INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Generating presentation documents from procedural resources

PEDRO RAFAEL FERREIRA PACHECO
Setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Generating presentation documents from procedural resources

Pedro Rafael Ferreira Pacheco

Dissertation
Computer Engineering
Specialisation Area of Software Engineering

Advisor: Isabel Azevedo

Oporto, September 27, 2025

Statement of Integrity

I hereby declare that I have conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore, the work presented in this document is original and authored by me, having not previously been used for any other end. The exceptions are explicitly recognised in the section "Ethical considerations" of the first chapter. This section also states how AI tools were used and for what purpose.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Oporto, September 27, 2025

Abstract

The creation of presentation slides from procedural documents is a repetitive and time consuming task that typically requires substantial manual effort. This dissertation presents the design and development of a system that automates this process by converting procedural documents into structured and visually coherent presentation slides. The proposed solution integrates Natural Language Processing (NLP) techniques, text and image extraction and the use of local Large Language Model (LLM)s to perform summarization and content structuring.

The system follows a modular pipeline composed of three main stages, extraction of textual and visual elements from Portable Document Format (PDF) documents, processing and structuring of the extracted content using LLMs based summarization and organization strategies and automatic generation of presentation slides in PowerPoint Open XML Presentation (PPTX) format, enriched with customizable themes. The implementation leverages open-source tools such as `PyPDF2`, `pdfminer.six` and `python-pptx`, while ensuring flexible integration with different LLMs through the Ollama framework.

Evaluation was carried out through unit, integration, functional and non-functional testing. The results confirmed that the system fulfills its functional requirements, reliably importing documents, generating coherent slides, supporting customization and export features. Non-functional testing highlighted strengths in usability and maintainability, while also exposing limitations in performance when using larger models, reliability under heavy load and the need for improved security hardening.

This work proves the concept of value and feasibility in semi-automatic transformation of procedural documents into presentations. The proposed system reduces the time and effort required for preparing presentations, offering an accessible, modular and extensible solution that can be further optimized and scaled in future research. The source code is publicly available as open source for testing, improvement and continued development (Pacheco, 2025).

Keywords: Document-to-Slide Automation, Automatic Slide Generation, Natural Language Processing, Large Language Models, Text Summarization, PDF Processing

Resumo

A criação de apresentações a partir de documentos de procedimentos é uma tarefa repetitiva e demorosa que, tipicamente, exige um esforço manual substancial. Esta dissertação apresenta o design e o desenvolvimento de um sistema que automatiza este processo, convertendo documentos de procedimentos em apresentações estruturadas e visualmente coerentes. A solução proposta integra técnicas de NLP, extração de texto e imagem e a utilização de LLMs locais para realizar a sumarização e a estruturação de conteúdo.

O sistema segue um pipeline modular composto por três fases principais, a extração de elementos textuais e visuais de documentos PDF, o processamento e a estruturação do conteúdo extraído através de estratégias de sumarização e organização baseadas em LLMs e a geração automática de apresentações em formato PPTX, enriquecidas com temas personalizáveis. A implementação utiliza ferramentas de código aberto como `PyPDF2`, `pdfminer.six` e `python-pptx`, ao mesmo tempo que assegura uma integração flexível com diferentes LLMs através da framework Ollama.

A avaliação foi realizada através de testes unitários, de integração, funcionais e não funcionais. Os resultados confirmaram que o sistema cumpre os seus requisitos funcionais, importando documentos de procedimentos de forma fiável, gerando slides coerentes e suportando funcionalidades de personalização e exportação. Os testes não funcionais destacaram pontos fortes em usabilidade e manutenibilidade, ao mesmo tempo que expuseram limitações no desempenho com a utilização de modelos maiores, na fiabilidade sob carga pesada e na necessidade de reforçar a segurança.

No geral, este trabalho demonstra a viabilidade e o valor da automação da conversão de documentos de procedimentos em apresentações. O sistema proposto reduz o tempo e o esforço necessários para a preparação de apresentações, oferecendo uma solução acessível, modular e extensível que pode ser otimizada e escalada em investigação futura. O código fonte está publicamente disponível como *open-source* para testes, melhorias e desenvolvimento contínuo (Pacheco, 2025).

Acknowledgement

Agradeço aos meus pais por todos os valores que me deram ao longo da vida e por me terem apoiado incondicionalmente. Sempre acreditaram em mim e no meu desempenho. Sempre me deram apoio e as condições necessárias para que pudesse concluir esta etapa com competência, conforto e alegria. Um agradecimento aos meus avós e aos restantes membros da família, que além de me incutirem valores importantes, foram uma fonte de motivação ao demonstrarem que a vontade e a perseverança podem superar qualquer obstáculo.

Agradeço de forma especial à minha namorada e melhor amiga, Ana Ferreira. Ela está ao meu lado durante quase uma década e é uma fonte de inspiração, alegria e um apoio incondicional. A sua motivação foi crucial para me ajudar a terminar este mestrado.

Agradeço ao Gonçalo Silva e Rui Alves, os meus melhores amigos, que me acompanham há mais de metade da minha vida. Agradeço por me terem ajudado nos bons e maus momentos e pelo apoio e bem estar que sempre partilharam comigo.

Gostaria de expressar a minha gratidão à minha orientadora, Isabel Praça, que me orientou ao longo deste projeto.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AI	Artificial Inteligence
API	Application Programming Interface
BRNN	Bidirectional Recurrent Neural Networks
CNN	Convolutional Neural Network
CSRF	Cross-Site Request Forgery
DBN	Deep Belief Networks
DL	Deep Learning
EC	Exclusion Criteria
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
IC	Inclusion Criteria
ILP	Integer Linear Programming
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
OCR	Optical Character Recognition
ORM	Object Relational Mapping
PCA	Principal Component Analysis
PDF	Portable Document Format
PII	Personally Identifiable Information
PIL	Python Imaging Library
PPTX	PowerPoint Open XML Presentation
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Networks
SMR	Systematic Mapping Review
SVM	Support Vector Machine
SVR	Support Vector Regressor
WBS	Work Breakdown Structure

Chapter 1

Introduction

The topic of the dissertation is introduced in this chapter along with its context, problem, objectives, sources of information and the approach used to carry it out.

1.1 Background and Context

The development of instructional materials represents a foundational element in the creation of educational resources, often requiring significant time investment and human intervention (Franco, Thirumoorthy, and Muneeswaran, 2016). Procedural documents represent a frequent and practical form of instructional content. These texts describe a sequence of ordered steps necessary to complete a specific task and are widely found in contexts such as technical manuals, instructional guides and user documentation. Their structure typically includes three core components, preconditions that set the context, a clear progression of steps and a concluding section that either summarizes the process or describes the expected outcome (Franco, Thirumoorthy, and Muneeswaran, 2016).

To meet this need, research has focused on the automation of such workflows. Approaches based on Natural Language Processing (NLP), text summarization and multimedia integration have shown promise in reducing manual effort and enabling the rapid generation of educational outputs. These technologies offer new possibilities for converting structured textual resources into engaging formats like slide presentations and learning modules (Nadkarni, Ohno-Machado, and Chapman, 2011; Sun et al., 2021).

Although promising, the automation of procedural content presents a unique set of difficulties. Unlike expository or narrative texts, procedural documents rely on strict sequential structure, instructional clarity and often visual correlation between text and diagrams. As such, successful automation requires not only linguistic understanding, but also the capacity to identify logical hierarchies and preserve instructional coherence throughout the transformation process (Liddy, 2001).

This issue is particularly relevant to stakeholders in both educational and technical sectors, including software trainers, content creators, instructors and learners, who depend on procedural resources to facilitate knowledge transfer and task completion. The current work assumes that the documents to be processed are digitally structured and that end users possess the fundamental competencies required to interpret the generated slide content. Additionally, the proposed system is designed under practical constraints, relying exclusively on open-source technologies and methods rooted in NLP and Machine Learning (ML), to ensure feasibility within the limits of available time and computational resources.

1.2 Problem Description

Digital documentation continues to grow and people face increasing challenges in transforming texts into structured presentation slides (Sun et al., 2021). The conventional method for performing this task remains largely manual, demanding substantial effort and a high degree of precision. This manual transformation process is not only time consuming, but also prone to inconsistencies, especially in preserving the logical flow of information, maintaining textual cohesion and integrating visual elements in a meaningful way (Franco, Thirumoorthy, and Muneeswaran, 2016).

Presentations serves as indispensable tools for sharing knowledge transfer and reinforcing assimilation. Even so, a discernible lacuna persists in the availability of automated systems capable of consistently converting structured procedural documents into concise and visually coherent slide formats. This gap hinders both the scalability and consistency of content production, ultimately limiting the ability of educators and organizations to deliver information efficiently and uniformly across diverse learning contexts (Franco, Thirumoorthy, and Muneeswaran, 2016).

This dissertation focuses on the semi-automation of slide generation from procedural documents through the application of Large Language Model (LLM)s. The main challenge is maintaining the original logic and relevance of the extracted information, while formatting it to suit a presentation. To reinforce the information and preserve the presentations purpose it is important to ensure all text, images and diagrams are properly aligned. (Fu et al., 2022).

By leveraging advancements in text summarization, layout detection and multimedia integration, this work proposes a system designed to bridge the gap between static documents and dynamic presentations. The objective is to deliver a dependable and practical tool that reduces the manual workload and enhances the quality and clarity of instructional materials.

1.3 Objectives and Research Questions

The primary goal of this dissertation is to develop an application capable of semi-automatically generating presentations from procedural documents. This involves applying techniques to transform structured documents containing procedural instructions into visually organized and easy to understand presentation materials. By leveraging advanced techniques in NLP, document layout analysis and visual content integration. The project seeks to address the challenges of automation in this domain and provide an efficient tool for educators and professionals.

To guide the research and ensure alignment with its goals, the following research questions have been defined:

- RQ1 - How can the conversion of procedural documents into slides be automated while maintaining content integrity, logical flow and clarity?
- RQ2 - What are the most efective methods to extract key steps from procedural resources while preserving their sequential nature?

By addressing these questions, the dissertation aims to propose and validate a new solution that balances automation with the quality and coherence required for presentations.

1.4 Research Methodology

The research methodology adopted in this dissertation is the 'Design and Creation' strategy. This method is suitable for projects focused on the development and evaluation of artifacts, such as software systems. It involves iterative processes of planning, implementation and evaluation, ensuring that the solution created aligns with the problem and meets the objectives.

The main objective of this dissertation is to meticulously engineer an application endowed with the capability to semi-autonomously generate presentation slides directly from comprehensive procedural documents.

This approach provides a structured way to address research questions, ensuring that the architectural design of the system meets user's requirements and project's objectives.

1.4.1 Design and Creation

The choice of the 'Design and Creation' methodology is justified by its suitability for software engineering research and its alignment with the project's goals.

The core objective of the dissertation is to create a functional application, making this strategy an ideal choice. It emphasizes the creation, implementation and iterative improvement of artifacts, ensuring that the final product meets its intended purpose. The research questions emphasize automation and the extraction of procedural steps, which are linked to the development of a system.

This approach enables us to iteratively test and refine the system ensuring its effectiveness. In addition, conducts an assessment to validate the effectiveness of the developed application. To determine the application's performance and several metrics are analyzed.

1.5 Report Structure

- **Introduction:** Establishes the context of the work, defines the problem, presents the objectives that guided the project and the research methodology.
- **Background:** Introduces the foundational concepts, tools and technologies that underpin the developed solution.
- **Skills and Planning:** Describes the technical skills required for the project, highlights strengths and weaknesses and outlines the project methodology, including planning and milestones.
- **Literature Review:** Summarizes relevant work in procedural document processing, text summarization and slide generation, highlighting challenges and research gaps.
- **Design and Development:** Presents the system architecture and provides details of the implementation, including extraction, processing and generation modules.
- **Testing and Evaluation:** Reports on functional and non-functional testing, usability studies, evaluation of performance, reliability, maintainability and security.
- **Conclusions:** Summarizes the main contributions, discusses limitations and describe possible directions for future work.

Chapter 2

Background

This section is dedicated to presenting a comprehensive overview of the fundamental concepts and methods inherent in document analysis, text summarization and automated slide generation. This foundational knowledge is crucial, as it prepares the ground for an in depth understanding of the methodologies and approaches that will be developed in the forthcoming chapters, thereby ensuring a complete assimilation of the essential components structuring the present project.

2.1 Artificial Inteligence

Generative models have a long history in science, dating back to the 1950s with the development of Hidden Markov Models (Knill and Young, 1997) and Gaussian Mixture Models (Reynolds, 2009). These models were mainly used to generate sequential data, such as speech and time series. However, it wasn't until the advent of Deep Learning (DL) that generative models saw significant improvements in performance (Goodfellow et al., 2014).

As illustrated in Figure 2.1, DL is a subset of ML, which in turn is a subset of Artificial Inteligence (AI). The hierarchical nature of these fields highlights how advancements in AI have progressively enabled more complex and effective generative models.

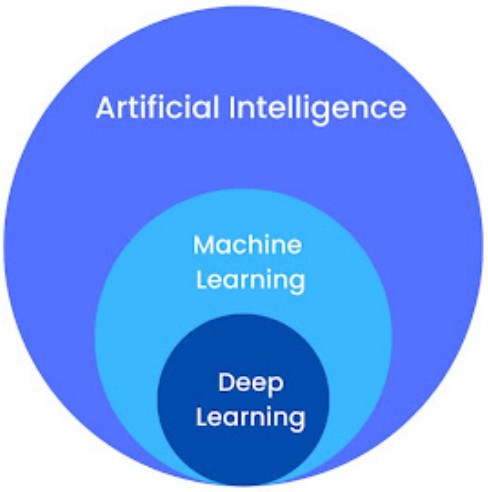

Figure 2.1: Hierarchy within Artificial Inteligence (Harth, 2022).

Computer systems have the ability to simulate human reasoning, allowing them to perform complex tasks that transcend programmed rules. This capacity includes continuous learning and adaptation to new information. According to (Poole, Mackworth, and Goebel, 1998), an intelligent system is one that acts appropriately to the circumstances, is flexible, learns from experience and makes informed decisions. Feature engineering encompasses the process of judiciously selecting and subsequently transforming raw data into salient and pertinent features, whereas feature extraction pertains to the identification of underlying patterns and significant structures within the data (Mesquita, 2021).

2.1.1 Machine Learning

According to Zhou (2021), ML supports the development of systems that adapt and improve based on observed data patterns. Pre-processing the data constitutes the main step in the process, focusing on selecting and refining the most pertinent data to guarantee more efficacious model training (Mesquita, 2021). Like human reasoning, wherein inferences are derived from recurrent experiences, ML simulates experiential learning by extracting meaning from data (Zhou, 2021).

Image 2.2 shows the outline of the ML algorithms.

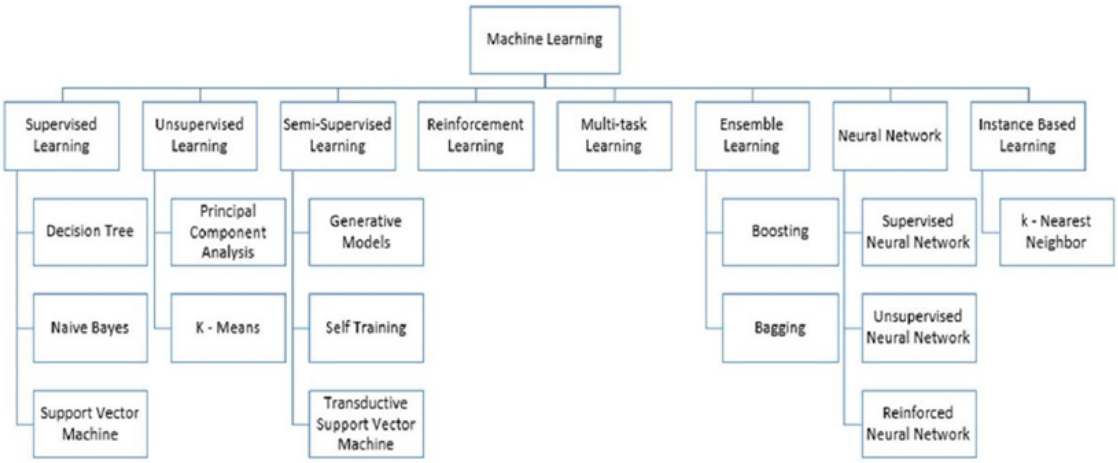

Figure 2.2: ML algorithms (Mahesh, 2019).

Some examples of algorithms and their explanations from ML include (Mahesh, 2019):

- **Support Vector Machine (SVM):** classifies data using filters generated from training data.
- **K-Means:** organizes data into *k* groups, positioning group centers strategically to optimize results.
- **Generative Models** can generate data by modeling both characteristics and classes, using the probability distribution P(x, y) to generate data points.
- **Decision Tree:** uses a tree structure to classify data.
- **Random Forest:** a classification algorithm composed of several Decision Trees, providing greater robustness in the results.

- **K-Nearest Neighbors (KNN):** estimates which group a given data point probably belongs to.

