Handover document

AI-Powered Technical Sales Assistant

Team Name: Zyntek

Course: Skills 3

Team Members:

Favour · Gabriela · Iljas · Eugène · Tuur · Martin · Tijs

Thomas More University of Applied Sciences

Submitted on: 23 January 2025

Table of Contents

Introduction

This document serves as a comprehensive overview of our AI-Powered Technical Sales Assistant. Its goal is to provide a seamless transition for the incoming team by detailing the project's architecture, development workflow, and deployment pipelines. This is to explain the decisions we made, but also the decisions we decided not to make.

Note: In the Repo there are md files with extra specific configuration details.

Local development

To run the application locally, we use two PowerShell scripts: setupLocalDev.ps1 and runLocalDev.ps1.

Prerequisites

A few tools should be installed to successfully run the setup script:

- Docker Desktop
- Python 3.11
- Ollama
- Node.js & npm

Initial setup – setupLocalDev.ps1

This script prepares the environment by managing dependencies and orchestrating Docker containers. It performs the following steps in order:

- Container Cleanup: Deletes any existing Docker containers to ensure a clean state.
- Database Provisioning: Spins up fresh Docker containers for both the Redis vector database and the PostgreSQL relational database.
- Model Initialization: Pulls the latest Llama3 model via Ollama for local testing.
- Python Environment: Creates a Python virtual environment (if not already present) and installs the necessary dependencies from pythonRequirements.txt.
- Database Schema & Seeding:
    - Runs setupRedis.py to initialize the vector database structure.
    - Runs setupPostgress.py to create the PostgreSQL schema.
    - Runs seederPostgress.py to populate the database with example data for immediate testing.

Execution – runLocalDev.ps1

Once the setup is complete, this script launches the application. It opens three separate terminal windows to run the core services concurrently:

1. Frontend: The React application.
2. Backend: The FastAPI server.
3. MCP Server: The FastMCP integration layer.

Environmental variables

For security and flexibility, the application relies on a `.env` file to manage sensitive configurations. This allows each developer to maintain their own local settings without hardcoding secrets into the source code.

Design – choices and remarks

AI inference – Ollama

We utilized Ollama for model inference due to its versatility and ease of configuration. It functions as a robust, standalone solution that simplifies setup across both local development and production server environments. Notably, Ollama automatically manages multi-GPU distribution, removing the need for manual resource orchestration or complex driver configurations.

One of Ollama's primary strengths is its streamlined model management; libraries can be pulled and updated via simple command-line instructions, significantly reducing the friction of environment setup. However, it is important to note that Ollama stores these models in a proprietary internal format rather than standard .safetensors or GGUF files in a visible directory. This abstraction makes it more difficult to extract weights for fine-tuning or to transition to other inference frameworks, adding a layer of complexity if the project eventually requires custom model training or architecture modifications.

MCP server – FastMCP

Communication with the AI systems is done with a MCP server using FastMCP. This provides a standardized interface for external systems to interact with the AI. By utilizing the Model Context Protocol, we've ensured that the AI's capabilities can be easily consumed by other services without tight coupling. From a development standpoint, this separation made development much smoother. it keeps the backend logic and the AI inference engine in distinct silos. This 'decoupled' approach allows us to iterate on the AI's tools and prompts independently of the core application logic, making the system easier to debug and more resilient to changes in the underlying model.

- **Transport:** Backend connects to the MCP server via streamable HTTP (not SSE for production). Watch for URL and protocol mismatches (e.g. http:// vs https:// on remote hosts).
- **400 Bad Request:** Often caused by wrong URL format or endpoint, rather than payload issues.

RAG and vector database – Redis

To enhance the model's intelligence without sacrificing clarity, we implemented Retrieval-Augmented Generation (RAG). This approach allows the system to remain highly transparent; unlike a 'black-box' AI, we can pinpoint exactly which source documents the model is referencing to generate its answers. Redis was chosen as our vector database to facilitate this, which we had no major problems with. Throughout development and testing, Redis provided a smooth, high-performance experience with no significant hurdles or complaints. Its reliability as a vector store has made the retrieval process both fast and easy to maintain. Additionally, running Redis via Docker for local development made the environment setup incredibly easy and consistent for the team

Relational database – Postgres

For general data management, we used PostgreSQL as our relational database to store things like chats, user feedback, emails, user profiles, etc. Much like our experience with Redis, Postgres performed exceptionally well throughout the project with no major issues. Furthermore, using Postgres with Docker for local development ensured that the database environment was easy to spin up and remained consistent across different machines.

Backend – FastAPI

The backend is built using FastAPI, chosen for its high performance and modern support. It serves as the central hub connecting the user interface, the Postgres database, and the AI components. A major benefit during development was FastAPI's automatic interactive documentation (accessible via `/docs`); this provided an instant, real-time interface to test all API endpoints without needing to write external documentation.

Frontend – React

For the user interface, we used React, a framework that makes making responsive and dynamic pages easier. A major advantage of using React was its component-based structure, which made it easier to manage and update specific parts of the UI independently.

Finetuning (proof of concept) – Unsloth

While the final application does not use a Qwen model, we used Qwen for our finetuning Proof of Concept. The primary reason for this choice was the geolocking of Meta's models (like Llama) in Europe. Due to regulatory constraints and the EU AI Act, Meta restricted access to many of its latest pre-quantized models for European users. Qwen served as a powerful, unrestricted alternative that allowed us to test the finetuning pipeline without regional licensing hurdles.

Note on Unsloth: While Unsloth was installed in the environment setup (setup.sh), the actual finetuning script (finetuning.py) uses the standard Hugging Face stack (transformers, PEFT, TRL, BitsAndBytes) rather than Unsloth's optimized training functions. Unsloth served as a potential optimization path, but the proof-of-concept was completed using the more widely-supported HF toolchain for better compatibility and documentation.

The included finetuning script demonstrates how to process PDF emails into a JSONL dataset and use LoRA (Low-Rank Adaptation) to train the model. The script handles the full lifecycle: 4-bit quantization, training, merging weights, and converting the result into a .gguf file for use in Ollama.

Successful Training: Beyond the Qwen proof-of-concept, we successfully trained a LoRA adapter on Mistral-7B-Instruct-v0.2 using a separate training pipeline (train_lora_hf.py). This adapter achieved significant loss reduction (from 1.74 to 0.0027) and produced a 27MB safetensors adapter file. However, integrating this adapter into the production Ollama pipeline proved challenging (see Integration Challenges below).

Integration Challenges: While LoRA training itself was successful, deploying trained adapters into production revealed several hurdles:

- **Ollama Compatibility:** Ollama doesn't natively support Hugging Face safetensors adapters. Conversion to GGUF format is required, but the convert-lora-to-gguf.py script may not be available in all llama.cpp versions, creating a dependency bottleneck.
- **Vision Model Limitation:** Our vision model (Llama 3.2 Vision) uses the mllama architecture, which is not currently supported by llama.cpp for LoRA training. This means we cannot apply LoRA adapters directly to our vision pipeline—only text-only models can be LoRA-trained via llama.cpp.
- **Multiple Training Approaches:** We experimented with three different training paths: (1) Standard HF PEFT (train_lora_hf.py), (2) llama.cpp finetune (train_lora.py - no Hugging Face dependency), and (3) Unsloth setup. Each has different integration requirements, making the production deployment path non-trivial.

Environment Setup & WSL: A critical part of this workflow is the environment. The script has very strict dependencies on NVIDIA drivers, CUDA versions, and specific builds of PyTorch and bitsandbytes. We found that running this through WSL was significantly easier and more stable than a native Windows installation. WSL allowed us to leverage Linux-native package management and better driver integration, which is often required for cutting-edge AI libraries. However, even with WSL, this section of the codebase remains "fragile" due to these specific hardware-level requirements. Additionally, different training approaches (HF PEFT vs llama.cpp) have different environment needs, further complicating setup. For production deployment, we recommend either: (1) using a dedicated Linux training server, or (2) implementing a Hugging Face backend that can load safetensors adapters directly without GGUF conversion.

Current Status: LoRA training is proven to work (successful Mistral 7B adapter), but full production integration is pending. The adapter exists in safetensors format (./lora/adapters/tse_adapter_mistral/) and can be used with a Hugging Face backend, but integration with our current Ollama-based production pipeline requires additional conversion steps that have not yet been fully automated. See LORA_INTEGRATION.md for detailed integration options and troubleshooting.

Embeddings – Sentence-Transformers

We use Sentence-Transformers (all-MiniLM-L6-v2, 384 dimensions) for generating document embeddings. This model runs locally (CPU or GPU) and integrates cleanly with Redis vector search. It performs well for general semantic similarity. One limitation is that purely semantic search can miss exact technical terms like product codes or model numbers. We researched hybrid search (BM25 + semantic) to address this, but it hasn’t been implemented in production yet.

Continuous Learning – Verified Q/A

We implemented a continuous learning system that improves the knowledge base over time. When drafts are approved with a rating of 4 or higher, they are automatically converted into Verified Q/A pairs. These pairs are embedded and indexed into Redis with a special verified_qa type tag, making them retrievable during RAG queries. The system includes deduplication logic to prevent indexing similar Q/A pairs multiple times. This allows the system to learn from

approved responses without requiring model retraining, creating a self-improving knowledge base that becomes more accurate as more drafts are approved.

RAG – Research vs Production

Our current production RAG pipeline uses single-stage semantic retrieval with quality gates (minimum character thresholds, maximum distance filters) to filter results. During development, we researched several optimization strategies documented in AI/rag_optimization/RESEARCH.md that could improve retrieval accuracy. These include hybrid search combining BM25 keyword matching with semantic similarity, cross-encoder re-ranking for more accurate scoring, and chunk deduplication to optimize context window usage. While these approaches show promise for handling technical terminology and improving answer grounding, they remain research-level implementations and have not yet been integrated into the production pipeline. The research provides a clear roadmap for future improvements, particularly for domain-specific technical queries.

Dual Model Setup

We use separate model paths for text and vision tasks. Text-based RAG queries use Ollama or llama.cpp for inference, while vision tasks (like processing email image attachments) use Llama 3.2 Vision through Ollama. This separation avoids compatibility issues, since vision models and LoRA support differ across frameworks. For example, our vision model uses the mllama architecture, which isn't supported by llama.cpp for LoRA training, so keeping vision separate allows us to use the best tool for each task without framework conflicts.

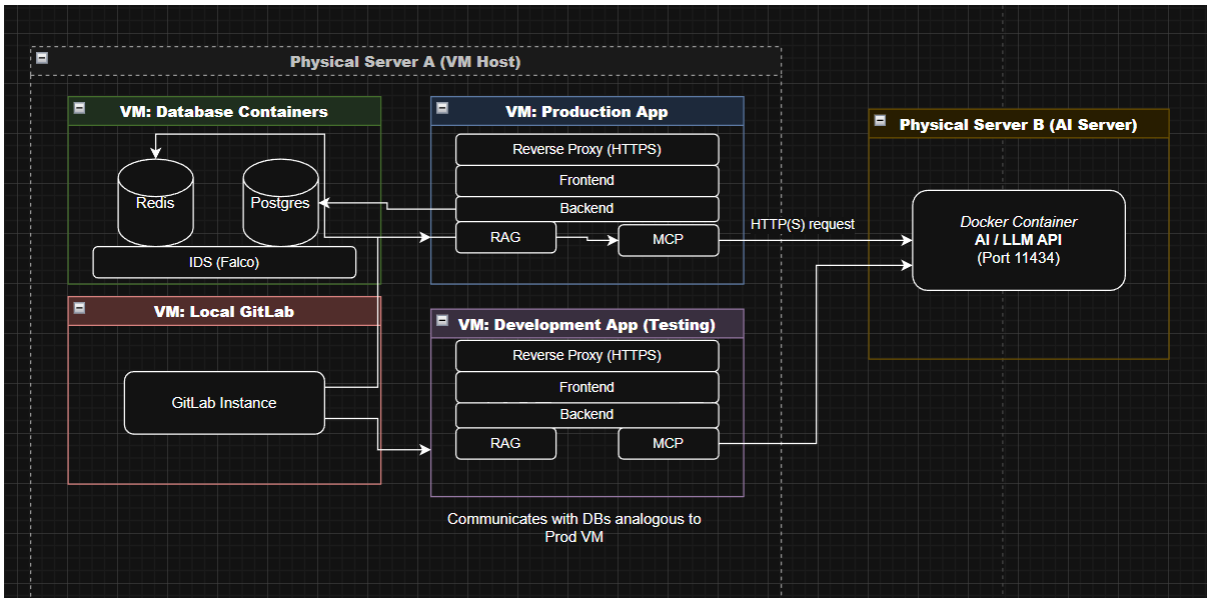
What Worked / Didn't Work Summary

Most components worked well: Ollama for inference with multi-GPU support, FastMCP for decoupled AI communication, Redis for RAG with no major issues, PostgreSQL for data management, Sentence-Transformers for embeddings, and the Verified Q/A continuous learning system. LoRA training succeeded (we trained a Mistral 7B adapter with significant loss reduction), but full production integration is pending due to Ollama compatibility issues. Unsloth finetuning was set up as a proof-of-concept but wasn't fully integrated into the main pipeline. Vision model LoRA training isn't supported by llama.cpp due to mllama architecture limitations. Hybrid search and re-ranking strategies were researched but not implemented in production.

Component	Status	Notes
Ollama inference	Working	Multi-GPU, easy model management; proprietary format
FastMCP	Working	Decoupled AI; streamable HTTP transport
Redis RAG	Working	Smooth, no major issues
Postgres	Working	Used for chats, feedback, emails

Sentence-Transformers	Working	all-MiniLM-L6-v2 for embeddings
Verified Q/A (continuous learning)	Working	Approved drafts → Redis for RAG
LoRA training	Partial	Adapter trained; full production integration pending
Unsloth finetuning	PoC	Setup used; full integration with main pipeline not done
LoRA + Ollama	Blocked	GGUF conversion needed; tool availability issues
Vision model LoRA	Not supported	mllama not supported by llama.cpp
Hybrid search / re-ranking	Researched	Documented; not yet in production

Infrastructure



The infrastructure utilizes a Virtual Machine (VM) approach to separate concerns and limit lateral movement in the event of a security exploit.

- **Server Segmentation:** The system is divided into three distinct VMs for increased security:
    - **Application VM:** Hosts the core application logic.
    - **Gitlab VM:** Runs a local Gitlab instance for code management and pipelines.
    - **Database VM:** Hosting is separated to allow for strict firewall rules.
- **AI Stack:** The backend utilizes an Ollama Docker container for the AI Server. It employs a RAG (Retrieval-Augmented Generation) system backed by Redis (vector database) to ground the AI with specific data.
- **Connectivity:**
    - **Reverse Proxy:** Users access the frontend via a reverse proxy.
    - **MCP Server:** Links the RAG system and data together with the AI.
    - **Mail Service:** A backend service is dedicated to receiving, processing, scanning, and sending emails.

Our repository contains docker compose files to set up the containers on the application VMs, the MCP server is still a python script that needs to be ran manually ([project root]/AI/mcpServer.py). We did this because of testing purposes it is easily and quickly restarted.

Change the .env file to match the passwords of your database containers.

To retrieve mail, we set up a getmail Docker container with docker files. In our repository, there is a mail processor to process said mails in that folder to send it to the database. The docker compose also includes a clamAV antivirus scanner configured to check incoming mail for malware.

