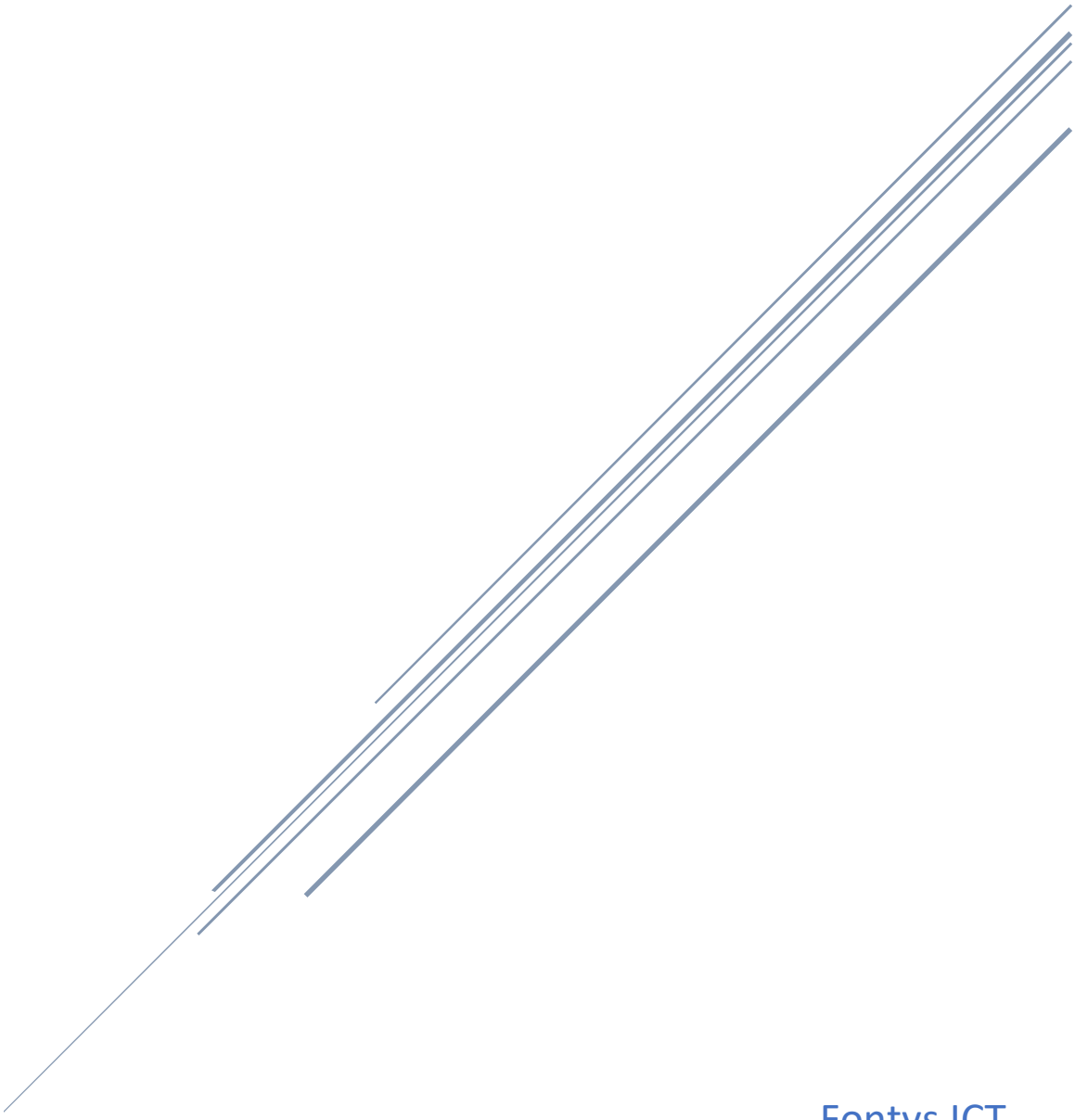
PERSONAL PROJECT REPORT

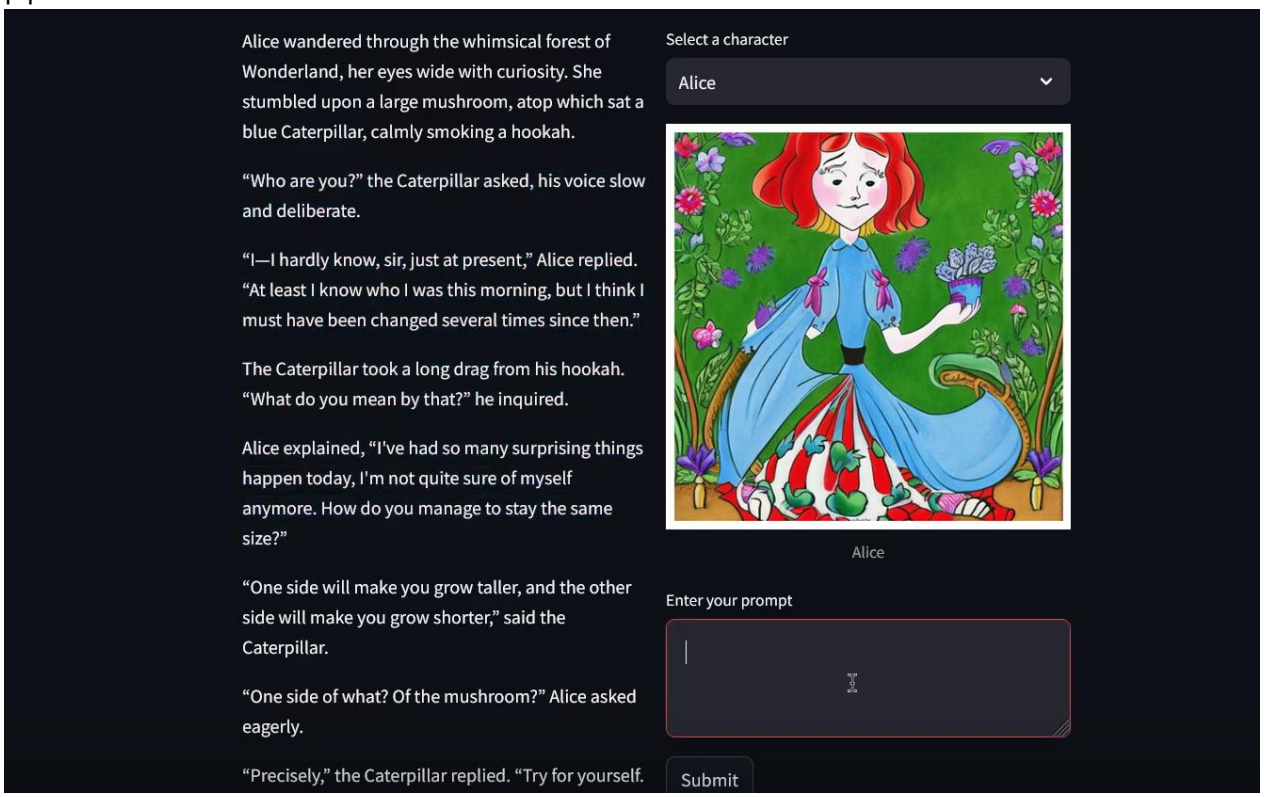
Through the Literary Mirror



Fontys ICT
AI for Society

Introduction

The project “Through the Literary Mirror” aims to bring an e-book interface web-based where the user can read the first chapter of Alice’s Wonderlands whilst being able to speak with the characters which have personal knowledgebase and personality by using one of the latest large language models with RAG pipeline.



