



# Project Proposal

## Automatic Manual Development

Romeo Bandinelli, Djonathan Quadras  
Università degli Studi di Firenze

April, 2025

## Summary

<b>1</b>	<b>About the Project</b>	<b>2</b>
1.1	About the Team . . . . .	2
<b>2</b>	<b>Proposed Solution</b>	<b>3</b>
2.1	Data Sources . . . . .	3
2.2	Large Language Models . . . . .	3
2.3	Clustering and Machine Learning . . . . .	4
2.4	Document Generation . . . . .	4
<b>3</b>	<b>Timetable and Activities</b>	<b>5</b>
3.1	Data Collection . . . . .	5
3.2	Machine Learning Modelling . . . . .	5
3.3	Prompt Modelling for LLM . . . . .	6
3.4	Template Development . . . . .	6
3.5	Validation and Training . . . . .	6
<b>4</b>	<b>Final Considerations</b>	<b>7</b>

# 1 About the Project

Fashion for Future, a research lab at the University of Florence, proposes a project to Frigel, a leading manufacturer of cooling and temperature control machines. This initiative focuses on automating the currently manual development process for new machine releases, significantly improving efficiency and reducing development time. The project will leverage advanced technologies to streamline workflows and optimize the entire production lifecycle.

## 1.1 About the Team

The Fashion For Future research lab is a multidisciplinary and multi-skilled environment focused on bridging the gap between academia and industry by connecting technological development with practical application. Within this framework, various projects are organized to empower professionals in the use of artificial intelligence tools. The technical and scientific lead is Prof. Dr. Romeo Bandinelli, Associate Professor in the Department of Industrial Engineering at the University of Florence (Italy) with expertise on management and innovation of industrial processes, with a special focus on the dynamics of product lifecycle management and supply chain operations. He is also a co-founder of Balance, a start-up that supports companies in production planning and continuous improvement initiatives.

The team also includes the researcher Djonathan Luiz de Oliveira Quadras. He has worked on projects involving databases integration, development of managerial dashboards using BI tools, and the application of machine learning and AI for automated data analysis and reporting. His current doctoral project focuses on applying AI to fashion supply chains, with an emphasis on data-driven decision-making support for businesses.

## 2 Proposed Solution

This section proposes an automated solution for generating technical manuals, significantly reducing the time and resources currently required for manual creation. This automated approach offers improved consistency, scalability, and the potential for multilingual support, ultimately streamlining the entire manual production process.

The figure below illustrates the proposed framework. The process begins with data collection from various internal company sources. This diverse data is then structured and consolidated within a central database, serving as the system's primary data source. This unified dataset is subsequently fed into machine learning algorithms to cluster similar machines and identify their key properties based on data from existing machines. Finally, these clustered data points and classifications are utilized by large language models to generate human-readable descriptions in multiple languages, resulting in the automatic generation and export of complete manuals in the desired format.

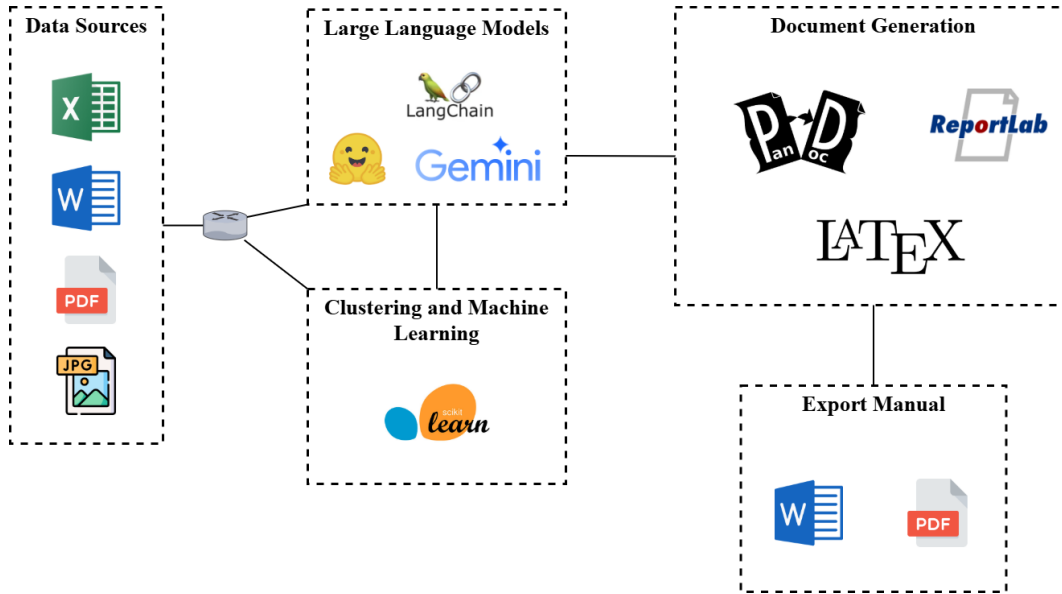


Figure 1: Proposed Framework.

