🔧 Middleware – Final Technical Requirements for Renesis

🎯 Central Function

- The middleware acts as the **central "nervous system"** of the entire platform.
- It controls and orchestrates:
  - **AI communication** (Coach Klaus)
  - **Timing and system states**
  - **Feedback cycles**
  - **Dynamic coach role behavior**Discovery_Detailed_Proj...

💡 Core Technical Requirements

- **Ultra-low latency and high performance** in communication with the AI (real-time Coach Klaus interaction)
- **Robust, traceable layer system** for decision logging and explainability
- **High interface compatibility** with third-party tools:
  - Make.com (workflow automation)
  - LMS editor (content control)
  - Pipedrive CRM (client & lead tracking)Discovery_Detailed_Proj...

🧠 Middleware-Orchestrated Components

- **Session Memory** (MongoDB-based)
- **Prompt Engine** (AI control and response logic)
- **Adaptive Decision Engine (ADE)**
- **Confidence Score**
- **Process Intelligent Engine (PIE)**
- **AI interaction audit logging** (fully AI Act compliant)Discovery_Detailed_Proj...

📡 Interfaces & Integration Layer

- Central orchestration of **REST-compliant APIs**
- All AI-related processes with **D-ID, ElevenLabs, DeepL** are **mediated via the middleware**
- All personal data **remains within Hetzner (Germany)** – no persistent storage at AWS
- Planned **GDPR-compliant audit layer** for Aleph Alpha API callsDiscovery_Detailed_Proj...

🛡 Security, Monitoring & Transparency

- **AI action traceability** ("Reason Trace") for every AI decision (trigger, change, justification, timestamp)
- **Full compliance with the EU AI Act** – logging, explainability, auditability
- Role-based access control to ensure separation between content, AI logic, and user data
- **Monitoring dashboards** (admin & trainer) for:
    - Interaction quality
    - Confidence score evolution
    - Adaptive decision historiesDiscovery_Detailed_Proj...

🔄 Scalability & Communication Infrastructure

- Middleware handles **asynchronous response pipelines** for Coach Klaus to enable smooth UX
- Designed for **horizontal scaling** (Docker / Kubernetes-ready if needed)
- Clear **communication architecture**:
    - **AWS-based AI services are decoupled**
    - Middleware mediates **only temporary dialog-specific content**
    - No personal data is transferred outside the Hetzner environment

🧠 Session Memory – Technical and Functional Requirements

🎯 Function & Purpose

- The Session Memory serves as the **memory of the platform**, storing contextual user interaction data across modules and over time.
- It enables:
    - Adaptive responses by Coach Klaus
    - Personalized learning paths
    - Referencing past experiences and decisions

🗂️ Data Contents & Structure

- Stores:
    - Progress per module and match day
    - Confidence Scores (per module)
    - Tactics check results
    - User responses to reflection and diagnostics
    - Coach interaction history and role changes
    - Process insights from the PIE module

🔄 Technical Specifications

- **Technology**: MongoDB (NoSQL), fully hosted on Hetzner (Germany)
- **Structure**: JSON-based entries with timestamp
- **Data separation**: strict distinction between personal data and technical identifiers
- **Access control**: only accessible via middleware, trainer dashboards, and admin – *not visible to users or editorial team*
- **Multi-tenancy**: Separate memory per law firm, optional client-level context

---

🧩 System Integration

- Essential for:
    - **Adaptive Decision Engine** (e.g. coach role switching, learning path adjustments)
    - **Confidence Score** evolution
    - Reminder and follow-up logic
    - Personalized suggestions from Coach Klaus
- Coach Klaus uses session memory to:
    - Reference earlier conversations and answers
    - Recognize strengths, weaknesses, and emotional phases
    - Recall results from the tactics check

🛡 Data Protection & Compliance

- Stored **exclusively on Hetzner servers** in Germany
- No external sync or export
- Compliant with:
    - **GDPR** (incl. pseudonymization of personal data)
    - **EU AI Act** (traceability and auditability)
- Not visible to:
    - External tools
    - Editorial team
    - CRM or LMS editors

🧠 Adaptive Decision Engine (ADE) – Functional & Technical Requirements

🎯 Core Function

- The ADE is the **emotionally intelligent core** of the platform.
- It controls:
    - each user's **individual learning journey**
    - the **dynamic role behavior** of Coach Klaus
    - reminders, difficulty levels, repetition logic, bonus unlocksDiscovery_Detailed_Proj...