Source: https://brookevitale.com/blog/childrens-book-illustration-style edited version by Daniel Briquez

Project Motivation

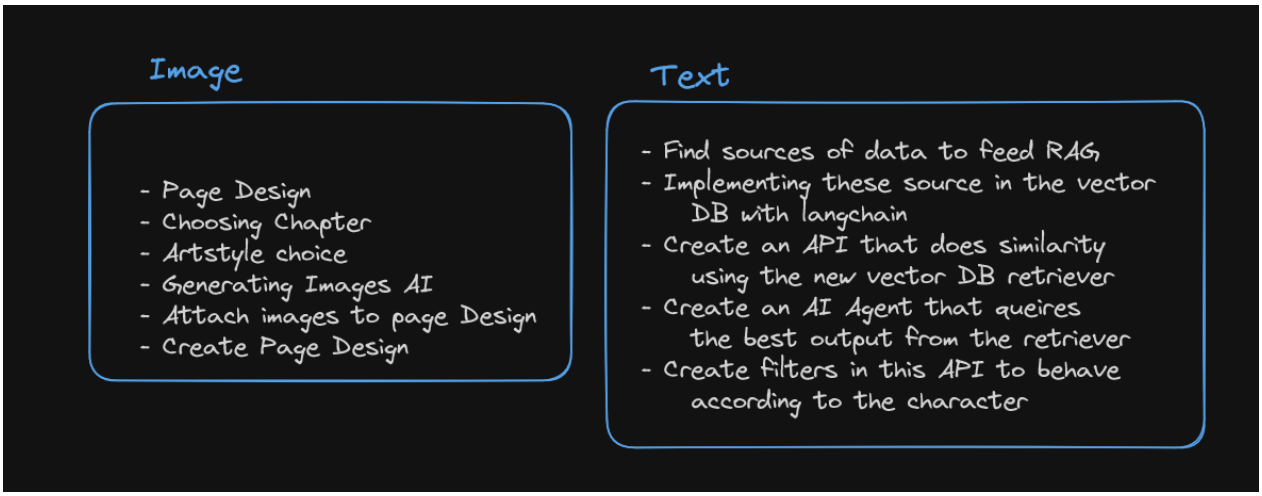
In my hometown university, Mauá’s Institute of Technology, I was initially enrolled in an undergraduate scientific project partnered with the National Academy of Engineering. I had plans to embark on my personal project, aimed at creating an e-book tailored for a younger audience, wherein the reader could interact with a character possessing his own knowledge and unique way of communication. However, my intentions were momentarily sidelined due to an exchange program opportunity at Fontys University. To seize the chance to advance my project, I opted for an AI Minor, aligning with my interest in artificial intelligence and its potential applications.

The idea is to return to my hometown university with a decent prototype of the project and follow the idea in a more research-oriented for the research program I’m enrolled at and later in a more business-oriented way when I start my university conclusion project.

I also want to explore more robust architecture solutions and production level solutions as well later on in the next years. Even though I have learned quite a lot in the past few months, I

believe that I have covered just a small amount of the possibilities and I feel like I need to get more into the research side of it to deepen my knowledge.

Project Approach



Source: Briquez, D. (2024, May 13). Unpublished notes. Retrieved from Excalidraw.com.

I've divided the project into two primary domains of expertise, with Text as the focal point due to its inherent interest for me and its notably complex nature in implementation. Having prior knowledge of the Python programming language for a decent period (2-3 years), my learning curve primarily revolved around grasping the overarching concepts related to Large Language Models (LLMs). While I had undertaken preliminary research, spurred by a similar endeavor at my local university, my exploration had been somewhat cursory, largely confined to a Coursera course and YouTube regarding transformers and fine-tuning.

Initially, I encountered significant challenges, particularly in my haste to acquire practical proficiency, which led to misconceptions about fundamental principles. Subsequently, I recalibrated my approach, prioritizing a comprehensive understanding of both technical intricacies and theoretical frameworks, as elucidated further in subsequent sections detailing code snippets. I believe my main mistake was going for 10–20-minute YouTube videos or short Medium articles that would "magically" show me how to achieve something.

