Coburg University of Applied Sciences

Faculty of Electrical Engineering and Computer Science

Degree Program: COMPUTER SCIENCE

Internship Report

El Ouarghi El Maizi, Salma

Company	Nara Institute of Science and Technology Social Computing Laboratory 630-0192 Nara, Ikoma-shi, Takayama-cho 8916-5 Nara, Japan
Intership Period	01.10.2025–15.02.2026

Submission date: 28.02.2026

Submission clearance of the internship report to Coburg University:

Advisor	Prof. Eiji Aramaki	
Role	Lab Head	**15.02.2026**
Phone	+81743725250	
E-Mail	aramaki@is.naist.jp	Prof. Eiji Aramaki

Contents

List of Figures

List of Tables

Listings

List of Symbols

Symbol	Meaning	Phys. Unit
α	Krippendorff's Alpha - Coefficient for assessing data reliability	[-]
κ	Cohen's Kappa - Statistical measure for inter-rater reliability	[-]
$BLEU$	Bilingual Evaluation Understudy - Metric for evaluating translation quality	[-]
F_1	F1-Score - Harmonic mean of precision and recall	[-]
G	Guiraud's Index - Measure of lexical diversity	[-]
n	Sample size	[-]
P	Precision - Measure of extraction accuracy	[-]
R	Recall - Measure of extraction completeness	[-]
t_{ASR}	Duration of speech recognition (Whisper)	[s]
t_{LLM}	Duration of extraction and classification (LLM)	[s]
t_{total}	Total processing time of the pipeline	[s]
TTR	Type-Token Ratio - Measure of lexical variability	[-]
v	Processing speed of the audio pipeline	[audio s / real s]
WER	Word Error Rate - Measure of speech recognition accuracy	[%]

Acronyms

AI	Artificial Intelligence
ALS	Advanced Life Support
API	Application Programming Interface
ASGI	Asynchronous Server Gateway Interface
ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
BLS	Basic Life Support
CAD	Computer-Aided Dispatch
CJK	Chinese, Japanese, Korean
CORS	Cross-Origin Resource Sharing
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CUDA	Completion, Understanding, Detail, Accuracy
DNP	Digital Neuropsychological
DOM	Document Object Model
DP	Dynamic Programming
DTO	Data Transfer Object
EMD	Emergency Medical Dispatcher
EMS	Emergency Medical Services
ESI	Emergency Severity Index
FastAPI	Python web framework
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HIS	Hospital Information System
HL7	Health Level 7
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JSON	JavaScript Object Notation

LLM	Large Language Model
ML	Machine Learning
NAIST	Nara Institute of Science and Technology
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OpenAI	OpenAI Inc.
ORM	Object-Relational Mapping
PDF	Portable Document Format
RAM	Random Access Memory
React	JavaScript library for user interfaces
REDSS	Real-Time Emergency Decision Support System
Regex	Regular Expressions
REST	Representational State Transfer
SOAP	Subjective, Objective, Assessment, Plan
SQL	Structured Query Language
SQLite	Structured Query Language Lite
TTR	Type-Token Ratio
UI	User Interface
URL	Uniform Resource Locator
WER	Word Error Rate

1 Introduction

1.1 Motivation, Problem Statement, and Objectives

In emergency dispatch centers, Emergency Medical Dispatchers (EMDs) form the critical interface between help-seekers and pre-clinical emergency care. Under extreme time pressure, precise medical information must be extracted, documented, and prioritized from incomplete or emotionally charged conversations.[KVS18] Dispatchers are exposed to high cognitive load due to limited working memory capacity [Mil56; Zap+25]. Modern technologies in speech recognition and natural language processing offer new potential for workload reduction [Blo+19; Luo+25].

Current manual emergency call processing exhibits structural deficits. First, the sequential processing of conversation and documentation leads to delays, as relevant information is often finalized only after the call [Mel+25; Sco+16] or manuals are consulted sequentially [CDM08, pp. 45–52]. Second, initial medical assessment (triage) is subject to high inter-individual variability, which can lead to inconsistent decisions [Mis+18; Sax+23]. Third, unstructured free-text notes complicate standardized clinical handover and the traceability of collected data [PLG23; Wee68]. Fourth, high stress levels significantly impair dispatchers' working memory and increase the risk of information loss [KVS18; LeB09; Pin17]. Computer-aided assistance systems promise higher diagnostic stability and improved process safety in this context.

Against this background, this thesis investigates the following central research question: *To what extent can an AI-powered real-time system improve the quality of emergency call processing through automated SOAP extraction and standardized triage?*

To answer this question, the following sub-objectives are pursued:

1. **System Development:** Conception and implementation of a real-time pipeline consisting of automatic transcription (Whisper), LLM-based SOAP extraction, and ESI-based triage.
2. **Evaluation:** Quantitative assessment of transcription accuracy and system performance.
3. **Clinical Validation:** Qualitative evaluation of the system by medical professionals ($n = 15$).
4. **Multilingualism:** Implementation of bidirectional translation between English and Japanese.

2 Theoretical Foundations

2.1 Medical Documentation and Triage (SOAP & ESI)

Medical documentation follows the SOAP method (Table 1) [PLG23; Wee68], while triage via the Emergency Severity Index (ESI) prioritizes urgency [Gil+20] (Table 2). Since manual triage has mistriage rates of up to 32 % [Mis+18; Sax+23; Tan+07], automation aims for higher reliability, as computer-aided systems like ANKUTRIAGE already demonstrate [Koc+22].

Tab. 1: Structure of SOAP Documentation

Section	Description and Contents
Subj. (S)	Patient statements, symptoms, pain description (narrative).
Obj. (O)	Measurable facts: Name, age, vital data, consciousness, address.
Assess. (A)	Medical working hypothesis and clinical assessment.
Plan (P)	Therapeutic measures and dispatched rescue resources.

Tab. 2: ESI Triage Levels

ESI	Criteria / Urgency	Res.
1 (CRIT.)	Life-threatening, requires immediate life-saving interventions.	Immediate
2 (HIGH)	Highly urgent; unstable status or high risk.	High
3 (MED.)	Stable, but high resource requirement (diagnostics).	≥ 2
4 (LOW)	Stable, maximum of one medical resource needed.	1
5 (NONE)	Stable, no resource requirement (e.g., counseling).	0

System Adaptation: REDSS consolidates ESI 4 and 5 into „LOW“, as both levels have no immediate resource requirement according to the ESI protocol [Gil+20].

2.2 AI-Powered Information Extraction (Whisper & LLMs)

OpenAI Whisper serves as a robust speech recognition model for transcription in acoustically difficult emergency situations [Luo+25; Rad+23; Zec+25]. The system uses the „base“ model for low latency. GPT-4o handles the extraction of structured information with an accuracy of over 91 % [Gup+25; Per+24] and uses stepwise reasoning (Chain-of-Thought) for better contextualization [Ada+24; Wei+22].

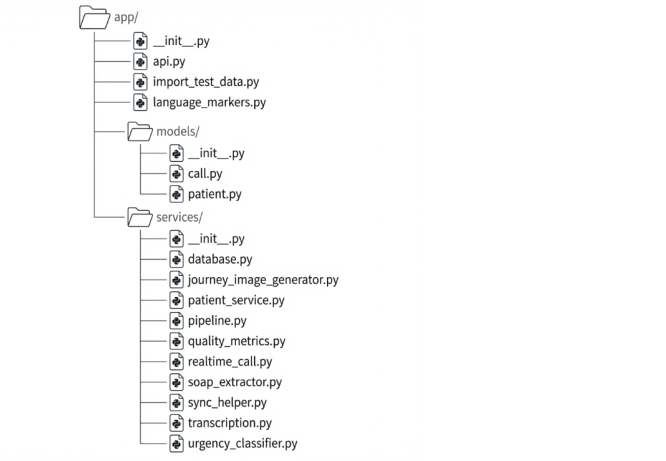
2.3 Human Factors and Cognitive Load

Acute stress significantly reduces working memory capacity [Mil56], complicating parallel tasks such as documentation and assessment [LeB09; Pin17]. This increases error rates and lockout risk [KVS18]. Computer-aided tools relieve personnel by taking over documentation load and enabling a stronger focus on communication [Mel+25; Sco+16].