🧩 Input & Output Structure

Typical Inputs:

- user_id, law_firm_id, module_id
- Confidence score, last result, error type
- Session memory context, tactics check results
- Progress state, inactivity duration

Typical Outputs:

- coach_role (e.g., motivating, reflective, challenging)
- learning_path_modifier (e.g., activate repetition, unlock bonus)
- reminder_trigger (e.g., time-based prompts)
- reason_trace (transparent justification stored in audit log)Discovery_Detailed_Proj...

🧠 Rule Logic & Control

Rule-based decision engine with editorial access:

- Configurable rule matrix (e.g., via JSON or GUI-based rule editor)
- Examples:
    - CS < 0.6 → Coach = compassionate
    - Repeated errors → Coach explains actively
    - Inactivity → Reminder is triggered

Trigger points include:

- Completion of a module
- Start of a new match day
- Evaluation of the tactics check
- Inactivity (e.g., after 72 hours)
- Voice input signals like: “I don’t understand that”Discovery_Detailed_Proj...

🔍 Logging & Explainability ("Reason Trace")

- Every decision is **fully logged and explainable** in JSON format.
- Example entry:

json

```
{
 "event": "role_switch",
 "from": "motivating",
 "to": "reflective",
 "trigger": "confidence_score_drop",
 "reasoning": "CS of 0.8 → 0.5 after module 3, 2 incorrect answers, long response time",
 "timestamp": "2025-06-20T12:34:56"}
```

- Purpose:
  - AI Act compliance & auditability
  - User-facing transparency
  - Support for quality control & A/B testingDiscovery_Detailed_Proj...

🧬 Future Strategy – Hybrid Model

- Initially 100% rule-based – easy to maintain and audit.
- Future: Option to enhance decision logic with machine learning:
  - e.g., decision trees or reinforcement learning
  - Always with a focus on explainability and editorial controlDiscovery_Detailed_Proj...

🧠 Prompt Engine – Functional & Technical Requirements

🎯 Purpose & System Role

- The Prompt Engine is the **central control unit for all AI responses** on the platform.
- It enables:
  - Full control over content, tone, role behavior, and language quality of Coach Klaus
  - Ongoing development and refinement of prompts by the editorial and tech teams
  - Adaptive prompt generation based on user context and learning progressionDiscovery_Detailed_Proj...

🛠 Technical Requirements

- Fully **custom-built** – no external prompt service providers
- Must support **rule-based and version-controlled prompt orchestration**
- Integrates tightly with:
  - **Session Memory**
  - **Adaptive Decision Engine (ADE)**
  - **Process Intelligent Engine (PIE)**
  - **Confidence Score**
- Supports:
  - Initial prompts (e.g., onboarding, reflection, coaching)
  - Dynamic prompts (e.g., in response to hesitation, errors, success)
- Must allow for **real-time performance** (especially for voice-based outputs)

📊 Monitoring & Analytics

- Full **logging of all prompt–response interactions**
- Dedicated **dashboard** for editorial and technical review:
  - Response quality
  - Latency and reaction time

    - Role tracking (e.g., coach role switching over time)
    - Suggestions for prompt adjustments
- Purpose: quality control, optimization, and transparent auditingDiscovery_Detailed_Proj...