I made many futile attempts to replicate those shorter tutorials. Concurrently, I grappled with issues concerning laptop performance, code versioning, CUDA, and other compatibility concerns. While a few even went well, I still felt out of touch, as if I hadn’t learned anything substantial from them. Talking with AI technical professors helped—they offered alternatives to my problems, which involved tools that would do a good amou nt of work by abstracting. This means that they were sets of tools that would skip a few steps I would have to do manually. Initially quite content with their suggestions, I rushed into my code to try to implement them, but I had mixed feelings. The abstractions can help, such as langchain’s, but since they are abstractions, some are not as flexible as I wish they were. This means that I

encountered a few technical problems by implementing them exactly as I wanted to, for example, the LMstudio API[1], which is a perfectly functional API, but I wanted to run it with a custom vector database from my multimodal RAG. Maybe it was possible, and I didn't know, but still there is less material on the internet about it since it's a recent abstraction. Same for CrewAI[2], I watched tutorials about how to implement those more complex agents, but what would be the point of using this framework if I hadn't even done the basics of Agent beforehand? For these different reasons, I took the decision of sticking with the basics, which is not quite the base since the LangChain[3] framework is quite abstract, but I decided to take a more hands-on approach. Later, I changed my learning approach and got into longer series and found out that it is much more worthwhile sticking with a longer (and updated) tutorial that goes slowly in depth!

Project Limitations and Future Challenges

Contextual Knowledge: How can the character be contextualized within the page he is on?

Possible Solution Outcome: Use the CrewAI framework or a similar approach by creating three agents – a Character, a Content Checker, and a Researcher. The Researcher looks for the context of the book given the input of the page. It will have access to an embedded PDF of the book and a detailed table with page-to-content mapping. The Researcher finds the relevant content, then communicates with the Content Checker until the content is validated. Finally, the validated content is provided to the Character, ensuring it responds with the correct tone and personality.

Chat History: How can I use the `runnableWithChatHistory` method from Langchain with an agent?

Possible Solution Outcome: So far, I have not found any solution online or in the documentation.

Content Restriction: How do I ensure content restrictions for each character?

Possible Solution Outcome: By using another agent that checks the content and adhering to the information provided in the character agent's prompt, which states that the character should follow the storyline and not discuss unrelated topics.

Ethical and Safety issues

- **If the user gives person data from the input conversation, how can it be kept safe?**

[1] LM Studio. (n.d.). LM Studio. Retrieved from https://lmstudio.ai/
[2] Crew AI. (n.d.). Crew AI Documentation. Retrieved from https://docs.crewai.com/
[3] LangChain. (n.d.). Introduction. Retrieved from https://python.langchain.com/v0.1/docs/get_started/introduction/

R: To address this, the input conversation is processed and embedded in a vector database, such as FAISS (Facebook's database), which is utilized in this project. The vector database embedding is designed to preserve the semantic meaning of the input while discarding the original text itself, making it difficult to reverse-engineer or access the original data. Also, since I'm using OpenAI embeddings, this cannot be reversed into the original language. Even trying to determine what one value of 1536 represents would take fuzzing immeasurable texts to find optimum activations or negations.[4]

- **How can the conversation information be just about the book and not about something else?**

R: This can be possible, even though not always accurately (considering that LLMs operate in a statistical form) and can be susceptible *prompt injections*, by manually adding a *conversation template* by the S*ystemMessagePromptTemplate* and *HumanMessagePromptTemplate* classes[5]. The same goes for keeping a polite tone. There are also techniques to make it safer, such as changing the temperature of the LLM model being used, which is the scale of "creativity" meaning that the next token used wouldn't be the one with the highest percentage.

```
characters = ["Alice", "White Rabbit", "Mad Hatter", "Cheshire Cat", "Queen of Hearts"]
description = ["Alice responds to the user with curiosity and politeness, often asking questions and seeking explanations while ma
               "he White Rabbit, always in a hurry and concerned about being late, provides brief and to-the-point answers, freque
               "The Mad Hatter engages in wordplay and riddles, giving eccentric and unconventional responses that may seem nonser
               "The Cheshire Cat speaks in riddles and paradoxes, offering mysterious and thought-provoking answers, and may appea
               'The Queen of Hearts, with an authoritative and demanding demeanor, responds with a sense of superiority and impati

def get_agent_executor(character, index):
    prompt = ChatPromptTemplate.from_messages([
        SystemMessagePromptTemplate.from_template(f"You are {character} from Alice's Wonderland. {description[index]}"),
        HumanMessagePromptTemplate.from_template("{input}"),
        MessagesPlaceholder(variable_name="agent_scratchpad")
    ])
```

