ISEN MÉDITERRANÉE | L'ÉCOLE DES INGÉNIEURS DU NUMÉRIQUE & Innovation 4You

A Generative AI Project

See4Me

By Alicia BOSCH, Loan BELLINI, William INIZIATO, Pierre-Louis LOPEZ, Cyndelle NAPOLETANO & Jassime ZAIETER

Sommaire

Abstract 2
Introduction - State of the art 2
Research 3
Data Engineering 4
ML-Training 5
Quality : Evaluation & Validation 6
Integration 7
Work Package Management 8
- Stakeholders board 8
- Macro-planning (physical progress) & Financial progress 9

Conclusion 10
References 11

Abstract

This project aims to address the difficulties faced by visually impaired students in following courses, particularly when it comes to reading content written on classroom or lecture hall boards. We propose a solution based on artificial intelligence, capable of automatically converting visual information from the board into digital text that can be directly accessed on a computer. Our approach relies on computer vision techniques and optical character recognition (OCR), with the goal of achieving real-time processing. In case of technical constraints, an alternative solution based on images captured by the student is also considered. Ultimately, this project seeks to enhance inclusion, improve learning comfort, and contribute to the academic success of students with visual impairments.

Introduction - State of the art

Access to teachers' notes is a key factor in the academic success of students. All learners should be able to follow lectures clearly and under optimal conditions. According to the World Health Organization, nearly 2.2 billion people worldwide live with some form of visual impairment, a significant proportion of whom are of school or university age. In France, it is estimated that approximately 1 in 100 students is affected by a visual impairment that reduces their ability to read the board or handwritten materials.

Visually impaired students face major challenges when it comes to reading information written on classroom or lecture hall boards. This limitation directly impacts their concentration, comprehension, and overall learning experience. For instance, some studies suggest that students with visual disabilities may lose up to 30% of the visual content transmitted in real time, creating a clear inequality compared to their peers.

Several approaches already exist in an attempt to alleviate these difficulties. Some students, for example, take photos of the board with their phones. However, this method is inconvenient, as it forces them to interrupt their attention during the lecture and often results in poor readability, particularly when the handwriting is unclear. Another option is to rely on classmates' notes, but this deprives the student of immediate access to information and increases their dependence on others. Smart boards also represent a technological alternative, yet their high cost and inability to resolve the core issue of handwritten legibility limit their effectiveness.

In light of these challenges, the objective of our project is to develop a simple, accessible, and efficient system capable of automatically detecting handwritten content on the board and transcribing it into digital text. Such a solution would enable visually impaired students to follow lectures in real time, enhance their concentration and comprehension, and ultimately strengthen both their autonomy and their academic success.

Research

We chose to use TrOCR (base-sized model, fine-tuned on IAM), available on Hugging Face. After evaluating several optical character recognition (OCR) models, we selected TrOCR due to its good balance of accuracy, robustness, and model size, making it particularly suitable for our academic use case. Lighter models often lacked transcription reliability, while larger models were difficult to deploy in real time, making TrOCR an optimal choice.

Technically, TrOCR is based on an encoder–decoder architecture: the encoder is an image Transformer initialized with BEiT weights, while the decoder is a text Transformer initialized with RoBERTa weights. Images are split into fixed 16×16 pixel patches, linearly encoded, and enriched with absolute position embeddings before being processed by the encoder layers. The decoder then autoregressively generates tokens to produce the corresponding text.

We chose the base-sized version of the model, which contains approximately 333 million parameters in F32 precision. This intermediate size provides high performance for both handwritten and printed text recognition, while remaining compatible with our computational constraints. This choice aligns with our goal of developing a reliable, accurate system that can be used in real time by the student concerned.

To transform handwritten or scanned images into clear and fluent English text, we carefully combined models chosen for their complementary strengths. For segmentation, we relied on DocTR, which provides an effective balance between accuracy and efficiency in detecting text regions within complex layouts.

Once these regions were isolated, we used Microsoft's TrOCR to transcribe lines. For the post-editing stage, we selected Flan-T5, which excels at following rewriting prompts and producing coherent, grammatically correct text. We experimented with both the large and base variants, the former delivering higher quality outputs and the latter being lighter for prototyping or when GPU memory was limited.