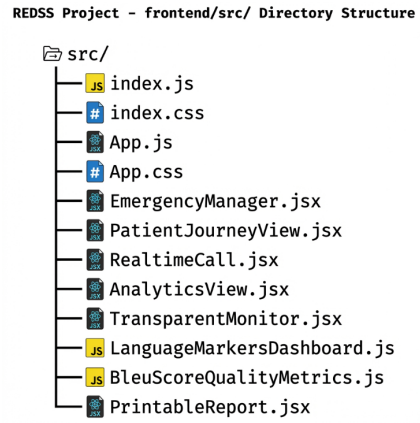
3 System Architecture and Implementation

3.1 System Architecture and Backend Logic

The REDSS system utilizes a **three-tier architecture** [BCK03]. The React frontend communicates via REST and WebSockets with the FastAPI backend, while data is stored in SQLite (Fig. 1). The pipeline coordinates all steps from recording to storage (Fig. 2).



1.a Backend



1.b Frontend

Fig. 1: Architecture Overview

Processing Pipeline: The asynchronous pipeline orchestrates the central processing steps from audio recording to structured data output (Listing 1).

```python
async def process_call(self, audio_path: str) -> dict:
    call_id = f"CALL_{datetime.now().strftime('%Y%m%d_%H%M%S')}"
    try:
        # Step 1: Transcribe audio
        logger.info(f"Starting transcription for {call_id}")
        transcription = await transcription_service.transcribe(audio_path)
        transcript = transcription['text']
        # Step 2: Extract SOAP notes
        logger.info(f"Extracting SOAP notes for {call_id}")
        soap = await soap_extractor.extract(transcript)
        # Step 3: Classify urgency
        urgency = await urgency_classifier.classify(transcript, soap)
        # Step 4: Save to database (atomic)
        with get_db() as db:
            call = save_call(db, call_id, audio_path, transcript)
            update_soap(db, call_id, soap)
            update_urgency(db, call_id, urgency)
            db.commit()
        return self._build_response(call_id, transcript, soap, urgency)
    except Exception as e:
        logger.error(f"Error during processing: {str(e)}")
        raise ProcessingError(f"Pipeline error: {str(e)}") from e
```

Code 1: Processing Pipeline Orchestration

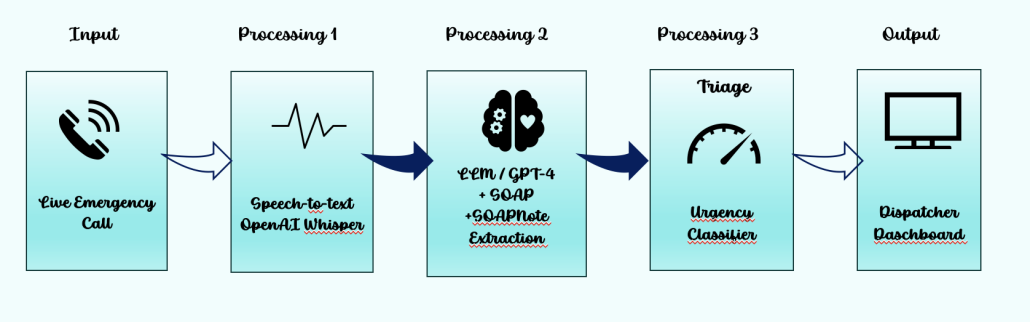


Fig. 2: System Flow

Transcription utilizes the Whisper „base“ model for an optimal trade-off between latency and accuracy [Rad+23; Zec+25]. **SOAP extraction** is performed through instruction optimization (*Prompt Engineering*) using a *Chain-of-Thought* approach [Wei+22]:

```
You are a medical AI assistant. TASK: Extract structured SOAP notes.
GUIDELINES: Identify symptoms, vitals, patient identifiers, and clinical reasoning.
OUTPUT FORMAT (STRICT):
<S>- Subjective observations and patient complaints</S>
<O>- Name: [extracted name]
   - Age: [number]
   - Address: [full address]
   - Symptoms: [list]
</O>
<A>- Clinical assessment and suspected condition</A>
<P>- Emergency response plan and instructions</P>
```

Code 2: SOAP Extraction Prompt (Excerpt)

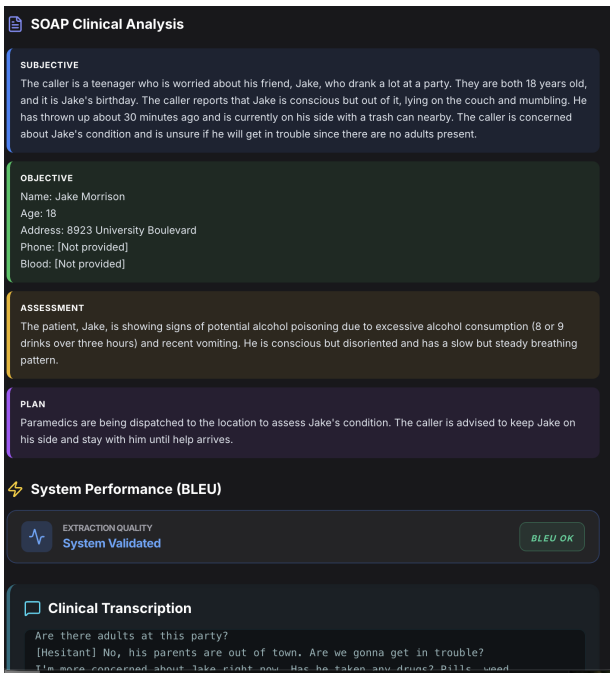


Fig. 3: SOAP Detailed View

The LLM-based extraction approach follows established methods for structured clinical information extraction [Ada+24; Gup+25].

For robust extraction of patient data, additional regex patterns are used for field extraction to secure the structuring by the LLM and catch potential formatting errors:

```
def parseField(text, fieldName):
    # Multilingual label support
    translations = {
        'Name': ['Name', 'Nom', 'Nombre'],
        'Age': ['Age', 'Alter', 'Edad'],
        'Address': ['Address', 'Adresse', 'Location']
    }
    keys = translations.get(fieldName, [fieldName])

    for key in keys:
        patterns = [
            f"{key}\\s*[:]\\s*([^\\n\\r\\.]+)", # Standard
            f"\\*\\s*{key}\\s*[:]\\s*([^\\n]+)", # Bullet
            f"^{key}\\s*[:]\\s*([^\\n]+)" # Start-of-line
        ]
        for pattern in patterns:
            match = re.search(pattern, text, re.IGNORECASE)
            if match and is_valid_value(match.group(1)):
                return match.group(1).strip()
    return 'N/A'
```

Code 3: Intelligent Field Extraction

Urgency classification follows a hybrid approach of keyword matching and contextual AI analysis according to ESI levels [Gil+20]. This strategy combines deterministic ESI requirements with probabilistic AI context analysis:

1. **ESI Level 1/2 (Critical/High-Risk):** Trust ESI (Evidence-based, always escalate).
2. **ESI Level 4/5 (Low/Minimal):** Trust ESI, unless AI detects higher urgency (Safety-first).
3. **ESI Level 3 (Medium):** Use AI refinement (Context-aware).
4. **Safety-first:** Always escalate to higher urgency if either method suggests it.
5. **Special Case:** AI detects test calls or nonsense and downgrades them appropriately.

```
class HybridUrgencyClassifier:
    # ESI Level 1: Immediate Lifesaving Criteria
    CRITERIA = {
        "en": ["intubated", "cardiac arrest", "unconscious", "apneic"],
        "ja": ["挿管", "心停止", "意識不明", "無呼吸"]
    }
    def classify(self, transcript, soap, lang="en"):
        # 1. Rule-based ESI Check (Speed & Safety)
        if any(term in transcript for term in self.CRITERIA.get(lang, [])):
            return {"level": "CRITICAL", "esi": 1}
        # 2. AI Contextual Analysis (Instruction Optimization)
        prompt = f"""
        Role: Emergency MD. Task: ESI Triage.
        Input: {transcript} | Context: {json.dumps(soap)}
        Format: {{ "level": "ESI_LEVEL", "logic": "Rationale in {lang}" }}
        """
        return self.call_ai(prompt)
```

Code 4: Evidence-Based Hybrid Urgency Classifier (ESI v4 + AI)

Client communication is handled via the following REST endpoints:

```
# Audio upload and processing
POST /api/upload
# Text-based input
POST /api/process-text
# Retrieve call list (with translation)
GET /api/calls?lang=en|ja
# Retrieve single call
GET /api/calls/{call_id}?lang=en|ja
# WebSocket for live calls
WS /ws/realtime-call
# Patient management
GET/POST/PUT /api/patients
```