### 2.1 Data Sources

Developing comprehensive manuals for cooling machines necessitates a multi-source data gathering process. Information is drawn from various sources including the company's Enterprise Resource Planning (ERP) system, engineering team-provided Excel spreadsheets, descriptive Word documents, and PDFs containing specifications and diagrams from previous models.

This diverse data compilation ensures the manual's accuracy and completeness, encompassing both technical specifications and operational details. The integration of data from disparate formats requires careful organization and verification to produce a user-friendly and reliable document.

### 2.2 Large Language Models

Large Language Models (LLMs) can significantly automate machine manual generation by processing technical specifications and transforming them into user-friendly documentation. LLMs can understand complex technical

jargon and translate it into plain language, generate different versions tailored to various user skill levels, and even incorporate images and diagrams based on provided data. This automation can be achieved using Python packages like Langchain, Hugging Face's Transformers, and Google's Gemini, which provide the necessary tools for interacting with and fine-tuning LLMs for this specific task.

Langchain offers tools to orchestrate the LLM workflow, connecting it to external data sources like databases containing machine specifications. Hugging Face provides access to a wide range of pre-trained LLMs and tools for fine-tuning them on specific datasets of machine documentation to improve accuracy and style. Gemini, a powerful LLM, can directly generate and structure the manual text, potentially integrating multimedia elements. These packages work together or independently to create a complete pipeline for automatic machine manual generation, from data ingestion and processing to final document creation.

## 2.3 Clustering and Machine Learning

Clustering in machine learning is an unsupervised learning technique that groups similar data points together into clusters. The goal is to find inherent structure in the data without pre-defined labels. In the context of cooling and temperature control machines, clustering could analyze data like energy consumption, operational parameters (temperature settings, run times, etc.), maintenance history, and error logs. By clustering these machines based on their feature similarities, one could identify groups of machines with similar performance characteristics, potentially revealing patterns in machine efficiency, failure modes, or optimal operating conditions. This allows for more targeted maintenance, optimized resource allocation, and improved system-wide performance analysis.

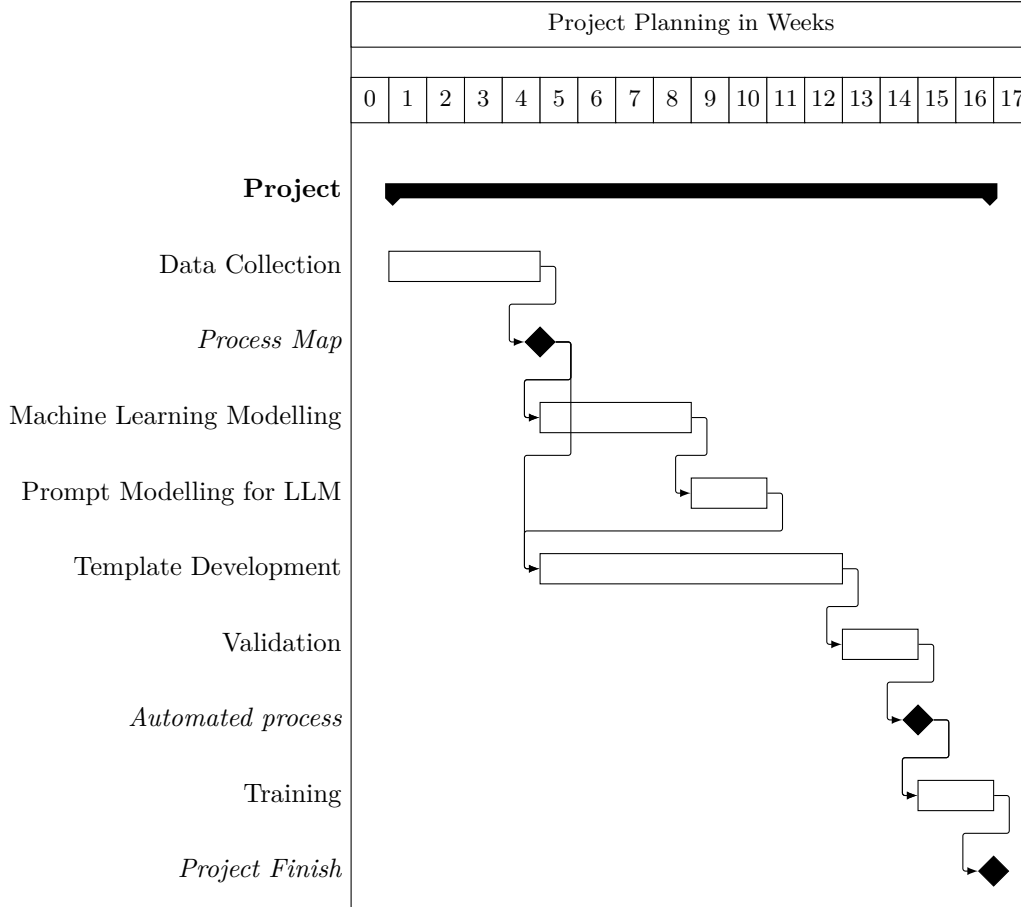
Scikit-learn (sklearn) is a popular Python library for machine learning that provides various algorithms for data mining and data analysis. For clustering machines based on their characteristics, several algorithms within Scikit-learn could be applied. These include K-Means clustering (for partitioning data into k clusters), hierarchical clustering (building a hierarchy of clusters), DBSCAN (Density-Based Spatial Clustering of Applications with Noise, for identifying clusters of arbitrary shape), and Gaussian Mixture Models (assuming data points are generated from a mixture of Gaussian distributions). The choice of algorithm depends on the specific characteristics of the data and the desired outcome of the clustering analysis.