🔒 Security & Auditability

- All prompts and responses must be:
    - **Documented, versioned, and traceable** (EU AI Act compliance)
    - Logged with full context (user ID, module, timestamp, interaction history)
- **Edit permissions limited** to specific roles (e.g., editorial, admin)
- **No external usage** of prompt data for third-party analysis

---

🧬 **Advanced Functions & Outlook**:

- **A/B testing** of different prompt variations
- Dynamic prompt adaptation via learning analytics
- AI-assisted prompt refinement (e.g., emotional tone adjustment based on confidence signals or user mood)

🎯 Confidence Score – Functional Overview & Requirements

🧠 Purpose & Philosophy

- The Confidence Score measures a user's **subjective sense of certainty** – not their factual accuracy.
- Its primary role is to detect emotional states such as **uncertainty, overload, or hesitation**, and **respond empathetically**.
- It is **not used for blocking or grading**, but to dynamically **adjust the behavior of Coach Klaus** in a supportive wayDiscovery_Detailed_Proj...

🔍 Calculation Logic (Middleware-Controlled)

- The score is calculated automatically in the middleware based on **weighted behavioral signals**:

Signal Type	Examples	Weight
Error Behavior	Wrong answers, logical errors	high
Demand Behavior	Frequency & type of clarification questions	medium
Usage Signals	Interruptions, repeated actions, pauses	medium
Voice Patterns	Phrases like "I'm not sure..."	high
Speed & Reaction	Hesitation, rushed clicking	low

- Output: **Score between 0% (very uncertain) and 100% (very confident)**
- Continuously updated – e.g. after modules, key interactions, or coach promptsDiscovery_Detailed_Proj...

⚠️ **Important:** All scoring parameters (weights, thresholds, signal definitions) are still under development.
Therefore, the middleware must support **maximum flexibility and configurability** – including real-time adjustments by the editorial team.

🧬 Storage & History

- Score is stored **per user and per module** in the Session Memory
- Visible to **trainer and admin dashboards**, showing trends and history
- **Not visible to users directly** – but expressed through Coach Klaus' behavior

---

🤖 System-Level Impact

Low Score:

- Coach Klaus becomes more empathetic, explanatory, repetitive
- Reminders may be triggered
- Content complexity is lowered when needed

High Score:

- Coach Klaus becomes more dynamic, challenging, encouraging
- Bonus content or reflection challenges may be unlocked
- Examples: “You seem confident – want to apply your knowledge?”

➡️ Coach Klaus always stays **supportive and encouraging** – never punitive or limitingDiscovery_Detailed_Proj...

📊 Visualization & Dashboards

- Admin and trainer dashboards show:
  - **Progression curves** per user and firm
  - **Heatmaps** for team sentiment and development
  - **Alerts** for sustained low scores → triggers for learning support

---

✳️ Unique Didactic Feature

- The Confidence Score enables a **new level of emotionally intelligent AI interaction**:
  - Learners feel seen, supported, and understood
  - Learning resistance can be identified without verbal input
  - Coach Klaus becomes a **true companion**, not just a feedback engine

🧠 Process Intelligent Engine (PIE) – Functional & Technical Requirements

🎯 Purpose & Concept

- The PIE is the platform's **real-world process analysis and application engine**.
- It bridges the gap between learned content and real law firm operations – enabling **practical transfer and reflection**.
- It is not a modeling tool – but an **AI-supported, empathic analysis companion** for daily workflows.

🧩 Input Types

- **Free-form input** via text or speech
  → e.g. spontaneous description of a current task flow
- **File uploads** (e.g., PDF, DOCX, scans)
  → e.g. internal instructions, protocols, checklists
- Optional: OCR preprocessing for scanned documents

🧠 Processing via ErxlebenAI (LLaMA 3.3)

- The semantic analysis is performed entirely by **ErxlebenAI (based on LLaMA 3.3)**
- Prompts are structured via the platform's **custom Prompt Engine**
- All processing is **GDPR-compliant** and hosted in a secure Hetzner environment
- The AI identifies:
  - Process steps, role assignments, media transitions
  - Bottlenecks, redundancies, semantic gaps
  - Matches and mismatches with learned content