These algorithms represent some of the most common techniques in ML and can be adjusted according to the characteristics of the data and the objectives of the (Mahesh, 2019) project.

2.1.2 Deep Learning

DL, a subfield of ML, has stood out for its ability to learn complex representations and perform advanced tasks from large volumes of data. Based on deep artificial neural networks, DL uses multiple layers of processing to extract high level features from structured or unstructured data, such as text, images and audio. This field has revolutionized areas such as computer vision, NLP and pattern recognition, becoming an essential tool for automation projects, such as the development of automatic presentations from procedural documents (Shinde and Shah, 2018).

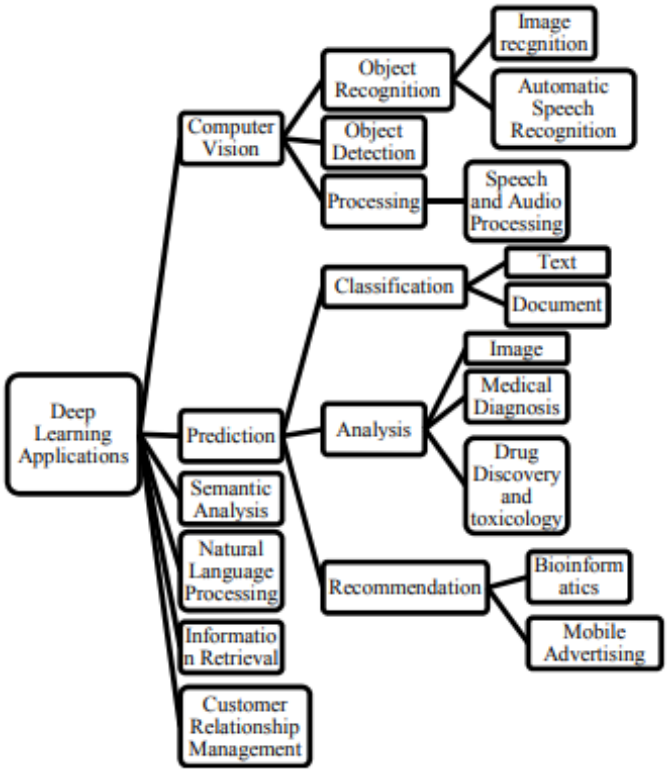

Figure 2.3: DL applications (Shinde and Shah, 2018).

DL has played a central role in recent advances in NLP, contributing significantly to tasks such as information extraction, content summarization and text generation, which are key areas for processing large volumes of data (Kim, 2016). These techniques are widely applicable, as illustrated in Figure 2.3 and have been explored in different contexts due to their effectiveness and versatility.

The main DL techniques include:

- **Restricted Boltzmann Machine (RBM):** A stochastic generative model of artificial neural networks that learns the probability distribution associated with its input set.

- **Recurrent Neural Networks (RNN):** Useful for recovering stored patterns from corrupted versions and precursors to techniques such as Boltzmann machines and autoencoders.
- **Convolutional Neural Network (CNN):** Architecture initially introduced by LeCun et al. in 1998 in the article Gradient-Based Learning Applied to Document Recognition, with application in visual and textual recognition tasks.
- **Bidirectional Recurrent Neural Networks (BRNN):** Introduced by Schuster and Paliwal in 1997, these networks can process data in both temporal directions, capturing contextual dependencies in sequences.
- **Long Short-Term Memory (LSTM):** Developed to overcome temporal limits in sequences, these networks maintain the constant flow of error by means of specific units called "constant error carrousels", allowing the learning of long term dependencies.
- **Deep Belief Networks (DBN):** Deep neural network models that learn feature hierarchies and are widely used in classification and dimensional reduction tasks.

These techniques represent some of the pillars of DL and have enabled significant advances in different areas, consolidating themselves as indispensable tools for solving complex data analysis problems (Shinde and Shah, 2018).

Chapter 3

Skills and Planning

The abilities needed and the planning for the project are covered in this chapter. The skills identified included technical relevant to tackling the challenges of automating the conversion of procedural documents into slide presentations. An evaluation of strengths and weaknesses was conducted, followed by strategies designed to mitigate the identified gaps. In parallel, an idealized project planning was outlined to establish a structured roadmap, enabling the alignment of resources, milestones and deadlines with the project objectives.

3.1 Skills required for the Project

To successfully execute this dissertation, a diverse set of skills is required. These include:

- **Large Language Models:** Understanding of LLM architecture, capabilities, and limitations, including their use for summarization, classification and content generation
- **Prompt Engineering:** Ability to design and refine prompts to guide LLM behavior effectively and obtain relevant and accurate responses
- **Data Preprocessing:** Knowledge of text cleaning, segmentation and structuring to ensure quality input for LLM based systems
- **Presentation Design:** Ability to structure content visually and logically for effective slide presentations

The Table 3.1 evaluates each skill in terms of its importance to the project (on a scale from 1 to 5) and includes a self assessment of required skills.

Skill	**Importance (1-5)**	**Self-assessment (1-5)**
Large Language Models	4	2
Prompt Engineering	5	2
Data Preprocessing	4	2
Machine Learning	4	2
Presentation Design	4	3

Table 3.1: Key skills for the project and self-assessment

This evaluation helps identify areas of strength and those requiring further improvement, guiding the planning of professional development activities throughout the project.

3.2 Strengths

The successful execution of this dissertation relies on a balanced combination of technical expertise and soft skills.

- **Presentation Design (3/5):** Structuring and visually organizing presentation content, ensuring logical flow and aesthetic consistency.

These strengths collectively contribute to the project's success, enabling the development of a robust and well documented solution.

3.3 Weaknesses

While the project benefits from several strengths, some weaknesses were identified that may pose challenges during its execution. These areas require attention and improvement to ensure the successful completion of the dissertation:

- **Large Language Models (2/5):** Proficiency in LLMs for tasks.
- **Prompt Engineering (2/5):** Ability to design and refine prompts to guide LLM outputs effectively.
- **Data Preprocessing (2/5):** Insufficient experience with data cleaning and transformation techniques could hinder the preparation of high quality datasets. As this step is critical to the accuracy of machine learning models, this weakness needs to be mitigated promptly
- **Machine Learning (2/5):** Although there is foundational knowledge in machine learning, the lack of advanced expertise may limit the project's ability to implement state of the art models effectively

These weaknesses highlight areas where focused effort is needed to bridge technical skill gaps. Addressing these limitations through targeted learning and practice in relevant technical areas will ensure that the project remains on track and meets its objectives.

3.4 Plans for Improvement

To address the weaknesses identified and enhance the overall skill set required for the project, a structured plan for improvement has been devised. The following strategies aim to bridge the gaps and ensure the successful execution of the dissertation:

- **Large Language Models:** To develop skills in working with LLMs, the course *"Ollama Course – Build AI Apps Locally"* by *freeCodeCamp.org* on YouTube was followed. This course covers the installation and configuration of Ollama, running local LLMs and integrating models like Gemma into Python applications. The goal is to build practical expertise in interacting with LLMs and effectively integrating them into the system architecture for summarization and content generation tasks.
- **Prompt Engineering:** To address the lack of experience in prompt engineering, the course *"Prompt Engineering Tutorial – Master ChatGPT and LLM Responses"* by *freeCodeCamp.org* was completed. This course provides strategies for crafting effective prompts, controlling output formats, and optimizing LLM responses through prompt iteration. The proficient acquisition of this aptitude is paramount, serving to

unequivocally guarantee the generation of presentation slide content that is both of superior quality and intrinsically relevant, derived directly from comprehensive procedural documentation.

- **Machine Learning and Data Preprocessing:** To address gaps in ML and data preprocessing skills, the course *"Machine Learning for Everybody – Full Course"* by *freeCodeCamp.org* on YouTube was completed. This course provides a comprehensive overview of key concepts, including feature selection, classification and regression techniques, model training, data preparation and implementations of various algorithms such as KNN, Naive Bayes, Logistic Regression, Support Vector Machine, Neural Networks, K-Means and Principal Component Analysis (PCA). Practical examples and exercises will be incorporated to reinforce the theoretical knowledge gained and to ensure applicability to the specific needs of the project.

By following this plan, the gaps identified in key areas will be effectively addressed, ensuring that the author is well prepared to meet the technical and methodological demands of the dissertation.

3.5 Project Charter

This project charter functions as a cornerstone document, assiduously articulating the nascent planning stages and the comprehensive breadth of the undertaking. Within this section, a detailed exposition of the essential constituents of the project charter is provided, thereby establishing a robust framework indispensable for its effective and successful implementation.

3.5.1 Stakeholders

Stakeholders play a critical role in the successful execution of the project. They contribute to the development process through their expertise, influence and interest in the project's outcomes. The table 3.2 outlines the key stakeholders, their roles, levels of power and interest and the benefits they derive from the project:

Stakeholder	**Role**	**Power (0-5)**	**Interest (0-5)**
Student	End-users of the application	5	5
Educational Institutions	Potential end-users	3	5
Teachers	End-users of the application	2	4
Business Institution	End-users of the application	3	4

Table 3.2: Stakeholder Analysis

The stakeholder analysis reveals the importance of the different groups to the success of the system. Students, with a score of 5 for both power and interest, emerge as the main users, whose feedback will be decisive in the adoption and evolution of the tool. Their interest is driven by the direct benefit of automation in the creation of educational materials.

Educational institutions, with a power of 3 and interest of 5, exert some influence on the adoption of the system, although they depend on the acceptance of teachers and students. Their interest lies in the efficiency and standardization of the generation of teaching materials.

Teachers, with a power of 2 and an interest of 4, have a moderate influence, as their acceptance can have an impact on institutional adoption. They benefit from automation, but may have concerns about control over content.

Business institutions, characterized by a power rating of 3 and an interest level of 4, possess the capacity, contingent upon their specific operational model, to either provide financial underpinning for the project or exert influence over its strategic trajectory. They see automation as a competitive advantage to reduce costs and improve efficiency.

Stakeholder interests are:

- **Student:** Gains technical expertise, project management skills and contributes to academic work
- **Educational Institutions:** Receives a tool that automates slide creation, reducing time and effort for educators
- **Teachers:** Receive an automated tool that simplifies the process of generating presentations
- **Business Institution:** Gains a competitive advantage by leveraging the application to enhance automation in content creation processes, reduce operational costs and expand their portfolio of solutions. They benefit from potential commercial applications and improved productivity for internal or client facing processes

3.5.2 Scope

This project focuses on the development of a system designed to generate presentation slides from procedural documents with limited user intervention. The primary goal is to convert structured written content into well organized, visually coherent slide presentations. To accomplish this, the system employs techniques in NLP, layout detection and text summarization, ensuring that the original sequence and logic of the instructions are preserved.

By clearly defining this scope, the project remains manageable and technically feasible within the constraints of the dissertation timeline. There are also opportunities for future development, including expanding support for multiple languages and enhancing visual design capabilities.

3.5.3 Objectives

The primary objective of this dissertation is to develop an application capable semi-automatically generating presentations from procedural documents. This goal focuses on creating a tool that transforms structured textual content into visually organized slides, ensuring that the logical flow and sequence of procedural steps are preserved.

To achieve this overarching goal, the following specific objectives are defined:

- **Utilize LLMs for Summarization and Content Generation:** Employ local LLMs to analyze procedural documents, summarize content and generate coherent slide content based on the information extracted.
- **Implement Prompt Engineering Techniques:** Design and optimize prompts to effectively guide LLM responses for extracting key steps, generating slide titles and formatting content suitable for presentations.

- **Develop document layout analysis skills:** Have tools to analyze and process layout documents, ensuring the identification of different sections.
- **Ensure logical flow and instructional consistency in slides:** Maintain the sequence and clarity of content during the conversion process to ensure instructional flow in the generated slides.
- **Integrate Visual and Structural Elements into Presentations:** Automatically associate diagrams, images or icons with key content sections to improve visual clarity and enhance the user's learning experience.

These objectives collectively contribute to the development of a reliable and efficient application, bridging the gap between static procedural documents and dynamic, presentation ready materials.

3.5.4 Benefits

The development of the project offers numerous benefits across educational, professional and technical domains. These benefits include:

- **Increased Efficiency:** By automating the conversion process, the application significantly reduces the time and effort required to create presentation materials, enabling educators and professionals to focus on content quality rather than manual formatting
- **Improved Consistency and Accuracy:** The application ensures that the logical flow and structure of procedural instructions are preserved, reducing the likelihood of human errors during content transformation
- **Scalability for Future Applications:** While focused on procedural documents, the methodologies developed in this project can be extended to other types of documents, paving the way for broader use cases in education, corporate training and beyond
- **Contribution to Research in Automation and LLMs:** The project advances the field by demonstrating the practical application of LLMs, document layout analysis and text summarization techniques in automating resource creation

These benefits underline the relevance and value of the project, both as a practical tool and as a contribution to ongoing research in educational and technological innovation.

3.5.5 Deliverables

The project aims to produce the following deliverables, covering both technical and academic aspects of the dissertation:

- **Project Plan:** A detailed timeline and task breakdown outlining the project's phases, milestones and critical deliverables to ensure effective planning and execution
- **Dissertation Report:** A comprehensive thesis documenting the research objectives, methodologies, experimental results and conclusions. This report will serve as the main academic output of the project
- **Software Code:** The complete set of code developed during the project, including the application itself and testing scripts. The code needs to be well documented to ensure reproducibility and future development opportunities

- **Testing Reports:** Detailed reports analyzing the performance of the developed application, focusing on key metrics such as accuracy, latency, usability and scalability.
- **Presentation and Discussion:** A final presentation summarizing the project's key findings, methodologies and conclusions

3.5.6 Costs

The implementation of this project incurs no direct financial expenses, as it relies entirely on open-source software and publicly available frameworks. Libraries such as `python-pptx`, `PyPDF2` and the Ollama runtime for local LLM execution are freely available and compatible with the project's requirements.

Nevertheless, there are notable indirect costs, primarily associated with the time and cognitive resources required throughout the development lifecycle. Considerable effort is allocated to phases such as literature review, coding, experimentation and iterative refinement. Implementing, debugging and testing the system also demand a consistent time investment.

An additional salient factor relates to the intrinsic learning trajectory linked with the integration of innovative tools and methodologies. Gaining proficiency in areas such as NLP, document layout analysis and ML especially in the context of working with LLM, requires sustained study and hands on experimentation. These learning efforts, while not monetary, represent a significant component of the overall project cost.

3.5.7 Assumptions and Restrictions

The project relies on specific assumptions to ensure its success while operating under defined restrictions that outline its boundaries and scope.

Assumptions

The following assumptions have been made for the project:

- **Structured Input Documents:** The input documents are assumed to be well structured procedural resources with logical steps and clear instructions, making them suitable for automated processing
- **Reliable Libraries and Tools:** The libraries used in this project such as Ollama for local LLM execution, PyPDF2 for document parsing and python-pptx for slide generation are assumed to be stable, well maintained and capable of supporting the required functionalities.
- **Sufficient Computational Resources:** The hardware and software available are assumed to be capable of running the necessary algorithms efficiently without performance issues.
- **Feasibility of Prompt Engineering with Local LLMs:** It is assumed that the LLMs will respond effectively to prompt engineering techniques and are capable of generating accurate and context aware summarizations and slide content.
- **Time and Skill Development:** It is assumed that the timeline allows enough time to acquire the required skills in LLMs usage, prompt engineering and Python development throughout the project timeline.

Restrictions

The project operates within the following constraints:

- **Input Format Limitations:** The application is restricted to processing structured procedural documents, excluding unstructured content such as handwritten notes or documents with highly complex layouts
- **Language Scope:** The project focuses on documents in a single language, with multilingual processing considered for future extensions
- **Visual Design:** Slide customization is limited to basic formatting, as advanced aesthetic or design elements are not within the scope of this project
- **Time and Resource Constraints:** The project must adhere to the defined timeline and work within the computational resources available, potentially limiting scalability and scope

3.5.8 Risk Management

In the context of developing an application for the automatic generation of presentations from procedural documents, it is essential to anticipate the risks that could compromise the success of the project. For each risk identified, the causes, probability of occurrence, impact on the project and actions to mitigate it were analyzed. Below are the main risks:

Description	Cause	Probability (1-5)	Impact (1-5)	PI Score	Response Description
Delay in delivering initial functionalities	Lack of planning	2	3	6	Monitor progress weekly
Critical bugs during development	Insufficient or careless testing	3	5	15	Perform continuous testing
Incomplete or inconsistent input data	Insufficient or poorly structured data	3	5	15	Receive and validate appropriate datasets
Incomplete documentation at closure	Overload at end of project	3	3	9	Plan documentation review sessions
Lack of efficient integration of visual resources	Limitation of libraries or APIs	3	4	12	Research alternative APIs or adapt manual methods
Failure to customize generated presentations	Algorithms not optimized or simplified	3	4	12	Test and validate customizations with real cases

Table 3.3: Description of the main project risks.

As shown in Table 3.3, the main risks identified include development delays, critical bugs and the inconsistency of input data. These risks were prioritized based on their likelihood, impact and an appropriate response plan was assigned to each one. Effective management of these risks is essential to ensure that the project's objectives are met on time and with the expected quality.