## 2.4 Document Generation

Generating documents in a specific format and template using Python involves leveraging libraries that handle document structure and formatting. These libraries allow for programmatic creation of documents by defining content, styles, and layouts within the code, ultimately producing output files such as PDFs or Word documents that adhere to the desired template. For this purpose, packages like Pylatex, ReportLab, and Pandoc can be used. Pylatex allows for the generation of LaTeX documents, providing control over the formatting through LaTeX commands embedded within Python code. ReportLab offers a more Pythonic approach, enabling the creation of PDFs directly from Python scripts using a drawing-based model. Pandoc acts as a universal document converter, allowing conversion between various formats like Markdown, LaTeX, and DOCX. By using Pandoc in conjunction with a templating engine, a Python script could generate a document in a specified format and then leverage Pandoc's conversion capabilities to output the desired file type according to a pre-defined template.

### 3 Timetable and Activities

"This section presents the timetable and task descriptions for the project development. The project is structured into six main steps, with two intermediate milestones. The total estimated duration is sixteen weeks, and the involvement of at least two researchers is required..



#### 3.1 Data Collection

The Data Collection phase will involve conducting interviews with relevant departments and manual developers. This will map the entire process, identify all components and stakeholders, define quality criteria, and gather necessary data for subsequent project phases. The deliverable for this phase, representing the first project milestone, is a comprehensive process map. This phase is estimated to require a minimum of four weeks.

#### 3.2 Machine Learning Modelling

The Machine Learning Modelling section encompasses data analysis of the previously collected dataset, followed by the application and fitting of machine learning algorithms to cluster machines and identify similarities. This crucial step, requiring a minimum of four weeks, provides the foundation for subsequent project phases. Its outputs—the resulting machine clusters and similarity analysis—are essential for accurate machine descriptions and information completion in the final report.

### 3.3 Prompt Modelling for LLM

Prompt modeling for the Large Language Model (LLM) will involve defining optimal prompts for each section of the manual. This process aims to identify the most effective prompting strategies and text inputs to generate accurate and consistent outputs aligned with the manual's requirements. The goal is to minimize inaccuracies and ensure textual coherence. This phase is allocated a minimum of two weeks.

### 3.4 Template Development

The Template Development phase will encompass the design and creation of LaTeX and PyLaTeX templates for manual generation. This is crucial for ensuring consistent quality and a standardized format across all manuals. Automated generation via these templates will streamline the process. This phase is estimated to require a minimum of nine weeks.

