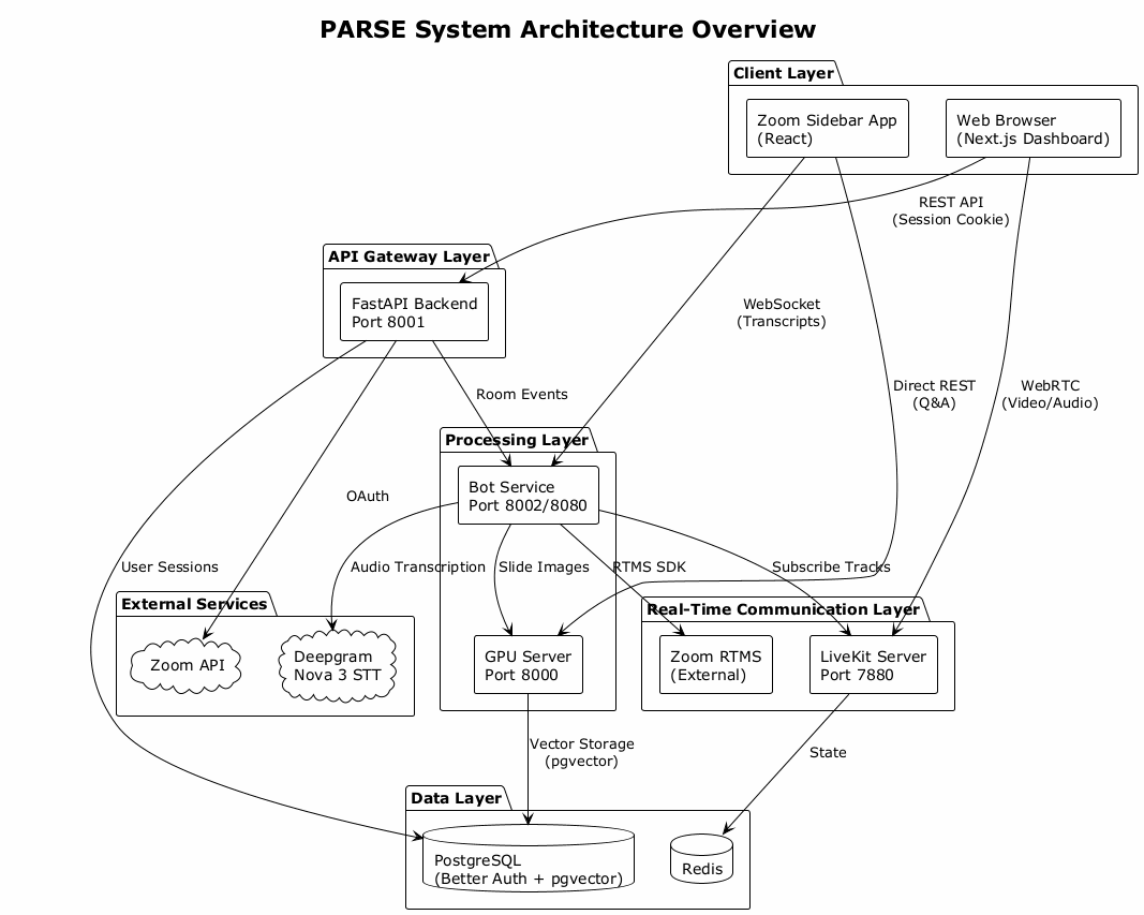

Figure 3.1: PARSE System Architecture Overview

3.2 Subsystem Decomposition

The PARSE system is decomposed into the following subsystems:

3.2.1 Frontend Subsystem (Next.js)

Purpose: Provides the user interface for room management, meeting participation, and viewing transcriptions.

Technology Stack:

- Next.js 16 with App Router
- React 19
- Better Auth for authentication
- LiveKit React SDK
- TailwindCSS v4

Key Components:

- `Dashboard Page`: Room listing and creation
- `Room Page`: LiveKit/Zoom meeting interface
- `Transcription Panel`: Real-time transcript display
- `Auth Pages`: Login and signup forms
- `API Proxy`: Secure proxy to backend with session validation

3.2.2 Backend Subsystem (FastAPI)

Purpose: Handles authentication, room management, and orchestrates communication between services.

Technology Stack:

- FastAPI (Python async framework)
- asyncpg for database access
- httpx for async HTTP client
- LiveKit Python SDK

Key Services:

- `LiveKitService`: JWT token generation for room access
- `BotService`: Bot notification and coordination
- `ZoomService`: Zoom OAuth and SDK signature generation
- `RoomStore`: In-memory room state management
- `AuthService`: Session validation with Better Auth

3.2.3 Bot Subsystem

Purpose: Connects to meeting rooms and processes audio/video streams.

Technology Stack:

- FastAPI (dual ports: 8002 API, 8080 RTMS webhook)
- LiveKit Python SDK
- Zoom RTMS SDK
- Deepgram API client
- PIL for image processing

Key Handlers:

- `AudioHandler`: Buffers audio, sends to Deepgram, publishes transcripts
- `VideoHandler`: Captures screenshare frames for LiveKit
- `ZoomHandler`: RTMS processing with slide change detection

3.2.4 GPU Server Subsystem

Purpose: Runs ML models for slide analysis and question answering.

Technology Stack:

- FastAPI
- PyTorch with CUDA
- DocLayout YOLO for object detection
- OpenCLIP for embeddings
- Qwen2.5-VL for vision-language Q&A

Key Components:

- `SlideProcessor`: YOLO detection + CLIP embedding
- `QAEngine`: Retrieval + VLM inference

3.2.5 Zoom App Subsystem

Purpose: React application embedded in Zoom's sidebar for transcript viewing and Q&A.

Technology Stack:

- React 18
- Zoom Meeting SDK
- WebSocket client

3.2.6 Data Subsystem

Purpose: Persistent storage for user sessions, slide embeddings, and Q&A history.

Technology Stack:

- PostgreSQL 15
- pgvector extension for vector similarity search
- Better Auth schema for user management

3.2.7 Component Diagram

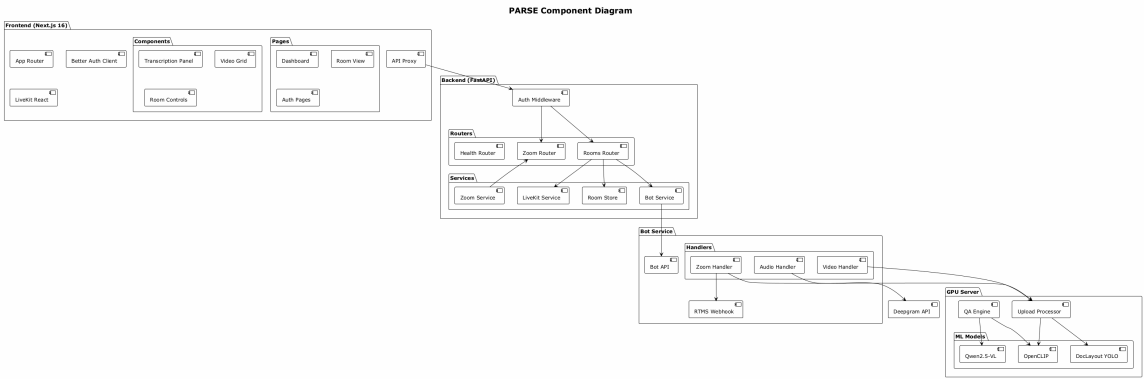

Figure 3.2: PARSE Component Diagram

3.3 Hardware/Software Mapping

3.3.1 Deployment Architecture

PARSE is designed for deployment on a single server with GPU capability, using Docker Compose for containerization.

Table 3.1: Hardware Requirements

Component	Minimum	Recommended
CPU	8 cores	16 cores
RAM	16 GB	32 GB
GPU	NVIDIA T4 (16GB)	NVIDIA RTX 4090 (24GB)
Storage	100 GB SSD	500 GB NVMe SSD
Network	100 Mbps	1 Gbps

3.3.2 Port Allocation

Table 3.2: Service Port Allocation

Port	Service	Purpose
3000	Frontend	Next.js dashboard
3001	Zoom App	React Zoom sidebar
5432	PostgreSQL	Database with pgvector
6379	Redis	LiveKit state storage
7880	LiveKit	WebRTC signaling
7881-7882	LiveKit	UDP media ports
8000	GPU Server	ML inference (direct, not Docker)
8001	Backend	FastAPI REST API
8002	Bot	Bot REST API
8080	Bot RTMS	Zoom RTMS webhook