3.6 Work Breakdown Structure

The Work Breakdown Structure (WBS) is a tool that organizes the project into hierarchical components to facilitate management, resource allocation and progress monitoring. The WBS was developed based on the project's general and specific objectives, detailing each stage from planning to final delivery (Siami-Irdemoosa, Dindarloo, and Sharifzadeh, 2015).

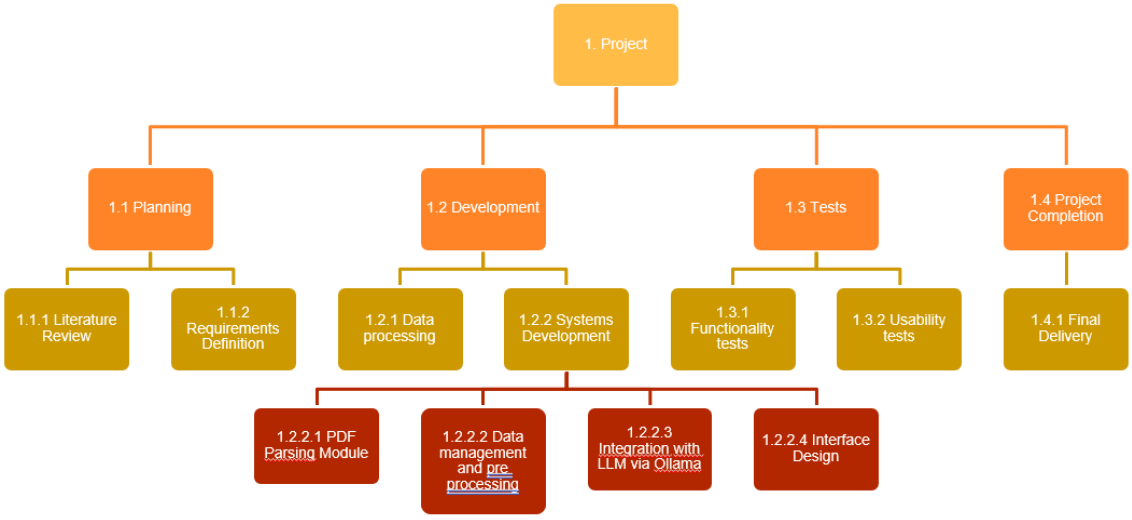

Figure 3.1: WBS Structure

The Figure 3.1 shows the project structure divided into four main phases, planning, development, testing and project closure. Each of these phases has been subdivided into specific tasks, which describe important activities such as defining requirements, developing NLP algorithms, integrating multimodal resources and testing functionality and usability.

This structured approach not only guarantees execution in line with the project's objectives, but also makes it easier to identify potential risks and clearly assign responsibilities.

3.6.1 Dictionary

The dictionary provides a detailed description of each element of the project structure, defining the phases, deliverables and criteria for progress and acceptance. The WBS dictionary presented on table 3.4 ensures clear communication among stakeholders and facilitates project management by specifying responsibilities and performance indicators.

WBS Dictionary	Type	Additional Description	Progress Criteria
1.1 Planning	Phase	Includes the literature review and requirements definition. This phase concludes only after formal approval of its deliverables.	Literature review completed – 25% Requirements – 50% Documentation finalized – 100%
1.2 Development	Phase	Involves data processing, NLP algorithm development, and document-to-slide system implementation.	Data processing completed – 25% NLP algorithm functional – 50% System integration completed – 100%

Continued on next page

WBS Dictionary	Type	Additional Description	Progress Criteria
1.2.1 Data Processing	Work Package	Ensures proper cleaning, structuring, and preparation of the dataset for algorithm development.	Dataset cleaned – 50% Dataset prepared for testing – 100%
1.2.2 Systems Development	Work Package	Development of system components, including PDF parsing, data management, and communication with the LLM model.	Core modules integrated – 50%, Functional interaction with Ollama – 100%
1.2.2.1 PDF Parsing Module	Work Package	Responsible for extracting structured text content from PDFs using PyPDF2.	Successful extraction from various samples – 100%
1.2.2.2 Data Management and Preprocessing	Work Package	Organizes and transforms the extracted content for LLM input, including sentence segmentation and keyword filtering.	Preprocessing pipeline tested – 100%
1.2.2.3 Integration with LLM via Ollama	Work Package	Sends preprocessed data to the local Gemma3 LLM using Ollama and retrieves summarized slide ready content.	Prompts validated – 50%, Model integration tested – 100%
1.2.2.4 Interface Design	Work Package	Creates a simple UI to visualize or export the generated presentations.	UI prototype complete – 50%, Final layout functional – 100%
1.3 Testing	Phase	Encompasses the testing of functionality and usability to validate system performance.	Functionality tests completed – 50% Usability tests completed – 100%
1.3.1 Functionality Tests	Phase	Validates the basic operations of the developed system.	Key functions tested – 50% Functional approval – 100%
1.3.2 Usability Tests	Phase	Evaluates the user experience, focusing on ease of navigation and system intuitiveness.	Usability assessment initiated – 100%
1.4 Project Completion	Deliverable	Finalizes the project with system delivery and associated documentation.	System delivered – 50% Final documentation completed – 100%

Table 3.4: WBS Dictionary

3.7 Timeline

The project timeline was drawn up to ensure organization and compliance with the established stages. Figures 3.2 and 3.3 show the timeline, which covers from the initial planning to the end of the project, highlighting the main phases and the estimated periods for each activity.

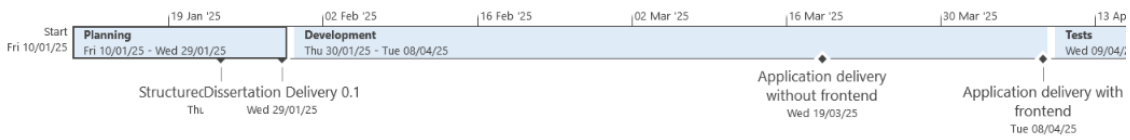

Figure 3.2: Part 1 of the timeline

The development of the project is structured in four main stages, but Figure 3.2 shows the first two stages:

1. **Initial Planning:** Completion of the "Ollama Course Build AI Apps Locally", & "Prompt Engineering Tutorial Master ChatGPT and LLM Responses" and "Machine Learning for Everybody – Full Course" by *freeCodeCamp.org* courses and definition of requirements
2. **Development:** Data pre-processing, development of NLP algorithms, adaptation of document methods for slides, integration of modal resources and development of the interface

Figure 3.3: Part 2 of the timeline

Figure 3.3 shows the last two stages of the schedule:

1. **Tests:** Functionality and usability tests
2. **Project closure:** Handover to supervisor and final delivery

Each stage includes the delivery of intermediate versions of the dissertation to the supervisor, which allows for continuous revisions and adjustments throughout the development of the project.

3.8 Time

The project schedule is defined by milestones each corresponding to a specific deadline for its phase.

3.8.1 Milestones

- Structured Requirements - 23/01/2025
- Dissertation Delivery 0.1 - 29/01/2025
- Application Delivery without frontend - 19/03/2025
- Application Delivery with frontend - 08/04/2025
- Dissertation Delivery 0.2 - 08/04/2025
- Tests Registration - 23/04/2025
- Dissertation Delivery 0.3 - 23/04/2025
- Dissertation Delivery 0.4 - 12/06/2025
- Dissertation Delivery 1.0 - 23/06/2025

Chapter 4

Literature Review

Document analysis is one of the fields of computer vision, whose specialty is the identification and classification of objects. The aim of this dissertation will be to develop an application that can use procedural documents to generate presentations semi-automatically. In this way, the various techniques and technologies that are relevant to the process and that are directly related to the topics involved in the context of the current problem will be presented.

4.1 Data Sources

Data sources play a critical role in research, serving as the foundation for answering research questions and achieving research objectives by providing access to indexed literature.

The digital libraries present in table 4.1 offer access to a wide variety of indexed resources, including articles, articles and books, all of which are peer reviewed.

Database	**URL**
Google	`https://scholar.google.com/`
ACM	`https://dl.acm.org/`

Table 4.1: Data Sources

4.2 Search Terms

A systematic search query was developed to explore literature relevant to the automation of procedural document to slide conversion and its associated methods with the next keywords:

- Document to slide
- Document layout analysis
- Automatic presentations creation
- Tools to summarize
- Summarization
- Artificial Inteligence

The search query incorporates essential terms related to summarization and AI to ensure a broad yet focused exploration of the subject.

4.2.1 Search Query

To gather relevant academic sources in a structured and focused manner, the following query was designed:

```
("document to slide generation" OR "automatic presentation creation"
 OR "document layout analysis" OR "tools to summarize") AND "
 summarization" AND ("ai")
```

Listing 4.1: Search query developed with the search terms

This query was created to align with the dissertation's research goals, ensuring that selected literature would address both theoretical and practical aspects of automating content transformation. The use of logical operators such as `AND` and `OR` made it possible to include a wide range of topics without losing specificity.

By combining terms related to summarization, slide generation and core AI technologies, the query helps capture studies that describe not only conceptual frameworks but also real world applications and experimental setups. As such, it supports the development of a technically grounded and practically relevant solution.

4.3 Eligibility Criteria

To rigorously ascertain the pertinence and scholarly caliber of the literature scrutinized within this dissertation, explicit inclusionary and exclusionary criteria were meticulously delineated. These criteria were designed to focus on studies and resources directly related to the automation of procedural document to slide conversion and its associated technologies.

4.3.1 Inclusion Criteria (IC)

- **IC1:** Publications that present techniques or methodologies specifically targeting automatic text summarization.
- **IC2:** Studies addressing the generation of presentation slides from textual content or exploring similar automated transformation tasks.
- **IC3:** Research involving the application, evaluation, or benchmarking of NLP, ML, or DL tools in the context of document processing or content generation.

4.3.2 Exclusion Criteria (EC)

- **EC1:** Studies that focuses exclusively on manual slide design or visual storytelling without using any type of automation.
- **EC2:** Studies lacking evaluation or comprehensive methodological detail concerning automated document analysis or content structuring relevant to this dissertation's scope.

4.4 Data Collection Processing

The process of selecting the relevant articles was conducted following the PRISMA systematic review methodology (Page et al., 2021). This methodology ensures a transparent and

reproducible approach for filtering and analyzing the literature. The procedure comprised three main stages and is summarized in Figure 4.1.

In the **identification** phase, a total of 362 articles were collected from Google Scholar and Association for Computing Machinery. From this initial pool, 212 records were excluded due to duplication or irrelevance, resulting in 150 records that were considered for the next stage. During the **screening** phase, the articles were assessed against the predefined eligibility criteria, which led to the exclusion of 113 items that did not meet the required conditions. This refinement reduced the set to 37 articles that were subjected to a more detailed review. In the final **inclusion** phase, these 37 articles were carefully examined to evaluate their alignment with the research objectives. Out of these, 32 were excluded for irrelevance and only 5 articles were ultimately selected for the study. These five articles constitute the basis for answering the research questions and are presented in Table 4.2.

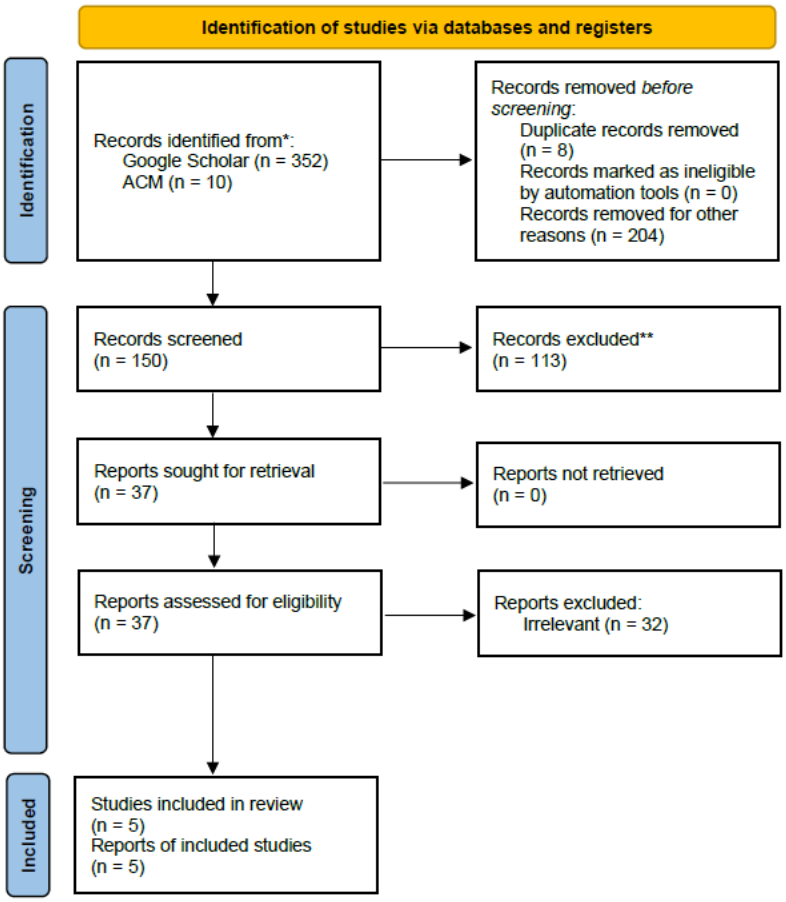

Figure 4.1: PRISMA flow diagram of the article selection process

ID	Title	Reference
1	D2S: Document-to-Slide Generation Via Query-Based Text Summarization	Sun et al. (2021)
2	DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents	Fu et al. (2022)
3	Automatic Extraction of Slides from Scientific Papers	Du (2021)
4	Unsupervised Paper2Slides Generation	Lu (2023)
5	Presentations by the Humans and For the Humans: Harnessing LLMs for Generating Persona-Aware Slides from Documents	Mondal et al. (2024)

Table 4.2: Final set of included articles after PRISMA filtering

All five selected studies contributed insights into **RQ1**, presenting strategies for automating the conversion of procedural or scientific documents into slides while aiming to preserve content integrity, logical sequencing and overall clarity. Concerning **RQ2**, each of the works explored techniques for extracting and structuring procedural or textual content, although the methods varied in complexity and scope.

From the comparative analysis of these contributions, it became evident that approaches based on LLMs provide a more direct and scalable solution to the challenges of content extraction and structuring. Their capacity to adapt across different domains and to handle unstructured input with reduced pre processing requirements makes them particularly suitable for the goals of this project. Following this section, the processes and tools presented in these studies are examined in greater depth to identify their practical contributions to the present work.

4.5 Natural Language Processing

NLP is a subfield of AI that focuses on the interaction between computers and human language, enabling machines to understand, interpret and generate text in a human like way. This area integrates computational linguistics, ML and DL techniques to analyze vast sets of textual data and extract relevant knowledge (Chopra, Prashar, and Sain, 2013).

NLP has been one of the pillars for advances in text automation tasks, including information extraction, summarization, machine translation, sentiment analysis and text generation (Liddy, 2001). These capabilities are particularly relevant to the aim of this dissertation, which involves the automatic transformation of textual documents into educational presentations, maintaining the logical sequence and organizing the content for pedagogical purposes.

4.5.1 Applications relevant to the dissertation

The applications of NLP are vast and highly relevant to the topic of automation in the creation of educational resources. In particular:

- **Slide Generation:** Work such as that by Sun et al., 2021 has already explored generic methods for converting text to slides, serving as a basis for adaptations in procedural materials.

- **Intelligent Education:** NLP has been used in intelligent tutorial systems to adapt content according to the student's profile.
- **Content Organization Aid:** The ability to summarize and structure complex texts is a solution for optimizing the process of creating teaching materials (Liddy, 2001).

4.6 Document Layout Analysis

Document layout analysis is an essential component of any system that aims to transform text based resources into structured and visual formats. The organization of the document relates the relational dynamics between sections, titles, text blocks and embedded graphic elements. Checking these structural parts is important to preserve the integrity of the sentences (Binmakhashen and Mahmoud, 2019).

This structural mapping is key to supporting logical flow, prioritization of content and consistency in the visual representation of procedural documents (Namboodiri and Jain, 1998).

4.6.1 Pre-processing

Pre-processing plays a foundational role in the document analysis workflow, ensuring that the input data is clean, consistent and ready for subsequent stages. It focuses on transforming raw documents often heterogeneous in format and quality into a standardized and structured form suitable for processing. This step directly influences the reliability and accuracy of downstream tasks such as layout analysis, summarization and slide generation (Binmakhashen and Mahmoud, 2019).

Key techniques used during pre-processing include:

- **Format Standardization:** Ensures all input documents follow a uniform format, making the pipeline consistent regardless of the source material.
- **Text conversion:** Extracts readable text to machines from Portable Document Format (PDF) and image content.
- **Removal of Visual Artifacts:** Elements such as borders, watermarks or background noise are removed to reduce interference with later stages of analysis (Binmakhashen and Mahmoud, 2019).