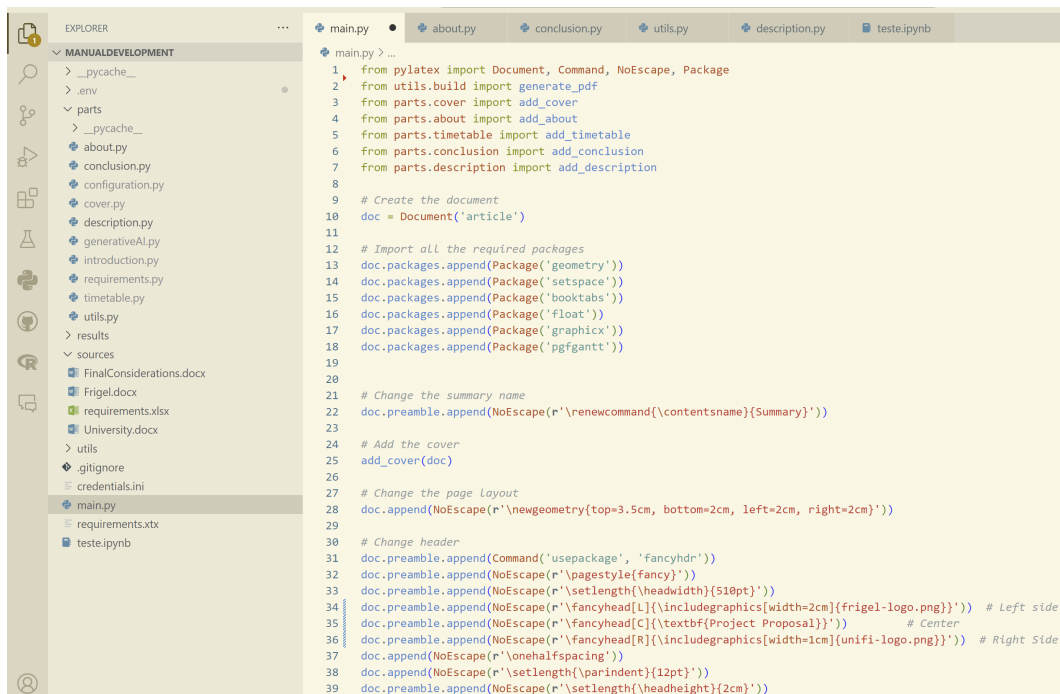
### 3.5 Validation and Training

The project includes a two-week validation phase, conducted collaboratively with a company intern, to assess the generated manual's quality and implement necessary revisions. Following validation, a four-week training program will be delivered to the manual development department. This training will cover the developed solution's functionality, operation, and the revised process implementation, encompassing two weeks of theoretical instruction and two weeks of practical application.

## 4 Final Considerations

The proposed project has the potential to significantly enhance both the efficiency and speed of manual development processes. Currently, these processes involve several time-consuming manual tasks that do not add value to the final product. Technicians often spend considerable time adjusting templates and aggregating information — activities that could be easily handled by software. Also, existing Generative AI platforms and APIs are already capable of structuring ideas, correcting grammar, and improving the overall coherence of texts.

As a simple demonstration of the model's effectiveness, this entire document was developed using it. The template was generated by combining Python and LaTeX code. The figure below shows the working environment of this project. You can find the repository with all codes here.



```

1  from pylatex import Document, Command, NoEscape, Package
2  from utils.build import generate_pdf
3  from parts.cover import add_cover
4  from parts.about import add_about
5  from parts.timetable import add_timetable
6  from parts.conclusion import add_conclusion
7  from parts.description import add_description
8
9  # Create the document
10 doc = Document('article')
11
12 # Import all the required packages
13 doc.packages.append(Package('geometry'))
14 doc.packages.append(Package('setspace'))
15 doc.packages.append(Package('booktabs'))
16 doc.packages.append(Package('float'))
17 doc.packages.append(Package('graphicx'))
18 doc.packages.append(Package('pgfgantt'))
19
20
21 # Change the summary name
22 doc.preamble.append(NoEscape(r'\renewcommand{\contentsname}{Summary}'))
23
24 # Add the cover
25 add_cover(doc)
26
27 # Change the page layout
28 doc.append(NoEscape(r'\newgeometry{top=3.5cm, bottom=2cm, left=2cm, right=2cm}'))
29
30 # Change header
31 doc.preamble.append(Command('usepackage', 'fancyhdr'))
32 doc.preamble.append(NoEscape(r'\pagestyle{fancy}'))
33 doc.preamble.append(NoEscape(r'\setlength{\headwidth}{510pt}'))
34 doc.preamble.append(NoEscape(r'\fancyhead[L]{\includegraphics[width=2cm]{frigel-logo.png}}')) # Left side
35 doc.preamble.append(NoEscape(r'\fancyhead[C]{\textbf{Project Proposal}}')) # Center
36 doc.preamble.append(NoEscape(r'\fancyhead[R]{\includegraphics[width=1cm]{unifi-logo.png}}')) # Right Side
37 doc.append(NoEscape(r'\onehalfspacing'))
38 doc.append(NoEscape(r'\setlength{\parindent}{12pt}'))
39 doc.preamble.append(NoEscape(r'\setlength{\headheight}{2cm}'))