CI/CD Automation Pipeline

The project uses a strict pipeline to ensure code quality and security before deployment. The local Gitlab instance automates this process.

Pipeline Workflow:

1. **Feature Branch Push:** Developers push code to a feature branch.
2. **Merge to Development:** Code is merged into the development branch.
3. **Automated Scan & Test:** The pipeline runs a code scan (SonarQube) and starts containers in the testing VM.
4. **Dev Environment Testing:** Developers test the application in the dev environment.
5. **Merge Request (Main):** If successful, a merge request to the Main branch is created.
6. **Approval:** A maintainer must approve the request.
7. **Production Deployment:** Upon approval, the production pipeline triggers, re-scans the code, and automatically deploys to the Production VM.

Setup guide:

The pipeline file is already in the gitlab repository. However, it relies on Gitlab CI/CD variables. And it requires a ssh key so the gitlab runners can access the machine.

DEV_SERVER_IP: The development/testing server IP

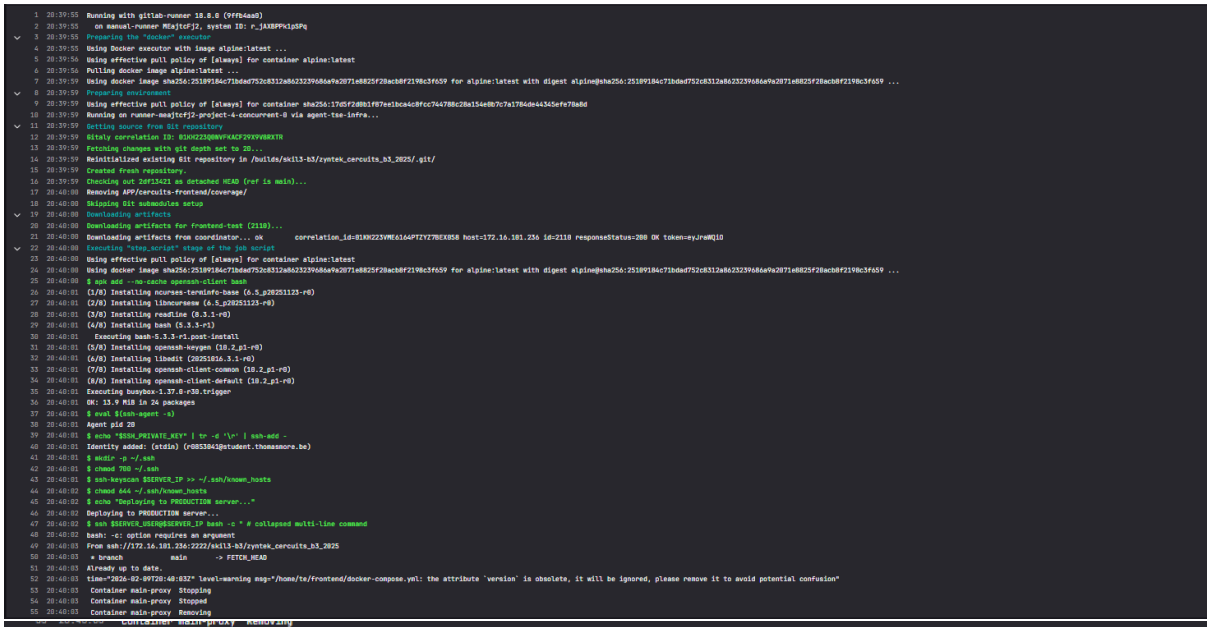
PROD_SERVER_IP: The production server IP

SERVER_USER: username of server account

SSH_PRIVATE_KEY: Generate a ssh keypair and put the private key here, put the public key on the servers the gitlab runners should access (so the development server and the production server) in the trusted/authorized keys folder

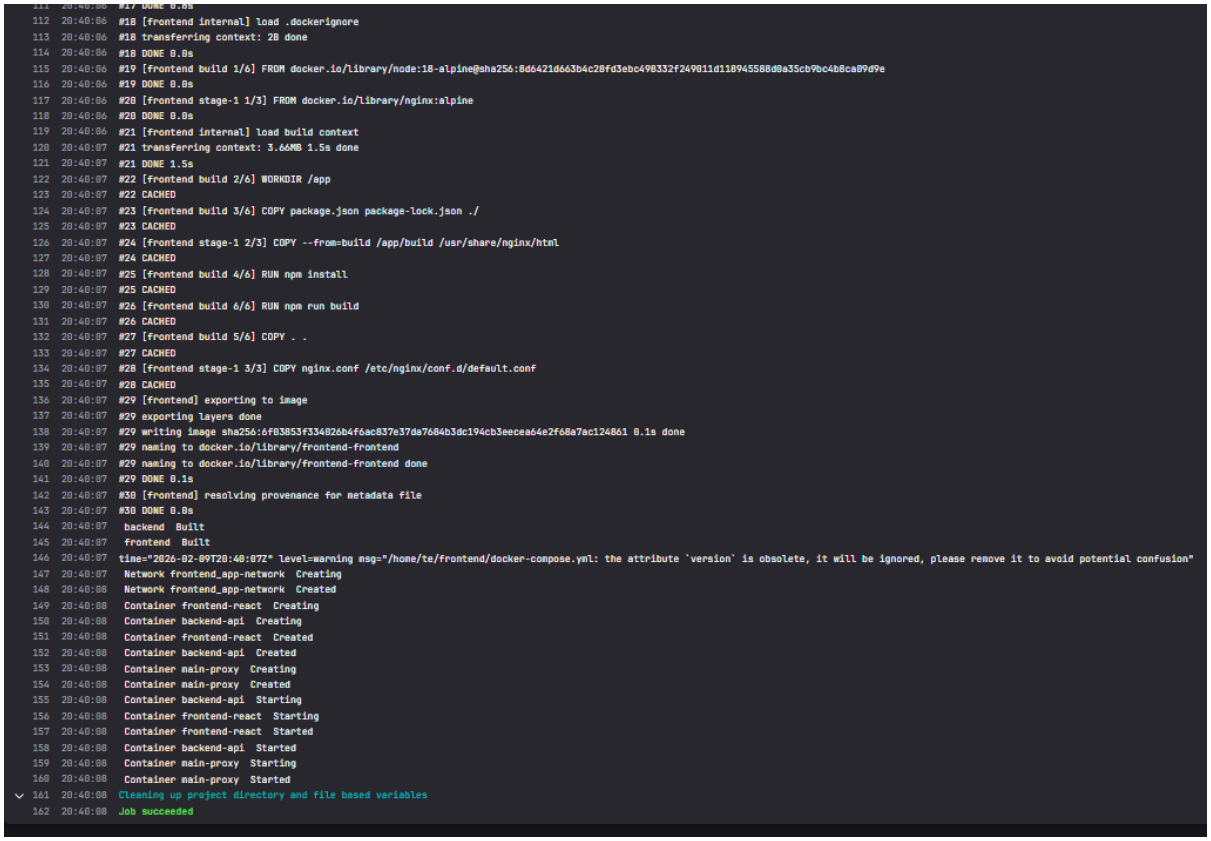
Note: our pipeline contains sonarqube scanning as well, if you do not use this, simply remove the front end-test job and the sonarqube-check jobs, you also won't need the sonar-related variables that way (SONAR_HOST_URL, SONAR_TOKEN and SONAR_PROJECT_KEY)

Here is what the pipeline should look like after it ran:



```
20:39:55  Running with gitlab-runner 18.8.0 (9ffb4aa8)
20:39:55    on manual-runner MEajtcFj2, system ID: r_jAX8PPk1p5Pq
20:39:55  Preparing the "docker" executor
20:39:55  Using Docker executor with image alpine:latest ...
20:39:56  Using effective pull policy of [always] for container alpine:latest
20:39:56  Pulling docker image alpine:latest ...
20:39:59  Using docker image sha256:25109184c71bdad752c8312a8623239686a9a2071e8825f28acb8f2198c3f659 for alpine:latest with digest alpine@sha256:25109184c71bdad752c8312a8623239686a9a2071e8825f28acb8f2198c3f659 ...
20:39:59  Preparing environment
20:39:59  Using effective pull policy of [always] for container sha256:17d5f2d8b1f87ee1bca4c8fcc744788c28a154e8b7c7a1784de44345efe78a8d
20:39:59  Running on runner-meajtcfj2-project-4-concurrent-0 via agent-tse-infra...
20:39:59  Getting source from Git repository
20:39:59  Gitaly correlation ID: 01HX223QQ0WVFK4CF2VXPV8RXTR
20:39:59  Fetching changes with git depth set to 20...
20:39:59  Reinitialized existing Git repository in /builds/skil3-b3/zyntek_cercuits_b3_2025/.git/
20:39:59  Created fresh repository.
20:39:59  Checking out 2df13421 as detached HEAD (ref is main)...
20:40:00  Removing APP/cercuits-frontend/coverage/
20:40:00  Skipping Git submodules setup
20:40:00  Downloading artifacts
20:40:00  Downloading artifacts for frontend-test (2118)...
20:40:00  Downloading artifacts from coordinator... ok       correlation_id=01KH223VME6164PTZYZ7BEX858 host=172.16.101.236 id=2118 responseStatus=200 OK token=eyJreWQiO
20:40:00  Executing "step_script" stage of the job script
20:40:00  Using effective pull policy of [always] for container alpine:latest
20:40:00  Using docker image sha256:25109184c71bdad752c8312a8623239686a9a2071e8825f28acb8f2198c3f659 for alpine:latest with digest alpine@sha256:25109184c71bdad752c8312a8623239686a9a2071e8825f28acb8f2198c3f659 ...
20:40:00  $ apk add --no-cache openssh-client bash
20:40:01  (1/8) Installing ncurses-terminfo-base (6.5_p20251123-r0)
20:40:01  (2/8) Installing libncursesw (6.5_p20251123-r0)
20:40:01  (3/8) Installing readline (8.3.1-r0)
20:40:01  (4/8) Installing bash (5.3.3-r1)
20:40:01    Executing bash-5.3.3-r1.post-install
20:40:01  (5/8) Installing openssh-keygen (10.2_p1-r0)
20:40:01  (6/8) Installing libedit (20251016.3.1-r0)
20:40:01  (7/8) Installing openssh-client-common (10.2_p1-r0)
20:40:01  (8/8) Installing openssh-client-default (10.2_p1-r0)
20:40:01  Executing busybox-1.37.0-r30.trigger
20:40:01  OK: 13.9 MiB in 24 packages
20:40:01  $ eval $(ssh-agent -s)
20:40:01  Agent pid 28
20:40:01  $ echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
20:40:01  Identity added: (stdin) (r0853841@student.thomasmore.be)
20:40:01  $ mkdir -p ~/.ssh
20:40:01  $ chmod 700 ~/.ssh
20:40:01  $ ssh-keyscan $SERVER_IP >> ~/.ssh/known_hosts
20:40:02  $ chmod 644 ~/.ssh/known_hosts
20:40:02  $ echo "Deploying to PRODUCTION server..."
20:40:02  Deploying to PRODUCTION server...
20:40:02  $ ssh $SERVER_USER@$SERVER_IP bash -s " # collapsed multi-line command
20:40:02  bash: -c: option requires an argument
20:40:03  From ssh://172.16.101.236:2222/skil3-b3/zyntek_cercuits_b3_2025
20:40:03   * branch            main       -> FETCH_HEAD
20:40:03  Already up to date.
20:40:03  time="2026-02-09T20:40:03Z" level=warning msg="/home/te/frontend/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
20:40:03   Container main-proxy  Stopping
20:40:03   Container main-proxy  Stopped
20:40:03   Container main-proxy  Removing
```

```
20:40:03   Container main-proxy  Removed
20:40:03   Container backend-api  Stopping
20:40:03   Container frontend-react  Stopping
20:40:03   Container frontend-react  Stopped
20:40:03   Container frontend-react  Removing
20:40:03   Container frontend-react  Removed
20:40:05   Container backend-api  Stopped
20:40:05   Container backend-api  Removing
20:40:05   Container backend-api  Removed
20:40:05   Network frontend_app-network  Removing
20:40:05   Network frontend_app-network  Removed
20:40:05  time="2026-02-09T20:40:05Z" level=warning msg="/home/te/frontend/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
20:40:05  time="2026-02-09T20:40:05Z" level=warning msg="Docker Compose is configured to build using Bake, but buildx isn't installed"
20:40:05  #0 building with "default" instance using docker driver
20:40:05  #1 [frontend internal] load build definition from Dockerfile
20:40:05  #1 transferring dockerfile: 681B done
20:40:05  #1 WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)
20:40:05  #1 DONE 0.0s
20:40:05  #2 [backend internal] load build definition from Dockerfile
20:40:05  #2 transferring dockerfile: 770B done
20:40:05  #2 DONE 0.0s
20:40:05  #3 [frontend internal] load metadata for docker.io/library/nginx:alpine
20:40:05  #3 DONE 0.0s
20:40:05  #4 [backend internal] load metadata for docker.io/library/python:3.12-slim
20:40:05  #4 DONE 0.0s
20:40:05  #5 [backend internal] load .dockerignore
20:40:05  #5 transferring context: 2B done
20:40:05  #5 DONE 0.0s
20:40:05  #6 [backend 1/8] FROM docker.io/library/python:3.12-slim
20:40:05  #6 DONE 0.0s
20:40:05  #7 [backend internal] load build context
20:40:05  #7 transferring context: 15.83kB 0.0s done
20:40:05  #7 DONE 0.0s
20:40:05  #8 [backend 2/8] WORKDIR /app
20:40:05  #8 CACHED
20:40:05  #9 [backend 6/8] COPY setupPostgress.py /app/setupPostgress.py
20:40:05  #9 CACHED
20:40:05  #10 [backend 3/8] RUN apt-get update && apt-get install -y     gcc     postgresql-client     && rm -rf /var/lib/apt/lists/*
20:40:05  #10 CACHED
20:40:05  #11 [backend 5/8] RUN pip install --no-cache-dir -r requirements.txt
20:40:05  #11 CACHED
20:40:05  #12 [backend 4/8] COPY APP/backend/requirements.txt .
20:40:05  #12 CACHED
20:40:05  #13 [backend 7/8] COPY AI/ /app/AI/
20:40:05  #13 CACHED
20:40:05  #14 [backend 8/8] COPY APP/backend/ /app/
20:40:05  #14 CACHED
20:40:05  #15 [backend] exporting to image
20:40:05  #15 exporting layers done
20:40:05  #15 writing image sha256:c7085af15b817d8814ed9e5539f54b60653e87a46a61b11d2b3943530bf80455 done
20:40:05  #15 naming to docker.io/library/frontend-backend done
20:40:05  #15 DONE 0.0s
20:40:05  #16 [backend] resolving provenance for metadata file
20:40:05  #16 DONE 0.0s
20:40:05  #17 [frontend internal] load metadata for docker.io/library/node:18-alpine
20:40:06  #17 DONE 0.8s
```



Security Implementation

Security is implemented at both the network level (infrastructure) and the application level (code).

Framework Alignment

The security architecture of this solution is aligned with recognized industry frameworks to ensure a structured and comprehensive security posture.

The implementation specifically aligns with:

- **OWASP Top 10 (2021)** – Mitigation of common web application vulnerabilities
- **CIS Critical Security Controls (v8)** – Secure configuration and infrastructure hardening
- **NIST Cybersecurity Framework (CSF)** – Risk-based approach covering Identify, Protect, Detect, Respond, and Recover

The implemented controls across infrastructure, application, and monitoring layers directly map to these standards.