Importantly, we fine-tuned Flan-T5 on a large dataset of noisy-to-clean OCR pairs, enabling it to specialize in post-editing: correcting spelling and punctuation, reordering fragments, and restoring coherence. We also tested alternative models such as BART and LLaMA, but they did not yield convincing improvements compared to Flan-T5, either because of weaker alignment with the rewriting instruction or instability on noisy OCR data.

Finally, we selected the TinyLlama model for the reformulation stage as a lightweight yet capable option specialized in conversational clarity. Since our fine-tuned model did not achieve the expected results, we turned to this alternative. Its efficient architecture runs smoothly on consumer-grade hardware such as an M1 Pro while still producing coherent, natural English. This makes it well suited to transform rough, line-based text into structured lessons. By using a chat-oriented LLM instead of a purely seq2seq model, the system delivers not only grammatical correction but also clear explanations and well-formed titles, directly supporting the educational needs of visually impaired students.

Data Engineering

A crucial pillar of our project has been the data engineering workflow, which ensures that the language model is exposed to realistic, noisy OCR inputs and their clean, human-readable counterparts. This preparation directly supports the fine-tuning stage that allows the model to transform raw OCR output into coherent English text.

The process is centered on the script prepare_pleias.py, designed to combine external public datasets with our own collected material. We leveraged the PleIAs/Post-OCR-Correction dataset, which provides pairs of "noisy" OCR text and corrected ground truth. From this base, we applied controlled transformations to simulate the kind of noise that appears in handwritten or low-quality scans. Specifically, the script introduced artificial line breaks and slight shuffling of line order, thereby mimicking fragmented or misordered OCR output. Each synthetic example was paired with its corresponding clean target paragraph. This step was critical because it exposed the model to exactly the type of correction task it would later face in practice.

To build a robust training corpus, the PleIAs examples were merged with our existing .jsonl datasets (train.jsonl and val.jsonl). The result was two consolidated files: train_merged.jsonl and val_merged.jsonl, which became the canonical training and validation sets. The merged training file, in particular, is substantial, around 150 MB, and represents thousands of instruction-input-output triples. Each entry follows a consistent schema:

- **instruction**: a fixed rewriting instruction (e.g., "Rewrite this noisy OCR text into one clean, well-ordered paragraph. Fix casing, punctuation, …").
- **input**: the noisy OCR-like text, often split or shuffled.
- **output**: the corrected and fluent English paragraph.

This design ensured that during fine-tuning the model always encountered data formatted in the same prompt style it would later receive at inference time.

By carefully curating and expanding the dataset, we achieved two objectives: (1) expose the model to a wide variety of OCR errors, making it resilient to real-world noise, and (2) maintain alignment between training and deployment prompts, reducing the risk of mismatched behavior. Ultimately, train_merged.jsonl and val_merged.jsonl served as the backbone of the fine-tuning process in finetune_flan_t5.py, enabling us to adapt a general-purpose sequence-to-sequence model (Flan-T5) into a specialized OCR post-editor that can reorder, clean, and polish raw text into human-readable lessons.

ML-Training

After preparing a rich dataset in the data engineering phase, the next step was to train a language model specialized in OCR post-editing. The goal was to move beyond raw spelling correction and enable the model to reorder lines, fix grammar, and produce fluent English while preserving meaning. To achieve this, we fine-tuned a Flan-T5 sequence-to-sequence model, chosen for its strong instruction-following skills.

The training process is implemented in finetune_flan_t5.py. It loads the merged datasets (train_merged.jsonl and val_merged.jsonl), each containing an instruction, noisy OCR-like input, and clean output. To ensure consistency, each example is reformatted into a canonical prompt combining the instruction with the input text. This prompt acts as the source, while the corrected version is the target. The tokenizer from Flan-T5 encodes both, with limits of 1024 tokens for inputs and 512 tokens for targets, balancing expressiveness with efficiency.

We initialized training from google/flan-t5-base, a model already equipped with general reasoning skills. Fine-tuning adapted it specifically to OCR correction. Key training settings included:

- Learning rate 2e-4 for controlled adaptation
- Batch size of 2 per device with gradient accumulation to simulate larger batches
- Two epochs to cover the dataset without overfitting
- Regular evaluation on validation data
- Checkpoint saving with retention limits

The Hugging Face Trainer API handled batching, optimization, and evaluation, while hardware acceleration (Apple Silicon MPS or GPU) ensured feasible runtime on local machines.