4.6.2 Noise reduction

Scanned documents can contain unwanted "noise" such as stains, marks or redundant elements, which can impair the analysis (Namboodiri and Jain, 1998). The following techniques were used to mitigate these effects:

- **Visual filters:** Application of filters such as median or Gaussian to smooth images and remove noise.
- **Background cleaning:** Removing complex backgrounds or textures that can interfere with text detection.
- **Border detection:** Elimination of unnecessary lines or borders to focus only on the central content of the document (Namboodiri and Jain, 1998).

4.6.3 Skew Detection and Correction

Scanned procedural documents frequently exhibit alignment inconsistencies, resulting in skewed or slanted text lines. Addressing skew is a foundational task in the pre-processing pipeline, particularly in systems that rely on precise segmentation and structural integrity, because can compromise the accuracy of text recognition algorithms and obstruct effective layout analysis (Binmakhashen and Mahmoud, 2019).

- **Skew Detection:** To estimate the degree of text misalignment, algorithms based on horizontal projection profiles or the Hough transform are typically employed. These techniques analyze the orientation of text lines and compute the approximate angle of rotation.
- **Automatic Correction:** After the skew angle is identified, the document image is rotated to realign the text horizontally. This correction enhances the legibility of the text and supports the subsequent extraction and classification tasks with greater precision (Binmakhashen and Mahmoud, 2019).

4.7 Slide Generation

Creating slides from text involves more than simply reducing content length. This process necessitates a structured reorganization, greater visual clarity and well defined emphasis, particularly when transposing procedural documentation into didactic slides. Its significance is notably amplified within pedagogical environments, where information must be rendered both readily assimilable and pedagogically congruent (Sefid et al., 2019).

4.7.1 Sentence Ranking and Selection

The identification of which sentences warrant inclusion within a presentation constitutes one of the main challenges in the domain of automated slide generation. Sentence importance can be ranked using ML based approaches, including supervised methods such as Support Vector Regressor (SVR) or more complex DL models trained to recognize salient information patterns within a given context.

In some systems, this ranking process is further enhanced using Integer Linear Programming (ILP) a mathematical optimization strategy that applies constraints to ensure quality in the output. The resulting output is a curated set of short, informative sentences optimized for presentation format (Sefid et al., 2019).

4.7.2 Bullet Point Generation

Bullet points constitute a pivotal characteristic of slide based presentations, their primary function being to disseminate essential information in a succinct and systematically structured fashion. In the context of this system, a two tiered bullet structure is adopted to improve clarity and facilitate comprehension (Sefid et al., 2019).

At the top level, bullet points are built from noun phrases extracted from content previously selected as relevant. These noun phrases are filtered using specific rules, they must include more than one word, be relatively uncommon in the overall document corpus and not exceed a limit of ten tokens. Such filtering helps eliminate overly generic phrases and ensures that each top level point remains specific and informative (Sefid et al., 2019).

4.7.3 Slide Organization and Titles

Organizing the content into slides involves adhering to length constraints and assigning meaningful titles to each slide (Sefid et al., 2019).

To maintain readability and visual clarity each slide contains a maximum of four sentences and typically includes five bullet points. If a topic contains more than four sentences, the content is split across multiple slides. Titles are generated based on the section headings from which the first sentence of the slide is extracted. These titles are truncated to the first five tokens to ensure conciseness while retaining relevance to the slide's content (Sefid et al., 2019).

4.7.4 Challenges and Considerations

Even with the progress made in the automatic creation of presentations, their practical implementation is hampered by some factors:

- **Coherence and Flow:** The sequence of slides must reflect the structure and logic of the original document. A lack of cohesion between sections can reduce the clarity of the presentation and compromise its effectiveness.
- **Content density:** Achieving a balance between content and aesthetic simplicity remains a challenge in design.
- **Adaptability:** Automatic slides are made to be changed, so needs to be flexible in the organization of the content and the presentation style. (Sefid et al., 2019).

4.8 Large Language Models

LLMs enable complex tasks such as summarization, classification and generation of coherent, contextually relevant text. LLMs are pre trained on large datasets and then fine tuned for specific downstream tasks. Their capabilities include, but are not limited to, text summarization, question answering, sentiment analysis and document classification. In the context of this project, LLMs are particularly relevant for transforming procedural text into structured presentations, extracting key information and organizing content coherently. (Zhao et al., 2024)

4.8.1 Gemma 3, LLaMA 3, LLaMA 3.2 and DeepSeek-R1

Gemma 3 and LLaMA 3.2 are recent open models known for their strong performance on a variety of language tasks. Both leverage the transformer architecture and support efficient inference on consumer grade hardware (Zhao et al., 2024).

- **Gemma 3** offers competitive results in summarization and classification tasks while being optimized for smaller deployments. Its balance between performance and efficiency makes it suitable for integration into educational tools. (Team et al., 2025)
- **LLaMA 3** provides state of the art performance across multiple NLP benchmarks. It delivers robust results in summarization and classification tasks. (Grattafiori et al., 2024)

- **LLaMA 3.2** focuses on scalability and adaptability, offering versions with different parameter sizes. It proves to be ideal for summarizing documents and organizing content with minimal supervision. (Gonçalves et al., 2025).
- **DeepSeek-R1** is designed for tasks that require complex reasoning and deep understanding of long contexts. The architecture is optimized for structured knowledge extraction and logical sequencing. (DeepSeek-AI et al., 2025).

4.9 Relevant Tools and Frameworks

In the context of this project, several tools and frameworks play a main role in implementing NLP tasks, document parsing and slide generation. These tools include libraries for PDF processing, integration with LLMs and presentation creation, which together support the pipeline from document input to output slide decks.

4.9.1 Flask

Flask is a lightweight yet powerful web framework that enables the development of dynamic user facing applications. In this project, Flask underpins the web interface that allows users to upload procedural documents and retrieve the resulting slide presentations.

Its role extends to managing the entire lifecycle of user interaction, from secure file uploads and backend processing to serving the final downloadable output. Additionally, Flask provides the routing infrastructure that connects user requests with internal components, enabling a seamless integration between front-end interactions and back-end logic (Mufid, Basofi, and Rasyid, 2019).

Flask vs Django

While Flask was chosen as the foundation for this project, it is important to compare it with Django, another widely used Python web framework. Django is a high level framework that comes with many built in features such as authentication, an Object-Relational Mapping (ORM) and an admin interface (Ghimiri, 2020).

Aspect	**Flask**	**Django**
Framework Philosophy	Minimalist, provides only core features	Comes with many built-in tools
Flexibility	Highly flexible, lets developers choose external libraries	Opinionated, enforces structured development
Complexity	Lightweight, easy to learn and use for small/medium projects	Heavier, designed for large scale and complex applications
Scalability	Suitable for modular proccesses and microservices	Excellent for monolithic, enterprise level solutions
Use Case Fit	Ideal for fast prototyping and integration focused solutions	Strong choice for data-heavy platforms with authentication, admin panels, etc.

Table 4.3: Comparison between Flask and Django

As shown in Table 4.3, both frameworks present advantages depending on the project requirements. Django is better suited for complex, large scale applications that benefit from its built-in authentication, ORM and administration features (Ghimiri, 2020). However, these capabilities introduce additional overhead and reduce flexibility. In contrast the Flask's minimalist design and modularity align better with the needs of this project, which required a lightweight and integration oriented framework to connect document processing components and LLM based summarization services. Its simplicity facilitated fast prototyping, clear separation of responsibilities and easier customization of the workflow, making it the most appropriate choice for the proposed system.

4.9.2 PyPDF2

PyPDF2 functions as a remarkably versatile Python library, engineered for the comprehensive elucidation and sophisticated manipulation of PDF files. Its broad array of capabilities encompasses the precise extraction of textual elements, enabling the retrieval of document information such as author or title and the elaborate re-organization of document constituents (Álvarez et al., 2023). In this specific project, PyPDF2 plays a key role, serving as the basis for retrieving textual content from procedural documents.

4.9.3 Fitz (PyMuPDF)

Fitz, also referred to as PyMuPDF, is a Python interface to the lightweight document renderer PDF MyPDF. In the context of this project, Fitz is strategically utilized for its sophisticated functionalities in visual content extraction and its discerning capabilities in layout sensitive document parsing.

A particularly salient attribute of this library is its capacity to directly extract embedded images from individual pages. This capability is unequivocally essential for precisely aligning visual constituents with their corresponding textual information throughout the process of automated slide creation. Furthermore, Fitz affords granular positional metadata per textual component, thus enabling layout conscious processing an important factor for scrupulously upholding the structural coherence of the source document upon its rendition into a visual medium.

In addition to its precision, Fitz offers excellent performance when handling large or complex documents. Its optimized architecture ensures efficient parsing without compromising accuracy, making it highly suitable for real time or batch oriented document transformation workflows (Yang and Yu, 2025).

4.9.4 Ollama

Ollama is a lightweight framework designed to enable seamless execution of LLMs on local machines. Within this project, Ollama serves as the bridge between the application and language models, facilitating tasks such as summarization, structural analysis and slide content generation.

Ollama's key advantage is its support for local inference. This means all LLM interactions happen within the user's system, protecting data privacy by eliminating the need to send sensitive documents over external networks. The Ollama's friendly Application Programming Interface (API) also makes it easy to submit prompts and handle responses.

Its prompt engineering flexibility allows fine tuned control over the model's behavior. Developers can construct highly targeted instructions that shape the structure and style of the model's output an essential feature for generating educational content with consistent tone and logical flow (Ollama, 2025).

Advantages for Educational Automation

Using LLMs via Ollama enhances the system's ability to:

- Identify and summarize key procedural steps accurately.
- Categorize content based on instructional goals.
- Generate structured and logically sequenced presentations. (Ollama, 2025)

These capabilities align closely with the goals of automating the creation of procedural educational resources, making LLMs a critical component in the proposed architecture.

4.9.5 python-pptx

Python-PPTX emerges as a specialized Python library engineered for the programmatic generation and comprehensive modification of Microsoft PowerPoint presentations. Within the purview of this project, it assumes a pivotal role by facilitating the automated creation of slides, thereby obviating the necessity for manual formatting and layout adjustments. This library accommodates the dynamic insertion of structured constituents, including, but not limited to titles, textual containers, graphical forms and embedded imagery, thereby affording exhaustive command over slide composition.

Customization is also possible, allowing to adjust fonts and align text elements. The integration of `python-pptx` into the project allows for the modification of structured content into coherent slides in an automated and efficient manner (Canny, 2023).

4.9.6 Pillow (PIL)

The Python Imaging Library (PIL) is employed in this system to process and refine images extracted from PDF documents. As visual content often requires adaptation before being used in slides, Pillow is responsible for preparing these assets for integration into the presentation.

Key operations include resizing images to fit predefined layouts, cropping to remove unnecessary elements and converting image formats to ensure compatibility with `python-pptx`. The implementation of preliminary processing is important to maintain clarity and consistency in the generated slides, contributing to the production of a consistent result. The use of PIL is important for managing visual data in the automatic flow of generating presentations (Clark, 2025).

4.9.7 Conclusion

The combined use of these libraries enables the end-to-end automation of the document to presentation pipeline. Each tool addresses a specific aspect of the workflow from raw data extraction and image preprocessing to structured slide generation and web based user interaction. Together, they form a cohesive foundation for a scalable and user centric solution.

4.10 Existing Tools

Various tools exist for automating the creation of presentations from text-based content. This section discusses SlideSpeak, MagicSlides and Wondershare PDFelement tools, evaluating their functionalities and limitations in the context of automated content generation.

4.10.1 SlideSpeak

SlideSpeak is an AI-assisted application designed to convert textual input into presentation slides automatically. Its primary strength lies in its rapid generation capabilities, which cater to users seeking efficiency over manual slide creation. However, its output is generally constrained by limited customization options and does not effectively accommodate complex procedural structures or domain specific instructional flows. Consequently, SlideSpeak is better suited for general purpose summarization rather than detailed procedural content delivery (SlideSpeak, 2025).

4.10.2 MagicSlides

MagicSlides is a Google Slides extension that enables the transformation of document based content into presentations through AI-driven summarization. Integrated seamlessly within the Google Workspace environment, it offers convenience for users already embedded in the Google ecosystem. Despite its accessibility and ease of use, the tool exhibits similar limitations in managing logically ordered instructional materials or adapting to nuanced pedagogical structures (MagicSlides, 2025).

4.10.3 Wondershare PDFelement

Wondershare PDFelement is a multi functional PDF management solution that facilitates content extraction and conversion into alternate formats, including PowerPoint presentations. While it provides robust features for handling document layouts and structure, its capabilities are largely oriented toward file format transformation rather than semantic understanding. As such, its generated presentations may lack the instructional clarity and coherence required for procedural education (Wondershare, 2025).

4.10.4 Comparison and Limitations

Although these commercial tools offer forms of automation in slide generation, they fall short in several areas critical for instructional effectiveness. Specifically, they do not ensure:

- The preservation of procedural logic and sequencing;
- The contextual integration of visual elements;
- The pedagogical structuring necessary for coherent knowledge delivery.

In contrast, the proposed system focuses explicitly on addressing these gaps by leveraging advanced LLM driven prompts, structured data extraction and multimodal slide generation. It is tailored to produce semantically aligned, logically structured and visually supported slides that enhance both instructional quality and user engagement in educational settings.

4.11 Ethical Considerations

The development of automated content creation solutions requires an analysis of their ethical implications so it is important that the system is aligned with privacy and accountability in AI applications.

4.11.1 Bias Mitigation and Inclusivity

To minimize unintentional misrepresentations in generated content is important that the data used reflects several language styles, cultural backgrounds and user needs. Therefore, the system must integrate validation or review standards with human intervention to identify and correct any results that may disclose incorrect information. Furthermore, special attention must be given to inclusive design to ensure the accessibility of content across varied user demographics.

Procedural documents often contain confidential or personally identifiable information. To address this, the system must enforce strict data privacy protocols. This includes:

- Ensuring temporary storage of input files with automatic deletion after processing;
- Complying with regulatory frameworks such as General Data Protection Regulation (GDPR);
- Preventing the retention of any form of Personally Identifiable Information (PII) beyond its immediate processing needs.

The integration of robust anonymization methodologies and stringent secure handling protocols is imperative within the architectural design of the tool, serving to effectively ameliorate potential hazards pertaining to data exfiltration or illicit access.

AI automation can simplify the creation of educational materials, but the accuracy, impartiality and appropriateness of the content are ultimately the responsibility of the user.

Although AI offers considerable advantages in automating workflows, its deployment must be accompanied by a strong ethical framework. Addressing concerns related to bias, privacy and human oversight not only safeguards user trust but also ensures that the technology contributes positively and responsibly to the learning ecosystem.

Chapter 5

Design and Development

This chapter presents the design and development of the system, detailing the architectural choices and implementation strategies. It describes how the requirements identified in previous chapters were translated into a functional prototype, integrating NLP, document layout analysis and slide generation techniques. Furthermore, the chapter includes the testing phase, outlining the methods used to validate the system's functionality, performance and usability.

5.1 Requirements

The requirements for the development of the system were organized into functional and non-functional categories to ensure project's objectives and user needs.

5.1.1 Functional Requirements

The system must provide the following functionalities:

Feature	**Description**
Procedural Document Import	Supports importing documents in formats such as PDF.
Content Extraction	Automatically identifies and extracts procedural steps from documents. Performs summarization to generate compact and structured content.
Slide Generation	Constructs presentation slides from the extracted content. Ensures preservation of logical sequence across slides. Integrates relevant visual elements, such as images or icons, linked to key topics.
Slide Customization	Allows users to choose the presentation theme from 3 available themes
Presentation Export	Exports finalized presentations into widely used formats such as PowerPoint Open XML Presentation (PPTX).

Table 5.1: Functional requirements of the proposed system

5.1.2 Non-Functional Requirements

The system must meet the following quality attributes:

Category	Description
Performance	Response Time: The system should respond to user actions within 60 seconds.
System Reliability	Availability: The system should be available 95% of the time. Error Handling: The system should handle errors and provide error messages to users.
Content Reliability	Ensure that the generated slides consistently maintain logical coherence and integrity of the extracted content. Reduce potential inaccuracies during text processing and slide creation.
Usability	User Interface: The graphical user interface should be intuitive and user-friendly, with clear workflows from document ingestion to final presentation generation.
Maintainability	Develop a modular and well-documented codebase to support future improvements and refinements.
Security	Safeguard the confidentiality and security of all user-submitted documents. Implement secure protocols for temporary data retention and ensure complete file deletion after processing.

Table 5.2: Non-functional requirements of the system

5.2 Solution Architecture

The architecture of the proposed solution is designed is organized into four main layers: Web Application, Extraction, Processing and Generation, as shown in Figure 5.1.