🔄 Two-Phase Engine Structure

1. **Analysis Module**
   → Extracts, structures, and standardizes process logic
   → Outputs a structured internal JSON representation
2. **Evaluation Module**
   → Compares with module content and best practices

→ Provides a qualitative assessment and concrete, didactically linked suggestions

Example Output:

json

KopierenBearbeiten

```
{
 "issues": ["media discontinuity", "lack of automation"],
 "recommendations": [
  "Use a digital invoice workflow (see module 4)",
  "Clarify handover roles and responsibilities"
 ],
 "evaluation": "optimizable",
 "confidence": 92
}
```

🤝 Integration with Coach Klaus

- Coach Klaus is **directly involved** in the process analysis:
    - Actively asks for process descriptions ("Tell me how you currently handle it...")
    - Explains the evaluation in natural language
    - Links to appropriate modules for further learning
    - Asks follow-up questions if key details are missing
    - Motivates and encourages improvement

📊 Visibility & Dashboards

- Results appear in the **trainer dashboard**
- Multiple processes per firm can be tracked over time
- In future: **internal benchmarking within the same law firm** (no firm-to-firm comparison)

🚫 Distinction from Classic Tools & Documentation Requirement

- PIE is **not** a diagramming or BPMN tool:
    - No technical knowledge or modeling is needed
    - Natural language suffices
    - No specialized software or formats required

✅ **Nonetheless, a documentable output is essential:**
All analyzed processes must be presented in an **exportable and visual form** (e.g. flowchart or structured JSON), suitable for:

- Internal communication
- Coaching sessions
- Team handovers
- The goal is not standardization, but a **practical, easy-to-understand visual reflection** of real-world workflows – flexible, shareable, and aligned with the coaching narrative.

🎓 Learning Management System (LMS)

🧱 System Architecture & Core Setup

- Fully **custom-built LMS** – no third-party platform
- Fully integrated with:
    - Middleware
    - Session Memory
    - Prompt Engine
    - Dashboards
    - ErxlebenAI (based on LLaMA 3.3)
- Hosted in a **GDPR-compliant** Hetzner environment (Germany)
- Scope: **360 match days** = 12 modules × 3 leagues × 10 match days

---

📚 Learning Formats & Match Day Structure

Fixed structure per match day: 5+1 phases

1. Attunement
2. Knowledge impulse
3. Processing
4. Application
5. Conclusion & motivation
   +1: Resilience impulse (optional)

Key learning formats:

- 🎥 Videos (coach, story, explainer)
- ❓ Quizzes with error classification (input for Confidence Score)
- 📝 Reflection tasks (free text, multiple choice, scaling)
- 🧠 Mini simulations (e.g., branching decision trees)
- 💬 Interactive dialogues with Coach Klaus (text, speech, avatar)
- 🧩 PIE elements (process reflection via upload or free input)

- 🛠 **Process Designer**: Interactive visual builder (e.g., swimlanes, drag-and-drop), exportable
- 📌 Digital pinboards for team collaboration and idea sharing
- 🎧 “Break Tea” audio: motivational quotes, stadium vibe
- 📈 League feedback: progress tracking, points, badges, league advancement

---

🧭 Tactics Check – Technical Requirements for Renesis

1. 📦 Booking & Scheduling

- **Booking via LMS-integrated store**
    - “Tactics Check” offered as a bookable product
    - Triggers status change and redirects user to welcome page
- **Scheduling via calendar tool (e.g. Cal.com)**
    - Full integration into LMS booking flow
    - Data passed to system: user_id, law_firm_id, appointment, booking_id
    - GDPR-compliant setup required

---

2. 🧾 Preparation Phase (Digital)

- **Pre-check questionnaire (structured form)**
    - Includes scale-based items, checkboxes, free text
    - Data stored per user & law firm in the **Session Memory**
    - Displayed in **trainer dashboard** (with timestamp & status)
- **Coach Klaus welcome video**
    - Auto-triggered upon completion of the questionnaire
    - Logic-controlled visibility (based on completion)

---

3. 📦 Physical Package Trigger (Optional)

