UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica

Master's degree program in Computer Science

MASTER'S THESIS

ChatGPT-4: a teaching and learning tool for Computer Science education.

Student

Gianluca Giuseppetti

ID Number 280973

Supervisor

Prof. Henry Muccini

ACADEMIC YEAR 2022-2023

To those who use their skills to make the world a better place for everyone
and for future generations.

I thank my supervisor Prof. Henry Muccini for the qualities and availability demonstrated in guiding me in the drafting of this thesis.

I thank Prof. Giuseppe della Penna for the exemplary tutoring activity carried out during my course of study.

I thank my family and friends for supporting and putting up with me.

CONTENT

INTRODUCTION..4
CHAPTER 1 - Generative Artificial Intelligence (GAI) and ChatGPT-4 5
1.1 - GAI: basic concepts, background and historical notes. 5
1.2 - Transformer models, GPT model.......................................11
1.3 - ChatGPT: competitors, historical notes and best features 13
CHAPTER 2 - ChatGPT-4: a teaching tool for Computer Science education..........20
2.1 - State-of-the-art teaching features..20
2.2 - Case study: personal experience as substitute teacher in secondary school..........23
CHAPTER 3 - ChatGPT-4: a learning tool for Computer Science education 38
3.1 - State-of-the-art learning features 38
3.2 - Case study 1: self-tested code for React app in MVVM architecture pattern. 43
3.3 - Case study 2: Software Architectures Project.................................54
3.4 - Case study 3: how-to manage smartphone cameras in Android app. 77
CHAPTER 4 - Conclusion 82
4.1 - GAI tools open questions: ethic, safety, privacy and copyright concerns. 82
4.2 - Future scenarios and final considerations....................................83
BIBLIOGRAPHY..87
APPENDICES..90
A - GAI: regulations in major country 90
B - ChatGPT-4: common troubleshooting examples 94
C - ChatGPT-4: OpenAI prompt engineering guidelines..............................96
D - ChatGPT-4: terms of use, privacy policy and current legal issues 97

INTRODUCTION

This thesis is part of research on Generative Artificial Intelligence (GAI) applications within Computer Science (CS) education, serving as both a teaching tool for the educator and a learning tool for the student.

The objective of this thesis is to introduce the fundamental concepts and applications of GAI, with a specific focus on ChatGPT-4; it aims to outline current state-of-the-art of use of ChatGPT-4 in Computer Science education. Additionally, the thesis presents some personal case study that illustartes the use of ChatGPT-4 based on my dual experience, as both as student at this university and a substitute teacher at secondary technical institutes, in the subjects of Computer Science. It will conclude exploring potential future scenarios and final considerations.

In the first Chapter, I present an introduction to basic concepts, background and historical notes for GAI, including Transformer GAI models, Generative Pre-Trained (GPT) model, chatbot competitors and a detailed look at ChatGPT-4, focusing on its historical notes and best features.

In the second Chapter, I outline the current state-of-the-art for use of ChatGPT-4 as teaching tool and a personal case study as teacher. I used ChatGPT-4 to prepare teaching material, especially for the needs of BES and DSA students, by following the precious suggestions on how to use ChatGPT-4 for teaching, contained in the videos of Prof. Marc Alier reported by my supervisor.

In the third Chapter, I outline the state-of-the-art of use of ChatGPT-4 as learning tool and three personal case study as student. The first project involved, implementing a self-tested code for React app following the MVVM architecture pattern, based on an article published on martinfowler.com web site. The second, the Software Architectures course project and the third, a personal study about how to manage smartphone cameras in an android app.

In the fourth Chapter, I show GAI tools current open questions about ethic, safety, privacy and copyright concerns. I conclude by presenting future potential benefit and risks associated with incorporating GAI tools to enhance the teaching and learning experience in Computer Science education.

CHAPTER 1

Generative Artificial Intelligence (GAI) and ChatGPT-4

1.1 - GAI: basic concepts, background and historical notes.

Artificial intelligence (AI) is the field of Computer Science that study how to leverage computers to mimic human intelligence. It aims to create systems that can perform tasks that would typically require human intelligence, such as understanding natural language, recognizing patterns, solving problems, and making decisions.

