BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

# INTELLIGENT DOCUMENT PROCESSING FOR ROMANIAN MEDICAL CERTIFICATES

**Team members**
Nichifor Dragos, IS, 258-2, dragos.nichifor@stud.ubbcluj.ro
Malancioiu Daniel-George, IS, 258-2, daniel.malancioiu@stud.ubbcluj.ro
Lupu Eduard-Adrian, IS, 258-2, eduard.lupu@stud.ubbcluj.ro

2025-2026

## Abstract

This project presents an intelligent system for automatic data extraction from Romanian medical leave certificates (*Certificat de Concediu Medical*). The motivation arises from the need to reduce manual transcription errors and accelerate administrative processing in healthcare and HR workflows.

The proposed solution integrates classical image preprocessing techniques (**OpenCV**) with a modern, deep-learning-based OCR engine (**EasyOCR**), controlled by a region-of-interest (**ROI**) mapping approach. Each document is aligned to a predefined template, cropped per field, recognized, and validated through rule-based postprocessing using regular expressions and format checks. The entire pipeline is deployed as an interactive **Streamlit** web application, offering instant visual feedback and Excel export of the structured results.

Experimental validation on a small dataset of 10 anonymized CNAS certificates demonstrates robust extraction performance for printed text and satisfactory results for mixed printed-handwritten inputs. Average processing time per document remained under 8 seconds on a standard CPU.

Overall, the system achieves a practical balance between classical computer vision and modern AI methods, proving that lightweight, interpretable intelligence can effectively automate document digitization in real-world administrative contexts.

**Keywords:** Intelligent Document Processing, OCR, EasyOCR, OpenCV, Streamlit, ROI Mapping, Medical Certificates.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Chapter 1

# Team composition and roles

## 1.1 Table of main skills and contributions

| **Name** | **Main Skills** | **Roles and Contributions** |
|---|---|---|
| Nichifor Dragos | Python development, OpenCV, Streamlit, OCR pipeline design, testing, documentation | |
| Malancioiu Daniel-George | Python development, OpenCV, Streamlit, OCR pipeline design, testing, documentation | |
| Lupu Eduard-Adrian | Python development, OpenCV, Streamlit, OCR pipeline design, testing, documentation | |
| **Note:** All team members shared identical technical responsibilities and actively participated in every project stage. Development followed a **pair programming** and **peer review** workflow to ensure code consistency and shared understanding. Documentation was written collaboratively, with all members contributing to structure, content, and refinement. This cooperative approach ensured a balanced workload, and higher software reliability. | | |

Table 1.1: Team members, shared skills, and collaborative responsibilities.