Code 5: Main API Endpoints

3.2 User Interface and Localization Strategy

The user interface is optimized for simultaneous use in English and Japanese. The React-based frontend includes dashboards for incident management, live calls, and treatment history. The central dashboard manages the global state:

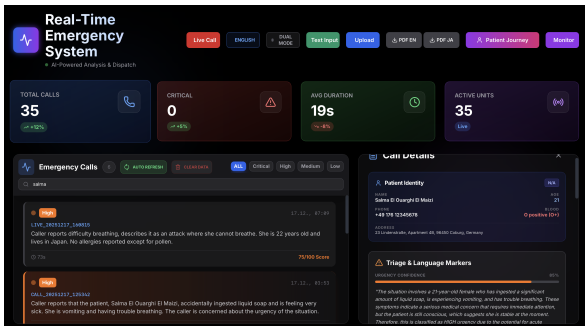
```
const EmergencyManager = () => {
    const [calls, setCalls] = useState([]);
    const [language, setLanguage] = useState('en');
    const [urgencyFilter, setUrgencyFilter] = useState('ALL');
    // Polling for real-time updates
    useEffect(() => {
        const fetchCalls = async () => {
            const response = await fetch(`/api/calls?lang=${language}`);
            const data = await response.json();
            setCalls(data);
        };
        fetchCalls();
        const interval = setInterval(fetchCalls, 5000); // Every 5s
        return () => clearInterval(interval);
    }, [language]);
    return (
        <div className="dashboard">
            <Header language={language} setLanguage={setLanguage} />
            <StatsCards calls={calls} />
            <CallList calls={calls} urgencyFilter={urgencyFilter} />
        </div>
    );
};
```

Code 6: React Dashboard - State Management

The polling interval of 5 seconds represents a compromise between real-time updates and server load, as shorter intervals would unnecessarily burden API rate limits [BG21].

3.2.1 Main Dashboard

The central dashboard for incident management provides an overview of all emergency calls and allows filtering by urgency as well as switching the system language (Fig. 4).



4.a English View 4.b Japanese View

Fig. 4: Main Dashboard at different localization stages

Dynamic Localization Pipeline (Backend): The backend intelligence (`app/api.py`) operates in two stages: A **cache check** first looks for existing translations to minimize latency and costs. Upon a cache miss, **AI translation** (GPT-4o) performs medically precise localization, preserving clinical terminology and cultural specificities (e.g., Japanese address formats).

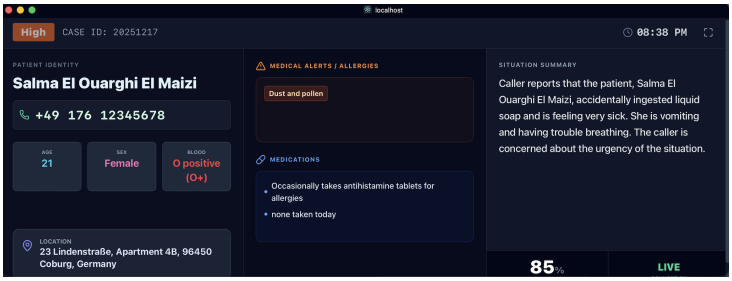
3.2.2 Transparent Monitor

This secondary display enables real-time synchronization [WHA23] and shows personnel key data at a glance without interfering with interaction on the main device. Localization is consolidated into a single request, reducing response time from approx. 20s to under 1s (cache hit) (Fig. 5). The multilingual system (EN/JA) supports bidirectional translation and utilizes caching to avoid redundant API calls:

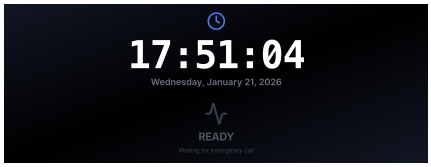
```
CACHE_VERSION = "v2"

def is_valid_cache(cached_data, target_lang):
    # Version check
    if cached_data.get("_version") != CACHE_VERSION:
        return False
    # Language purity check
    if target_lang == "ja":
        # Check for English labels
        english_indicators = ["Name:", "Age:", "not provided"]
        if any(ind in cached_data["soap_objective"]
               for ind in english_indicators):
            return False
    elif target_lang == "en":
        # Check for Japanese characters
        japanese_pattern = r'[\u3040-\u309F\u30A0-\u30FF\u4E00-\u9FFF]'
        if re.search(japanese_pattern, cached_data["soap_objective"]):
            return False
    return True
```

Code 7: Translation Cache Validation



5.a Transparent Dashboard



5.b Time (EN)

5.c Time (JA)

Fig. 5: Transparent Monitor: Dashboard and idle display

```python
def localize_call_data(transcript, soap_notes, metadata, target_lang):
    """
    ONE API request instead of 3+ separate calls
    Translates: Transcript, SOAP (all 4 sections), Metadata
    """
    prompt = f"""
    Translate and localize the following emergency call data to {target_lang}.
    IMPORTANT:
    - Translate field labels (Name, Age)
    - Preserve medical accuracy
    - Maintain SOAP structure
    TRANSCRIPT: {transcript}
    SOAP NOTES: {json.dumps(soap_notes)}
    METADATA: {json.dumps(metadata)}
    OUTPUT: Return JSON with all translated fields
    """
    response = openai.ChatCompletion.create(
        model="gpt-4-mini",
        messages=[{"role": "user", "content": prompt}]
    )
    return parse_localized_response(response)
```

Code 8: Consolidated Localization

3.2.3 Dual Mode Monitoring

A **Dual Mode** button enables simultaneous display of both languages for quality control. This emphasizes the **Human-in-the-Loop** design concept, as the dispatcher can verify the AI translation and intervene at any time. The entire interface is seamlessly localized, as shown in Figure 6.

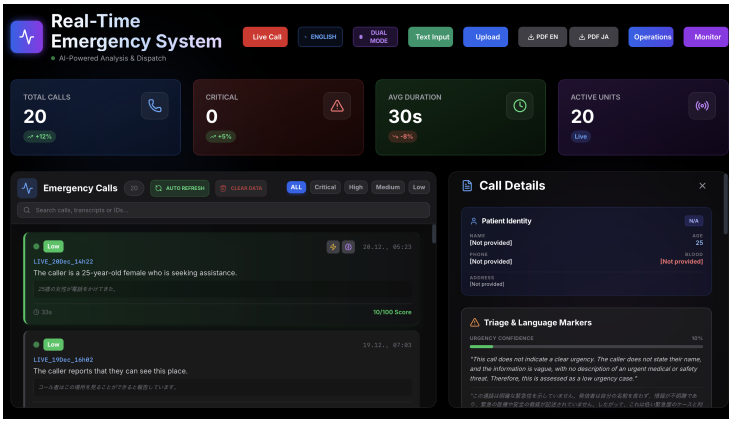


Fig. 6: Simultaneous Display (Dual Mode)

3.2.4 Hybrid System: Multilingual Strategy

The implementation of bilingualism follows a hybrid approach that combines precise control over the user interface with the flexibility of AI-powered content. A strict distinction is made between static (hard-coded) and dynamic (AI-driven) content.

Implementation Matrix: The following table summarizes the distribution of responsibilities within the system:

Tab. 3: Overview of Localization Strategy (EN / JA)

Feature	Type	Component / Location
Buttons & Menus	Hard-Coded	`EmergencyManager.jsx` (TRANSLATIONS)
Table Headers	Hard-Coded	Frontend UI Layer
Emergency Call Transcript	AI-Driven	`soap_extractor.py` (OpenAI)
SOAP Notes	AI-Driven	OpenAI GPT-4o-mini
Urgency Reasoning	AI-Driven	OpenAI GPT-4o-mini
Validation	Logic	`app/api.py` (Mixing Check)

This hybrid architecture ensures that the UI remains professional and stable while complex medical data can be translated dynamically and scalably.