3.3.3 Deployment Diagram

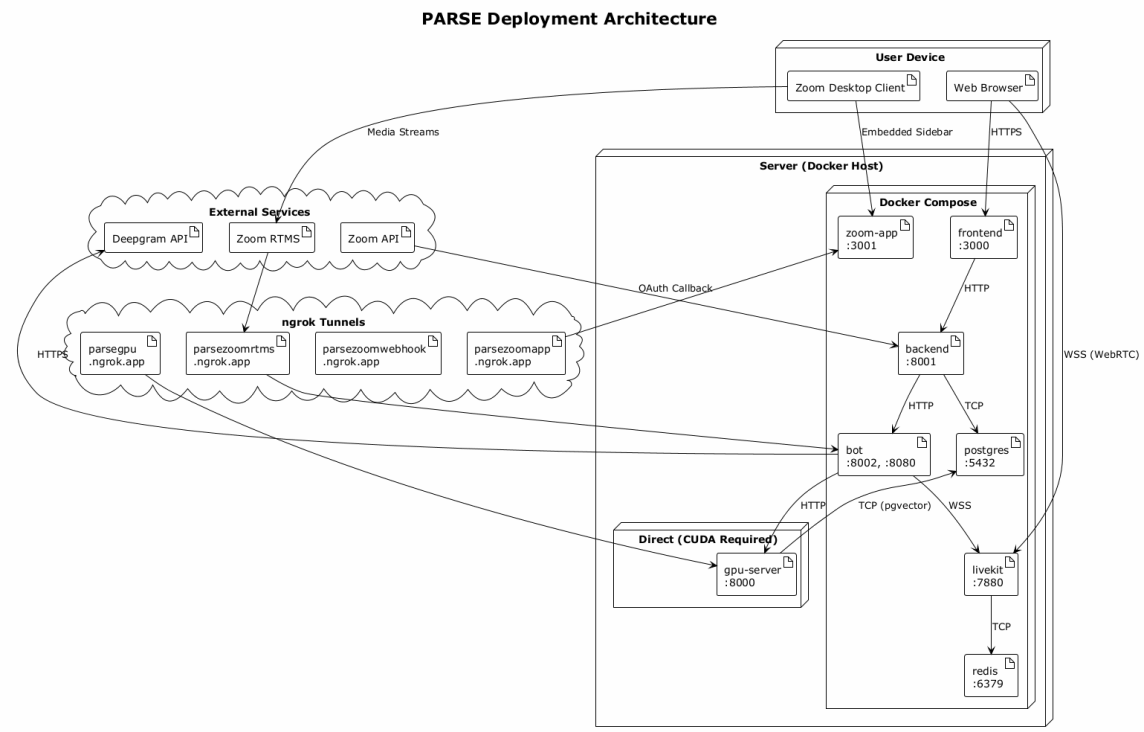

Figure 3.3: PARSE Deployment Architecture

3.4 Persistent Data Management

3.4.1 Database Schema

PARSE uses PostgreSQL with the pgvector extension for efficient vector similarity search. The schema is divided into two logical groups:

Authentication Tables (Better Auth)

- `user`: User accounts with email, name, and metadata
- `session`: Active session tokens with expiry
- `account`: OAuth provider tokens (Zoom integration)
- `verification`: Email verification codes

ML/Analytics Tables

```
CREATE TABLE slide_patches (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    meeting_id VARCHAR(255) NOT NULL,
    slide_id VARCHAR(255) NOT NULL,
    patch_index INTEGER NOT NULL,
    class_name VARCHAR(100),  -- 'text', 'figure', 'table', etc.
```

```
    image_path VARCHAR(500),
    embedding VECTOR(768),     -- CLIP embedding
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_patches_meeting ON slide_patches(meeting_id);
CREATE INDEX idx_patches_embedding ON slide_patches
    USING ivfflat (embedding vector_cosine_ops);
```

Listing 3.1: Slide Patches Table Schema

```
CREATE TABLE qa_history (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    meeting_id VARCHAR(255) NOT NULL,
    session_id UUID NOT NULL,
    question TEXT NOT NULL,
    answer TEXT NOT NULL,
    confidence FLOAT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_qa_meeting ON qa_history(meeting_id);
CREATE INDEX idx_qa_session ON qa_history(session_id);
```

Listing 3.2: Q&A History Table Schema

3.4.2 Entity-Relationship Diagram

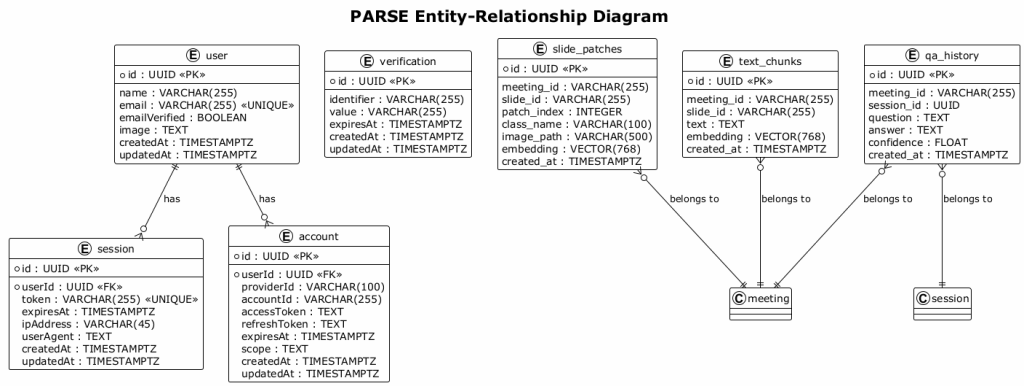

Figure 3.4: PARSE Database Entity-Relationship Diagram

3.5 Access Control and Security

3.5.1 Authentication Flow

PARSE implements a multi-layered authentication system:

1. **User Authentication**: Better Auth manages user sessions with httpOnly cookies
2. **API Proxy Security**: Next.js API routes validate sessions before forwarding to backend

3. **Backend Validation**: FastAPI middleware verifies proxy secret and user headers
4. **LiveKit Tokens**: JWT tokens with specific video grants per user role

3.5.2 Security Measures

Table 3.3: Security Implementation Details

Security Aspect	**Implementation**
Session Management	Better Auth with PostgreSQL-backed sessions; httpOnly cookies prevent XSS
API Security	Proxy secret header (`X-Proxy-Secret`) validates internal requests
Token Security	LiveKit JWTs expire after 6 hours; include specific room permissions
OAuth Security	Zoom OAuth tokens stored server-side; refresh before 5-minute expiry
Transport Security	All external communication over HTTPS/WSS
Data Isolation	Meeting data isolated by `meeting_id`; no cross-meeting access

3.5.3 LiveKit Token Grants

```
# User Token (can publish video/audio, subscribe to all)
user_grants = VideoGrants(
    room=room_name,
    room_join=True,
    can_publish=True,
    can_subscribe=True,
    can_publish_data=True
)

# Bot Token (subscribe only, publish transcription data)
bot_grants = VideoGrants(
    room=room_name,
    room_join=True,
    can_publish=False,      # Cannot send video/audio
    can_subscribe=True,     # Can receive all tracks
    can_publish_data=True,  # Can send transcriptions
    hidden=True             # Not visible in participant list
)
```

Listing 3.3: Token Generation with Video Grants

Chapter 4

Subsystem Services

This chapter documents the API contracts and services provided by each subsystem.

4.1 Backend API Services

4.1.1 Room Management API

Table 4.1: Room Management Endpoints

Method	Endpoint	Description
POST	/api/rooms	Create a new room
GET	/api/rooms	List all rooms for current user
GET	/api/rooms/{room_id}	Get room details
POST	/api/rooms/{room_id}/join	Join room and get access token
POST	/api/rooms/{room_id}/end	End the room session
DELETE	/api/rooms/{room_id}	Delete a room

Create Room Request/Response

```
# Request
POST /api/rooms
{
    "name": "Team Meeting",
    "platform": "livekit"  # or "zoom"
}

# Response (201 Created)
{
    "id": "room_abc123",
    "name": "Team Meeting",
    "platform": "livekit",
    "creator_id": "user_xyz",
    "created_at": "2024-01-15T10:30:00Z",
    "livekit_room_name": "parse-room_abc123"
}
```

Listing 4.1: Create Room API Contract

Join Room Request/Response

```
# Request
POST /api/rooms/{room_id}/join
```

```

# Response (200 OK)
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "ws_url": "wss://livekit.example.com:7880"
}
```

Listing 4.2: Join Room API Contract

4.1.2 Zoom Integration API

Table 4.2: Zoom Integration Endpoints

Method	Endpoint	Description
GET	/api/zoom/authorize	Initiate Zoom OAuth flow
GET	/api/zoom/callback	Handle OAuth callback
GET	/api/zoom/token/{user_id}	Get user's access token
GET	/api/zoom/status/{user_id}	Check authorization status
POST	/api/zoom/refresh/{user_id}	Refresh access token
DELETE	/api/zoom/revoke/{user_id}	Revoke authorization

4.2 Bot API Services

4.2.1 LiveKit Integration

Table 4.3: Bot LiveKit Endpoints

Method	Endpoint	Description
POST	/join-room	Bot joins a LiveKit room
DELETE	/leave-room/{room_name}	Bot leaves a room
GET	/status	Get bot service status

4.2.2 Zoom RTMS Integration

Table 4.4: Bot Zoom Endpoints

Method	Endpoint	Description
POST	/zoom/join	Register meeting for RTMS
DELETE	/zoom/leave/{meeting_id}	Stop RTMS processing
GET	/zoom/meetings	List active RTMS meetings
WS	/zoom/ws/{room_id}	WebSocket for transcripts