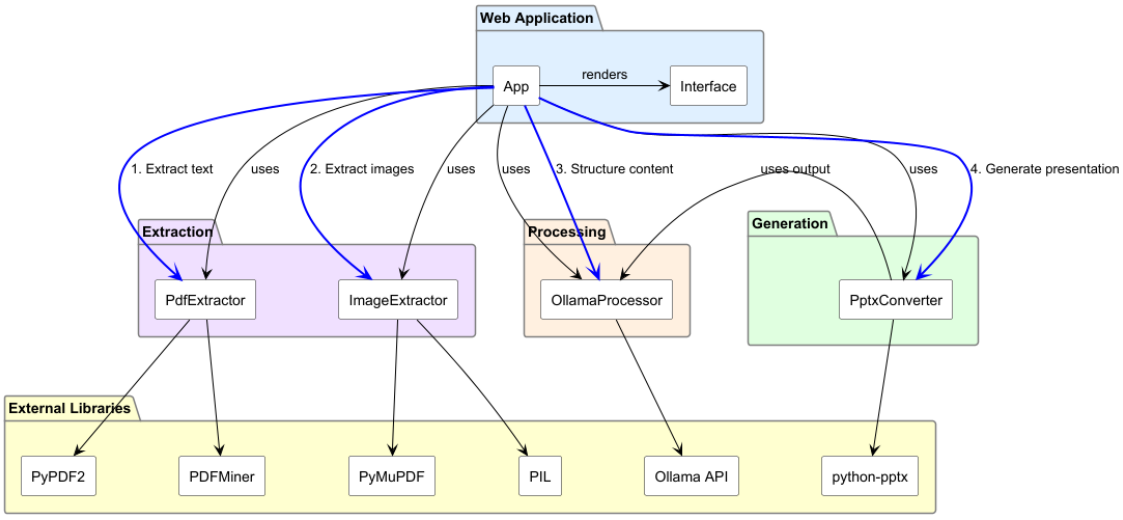

Figure 5.1: Solution Architecture Diagram

5.2.1 Web Application Layer

The **Web Application** layer is represented by a simple front-end interface in HyperText Markup Language (HTML) that allows users to upload documents in PDF format, choose the PPTX theme and the LLM to use. After submission, the document is processed and, when completed, the generated presentation is downloaded.

The `App` component acts as the backend controller and orchestrates the entire workflow. It manages communication between components, processes user input/output and invokes the modules responsible for text extraction, image extraction, content processing and slide generation.

5.2.2 Extraction Layer

This layer is responsible for extracting content from the uploaded PDF document:

- `PdfExtractor`: Extracts raw textual content from PDF pages using libraries such as **PyPDF2** and **PDFMiner**, which are ideal for handling structural variations across documents.
- `ImageExtractor`: Retrieves embedded images from the PDF using **PyMuPDF (fitz)** and processes them using **Pillow (PIL)** to prepare them for slide integration.

5.2.3 Processing Layer

The extracted text is passed to the `OllamaProcessor` module, which processes and structures the information. This includes:

- Text cleaning and segmentation into key sentences or steps
- Keyword and title extraction
- Communication with the local language model using **Ollama**'s LLMs to summarize content and generate bullet points for slides

The results from this module include summarized sections, categorized topics, and recommended visual elements.

5.2.4 Generation Layer

The `PptxConverter` module takes the structured and summarized content and uses the **python-pptx** library to programmatically generate the slide presentation. It formats the content into appropriate layouts, integrates images and styles the slides for visual clarity.

5.2.5 Workflow Summary

The entire process follows these key steps:

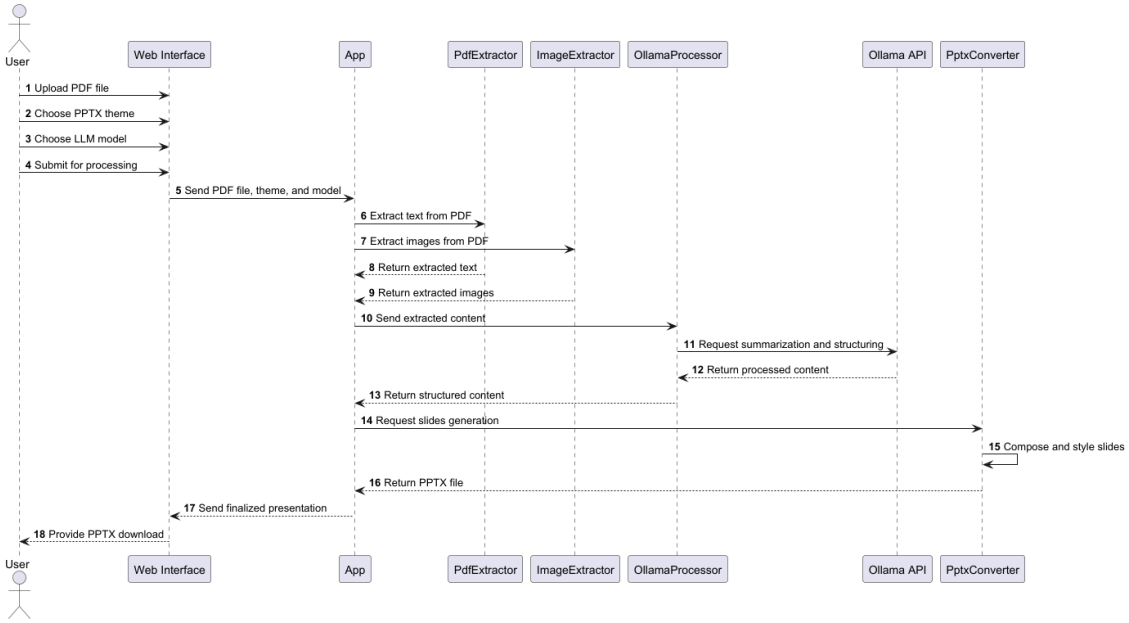

Figure 5.2: Workflow Summary

The workflow of the proposed system integrates multiple components in a sequential and coordinated manner, enabling the automated conversion of procedural documents into presentation slides. Figure 5.2 illustrates this process through a sequence diagram that highlights both user interactions and system operations.

The workflow begins with the user, who uploads a procedural document in PDF format, selects a slide theme and chooses the LLM to be used. Once these parameters are submitted, the web interface transmits the inputs to the application backend. At this stage, the document is processed by specialized modules: the `PdfExtractor` retrieves textual data, while the `ImageExtractor` isolates relevant visual elements. These outputs are then combined and passed to the `OllamaProcessor`, which communicates with the `Ollama API` to perform summarization and restructuring of the content. The processed information, now logically organized, is returned to the application.

Subsequently, the `PptxConverter` module is invoked to transform the structured content into presentation slides. This stage ensures that extracted text and images are integrated consistently with the selected theme, maintaining both coherence and visual clarity. The finalized presentation is then transmitted back to the web interface, where its downloaded the final `.pptx` file.

5.3 Implementation Details

This section explores the implementation details behind the core components of the project. While the architecture section offers a high level overview of the system's design, this section presents a depth analysis of each module's internal logic, focusing on how procedural documents are processed, summarized and transformed into presentation slides. The entire implementation is available for analysis in (Pacheco, 2025).

5.3.1 Extraction

The extraction phase constitutes the first step of the pipeline, responsible for retrieving both textual and visual content from procedural documents. As illustrated in Figure 5.3, this stage involves coordinated interactions between multiple components and libraries to ensure accuracy and robustness.

As illustrated in Figure 5.3, when a PDF document is uploaded, the application first stores it temporarily in the file system. The `PdfExtractor` component then initiates text extraction using `PyPDF2` and `PDFMiner`. Both libraries analyze the document independently, each returning a text representation. The system compares the two outputs and selects the version with more content.

After the text extraction, the `ImageExtractor` module employs `PyMuPDF` to retrieve images embedded within the PDF. The process iterates through all detected objects, extracting raw binary image data, saving them temporarily and filtering them by minimum size to discard irrelevant or noisy elements (e.g., icons or artifacts). The result of this phase is a structured set of text and image metadata, which is then returned to the application and prepared for subsequent processing.

This dual strategy leveraging multiple libraries for text extraction and applying selective filtering for images, ensures that the system maintains both completeness and quality in the extracted content, which is fundamental for accurate summarization and slide generation.

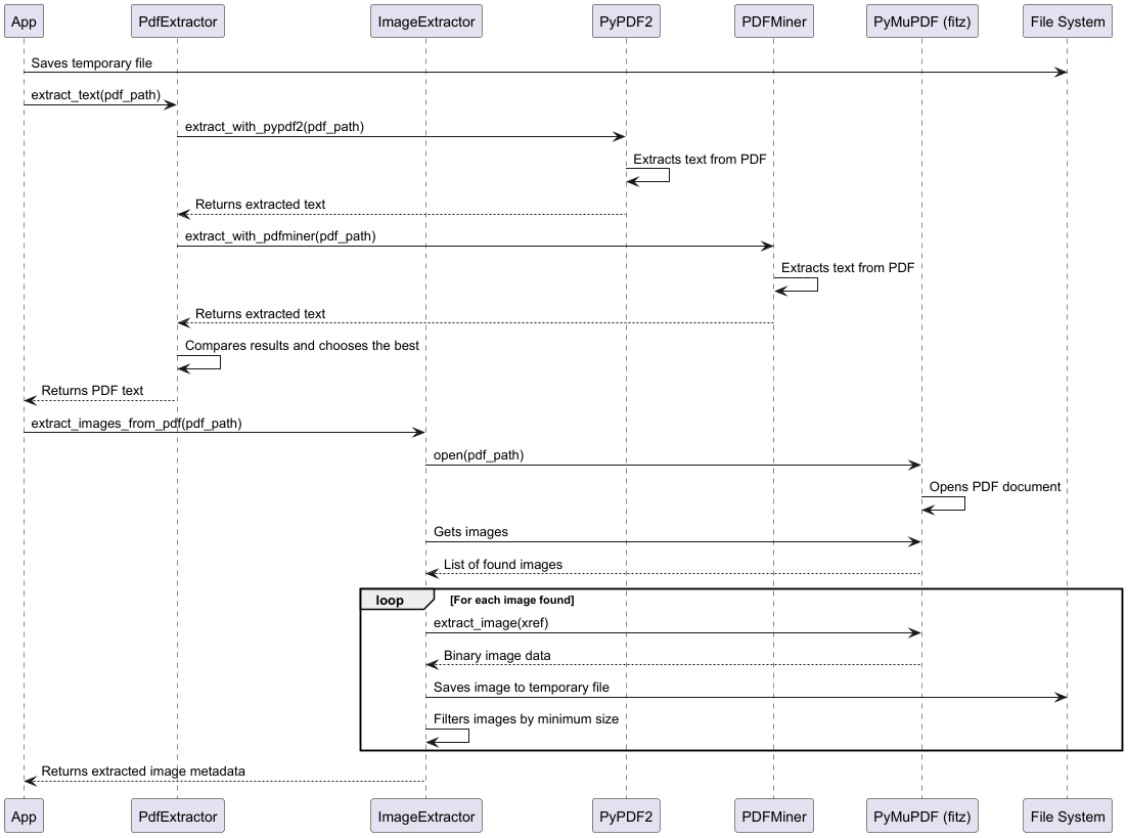

Figure 5.3: Sequence diagram of the extraction phase

5.3.2 Processing

The processing phase is responsible for transforming the raw extracted data into a structured and semantically meaningful representation, ready for slide generation. This step leverages LLMs via the Ollama framework to clean, normalize and organize the extracted content. Figure 5.4 illustrates the workflow of this stage.

The process begins with the `OllamaProcessor`, which submits the extracted text to the `Ollama API`. The model performs text cleaning and restructuring, returning a coherent representation of the input. This ensures that fragmented sentences or irregular formatting from the extraction phase are resolved into well formed text.

Subsequently, the document undergoes structural analysis, where the LLM identifies the logical hierarchy of content, such as titles, step by step instructions and conclusions. The results are returned in a structured JavaScript Object Notation (JSON) format, providing a readable outline of the procedural flow. To complement this, an additional analysis is carried out that examines the relationship between text and images. The model is prompted to associate visual elements with the most relevant textual segments, thus enriching the instructional quality of the material.

Finally, the system consolidates both text and image metadata into a unified document structure, which is normalized to ensure consistency across different documents. The outcome of this stage is a complete, logically ordered representation of the procedural resource, integrating both semantic content and visual cues for subsequent slide generation.

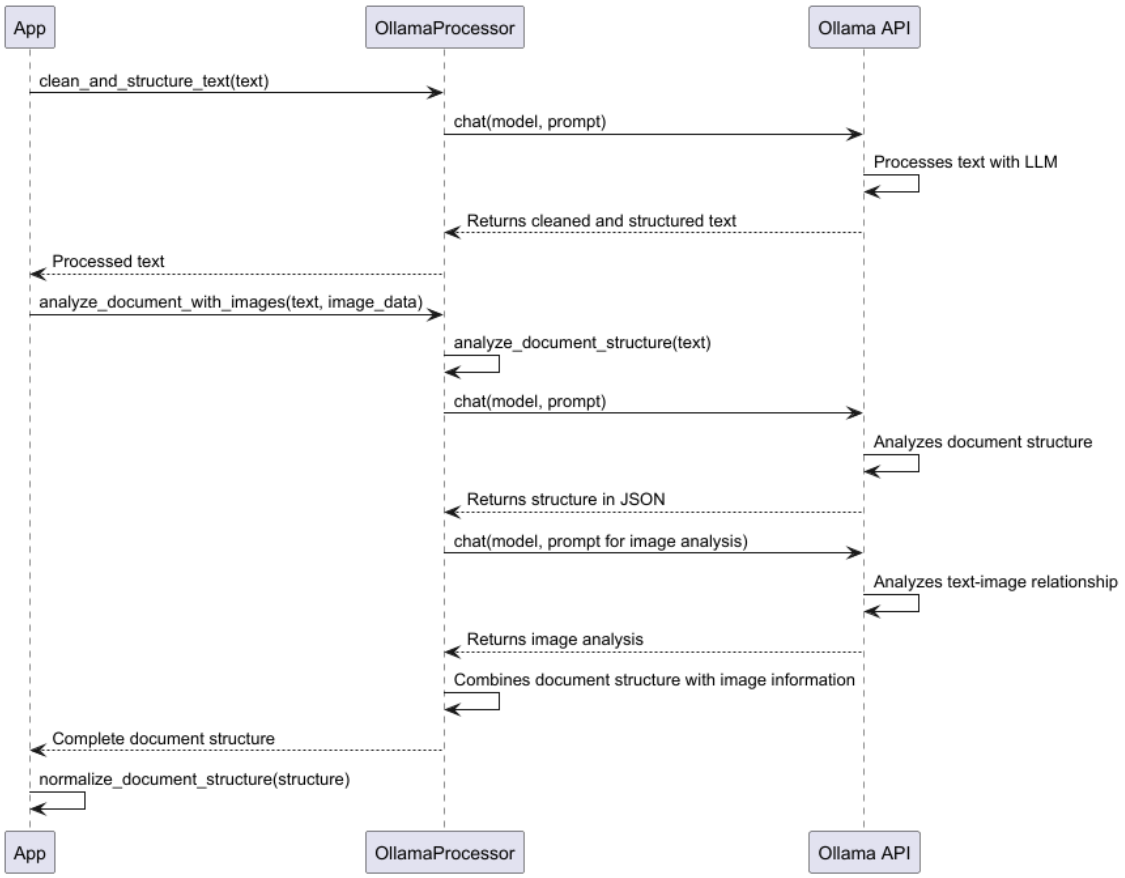

Figure 5.4: Sequence diagram of the processing phase

5.3.3 Generation

The generation phase is responsible for converting the structured document representation into a coherent and visually consistent presentation. This stage relies on the `PdfToPptxConverter` component, which maps the hierarchical structure of the processed content into slide elements.

As illustrated in Figure 5.5, the process begins with the creation of a title slide, where the system assigns the document's main title and optional subtitle. Subsequently, the module iterates over each section of the structured content, dynamically determining the appropriate slide type based on the nature of the information. For example, procedural instructions are rendered as content slides, tabular information is allocated to table slides and visual data is paired with illustrative images when available. Two main pathways exist within this process. If a section is associated with visual assets, a combined layout is generated that integrates textual instructions alongside the relevant image, thereby enhancing the communicative value of the slide. Conversely, if no image is linked, the system generates a clean content slide containing only the title and corresponding textual elements. This branching logic ensures that each slide remains both concise and contextually appropriate.

Once all sections have been processed, the presentation is compiled into a PPTX file and stored temporarily in the file system. The application then retrieves the finalized presentation and delivers it back to the user interface. As a final step, temporary resources are deleted to ensure secure handling of user data and optimal system performance.