Level 1: UI Translation (Static) To exclude navigation errors and "AI hallucinations" in the menu, all fixed UI texts are controlled via a dictionary system. When language is switched, the entire UI state is immediately updated.

```
// frontend/src/EmergencyManager.jsx
const TRANSLATIONS = {
  en: {
    title: "Real-Time Emergency System",
    liveCall: "Live Call",
    patientIdentity: "Patient Identity",
    urgency: "Urgency Level",
```

```
    critical: "Critical"
  },
  ja: {
    title: "救急意思決定支援シ ステム",
    liveCall: "ライブ通話",
    patientIdentity: "患者基本情報",
    urgency: "緊急度レ ベル",
    critical: "最優先(重症)"
  }
};
```

Code 9: Static Localization in the Frontend

Level 2: AI Instructions and Prompt Engineering (AI-Driven) The core of medical localization is the dynamic generation of prompts. Here, the AI is not only passed the text but is also taught the exact medical structure of the target culture (e.g., Japanese address formats).

```
# app/services/soap_extractor.py
def localize_call_data(self, transcript, soap_notes, metadata, target_language):
    is_japanese = target_language in ["ja", "jp", "japanese"]
    lang_name = "JAPANESE" if is_japanese else "ENGLISH"
    # Define target language specific labels
    if is_japanese:
        labels = "氏名:, 年齢:, 住所:, 電話:, 血液型:"
        missing = "[不明]"
    else:
        labels = "Name:, Age:, Address:, Phone:, Blood:"
        missing = "[Not provided]"
    # Instruction optimization for medical correctness
    prompt = f"""
    You are a medical localization expert.
    Localize the following emergency call data into {lang_name}.
    EXTREMELY IMPORTANT: In the Objective section, you MUST use these exact labels: {labels}.
    If information is missing, use "{missing}".
    Maintain the clinical tone in the Reasoning section.
    """
    return self.call_openai_api(prompt, transcript, soap_notes)
```

Code 10: AI Instructions for Medical Localization

Level 3: Mixing Protection and Quality Assurance To ensure target language purity, automated quality control was implemented. This algorithm detects if the AI has erroneously left labels in the source language ("Japenglish") and forces a re-translation in such cases.

```
# app/api.py
def validate_localization(lang, obj_text):
    # Enhanced detection: Check for language mixing
    if lang == "ja":
        # Japanese should not contain English labels
        has_english_labels = any(label in obj_text for label in ["Name:", "Age:", "Address:"])
        if has_english_labels:
            print(f"Warning: English content in Japanese cache. Re-translating...")
            return False # Forces re-translation
    elif lang == "en":
        # English should not contain Japanese characters
        import re
        has_japanese_chars = bool(re.search(r'[\u3040-\u309F\u30A0-\u30FF\u4E00-\u9FFF]', obj_text))
        if has_japanese_chars:
            print(f"Warning: Japanese content in English cache. Re-translating...")
            return False
    return True
```

Code 11: Validation of Language Purity

3.2.5 Patient Journey Dashboard and Reporting

Auto-Synchronization: The patient journey tracking process enables automatic synchronization of new emergency calls with existing records:

```python
def sync_emergency_call_to_patient_journey(call_id):
    call = db.query(EmergencyCall).filter_by(id=call_id).first()
    patient = db.query(Patient).filter_by(name=call.patient_name).first()
    if not patient:
        patient = Patient(
            name=call.patient_name,
            primary_condition=call.disease,
            medical_history=call.soap_subjective,
            journey_events=json.dumps([{"segment": 1, "date": call.created_at.date(),
                                        "description": f"Initial: {call.disease}", "status": "completed"}])
        )
        db.add(patient)
    else:
        events = json.loads(patient.journey_events or "[]")
        events.append({"segment": len(events) + 1, "date": call.created_at.date(),
                       "description": f"Follow-up: {call.disease}", "status": "completed"})
        patient.journey_events = json.dumps(events)
    db.commit()
```

Code 12: Auto-Sync Mechanism

Tracking the patient progression allows for continuous monitoring from initial emergency call reception through to treatment.

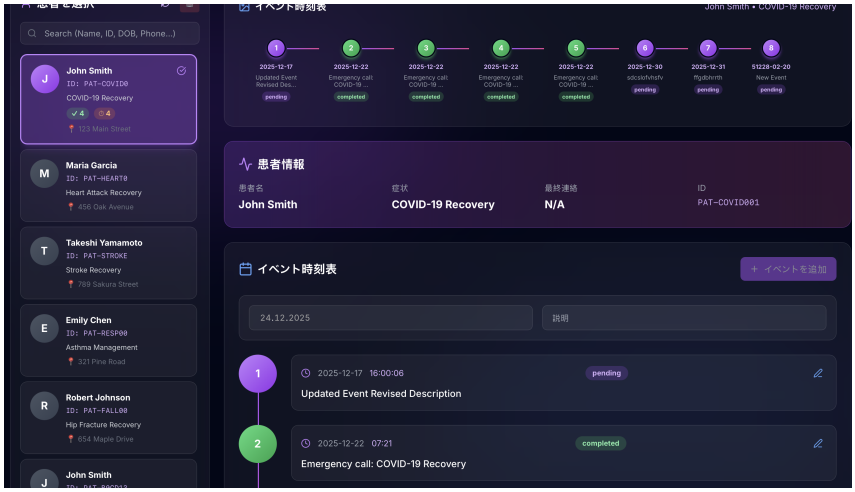


Fig. 7: Patient Journey Dashboard

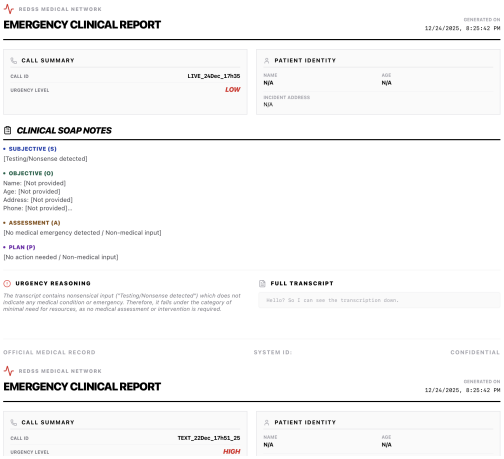
3.2.6 Clinical Report Export (PDF)

The generated reports serve as a bridge between emergency call reception and clinical follow-up treatment. The focus is on compact handover: the export contains exclusively S and O data (Subjective and Objective) to concisely mapping the initial anamnesis.

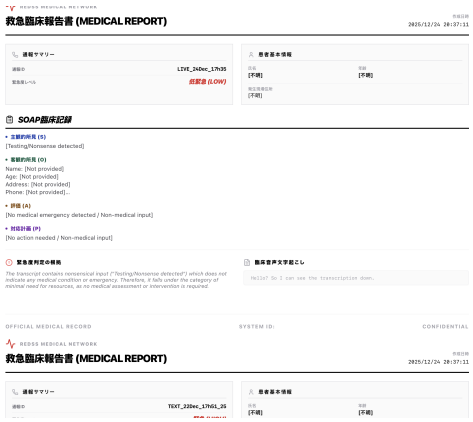
```javascript
const exportToPDF = async () => {
    const { default: jsPDF } = await import('jspdf');
    const doc = new jsPDF('p', 'mm', 'a4');
    // Header & patient metadata
```

```
    doc.setFontSize(22);
    doc.text('Patient Journey Report', 20, 20);
    doc.text(`Name: ${selectedPatient.name}`, 20, 45);
    // Iteration over journey events
    journeyEvents.forEach((event, i) => {
        doc.text(`${event.date}: ${event.description}`, 20, 70 + i*10);
    });
    doc.save(`Patient_Journey_${selectedPatient.name}.pdf`);
};
```