## 1.2 Performed tasks (Gantt diagram)

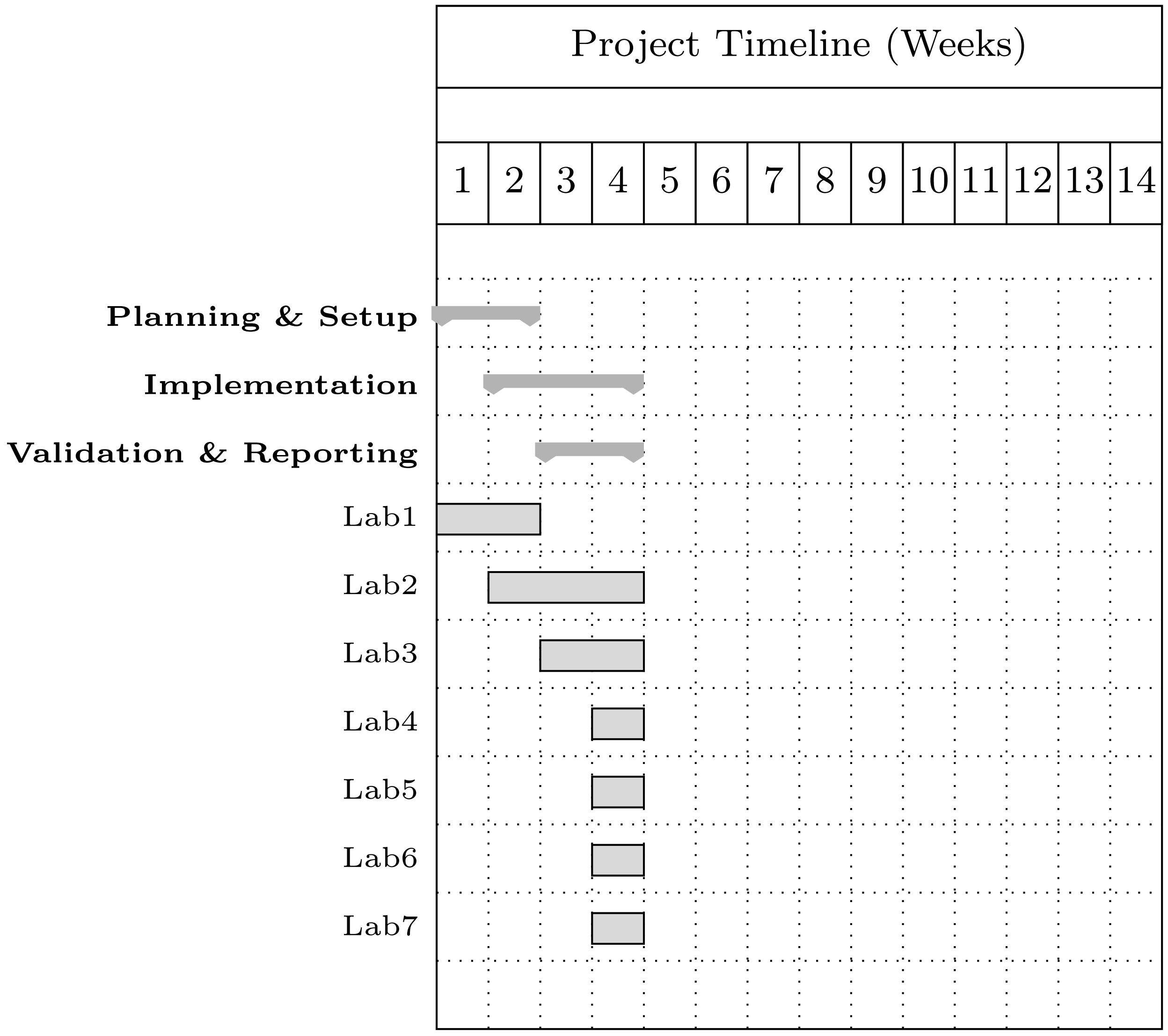

Figure 1.1: Compact schedule by lab milestones.

Chapter 2

# Introduction

## 2.1 What? Why? How?

This report presents an intelligent web-based system designed to automatically extract structured information from Romanian **medical leave certificates** (*Certificat de Concediu Medical*) issued by CNAS.

The main idea behind the project is to build a practical and lightweight solution that transforms scanned or photographed documents into digital records containing key fields such as `Series`, `Number`, `CNP`, `Start Date`, `End Date`, and `Diagnostic Code`. The application provides an end-to-end automated workflow — from image upload to structured JSON or Excel output — reducing manual effort and minimizing transcription errors.

The motivation for this work stems from the inefficiency of manual data entry and the difficulty of using traditional rule-based OCR systems in real-world environments [14]. Romanian medical certificates vary significantly in layout, image quality, and the presence of handwritten text. Such inconsistencies make static template-based extraction unreliable. To address this, the proposed system integrates **EasyOCR** [8] for multilingual printed and handwritten text recognition with **OpenCV**-based preprocessing [3] and a normalized **ROI grid** that maps the layout of the certificate.

Unlike fixed-coordinate OCR pipelines, the ROI-based approach operates on a canonical aligned template (1400×1980) and maps the normalized coordinates to each uploaded image by scaling. Each ROI is cropped and processed individually, providing stable results without full-page OCR.