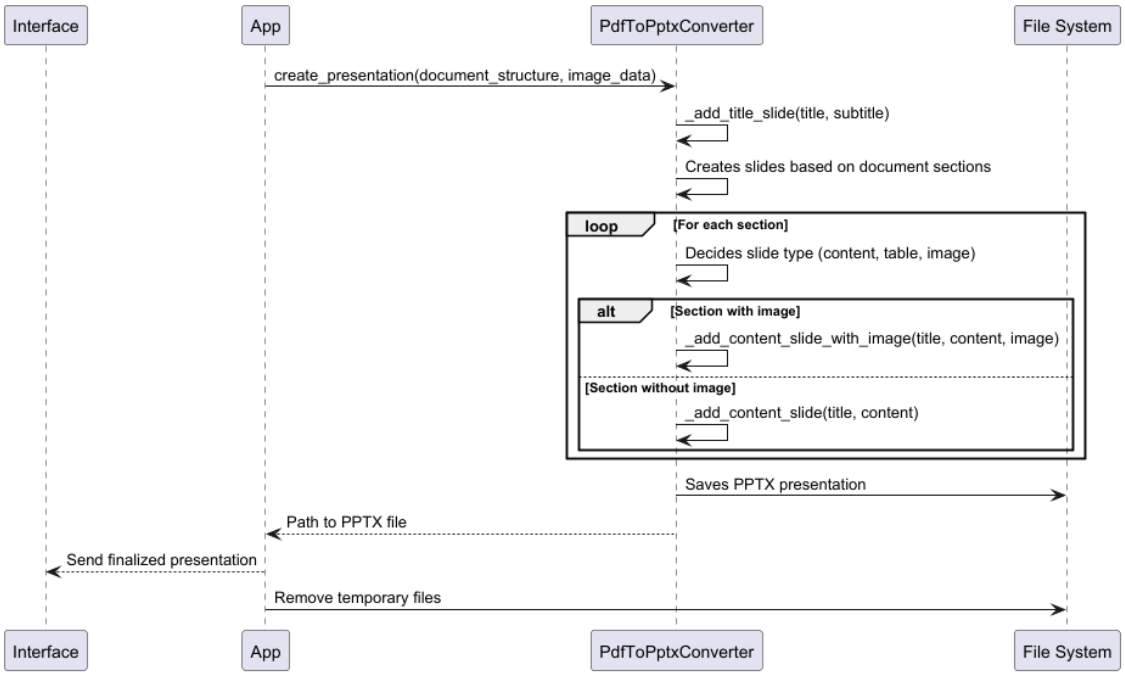

Figure 5.5: Sequence diagram of the generation phase

5.3.4 Conclusion

The methods described above form a robust, modular pipeline that automates the conversion of procedural documents into coherent and well structured slide presentations. From

document parsing and data extraction to AI-driven summarization and final rendering, each function is designed to meet the project's goals of automation, clarity and educational value.

5.4 Prompt Engineering and LLM Integration

The interaction with the LLM, running locally via the Ollama framework, is entirely prompt driven. `Clean and Structure`, `Analyze Document` and `Analyze Document with images` are the three key phases for this workflow. Each phase is designed to perform a specific stage of reasoning and transformation over the document content using highly curated prompt templates.

5.4.1 1. Text Cleaning and Structuring Prompt

The `Clean and Structure` phase is the first to be invoked. It sends a prompt to the LLM requesting a structural and visual cleaning of raw textual input, often extracted from PDF files using Optical Character Recognition (OCR) or similar tools. The objective is to correct layout issues and preserve procedural integrity.

Key Elements of the Prompt:

- **Preserve procedural elements:** Numbered steps, bullet points and headings must remain untouched in semantics and order.
- **Fix visual noise:** OCR errors, broken lines, or misplaced whitespace are to be corrected.
- **Improve readability:** The document must be transformed into a coherent, logically formatted textual resource suitable for downstream processing.
- **No extra explanation:** The LLM is instructed to return only the cleaned result, ensuring a minimal and usable output.

5.4.2 2. Semantic Document Structuring Prompt

Following text cleanup, the phase `Analyze Document` sends a second prompt designed to extract semantic meaning and transform the document into a slide ready structure in JSON format.

Key Goals and Instructions:

- **Decompose the document:** Identify the title, subtitle, version and date.
- **Extract and classify sections:** Each section should be labeled with a title, key points, an importance score (*high, medium, low*) and a type (*overview, procedure, warning, summary*).
- **Simplify content:** Complex explanations are to be reduced to short, presentation friendly phrases.
- **Structure preservation:** Sections should follow a narrative or instructional flow.
- **Enforce strict output format:** Only valid and parseable JSON is returned, ensuring integration with backend logic.

5.4.3 3. Slide Oriented Image Mapping Prompt

The third phase, `Analyze Document With Images`, expands the analysis by integrating image metadata extracted from the PDF (e.g., page number, dimensions). A prompt is sent to the LLM to determine the relationship between each image and the previously identified sections.

Prompt Objectives:

- **Image relevance:** Associate images to appropriate document sections.
- **Contextual reference detection:** Extract textual mentions of figures, diagrams, or charts to improve semantic mapping.
- **Slide layout recommendation:** Suggest how each image should be presented (e.g., side-by-side, background, standalone).
- **Output structure:** JSON format indicating for each section the list of relevant images, any textual references and the preferred visual style.

5.4.4 Conclusion

These prompt based strategies allow the system to offload complex reasoning and document understanding to the LLM while maintaining full control over formatting, structure and visual alignment. With this design, repeatable and explainable results are obtained that can be easily processed into educational slides. This demonstrates the relevance of well structured prompts to maximize the potential of LLM.

Chapter 6

Testing and Evaluation

Testing and evaluation are essential stages in the validation of software systems, ensuring that the developed solution not only functions correctly but also meets the quality attributes defined in the requirements. The source code for all tests is publicly available for analysis in the project's repository Pacheco, 2025. In this chapter, the proposed system is systematically examined through a combination of unit, integration, functional and non-functional tests, complemented by usability studies and performance assessments.

6.1 Unit Tests

In this project, a suite of unit and integration tests was designed to evaluate the correct behavior of each module involved in the pipeline from document parsing to slide generation. This section describes the testing approach applied to different components.

6.1.1 Integration and Main Workflow Tests

The `App` module tests validate the behavior of the complete pipeline from document ingestion to slide generation. It focuses on the integration between key components and their correct orchestration during execution.

Objectives of the Tests

The tests for the main module covers the following objectives:

- Confirm that all main modules are correctly invoked
- Validate fallback behavior and structure normalization logic
- Ensure that the output is consistent with the input document and format expectations

Key Test Scenarios

- **End-to-End Flow:** Mocks the interaction of all key components involved in the slide generation pipeline.
- **Structure Normalization:** Tests with different input types like valid dictionary structures, JSON strings and invalid inputs that trigger a fallback mechanism. This ensures resilience in handling potentially inconsistent or malformed output from the LLM.

- **Fallback Generator:** Validates the generation of a minimal valid structure using only section headers parsed from the raw text. This is a critical fail safe when LLM output is unavailable or corrupted.

To ensure system stability, these integration tests mock external dependencies and verify that components interact correctly. They also confirm that fallback mechanisms are in place. This increases the robustness of the solution in handling unpredictable input formats and LLM output anomalies.

6.1.2 PDF Text Extraction Module

The module `PdfExtractor` is responsible for extracting the full textual content from PDF documents. It attempts to ensure reliable extraction using two complementary libraries: `PyPDF2` and `pdfminer.six`, adopting a fallback mechanism to choose the best available result.

Testing Strategy

The tests for this module covers the following objectives:

- Validate the behavior of each individual extraction method
- Ensure fallback logic operates correctly when one method fails
- Confirm that the module raises appropriate errors when both methods fail
- Test integration scenarios with real or simulated PDF files

Mocks are used to simulate file access and control the behavior of the external libraries, thus ensuring that tests remain deterministic and independent of actual PDF content.

Key Tests and Coverage

- **PyPDF2 Extraction:** Tests whether text is correctly extracted from multiple pages using `PyPDF2`. It also verifies the number of pages processed and checks that the text content is accumulated properly.
- **PDFMiner Extraction:** Validates the function that extracts text, ensuring that the returned string matches expected output and that the external function is called once.
- **Hybrid Strategy:** Tests the logic that chooses the better of the two extractions (preferring longer content, assuming it's more complete). This is essential for ensuring data quality.
- **Fallback Logic:** Simulates the scenario where `pdfminer` fails and the module correctly falls back to using `PyPDF2`.
- **Error Handling:** Tests the case in which both extraction methods fail, ensuring that the function raises a descriptive `Exception`.
- **Integration Test:** Simulates the extraction process on a minimal but syntactically valid PDF file using a temporary directory, validating integration readiness.

The tests demonstrate that the `PdfExtractor` component is capable of reliably extracting textual content even in degraded conditions. Its layered approach to PDF processing

improves robustness, while its error handling logic ensures that the system can gracefully manage file corruption or unexpected formats.

6.1.3 Image Extraction Module

The module `ImageExtractor` is responsible for scanning PDF documents, detecting embedded images and storing them with metadata.

Testing Strategy

To ensure reliability, the image extractor is tested under several different scenarios, including:

- Standard PDF files containing extractable images
- PDFs without any images
- PDFs with images too small to be useful
- Handling of unexpected runtime exceptions

Mocking was used extensively to isolate the functionality of the extractor and simulate conditions such as malformed files, small image dimensions or internal library failures.

Key Tests and Coverage

- **Valid Image Extraction:** Ensures that when a PDF contains images, the extractor correctly returns a list of image dictionaries containing path, page number and dimensions.
- **Mocked Extraction Test:** Simulates PDF and image extraction using mocked `fitz` and `PIL.Image` objects to verify integration logic.
- **Empty Image Set:** Tests behavior when no images are detected in the document (the expected result is an empty list).
- **Small Image Filtering:** Validates that images below a defined threshold are discarded to avoid irrelevant visual elements.
- **Error Handling:** Verifies that the method handles exceptions gracefully and does not crash, returning an empty result instead.

The tests confirm that the `ImageExtractor` module can reliably identify, validate and store images extracted from procedural PDFs. The robustness of the image handling logic is crucial to ensuring that visual content can be later integrated into the slide generation process in a meaningful and scalable manner.

6.1.4 Prompt Processing and LLM Integration Tests

The `OllamaProcessor` module tests verifies the correct functioning of the class responsible for managing the interaction with the local LLM via the Ollama framework. Since this component plays a key role in content cleaning, structural analysis, its correctness is critical for ensuring coherent slide generation.

Objectives of the Tests

- Confirm that each method constructs prompts correctly and handles responses from the LLM
- Verify the parsing of structured outputs such as JSON
- Ensure robustness when integrating textual and visual data during the structuring process

Key Test Scenarios

- **Text Cleaning and Formatting:** Ensures that the prompt includes the original text content and the response returned from the LLM is correctly extracted and returned by the method
- **Text to Structure Transformation:** Validates that JSON content returned by the LLM is correctly parsed into a dictionary and sections are accurately mapped
- **Image Aware Structuring:** Evaluates the integration between the textual analysis and visual components. It confirms that:
  - The document is initially parsed into a structured format
  - The prompt properly incorporates image metadata
  - The final structure includes image associations and styling suggestions layout)

These tests are important in validating the correctness of requests and responses exchanged with LLM. By mocking the `ollama.chat` function, the tests isolate the application logic from external dependencies while ensuring that prompts are well formed and returned content is parsed as expected. This prevents malformed slide structures and supports reliable slide generation based on AI understanding.

6.1.5 Presentation Generation Tests

The `PptxConverter` module tests is responsible for validating the behavior of the `PdfToPptxConverter` class, which handles slide creation and visual structuring of the final presentation. This step is critical to ensure that the extracted and processed information is transformed into a clear and professionally styled slide deck.

Test Objectives

- Confirm the correct initialization of the presentation generator
- Verify that slides are properly created with correct layouts and content
- Ensure compatibility with the `python-pptx` library for file export

Key Test Scenarios

- **Initialization:** Ensures that the constructor initializes all key attributes, such as the output filename and theme configuration.
- **Title Slide Creation:** Uses mocks to simulate slide creation. It confirms that:

- A new slide is correctly added to the presentation
- The title and subtitle are set with the appropriate content
- Layouts and placeholders are properly accessed and modified

6.1.6 Functional Testing

Functional testing was conducted to validate that the system meets all the requirements defined in Table 5.1. Each functionality was tested in isolation as well as in an end-to-end workflow to ensure that the behavior was consistent with user expectations. The objective was to confirm that the implemented modules satisfy their intended purpose and that the final presentations are coherent and correctly structured.

The testing strategy followed a requirement based validation, where each functional requirement was mapped to a set of manual test cases.

Table 6.1 summarizes the functional test coverage and results.

Feature	**Test Performed**	**Result**
Procedural Document Import	Imported multiple PDF files of varying sizes (5–50 pages) to validate parsing robustness.	All documents imported successfully.
Content Extraction	Verified that procedural steps were identified and summarized. Compared generated summaries with source text to check information coverage.	Extraction consistent with input; minor redundancy in long paragraphs.
Slide Generation	Generated slides from structured content. Checked logical sequence, layout correctness and integration of extracted images.	Slides created with correct order and relevant visuals; logical coherence preserved.
Slide Customization	Tested all 3 available presentation themes during generation. Validated correct application of colors and layouts.	All themes applied correctly; user could switch themes without errors.
Presentation Export	Exported final presentation to PPTX format and opened in Microsoft PowerPoint.	Export successful; presentations fully compatible across platforms.

Table 6.1: Functional testing of system requirements

The results confirm that all core functional requirements were satisfied. The system reliably imports procedural documents, extracts structured content, generates logically consistent slides enriched with images, applies customized themes and exports the final output in a widely compatible format.

6.1.7 Non-Functional Testing

In addition to unit and integration testing, the system was evaluated against the non-functional requirements established. These tests aim to validate performance, reliability,

usability, maintainability and security attributes, ensuring that the system not only functions correctly but also provides a high quality experience.

Performance

To assess the system's performance, load tests were executed using Apache JMeter (Halili and H, 2008). Three scenarios were tested: retrieving available models (`GET Models`), processing a valid PDF document with 18 pages using Llama3.2 (1B) and Llama3 (8B) models, and submitting a request without a file (`Convert PDF - No File`). The results are summarized in Table 6.2.

Test Case	# Samples	Avg. Time (ms)	Min (ms)	Max (ms)	Error %
GET Models	10	213	58	555	0.00%
Convert PDF – Llama3.2 (1B)	10	25,253	19,671	40,553	0.00%
Convert PDF – Llama3 (8B)	10	799,383	97,385	1,033,508	10.00%
Convert PDF – No File	10	108	2	1,055	100.00%

Table 6.2: Performance results obtained with Apache JMeter

The `GET Models` request exhibited an average response time of only 213 ms, demonstrating that lightweight interactions with the API are handled efficiently.

The main evaluations focused on the `Convert PDF - Llama3.2 (1B)` and `Convert PDF - Llama3 (8B)` scenarios, where a PDF with 18 pages was submitted for processing. On the first one, the average response time was approximately 25 seconds, with a minimum of 19 seconds and a maximum of 40 seconds. For the second scenario the average response time was approximately 13 minutes, with a minimum of 97 seconds and a maximum of 1033 seconds. While this duration is within the acceptable range for complex document processing involving text extraction, summarization and slide generation, it demonstrates that performance is highly dependent on document size and LLM processing time. The observed error rate of 10% was primarily linked to timeouts on particularly large requests.

For comparison, the `Convert PDF - No File` case returned almost instantly (average 108 ms) but, as expected, failed in 100% of the attempts since no input was provided. This confirms that the error handling mechanisms were triggered correctly.

Overall, the results demonstrate that the system only meets the performance requirement when using Llama3.2 (1B). While more complex models like Llama3 (8B) may enhance summarization quality, their computational cost makes them unsuitable for time sensitive document conversion.

System Reliability

To assess the system's reliability and availability, a stress test was done using Apache JMeter. A total of 300 requests were executed against the conversion endpoint, each submitting a document with 18 pages, for processing with the `Llama3.2 (1B)` model. The objective was to validate whether the system could maintain stable operation and remain available in at least 95% of the requests.

Test Case	Samples	Avg. Time (ms)	Min (ms)	Max (ms)	Error %
Convert PDF	300	633,419	33,538	3,933,442	35.67%

Table 6.3: Stress test results.

The results reveal significant performance degradation under load. Although some requests were processed in as little as 33 seconds, the average response time was approximately 10.5 minutes, and the slowest request took more than one hour to complete. Furthermore, the error rate reached 35.67%, which falls considerably short of the targeted availability of 95%. Failures were primarily due to request timeouts and resource exhaustion during high concurrency. Despite these limitations, the system successfully processed the majority of the requests and demonstrated resilience by handling long running operations without crashing the service entirely.

In conclusion, the stress test indicates that while the system is reliable under moderate load, it does not yet meet the defined availability requirement of 95% when subjected to heavy stress conditions.

Content Reliability

One of the key non-functional requirements concerns the reliability of the generated content, ensuring that the autonomously created slides preserve the logical coherence and integrity of the document. To achieve this it is necessary to verify that the summarized text, extracted images and slide layouts remain consistent with the procedural structure of the original file. During testing it was observed that the system consistently produced concise and well structured summaries. The resulting slides offered a clear presentation of instructions, highlighting essential points. This confirms that the summarization component of the pipeline works reliably.

However, certain discrepancies were noted between text and visual content. For instance, in some slides the extracted diagrams or images didn't fully match the accompanying text, as we can see on Figure 6.1. These mismatches were relatively minor and did not compromise the overall understanding of the material. The association between text and images is based on heuristics and metadata rather than a deep semantic matching.