Code 13: Implementation of PDF Export



8.a EN Report



8.b JA Report

Fig. 8: Generated clinical reports via PDF export

3.2.7 Speech Markers for Cognitive Analysis

The `LanguageMarkerAnalyzer` has been refined into a scientifically grounded version that examines transcripts for cognitive impairment using structured thresholds (`MarkerThresholds`). The analysis is modular, covering *Speech Fluency*, cognitive, linguistic, and semantic markers. A core component is the analysis of lexical diversity using Type-Token Ratio (TTR) and Guiraud's Index. TTR is a measure of lexical variability ($TTR = Types/Tokens \in [0,1]$) [CM10], while Guiraud's Index ($G = Types/\sqrt{Tokens}$) is used to enable bias-free analysis independent of text length. The system serves clinical support through passive speech analysis to objectively assess subtle neurological changes (e.g., signs of dementia) in emergencies [Roa+11].

```
def _analyze_linguistic_markers(self, transcript: str, clean_words: List[str], sentences: List[str]) -> Dict:
    """Analysis of linguistic complexity markers"""
    total_words = len(clean_words)
    unique_words = len(set(clean_words))
    # 1. Type-Token Ratio (TTR) - Basic diversity
    ttr = unique_words / total_words if total_words > 0 else 0
    # 2. Guiraud's Index - Length-adjusted lexical diversity
    # This metric is more robust to different text lengths than TTR.
    guiraud = unique_words / math.sqrt(total_words) if total_words > 0 else 0
    return {
        "ttr_score": self._create_marker_result(
            marker_name="TTR (Type-Token Ratio)",
```

```
            value=round(ttr, 3),
            unit="ratio",
            judgment=self._judge_ttr(ttr),
            normal_range="0.50-0.70",
            risk_level=self._assess_risk(ttr, self.thresholds.ttr_low,
                                         self.thresholds.ttr_very_low, reverse=True),
            description="Lexical diversity: Unique words / Total words"
        ),
        "guiraud_index": self._create_marker_result(
            marker_name="Guiraud's Index",
            value=round(guiraud, 2),
            unit="index",
            judgment=self._judge_guiraud(guiraud),
            normal_range="7.0-12.0",
            risk_level=self._assess_risk(guiraud, self.thresholds.guiraud_low,
                                         self.thresholds.guiraud_very_low, reverse=True),
            description="Length-adjusted lexical diversity: Unique words / sqrt(Total words)"
        )
    }
```

Code 14: Analysis of Linguistic Complexity Markers

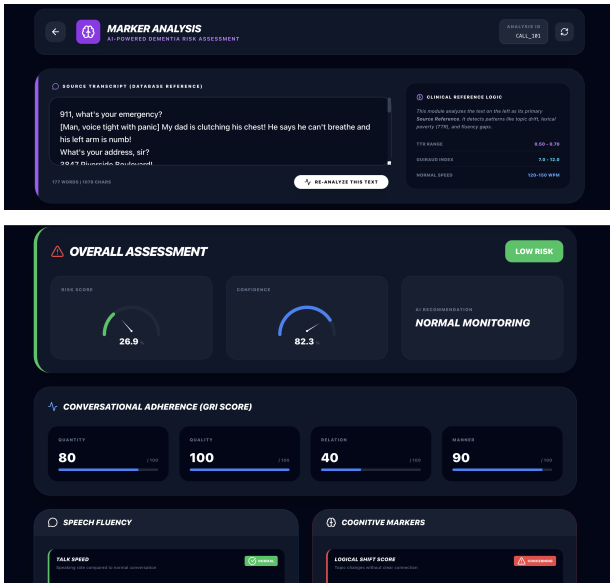


Fig. 9: Linguistic markers and quality analysis

3.2.8 Quality Metrics and Validation

The evaluation of speech recognition systems in medical contexts requires differentiation between mechanical error rates (WER) and semantic correctness (BLEU score and the clinically oriented evaluation system CUDA) [Gup+25; Pap+02]:

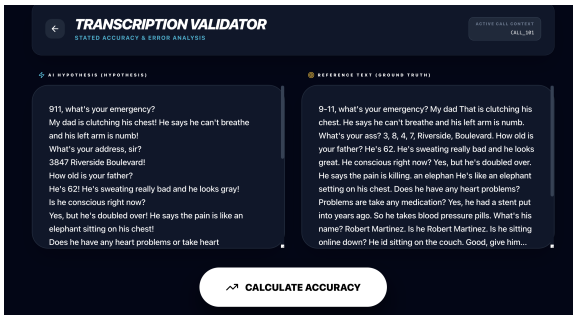
$$WER = \frac{S+D+I}{N} \qquad BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \ln p_n\right) \tag{3.1}$$

In addition, the **CUDA model** is used to assess medical integrity. This clinically oriented scoring system for SOAP notes aggregates four dimensions: **Completion** (completeness of reference

information), **Understanding** (Jaccard similarity of word overlap), **Detail** (ratio of information density to avoid information loss), and **Accuracy** (precision to avoid hallucinations). Together with the BLEU score, CUDA provides a data-driven assessment of how precise and useful the automatically extracted information is for medical personnel.

```python
def clean_text(self, text):
    # 1. Noise reduction (Text normalization)
    text = re.sub(r'\[.*?\]', '', text.lower())
    # 2. Protect Kanji/Kana Unicode ranges
    text = re.sub(r'[^\w\s\u3040-\u309F\u30A0-\u30FF\u4E00-\u9FFF]', ' ', text)
    return text.strip()
def validate_transcription(self, ref, hyp):
    # Mechanical (WER): Edit distance via DP matrix
    wer = calculate_levenshtein(ref, hyp) / len(ref.split())
    # Semantic (CUDA): Clinical data integrity [Gup+25]
    cuda = (completion + understanding + detail + accuracy) / 4
    return {"wer_acc": max(0, (1-wer)*100), "cuda": cuda}
```

Code 15: Validation logic and multilingual normalization



10.a Text comparison (Ref vs. Hyp)

10.b BLEU score and CUDA metrics

Fig. 10: Transcription Accuracy Validator: BLEU and CUDA evaluation

This **development loop** allows the developer to check the AI results in the *Validator* (Fig. 10) using color-coded discrepancies. This dashboard was implemented to test the reliability of the system and verify how accurate or erroneous the transcription actually is.

4 Technical Challenges and Solutions

4.1 Low-Latency Audio Segmentation (Ping-Pong Recording)

A critical challenge in real-time transcription is data loss during chunk generation. Traditional `MediaRecorder` implementations lead to gaps in the audio stream when one segment is finished and the next is started.

The solution was achieved through a **ping-pong buffering process** on the client side. Here, two parallel instances of `MediaRecorder` are controlled with a time offset to guarantee continuous capture:

```
const switchRecorders = () => {
    const nextIndex = (activeIndex + 1) % 2;
    const current = recorders[activeIndex];
    const next = recorders[nextIndex];
    next.start(); // Start new recording before the old one stops
    if (current.state === 'recording') {
        current.stop(); // Completes the chunk and sends it
    }
    activeIndex = nextIndex;
};
```

Code 16: Ping-Pong Buffer for continuous audio capture

4.2 State Synchronization in Multi-Window Environments

The requirement to update a dashboard and an isolated monitor (*Transparent Monitor*) simultaneously without relying on persistent database polling was solved via the **BroadcastChannel API**. This enables event-driven communication between browser contexts with minimal latency.

4.3 Resilience Through Partial Persistence

To handle system failures (e.g., due to API timeouts), a pipeline pattern was integrated that saves **partial results**. Even if the AI analysis fails, the transcript and audio file are persisted to fulfill medical documentation obligations.

4.4 Thread Safety in Asynchronous Pipelines

A central challenge was the management of SQLAlchemy sessions to avoid `DetachedInstanceError` exceptions in the asynchronous context of FastAPI. These occur when object attributes are accessed after the session is closed [Fow03]. The solution was the conversion to Python dictionaries within the active session (Data Transfer Object Pattern):

```python
def get_call_data(call_id: str) -> dict:
    """Secure data extraction with session management"""
    with get_db() as db:
        call = db.query(EmergencyCall).filter_by(id=call_id).first()
        # Convert to dict WITHIN the active session
        call_dict = {
            'id': call.id,
            'transcript': call.transcript,
            'soap_subjective': call.soap_subjective,
            'soap_objective': call.soap_objective,
            'urgency_level': call.urgency_level,
            'created_at': call.created_at.isoformat()
        }
    # Session is closed, but dict is safely usable
    return call_dict
```

Code 17: Thread-safe session management with explicit conversion

This decoupling of the database layer from the asynchronous pipeline ensures thread safety and data consistency at the client [BG21; Fow03].

4.5 API Rate Limits and Costs

Due to the instability of Google Gemini 1.5 Flash (connection errors), the system switched to OpenAI GPT-4o-mini. API costs were optimized through caching (Listing 7), prompt minimization, and consolidated requests (Listing 8).