Infrastructure & Network Security

- **Firewalling:** Connections to the database are blocked by a firewall, allowing access *only* from the frontend/application VM.
- **Intrusion Detection System (IDS):** Falco is used as an IDS. Alerts regarding suspicious activity (e.g., unauthorized access to DB files) are sent directly to Discord.
- **Mail Security:** Incoming emails are pulled from a Gmail/IMAP server and immediately scanned for malware (using clamAV). Infected files (like Eicar test signatures) are moved to quarantine.
- **SSH Protection:** Fail2Ban is configured to automatically block IP addresses attempting SSH brute-force attacks.
- **Gitlab:** Connections to the Gitlab instance are secured via HTTPS.
- **Backups:** Manual/automated backups are possible with a script we made that its in repo.

Application Security (Backend)

The application implements several distinct layers of security middleware and validation.

1. Rate Limiting Implemented via `slowapi`, limits are applied globally and on specific endpoints to prevent abuse.

Endpoint / Action	Limit
Login	10 requests / minute
General Endpoints	60 requests / minute
Document Upload	20 requests / hour

2. CORS Configuration Enhanced Cross-Origin Resource Sharing (CORS) is configured with explicit origin whitelisting rather than allowing all (*).

- **Config:** Origins are set in `.env` via `BACKEND_CORS_ORIGINS` (e.g., `["http://localhost:3000"]`).
- **Middleware:** Allows standard methods (GET, POST, etc.) and headers (Authorization), with a max age of 3600 seconds.

3. Input & File Validation

- **Data Models:** Pydantic models validate inputs, such as sanitizing emails to allow specific internal domains.
- **File Uploads:** Restricted to specific extensions (`.pdf`, `.docx`, etc.) and a maximum file size of 50MB.

Production Features & Monitoring

The backend includes specific features to ensure stability and observability in a production environment.

Logging & Monitoring

- **Middleware Order:** The application enforces a strict middleware order: 1. Security Headers, 2. Request Logging, 3. CORS.
- **Structured Logging:** In production, logs are formatted as JSON to facilitate log aggregation.
- **Request Logging:** Captures IP, method, path, status, and response time for every request.
- **Security Events:** Specific loggers capture failed login attempts and account lockouts.

Health Checks

A comprehensive `/health` endpoint checks the status of critical services.

- **Scope:** Checks connectivity to the Database, Redis, and AI Service.
- **Response:** Returns 503 if critical services are down, otherwise returns JSON status "healthy".

Error Handling

Global exception handlers are environment-aware to prevent information leakage.

- **Production:** Returns generic "Internal server error" messages.
- **Development:** Returns detailed stack traces and error descriptions.

Database & Configuration

Database Security

- **Pooling:** Uses `QueuePool` with a base size of 10 and a max overflow of 20 connections.
- **Timeouts:** Enforces connection timeouts (10s) and query statement timeouts (30s).
- **SSL:** SSL mode is set to "prefer" in production.

Configuration Validation The application validates environment settings on startup.

- **Secret Key:** Must be at least 32 characters in production.
- **Database URL:** Warnings are issued if default credentials are detected in production.

Automated Backups A script (`scripts/backup_database.sh`) handles data persistence.

- **Policy:** 30-day retention period.
- **Format:** Compressed gzip SQL dumps.
- **Usage:** Can be scheduled via cron (e.g., `0 2 * * *`).

Security Control Mapping Overview

Implemented Control	**Framework Reference**
Rate limiting (SlowAPI)	OWASP A07 – Identification & Authentication Failures
Input validation (Pydantic)	OWASP A03 – Injection
CORS whitelisting	OWASP A05 – Security Misconfiguration
Firewall DB isolation	CIS Control 12 – Network Infrastructure Management
Falco IDS monitoring	CIS Control 13 – Network Monitoring & Defense
Structured logging	CIS Control 8 – Audit Log Management
Backup & retention policy	NIST CSF – Recover
Fail2Ban SSH protection	CIS Control 6 – Access Control Management

Cost Analysis (in euros)

Cost of a human engineer: ~100k yearly (salary, benefits, taxes, ...)

Availability of a human engineer: 38-40h a week, requires months of training, holidays, can serve limited clients at a time

Cost of an AI:

- One-time for POC infrastructure (capex): ~60k

    - AI server: ~55k
        - 5 NVIDIA Tesla V100 GPUs ~45k (note: this pricing reflects the replacement value of the current POC specs. For a new purchase, we suggest swapping the V100s with more modern GPUs)
        - ~500GB DDR4 RAM ~5k
        - Chassis + motherboard ~2-3k
    - Application server: ~5k (Fujitsu PRIMERGY RX2530 M4)
        - Has around 256GB of DDR4 RAM for near-instant in-memory vector retrieval of technical documents for the AI
        - Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz
- Monthly cost (opex): ~500-1000 (electricity, cooling + bandwidth)

Availability of an AI: 24/7, can serve multiple users at a time, except for maintenance downtime/outages

	Human Engineer	**AI System (On-Prem POC)**
Upfront cost	€0	~€60k (one-time)
Yearly cost	~€100k / year (salary, benefits, taxes)	~€6–12k / year (
Availability	38–40h/week	24/7 / 365
Training time	Months	Immediate once deployed (Gets better overtime)
Scalability	1 engineer = 1 workload	One system, many users
Downtime	Holidays, sick leave	Maintenance / outages

Knowledge retention	Lost when employee leaves	Persistent & reusable

Break-even: ~7–8 months vs a single engineer

Application documentation

This document will guide you through the usage of the application. It will help you start the development process but also how to use the application itself. In the first section we will discuss the development process and how you could update the application if there are any updates needed.

Development environment

The application Git repository contains a lot of branches but there are 2 branches which are the most important: Development and Main. When you want to make changes to the application you work on the development branch. You make a branch from the development branch, and you work further on that branch. How could you achieve this? Well, I will explain it below:

The first step should already be done because you will get the repository from us.

1. Make Git repository from ZIP

When you receive our application, you'll get a ZIP file from us. Follow the steps below to make the Git repository:

Step 1: Unzipping ZIP file

Unzip the ZIP file to a folder on your machine.

Step 2: Navigate to your project folder

Open a terminal or command prompt and navigate to your unzipped folder:

cd [name_folder]

Step 3: Making the Git repository

Create a new Git repository:

git init

Step 4: Create initial commit

Add all files to Git and create the first commit:

git add .

git commit -m "Initial commit: Agent TSE Application"

Step 5: Create Development and Main branches

The application works with two main branches: Main and Development. Create these branches:

```
# Main branch already exists (current branch)
git branch -M main

# Create development branch
git branch development
git checkout development
```

2. Git workflow for Development

When you want to make changes to the application, follow this workflow:

Create a feature branch

Always create a new feature branch from the development branch:

```
# Make sure you are on the development branch
git checkout development

# Pull the latest changes
git pull origin development

# Create a new feature branch
git checkout -b feature/new-functionality
```

Making changes

Make your changes in the code and commit them regularly:

```
# Check which files have been modified
git status

# Add specific files
git add frontend/src/components/NewComponent.tsx

# Or add all changes
git add .
```

Create a commit with a clear message

git commit -m "Add: new functionality for email review"

Merging the feature branch

When your feature is ready, merge it with the development branch:

Push your feature branch first

git push origin feature/new-functionality

Switch to development branch

git checkout development

Merge your feature branch

git merge feature/new-functionality

Push the updates to development

git push origin development

Delete the feature branch (local and remote)

git branch -d feature/new-functionality

git push origin --delete feature/new-functionality

Production release (Development → Main)

When you are ready to go to production, merge development with main via a pull request. In the documentation below you'll find a detailed article about how to make a pull request in Github.

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request

3. Useful Git commands

Below you will find an overview of the most used Git commands:

Status and history

- **`git status`** - Check which files have been modified
- **`git log`** - View commit history
- **`git log --oneline`** - Compact view of commits
- **`git diff`** - View differences in files

Branch management

- **`git branch`** - View all branches
- **`git branch name`** - Create new branch
- **`git checkout name`** - Switch to another branch
- **`git checkout -b name`** - Create and switch to new branch
- **`git branch -d name`** - Delete branch locally

Remote synchronization

- **`git pull origin branch`** - Pull changes from remote
- **`git push origin branch`** - Push changes to remote
- **`git fetch`** - Fetch remote updates without merging
- **`git remote -v`** - View remote repositories

Undoing changes

- **`git restore file`** - Restore uncommitted changes
- **`git reset HEAD~1`** - Undo last commit (changes remain)
- **`git reset --hard HEAD~1`** - Remove last commit completely (careful!)
- **`git stash`** - Temporarily save changes
- **`git stash pop`** - Restore temporarily saved changes

4. Best practices

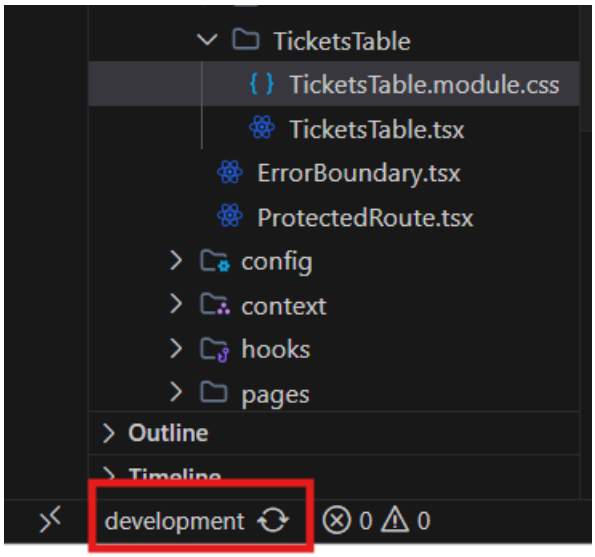
- **Commit regularly** - Make small, logical commits with clear messages
- **Use descriptive branch names** - For example: feature/email-review, fix/database-connection, hotfix/security-patch
- **Always pull before you push** - Prevent merge conflicts by pulling the latest changes first
- **Test before you merge** - Check that your code works before merging it
- **Use .gitignore** - Prevent sensitive files or dependencies from being committed to Git
- **Never work directly on main** - Main branch is for stable, production-ready code
- **Delete feature branches after merge** - Keep your repository clean and organized

References:

- *https://git-scm.com/docs*
- *https://docs.github.com/en/get-started/quickstart/github-flow*
- *https://docs.gitlab.com/ee/topics/git/numerous_undo_possibilities_in_git/ (You'll have to login)*
- *https://www.conventionalcommits.org/*

Usage of the application

When you open the application in your IDE (such as VS Code, Cursor, or Antigravity), you will typically see the current branch displayed in the bottom-left corner:



Once you are on the correct branch, open a terminal inside your IDE or use an external terminal and activate the virtual environment by running the following script:

Path_to_your_project/.venv/Scripts/Activate.ps1

This command should be executed from the root of the project folder. Once the virtual environment is activated, you can start the application by executing these two files:

.\setupLocalDev.ps1

.\runLocalDev.ps1

Note: These commands are designed to work on Windows.

Docker Issues: If you encounter errors related to Docker, ensure that Docker is installed and running. The easiest solution is to download Docker Desktop, which handles the necessary configuration automatically.

Configuration & Environment Variables

The application requires specific environment variables to function correctly. These must be set in .env files.

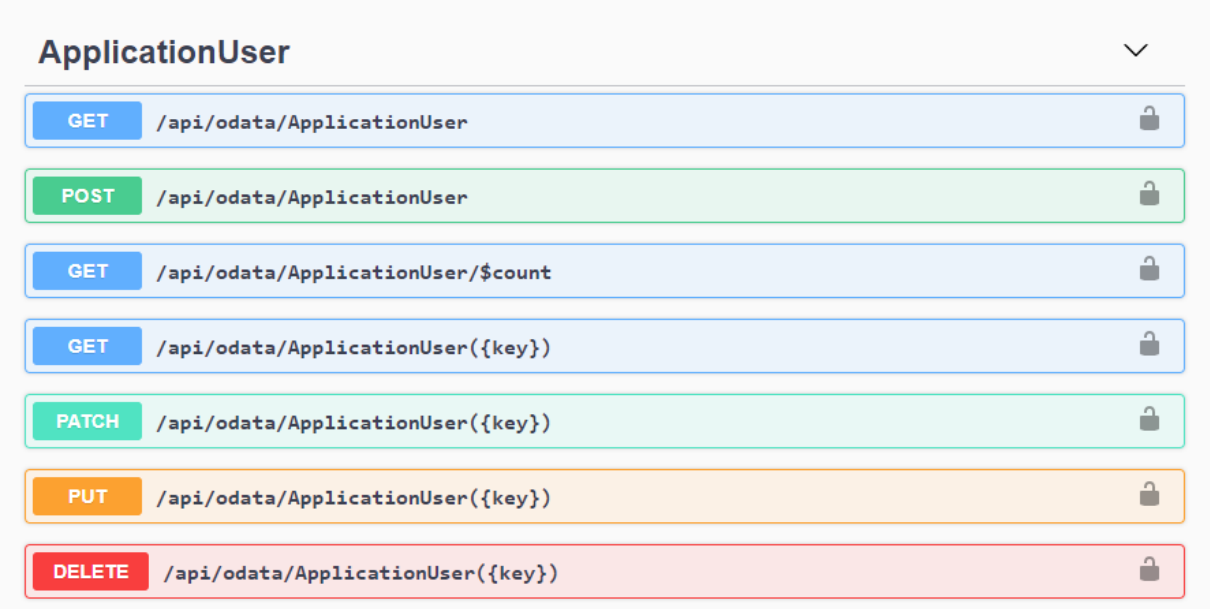
Frontend Configuration

Location:

APP/cercuits-frontend/.env

Variable	Description	Example Value
REACT_APP_API_BASE_URL	The URL where the backend API is reachable. (We use swagger interface	http://localhost:8000/api

	for this, check picture below)	
REACT_APP_ENABLE_CHAT	Enables the chat functionality.	true
REACT_APP_ENABLE_MONITORING	Enables the monitoring dashboard.	true
REACT_APP_ENABLE_KNOWLEDGE_BASE	Enables the knowledge base functionality.	true
REACT_APP_AUTH_TOKEN_KEY	Key for storing the authentication token in local storage.	auth_token
REACT_APP_REFRESH_TOKEN_KEY	Key for storing the refresh token.	refresh_token



Note: The image above is a generic example of the Swagger UI interface. Our application's backend uses Swagger to make API endpoints (such as /api/tickets, /api/users, etc.) visible and testable. Once you have the backend running locally and navigate to the REACT_APP_API_BASE_URL, you will see the specific endpoints for the Agent TSE application.