At the end, the fine-tuned weights and tokenizer were saved into outputs/flan5-rewriter-base. The result is no longer a generic Flan-T5, but a domain-specialized rewriter trained to convert fragmented OCR text into fluent, grammatical English.

This training step is the bridge between noisy OCR recognition and human-readable educational material. By adapting a large pre-trained model to our dataset, we transformed it into the backbone of the OCR pipeline, capable of polishing raw text into coherent lessons.

Finally, we uploaded our fine-tuned model to Hugging Face (cf. References) to make our project easier to share and lighter to run.

Quality : Evaluation & Validation

The evaluation of our system followed a progressive, iterative approach. In the earliest testing phases, the raw OCR model was able to identify text only partially, often recognizing isolated characters or a single line. This first stage provided outputs that lacked coherence, with many errors and little overall meaning.

Through successive improvements, the system began producing longer sequences of text. However, at this stage the recognition was still fragmentary: some characters were missing, substitutions were frequent, and the reconstructed lines rarely formed a logical sentence. The results were technically faithful to what the OCR captured but not usable in practice for understanding course material.

To address this limitation, we introduced a post-editing component designed to transform noisy, incoherent outputs into structured and intelligible paragraphs. This function was trained to reassemble broken fragments, correct spelling and grammar, and infer a logical flow of ideas. While this approach significantly improved readability, it sometimes produced approximations that altered specific terms. In other words, the corrected text was easier to understand but not always perfectly faithful to the exact original content.

The final version of the model therefore delivers two complementary outputs:

- A **raw output**, which preserves the recognition as it is captured, even if it lacks meaning.
- A **translated output**, which reformulates the same content into a coherent and readable paragraph, at the cost of occasionally replacing technical words with approximate equivalents.

This dual-output design allows users to choose between fidelity and readability depending on their needs. For visually impaired students, this flexibility ensures both access to the precise transcription when required, and a more comprehensible reformulation for general study purposes. The evaluation process thus confirmed the robustness of our solution while also highlighting the importance of user choice in balancing accuracy and accessibility.

Overall, the system achieved a success rate of 84.6%, which can be considered promising. By selecting the best output between the two models for each case, 92% of the test images reached a satisfactory readability score (≥3.8/5).
The results demonstrate that the combined approach of segmentation, TrOCR recognition, and fine-tuned language model post-editing significantly improves text readability and preserves meaning. Future improvements could include enlarging the training dataset, fine-tuning on more diverse handwriting styles, and further optimizing error cases.

Integration

The integration phase ensures smooth cooperation of all components and enables deployment on different machines. To make the environment accessible to all group members, we proposed two setups: Docker for maximum reproducibility and isolation, or a lighter Python virtual environment when Docker is not available. Both guarantee consistent installation of libraries such as PyTorch, Hugging Face Transformers, and OCR packages, while remaining adaptable to each workstation.

Once the environment is ready, the integration connects the modules into one automated pipeline. From a handwritten or scanned image, the system first runs segmentation to detect text regions and generate a manifest of crops. These crops are then processed by the TrOCR model, producing raw line transcriptions. Next, reconstructed sentences are generated by the fine-tuned Flan-T5 model, which cleans errors and reorders fragments. Finally, the reformulation step uses TinyLlama/TinyLlama-1.1B-Chat-v1.0 to improve clarity and produce structured lessons in fluent English. At the end, the user obtains a clean text file with an optional title, stored with intermediate outputs in a dedicated folder.

This phase shows how segmentation, OCR recognition, post-editing, and reformulation are no longer isolated tools but a unified workflow that runs with a single command. The loop with training is closed since the fine-tuned model built on engineered data is directly applied in inference, while reformulation guarantees user-friendly output. By aligning training and inference prompts, the pipeline ensures consistent behavior and delivers reliable corrections, making the system practical to use.

End-to-End Workflow

The integration phase connects all parts of the system into a seamless process. From a single command, an input image becomes clean, readable text:

- **Segmentation**: The image is analyzed to detect blocks, lines, or words, cropped, and listed in a manifest.json with the help of DocTR.

- **OCR Recognition**: Each crop is processed with Microsoft TrOCR, producing raw line text saved in ocr_results.json.