4.3 GPU Server API Services

4.3.1 Slide Processing

```
# Request
POST /upload-and-process
Content-Type: multipart/form-data
- file: <slide_image.png>
- meeting_id: "zoom-meeting-123"
- slide_id: "slide_001"

# Response (200 OK)
{
    "status": "success",
    "slide_id": "slide_001",
    "patches": [
        {
            "index": 0,
            "class": "text",
            "bbox": [10, 20, 300, 100],
            "confidence": 0.95
        },
        {
            "index": 1,
            "class": "figure",
            "bbox": [50, 150, 400, 350],
            "confidence": 0.89
        }
    ],
    "knowledge_base": {
        "total_patches": 5,
        "embedded": true
    }
}
```

Listing 4.3: Upload and Process Slide API

4.3.2 Question Answering

```
# Request
POST /ask
{
    "question": "What is the main topic of slide 2?",
    "meeting_id": "zoom-meeting-123",
    "session_id": "550e8400-e29b-41d4-a716-446655440000"  # optional
}

# Response (200 OK)
{
    "answer": "Slide 2 discusses the system architecture...",
    "session_id": "550e8400-e29b-41d4-a716-446655440000",
    "sources": {
        "image_patches": ["slide_002_patch_0", "slide_002_patch_1"],
        "text_snippets": ["The architecture consists of..."]
    },
    "confidence": 0.92
}
```