The entire pipeline is implemented as an interactive **Streamlit** web application [6]. Users can upload certificate images, preview detected regions highlighted in the document, and export extracted data in JSON or Excel format. This design emphasizes interpretability and usability, making the solution suitable for both technical and non-technical personnel in healthcare and HR departments.

In summary, the project introduces an intelligent document processing workflow that combines modern OCR technology with adaptive region mapping, enabling automatic, accurate, and scalable data extraction from heterogeneous Romanian medical certificates.

## 2.2 Paper structure and original contribution(s)

The research presented in this report advances the design and implementation of intelligent algorithms for document processing in administrative and healthcare contexts.

The main contributions of this work are as follows:

- **ROI-based extraction algorithm.** Development of an OCR pipeline that operates on normalized, pre-defined regions of interest (ROIs) defined on a canonical aligned certificate template. Each ROI is dynamically scaled to match the uploaded image, cropped, and passed to EasyOCR for recognition.
- **Lightweight intelligent OCR integration.** Use of EasyOCR for multilingual text recognition, capable of handling both printed and handwritten text, combined with OpenCV preprocessing for image alignment and thresholding.
- **User-friendly Streamlit interface.** An intuitive web application that allows users to upload certificates, preview the detected regions, view structured results, and export them to Excel.
- **Experimental validation.** Evaluation on a small dataset of anonymised certificates showing robust performance under varying lighting, skew, and noise conditions.

The report is structured into the following chapters:

- **Chapter 1** presents the *team composition, individual skills, roles, and the overall work distribution*, together with a Gantt diagram summarizing the performed tasks across labs.
- **Chapter 2** introduces the context, motivation, and project objectives.
- **Chapter 3** defines the scientific problem addressed and explains why intelligent algorithms are required.
- **Chapter 4** reviews related work and key technologies.
- **Chapter 5** details the ROI-driven approach, algorithms, and technologies.
- **Chapter 6** presents the experimental methodology and results.

- **Chapter 7** provides a SWOT analysis summarizing the system's strengths, weaknesses, opportunities, and threats.
- **Chapter 8** concludes the report with final observations and potential future improvements.

# Chapter 3

# Scientific Problem

## 3.1 Problem definition

The addressed problem concerns the automatic extraction of structured information from scanned or photographed Romanian **medical leave certificates**. Such certificates contain key fields required by HR and healthcare systems, including the unique personal identification number (CNP), the start and end dates of the leave, diagnostic codes, and issuer details.

Currently, this information is entered manually, which is slow, error-prone, and inconsistent. The challenge lies in handling multiple certificate layouts, variable image quality, and mixed printed-handwritten text.

Formally, the task is defined as follows:

- **Input:** a scanned or photographed certificate image $I$ with possible skew, noise, or lighting variations.
- **Output:** a structured dictionary represented as:

$$\left\langle \begin{array}{l} \text{SurgicalUrgency, Initial, Continuation, Series, Number,} \\ \text{AvailableForMonth, AvailableForYear, IndemnityCode, CountryInsurer,} \\ \text{CNP, ChildCNP, RegistrationCode, StartDate, NumberOfDays,} \\ \text{EndDate, DiagnosticCode, Adult, RegistrationCUI,} \\ \text{PhysicianCodeDirector, PhysicianCodeHeadMedic, CAS, ReceivedOnDate} \end{array} \right\rangle$$

  that can be exported to Excel.

Challenges include:

1. **Template variability:** minor design differences between CNAS-issued forms.
2. **Image quality:** low resolution, rotation, and shadows in phone captures.
3. **Handwritten text:** reduces OCR reliability on critical fields.
4. **Validation:** ensuring extracted CNPs, dates, and numeric codes have valid formats.

This problem requires an **intelligent** approach because purely rule-based systems cannot generalize beyond a single static template or font. The combination of region-based OCR, learned character recognition, and normalization makes the extraction robust to visual distortions while remaining efficient and interpretable.