- **Sentence Reconstruction**: The noisy text is refined with a fine-tuned Flan-T5 model, correcting grammar, punctuation, and structure. Results are written to reconstructed.json, with text_final.txt and title.txt.

- **Reformulation**: The reconstructed text is passed to TinyLlama/TinyLlama-1.1B-Chat-v1.0, which rewrites it into fluent English lessons with a clear title, ensuring readability.

A main script is available to launch the full application.

Work Package Management

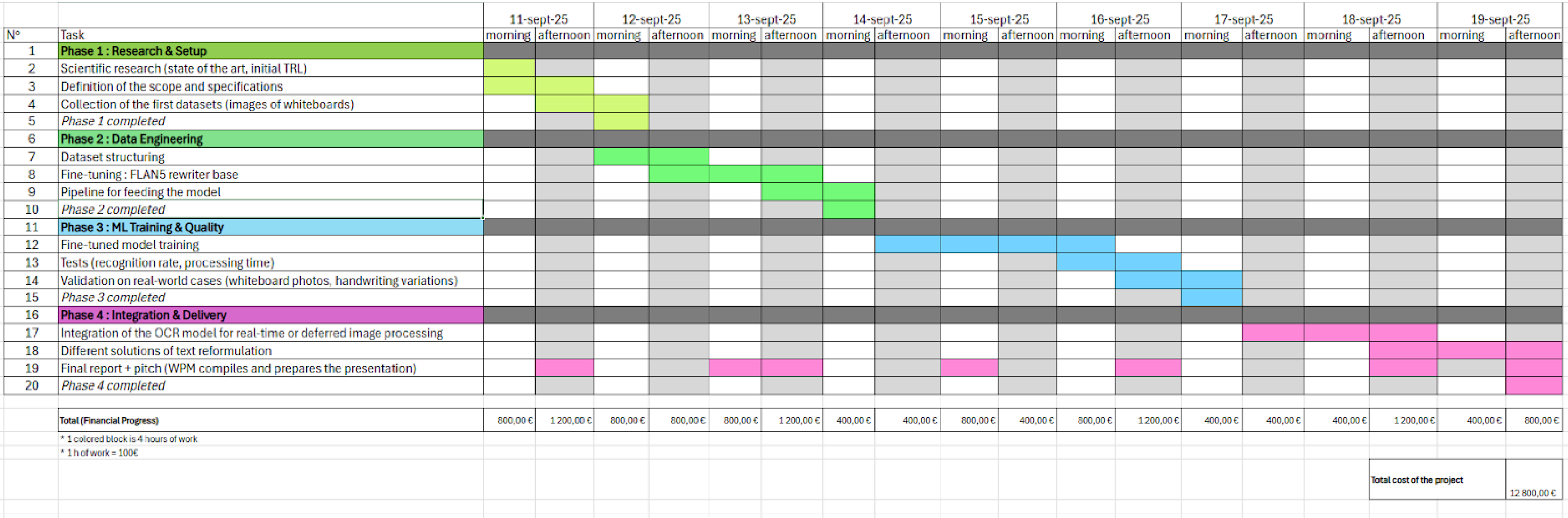
Stakeholders board

Stakeholder	Name	Main Role	Key Responsibilities
Research	Alicia BOSCH	Scientific research and state-of-the-art	Benchmark existing solutions, literature review on OCR and computer vision, dataset research.
Data Engineering	Pierre-Louis LOPEZ	Data preparation and management	Collect and clean images, structure the dataset, design the data pipeline.
ML Training	Jassime ZAIETER	Model training and optimization	Implement vision and OCR algorithms, train and test models, monitor performance.
Quality / Validation	Loan BELLINI	Quality control and evaluation	Define performance metrics (e.g., recognition rate), run validation tests, ensure compliance.
Integration	William INIZIATO	Application development and integration	Build the global pipeline, integrate OCR model into a usable classroom application.
WPM (Project Manager)	Cyndelle NAPOLETANO	Project management and coordination	Coordinate the team, track progress, compile the report, prepare the final pitch and delivery.
Client	Clément GICQUEL	Academic client / Head of Innovation (role)	Provide expectations, validate project relevance, act as decision-maker in simulation.