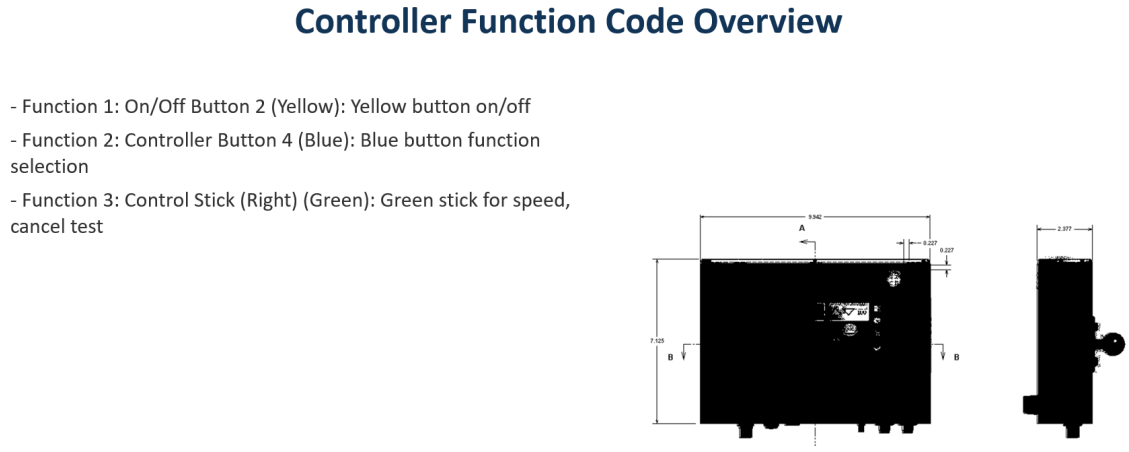

Figure 6.1: Incoherence between text and image example

Overall, the evaluation confirms that the system achieves a high degree of content reliability at the textual level, with consistent preservation of logical flow.

Usability Testing

To evaluate the usability of the system, a small test was conducted with four university students from different academic backgrounds and one university professor from the health sciences. Each participant was asked to perform a complete workflow: upload a PDF, choose a presentation theme and the LLM model and generate the corresponding slides. The tasks were timed, errors were recorded and participants rated their overall satisfaction on a scale from 1 (very dissatisfied) to 5 (very satisfied) of the interface.

Participant	Background	Errors	Time (s)	Satisfaction (1-5)
Student A	Health Sciences	0	18	4
Student B	Health Sciences	0	24	4
Student C	Social Sciences	0	21	4
Student D	Technology	0	20	5
Professor	Health Sciences	0	21	4

Table 6.4: Usability test results

The results indicate that all participants were able to complete the task successfully without encountering any errors. Task completion times ranged between 18 and 24 seconds, showing consistent performance across different user profiles. The average satisfaction score was **4.2/5**, suggesting a high level of usability and acceptance of the interface. Notably, the participant from a technological background rated the experience with the highest satisfaction score (5/5), while the remaining participants consistently gave a rating of 4/5. This confirms that the system interface is intuitive and user friendly, even for non-technical users.

One observation is that some participants, particularly those outside the technology field, expressed uncertainty regarding the step where they were required to select a LLM model.

Maintainability

Maintainability refers to the ease with which the system can be understood, modified, and extended in the future. To assess this quality attribute, the project was analyzed using *SonarQube*.

The analysis yielded a global rating of **A**, indicating a technical debt ratio below 5%. This confirms that the codebase is highly maintainable and that future enhancements can be integrated with minimal effort. However, the tool identified a total of **16 code smells**, categorized according to severity as follows:

- 11 issues classified as **high severity**, requiring attention due to potential long-term maintenance complexity.
- 4 issues of **medium severity**, typically related to code readability or minor refactoring opportunities.
- 1 issue of **low severity**, with negligible impact on maintainability.

Despite these findings, none of the identified smells compromise the correctness or stability of the system. The SonarQube report reinforces the combination of modular design, clear separation of concerns and consistent documentation already ensures that the system can be effectively maintained.

Beyond static analysis, the project emphasized readability through descriptive comments and module level that explain non obvious decisions, external dependencies and expected data structures. This documentation, combined with a modular design and clear separation of concerns, facilitates future evolution.

Test coverage results are summarized in Table 6.5. Overall line coverage reached **53%**. Coverage is high in the `ImageExtractor` (92%) and `PdfExtractor` (80%) modules, satisfactory in `OllamaProcessor` (70%) and lower in `App` (47%) and `PptxConverter` (20%).

Module	Coverage
`ImageExtractor`	92%
`App`	47%
`OllamaProcessor`	70%
`PptxConverter`	20%
`PdfExtractor`	80%
TOTAL	**53%**

Table 6.5: Unit test coverage by module.

Security Testing

Security plays a crucial role in ensuring that documents used are handled with confidentiality and that no sensitive data persists beyond its intended use. In the implemented system, all input and output files are stored only temporarily in the file system. Once the conversion process is finalized and the presentation is delivered to the user, these files are automatically deleted. This mechanism guarantees that no residual user data remains stored on the server after processing.

To further validate the security of the web interface, an analysis was performed using the **OWASP ZAP** vulnerability scanner (OWASP, 2025). The tool highlighted one relevant security issue:

- **ClickJacking Protection:** The response headers did not include either a `Content-Security-Policy` with the `frame-ancestors` directive or the `X-Frame-Options` header. This omission makes the application potentially vulnerable to ClickJacking attacks.
- **Cross-Site Request Forgery (CSRF):** The analysis detected missing CSRF protection in form submissions. This vulnerability is common in Flask applications that do not integrate explicit CSRF protection.

6.1.8 Test Strategy Conclusion

The testing strategy adopted in this project encompasses all critical components of the application:

- **Text Extraction:** Ensures reliability of PDF parsing across multiple tools

- **Image Handling:** Validates the ability to extract and filter relevant visual content from procedural documents
- **LLM Integration and Prompt Processing:** Confirms the quality and structure of AI assisted summarization and classification
- **Presentation Generation:** Ensures the final output adheres to formatting expectations and structural integrity

A combination of unit testing, fixture based simulation and extensive mocking guarantees that each module behaves independently and robustly, even in the face of failures or invalid inputs. This modular and test-driven approach enhances maintainability, supports scalability and reinforces confidence in the system's reliability across various use cases.

6.2 Results

The system was deployed with a functional web interface, enabling users to easily interact with the document to presentation generation. This section presents the main outcomes, illustrating both the interface design and the capabilities of the implemented features.

6.2.1 Main Interface

Figure 6.2 shows the main page of the system. The interface was designed to be minimalistic and intuitive, allowing users to upload their PDF documents either via drag-and-drop or by selecting a file directly. The maximum allowed file size is 16 MB, which provides a balance between supporting large documents and ensuring efficient processing.

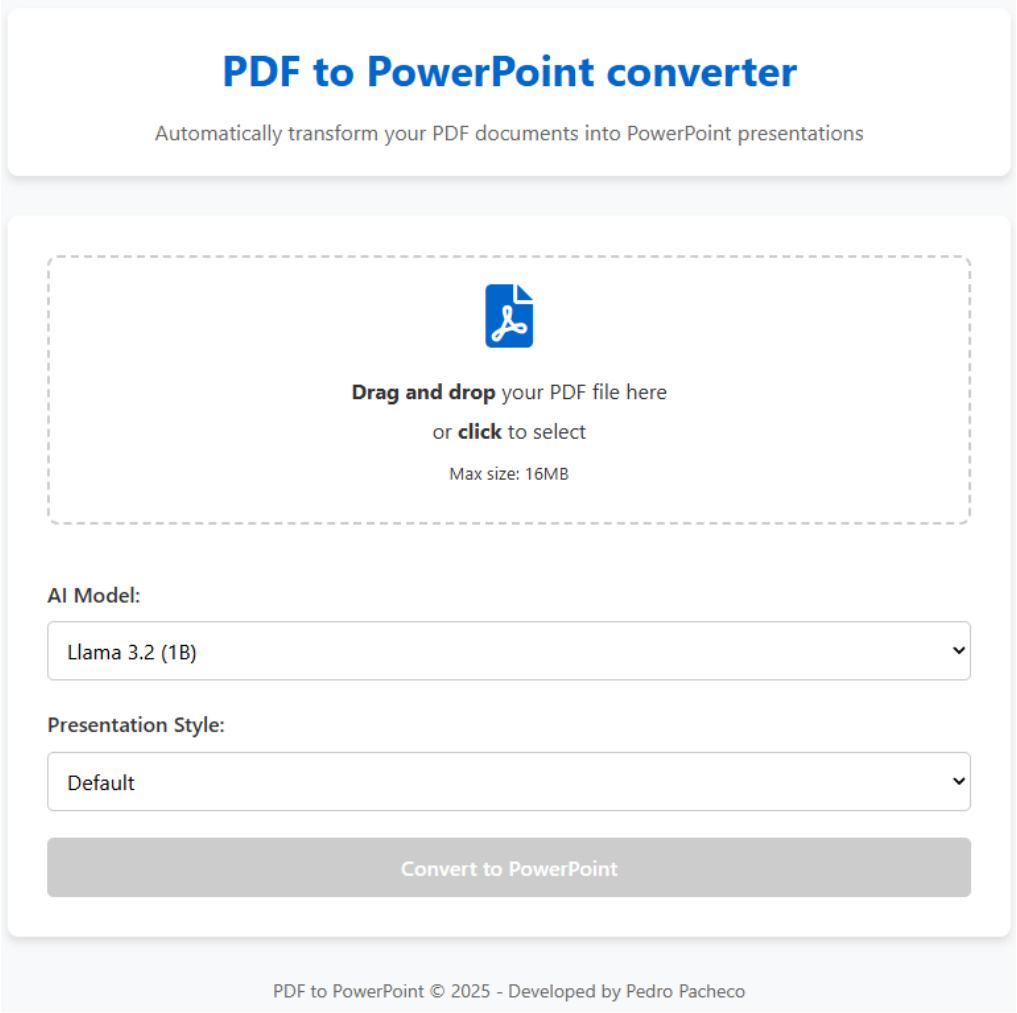

Figure 6.2: Main interface of the PDF to PowerPoint converter

The interface also provides two key configuration options:

- **AI Model Selection:** Users can select the language model responsible for data processing (Figure 6.3). Options include lightweight models, such as `Llama 3.2 (1B)`, which prioritize faster response times, as larger models like `Llama 3 (8B)`, `DeepSeek R1 (14B)` and `Gemma3 (12B)`, which may offer richer text generation at the cost of higher processing times.
- **Presentation Style:** Users can choose one of three available themes to customize the visual layout of the generated slides. This feature, shown in Figure 6.4, allows users to select between "Default", "Corporate" and "Minimal" styles, ensuring the final presentation aligns with their aesthetic preferences.

Figure 6.3: Dropdown menu for selecting the AI model

Figure 6.4: Dropdown menu for selecting the presentation style

Once a file is selected and the user clicks "Convert to PowerPoint" button, the system initiates the backend processing. To provide a clear user experience during this waiting period, a dedicated status screen is displayed. As depicted in Figure 6.5, a loading spinner and a message inform the user that the document is being processed. The message also indicates that the duration may vary depending on the size of the file.

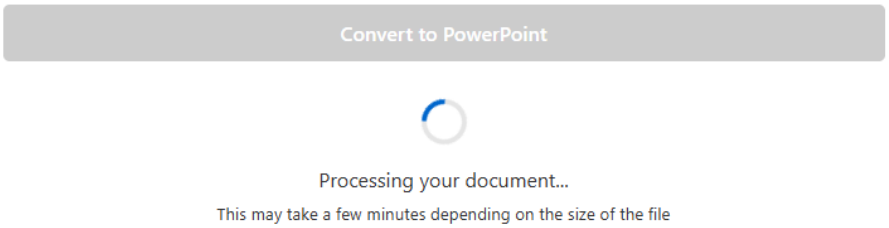

Figure 6.5: Processing status screen displayed during file conversion

6.2.2 Generated PowerPoint Presentations

To demonstrate the system's output, several PDF documents were processed into presentation slides. Figures 6.6 and 6.7 show examples of the generated results, including the first title slide and an example of a content slide.

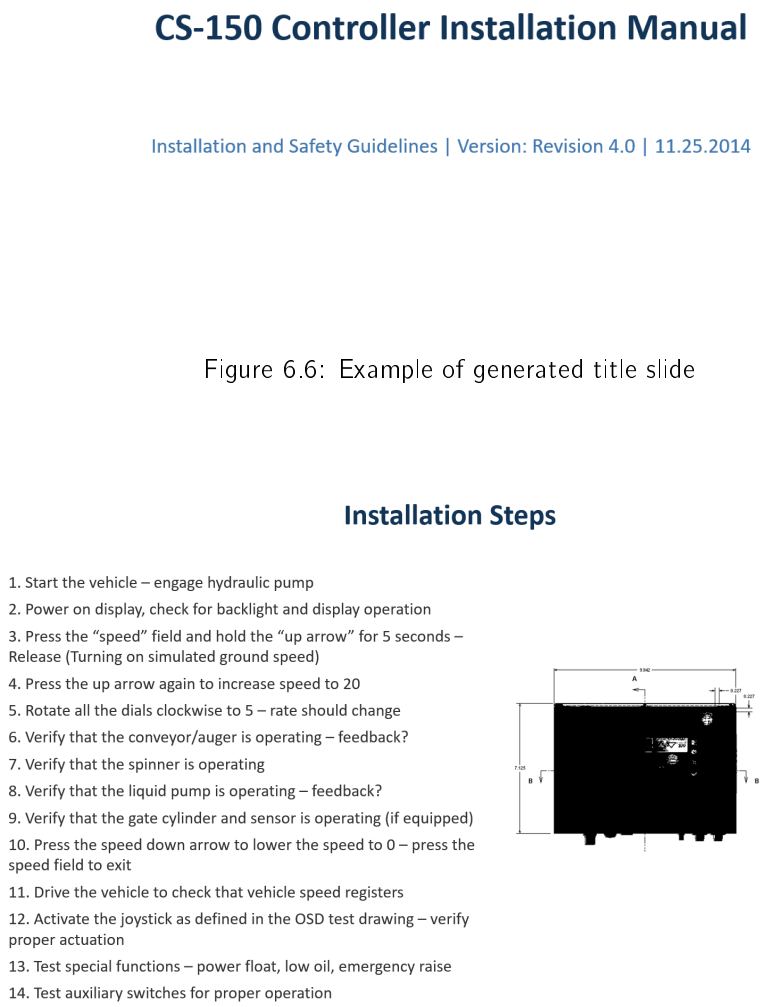

Figure 6.6: Example of generated title slide

Figure 6.7: Example of generated content slide

The **title slides** were consistently generated, containing the document title and in some cases the subtitle. This confirms that the pipeline is able to extract high level document information reliably.

The **content slides**, however, revealed mixed results. In some instances, the extracted text was correctly summarized and structured according to the procedural steps in the source document. In other cases, the slides displayed unrelated or incoherent content. Additionally, the system often reused the same extracted image across all content slides, except the title slide.

Table 6.6 summarizes the most frequent issues encountered during the generation of the PowerPoint files.

Observed Issue	Description
Unrelated Content	Certain slides contained text not related to the intended section of the document.
Incoherent Content	Occasionally, the generated text lacked coherence or did not make sense in context.
Repeated Images	The same image was reused across most content slides, reducing relevance and variety.
Missing Visual Association	In some cases, images were not appropriately linked to the correct section of text.

Table 6.6: Common issues identified in generated PowerPoint slides

6.3 Summary

The testing and evaluation stage provided a comprehensive assessment of the developed system, addressing both functional and non-functional requirements.

From a **functional perspective**, the implemented modules performed reliably, documents could be imported, content was extracted and summarized and presentations were generated in the expected format. Requirement based validation confirmed that all functional requirements defined in Table 5.1 were satisfied. Manual testing showed that procedural documents of different sizes were consistently imported, key steps were correctly summarized, slides were generated in logical sequence with visual elements, customization options were applied correctly and final presentations were successfully exported in PPTX format. This demonstrates that the system delivers on its core functionalities, ensuring an end-to-end workflow that aligns with user expectations.

Unit and integration tests further reinforced this by confirming their correct orchestration in the main pipeline and validating individual modules, such as text and image extraction, slide generation and LLM integration. These tests also verified the fallback mechanisms that guarantee robustness when handling degraded conditions or inconsistent outputs.

In terms of **non-functional aspects**, the evaluation highlighted several important findings. Performance tests confirmed that the system can meet the expected response times when using lightweight models such as Llama3.2 (1B), although larger models introduced significant delays that limit their practical use. Reliability tests showed that the system remains functional under stress but falls short of the defined availability target, mainly due to timeouts and resource limitations. Usability testing demonstrated that the interface is intuitive and well accepted by users with different backgrounds, though some uncertainty was noted when selecting AI model. Content reliability results were mixed, while the textual summarization generally preserved logical flow and clarity, inconsistencies between text and associated images were observed. The maintainability analysis indicated that the codebase is modular, well documented, and supported by testing, which facilitates future enhancements despite the presence of manageable technical debt. Security tests confirmed that temporary file handling effectively protects user data, although specific vulnerabilities such as missing CSRF protection and anti-ClickJacking headers were identified.

Overall, the results confirm that the system provides a solid foundation for automating the conversion of procedural documents into presentation slides. It successfully integrates different technologies to deliver functional results, while also exposing areas, particularly

performance optimization, reliability under stress and security hardening, that require further attention. This evaluation not only validates the current solution but also informs the directions for future improvements and refinements.