Backend Configuration

Location:

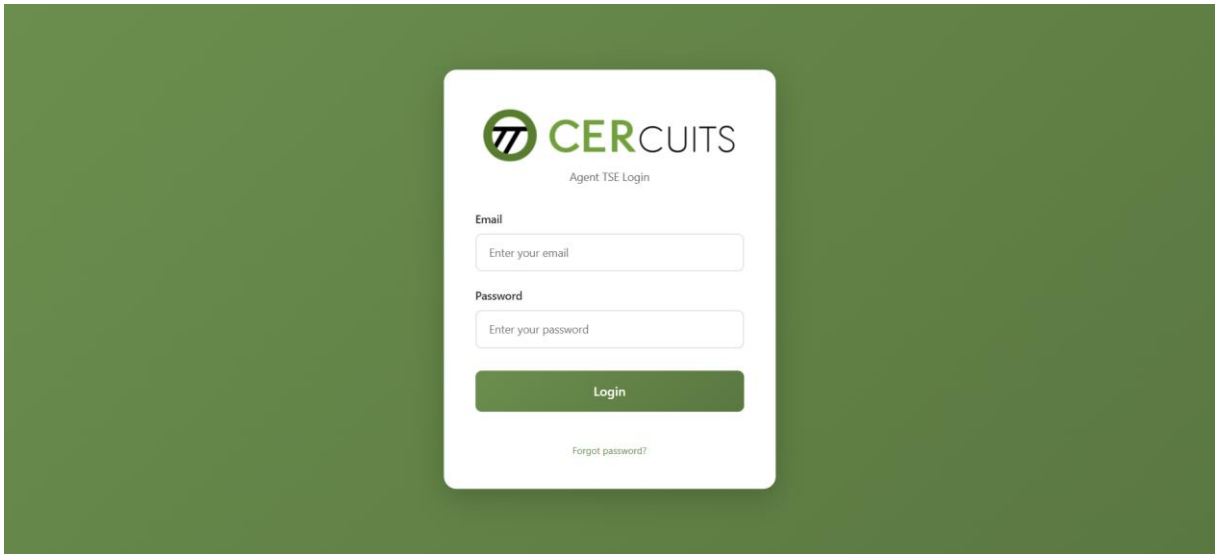
APP/backend/.env

Variable	Description	Example Value
SMTP_HOST	Hostname of the SMTP server (for sending emails).	smtp.gmail.com
SMTP_PORT	Port number of the SMTP server.	587
SMTP_USERNAME	Username for authentication with the SMTP server.	your.email@gmail.com
SMTP_PASSWORD	Password or app-specific password for SMTP.	your-secret-password
OLLAMA_HOST	URL of the Ollama AI service.	http://localhost:11434
SECRET_KEY	Secret key for encryption and session security. **Required in production.**	a-long-and-secure-random-string
DATABASE_URL	Connection string for the PostgreSQL database.	postgresql://user:pass@localhost:5432/db

If the setup completes successfully and the application is running, you will see three command prompt windows appear. This is normal behavior:

- One for the AI server
- One for the backend
- One for the web server

Once the application is running, your default web browser will open automatically and display the login screen:



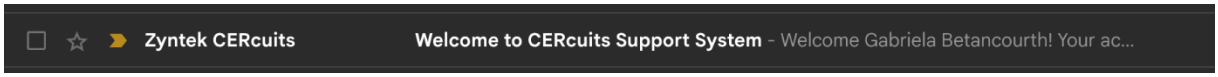
If you see this screen, congratulations! The application is running successfully on your local machine.

If you encounter any issues, please contact our support team at: **zyntek06@gmail.com**

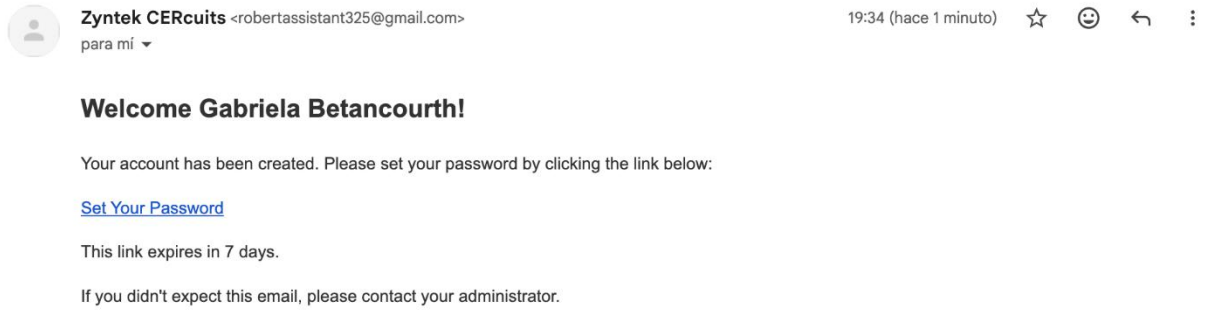
Login

Login: creating a new account

New accounts can only be created through an administrator invitation. An admin must invite a user to join the application (this process is explained later in the document).



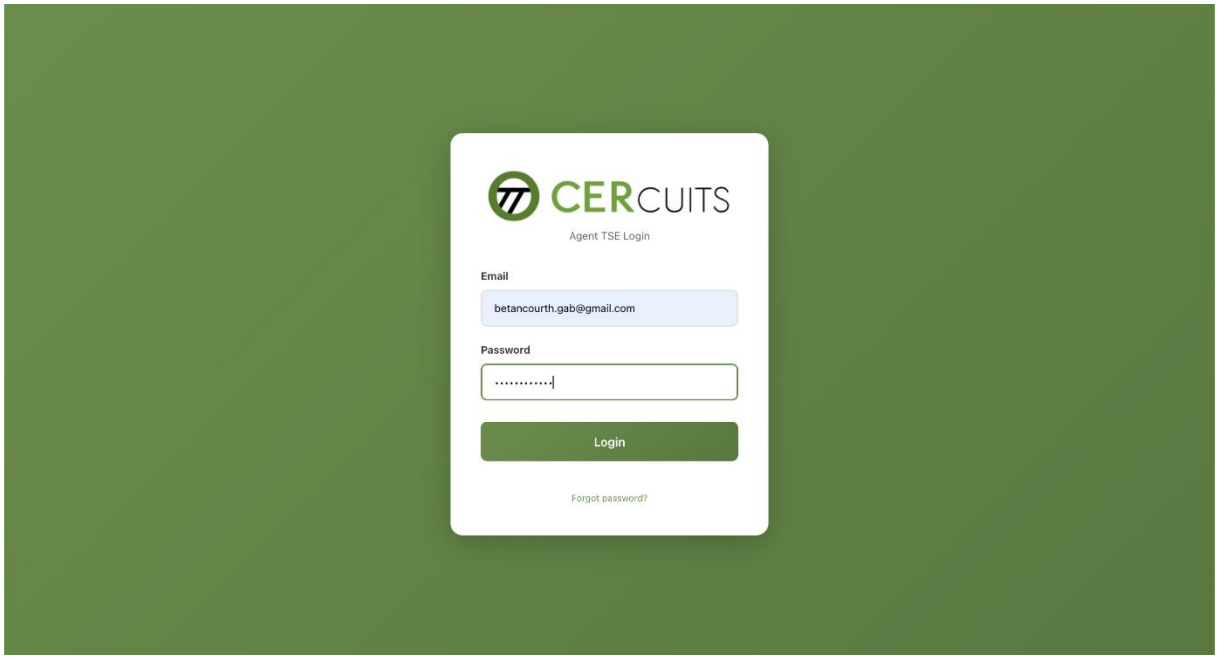
Once invited, you will receive an email containing a link to the application. Through this link, you will be asked to set your password. After setting your password, you can proceed with the normal login process and access the application.



Login: with your account

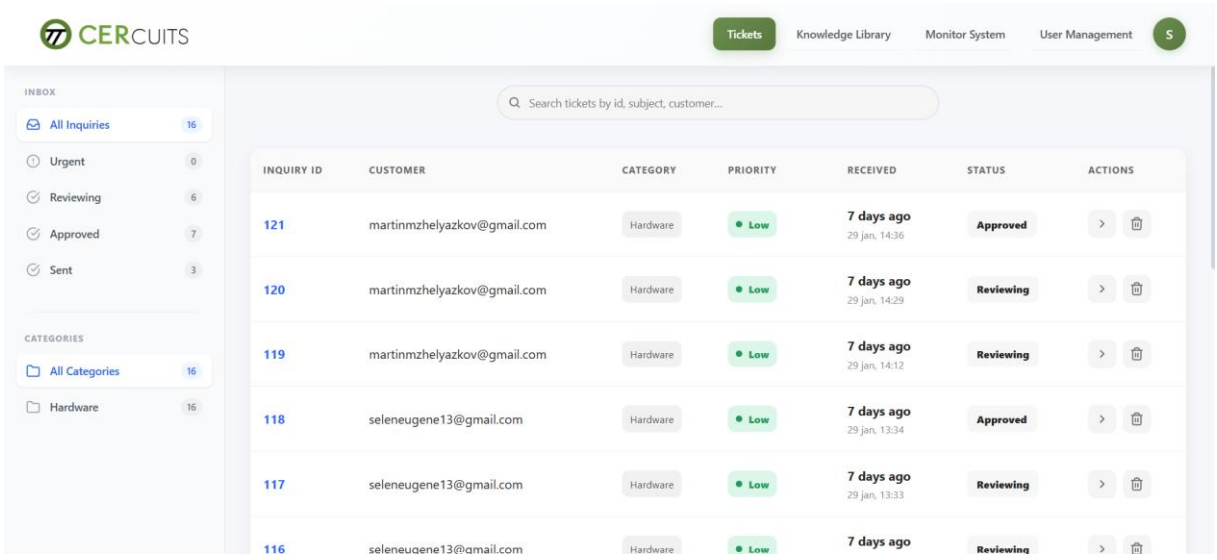
To log in, enter your email address and password in the designated fields and submit the form to access the application.

For security reasons, multiple failed login attempts will temporarily block further attempts. This measure helps prevent unauthorized access.



Application Interface Overview

After logging in with the credentials provided to you, you will see the main dashboard:



The main dashboard is organized into several sections:

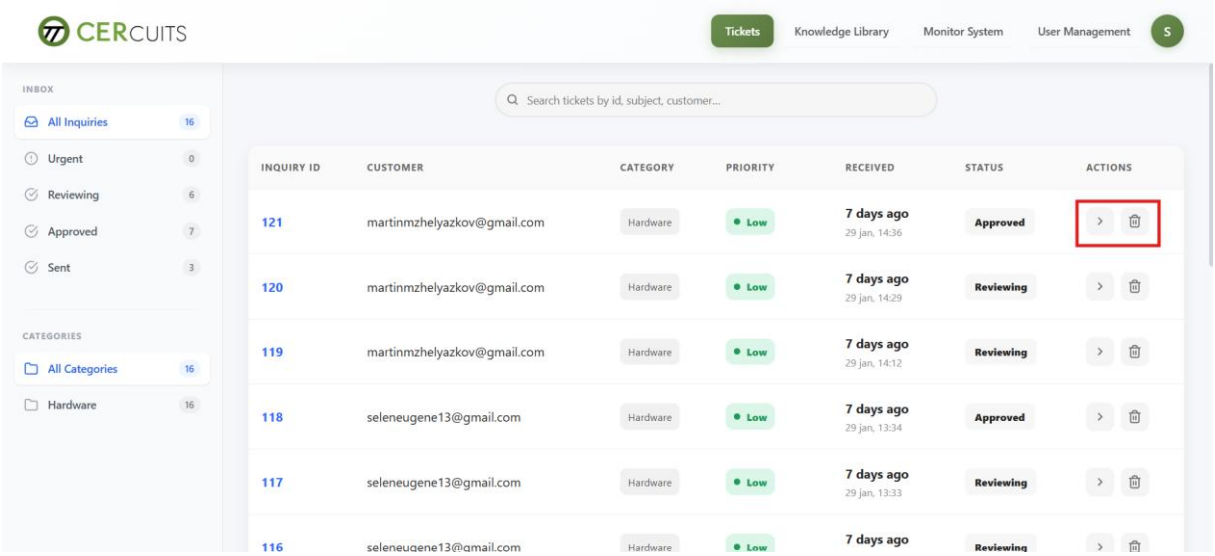
Left Sidebar

The left sidebar displays emails categorized by:

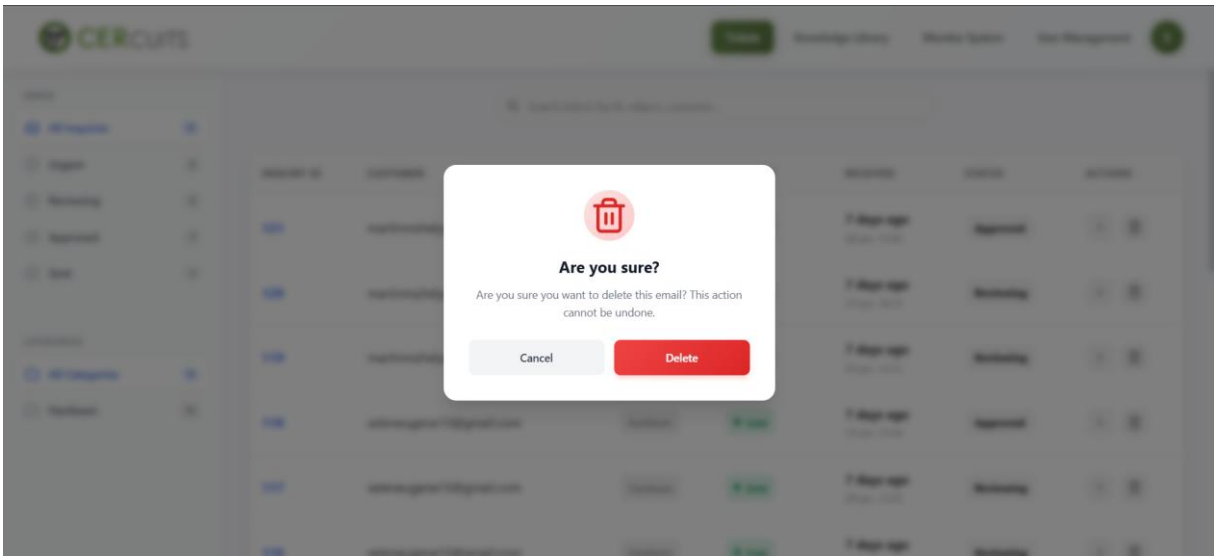
- **Urgency:** Emails marked as urgent will appear in a dedicated "Urgent" category (this category is only visible when urgent emails are present)
- **Status:** Emails are organized by their current status - Reviewing, Approved, or Sent
- **Category:** Emails are automatically categorized by the AI based on their content

Email Action Buttons

Each email in the list has two action buttons:

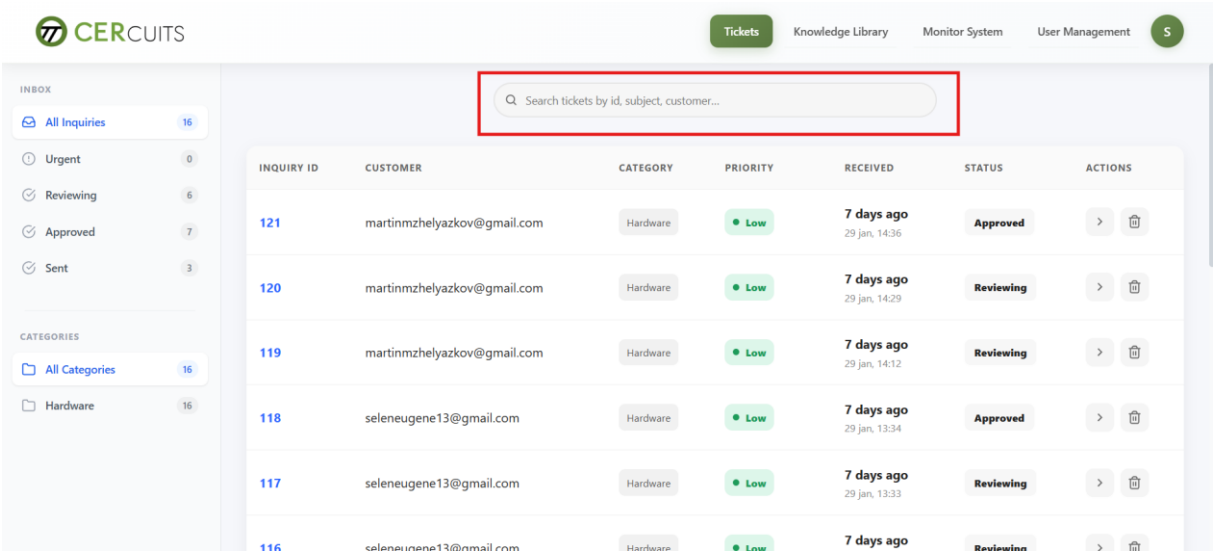


- **Left button (Arrow):** Navigate to the AI draft response screen to generate and review an AI-generated reply
- **Right button (Delete):** Delete the email. A confirmation dialog will appear to prevent accidental deletion