The importance of solving this problem lies in its real-world applicability: automating certificate processing reduces human error, accelerates HR procedures, and enables further AI-powered analytics within the healthcare system.

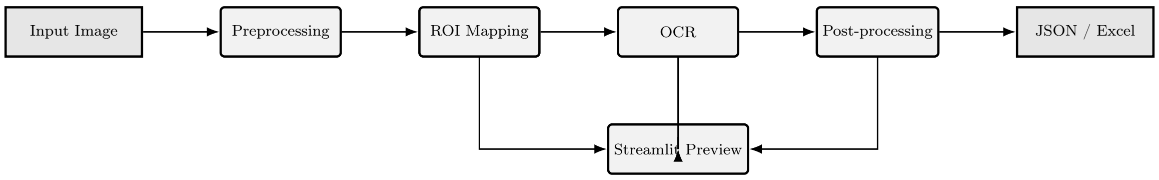

Figure 3.1: Simplified graphical abstract of the document processing pipeline.

Chapter 4

# State of the art/Related work

Document digitization and text extraction have been long-standing goals in intelligent automation. Traditional OCR-based systems have evolved from deterministic template matching toward adaptive, machine-learning-based methods.

## 4.1 Classical approaches

Early OCR engines such as **Tesseract** [14] and **ABBYY FineReader** [1] rely on rule-based character segmentation and pattern matching. They achieve good results on printed text but degrade when faced with handwriting, noise, or low contrast. Classical preprocessing — deskewing, thresholding, and denoising — remains crucial for improving recognition quality [11, 13].

## 4.2 Modern learning-based methods

Recent advances use deep learning to recognize both printed and cursive text. Libraries like **EasyOCR** [8] employ convolutional networks trained on multilingual datasets to perform character detection and recognition jointly. In contrast to Tesseract, EasyOCR generalizes better to varied fonts and handwritten inputs, making it suitable for heterogeneous administrative documents. Other research directions, such as **LayoutLM** [15] and **DocFormer** [2], combine text with spatial layout embeddings, but these models require large datasets and substantial compute power.

## 4.3 Hybrid and practical solutions

For specialized, small-data domains such as Romanian CNAS certificates, lightweight hybrid approaches are preferable. Our solution combines classical image preprocessing (**OpenCV** [3]) with a deep-learning OCR engine (**EasyOCR** [8]), controlled by a region-based extraction logic defined through an ROI map. This design keeps inference deterministic, fast, and interpretable while maintaining adaptability to new templates.

## 4.4 Tools and technologies used

- **OpenCV:** grayscale conversion, deskewing, thresholding, and image normalization [3].
- **EasyOCR:** recognition engine for printed and handwritten text [8].
- **Streamlit:** front-end framework for fast UI prototyping and visualization [7].
- **Pandas & OpenPyXL:** for structured data management and Excel export [9, 4].
- **Python:** main programming language ensuring modularity and cross-platform compatibility [5].

Compared to previous rule-based systems, this hybrid pipeline delivers robust performance with limited data, minimal manual configuration, and full reproducibility.

# Chapter 5

# Investigated approach

## 5.1 Overview

The system operates by mapping a predefined set of normalized regions of interest (ROIs) onto the uploaded image, cropping each region, and running OCR individually. This ensures consistent extraction even if the document is slightly misaligned or scaled.

**ROI grid definition and mapping.** Rectangles are manually annotated on an aligned certificate image of fixed size (1400×1980). Each rectangle is stored as normalized coordinates $[t, l, b, r]$ in a JSON file (`roi_map.json`). When a new image is uploaded, these coordinates are scaled to the image's actual dimensions. Each cropped region is passed to EasyOCR [8], and the recognized text is postprocessed according to its field type.