Listing 4.4: Question Answering API

4.4 Sequence Diagrams

4.4.1 Room Creation and Join Flow

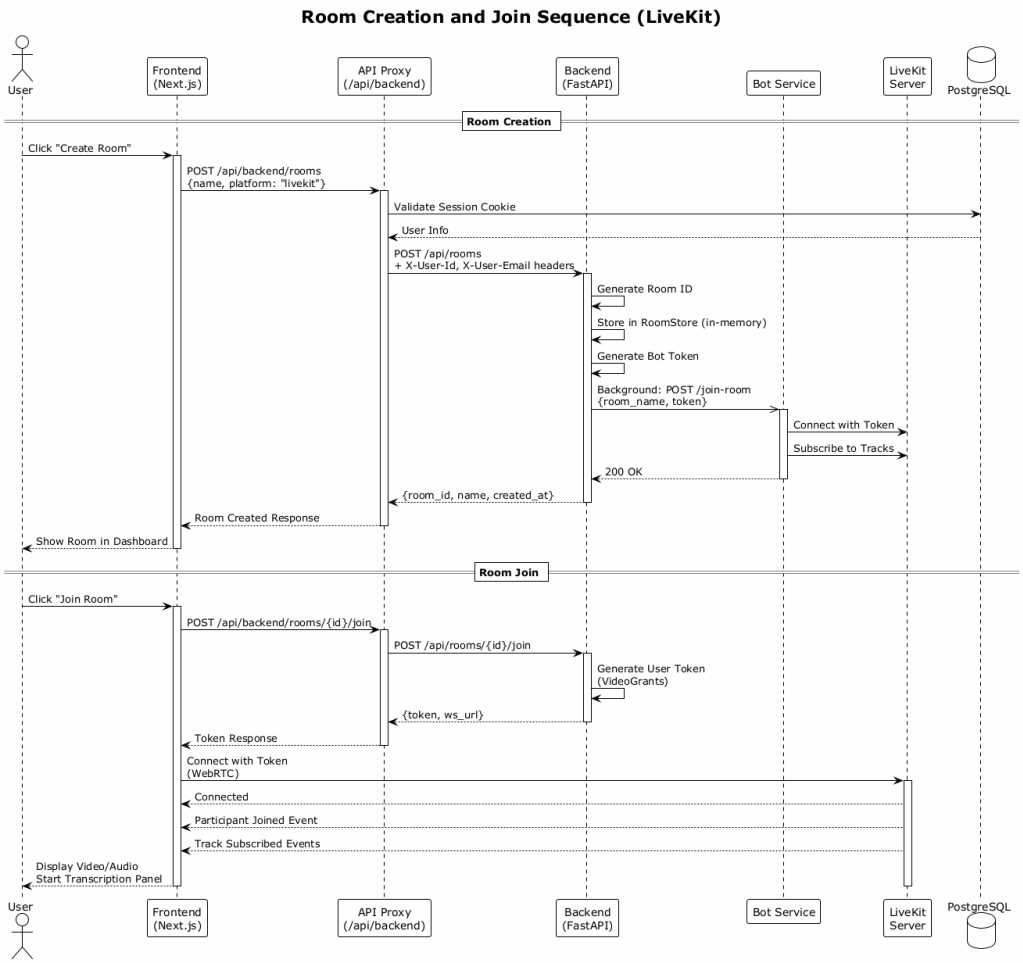

Figure 4.1: Room Creation and Join Sequence

4.4.2 Slide Processing and Q&A Flow

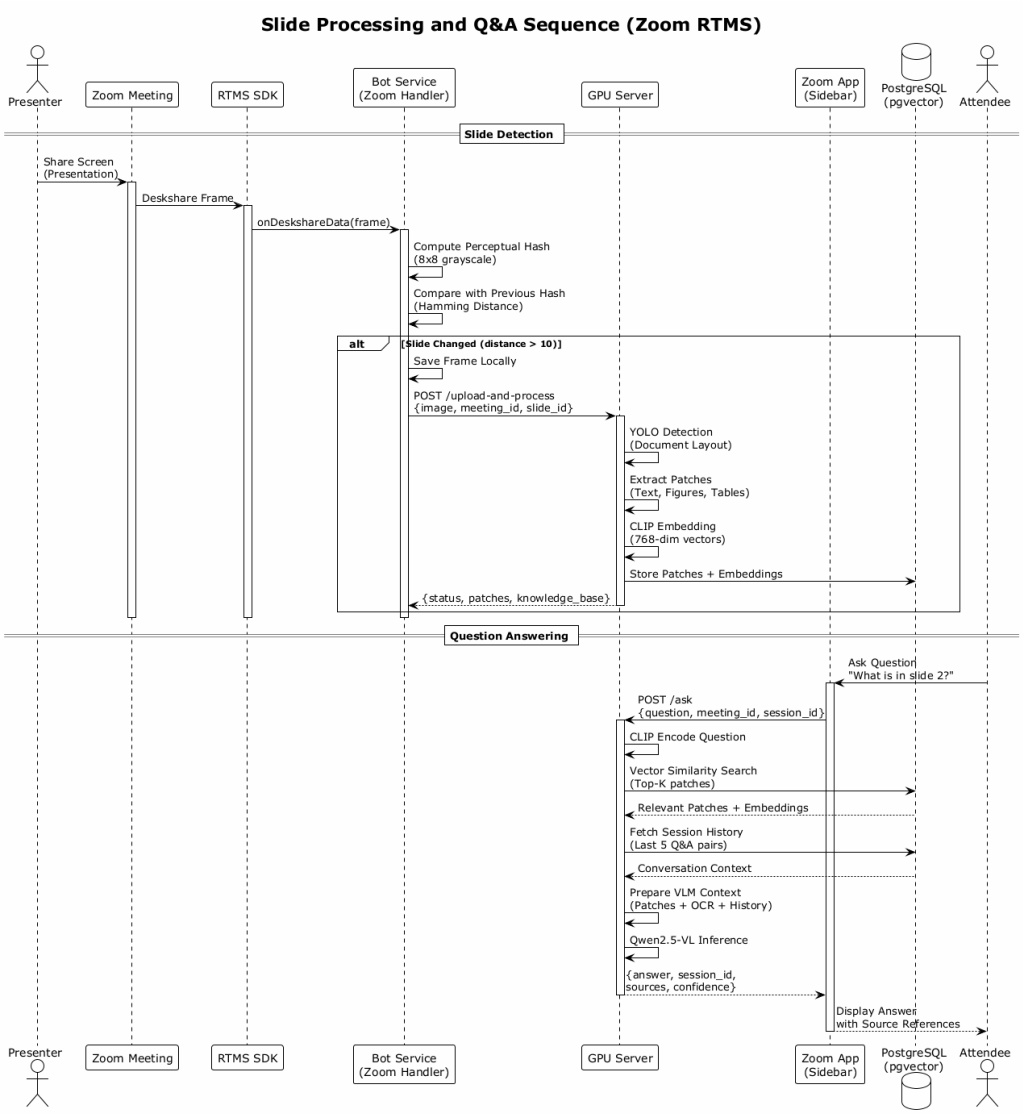

Figure 4.2: Slide Processing and Q&A Sequence

4.5 Class Diagram

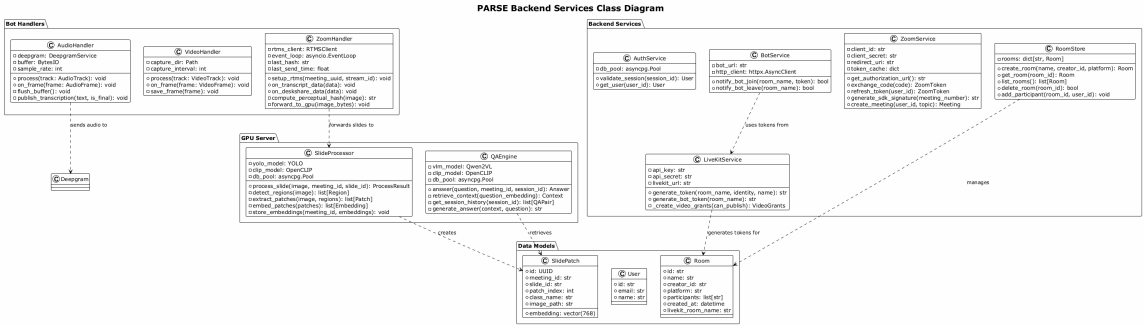

Figure 4.3: PARSE Backend Services Class Diagram

4.6 Data Flow Diagram

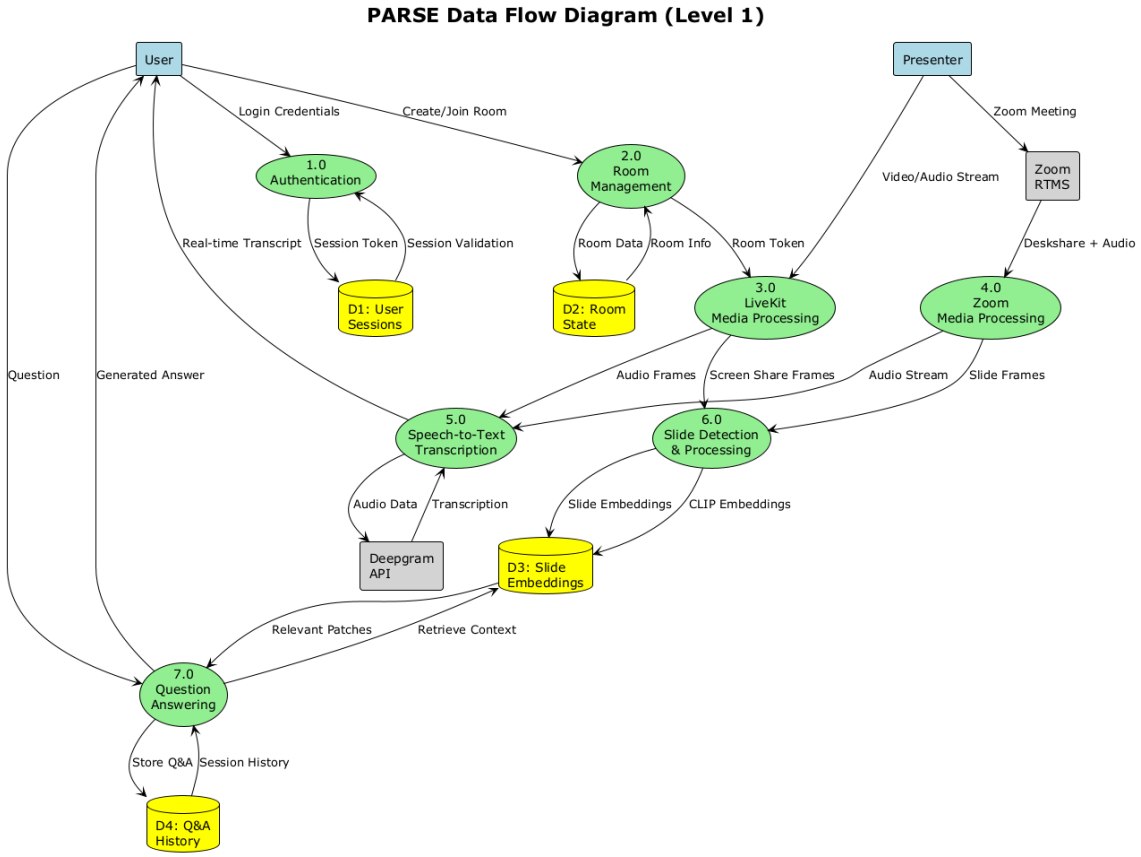

Figure 4.4: PARSE Data Flow Diagram (Level 1)

Chapter 5

Test Cases

This chapter presents the functional and non-functional test cases to be executed during and after the implementation phase. Test cases are designed for manual execution by software test engineers without access to source code.

5.1 Functional Test Cases

5.1.1 Authentication Module

Table 5.1: TC-AUTH-001: User Registration

Test ID	TC-AUTH-001
Category	Functional, Integration
Objective	Verify that a new user can successfully register with valid credentials
Priority	Critical
Procedure	1. Navigate to the signup page (/signup) 2. Enter a valid email address (e.g., test@example.com) 3. Enter a valid password (minimum 8 characters, with uppercase and number) 4. Enter the same password in the confirmation field 5. Enter a display name 6. Click the “Sign Up” button 7. Wait for the system response
Expected Result	• User is redirected to the dashboard page • A success message is displayed • User session is created (cookie set) • User can access protected routes
Date Tested	
Result	

Table 5.2: TC-AUTH-002: User Login

Test ID	TC-AUTH-002
Category	Functional, Integration
Objective	Verify that an existing user can log in with valid credentials
Priority	Critical

Procedure	1. Navigate to the login page (/login) 2. Enter a registered email address 3. Enter the correct password 4. Click the “Login” button 5. Wait for the system response
Expected Result	• User is redirected to the dashboard • Session cookie is set • User name is displayed in the header
Date Tested	
Result	

Table 5.3: TC-AUTH-003: Invalid Login Attempt

Test ID	TC-AUTH-003
Category	Functional, Security
Objective	Verify that login fails with incorrect credentials
Priority	Critical
Procedure	1. Navigate to the login page 2. Enter a valid email address 3. Enter an incorrect password 4. Click the “Login” button
Expected Result	• User remains on login page • Error message “Invalid credentials” is displayed • No session cookie is set
Date Tested	
Result	

Table 5.4: TC-AUTH-004: User Logout

Test ID	TC-AUTH-004
Category	Functional, Integration
Objective	Verify that a user can successfully log out
Priority	Major
Procedure	1. Log in with valid credentials 2. Navigate to any authenticated page 3. Click the user profile dropdown 4. Click the “Logout” button
Expected Result	• User is redirected to login page • Session cookie is invalidated • Attempting to access dashboard redirects to login
Date Tested	
Result	

Table 5.5: TC-AUTH-005: Session Persistence

Test ID	TC-AUTH-005
Category	Functional, Integration
Objective	Verify that user session persists across browser refresh
Priority	Major
Procedure	1. Log in with valid credentials 2. Note the dashboard is accessible 3. Refresh the browser (F5 or Ctrl+R) 4. Observe the page after refresh
Expected Result	• User remains logged in after refresh • Dashboard content is displayed correctly • No re-authentication required
Date Tested	
Result	

5.1.2 Room Management Module

Table 5.6: TC-ROOM-001: Create LiveKit Room

Test ID	TC-ROOM-001
Category	Functional, Integration
Objective	Verify that a user can create a new LiveKit room
Priority	Critical
Procedure	1. Log in to the application 2. Navigate to the dashboard 3. Click the “Create Room” button 4. Enter a room name (e.g., “Team Standup”) 5. Select “LiveKit” as the platform 6. Click “Create”
Expected Result	• New room appears in the room list • Room shows “LiveKit” badge • Room shows creator’s name • Creation timestamp is displayed
Date Tested	
Result	

Table 5.7: TC-ROOM-002: Create Zoom Room

Test ID	TC-ROOM-002
Category	Functional, Integration
Objective	Verify that a user can create a Zoom-integrated room

Priority	Critical
Procedure	1. Log in to the application 2. Ensure Zoom account is connected (via OAuth) 3. Navigate to the dashboard 4. Click “Create Room” 5. Enter room name 6. Select “Zoom” as the platform 7. Click “Create”
Expected Result	• New room appears with “Zoom” badge • Zoom meeting is created via API • Meeting join URL is displayed
Date Tested	
Result	

Table 5.8: TC-ROOM-003: Join LiveKit Room

Test ID	TC-ROOM-003
Category	Functional, Integration
Objective	Verify that a user can join an existing LiveKit room
Priority	Critical
Procedure	1. Log in to the application 2. Navigate to the dashboard 3. Locate a LiveKit room in the list 4. Click “Join” button on the room card 5. Allow camera/microphone permissions when prompted 6. Wait for connection
Expected Result	• User is redirected to room view page • Local video preview is displayed • Connection status shows “Connected” • Controls (camera, mic, share) are visible • Transcription panel is visible on the left
Date Tested	
Result	

Table 5.9: TC-ROOM-004: Leave Room

Test ID	TC-ROOM-004
Category	Functional, Integration
Objective	Verify that a user can leave a room cleanly
Priority	Major

Procedure	1. Join an existing room 2. Verify connection is established 3. Click the “Leave” button (red phone icon) 4. Confirm leave action if prompted
Expected Result	• User is disconnected from the room • User is redirected to dashboard • Camera and microphone are released • Other participants see user leave
Date Tested	
Result	