Search Functionality

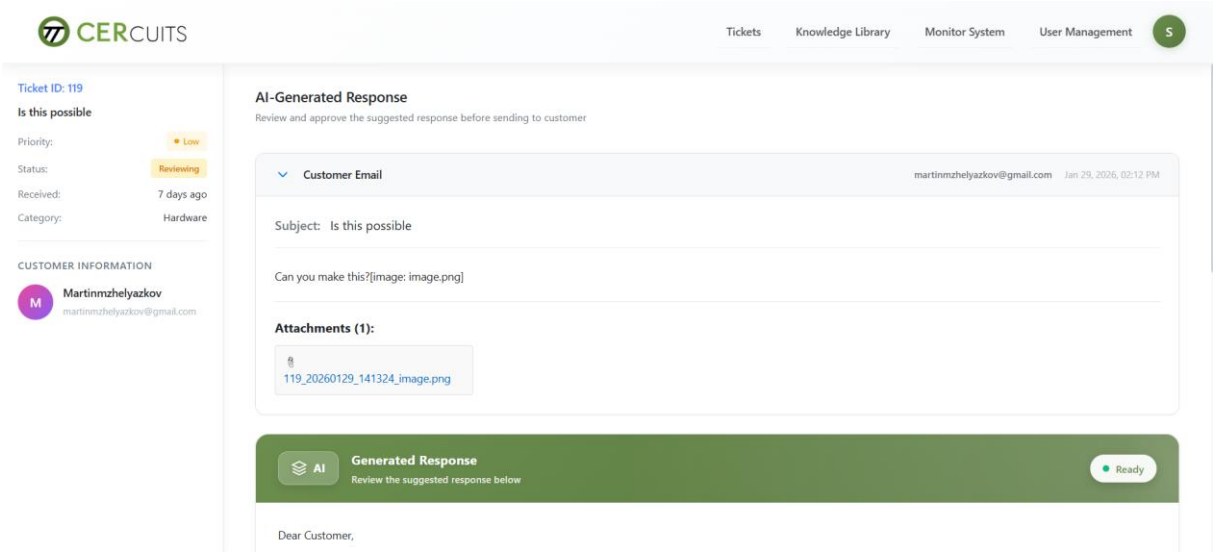
The dashboard includes a search bar that allows you to quickly find specific emails:



Use the search functionality to filter emails by keywords, customer names, or other relevant criteria.

AI Response Generation

When you click the arrow button on an email, you will be taken to the AI-generated response page:



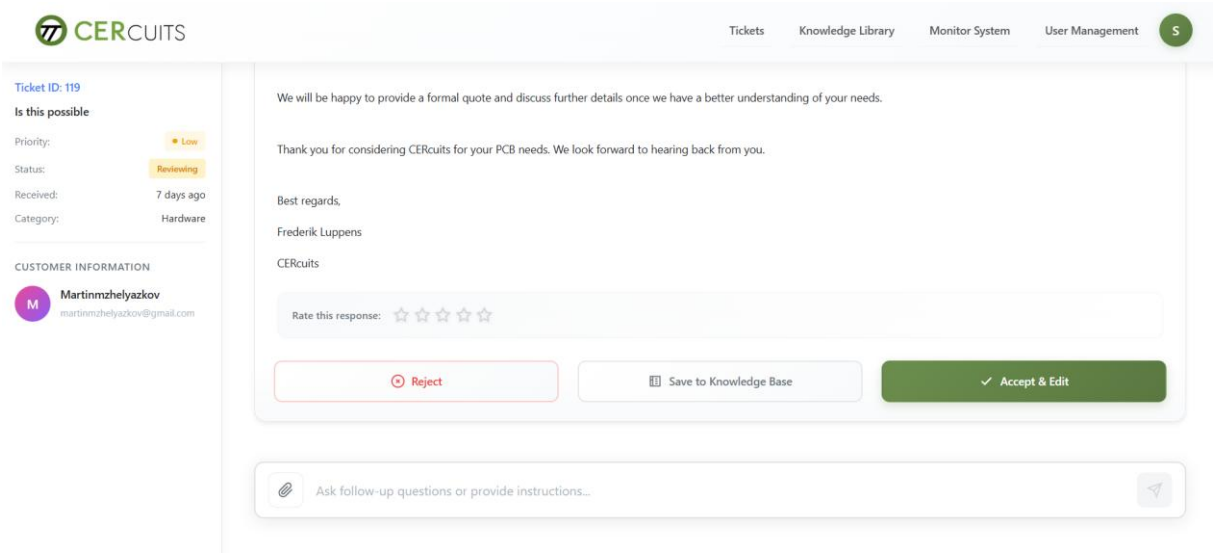
Email Details

The left side of the screen displays:

- Priority level
- Current status
- Date and time received
- Email category
- Customer information
- Email subject
- Email content
- Attachments (clickable to view)

AI-Generated Draft Response

Scrolling down on this page reveals the AI-generated response section:



This section displays the most recently saved version of the AI-generated response. You have several options:

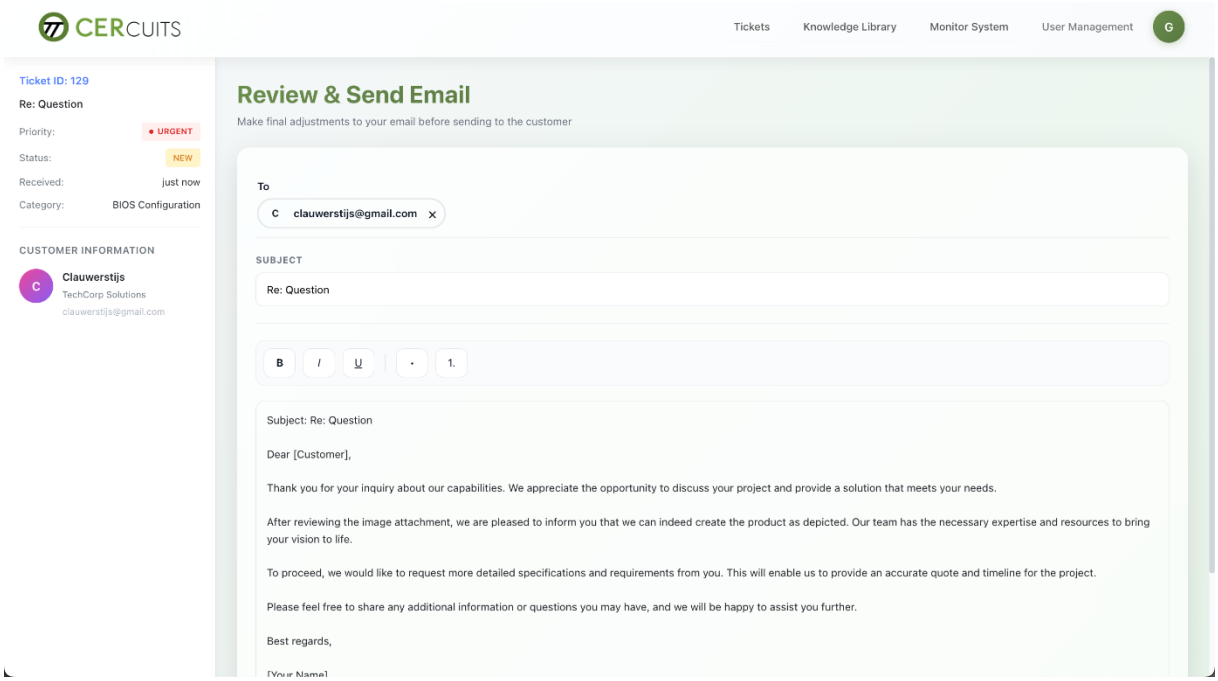
- **Rate the response:** Provide feedback using the star rating system
- **Reject:** Discard the current draft and generate a new response
- **Save to knowledge base:** Store the current draft in the knowledge base for future reference
- **Accept & Edit:** Move forward to edit and finalize the response

Accept & Edit – Email Editor

After approving a draft, you will be redirected to the Accept & Edit page. Here, you can review and manually adjust the email before sending it to the recipient.

You can modify the list of recipients by adding, editing, or removing email addresses. The subject line can also be changed if needed. In the main text editor, you are free to edit the content or add any additional information to the email.

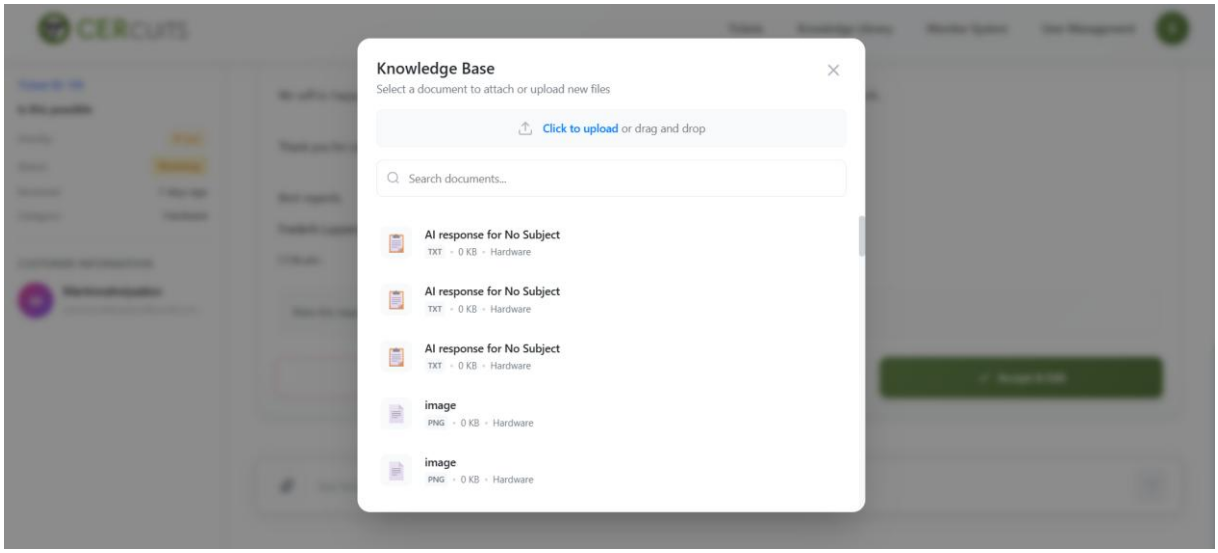
At the bottom of the page, you can choose to either save the draft or send the email. When selecting “Send,” the email will be automatically delivered to all recipients listed.

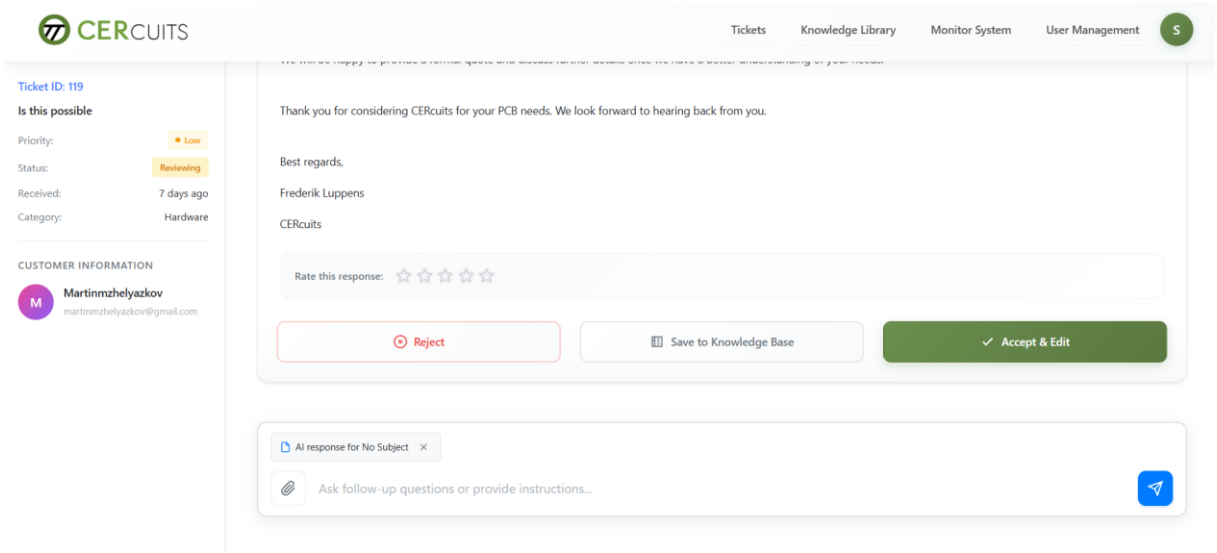

CERCUITS
Tickets
Knowledge Library
Monitor System
User Management
G
Ticket ID: 129
Re: Question
Priority:
URGENT
Status:
NEW
Received:
just now
Category:
BIOS Configuration
CUSTOMER INFORMATION
C
Clauwerstijs
TechCorp Solutions
clauwerstijs@gmail.com
Review & Send Email
Make final adjustments to your email before sending to the customer
To
C
clauwerstijs@gmail.com
SUBJECT
Re: Question
B
I
U
1.
Subject: Re: Question
Dear [Customer],
Thank you for your inquiry about our capabilities. We appreciate the opportunity to discuss your project and provide a solution that meets your needs.
After reviewing the image attachment, we are pleased to inform you that we can indeed create the product as depicted. Our team has the necessary expertise and resources to bring your vision to life.
To proceed, we would like to request more detailed specifications and requirements from you. This will enable us to provide an accurate quote and timeline for the project.
Please feel free to share any additional information or questions you may have, and we will be happy to assist you further.
Best regards,
[Your Name]

Interactive Chat

Below the response, you will find an interactive chat interface where you can:

- Ask follow-up questions to the AI
- Provide additional feedback
- Attach files from your knowledge base or upload new ones to provide additional context





When you attach files from the knowledge base, they will appear in the chat as confirmation that they have been added to the context.

Once you have entered your message or feedback, a blue send button will appear. Click it to submit your input to the AI.

Response Actions

Rejecting a draft: Clicking the "Reject" button will display a confirmation dialog asking if you want to discard the current draft and generate a new response.