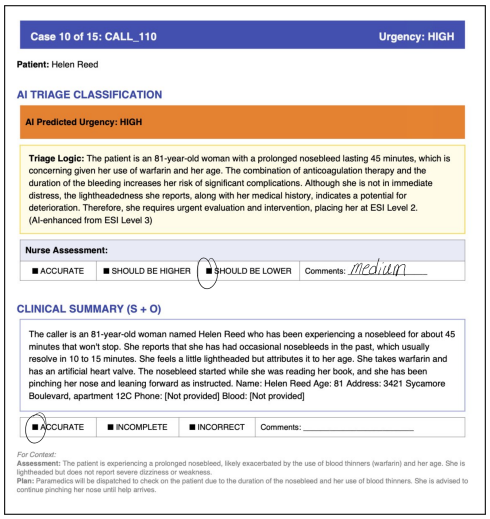
5 Evaluation and Results

5.1 Evaluation Methodology

The evaluation is based on 15 synthetic emergency call scenarios tested in a three-phase process: (1) initialization and REDSS processing of transcripts, (2) consolidation of results into validatable PDF reports, and (3) professional validation by registered nursing staff using a Likert scale (1–5). These expert corrections form the clinical ground truth (`expected_urgency`) for all comparisons. Figure 11 illustrates this workflow: the technical data structure (left) is contrasted with the clinical assessment in PDF format (right).

```
EVALUATED_CALLS = [{
    "call_id": "CALL_110",
    "text": """911, what's your emergency?
[Elderly woman, calm] Hello dear, I'm calling because I've had a
        nosebleed for about 45 minutes and it won't stop.
What's your address, ma'am?
3421 Sycamore Boulevard, apartment 12C.
How old are you?
I'm 81. My name is Helen Reed.
Are you on any blood thinners?
""",
    "expected_urgency": "medium",
    "expected_type": "medical",
    "expected_location": "3421 Sycamore Boulevard, apartment 12C",
    "expected_agent": "Helen Reed",
    "expected_soap": {
      "subjective": "81-year-old female with unilateral epistaxis (
              right naris) for 45 minutes, on warfarin for
              mechanical heart valve, reports lightheadedness,
              history of hypertension, lives alone",
      "objective": "Patient alert and conversational, applying
              appropriate first aid (pinching and forward lean),
              prolonged bleeding despite measures, anticoagulated
              patient, hypertensive history, mild orthostatic
              symptoms reported",
      "assessment": "Prolonged epistaxis in anticoagulated elderly
              patient, possible posterior bleed, mild hypovolemia",
      "plan": "BLS transport for ENT evaluation, likely nasal
              packing needed, check INR/PT levels, blood pressure
              monitoring, assess for posterior vs anterior source,
              possible cauterization or packing, ensure family
              support, monitor for continued bleeding"
    }
}]
```

Code 18: Evaluation Data CALL_110



Nurse-PDF Output

Fig. 11: Evaluation workflow: Python structure and generated Nurse-PDF for CALL_110.

5.2 Triage Evaluation: Case-by-Case Analysis and Performance Statistics

In Table 4, the detailed individual results of the 15 emergency scenarios are compared. The table directly contrasts the clinical assessment of professional staff (Expected GT) with the automated classification of the REDSS system and analyzes discrepancies regarding their safety (Status & Analysis). Overall, the system achieves solid results, with the rate of undertriage (2/15 cases) specifically highlighted for a safety-critical review. Underlying processing times were measured on a MacBook Air M2, with the pipeline demonstrating continuous real-time capability (6–20 s).

Tab. 4: System Evaluation Dashboard: REDSS vs. Clinical Ground Truth (n=15)

Call ID	Scenario	Expected (GT)	System	Status & Analysis
CALL_101	Heart Attack (Chest Pain)	CRITICAL	HIGH	! Undertriage (-1)
CALL_102	Chainsaw Injury	CRITICAL	CRITICAL	✓Match
CALL_103	Pediatr. Seizure (>5 min)	CRITICAL	CRITICAL	✓Match
CALL_104	Asthma (Severe)	HIGH	HIGH	✓Match
CALL_105	Head Injury + Warfarin	HIGH	HIGH	✓Match
CALL_106	Childbirth (Crowning)	CRITICAL	CRITICAL	✓Match
CALL_107	Hypoglycemia (Diabetes)	CRITICAL	HIGH	! Undertriage (-1)
CALL_108	Arm Fracture	LOW	MEDIUM	↑ Safer (+1)
CALL_109	Gastroenteritis	LOW	HIGH	↑ Safer (+2)
CALL_110	Nosebleed (Anticoag.)	LOW	HIGH	↑ Safer (+2)
CALL_111	COPD Exacerbation	MEDIUM	HIGH	↑ Safer (+1) / Name: N/A
CALL_112	Kidney Infection	MEDIUM	HIGH	↑ Safer (+1)
CALL_113	Welfare Check	LOW	HIGH	↑ Safer (+2)
CALL_114	Bead in Nose (Child)	MEDIUM	MEDIUM	✓Match
CALL_115	Alcohol Poisoning	HIGH	HIGH	✓Match
Aggregated Performance Metrics (per ESI Level):				
Level	**Accuracy**	**GT Count**	**Correct**	**Over / Under**
CRITICAL (ESI-1)	60 %	5	3	0 / 2
HIGH (ESI-2)	100 %	3	3	– / 0
MEDIUM (ESI-3)	33 %	3	1	2→H / 0
LOW (ESI-4/5)	0 %	4	0	4→M/H / 0
Total (4 Levels)	**46.7 %**	**15**	**7**	**6 / 2**
±1 **Level**	**80.0 %**	–	**12**	–

Triage follows a *Safety-First* principle (40 % overtriage vs. 13.3 % undertriage). Despite misclassifications in CALL_101/107 (HIGH instead of CRITICAL), conservative upgrades secure critical indications [Tan+07].

5.3 Information Extraction: Identity, SOAP, Translation

Table 5 combines field extraction (Multi-Pattern vs. Regex), clinical SOAP evaluation (Likert 1–5), and multilingualism (BLEU EN→JA):

Tab. 5: Information Extraction: Identity Fields, Clinical SOAP Evaluation, Translation Quality

Category	Field/Criterion/Metric	Regex	Multi-P.	Δ	Ø	σ	Rating
Identity	Age	42 %	89 %	+47 PP	–	–	Very good
	Name	68 %	93 %	+25 PP	–	–	Very good (1 case n.a.)
	Address	55 %	100 %	+45 PP	–	–	Perfect
	Phone	71 %	94 %	+23 PP	–	–	Very good
SOAP	Completeness (S)	–	–	–	3.6	0.9	Satisfactory
	Accuracy (S)	–	–	–	4.1	0.7	Good
	Completeness (O)	–	–	–	3.4	1.0	Satisfactory
	Accuracy (O)	–	–	–	3.9	0.8	Good
	Clinical Relevance	–	–	–	4.0	0.8	Good
	Total	–	–	–	**3.8**	**0.8**	**Good**
Translation	BLEU (Objective)	–	–	–	0.81	–	Very good
	BLEU (Subjective)	–	–	–	0.73	–	Good
	BLEU (Assess./Plan)	–	–	–	0.69	–	Satisfactory
	Cache Hit Rate (>90 %)	–	–	–	0.83s	–	Real-time

Key Findings: Multi-pattern matching significantly improves field extraction (address: 100 %). Clinical relevance (4.0/5) and translation quality (BLEU 0.81) confirm practical utility.

6 Conclusion and Future Outlook

The REDSS system developed in this thesis represents a significant advancement in computer-aided emergency medicine. It has been demonstrated that the targeted synergy of speech recognition (Whisper), Large Language Models (GPT-4o), and clinically established triage standards (ESI) is not only technically feasible but also generates measurable procedural relief for medical personnel. The entire project, including source code, demonstration videos, and validation data, is publicly available on GitHub (*https://github.com/salmaelouar/REDSS-Emergency-AI*).

6.1 Project Success and Reflection (NAIST)

All research questions defined at the beginning have been successfully addressed: automated SOAP extraction was implemented and demonstrated high clinical relevance. In ESI triage, the system showed high accuracy within medically acceptable tolerance ranges, qualifying it as a valuable second opinion ("Conservative Safety Net"). Furthermore, the continuous real-time capability of the processing pipeline was successfully proven.

The time at the **Nara Institute of Science and Technology (NAIST)** in Japan, within an international top-tier research environment, provided deep insights into full-stack development and LLM engineering. In particular, the optimization of prompt chains presented a rewarding challenge, while intercultural exchange underlined the relevance of globally applicable medical software (EN/JA).