Table 5.10: TC-ROOM-005: Delete Room

Test ID	TC-ROOM-005
Category	Functional, Integration
Objective	Verify that a room creator can delete their room
Priority	Major
Procedure	1. Log in as the room creator 2. Navigate to dashboard 3. Locate the room to delete 4. Click the delete icon on the room card 5. Confirm deletion in the dialog
Expected Result	• Room is removed from the list • Success message is displayed • Room is no longer accessible via URL • Any active participants are disconnected
Date Tested	
Result	

Table 5.11: TC-ROOM-006: List User Rooms

Test ID	TC-ROOM-006
Category	Functional
Objective	Verify that dashboard displays all rooms correctly
Priority	Major
Procedure	1. Log in to the application 2. Navigate to dashboard 3. Observe the room list
Expected Result	• All created rooms are displayed • Each room shows name, platform, creator, and timestamp • Rooms are sorted by creation date (newest first) • Empty state shown if no rooms exist

Date Tested	
Result	

5.1.3 Video/Audio Module

Table 5.12: TC-AV-001: Camera Toggle

Test ID	TC-AV-001
Category	Functional, Component
Objective	Verify that user can toggle camera on/off during meeting
Priority	Critical
Procedure	1. Join a LiveKit room 2. Verify camera is initially on (video preview visible) 3. Click the camera toggle button 4. Observe the change 5. Click the camera toggle button again
Expected Result	• First click: Video preview disappears, button shows "off" state • Second click: Video preview reappears, button shows "on" state • Other participants see the video state change
Date Tested	
Result	

Table 5.13: TC-AV-002: Microphone Toggle

Test ID	TC-AV-002
Category	Functional, Component
Objective	Verify that user can toggle microphone on/off
Priority	Critical
Procedure	1. Join a LiveKit room 2. Verify microphone is initially on 3. Click the microphone toggle button 4. Speak and observe audio indicator 5. Click the microphone toggle button again
Expected Result	• First click: Mic muted, button shows muted state • When muted: Audio is not transmitted to other participants • Second click: Mic unmuted, audio transmitted again
Date Tested	
Result	

Table 5.14: TC-AV-003: Screen Share Start

Test ID	TC-AV-003
Category	Functional, Integration
Objective	Verify that user can start screen sharing
Priority	Critical
Procedure	1. Join a LiveKit room 2. Click the “Share Screen” button 3. Select a screen/window from the browser picker 4. Click “Share”
Expected Result	• Screen share appears in the video grid • Share button shows active state • Other participants see the shared screen • Bot begins processing shared content
Date Tested	
Result	

Table 5.15: TC-AV-004: Screen Share Stop

Test ID	TC-AV-004
Category	Functional
Objective	Verify that user can stop screen sharing
Priority	Major
Procedure	1. Join a LiveKit room and start screen sharing 2. Click the “Stop Share” button
Expected Result	• Screen share is removed from the video grid • Button returns to inactive state
Date Tested	
Result	

Table 5.16: TC-AV-005: Multiple Participants Video

Test ID	TC-AV-005
Category	Integration
Objective	Verify video display with two participants
Priority	Major
Procedure	1. Create a room and join as User A with camera on 2. Have User B join the same room with camera on 3. Observe the video grid on both clients
Expected Result	• Each participant sees the other participant’s video • Participant names are displayed under videos
Date Tested	
Result	

5.1.4 Transcription Module

Table 5.17: TC-TRANS-001: Real-time Transcription Display

Test ID	TC-TRANS-001
Category	Functional, Integration
Objective	Verify that speech is transcribed in real-time
Priority	Critical
Procedure	1. Join a LiveKit room 2. Ensure microphone is on 3. Speak clearly into the microphone 4. Observe the transcription panel on the left
Expected Result	• Transcript appears within 3 seconds of speech • Text is reasonably accurate ($>$ 90% words correct) • Speaker name is associated with transcript • Interim results show as they're processed • Final results are clearly marked
Date Tested	
Result	

Table 5.18: TC-TRANS-002: Multi-Speaker Transcription

Test ID	TC-TRANS-002
Category	Integration
Objective	Verify transcription works with multiple speakers
Priority	Major
Procedure	1. Have two users join the same room 2. User A speaks a sentence, then User B responds 3. Check the transcription panel
Expected Result	• Both speakers' text appears in the panel • Speaker names are correctly attributed
Date Tested	
Result	

Table 5.19: TC-TRANS-003: Transcription Panel Scroll

Test ID	TC-TRANS-003
Category	Usability
Objective	Verify transcription panel is scrollable during long sessions
Priority	Minor
Procedure	1. Join a room and speak continuously for 1+ minutes 2. Observe the transcription panel as it fills up 3. Scroll up in the panel to view earlier text

Expected Result	• Panel auto-scrolls to show new transcripts • Previous transcripts remain accessible by scrolling up
Date Tested	
Result	

5.1.5 Slide Processing Module

Table 5.20: TC-SLIDE-001: Slide Detection

Test ID	TC-SLIDE-001
Category	Functional, Integration
Objective	Verify that shared presentation slides are detected and processed
Priority	Critical
Procedure	1. Join a room and open a presentation 2. Start screen sharing the presentation 3. Advance through 3 slides, pausing 5 seconds on each 4. Ask a question about the content of one of the slides
Expected Result	• Q&A returns a relevant answer about the slide content • This confirms slides were detected, processed, and embedded
Date Tested	
Result	

Table 5.21: TC-SLIDE-002: Slide Change Detection

Test ID	TC-SLIDE-002
Category	Functional
Objective	Verify that advancing slides triggers new processing
Priority	Major
Procedure	1. Join a room and share a presentation 2. Advance to the next slide and wait 5 seconds 3. Ask a question about the new slide content
Expected Result	• The Q&A answer references content from the latest slide • Previous slide content is also still available
Date Tested	
Result	

5.1.6 Question Answering Module

Table 5.22: TC-QA-001: Ask Question About Slide

Test ID	TC-QA-001

Category	Functional, Integration
Objective	Verify user can ask questions about slide content
Priority	Critical
Procedure	1. Join a Zoom meeting with PARSE sidebar 2. Share a presentation with text content 3. Wait for slides to be processed 4. In the sidebar, type a question about the slide content 5. Click "Ask" or press Enter
Expected Result	• Question is sent to GPU server • Answer appears within 5 seconds • Answer is relevant to slide content • Source references are provided
Date Tested	
Result	

Table 5.23: TC-QA-002: Follow-up Question

Test ID	TC-QA-002
Category	Functional, Integration
Objective	Verify context is maintained for follow-up questions
Priority	Major
Procedure	1. Complete TC-QA-001 2. Ask a follow-up question referencing "it" or "that" 3. For example: "Can you explain that in more detail?"
Expected Result	• System understands context from previous Q&A • Answer relates to the previous topic • Session ID is maintained
Date Tested	
Result	

Table 5.24: TC-QA-003: Question Without Context

Test ID	TC-QA-003
Category	Functional
Objective	Verify system handles questions when no slides processed
Priority	Minor
Procedure	1. Join a new meeting without sharing any slides 2. Ask a question about slide content 3. Observe the response
Expected Result	• System returns appropriate message • "No slides have been processed yet" or similar • No error is thrown

Date Tested	
Result	

5.1.7 Zoom Integration Module

Table 5.25: TC-ZOOM-001: Zoom OAuth Connection

Test ID	TC-ZOOM-001
Category	Functional, Integration
Objective	Verify user can connect their Zoom account
Priority	Critical
Procedure	1. Log in to PARSE 2. Navigate to settings or Zoom integration page 3. Click "Connect Zoom Account" 4. Log in to Zoom when redirected 5. Authorize the PARSE application
Expected Result	• User is redirected back to PARSE • Zoom connection status shows "Connected" • Access token is stored securely • User can create Zoom rooms
Date Tested	
Result	

Table 5.26: TC-ZOOM-002: Zoom RTMS Activation

Test ID	TC-ZOOM-002
Category	Functional, Integration
Objective	Verify RTMS streaming works for Zoom meetings
Priority	Critical
Procedure	1. Create a Zoom room through PARSE 2. Join the meeting via Zoom client and open the PARSE sidebar 3. Start sharing a presentation 4. Speak into the microphone and observe the sidebar
Expected Result	• Transcripts appear in the sidebar • Slide content is available for Q&A queries
Date Tested	
Result	

Table 5.27: TC-ZOOM-003: Zoom Transcript Display

Test ID	TC-ZOOM-003
Category	Functional, Integration

Objective	Verify Zoom transcripts appear in sidebar
Priority	Major
Procedure	1. Join a Zoom meeting with RTMS active 2. Open the PARSE sidebar 3. Speak into the microphone 4. Observe the transcript panel in the sidebar
Expected Result	• Transcripts appear via WebSocket • Speaker identification is shown • Updates occur in real-time
Date Tested	
Result	

Table 5.28: TC-ZOOM-004: Zoom OAuth Token Refresh

Test ID	TC-ZOOM-004
Category	Functional, Security
Objective	Verify that Zoom OAuth tokens are refreshed before expiry
Priority	Major
Procedure	1. Log in to PARSE and connect a Zoom account via OAuth 2. Create a Zoom room and join the meeting 3. Keep the meeting active or check token status after 50+ minutes 4. Attempt to create another Zoom room
Expected Result	• Initial OAuth connection shows “Connected” status • Token is refreshed automatically before expiry • Creating a new Zoom room succeeds without re-authentication
Date Tested	
Result	