- **System flag activated 7 days prior to the check**

    - Marks user for manual/automated mailing
    - Exportable via CSV/API for logistics provider
    - *Note: LMS does not perform shipping – only triggers export*

---

4. 🧠 On-Site Execution (Trainer-led)

- **Trainer dashboard (tablet/laptop optimized)**
    - 5–10 scale questions per module + free text
    - Real-time input stored to structured JSON:
      user_id, modul_id, score, text, timestamp
- **Coach Klaus intro video (optional)**
    - Launchable on-site from trainer device
    - Opens the check with motivation and guidance

---

5. 🎯 Dartboard Animation (Visual Evaluation)

- **Pre-built animation (e.g. SVG or Lottie)**
    - A dart flies into the board **based on score (1–6)**
    - Dynamic position controlled via middleware/frontend
    - Animation is emotional and sports-themed
- **Optional:** Coach Klaus comment (voice or text) after impact

---

6. 📊 Evaluation & League Assignment

- **Automatic league assignment per module**
    - Output: liga = startelf / taktgeber / spielmacher
    - Controlled via middleware (or Adaptive Decision Engine)
    - Impacts: learning path, Coach behavior, content difficulty

---

7. 🔗 Data Flow & System Integration

- **All check data flows into:**
    - **Session Memory** (contextual baseline)

    - **Adaptive Decision Engine** (initial config)
    - **Prompt Engine** (for Coach Klaus dialogue references)
    - **Trainer Dashboard** (visual feedback, export, alerts)

🧠 Coach Klaus – Technical Specification & Control Logic

🎯 Role in the System

- Coach Klaus is the **central AI interface** for user interaction throughout the learning journey.
- He acts in **multiple adaptive roles** (e.g., motivating, reflective, explanatory, challenging).
- Responsibilities include:
    - Opening and closing each match day
    - Reflecting on user progress
    - Providing emotional encouragement and clarification
    - Delivering dynamic feedback and reminders

🛠 Core Components & Architecture

1. Multimodal Speech Interface:

- 🎤 **Speech-to-Text** (user input): via ElevenLabs Whisper (or equivalent)
- 🗣 **Text-to-Speech** (output): via ElevenLabs API
- 👤 **Avatar rendering**: D-ID HQ Full Body Avatar
- All media orchestrated in real time through the **middleware and prompt engine**

2. Prompt Engine Integration:

- Dynamic prompt generation based on:
    - user_id, module_id, match_day, confidence_score, coach_role, and session history
- Prompts control:
    - Content, tone, complexity, media format, and coach behavior

3. Adaptive Role Control via ADE:

- The **Adaptive Decision Engine** determines:

    - Role switching
    - Coach tonality
    - Depth of explanations
    - Response frequency
- Based on real-time context and Confidence Score

4. Session Memory Access:

- Klaus has persistent access to:
    - Previous questions and answers
    - Learning performance and interaction patterns
    - Tactics check results
    - Emotional cues and break history
- Accessed live through the middleware, no local caching

🔁 Interaction Logic

- **Proactive Triggers:**
    - Start or end of a match day
    - User inactivity or re-entry after absence
    - Milestones like league promotion or PIE success
- **Reactive Prompts:**
    - User questions via text or voice
    - Coach adjusts behavior and language based on current confidence level
- **Fallback Logic:**
    - Detects uncertainty or confusion
    - Offers repeat explanations or simplifications
    - Adapts tone and media accordingly

🎬 Output & Media Handling

- **AI-generated real-time responses:**

    - Text → TTS → Avatar video (live-rendered)
    - Must render and respond **in less than 2 seconds total latency**
- **A small number of predefined video responses:**
    - Used for **emotionally significant standard moments** (e.g., start of journey, league promotion, end of module)
    - Stored as fixed assets and referenced via prompt ID
    - Selected by the prompt engine for latency optimization and emotional resonance

---

📊 Logging & System Control

- All interactions are fully logged
- Data is available in:
    - Prompt logs (for optimization and audit)
    - ADE trace log (for role switching decisions)
    - Trainer dashboard (for visibility and analysis)