Generative Artificial Intelligence (GAI) involves the creation of digital content, such as images, music, and natural language, through AI systems. The goal of GAI is to make the content creation process more efficient and accessible, allowing to produce high-quality content at a faster pace. GAI is achieved by extracting and understanding intent information from instructions provided by human and generating the content according to its knowledge and the intent information. In recent years, large-scale systems have become increasingly important in GAI as they provide better intent extraction and thus, improved generation results. With the growth of data and the size of the models, the distribution that the model can learn becomes more comprehensive and closer to reality, leading to more realistic and high-quality content generation. GAI models can be categorized into two types: unimodal models and multimodal models. Unimodal models receive instructions from the same modality (text-only, voice-only or image-only) as the generated content modality, whereas multimodal models accept cross-modal instructions and produce results of different modalities.

The journey from traditional rule-based AI to GAI represents a shift from rigid, programmed systems to flexible, learning, and creative systems. Each stage of AI's evolution builds upon the last (Figure 1.1), with GAI representing one of the most exciting frontiers in AI research and application today.

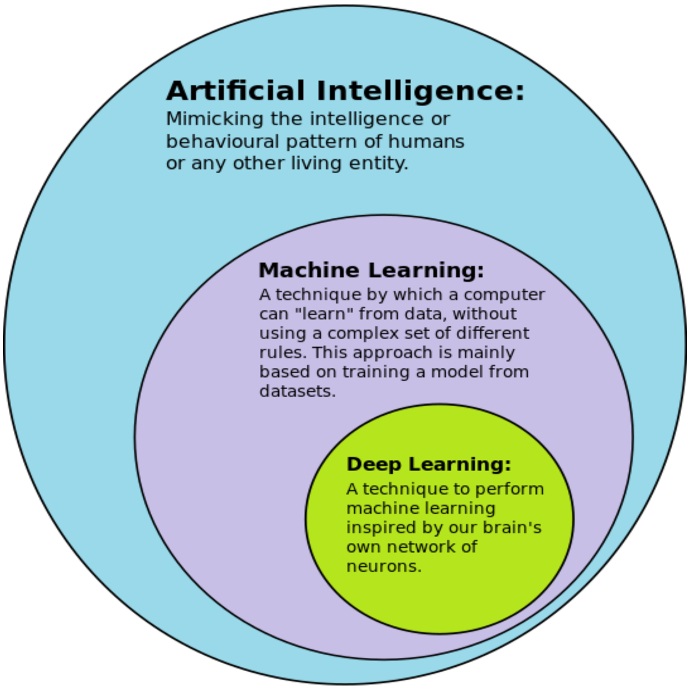

Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.

Figure 1.1- A graphic showing how AI, machine learning, and deep learning relate to each other. [1]

In table 1.1, I provide overview of some different techniques along with their descriptions from traditional rule-based systems to the avant-garde deep learning models.

Technique	Description
Rule-Based Systems	Early AI systems were primarily rule-based, meaning they operated by following explicitly programmed instructions and rules.
Expert Systems	These are computer systems that emulate the decision-making ability of a human expert. They use "if-then" rules to solve specific problems within a certain domain.
Fuzzy Logic Systems	Use fuzzy logic rather than binary (true/false) logic, allowing for more nuanced decision-making based on degrees of truth.
Case-Based Reasoning	Solve new problems based on the solutions of similar past problems. *(technique used in my master's thesis in Information Science in 1999)*
Search and Optimization Algorithms	Use systematic methods to explore possible solutions and choose the optimal one.
Supervised Learning	Involves training a model on labelled data, meaning that each training example is paired with an output label. The model makes predictions or decisions based on input data and is corrected when its predictions are incorrect.
Unsupervised Learning	Involves modelling with datasets that contain no labels. The system tries to learn the patterns and the structure from the data without any supervision.
Semi-Supervised Learning	Falls between supervised learning and unsupervised learning. Models are trained on a dataset that contains both labelled and unlabelled data. Generally, a small amount of data is labelled while a large amount of data is unlabelled.