ISEN MÉDITERRANÉE | L'ÉCOLE DES INGÉNIEURS DU NUMÉRIQUE X Innovation 4You

See4Me

Macro-planning (physical progress) & Financial progress



		11-sept-25		12-sept-25		13-sept-25		14-sept-25		15-sept-25		16-sept-25		17-sept-25		18-sept-25		19-sept-25	
N°	Task	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon	morning	afternoon
1	**Phase 1 : Research & Setup**																		
2	Scientific research (state of the art, initial TRL)	🟩																	
3	Definition of the scope and specifications	🟩	🟩																
4	Collection of the first datasets (images of whiteboards)		🟩	🟩															
5	*Phase 1 completed*			🟩															
6	**Phase 2 : Data Engineering**																		
7	Dataset structuring			🟩	🟩														
8	Fine-tuning : FLAN5 rewriter base				🟩	🟩	🟩												
9	Pipeline for feeding the model						🟩	🟩											
10	*Phase 2 completed*							🟩											
11	**Phase 3 : ML Training & Quality**																		
12	Fine-tuned model training								🟦	🟦	🟦	🟦							
13	Tests (recognition rate, processing time)											🟦	🟦						
14	Validation on real-world cases (whiteboard photos, handwriting variations)												🟦	🟦					
15	*Phase 3 completed*													🟦					
16	**Phase 4 : Integration & Delivery**																		
17	Integration of the OCR model for real-time or deferred image processing														🟪	🟪	🟪		
18	Different solutions of text reformulation																🟪	🟪	🟪
19	Final report + pitch (WPM compiles and prepares the presentation)		🟪			🟪	🟪			🟪			🟪				🟪		🟪
20	*Phase 4 completed*																		🟪
	Total (Financial Progress)	800,00 €	1 200,00 €	800,00 €	800,00 €	800,00 €	1 200,00 €	400,00 €	400,00 €	800,00 €	400,00 €	800,00 €	1 200,00 €	400,00 €	400,00 €	400,00 €	1 200,00 €	400,00 €	800,00 €

* 1 colored block is 4 hours of work
* 1 h of work = 100€

Total cost of the project	12 800,00 €

Conclusion

The project made it possible to develop a complete solution, from image segmentation to the reformulation of clear and structured text. The technical objectives were achieved: fine-tuning of a Flan-T5 model for OCR post-processing, integration of the different modules (DocTR, TrOCR, Flan-T5, TinyLlama), and the implementation of a functional and reproducible pipeline.

Project management was carried out effectively thanks to a good distribution of roles and the use of free resources (Google Docs, Hugging Face, GitHub), which made it possible to remain within budget constraints. The team acquired skills in artificial intelligence, data management, and project coordination.

When comparing the budget of our one-week project (12,800 € for six people) with the actual cost of employing engineers in the French market, the difference becomes clear. Based on average salaries, six AI engineers working full-time for nine days would realistically represent a cost closer to 18,000–20,000 €, not including infrastructure and additional expenses. This highlights that our project has been carried out with a budget well below typical market conditions, underlining both the efficiency of our approach and the strong value generated relative to real-world costs.

Despite these achievements, the evaluation phase also revealed one major limitation. The post-edited outputs, while generally more coherent, sometimes produced responses that could be described as “naive” or inconsistent, leading to frustration within the team. Even with the dual-output strategy, the system could not guarantee a fully reliable experience for visually impaired students. This means that the solution does not yet provide complete autonomy, as occasional errors or illogical reformulations may still hinder understanding. Nevertheless, by combining raw recognition with translated outputs, our work represents a concrete step toward greater accessibility. The project has laid a foundation that can be further developed to enhance reliability and, ultimately, move closer to true independence for students with visual impairments.

Finally, the project has a real societal impact, offering an accessible solution that promotes the autonomy and inclusion of visually impaired students, while opening up perspectives for improvement (handling mathematical symbols, mobile interface, real-time optimization).

References

Link to the database of HuggingFace :
https://huggingface.co/datasets/PleIAs/Post-OCR-Correction

Link to our fine-tuned model on HuggingFace :
https://huggingface.co/ouiyam/see4me-flan5-rewriter-base

Link to our project on GitHub : https://github.com/piloulpz/See4Me

Links to all the models that we used :

- https://huggingface.co/microsoft/trocr-base-handwritten
- https://github.com/mindee/doctr
- https://huggingface.co/google/flan-t5-base
- https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0