Chapter 7

Conclusions

The work developed in this dissertation addressed the challenge of automating the conversion of procedural documents into structured and visually coherent presentation slides. The proposed system combined NLP techniques, image extraction and the integration of local LLMs to achieve a workflow capable of importing documents, extracting and summarizing their content, and generating corresponding presentation files in PPTX format.

The main contribution of this project lies in demonstrating the feasibility of integrating open-source tools and language models to reduce the time and effort typically required for preparing didactic and professional presentation material. The application proved effective in handling document parsing, text summarization and slide creation, while also offering customization options for themes and layouts. This shows that a practical solution can be achieved with accessible resources and modular design.

The evaluation confirmed that the system performs well in terms of functionality, with unit and integration tests validating its reliability across different modules. Usability studies showed that users with diverse backgrounds could interact with the interface successfully, completing tasks in less than half a minute on average and reporting high levels of satisfaction. The maintainability analysis further highlighted the robustness of the codebase, which was rated as highly maintainable by SonarQube.

Nonetheless, the testing phase also exposed important limitations. Performance was strongly dependent on the chosen model, while lightweight models met the time constraint of 60 seconds, larger models significantly exceeded this threshold, making them unsuitable for time sensitive scenarios. Reliability under heavy load was below the target availability of 95%, with high error rates during stress testing. Content reliability revealed mixed results, as textual summarization was consistent, but visual coherence between images and slides was not always preserved. Additionally, security analysis detected missing protections against CSRF and ClickJacking, requiring further hardening of the web interface.

It is important to note that the initial project planning was not fully adhered to and several milestones were delayed. A major reason was the significant amount of time required for the implementation phase. Originally, the system was intended to rely on specialized summarization and analysis libraries, however this strategy was later revised in favor of a different approach based on large scale language models. While this decision simplified some aspects of summarization and content structuring, it required substantial effort to redesign and integrate the processing pipeline. Additional factors also contributed to the deviation from the planned schedule, including the steep learning curve associated with some of the libraries and frameworks, the time invested in ensuring proper testing coverage and the challenges of aligning functional requirements with non-functional aspects such as performance, security

and usability. These adjustments, although time consuming, ultimately allowed the project to reach a more robust and extensible implementation.

In summary, the system successfully achieved its main objectives and demonstrated that automating the conversion of procedural documents into presentations is viable and valuable. However, the identified limitations also indicate the need for further refinement to enhance scalability, reliability and content accuracy.

The source code is publicly available on GitHub, fully open source, allowing for testing, issue reporting, improvements and further extensions of the work developed Pacheco, 2025.

7.1 Future Work

Building upon the results of this dissertation, several directions can be pursued to extend and improve the system:

- **Performance Optimization:** Introduce caching strategies, batching or parallel processing to reduce response times, especially when processing large documents with more advanced models.
- **Improved Content Alignment:** Develop methods to ensure stronger semantic matching between text and images, potentially leveraging multimodal models to improve visual coherence.
- **Scalability and Reliability:** Adapt the system to support distributed execution or container based deployments, enhancing resilience under heavy load and improving availability metrics.
- **Security Enhancements:** Integrate CSRF tokens and strict security headers to address the vulnerabilities identified by the OWASP ZAP analysis.
- **User Experience:** Implement additional customization features including slide templates and automatic layout adjustments.
- **Extended Evaluation:** Conduct larger scale usability studies and more extensive benchmarking across different domains to validate generalization and robustness.

These improvements would contribute to transforming the system into a more mature, scalable, and secure tool, capable of supporting real world adoption in academic, corporate and training environments.

Bibliography

Álvarez, Yessenia Díaz et al. (Apr. 2023). "Comparación de los recursos de tiempo y memoria en la ejecución de una tarea de preprocesamiento de textos". PhD thesis. *Tecnológico Nacional de México/Instituto Tecnológico Superior de Xalapa (ITSX). url: www.diputados.gob.mx%20https://www.researchgate.net/publication/371504198.

Binmakhashen, Galal M. and Sabri A. Mahmoud (Oct. 2019). "Document layout analysis: A comprehensive survey". In: *ACM Computing Surveys* 52 (6). issn: 15577341. doi: 10.1145/3355610.

Canny, Steve (Aug. 2023). *python-pptx Documentation*. url: https://app.readthedocs.org/projects/python-pptx/downloads/pdf/stable/.

Chopra, Abhimanyu, Abhinav Prashar, and Chandresh Sain (2013). "Natural Language Processing". In: *INTERNATIONAL JOURNAL OF TECHNOLOGY ENHANCEMENTS AND EMERGING ENGINEERING RESEARCH* 1 (4). issn: 2347-4289. url: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=eeace1d14e266a5cd44fe781a874c662928602fd.

Clark, Jeffrey A (Apr. 2025). *Pillow (PIL Fork) Documentation Release 11.2.1*. url: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://app.readthedocs.org/projects/pillow/downloads/pdf/stable/.

DeepSeek-AI et al. (Jan. 2025). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". In: url: http://arxiv.org/abs/2501.12948.

Du, Yaxue (Mar. 2021). *Automatic Extraction of Slides from Scientific Papers*. Tech. rep. Politecnico di Torino. url: https://webthesis.biblio.polito.it/secure/18171/1/tesi.pdf.

Franco, P Regnath, K Thirumoorthy, and K Muneeswaran (Mar. 2016). "Automatic Creation of Well-Organized Slides From Documents". In: IEEE, pp. 1173–1178. isbn: 978-1-4673-9338-6. doi: 10.1109/WiSPNET.2016.7566321. url: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7566321.

Fu, Tsu-Jui et al. (June 2022). "DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents". In: PKP - Publishing Services Network, pp. 634–642. doi: https://doi.org/10.1609/aaai.v36i1.19943. url: www.aaai.org.

Ghimiri, Devndra (May 2020). *Comparative study on Python web frameworks: Flask and Django*. Tech. rep. Metropolia University of Applied Sciences. url: https://www.theseus.fi/bitstream/handle/10024/339796/Ghimire_Devndra.pdf?sequence=2&isAllowed=y.

Gonçalves, José et al. (Mar. 2025). *Evaluating LLaMA 3.2 for Software Vulnerability Detection*. Tech. rep. ISEP. url: http://arxiv.org/abs/2503.07770.

Goodfellow, Ian J et al. (2014). *Generative Adversarial Nets*. Tech. rep. Montreal University. url: http://www.github.com/goodfeli/adversarial.

Grattafiori, Aaron et al. (Nov. 2024). "The Llama 3 Herd of Models". In: url: http://arxiv.org/abs/2407.21783.

Halili and Emily H (2008). *Apache JMeter : a practical beginner's guide to automated testing and performance measurement for your websites*. Packt Pub., p. 129. isbn: 9781847192950.

url: http://download.51testing.com/ddimg/uploadsoft/20131113/ApacheJMeter_English.pdf.

Harth, Sid (Dec. 2022). *AI vs ML vs DL: Complete Understanding*. url: https://medium.com/@sidharthgn/ai-vs-ml-vs-dl-what-is-the-difference-397dabf59911.

Kim, Kwang Gi (2016). *Book Review: Deep Learning*. Vol. 22. The Korean Society of Medical Informatics, p. 351. doi: 10.4258/hir.2016.22.4.351.

Knill, K and S Young (1997). *Chapter 2 Hidden Markov Models in Speech and Language Processing*. Ed. by UK Engineering Department Cambridge University Cambridge. Springer, Dordrecht, pp. 27–28. isbn: 978-94-017-1183-8. doi: https://doi.org/10.1007/978-94-017-1183-8_2.

Liddy, Elizabeth D (2001). *Natural Language Processing*. Tech. rep. Syracuse University. url: https://surface.syr.edu/istpub.

Lu, Zehao (Aug. 2023). *Unsupervised Paper2Slides Generation*. Tech. rep. Universiteit Utrecht. url: https://studenttheses.uu.nl/bitstream/handle/20.500.12932/45939/master_thesis_zehao_lu_unsupervised_paper2slides_generation.pdf?sequence=1&isAllowed=y.

MagicSlides (Mar. 2025). *MagicSlides: Crie apresentações a partir de PDFs online*. url: https://www.magicslides.app/pt.

Mahesh, Batta (Jan. 2019). *Machine Learning Algorithms - A Review*. Tech. rep. International Journal of Science and Research (IJSR), pp. 381–386. doi: 10.21275/art20203995. url: https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096t.

Mesquita, Déborah (2021). *Python AI: How to Build a Neural Network Make Predictions*. url: https://realpython.com/python-ai-neural-network/.

Mondal, Ishani et al. (Mar. 2024). "Presentations by the Humans and For the Humans: Harnessing LLMs for Generating Persona-Aware Slides from Documents". In: *the Association for Computational Linguistics*. Vol. 1. Long Papers, pp. 2664–2684. url: https://github.com/Ishani-Mondal/.

Mufid, Mohammad, Arif Basofi, and M Rasyid (Sept. 2019). *Design an MVC Model using Python for Flask Framework Development*. Ed. by Politeknik Elektronika Negeri Surabaya and Akademi Komunitas Negeri Lamongan. IEEE, pp. 214–219. isbn: 9781728144498. doi: 978-1-7281-4449-8/19.

Nadkarni, Prakash M., Lucila Ohno-Machado, and Wendy W. Chapman (Sept. 2011). "Natural language processing: An introduction". In: *Journal of the American Medical Informatics Association* 18 (5), pp. 544–551. issn: 10675027. doi: 10.1136/amiajnl-2011-000464. url: https://academic.oup.com/jamia/article/18/5/544/829676?login=false.

Namboodiri, Anoop M and Anil K Jain (1998). "Document Structure and Layout Analysis". In: *ITC-irst Technical Report* 9703.

Ollama (2025). *Ollama*. url: https://ollama.com/.

OWASP (2025). *OWASP ZAP*. url: https://www.zaproxy.org/docs/.

Pacheco, Pedro (2025). *PedroRFPacheco/genereratePPTwithPDF*. url: https://github.com/pedrorfpacheco/genereratePPTwithPDF.

Page, Matthew J et al. (2021). "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews". In: *BMJ* 372. doi: 10.1136/bmj.n71. url: https://www.bmj.com/content/372/bmj.n71.

Poole, David L.., Alan K.. Mackworth, and Randy. Goebel (1998). *Computational intelligence : a logical approach*. Vol. 1. Oxford University Press. isbn: 0-19-510270-3.

Reynolds, Douglas (2009). *Gaussian Mixture Models **.

Sefid, Athar et al. (Nov. 2019). "Automatic Slide Generation for Scientific Papers". In: *Third International Workshop on Capturing Scientific Knowledge co-located with the 10th International Conference on Knowledge Capture (K-CAP 2019),SciKnow@K-CAP 2019*, pp. 11–16. url: https://par.nsf.gov/biblio/10173903.

Shinde, PP and S Shah (2018). "A Review of Machine Learning and Deep Learning Applications". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE. isbn: 9781538652572. doi: 10.1109/ICCUBEA.2018.8697857. url: https://ieeexplore.ieee.org/abstract/document/8697857.

Siami-Irdemoosa, Elnaz, Saeid R. Dindarloo, and Mostafa Sharifzadeh (Oct. 2015). "Work breakdown structure (WBS) development for underground construction". In: *Automation in Construction* 58, pp. 85–94. issn: 09265805. doi: 10.1016/j.autcon.2015.07.016.

SlideSpeak (2025). *SlideSpeak: Transforme seus PDFs em apresentações interativas*. url: https://slidespeak.co/.

Sun, Edward et al. (May 2021). *D2S: Document-to-Slide Generation Via Query-Based Text Summarization*. Tech. rep. doi: https://doi.org/10.48550/arXiv.2105.03664. url: https://github.com/IBM/.

Team, Gemma et al. (Mar. 2025). *Gemma 3 Technical Report*. Tech. rep. Google. url: http://arxiv.org/abs/2503.19786.

Wondershare (2025). *PDFelement: Converta PDFs em apresentações com IA*. url: https://pdf.wondershare.com/.

Yang, Wen and Min Yu (May 2025). "Automatic Extraction of PDF Table Data Based on Deep Learning". In: *Proceedings of 2025 4th International Conference on Big Data, Information and Computer Network, BDICN 2025*. Association for Computing Machinery, Inc, pp. 222–226. isbn: 9798400712425. doi: 10.1145/3727353.3727391. url: https://dl.acm.org/doi/10.1145/3727353.3727391.

Zhao, Haiyan et al. (Feb. 2024). "Explainability for Large Language Models: A Survey". In: *ACM Transactions on Intelligent Systems and Technology* 15 (2). issn: 21576912. doi: 10.1145/3639372.

Zhou, Zhi-Hua (Jan. 2021). *Machine learning*. Vol. 1. Springer nature, pp. 1–60. isbn: 978-981-15-1966-6. doi: https://doi.org/10.1007/978-981-15-1966-6. url: https://books.google.com.br/books?hl=pt-PT&lr=&id=ctM-EAAAQBAJ&oi=fnd&pg=PR6&dq=machine+learning+Zhou,+Zhi-Hua&ots=o_LnT1SqYt&sig=apYvbpxRbxqN2FhZWKz0_E7tCdk#v=onepage&q=machine%20learning%20Zhou%2C%20Zhi-Hua&f=false.

Appendix A

Risk ID	Description	Cause	Effect	Risk Owner	Probability (1-5)	Impact (1-5)	PI	Expected	Risk	Response
	Description of the risk	Cause of the risk	Effect on the project	Name of person who monitors the risk	Group sourced rough estimate of how likely this is to occur	Rough estimate of how significant	Probability multipli	What will happen if the risk becomes	Decision made by group on how to respond	How do you know it is time to put the response into play
1	Delay in the delivery of initial functionalities	Lack of detailed planning	General project delays	Pedro Pacheco	2	3	6	Accumulated delays impact the final delivery	Mitigate	Weekly follow-up on progress and adjusting the schedule
2	Critical bugs during development	Insufficient tests or careless validations	Deliverables compromised with critical errors	Pedro Pacheco	3	5	15	Inconsistent results	Mitigate	Conduct continuous tests
3	Incomplete or inconsistent input data	Poorly trained or inaccurate data	Functionality is poorly trained or inaccurate	Pedro Pacheco	3	5	15	Inconsistent results	Mitigate	Receive and validate adequate datasets in advance
4	Incomplete documentation at closing	Overload at the end of the project	Difficulty in delivering a complete solution	Pedro Pacheco	3	3	9	Difficulty in delivering a complete solution	Mitigate	Planning sessions to review documentation
5	Failure to efficiently integrate visual resources	Limitation of external API libraries	Presentations without relevant visual elements	Pedro Pacheco	3	4	12	Results are not as attractive to stakeholders	Mitigate	Search for alternative APIs or adapt manual methods
6	Failure to customize generated presentations	Unoptimized or simplified algorithms	Results are not as attractive to stakeholders	Pedro Pacheco	3	4	12	Results are not as attractive to stakeholders	Mitigate	Test and validate customizations with real cases

Figure A.1: Project Risks Management

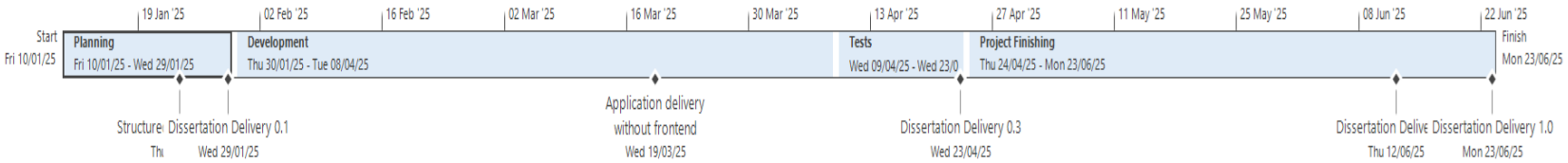

Figure A.2: Project Timeline

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
GET Models	10	213	58	555	148.82	0.00%	21.9/min	0.13	0.04	358.0
Convert PDF - Valid	10	799383	97385	1033508	273731.48	10.00%	34.1/hour	0.55	18.90	59595.3
Convert PDF - No File	10	108	2	1055	315.53	100.00%	38.2/hour	0.00	0.00	208.0
TOTAL	30	266568	2	1033508	408560.99	36.67%	1.7/min	0.56	18.90	20053.8

Figure A.3: Load Tests Results

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Convert PDF - Valid	300	633419	33538	3933442	826444.24	35.67%	1.6/min	1.04	52.95	39254.3
TOTAL	300	633419	33538	3933442	826444.24	35.67%	1.6/min	1.04	52.95	39254.3

Figure A.4: Stress Tests Results

```
Name                    Stmts   Miss  Cover
-------------------------------------------
image_extractor.py         40      3    92%
main.py                   222    117    47%
manageData.py              87     26    70%
ppt_generator.py          407    324    20%
readPDF.py                 40      8    80%
-------------------------------------------
TOTAL                    1042    493    53%
```

Figure A.5: Coverage of unit tests