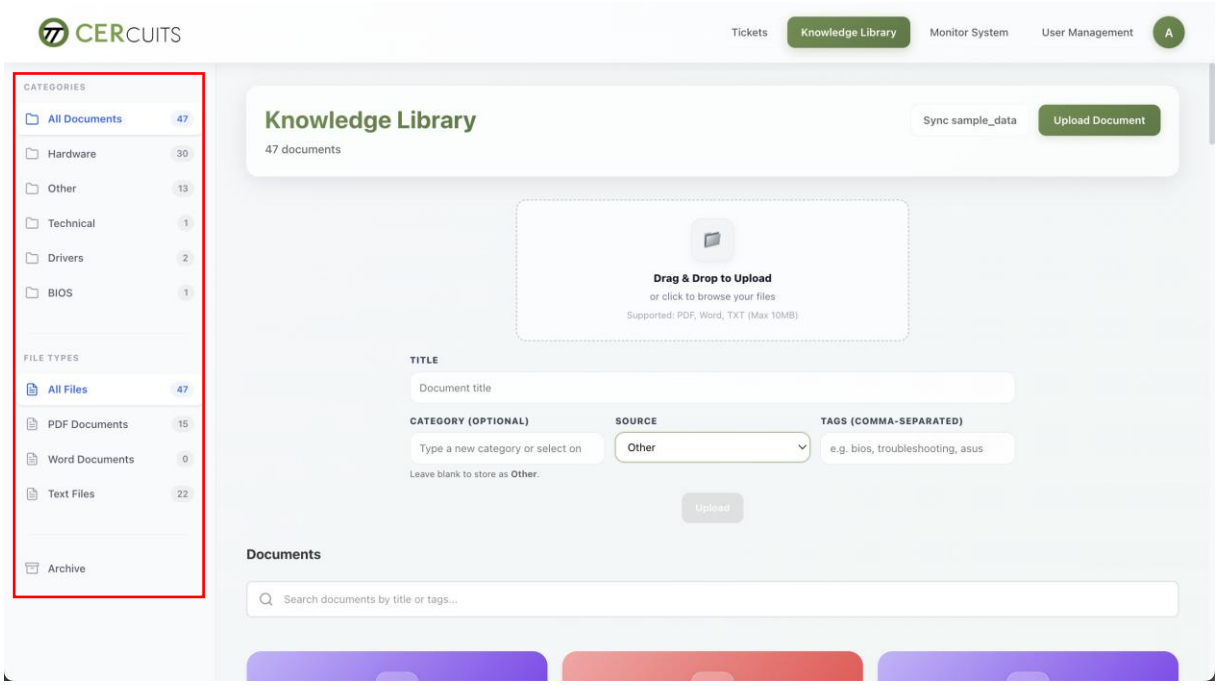
Accepting and editing: Clicking the "Accept & Edit" button will navigate you to a new screen where you can make final edits to the email before sending it to the customer.

This workflow ensures that all AI-generated responses are reviewed and approved by a technical support specialist before being sent to customers, maintaining quality and accuracy in customer communications.

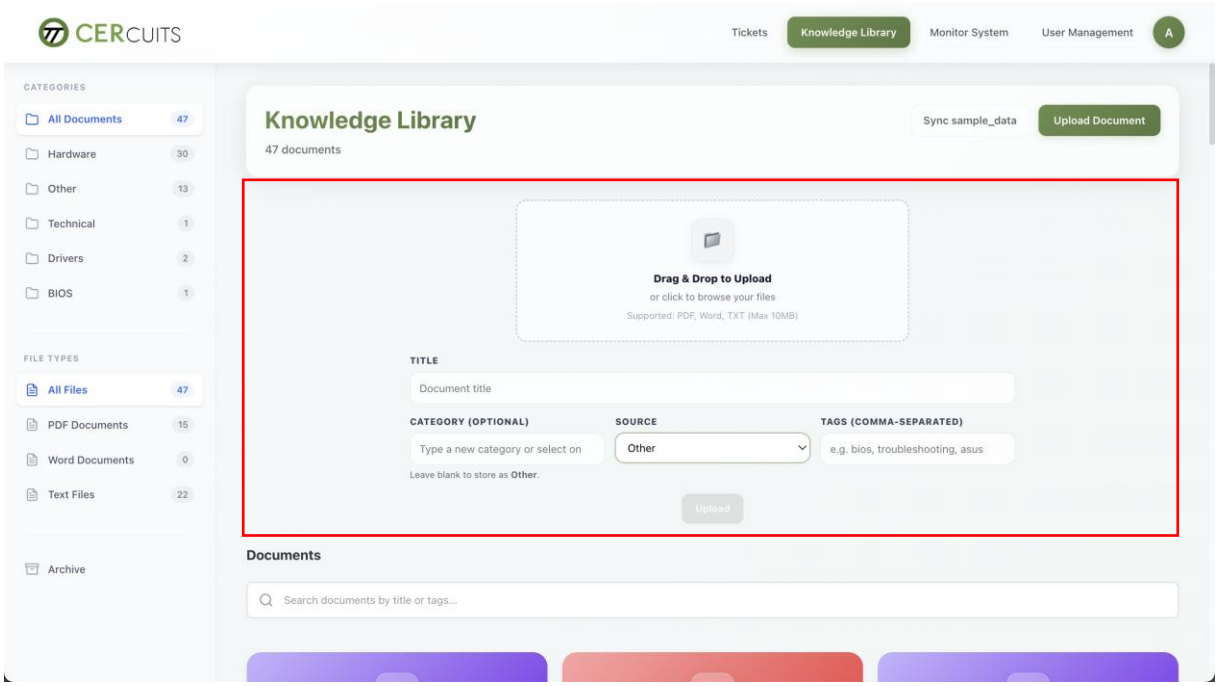
Knowledge Library

Knowledge Library: Overview & Archive

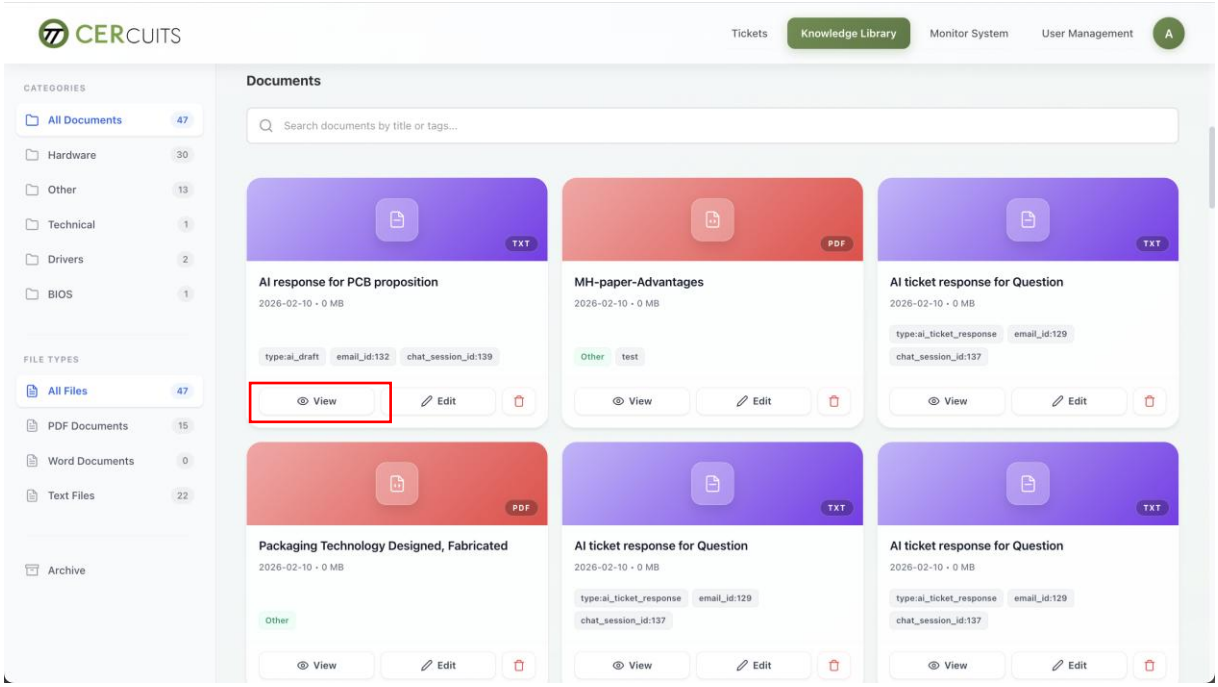
The Knowledge Library contains all documents used as source material for training Agent TSE.



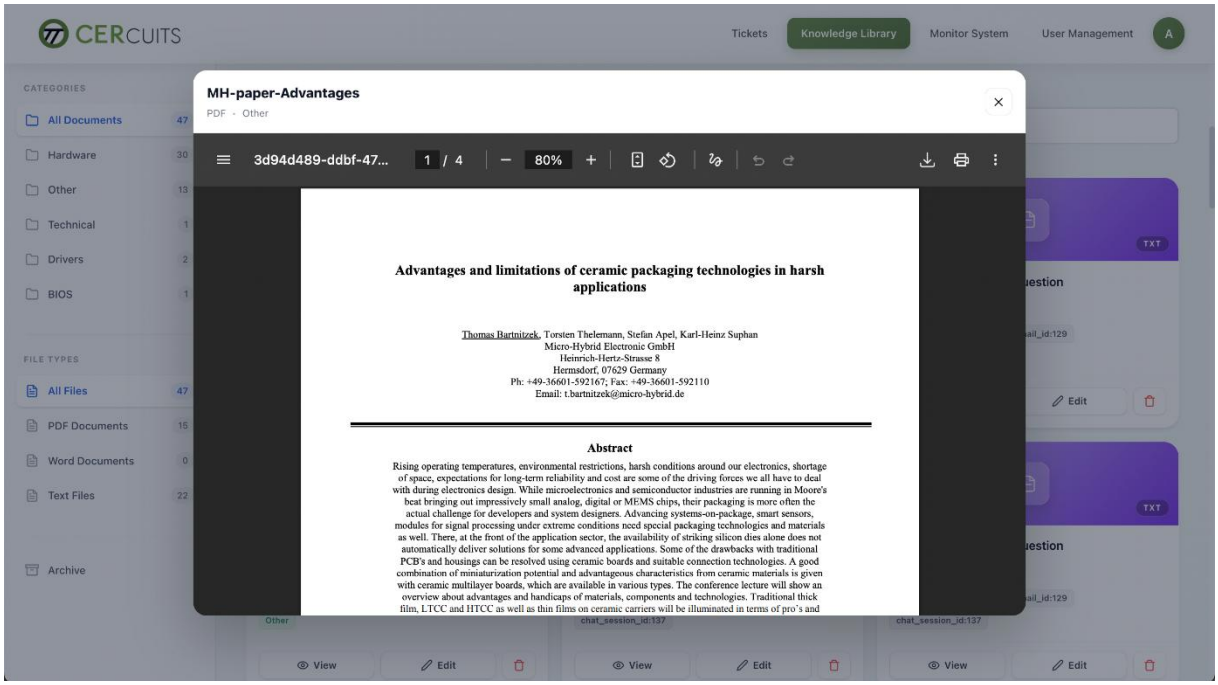
On the left side of the page, you will find a filter panel. This allows you to filter documents by category or file type, and to access the Archive, where deleted documents are stored.



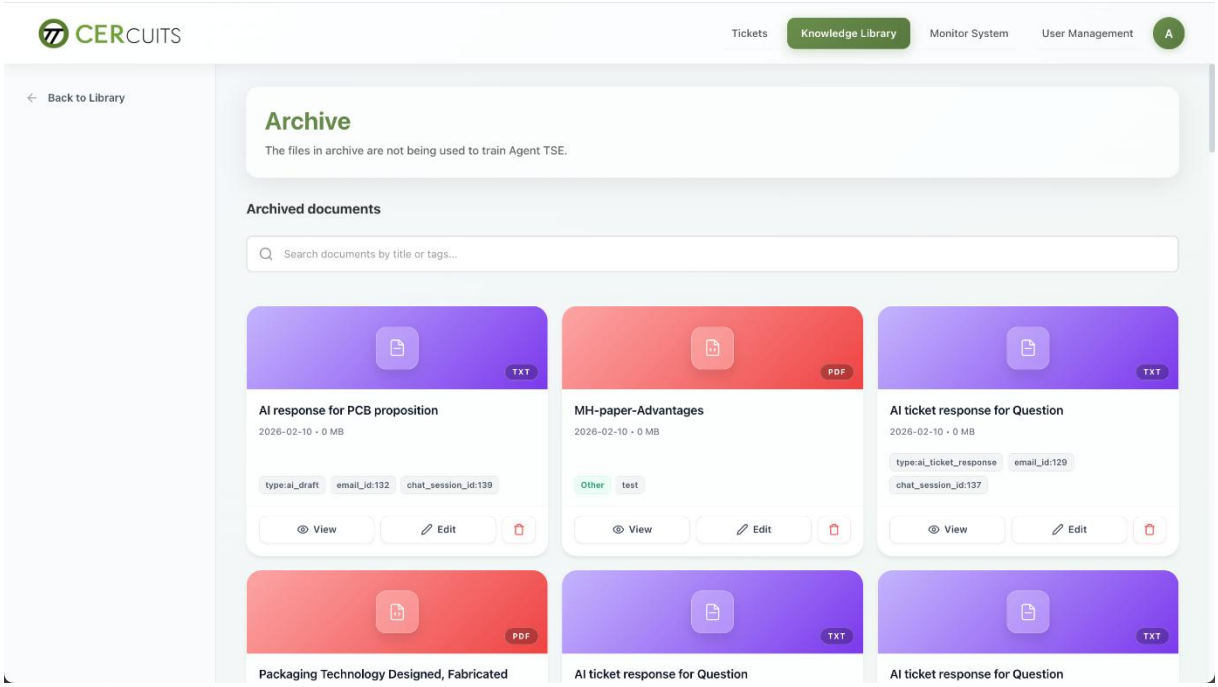
On the right side, the page is divided into two main sections. At the top, there is the Upload Zone, where new documents can be added to the system. Below it, you will see the list of all active documents. These are the documents currently used as training sources for Agent TSE.



Each document can be previewed directly in the application. You can also edit its details, such as the title, category, source, and tags, or delete it if necessary.

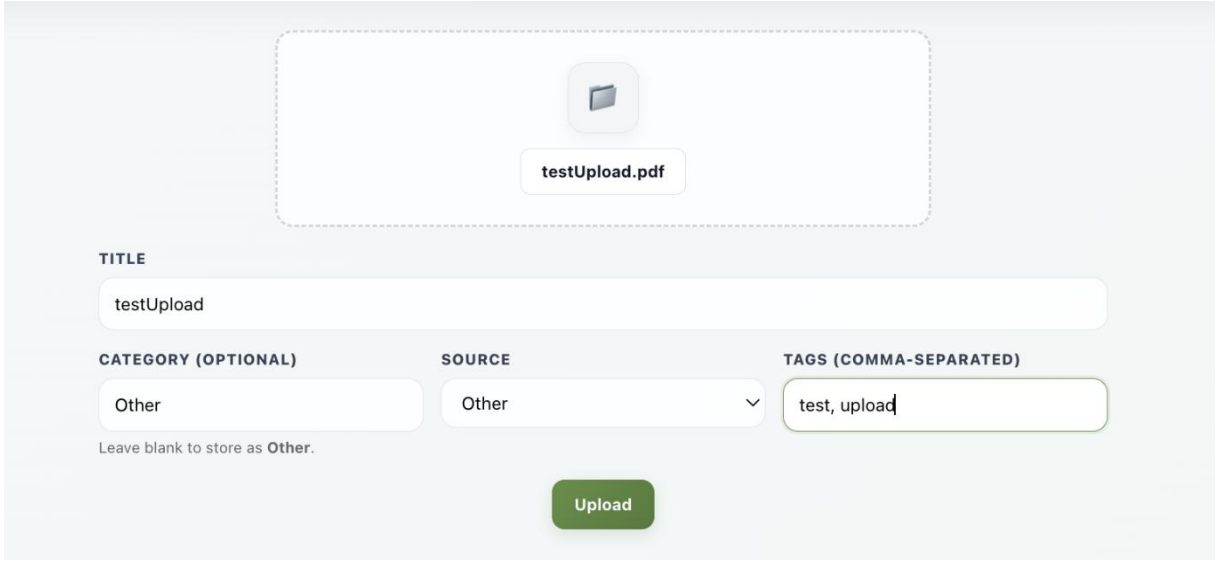


When a document is deleted, it is not permanently removed. Instead, it is moved to the Archive (soft delete). Archived documents are no longer used for training, but they can easily be restored. Once reactivated, they are automatically included again as training sources.