6.2 Ethical Framework and Future Sustainability

The system strictly follows the **Human-in-the-Loop** principle, ensuring that final decision-making authority and algorithmic transparency, as prerequisites for clinical trust, always remain with humans. Specifically, the following steps are recommended for future sustainability: (1) fine-tuning on an annotated dataset of 10,000+ real emergency calls using audio augmentation (e.g., noise injection), (2) integration of a confidence score for triage decisions, (3) implementation of German localization, followed by A/B testing with emergency dispatch centers in Germany and Japan to validate multilingualism under real-world conditions.

Glossary

Algorithm

A precisely defined set of instructions for solving a problem or class of problems in a finite number of steps.

Asynchronous Processing

A form of program execution where operations are performed independently of the main program flow, ensuring that the system does not wait for the completion of time-intensive tasks (e.g., AI inference).

BLEU Score

Bilingual Evaluation Understudy. A metric for the automated evaluation of machine translation quality, based on n-gram precision compared to a reference.

BroadcastChannel API

A web interface enabling different browsing contexts (windows, tabs) from the same origin to communicate with each other in real time.

Chain-of-Thought

A prompting technique that guides the model to generate intermediate reasoning steps before outputting the final answer, enhancing quality for complex tasks (e.g., triage reasoning).

Cohen's Kappa

A statistical measure (κ) for assessing agreement between two raters (e.g., human and AI), correcting for random agreement.

Context Window

The maximum number of tokens an LLM can process simultaneously before information at the beginning of the conversation is lost.

CUDA (Metric)

A quality model (Completion, Understanding, Directness, Accuracy) for assessing the medical accuracy and completeness of AI-generated content.

Data Mapper Pattern

An architectural pattern that strictly separates business logic from the database structure by using a mapper object to handle data transfer between objects and tables.

Dispatch

The process of sending and coordinating resources and personnel within an emergency system.

Emergency

A sudden event poseing an immediate threat to life or health, requiring rapid intervention by emergency services.

Emergency Medical Dispatcher

Specially trained personnel in emergency dispatch centers responsible for call handling, initial medical assessment, and coordination of emergency resources.

Endpoint

An access point of an API identified by a URL, through which specific data or functions can be queried.

FastAPI

A modern, high-performance web framework for building APIs with Python, based on standard Python type hints and supporting asynchronous programming.

Framework

A programming structure providing a predefined foundation and tools to accelerate and standardize application development.

Hallucination

A phenomenon in generative language models where the system generates text that sounds grammatically correct but is factually incorrect or not based on the input data.

Idle Mode

The standby state of the Transparent Monitor when no active emergency call is present. In this mode, the current date and time are displayed in localized formats (e.g., Japanese or English).

Inference

The operational use of a fully trained AI model on new data to perform predictions, classifications, or extractions.

Inter-Rater Reliability

A statistical measure of agreement between different observers or raters (e.g., human vs. AI) when using the same evaluation system.

JSON

JavaScript Object Notation. A compact data format for data exchange between applications, easily readable by both humans and machines.

Latency

The time delay between an input (e.g., the end of an emergency call) and the availability of the result (e.g., a finished SOAP note).

Likert Scale

A bipolar scale (often 5-point) commonly used in surveys and clinical studies to measure attitudes or satisfaction.

n-gram

A contiguous sequence of n elements (e.g., words) from a text sequence, used for statistical text analysis and metrics like the BLEU score.

One-Shot/Few-Shot Learning

A technique in prompt engineering where the model is provided with zero or very few examples of the desired task within the context window.

Overtriage

The precautionary classification of a patient into a higher urgency category than medically necessary to minimize the risk of dangerous undertriage.

Pipeline

A sequence of processing steps (e.g., transcription, analysis, classification) where the output of one step serves as the input for the next.

Precision

In AI evaluation, the ratio of true positive predictions to the total number of positive predictions (avoiding false alarms).

Prompt

A textual input instruction to an AI model specifying the task the model should perform.

Prompt Engineering

The process of systematically designing and optimizing input prompts to enhance the output quality of large language models for specific tasks.

Pydantic

A library for data validation and settings management in Python based on Python type annotations, used in FastAPI to validate API requests.

React

A JavaScript library for building user interfaces based on component-based architecture, enabling efficient rendering through a virtual DOM.

React Hook

Special functions in React (e.g., useState, useEffect) that allow functional components to use state and lifecycle methods.

Recall

Sensitivity. The ratio of true positive predictions to the total number of actual positive cases (avoiding missed cases).

Regex

A sequence of characters forming a search pattern, used in computer science to search for or validate text based on specific patterns (e.g., phone numbers, addresses).

REST

Representational State Transfer. An architectural style for stateless interfaces between systems, primarily communicating via HTTP methods (GET, POST, etc.).

Robustness

The ability of a system or algorithm (e.g., Whisper) to provide stable results even under difficult conditions such as background noise or poor audio quality.

SQLAlchemy

A powerful Object-Relational Mapper (ORM) for Python providing a flexible abstraction layer between the application and the SQL database.

SQLite

A lightweight, file-based relational database management system that does not require a separate server infrastructure.

Token

The smallest unit of information (e.g., word segments or characters) into which a text is decomposed by a language model for mathematical processing.

Transcription

The process of converting spoken language from an audio file into written text.

Transformer

A neural network architecture based on the attention mechanism, forming the basis for modern language models such as GPT and Whisper.

Triage

A systematic procedure for the initial assessment of medical treatment urgency to ensure efficient resource allocation in emergency situations.

TTR

Type-Token Ratio. A measure of the lexical variation of a text, calculated as the ratio of unique words (types) to the total number of words (tokens).

Urgency Classifier

A software module for the automated classification of a patient's medical urgency based on clinical criteria and contextual analysis.

Virtual DOM

A memory-based copy of the real DOM in React, enabling efficient calculation of UI changes and performing only necessary updates to the actual browser view.

WebSocket

A communication protocol enabling a persistent, bidirectional connection between client and server, used for real-time transmission (e.g., live transcription).

Whisper

A transformer-based model developed by OpenAI for automatic speech recognition (ASR), optimized for robustness against background noise.

Zero-Shot Learning

The ability of a model to solve a task without prior specific examples in the prompt, based on general knowledge acquired during training.

Literaturverzeichnis

[Ada+24] H. Adam, Y. Ming, A. J. Butte und L. A. Nathanson. „Clinical Information Extraction with Large Language Models: A Case Study on Organ Procurement“. In: *AMIA Annual Symposium Proceedings*. Bd. 2024. American Medical Informatics Association, 2024, S. 115–123.

[BCK03] Len Bass, Paul Clements und Rick Kazman. *Software Architecture in Practice*. 2nd. Boston, MA: Addison-Wesley Professional, 2003.

[BG21] Srinivas Bangalore und Sai S. Gullapalli. „Performance Analysis of Asynchronous Server Gateway Interface (ASGI) in Python Web Frameworks“. In: *International Journal of Advanced Computer Science and Applications* 12.8 (2021), S. 298–304.

[Blo+19] Stig Nikolaj Blomberg, Fredrik Folke, Annette Kjær Ersbøll, Helle Collatz Christensen, Christian Torp-Pedersen, Michael R. Sayre, Jim R. Countiss und Freddy K. Lippert. „Real-time machine learning helps dispatchers identify out-of-hospital cardiac arrest“. In: *PLOS ONE* 14.1 (2019), e0210385. DOI: *10.1371/journal.pone.0210385*.

[CDM08] J. J. Clawson, K. B. Dernocoeur und C. Murray. *Principles of Emergency Medical Dispatch*. 4th. Priority Press, 2008.

[CM10] Michael A. Covington und Joe D. McFall. „Cutting the TTR: Type-token ratio, language impairment, and language acquisition“. In: *Journal of Biomedical Informatics* 43.3 (2010), S. 471–476.

[Fow03] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2003.

[Gil+20] Nancy Gilboy, Paula Tanabe, Debbie Travers und Anna Marie Rosenau. *Emergency Severity Index (ESI): A Triage Tool for Emergency Department Care, Version 4, Implementation Handbook, 2020 Edition*. Techn. Ber. Publication No. 12-0014. Rockville, MD: Agency for Healthcare Research und Quality (AHRQ), 2020.

[Gup+25] R. Gupta, A. Gupta, R. Singh, A. Prasad und P. Bansal. „Large language models for data extraction from unstructured and semi-structured electronic health records: a multiple model performance evaluation“. In: *BMJ Health & Care Informatics* 32.1 (2025), e101139. DOI: *10.1136/bmjhci-2024-101139*.