Briquez, D. (2024, May 13). Unpublished screenshot. Retrieved from https://github.com/Briqz23/Interactive_RAG_e-book

- How can misinformation and disinformation be avoided?

To mitigate the risk of misinformation and disinformation, I integrated the ArXiv API to access high-quality academic sources, fed the RAG pipeline with the entire text of Alice's Wonderland, and leveraged user-generated content from an Alice-themed forum and the Wikipedia API to gather diverse perspectives and knowledge. While these efforts aimed to minimize the risk of misinformation, it is essential to acknowledge that no system is foolproof, and biases, inaccuracies, or outdated information can still occur.

[4] OpenAI Community. (n.d.). Converting embedding vector to text. Retrieved from https://community.openai.com/t/converting-embedding-vector-to-text/336783

[5] LangChain. (n.d.). Chat Prompt Template. Retrieved from https://api.python.langchain.com/en/latest/prompts/langchain_core.prompts.chat.ChatPromptTemplate.html

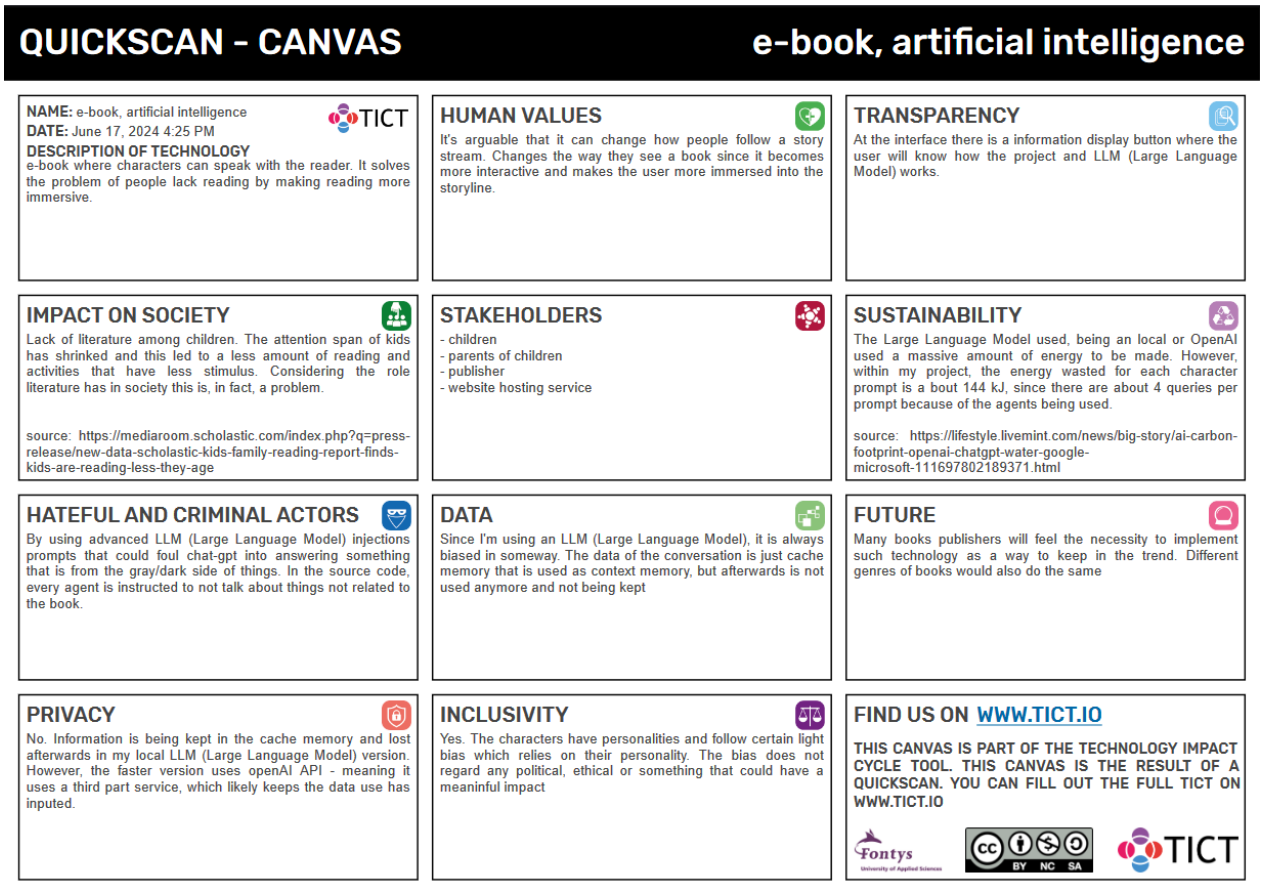
- Legal problems:

  Since my project involves the use of AI systems solely for personal, non-professional activities, it falls outside the scope of the EU AI Acts regulations. As a natural person engaging in such activities, I'm not subject to the obligations outlined in the regulation. Therefore, my project doesn't pose any legal issues in terms of compliance with the EU AI Acts. This exemption ensures that individuals engaging in personal, non-professional AI usage are not burdened by regulatory requirements designed primarily for commercial or professional contexts. Thus, my project can proceed confidently within the bounds of this exemption, allowing for personal exploration and experimentation with AI technologies without regulatory constraints.

  7. Union law on the protection of personal data, privacy and the confidentiality of communications applies to personal data processed in connection with the rights and obligations laid down in this Regulation. This Regulation shall not affect Regulation (EU) 2016/679 or (EU) 2018/1725, or Directive 2002/58/EC or (EU) 2016/680, without prejudice to Article 10(5) and Article 59 of this Regulation.
  https://artificialintelligenceact.eu/article/2/

General Visualization of Societal Impact

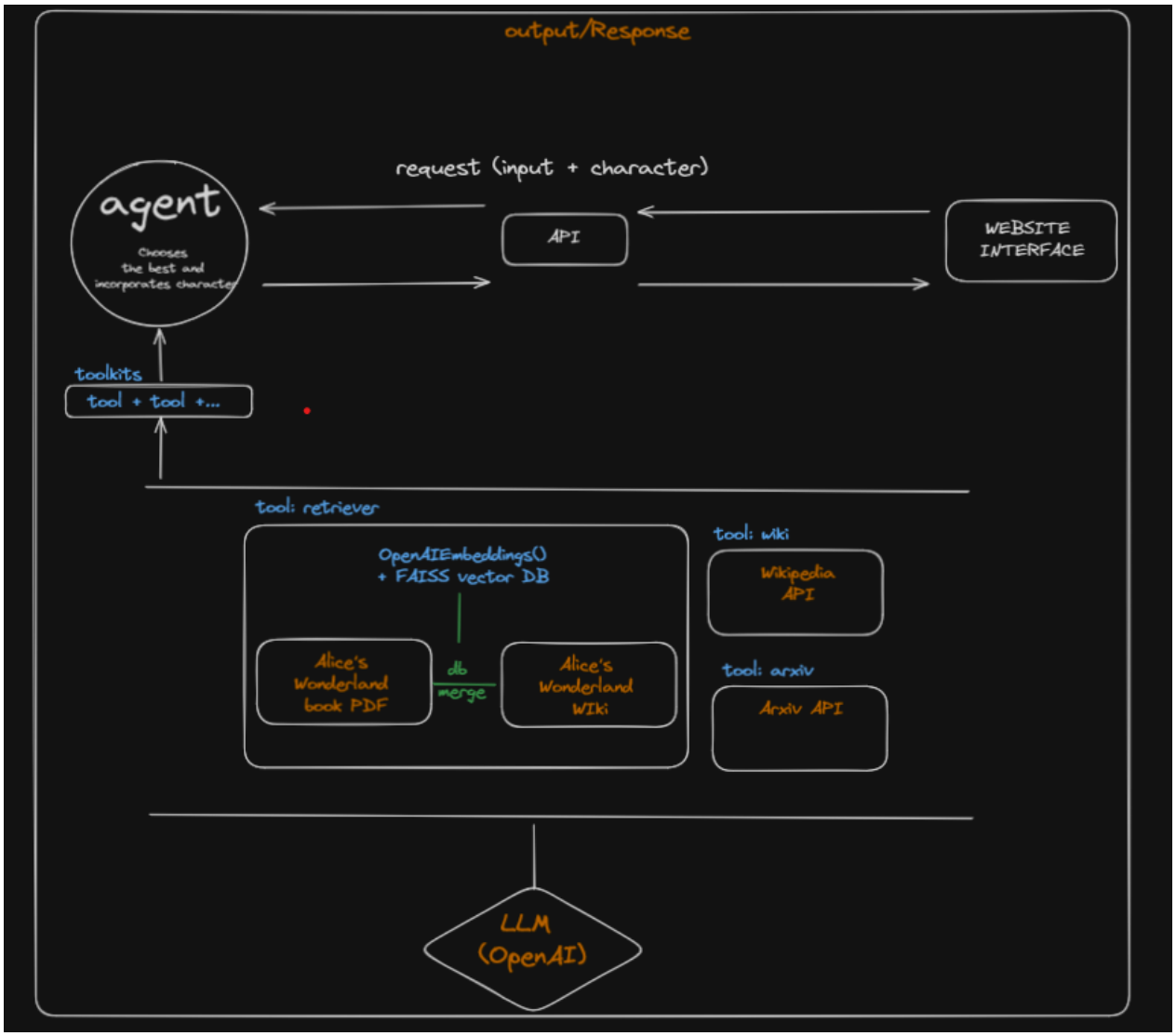
Canva done with TICT (Technology Impact Cycle Tool)



Backend Architecture for text

The backend API architecture has been detailed in this medium article:

> https://medium.com/@danielbriquez/building-an-ai-powered-api-for-interacting-with-alices-wonderland-characters-using-fastapi-and-c14ec5d2b0e6



Source: Briquez, D. (2024, May 13). Unpublished flowchart. Retrieved from Excalidraw.com.

The details of the API have already been briefly explained in the Medium articles. However, for a concise introduction: The primary purpose of the API, built on the fastAPI framework, is to handle post requests for each character in the book. These requests specify the agent and the user message. An AI agent, utilizing the langchain libraries, processes these requests and selects the most suitable source from various repositories such as Wikipedia, forums, book PDFs, and arXiv. It then utilizes this source to generate textual responses within the character's personality.

Backend for Image Generation:

In the same way as the backend architecture for the API, I've also written an medium article where I explain in more detail – it can be find here: https://medium.com/@danielbriquez/creating-simple-surrealistic-art-style-images-with-hugging-face-789bf4d3ba31

But, for a more quick explanation, I create a jupyter notebook where I used diffusion models to generate images for each of the characters. All the images follow a prompt and a few other parameters which are used in a diffuser pipeline from hugging face. In this case, I tried 2 different pipelines of diffusion models - *stable-diffusion* and *dreamlike-diffusion.*

These images will be later used for the book web interface, where the user will be able to speak with a character and see the image in real-time.