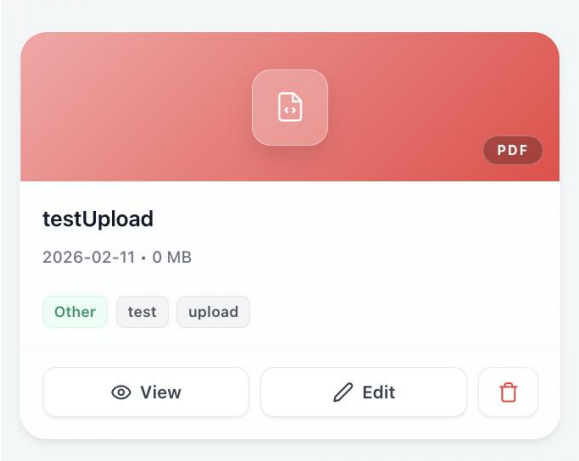


Knowledge Library: Uploading Document

New documents can be added through the Drag and Drop zone. The system supports PDF, Word, TXT, and PNG files.



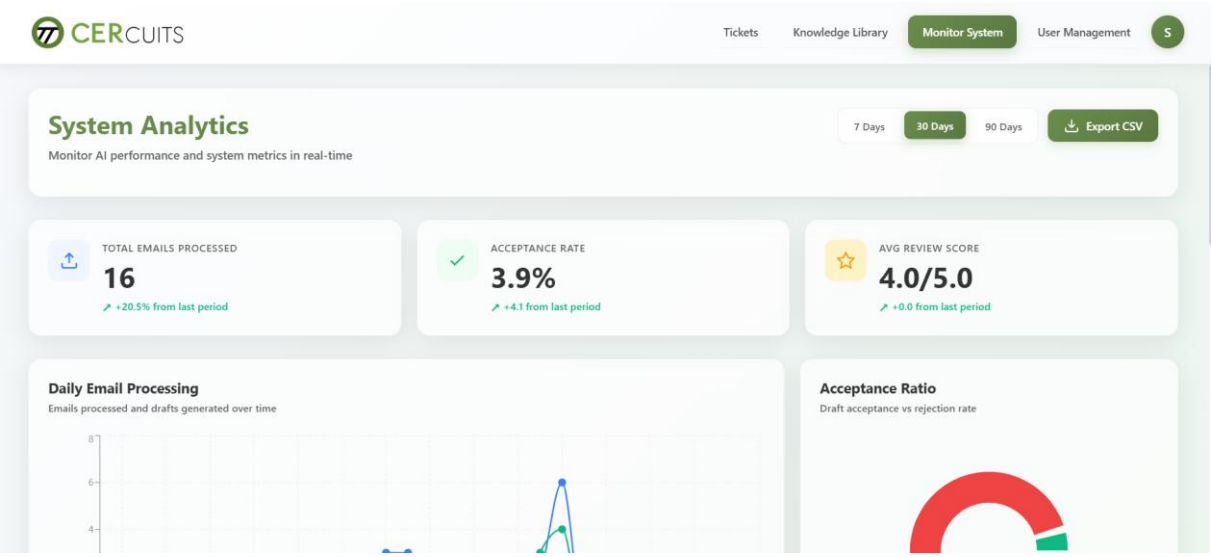
When uploading a document, you must provide a title, select a category (Hardware, Technical, Drivers, BIOS, or Other), indicate the source (such as research papers, emails, or images), and add relevant tags to make future navigation easier.



After saving, the document becomes active and is immediately available as a training source for Agent TSE.

Metrics

In the top navigation menu, you will find the **Monitor System** option. This page provides comprehensive insights into the application's performance and allows you to track how AI-generated drafts are performing over time.



Time Period Selection and Data Export

At the top of the page, you can switch between different time periods to analyze performance metrics:

- 7 days
- 30 days
- 90 days

These metrics can be exported as a CSV file by clicking the **Export CSV** button in the top-right corner, allowing for further analysis by multiple users or integration with external tools.

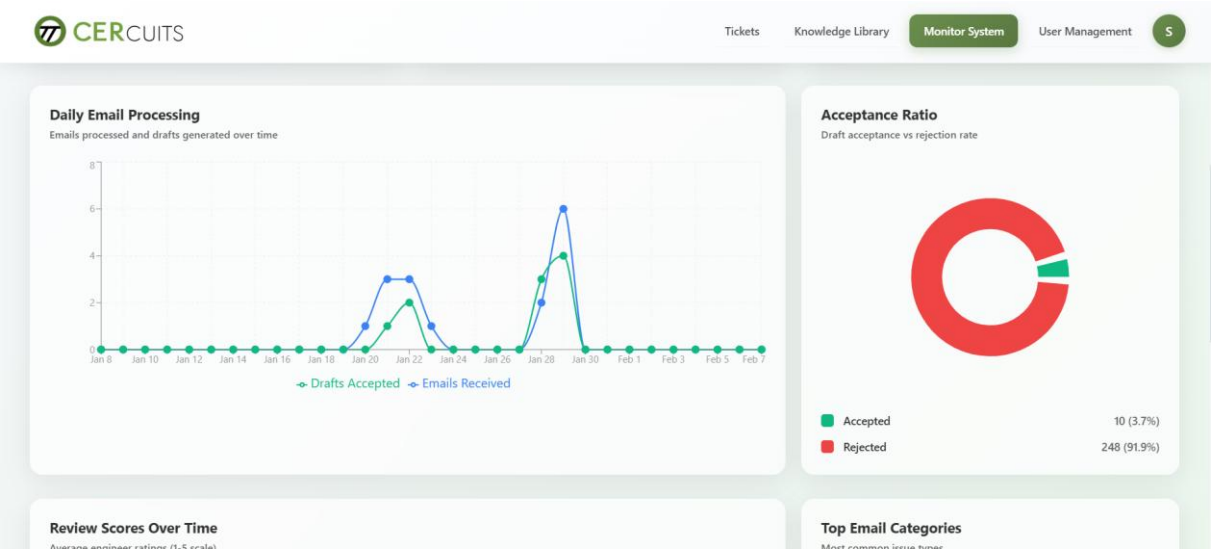
Key Performance Indicators (KPIs)

The page displays three primary KPIs that provide a quick overview of system performance:

- **Total Emails Processed**: Shows the total number of emails processed during the selected time period
- **Acceptance Rate**: Displays the percentage of drafts that were accepted versus rejected by engineers
- **Average Review Score**: Shows the average star rating (out of 5.0) given to AI-generated drafts by reviewers

These metrics help track the overall effectiveness of the AI system and identify trends in draft quality and acceptance.

Performance Visualizations



Daily Email Processing

This line graph displays:

- **Blue line**: Number of emails received over time
- **Green line**: Number of drafts accepted by engineers

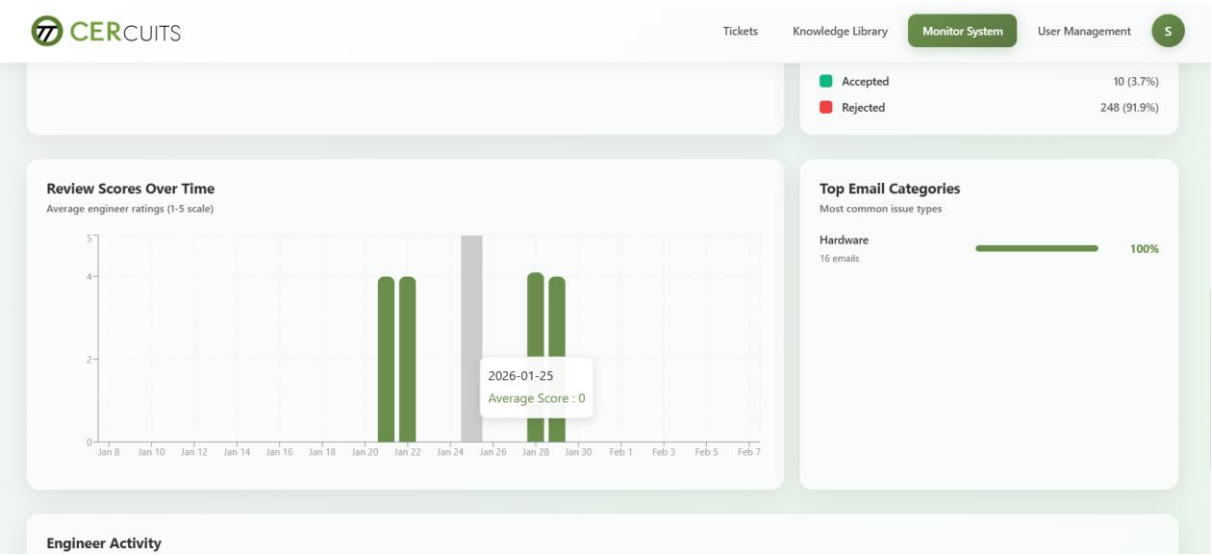
This visualization helps identify patterns in email volume and acceptance trends, making it easier to spot peak processing times and acceptance rate fluctuations.

Acceptance Ratio

The donut chart provides a visual breakdown of:

- **Green segment**: Accepted drafts (with percentage and count)
- **Red segment**: Rejected drafts (with percentage and count)

This chart offers an at-a-glance view of the overall draft acceptance rate for the selected period.



Review Scores Over Time

This bar chart displays the average engineer ratings (on a 1-5 scale) for AI-generated drafts over the selected period. Each bar represents a specific date, with the height indicating the average score given by engineers on that day. Hovering over a bar reveals the exact date and average score.

This helps identify:

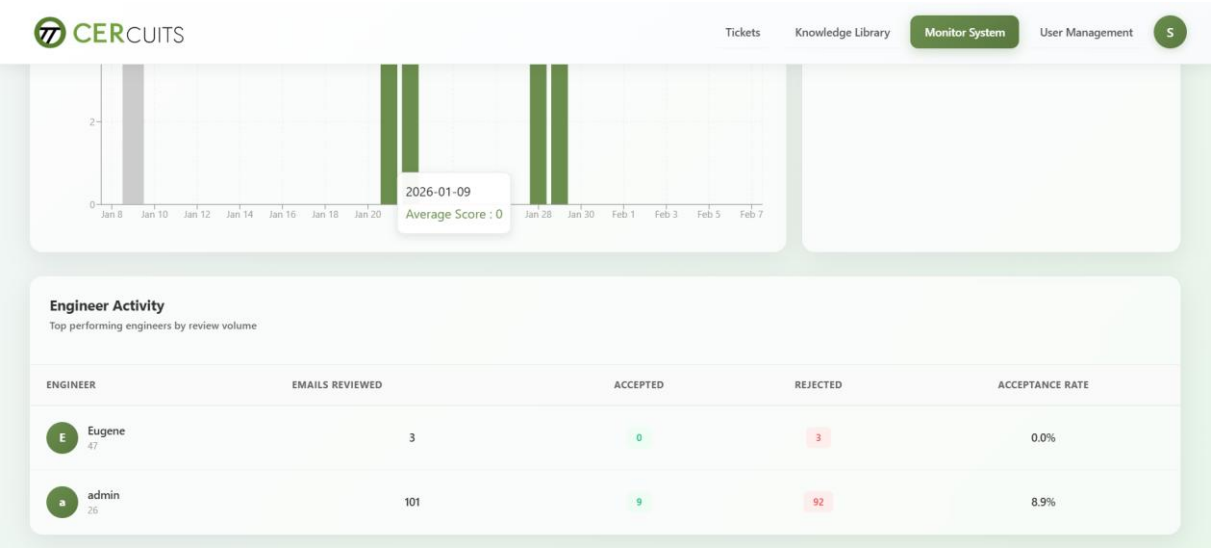
- Trends in draft quality over time
- Days with particularly high or low satisfaction ratings
- Overall consistency of AI-performance

Top Email Categories

This panel shows the distribution of emails across different categories:

- **Category name**: The type of issue (e.g., Hardware, BIOS, Driver)
- **Email count**: Number of emails in each category
- **Percentage**: Proportion of total emails

This breakdown helps understand which types of technical issues are most common and allows for resource allocation planning.



Engineer Activity

The bottom section displays a table showing individual engineer performance metrics:

Column	Description
Engineer	Name and avatar of each technical support engineer
Emails Reviewed	Total number of emails reviewed by the engineer
Accepted	Number of drafts accepted (shown in green)
Rejected	Number of drafts rejected (shown in red)
Acceptance Rate	Percentage of drafts accepted by the engineer

This table helps identify:

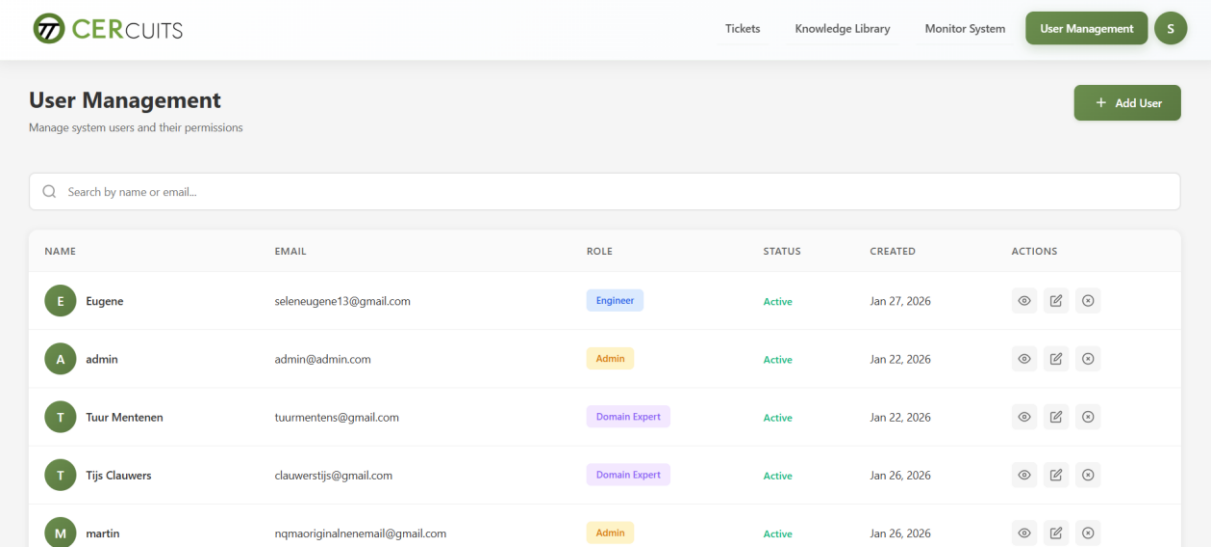
- Top-performing engineers by review volume
- Individual acceptance patterns
- Engineers who may need additional training or support

The data provides valuable insights for team management and helps ensure consistent quality across all technical support responses.