Reinforcement Learning	The algorithm learns by interacting with an environment and receiving feedback in the form of rewards or penalties. It learns to achieve a goal or maximize some notion of cumulative reward.
Self-Supervised Learning	Self-supervised learning is a type of unsupervised learning where the data itself provides supervision. It doesn't require external labels, rather it uses the inherent structure of the data to learn representations and features. Algorithms are designed to predict certain parts of the data from other parts, hence learning useful data representations.
Recurrent Neural Networks (RNNs)	RNNs are a class of neural networks that are particularly well-suited for processing sequential data. Ideal for tasks where the sequence of data points is important, such as in time series prediction, natural language processing, speech recognition, and more.
Convolutional Neural Networks (CNNs)	CNNs use convolutional layers to process input data in a hierarchical way to detect patterns, like edges, textures, images, and more complex patterns in higher layers.
Generative Adversarial Networks (GANs)	GANs consist of two networks—a generator and a discriminator—that are trained simultaneously through adversarial training. The generator creates new content, while the discriminator evaluates it. The process continues until the generator produces high-quality content that the discriminator can hardly distinguish from real data.
Variational Autoencoders (VAEs)	VAEs are a type of autoencoder that adds a probabilistic spin to autoencoding. They are used to generate new content that's similar to the training data.
Transformer Models (including GPT models)	Transformer models use attention mechanisms to capture the dependencies between words or sub-words in the data. These models can be used for both discriminative tasks (like classification) and generative tasks (like text generation).

Table 1.1- Overview of AI models evolution.

Each technique represented in Table 1.1, has specific use cases, advantages, and disadvantages. To choose an appropriate model, it's essential to consider the nature of data, the problem to solve, and the computational resources at disposal.

GAI models process usually consists of two steps: extracting intent information from human instructions and generating content according to the extracted intentions. However, the paradigm of GAI models containing the above two steps is not entirely novel, as demonstrated by previous studies. The core advancements in recent GAI compared to prior works are the result of training more sophisticated GAI models on larger datasets, using larger foundation model architectures, and having access to extensive computational resources. In addition to the benefits brought by the increase in data volume and computational power, researchers are also exploring ways to integrate new technologies with GAI algorithms. For example, utilizing reinforcement learning from human feedback to determine the most appropriate response for a given instruction, improve model's reliability and accuracy over time. This approach allows to better understand human preferences in long dialogues. Meanwhile, in computer vision, stable diffusion, has also shown great success in image generation. Unlike prior methods, GAI diffusion models can help generate high-resolution images by controlling the trade-off between exploration and exploitation, resulting in a harmonious combination of diversity in the generated images and similarity to the training data. In addition to the benefits brought by the increase in data volume and computational power, researchers are also exploring ways to integrate new technologies with GAI algorithms.

GAI models have a long history (Figure 1.2) in artificial intelligence [2], dating back to the 1950s with the development of Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs). These models generated sequential data such as speech and time series. However, it wasn't until the advent of deep learning that GAI models saw significant improvements in performance. In early years of deep GAI models, different areas do not have much overlap in general. In natural language processing (NLP), a traditional method to generate sentences is to learn word distribution using N-gram language modelling and then search for the best sequence. However, this method cannot effectively adapt to long sentences. To solve this problem, recurrent neural networks (RNNs) were later introduced for language modelling tasks, allowing for modelling relatively long dependency. This was followed by the development of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which leveraged gating mechanism to control memory during training. These methods can attend to around 200 tokens in a sample, which marks a significant improvement compared to N-gram language models. Meanwhile, in computer vision (CV), before the advent of deep learning-based methods, traditional image generation algorithms used techniques such as texture synthesis and texture mapping. These algorithms were based on hand-designed features and were limited in their ability to generate complex and diverse images. In 2014, Generative Adversarial Networks (GANs) was first proposed, which was a significant milestone in this area, due to its impressive results in various applications. Variational Autoencoders (VAEs) and other methods like diffusion GAI models have also been developed for more fine-grained control over the image generation process and the ability to generate high-quality images. The advancement of GAI models in various domains has followed different paths, but eventually, the intersection emerged: the transformer architecture. Introduced by Vaswani et al. for NLP tasks in 2017, Transformer has later been applied in CV and then become the dominant backbone for many GAI models in various domains. In the field of NLP, many prominent large language models, e.g., Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT), adopt the transformer architecture as their primary building block, offering advantages over previous building blocks, i.e., LSTM and GRU. In CV, Vision Transformer

(ViT) and Swin Transformer later takes this concept even further by combining the transformer architecture with visual components, allowing it to be applied to image based downstream. Except for the improvement that transformer brought to individual modalities, this intersection also enabled models from different domains to be fused together for multimodal tasks. One such example of multimodal models is CLIP. CLIP is a joint Vision-Language (VL) model that combines the transformer architecture with visual components, allowing it to be trained on a massive amount of text and image data. Since it combines visual and language knowledge during pre-training, it can also be used as image encoders in multimodal prompting for generation. In all, the emergence of transformer-based models revolutionized AI generation and led to the possibility of large-scale training. In recent years, researchers have also begun to introduce new techniques based on these models. For instance, in NLP, instead of fine-tuning, people sometimes prefer few-shot prompting, which refers to including a few examples selected from the dataset in the prompt, to help the model better understand task requirements. In visual language, researchers often combine modality-specific models with self-supervised contrastive learning objectives to provide more robust representations. In the future, as GAI becomes increasingly important, more and more technologies shall be introduced, empowering this area with vitality.