5.1.8 Bot Integration Module

Table 5.29: TC-BOT-001: Bot Auto-Join on Room Creation

Test ID	TC-BOT-001
Category	Functional, Integration
Objective	Verify that the bot automatically joins a LiveKit room when a new room is created
Priority	Critical

Procedure	1. Log in and navigate to the dashboard 2. Create a new LiveKit room 3. Join the room as a user 4. Wait up to 5 seconds for the bot to connect 5. Speak a short sentence into the microphone 6. Observe the transcription panel
Expected Result	• Bot connects to the room automatically • Bot does NOT appear in the participant list • Transcription text appears in the panel after speaking
Date Tested	
Result	

Table 5.30: TC-BOT-002: Zoom WebSocket Transcript Delivery

Test ID	TC-BOT-002
Category	Functional, Integration
Objective	Verify that real-time transcripts are delivered to the Zoom sidebar app via WebSocket
Priority	Critical
Procedure	1. Create a Zoom room and join the meeting via Zoom client 2. Open the PARSE sidebar app and click “Start Session” 3. Wait for the status to show “Connected” 4. Speak clearly into the microphone 5. Observe the transcript panel in the sidebar
Expected Result	• Transcripts appear in the sidebar within 3 seconds of speech • Speaker name is displayed with each transcript entry • New transcripts are appended below previous ones • No duplicate transcripts are displayed
Date Tested	
Result	

5.1.9 GPU Server Module

Table 5.31: TC-GPU-001: Meeting Knowledge Base Reset

Test ID	TC-GPU-001
Category	Functional, Integration, Security
Objective	Verify that the GPU server clears all meeting data when a reset is requested
Priority	Major

Procedure	1. Join a meeting and share a presentation with at least 3 slides 2. Wait for slides to be processed and ask a question to confirm Q&A works 3. End the meeting session via the room UI (click "End Room" button) 4. Start a new meeting and ask the same question
Expected Result	• Before reset: Q&A returns relevant answers about the slides • After reset: Meeting data is cleared from the database • New meeting returns "No slides have been processed yet" or similar
Date Tested	
Result	

5.1.10 LiveKit Module

Table 5.32: TC-LK-001: Bot Hidden Participant Verification

Test ID	TC-LK-001
Category	Functional, Integration, Usability
Objective	Verify that the bot does not appear in the participant list
Priority	Major
Procedure	1. Create a new LiveKit room and join as a user 2. Wait for the bot to auto-join (up to 5 seconds) 3. Observe the participant list in the room UI 4. Have a second user join the same room 5. Speak and verify transcripts appear in the panel 6. Check that the participant count shows only human users
Expected Result	• Participant list shows only human users (bot is not listed) • Participant count is accurate (does not include bot) • Transcription functions correctly, confirming the bot is active • Bot can send transcription data but does not publish video or audio
Date Tested	
Result	

5.2 Non-Functional Test Cases

5.2.1 Performance Tests

Table 5.33: TC-PERF-001: Transcription Latency

Test ID	TC-PERF-001
Category	Performance
Objective	Verify transcription appears within acceptable delay
Priority	Critical
Procedure	1. Join a room with microphone enabled 2. Speak a clear sentence and note the time 3. Observe when the transcript appears in the panel
Expected Result	• Transcript appears within 3 seconds of speaking • Text is reasonably accurate
Date Tested	
Result	

Table 5.34: TC-PERF-002: Q&A Response Time

Test ID	TC-PERF-002
Category	Performance
Objective	Verify Q&A responses are generated within time limit
Priority	Major
Procedure	1. Process 5 slides through the system 2. Ask a question and start a timer 3. Stop timer when answer appears 4. Repeat with 5 different questions 5. Calculate average response time
Expected Result	• Average response time < 5 seconds • Maximum response time < 10 seconds • All questions receive valid responses
Date Tested	
Result	

Table 5.35: TC-PERF-003: Room Join Time

Test ID	TC-PERF-003
Category	Performance
Objective	Verify users can join rooms quickly
Priority	Major
Procedure	1. Create a room 2. Click “Join” and start timer 3. Stop timer when video preview appears 4. Repeat 5 times with new rooms

Expected Result	• Average join time < 3 seconds • Connection established reliably • No timeout errors
Date Tested	
Result	

Table 5.36: TC-PERF-004: Slide Processing Time

Test ID	TC-PERF-004
Category	Performance
Objective	Verify slides are processed in a reasonable time
Priority	Major
Procedure	1. Share a presentation with 5 slides 2. Advance through all slides (5 seconds each) 3. After the last slide, ask a question about it
Expected Result	• Q&A returns a relevant answer within 10 seconds of the last slide • All slides are available for querying
Date Tested	
Result	

Table 5.37: TC-PERF-005: Dashboard Load with Multiple Rooms

Test ID	TC-PERF-005
Category	Performance
Objective	Verify dashboard loads quickly when many rooms exist
Priority	Major
Procedure	1. Log in to the application 2. Create 10 rooms from the dashboard 3. Navigate away from the dashboard 4. Navigate back to the dashboard and observe load time
Expected Result	• Dashboard loads within 2 seconds • All 10 rooms are displayed correctly • No lag or freezing during room list rendering
Date Tested	
Result	

5.2.2 Security Tests

Table 5.38: TC-SEC-001: Unauthorized Room Access

Test ID	TC-SEC-001

Category	Security
Objective	Verify users cannot access rooms without authentication
Priority	Critical
Procedure	1. Open browser in incognito/private mode 2. Try to directly access /room/[room_id] URL 3. Try to call /api/rooms endpoint without session 4. Try to call backend API directly without proxy headers
Expected Result	• UI redirects to login page • API returns 401 Unauthorized • Direct backend calls are rejected • No room data is exposed
Date Tested	
Result	

Table 5.39: TC-SEC-002: Session Expiry

Test ID	TC-SEC-002
Category	Security
Objective	Verify that logging out invalidates the session
Priority	Major
Procedure	1. Log in and copy the current page URL 2. Click “Logout” 3. Paste the copied URL into the browser address bar and press Enter
Expected Result	• User is redirected to the login page • Dashboard content is not accessible after logout
Date Tested	
Result	

Table 5.40: TC-SEC-003: Direct Backend Access Prevention

Test ID	TC-SEC-003
Category	Security
Objective	Verify the backend cannot be accessed by bypassing the frontend
Priority	Critical
Procedure	1. Open a new browser tab 2. Enter the backend API URL directly (e.g., http://localhost:8001/api/rooms) 3. Observe the response
Expected Result	• The request is rejected with an error (401 or 403) • No room data or user data is returned
Date Tested	

Result	

Table 5.41: TC-SEC-004: Meeting Data Isolation

Test ID	TC-SEC-004
Category	Security
Objective	Verify meeting data is isolated between meetings
Priority	Critical
Procedure	1. Create Meeting A, process slides, ask questions 2. Create Meeting B with different content 3. In Meeting B, ask questions about Meeting A content 4. Check if any data leaks between meetings
Expected Result	• Meeting B cannot access Meeting A slides • Q&A only returns Meeting B context • Database queries filter by meeting_id
Date Tested	
Result	

Table 5.42: TC-SEC-005: XSS Prevention

Test ID	TC-SEC-005
Category	Security
Objective	Verify application is protected against XSS attacks
Priority	Critical
Procedure	1. Try to create room with name containing script tag 2. Try to inject JavaScript in Q&A input 3. Check if any user input is rendered unescaped
Expected Result	• Script tags are escaped/sanitized • No JavaScript execution from user input • React's built-in escaping prevents XSS
Date Tested	
Result	

5.2.3 Reliability and Stability Tests

Table 5.43: TC-REL-001: Frontend Error Display on Backend Unavailability

Test ID	TC-REL-001
Category	Reliability
Objective	Verify the frontend shows a clear error when the backend is unreachable
Priority	Major

Procedure	1. Log in to the application 2. Stop the backend service 3. Attempt to create a new room from the dashboard 4. Observe the error message displayed
Expected Result	• A user-friendly error message is shown (e.g., "Service unavailable") • The application does not crash or show a blank page • No technical stack trace is exposed to the user
Date Tested	
Result	

Table 5.44: TC-REL-002: API Error Response Format

Test ID	TC-REL-002
Category	Reliability
Objective	Verify backend API returns proper error responses for invalid requests
Priority	Major
Procedure	1. Log in to the application 2. Try to join a room that does not exist by entering a fake room ID in the URL 3. Try to create a room with an empty name 4. Observe the responses
Expected Result	• Non-existent room returns a clear "Room not found" message • Invalid input returns a validation error message • User is not stuck on a broken page
Date Tested	
Result	

Table 5.45: TC-REL-003: Meeting Reconnection After Network Interruption

Test ID	TC-REL-003
Category	Stability
Objective	Verify that a user can recover from a brief network drop during a meeting
Priority	Major
Procedure	1. Join a LiveKit room and confirm connection is established 2. Disconnect from Wi-Fi or network for 5 seconds 3. Reconnect to the network 4. Observe the room UI

