```
\documentclass[12pt,a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}
\usepackage[margin=2.5cm]{geometry}
\usepackage{inconsolata} % Pleasant monospaced font
\usepackage{hyperref}
\hypersetup{colorlinks=true, linkcolor=MidnightBlue, urlcolor=MidnightBlue}
\usepackage{fancyvrb} % for displaying code

\title{PROJECT ASSIMOV: A Manifesto for Educators in the Age of Transformers}
\author{Dídac Valenciano Gener}
\date{May 2025}

\begin{document}
\section*{2025 is — and will be — the year of ChatGPT.}.
```

In Iberian and Mediterranean Europe within the EU, with a latent mecha subconscious still noticeable today, *Mazinger Z* left a deep cultural, emotional, and multi-domain mark that explains certain shared behaviors of *Generation X*. If you're between 45 and 60 years old, you've already thought "*¡Puños fuera!*", recalled how you survived *Blandiblú*, the *Famobil Clicks*, and watched violent cartoons without becoming a psychopath.

In the United States, *Hasbro* initiated the generational transition with the *Transformers* saga, which reached its cultural peak in 2007 when, right at the beginning of the footage, the *Decepticons* attacked a U.S. military base in the Qatari desert. Meanwhile, *Megan Fox* and *Shia LaBeouf* left us breathless. It was at that moment that a new image stuck in the subconscious of another generation: a small *Decepticon* — a *Transformer*, but one of the bad guys — hidden among the protagonists' belongings, absorbing metal from its surroundings to modify itself, stretching its forms, elongating and adapting until it reached its objective.

Ten years later, as the result of a long journey of research by multiple academic and private entities, in parallel with the world of entertainment, *Google* publicly introduced in 2017 an algorithmic technology that replicates itself, stretches and expands its own structure, modifying and elongating until, in a purely probabilistic way, it achieves its goal. What a surprise — and what a coincidence — that this technology, previously unknown, was named: *transformers*.

"Attention is All You Need", published by *Google* in 2017, caused quite a stir — especially on *Twitter* (or *Grok*, or whatever it's called this week). But it wasn't until *OpenAI* took the baton that they placed, quite literally, a small *transformer* in front of us. One that, every time we speak to it, *tokenizes* our language, and like that *Decepticon* in the Qatari desert, stretches, adapts and reorganizes its form to generate, in a purely probabilistic way, a response optimized for the situation at hand.

And the beauty of it, my friends, is that it simulates so well, that if we understand *artificial intelligence* as a functional simulation of intelligence... well, there might be other implementations, but this one — definitely — is one.

What follows is another implementation. One that, like the algorithms from *OpenAI*, stretches, adapts and reorganizes its own structure to generate, in a purely probabilistic way, an optimal response to the situation at hand: efficiently creating didactic content, an article, a book. All so that we, *teachers*, may have our own weapons for the battle that awaits us.

AI-enhanced LaTeX generation pipeline

github.com/nassaba/project-assimov - AI-enhanced content generation pipeline

```
# Phase 0: Interpret prompt and assign roles with grammarnaut + llm_router

# Phase 1: Generate structural skeleton with yaml_generator using Claude or o3 or ...

# Phase 2: Expand each section with role-assigned LLMs (GPT-4o, MythoMax, Lit-6B...) using cached YAMLs

# Phase 3: Review narrative coherence, argument and tone with o3 or Claude using section context

# Phase 4: Compile formatted output into .tex with writer or translator

# Phase 5 (optional): Final stylistic polish by purist LLM or human reviewer

# CONFIGURATION
# Load API key (replace with your secure method)

import openai
from pathlib import Path
import time


# AI-enhanced content generation pipeline (Assimov)

def phase_0_prompt_intake():
    """Receive user prompt and determine functional roles + LLMs."""
    prompt = user_input()
    roles = assign_roles(prompt)  # via gramaneute.py -> llm_router.py
    llms = select_models(roles, config="config.yaml")
    return plan(roles, llms)

def phase_1_generate_structure(plan):
    """Create YAML skeleton per section with titles, themes, targets."""
    yamls = []
    for section in plan.sections:
    yamls.append(generate_yaml(section))  # via yaml_generator.py or real LLM
    return yamls

```

```python
def phase_2_expand_sections(yamls):
    """Expand each section via assigned LLM, using cached context."""
    content = []
    for yaml in yamls:
        llm = yaml.assigned_model
        section_text = expand_from_yaml(yaml, llm=llm, use_cache=True)
        content.append(section_text)
    return content

def phase_3_refine_coherence(content):
    """Polish narrative flow, argument structure, and tonal coherence."""
    polished = []
    for section in content:
        coherent = enforce_coherence(section)  # coherence supervisor
        refined = polish_argument(coherent)
        toned = adjust_tone(refined)
        polished.append(toned)
    return polished

def phase_4_compile_tex(polished):
    """Format polished content into a .tex document with structure."""
    doc = initialize_tex()
    for section in polished:
        doc.append(format_section(section))  # via writer.py or translator.py
    return doc

def phase_5_final_review(tex_document):
    """Optional: stylistic and poetic pass by a purist LLM or human."""
    reviewed = manual_review(tex_document)
    return reviewed

# Main pipeline execution

if __name__ == "__main__":
    plan = phase_0_prompt_intake()
    yamls = phase_1_generate_structure(plan)
    raw_content = phase_2_expand_sections(yamls)
    refined = phase_3_refine_coherence(raw_content)
    tex = phase_4_compile_tex(refined)
    final_output = phase_5_final_review(tex)
    save(final_output, "output/document.tex")
```