Note: All metrics and visualizations update automatically based on the selected period (7, 30, or 90 days), providing flexible analysis options for different reporting needs.

User Management

In the top navigation menu, you will find the User Management option. This page provides administrators with comprehensive tools to manage system users, including the ability to create, view, edit, deactivate, and delete user accounts.



Overview

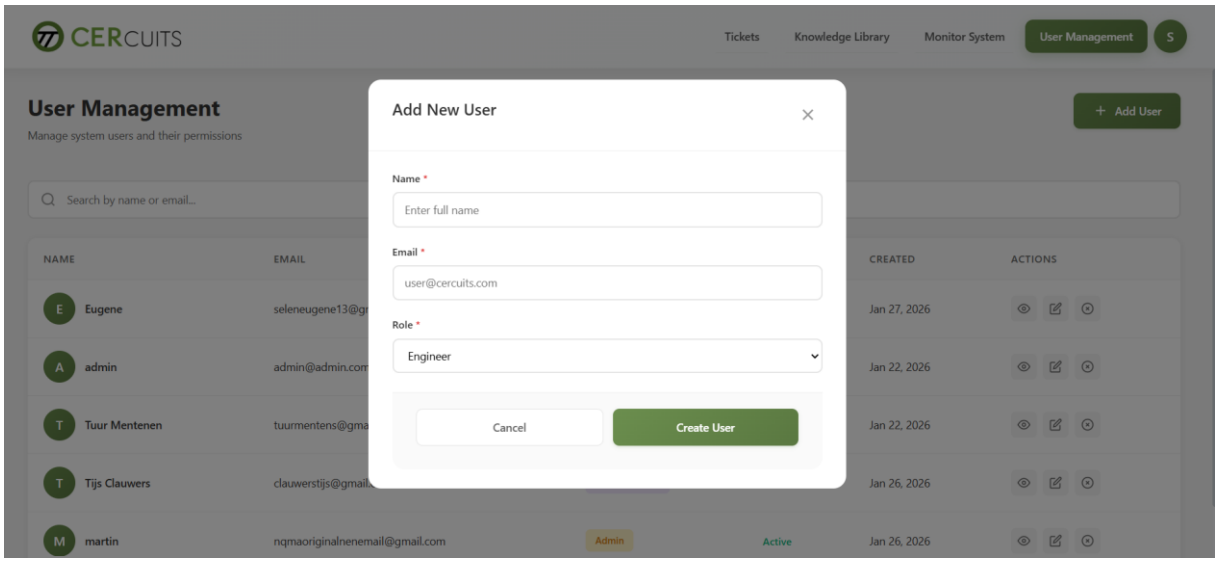
The User Management page displays a searchable table of all system users with the following information:

Column	Description
Name	User's full name with avatar
Email	User's email address
Role	User's permission level (Engineer, Admin, or Domain Expert)
Status	Account status (Active or Inactive)
Created	Date the account was created
Actions	Three action buttons: View, Edit, and Delete

A search bar at the top of the page allows you to quickly find users by name or email address.

Adding a New User

To create a new user account, click the **+ Add User** button in the top-right corner of the page. This will open a modal dialog.



The "Add New User" form requires the following information:

- **Name** (required): Enter the user's full name or preferred display name
- **Email** (required): Enter a valid email address (e.g., user@cercuits.com)
- **Role** (required): Select the appropriate permission level from the dropdown:
    - **Engineer**: Standard technical support role with access to ticket review and draft generation
    - **Admin**: Full system access including user management and system configuration
    - **Domain Expert**: Specialized role with access to knowledge base management

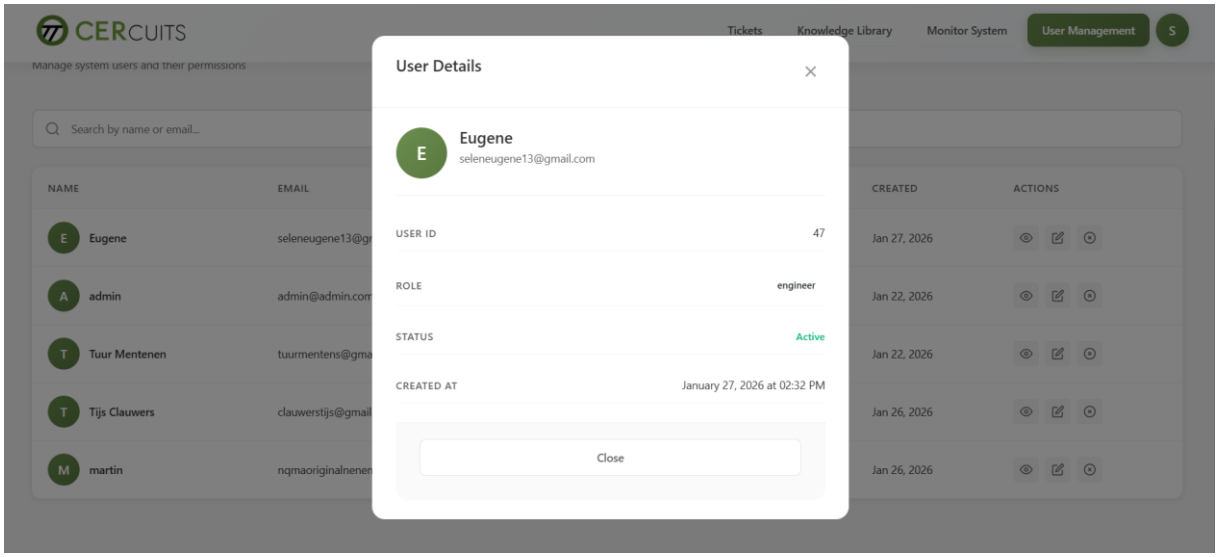
After entering all required information, click the **Create User** button.

Email Notification: Once the user account is created, an automated email will be sent to the provided email address containing instructions to set up their password. The email includes a secure link that directs the user to the login page with a password creation modal. (Refer to the "Login Workflow" section of this document for detailed password setup instructions.)

Click **Cancel** to close the modal without creating a user.

Viewing User Details

To view detailed information about a specific user, click the **eye icon** in the Actions column for that user.



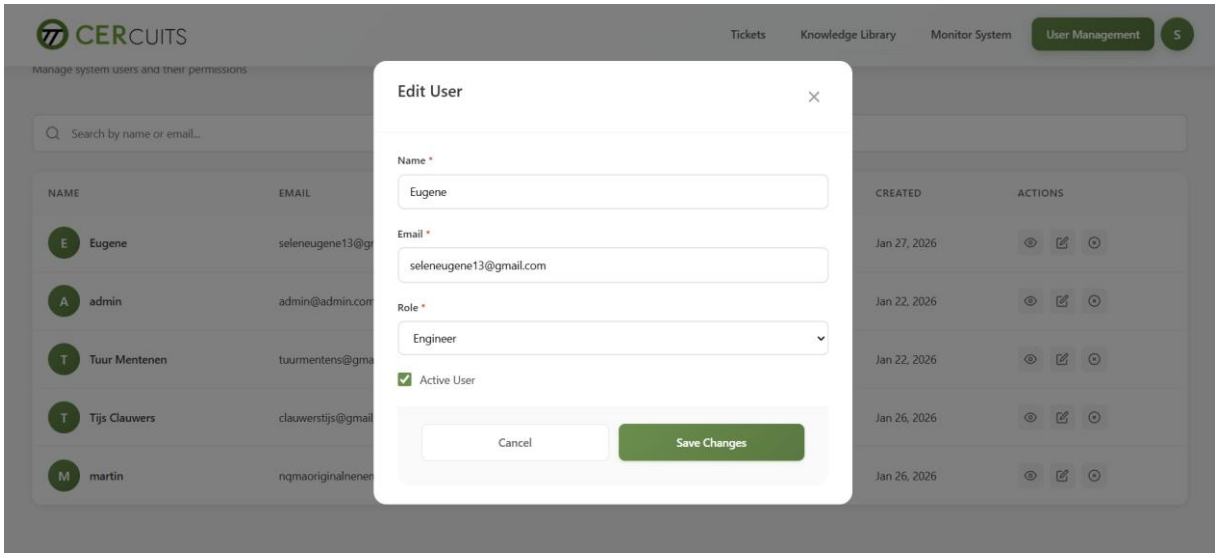
The "User Details" modal displays:

- **User avatar and name**
- **Email address**
- **User ID**: Unique identifier assigned by the system
- **Role**: Current permission level
- **Status**: Account status (Active or Inactive, shown in green or red)
- **Created At**: Exact date and time the account was created

Click the **Close** button to exit the details view.

Editing User Information

To modify a user's account details, click the **pencil icon** in the Actions column for that user.



The "Edit User" form allows you to update:

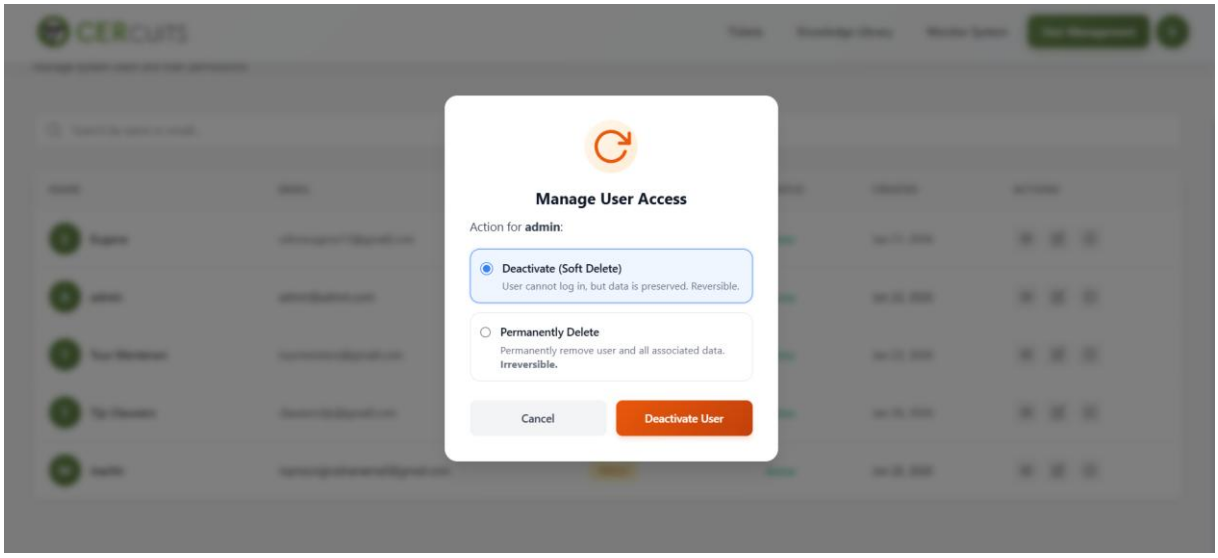
- **Name**: Modify the user's display name
- **Email**: Change the user's email address
- **Role**: Update the user's permission level using the dropdown menu
- **Active User**: Toggle the checkbox to activate or deactivate the account
    - Checked = Active user (can log in)
    - Unchecked = Inactive user (cannot log in)

After making changes, click **Save Changes** to apply the updates. Click **Cancel** to discard changes and close the modal.

Note: Changing a user's email address will trigger a new verification email to be sent to the updated address.

Deleting or Deactivating Users

To remove or deactivate a user account, click the **trash icon** in the Actions column for that user.



The "Manage User Access" modal provides two deletion options:

Option 1: Deactivate (Soft Delete) - Recommended

- **Radio button**: "Deactivate (Soft Delete)"
- **Description**: "User cannot log in, but data is preserved. Recoverable."
- **Effect**:
    - The user will be unable to log in to the system
    - All user data, activity history, and associated records remain in the database
    - The account can be reactivated later by editing the user and checking the "Active User" checkbox
    - This is the recommended option for temporarily removing access or for users who may return

Option 2: Permanently Delete - Use with Caution

- **Radio button**: "Permanently Delete"
- **Description**: "Permanently remove user and all associated data. Irreversible."
- **Effect**:
    - The user account is permanently removed from the database
    - All associated data, including review history and activity logs, will be deleted
    - **This action cannot be undone**
    - Use this option only when you are certain the user account and all related data should be completely removed

After selecting your preferred option, click the **Deactivate User** button (the button text matches your selected action) to proceed. Click **Cancel** to close the modal without making any changes.

Unlocking User Accounts

As a security measure, a user account is automatically locked after a number of failed login attempts (default is 5). An administrator can manually unlock the account.

Steps to unlock:

1. Navigate to the **User Management** page.
2. Find the relevant user in the list.
3. Click the **Unlock** icon (a padlock) in the "Actions" column.
    - *Note: This icon is only visible/active if the user is actually blocked or as a preventive measure.*
4. A confirmation window will appear ("Unlock User Account").
5. Click **Unlock User** to confirm.
6. The system resets the number of failed login attempts to 0, allowing the user to log in immediately with the correct password.

Best Practices for User Management

- **Regular audits**: Periodically review the user list to ensure only authorized personnel have access
- **Role assignment**: Assign users the minimum permission level required for their role
- **Deactivation vs. deletion**: Use deactivation (soft delete) when you want to preserve user history and may need to restore access later
- **Permanent deletion**: Only use permanent deletion when you are certain the user data is no longer needed and you want to comply with data retention policies
- **Email verification**: Always ensure email addresses are correct when creating users, as they are used for password reset and system notifications

Security Note: Only users with the Admin role can access the User Management page. Engineers and Domain Experts do not have permission to manage user accounts.

Future Improvements

Customer Context for AI

To improve response quality and reduce repetitive configuration, we recommend introducing a "customer context" mechanism for the AI/RAG system. This enables the AI to consistently use customer-specific details.

- A detailed technical implementation guide is provided as a companion handover artifact: CUSTOMER_AWARENESS_IMPLEMENTATION.md (attached with this handover package). It includes proposed schema changes, API/service updates, and rollout/testing steps.

Geo-blocking Strategy (Network-level, pre-VPN)

Geo-blocking at the application level is not effective in this setup because users access the system through a VPN; the application will typically see the VPN egress IP rather than the user's real location.

- Recommended approach: If geo-restrictions are required, they should be implemented before VPN access, at the network edge (e.g., firewall/edge gateway) where the client IP geolocation is still visible.

- Current limitation (project context): In the current environment, the VPN endpoint and edge controls are managed externally (school-managed VPN), so implementing pre-VPN geo-blocking was not possible within scope.

- Client-side action: If the client operates their own VPN gateway/firewall in production, geo-blocking can be implemented there (with allow/deny rules per country and exception handling for traveling users).

RAG Retrieval Quality (Hybrid Search)

To improve answer precision for technical support queries, we recommend implementing a hybrid retrieval approach that combines BM25 keyword scoring with semantic vector search. This is especially useful for exact terminology such as part numbers, model codes, and hardware-specific terms that semantic search alone may miss. A weighted scoring strategy can be used to balance lexical matching and semantic relevance, then benchmarked against the current RAG baseline before rollout.