Expected Result	• UI shows a "Reconnecting" or connection lost indicator • After network returns, connection re-establishes automatically • Video and audio resume without leaving and rejoining the room
Date Tested	
Result	

Table 5.46: TC-REL-004: GPU Server Health Check

Test ID	TC-REL-004
Category	Reliability
Objective	Verify GPU server status endpoint reports system health correctly
Priority	Major
Procedure	1. Ensure the GPU server is running 2. Access the GPU server status endpoint (/status) 3. Process one slide through the system 4. Access the status endpoint again
Expected Result	• Status endpoint returns a successful response with system information • After processing, the slide count is incremented • No error is returned from the status endpoint
Date Tested	
Result	

5.2.4 Compatibility Tests

Table 5.47: TC-COMPAT-001: Browser Compatibility

Test ID	TC-COMPAT-001
Category	Compatibility
Objective	Verify application works on Chrome and Firefox
Priority	Major
Procedure	1. Open the application in Google Chrome and complete the full user journey (register, create room, join, share screen, view transcript) 2. Repeat the same steps in Mozilla Firefox
Expected Result	• All features work correctly in both browsers • UI renders consistently across both
Date Tested	
Result	

Table 5.48: TC-COMPAT-002: Mobile Responsiveness

Test ID	TC-COMPAT-002
Category	Compatibility
Objective	Verify dashboard is viewable on mobile screens
Priority	Minor
Procedure	1. Open the dashboard on a mobile device or use browser dev tools to simulate a mobile screen 2. Check that the room list and navigation are usable
Expected Result	• Dashboard layout adjusts to the smaller screen • Room list and buttons are accessible without horizontal scrolling
Date Tested	
Result	

5.2.5 Usability Tests

Table 5.49: TC-USAB-001: First-Time User Onboarding

Test ID	TC-USAB-001
Category	Usability
Objective	Verify a new user can register and join a room without guidance
Priority	Major
Procedure	1. Have a person with no prior knowledge of the system register an account 2. Ask them to create and join a room 3. Observe whether they can complete the tasks without assistance
Expected Result	• User completes all tasks without external help • Error messages (if any) are clear and helpful
Date Tested	
Result	

Table 5.50: TC-USAB-002: Error Message Clarity

Test ID	TC-USAB-002
Category	Usability
Objective	Verify error messages are user-friendly
Priority	Minor
Procedure	1. Attempt to log in with incorrect credentials 2. Attempt to join a non-existent room 3. Read the error messages shown

Expected Result	• Error messages explain what went wrong in plain language • No raw technical errors or stack traces are shown
Date Tested	
Result	

Chapter 6

Consideration of Various Factors in Engineering Design

This chapter discusses how various factors affected the engineering design process of PARSE, including public health, safety, security, welfare, and global, cultural, social, environmental, and economic factors.

6.1 Constraints

6.1.1 Technical Constraints

1. **GPU Availability**: System requires NVIDIA GPU with CUDA support for ML inference
2. **Network Bandwidth**: Real-time video processing requires stable, high-bandwidth connection
3. **API Rate Limits**: Deepgram and Zoom APIs have usage limits
4. **Browser Compatibility**: WebRTC support varies across browsers

6.1.2 Business Constraints

1. **Development Timeline**: Academic semester constraints
2. **Budget**: Limited to open-source tools and free API tiers where possible
3. **Team Size**: 5-member team with distributed expertise

6.2 Standards

The following standards and best practices were followed:

- **OAuth 2.0**: For Zoom authentication integration
- **JWT (RFC 7519)**: For LiveKit token generation
- **WebRTC**: W3C standard for real-time communication
- **REST API Design**: Following OpenAPI specifications
- **WCAG 2.1**: Accessibility guidelines for UI
- **GDPR**: Data protection considerations for EU users

6.3 Factor Analysis

6.3.1 Public Health

Effect Level: 7/10

PARSE contributes positively to public health by:

- Enabling remote participation, reducing need for physical meetings
- Providing accessibility features (transcription) for hearing-impaired users
- Supporting educational sessions that can include health-related content

Design Decisions:

- Implemented real-time transcription for accessibility
- Designed for remote-first usage patterns
- Ensured low latency to prevent eye strain from delayed video

6.3.2 Safety

Effect Level: 5/10

Safety considerations in PARSE design:

- No physical safety risks (software-only system)
- Cognitive safety through clear UI/UX design
- Data safety through encryption and access controls

Design Decisions:

- Clear visual indicators for recording/sharing status
- Confirmation dialogs for destructive actions
- Session timeout to prevent unauthorized access

6.3.3 Security

Effect Level: 9/10

Security is a critical concern for PARSE:

- Handles sensitive meeting content (video, audio, slides)
- Manages user authentication and sessions
- Processes potentially confidential business presentations

Design Decisions:

- Multi-layer authentication (Better Auth + proxy secret)

- JWT tokens with limited lifetime
- Data isolation between meetings
- HTTPS/WSS for all external communication
- No persistent storage of video/audio by default

6.3.4 Welfare

Effect Level: 6/10
PARSE supports user welfare through:

- Reducing cognitive load with AI-assisted Q&A
- Enabling asynchronous access to meeting content
- Supporting diverse learning styles through multiple modalities

Design Decisions:

- Clean, uncluttered interface
- Optional transcription panel (can be hidden)
- Configurable notification settings

6.3.5 Global Factors

Effect Level: 6/10
Global considerations:

- Users may be distributed across time zones
- Varying internet infrastructure quality globally
- Different data protection regulations by country

Design Decisions:

- Timestamps use UTC with local conversion
- Adaptive bitrate for varying network conditions
- Self-hosted option for data sovereignty

6.3.6 Cultural Factors

Effect Level: 4/10
Cultural considerations:

- UI text in English (primary market)
- Meeting norms vary by culture
- Visual design preferences differ

Design Decisions:

- Neutral, professional color scheme
- Extensible architecture for future localization
- No culturally-specific imagery or icons

6.3.7 Social Factors

Effect Level: 7/10
Social impact of PARSE:

- Enables inclusive participation (accessibility)
- Supports education and knowledge sharing
- May affect presenter-audience dynamics

Design Decisions:

- Transcription visible to all participants equally
- Q&A feature democratizes asking questions
- Speaker identification maintains accountability

6.3.8 Environmental Factors

Effect Level: 5/10
Environmental considerations:

- Server energy consumption for GPU inference
- Reduced travel through remote meetings (positive)
- Electronic device usage

Design Decisions:

- Efficient model inference (batching where possible)
- On-demand GPU usage (not always-on)
- Optimized data transfer to reduce bandwidth

6.3.9 Economic Factors

Effect Level: 8/10
Economic considerations:

- Cost of GPU infrastructure
- API costs (Deepgram, Zoom)
- Potential cost savings for users vs. competitors

Design Decisions:

- Open-source architecture reduces licensing costs
- Efficient transcription (3-second chunks) balances cost/accuracy
- Self-hosted option eliminates recurring SaaS fees

6.4 Factor Summary Table

Table 6.1: Engineering Design Factor Impact Summary

Factor	**Effect (0-10)**	**Primary Impact**
Public Health	7	Accessibility, remote work enablement
Safety	5	Data protection, clear UI indicators
Security	9	Multi-layer auth, data isolation
Welfare	6	Reduced cognitive load, accessibility
Global	6	Time zones, varying infrastructure
Cultural	4	Neutral design, localization-ready
Social	7	Inclusive participation, knowledge sharing
Environmental	5	GPU efficiency, reduced travel
Economic	8	Open-source, efficient API usage

Chapter 7

Teamwork Details

This chapter describes how the team collaborated throughout the project, including individual contributions, collaborative environment creation, and leadership sharing.