**Algorithm 1** ROI-driven OCR pipeline (as implemented)

```
Load normalized ROI map $\mathcal{R}$ from roi_map.json
Read input image $I$, convert to grayscale, deskew, and threshold
for each field $k$ with ROI $(t, l, b, r) \in \mathcal{R}$ do
    $(y_1, y_2, x_1, x_2) \leftarrow \text{scale}(t, l, b, r, \text{height}(I), \text{width}(I))$
    $C \leftarrow I[y_1 : y_2,\ x_1 : x_2]$
    $s \leftarrow \text{EasyOCR}(C)$
    $v_k \leftarrow \text{postprocess}(s, \text{kind}(k))$
end for
return JSON dictionary $\{k \mapsto v_k\}$ and preview image with ROI overlays
```

**Per-field postprocessing**

- **digits:** keep numeric characters, enforce expected length (e.g., CNP = 13 digits).
- **date:** regex-based parsing and normalization to ISO format.

- **text:** trim, preserve diacritics, remove duplicates.
- **checkbox:** classify as checked/unchecked based on average pixel intensity.

## 5.2 Validation and output

After all ROIs are processed, the system validates the extracted values using regular expressions and consistency checks (e.g., date intervals, numeric formats). The results are displayed in the Streamlit interface [7], shown as both JSON and a pandas [9] table, and optionally exported as an Excel file via OpenPyXL [4].

## 5.3 Software architecture

The implementation follows a modular structure:

- `ocr_utils.py` - ROI scaling, OCR execution, postprocessing, validation.
- `app.py` - Streamlit UI, language switching (English/Romanian), and Excel export.
- `roi_map.json` - ROI definitions.

Components are loosely coupled, and the OCR engine can be replaced or retrained independently.
*AI serving:* the OCR component is **embedded** in the application (no external microservice); model weights are loaded at runtime by EasyOCR.

## 5.4 Debugging and testing

During development, a dedicated debugging script (`debug_rois.py`) crops all regions of interest (ROIs) from the aligned certificate template and saves them to disk for visual verification. This process ensures that ROI coordinates are correctly aligned with the target document layout before running OCR.

The Streamlit interface itself serves as a real-time visualization and testing environment: each uploaded image is processed, and detected text regions are overlaid and displayed interactively for inspection. Formal validation tests (e.g., for CNP or date formats) are planned as future improvements.
*Iterative perspective:* ROI coordinates were adjusted iteratively based on the Streamlit preview until misalignments were eliminated, after which the ROI map was fixed and versioned.

## 5.5 Development environment.

The application was implemented in **Python** [12] using the **Visual Studio Code** editor [10] on Windows 11. Development used a local `venv` virtual environment, with dependencies managed via a `requirements.txt` file. Bugs were identified through ROI misalignment inspection, logging OCR outputs, and verifying extracted field validity during live runs in the Streamlit interface. Average workload during testing was below 8 seconds per certificate on a standard laptop CPU.

## 5.6 Versioning and Continuous Delivery

The project was developed collaboratively in a single **Git** repository, which contains both the application code and documentation. Version control was used to track progress, synchronize work between team members, and maintain a clean commit history.

All testing and debugging were performed locally in a controlled `venv` environment. Once validated, updates were committed and pushed to the shared repository, ensuring all members had the latest working version. This simple yet effective workflow provided continuous integration at the local level and reproducible delivery through Git.

# Chapter 6

# Application (numerical validation)

## 6.1 Methodology

The evaluation focuses on the correctness of extracted fields and runtime performance. Three questions guided the experiments:

1. How accurately are key fields (Series, Number, CNP, Dates, Diagnostic Code) extracted from typical CNAS certificates?
2. How robust is the ROI scaling under skew and resolution changes?
3. What is the end-to-end runtime for a single document?

**Testing scope.** We conducted (i) **data testing** by validating each field against expected formats (digits, dates, lengths), (ii) **integration testing** by running end-to-end flows (UI upload → ROI → OCR → Excel export) on multiple images, and (iii) **AI quality testing** by computing field-wise extraction accuracy and checkbox recognition accuracy against manually verified ground truth.