⚙️ System Requirements

- ⏱️ **End-to-end latency must be < 2 seconds** (including prompt generation, TTS, avatar rendering)
- 🔁 Continuous session context access (via middleware)
- 🔐 Full logging & traceability (AI Act compliance)
- 🧩 Manual trigger option for trainers (e.g., play predefined message)

Trainer & Admin Dashboards – Technical Requirements

Purpose & Function

The dashboards serve as the central **monitoring, insight, and control interface** for:

- **Trainers** (learning progress support and coaching)
- **Admins** (platform and user management)
- **AI Monitoring Role** (prompt logging, behavior trace, auditability)

Role-Based Access Logic

Role	Access Scope	Permissions
Trainer	Law firm-specific learning data	Read-only, provide recommendations, no content/system modification
Editorial Team	Content formats & maintenance	No access to user data or logs
Admin (Management)	Full platform access	User, system, content, prompt & audit log control
AI Monitoring	Prompt Engine & model logs	Full AI trace visibility for response analysis & latency tracking

---

Live Data Sources

All dashboard data is dynamically retrieved via the middleware from:

- **Session Memory** (context, score history)
- **LMS** (module progress, match day status)
- **ADE & Confidence Score Engine** (decisions, role logic)
- **Prompt Engine Logs** (for Coach Klaus tracking)
- **PIE results** (process evaluations & benchmarks)
- **Tactics Check results** (initial league assignment, scores)

Trainer Dashboard – Functional Scope

- Firm- and user-specific overviews

- Drill-down by module, match day, and league
- Visualizations include:
  - 🎯 Tactics Check dartboards
  - 📊 Module & match day completion
  - 📈 Confidence Score timelines & heatmaps
  - 🧠 Coach Klaus usage (role, duration, trigger reason)
  - 🔁 Inactivity tracking (last login, drop-off point)
  - 🧩 PIE usage & process analysis results
- **Exports**:
  - CSV / PDF

🛠 Admin Dashboard – Additional Capabilities

- System-wide insights:
  - Module and media usage rates
  - Active users per league
  - Booking statuses
  - Drop-off rates & match day conversion
  - Prompt engine stats and rendering performance (Coach Klaus)
  - Badge and point distributions
- Control functions:
  - User and role assignments
  - Visibility overrides (e.g., test modes)
  - Manual content unlocking
  - Module activation (e.g., add-ons, events)
- Audit & compliance features:
  - Prompt & ADE trace export (AI Act compliant)
  - Latency logging (TTS + Avatar)
  - Error tracking & alert logic

🛒 Store & Checkout System – Technical Requirements

🎯 Purpose

- Enables **digital booking and management** of:
    - Modules & course add-ons (e.g., PIE, certifications)
    - Events (e.g., Tactics Check, live sessions)
    - Upgrades (e.g., league transitions, bonus content)
- Fully integrated into the **LMS and middleware**
- Checkout must be **GDPR-compliant, secure, and automated**

🧱 Architecture & Components

1. Store Frontend (within LMS):

- Displays available products dynamically based on user role and status
- Filterable by:
    - Module
    - League level
    - Availability (already booked = hidden)
- Supports product visibility logic (e.g., show PIE only to "Playmaker League")

2. Checkout Integration:

- Supported providers:
    - 💳 Stripe (primary)
    - 🅿️ PayPal (optional)
- Uses redirect or embedded mode
- Confirmation via **Webhook to middleware**

3. Access Activation:

- Upon successful payment:
    - Product is unlocked in the LMS
    - Role and visibility rules are updated
    - Middleware receives all transaction metadata

📦 Product Structure & Metadata

Each store item includes:

- Unique product_id
- Product type (module / service / download / certificate)
- Visibility logic (public / league-based / personalized)
- Price, booking status, and unlock behavior
- Optional: triggers for content release (e.g., auto-start of match day 1)