7.1 Contributing and Functioning Effectively on the Team

7.1.1 Anıl Kılıç – DevOps and Infrastructure Lead

Primary Contributions:

- Designed Docker Compose deployment architecture
- Set up LiveKit server configuration
- Implemented bot service for media processing
- Configured ngrok tunnels for Zoom integration
- Created CI/CD pipelines and deployment scripts

Technical Skills Applied:

- Docker and container orchestration
- LiveKit server administration
- Network configuration and tunneling
- Shell scripting and automation

Effective Functioning:

- Maintained reproducible deployment environments
- Created comprehensive setup documentation
- Supported team with local development issues

7.1.2 Aybars Buğra Aksoy – Backend Development Lead

Primary Contributions:

- Designed and implemented the FastAPI backend
- Created the room management and authentication services
- Integrated LiveKit token generation
- Implemented Zoom OAuth flow and SDK integration
- Set up the database schema for Better Auth

Technical Skills Applied:

- FastAPI and async Python
- PostgreSQL and asyncpg
- OAuth 2.0 implementation
- JWT token handling

Effective Functioning:

- Provided clear API documentation for frontend
- Implemented comprehensive error handling
- Maintained backward compatibility during iterations

7.1.3 Barış Yaycı – ML/AI Integration Lead

Primary Contributions:

- Designed and implemented the GPU server
- Integrated YOLO, CLIP, and Qwen2.5-VL models
- Built the slide processing pipeline
- Implemented the Q&A engine with RAG
- Set up pgvector for semantic search

Technical Skills Applied:

- PyTorch and Transformers
- Computer vision (YOLO, CLIP)
- Vision-language models (Qwen2.5-VL)
- Vector databases and RAG

Effective Functioning:

- Optimized models for production GPU constraints
- Documented ML pipeline for reproducibility
- Provided benchmarks for performance testing

7.1.4 Bora Yetkin – Frontend Development Lead

Primary Contributions:

- Designed and implemented the Next.js frontend architecture
- Built the dashboard, room management, and authentication pages
- Integrated LiveKit React components for video/audio
- Implemented the transcription panel with real-time updates
- Created responsive UI with TailwindCSS

Technical Skills Applied:

- React 19 and Next.js 16 App Router
- Better Auth integration
- WebSocket and DataChannel handling
- UI/UX design principles

Effective Functioning:

- Maintained clear component documentation
- Responded to integration issues within 24 hours
- Participated in all code reviews

7.1.5 Eren Berk Eraslan – Zoom and AI/ML Integration Lead

Primary Contributions:

- Designed and implemented the GPU server
- Integrated AI pipeline to GPU Server and Zoom SDK
- Implemented Speech to Text transcription
- Designed and implemented the Zoom sidebar React application
- Built Zoom OAuth integration flow and token management
- Implemented RTMS webhook handling and media stream routing
- Developed the end-to-end testing strategy and test case specifications
- Created integration test procedures for cross-service validation

Technical Skills Applied:

- React 18 and Zoom Meeting SDK
- OAuth 2.0 and Zoom API integration

- WebSocket communication and real-time data streaming
- Manual integration testing methodology

Effective Functioning:

- Coordinated Zoom-specific testing across bot and frontend teams
- Maintained Zoom app compliance with marketplace requirements
- Provided test coverage reports and identified critical gaps

7.2 Helping Create a Collaborative and Inclusive Environment

7.2.1 Communication Practices

The team established effective communication channels:

- **Daily Standups**: 15-minute daily sync via Discord
- **Weekly Deep Dives**: 1-hour technical discussions
- **Shared Documentation**: Confluence wiki for decisions
- **Code Reviews**: All PRs required 1+ approval

7.2.2 Knowledge Sharing

Team members actively shared knowledge:

- Bora Yetkin conducted React hooks workshop for the team
- Aybars Buğra Aksoy documented API patterns for frontend integration
- Barış Yaycı presented ML model selection rationale
- Anıl Kılıç created deployment runbooks
- Eren Berk Eraslan organized Zoom SDK integration sessions and testing workshops

7.2.3 Inclusive Practices

The team ensured all voices were heard:

- Rotating meeting facilitator role
- Anonymous feedback surveys after milestones
- Pair programming sessions across expertise areas
- Asynchronous decision-making for flexibility

7.2.4 Conflict Resolution

When disagreements arose, the team:

- Discussed technical trade-offs objectively
- Used prototyping to validate competing approaches
- Escalated to advisor when consensus couldn't be reached
- Documented decisions and rationale

7.3 Taking Lead Role and Sharing Leadership on the Team

7.3.1 Distributed Leadership Model

Each member led specific project areas:

Table 7.1: Leadership Distribution

Member	**Lead Role**	**Leadership Activities**
Anıl Kılıç	DevOps Lead	Infrastructure decisions, deployment strategy, security configuration
Aybars Buğra Aksoy	Backend Lead	API design, database schema, integration coordination
Barış Yaycı	ML Lead	Model selection, inference optimization, accuracy benchmarking
Bora Yetkin	Frontend Lead	UI/UX decisions, component architecture, user experience review
Eren Berk Eraslan	Zoom & Testing Lead	Zoom integration, RTMS setup, test strategy, quality assurance

7.3.2 Leadership Rotation

For cross-cutting concerns, leadership rotated:

- **Sprint Planning**: Rotated every 2-week sprint
- **Demo Presentations**: Each member presented their component
- **Documentation Review**: Weekly rotation
- **Integration Testing**: Collaborative, Aybars Buğra Aksoy coordinated

7.3.3 Decision-Making Authority

Leaders had authority within their domain:

- Technology choices within component (e.g., Frontend Lead chose TailwindCSS)
- Implementation details without team vote
- Code review approval for their area

Cross-domain decisions required consensus:

- API contract changes
- Data model modifications
- Security policy changes

7.3.4 Leadership Development

Each member grew their leadership skills:

- Bora Yetkin: Improved stakeholder communication
- Aybars Buğra Aksoy: Enhanced technical documentation
- Barış Yaycı: Developed cross-functional collaboration
- Anıl Kılıç: Strengthened project coordination
- Eren Berk Eraslan: Built quality assurance leadership

Chapter 8

Glossary

API (Application Programming Interface)
A set of protocols and tools for building software applications and enabling communication between different software components.

Better Auth
An open-source authentication library for JavaScript/TypeScript applications, used in PARSE for user session management.

Bot In PARSE context, an automated service that joins meetings to process audio/video streams.

CLIP (Contrastive Language-Image Pre-training)
OpenAI's neural network model that learns visual concepts from natural language supervision, used for generating image embeddings.

CUDA (Compute Unified Device Architecture)
NVIDIA's parallel computing platform and programming model for GPU computing.

DataChannel
WebRTC feature allowing peer-to-peer data transfer, used in PARSE for transmitting transcriptions.

Deepgram
A speech recognition platform providing STT (Speech-to-Text) services, specifically the Nova 3 model used in PARSE.

Docker
A platform for developing, shipping, and running applications in containers.

Embedding
A numerical representation of data (text or images) in a high-dimensional vector space.

FastAPI
A modern, fast Python web framework for building APIs.

GPU (Graphics Processing Unit)
A specialized processor designed for parallel processing, used for ML model inference.

JWT (JSON Web Token)
A compact, URL-safe means of representing claims to be transferred between two parties.

LiveKit
An open-source, self-hosted WebRTC platform for building real-time video and audio applications.

Next.js
A React framework for building full-stack web applications.

OAuth 2.0
An authorization framework enabling third-party applications to obtain limited access to a web service.

pgvector
PostgreSQL extension for storing and querying vector embeddings.

Qwen2.5-VL
A vision-language model capable of understanding both images and text, used for Q&A generation.

RAG (Retrieval-Augmented Generation)
A technique combining information retrieval with text generation for improved AI responses.

RTMS (Real-Time Media Streams)
Zoom's API for accessing raw audio and video streams from meetings.

STT (Speech-to-Text)
The process of converting spoken language into written text.

Transcription
The written form of spoken content, generated in real-time during meetings.

VLM (Vision Language Model)
An AI model that can process and understand both visual and textual information.

WebRTC (Web Real-Time Communication)
A technology enabling peer-to-peer audio, video, and data sharing in web browsers.

WebSocket
A protocol providing full-duplex communication channels over a single TCP connection.

YOLO (You Only Look Once)
A real-time object detection algorithm, specifically DocLayout YOLO for document analysis.

Chapter 9

References

1. Bruegge, B., & Dutoit, A. H. (2004). *Object-Oriented Software Engineering: Using UML, Patterns, and Java* (2nd ed.). Prentice Hall.
2. Next.js Documentation. (2024). Retrieved from https://nextjs.org/docs
3. FastAPI Documentation. (2024). Retrieved from https://fastapi.tiangolo.com/
4. LiveKit Documentation. (2024). Retrieved from https://docs.livekit.io/
5. Zoom Developer Documentation. (2024). RTMS API. Retrieved from https://developers.zoom.us/docs/api/rtms/
6. Deepgram Documentation. (2024). Nova 3 Model. Retrieved from https://developers.deepgram.com/
7. pgvector GitHub Repository. (2024). Retrieved from https://github.com/pgvector/pgvector
8. Ultralytics YOLO Documentation. (2024). Retrieved from https://docs.ultralytics.com/
9. OpenCLIP GitHub Repository. (2024). Retrieved from https://github.com/mlfoundations/open_clip
10. Qwen2.5-VL Model Card. (2024). Hugging Face. Retrieved from https://huggingface.co/Qwen/Qwen2.5-VL
11. Better Auth Documentation. (2024). Retrieved from https://www.better-auth.com/docs
12. PostgreSQL Documentation. (2024). Retrieved from https://www.postgresql.org/docs/
13. Docker Documentation. (2024). Retrieved from https://docs.docker.com/
14. OAuth 2.0 Authorization Framework. RFC 6749. Retrieved from https://tools.ietf.org/html/rfc6749
15. JSON Web Token (JWT). RFC 7519. Retrieved from https://tools.ietf.org/html/rfc7519
16. WebRTC 1.0: Real-Time Communication Between Browsers. W3C. Retrieved from https://www.w3.org/TR/webrtc/

17. OWASP Top Ten. (2021). Retrieved from https://owasp.org/www-project-top-ten/

18. Software Testing Help. (2024). Manual Testing Tutorial. Retrieved from https://www.softwaretestinghelp.com/manual-testing-tutorial-1/