```

The “About the Project” section was developed by a mix of GenAI (using Gemini API) and personal texts.



```

parts > about.py > ...
1
2 from pylatex import Section, NoEscape, Subsection, Subsubsection
3 from parts.generativeAI import makeQuestion
4 from parts.utils import read_docx
5
6
7 def add_about(doc):
8     with doc.create(Section('About the Project')):
9         question = """Please, generate a paragraph to introduce a project
10        between the Fashion for Future, a research lab in the University
11        of Florence, and a Company called Frigel, which produces machines
12        for cooling and temperature controlling, to automatize the process
13        of manual development for the new machines."""
14        makeQuestion(doc, question)
15
16
17
18     with doc.create(Subsection('About the Team')):
19         text = read_docx("sources/University.docx")
20         doc.append(NoEscape(text))
21

```

On the other hands, for the “Proposed Solution” section, besides the image that was fully developed manually, all the texts were generated by GenAI.

<pre> 1 from pylatex import Section, NoEscape, Figure, Subsection 2 from parts.generativeAI import makeQuestion 3 4 def add_description(doc): 5     with doc.create(Section('Proposed Solution')): 6 7         question = """ 8         Please, generate an introduction paragraph for a section that 9         propose a solution to automatize the process of generating 10        manuals. 11 12        In the second paragraph, please, start by explaining that the 13        figure below shows the proposed framework solution. Then start 14        describing it briefly: the first step is the data collection from 15        diverse sources within the company. Then, the data collected is 16        structure in a common database that will feed all system. This 17        data will be inputted in machine learning algorithms to clusterize 18        the new machines and specify its properties according to the data 19        available from previous machines. All these data and classifications 20        will be used by large lange models to write all the descriptions in 21        an undestainable human language, with the possibility of doing it 22        in multiple languages and, in the end, the manual will be generated 23        automatically, exported in the desired format.. 24        """ 25        makeQuestion(doc, question) 26 27        with doc.create(Figure(position="h")) as fig: 28            fig.add_image('images/Tools.png', width=NoEscape(r'\0.8\textwidth')) 29            fig.add_caption('Proposed Framework.') 30 31        with doc.create(Subsection('Data Sources')): 32            question = """ 33            Please, create a text with maximum of two paragraphs to explain 34            that the process of developing manuals for cooling machines requires 35            several data from differente sources, such as the company ERP, 36            excel files with data provided by the engineering teams, word 37            files with descriptions, pdf files with previous machines and relevant 38            figures to describe the machine.""" 39            makeQuestion(doc, question) </pre>	<pre> 4 def add_description(doc): 5 6     with doc.create(Subsection('Large Language Models')): 7 8         question = """ 9         Please, create a text with maximum of one paragraph to explain how useful 10        Large Language Models can be used to generate automatically machine manuals. 11        Cite that as example, python packages could be used, such as langchain, 12        hugging face or Gemini. 13 14        In the next paragraph, please, explain what are and how the packages 15        Langchan, Hugging Face and Gemini could be used.""" 16        makeQuestion(doc, question) 17 18     with doc.create(Subsection('Clustering and Machine Learning')): 19 20         question = """Please, explain in one paragraph what is clustering 21        in the machine learning context. Explain how it could be used to 22        cluster machines (such machines are used for cooling and temperature 23        control systems) to understand similarities between them to analyze 24        data and describe the machine characteristics. 25 26        In the next paragraph, please, explain what is the package Scikitlearn 27        and which algorithms could be used for such aim.""" 28        makeQuestion(doc, question) 29 30     with doc.create(Subsection('Document Generation')): 31 32         question = """Please, explain in one paragraph how a document can 33        be generated in the correct format and template using python. 34        Finish your paragraph saying that for that purpose, packages like 35        Pylatex, Reportlab and Pandoc can be used. 36 37        In the next paragraph, please, explain how the packages Pylatex, 38        Pandoc and Report Lab could be used for such aim.""" 39        makeQuestion(doc, question) </pre>
--	---

Finally, this conclusion section was fully written in a word file, with the figures placed in the desired positions, and the code successfully collected all the information and placed in the desired positions.