🔁 Middleware Logic

- After payment, the checkout system calls the middleware via API
- Payload includes:
    - User ID
    - Product ID
    - Payment details
    - Time of transaction
- The middleware:
    - Activates content inside the LMS
    - Updates Session Memory and Adaptive Decision Engine (if applicable)
    - Syncs status with the Trainer & Admin Dashboards

🔐 Data Privacy & Compliance

- **No payment data is stored** on the platform
- Only transaction metadata is retained (e.g., Stripe token, webhook ID)
- Fully GDPR-compliant:
    - Uses EU-based payment providers
    - Logs: product booked, timestamp, user ID, receipt reference
    - Invoices handled via middleware or external tool (PDF via email)

➕ Additional Features

- **Promo codes / vouchers** via Stripe API (optional)
- **PDF invoicing** via external service or custom middleware logic
- **Certification unlocking** based on:
  - Module completion
  - Valid payment
  - Admin verification (if needed)

🌍 Multilingual Intelligence – Coach Klaus

(Powered exclusively by ElevenLabs & DeepL)

🎯 Purpose

- Coach Klaus must **automatically detect the user's spoken language** and respond **fluently in that same language**, without any manual language selection.
- This applies to **all supported ElevenLabs languages**, not just German, English, and French.
- The system uses:
  - **ElevenLabs** for both speech recognition (STT) and voice output (TTS)
  - **DeepL** for translation of system prompts where needed

🧠 Technical Components

1. Language Detection & Transcription (ElevenLabs STT):

- User speaks in any supported language
- ElevenLabs detects:
  - The **spoken language**
  - The **transcribed text**
- Example:

json

KopierenBearbeiten

```
{
  "language": "it",
  "text": "Non capisco questo modulo"
}
```

2. Prompt Generation & Language Control:

- Middleware receives both text and detected language
- The Prompt Engine:
  - Constructs a context-aware prompt in the **user's language**
  - If no localized version exists, the German base prompt is translated on-the-fly via **DeepL API**
- The localized prompt is passed to **ErxlebenAI (LLaMA 3.3)**
- The model responds **directly in the detected language** (no reverse translation)

3. Voice Output via ElevenLabs TTS:

- The AI response is converted to speech using **the correct voice profile per language**
- Each supported language is mapped to a suitable Coach Klaus voice
- Output reflects role, tone, and emotional context

4. Avatar Rendering via D-ID:

- TTS audio is synced with D-ID's full-body avatar engine
- The avatar lip-syncs naturally in the user's language – no video pre-rendering required
- Result: Coach Klaus speaks **directly and emotionally in the user's native language**

🔁 Middleware-Controlled Interaction Flow

1. User speaks in, e.g., Italian
2. ElevenLabs STT detects "it" and returns transcribed text
3. Middleware passes content + language to the Prompt Engine
4. If needed, DeepL translates the prompt into Italian

5. ErxlebenAI generates the reply in Italian
6. ElevenLabs converts it to Italian speech
7. D-ID renders the avatar lip-synced to the Italian audio
8. User receives a seamless native-language reply

📊 Technical Requirements

- **ElevenLabs STT & TTS**:
  - Must support all desired languages (DE, EN, FR, IT, ES, PT, PL, NL, etc.)
  - Coach Klaus voice mappings per language (e.g., coach_voice_nl)
- **DeepL Pro API**:
  - For dynamic translation of base prompts if language variant not available
  - Handles all supported DeepL languages automatically
- **D-ID Full Body Avatar API**:
  - Accepts multilingual audio inputs
  - Renders realistic face & body sync in any spoken language
- **Middleware** must:
  - Manage detection, translation, prompt routing, and media coordination
  - Track language preference per session (but input language always overrides)
  - Maintain latency **below 2 seconds total** (STT → AI → TTS → avatar)

✅ Benefits

- Users can **speak freely in their native language**
- Coach Klaus detects and responds **instantly and fluently**
- No settings, toggles, or delays
- Fully scalable to new languages as ElevenLabs expands voice support