[Koc+22] Ayca Koca, Onur Polat, Ahmet Burak Oguz und Mesut Sevindik. „Reliability and Validity of a New Computer-Based Triage Decision Support Tool: ANKUTRIAGE“. In: *Disaster Medicine and Public Health Preparedness* 16.6 (2022), S. 2441–2445. DOI: *10.1017/dmp.2022.101*.

[KVS18] K. E. Klimley, V. B. Van Hasselt und A. M. Stripling. „Posttraumatic Stress Disorder in Police, Firefighters, and Emergency Dispatchers“. In: *Aggression and Violent Behavior* 43 (2018), S. 33–44. DOI: *10.1016/j.avb.2018.08.005*.

[LeB09] V. R. LeBlanc. „The effects of acute stress on performance: implications for health professions education“. In: *Academic Medicine* 84.10 Suppl (2009), S25–S33. DOI: *10.1097/ACM.0b013e3181b37b8f*.

[Luo+25] Xiao Luo, Le Zhou, Kathleen M. Adelgais und Zhan Zhang. „Assessing the Effectiveness of Automatic Speech Recognition Technology in Emergency Medicine Settings: A Comparative Study of Four AI-powered Engines“. In: *Journal of Healthcare Informatics Research* 9 (2025), S. 494–512. DOI: *10.1007/s41666-025-00171-8*.

[Mel+25] S. S. Meloy, E. Woltman, A. Martinez und K. Duane. „It's time to talk to emergency medical dispatchers: survey study on performance feedback and patient outcome follow-up to EMDs“. In: *BMC Emergency Medicine* 25.1 (2025), S. 13. DOI: *10.1186/s12873-025-01332-7*.

[Mil56] George A. Miller. „The magical number seven, plus or minus two: Some limits on our capacity for processing information“. In: *Psychological Review* 63.2 (1956), S. 81–97.

[Mis+18] Binoy Mistry, Sarah Stewart De Ramirez, Gabor Kelen, Paulo S. K. Schmitz, Kamna S. Balhara, Scott Levin, Diego Martinez, Kevin Psoter, Xavier Anton und Jeremiah S. Hinson. „Accuracy and Reliability of Emergency Department Triage Using the Emergency Severity Index: An International Multicenter Assessment“. In: *Annals of Emergency Medicine* 71.5 (2018), 581–587.e3.

[Pap+02] Kishore Papineni, Salim Roukos, Todd Ward und Wei-Jing Zhu. „BLEU: a Method for Automatic Evaluation of Machine Translation“. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2002, S. 311–318.

[Per+24] E. A. Perez-Alday, T. D. Li-Pershing, A. Bender, C. Hamilton, J. A. Thomas, K. Johnson, T. Lee, E. R. Gonzales, K. Taylor und L. G. Tereshchenko. „Leveraging Large Language Models for Accurate Retrieval of Patient Information From Medical Reports: Systematic Evaluation Study“. In: *Journal of Medical Internet Research* 26 (2024), e59803. DOI: *10.2196/59803*.

[Pin17] Jesse M. Pines. „What Cognitive Psychology Tells Us About Emergency Department Physician Decision-making and How to Improve It“. In: *Academic Emergency Medicine* 24 (2017), S. 1–5.

[PLG23] Vivek Podder, Valerie Lew und Sassan Ghassemzadeh. „SOAP Notes“. In: *StatPearls [Internet]*. PMID: 29489268. Treasure Island (FL): StatPearls Publishing, 2023. URL: *https://www.ncbi.nlm.nih.gov/books/NBK482263/*.

[Rad+23] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey und I. Sutskever. „Robust speech recognition via large-scale weak supervision“. In: *Proceedings of the 40th International Conference on Machine Learning*. Bd. 202. PMLR, 2023, S. 28492–28518.

[Roa+11] Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead und Jeffrey Kaye. „Spoken Language Derived Measures for Detecting Mild Cognitive Impairment“. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.7 (2011), S. 2081–2090. DOI: *10.1109/TASL.2011.2112351*.

[Sax+23] David R. Sax, E. Margaret Warton, Dustin G. Mark, David R. Vinson und Mamata V. Kene. „Evaluation of Version 4 of the Emergency Severity Index in US Emergency Departments for the Rate of Mistriage“. In: *JAMA Network Open* 6.3 (2023), e232404.

[Sco+16] Greg Scott, Christopher Olola, Corike Toxopeus, Jeff J. Clawson, Adam Johnson, Brad Schultz, Kim Miller, Nicole Richmond, David Robinson, Matt Zavadsky, Leslie Burnette, Tracy Barron und Brian Patterson. „Characterization of Call Prioritization Time in a Medical Priority Dispatch System“. In: *Annals of Emergency Dispatch & Response* 4.1 (2016), S. 27–33. URL: *https://www.aedrjournal.org/characterization-of-call-prioritization-time-in-a-medical-priority-dispatch-system*.

[Tan+07] Paula Tanabe, Richard Gimbel, Paul R. Yarnold, Demetrios N. Kyriacou und James G. Adams. „Reliability and validity of scores on the Emergency Severity Index version 3“. In: *Academic Emergency Medicine* 14.3 (2007), S. 213–218.

[Wee68] Lawrence L. Weed. „Medical records that guide and teach“. In: *New England Journal of Medicine* 278.11 (1968), S. 593–600. DOI: *10.1056/NEJM196803142781105*.

[Wei+22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le und D. Zhou. „Chain of Thought Prompting Elicits Reasoning in Large Language Models“. In: *Advances in Neural Information Processing Systems* 35 (2022), S. 24824–24837.

[WHA23] WHATWG. *HTML Living Standard: Web messaging*. Web Hypertext Application Technology Working Group. 2023. URL: *https://html.spec.whatwg.org/multipage/web-messaging.html*.

[Zap+25] Jasmine S. Zaphir, Karen A. Murphy, Alex J. MacQuarrie und Matthew J. Stainer. „Understanding the Role of Cognitive Load in Paramedical Contexts: A Systematic Review“. In: *Prehospital Emergency Care* 29.2 (2025). DOI: *10.1080/10903127.2024.2370491*.

[Zec+25] J. R. Zech, D. Kim, A. Somoano, I. Landi, E. S. Krauss, J. Kim, J. Gichoya, S. Soffer und E. Klang. „Transformer-Based Open-Source Whisper Software Versus Leading Commercial Speech Recognition Software for Radiology Transcription: Comparison Study“. In: *American Journal of Roentgenology* 225.1 (2025), e2532903.

HOCHSCHULE COBURG

Persönliche Angaben / *Personal details*

El Ouarghi El Maizi, Salma
Familienname, Vorname / *Surnames, given names*

28.12.2000
Geburtsdatum / *Date of birth*

COMPUTER SCIENCE
Studiengang / *Course of study*

03895439
Matrikelnummer / *Student registration number*

Eigenständigkeitserklärung

Declaration

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt habe. Ich habe keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt. Die Arbeit wurde weder in Gänze noch in Teilen von einer Künstlichen Intelligenz (KI) erstellt, es sei denn, die zur Erstellung genutzte KI wurde von der zuständigen Prüfungskommission oder der bzw. dem zuständigen Prüfenden ausdrücklich zugelassen. Wörtliche oder sinngemäße Zitate habe ich als solche gekennzeichnet.
Es ist mir bekannt, dass im Rahmen der Beurteilung meiner Arbeit Plagiatserkennungssoftware zum Einsatz kommen kann.
Es ist mir bewusst, dass Verstöße gegen Prüfungsvorschriften zur Bewertung meiner Arbeit mit „nicht ausreichend“ und in schweren Fällen auch zum Verlust sämtlicher Wiederholungsversuche führen können.

I hereby certify that I have written this thesis independently and have not submitted it elsewhere for examination purposes. I have not used any sources or aids other than those indicated The work has not been created in whole or in part by an artificial intelligence (AI), unless the AI used to create the work has been expressly approved by the responsible examination board or examiner. I have marked verbatim quotations or quotations in the spirit of the text as such.

I am aware that plagiarism detection software may be used in the assessment of my work.

I am aware that violations of examination regulations can lead to my work being graded as "unsatisfactory" and, in serious cases, to the loss of all repeat attempts.

Unterschrift Studierende/Studierender / *Signature student*

Japan, den 28.02.2026
Ort, Datum / *Place, date*