## 6.2 Data

A dataset of 10 anonymised certificates was used. All certificates share the CNAS form layout but differ slightly in proportions. Fields were annotated manually to verify accuracy. Images were stored in a versioned data folder to ensure reproducibility.
*Storage rationale:* a simple, versioned folder structure ("data/" and "results/") was preferred over a data warehouse or lake due to the small dataset size and the need for quick, local iteration.

## 6.3 Results

Table 6.1: Summary of extraction performance.

| Metric | Value |
|---|---|
| Field extraction accuracy (printed text) | 85-90% |
| Field extraction accuracy (mixed text) | 60-65% |
| Checkbox recognition accuracy | 100% |
| Average runtime / document | $< 30$ s |

The ROI mapping proved stable across all evaluated certificate samples that followed the standard CNAS layout. Most recognition errors appeared in low-contrast regions or handwritten zones, particularly for diagnostic codes and partially filled checkboxes.

## 6.4 Discussion

The results confirm that the hybrid, ROI-driven pipeline is effective for structured document extraction. While EasyOCR's handwritten recognition occasionally misclassifies letters, the overall performance is satisfactory given the small dataset and real-world noise. Future work will focus on dynamic ROI correction using contour detection and on integrating a confidence-based verification layer.

## 6.5 Experiments tracking

All test results were stored directly in the shared Git repository for traceability. Each experiment produced an Excel file saved under the `results/` folder, containing the extracted fields and manual validation data. This folder served as a simple yet effective tracking mechanism, allowing all team members to compare outputs and verify improvements across commits.

Given the small dataset and deterministic OCR pipeline, no external experiment-tracking tools (e.g., DVC or MLflow) were required.

# Chapter 7

# SWOT Analysis

Chapter 8

# Conclusion and future work

Try to emphasise the strengths and the weaknesses of your approach. What are the major shortcomings of your current method? For each shortcoming, propose additions or enhancements that would help overcome it.

Briefly summarize the important results and conclusions presented in the paper.

- What are the most important points illustrated by your work?
- How will your results improve future research and applications in the area?

# Bibliography

[1] ABBYY. *ABBYY FineReader Engine: OCR SDK*, 2024. Vendor documentation.

[2] Srinivas Appalaraju et al. Docformer: End-to-end transformer for document understanding. In *Proc. ICCV*, 2021.

[3] G. Bradski. The opencv library. In *Dr. Dobb's Journal of Software Tools*, 2000.

[4] OpenPyXL Developers. Openpyxl: A python library to read/write excel 2010 xlsx/xlsm files. https://openpyxl.readthedocs.io/, 2024.

[5] Python Software Foundation. Python 3.11 documentation. https://docs.python.org/3.11/, 2023.

[6] Streamlit Inc. Streamlit: The fastest way to build data apps in python. https://streamlit.io, 2020.

[7] Streamlit Inc. Streamlit documentation. https://docs.streamlit.io, 2023.

[8] JaidedAI. Easyocr: Ready-to-use ocr with 80+ supported languages. https://github.com/JaidedAI/EasyOCR, 2020.

[9] Wes McKinney. Data structures for statistical computing in python. *Proc. SciPy*, 2010.

[10] Microsoft Corporation. Visual studio code: Code editing. redefined. https://code.visualstudio.com/, 2024. Accessed: 2025-10-20.

[11] W. Niblack. An introduction to digital image processing. 1985.

[12] Python Software Foundation. Python programming language, version 3.11. https://www.python.org/, 2024. Accessed: 2025-10-20.

[13] J. Sauvola and M. PietikÃ¤inen. Adaptive document image binarization. *Pattern Recognition*, 33:225–236, 2000.

[14] Ray Smith. An overview of the tesseract ocr engine. *Proc. ICDAR*, pages 629–633, 2007.

[15] Yiheng Xu et al. Layoutlm: Pre-training of text and layout for document image understanding. In *Proc. KDD*, 2020.