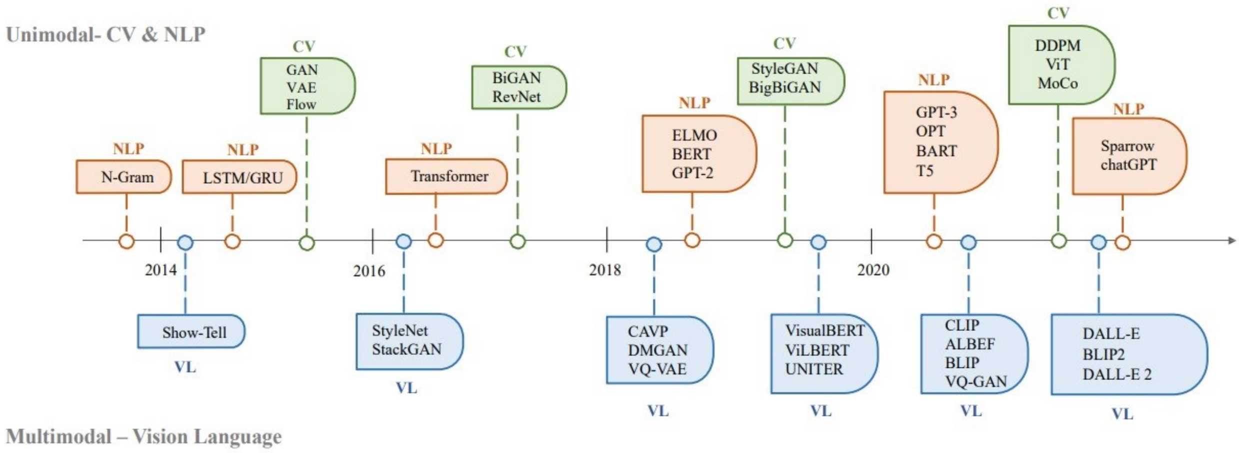


Figure 1.2-The history of Generative AI in CV, NLP and VL. [2]

Unimodal GAI models. These models are designed to accept a specific raw data modality as input, such as text or images, and then generate predictions in the same modality as the input. We will discuss some of the most promising approaches and techniques used in these models, including generative language models, e.g., GPT-3, BART, T5 and generative vision models, e.g., GAN, VAE and normalizing flow. Generative language models (GLMs) are a type of NLP models that are trained to generate readable human language based on patterns and structures in input data that they have been exposed to. These models can be used for a wide range of NLP tasks such as dialogue systems, translation and question answering. Recently, The use of pre-trained language models has emerged as the prevailing technique in the domain of NLP. Generally, current state-of-the-art pre-trained language models could be categorized as masked language models (encoders), autoregressive language models (decoders) and encoder-decoder language models. Decoder models are widely used for text generation, while encoder models are mainly applied to classification tasks. By combining the strengths of both structures, encoder-decoder models can leverage both context information and autoregressive properties to improve performance across a variety of tasks.

Multimodal generation serves as an essential part in nowadays GAI. The goal of multimodal generation is to learn a model that generates raw modalities by learning the multimodal connection and interaction from data. This connection and interaction between modalities can sometimes be very intricate, which makes the multimodal representation space hard to learn compared to the unimodal one. However, with the emergence of the powerful modality-specific foundation architectures mentioned in previous sections, a growing number of methods are proposed in response to this challenge. In this section, we introduce the state-of-the-art multimodal models in vision language generation, text audio generation, text graph generation and text code generation. Since most multimodal GAI models are always highly related to real-world applications, this section will mainly introduce from the perspective of downstream tasks.

1.2 - Transformer models, GPT model

Transformer [3] is the backbone architecture for many state-of-the-art GAI models, such as GPT. It was first proposed to solve the limitations of traditional models such as RNNs in handling variable-length sequences and context awareness. Transformer architecture is mainly based on a self-attention mechanism that allows the model to attend to different parts in an input sequence. Transformer consists of an encoder and a decoder. The encoder takes in input sequence and generates hidden representations, while the decoder takes in the hidden representation and generates output sequence. Each layer of the encoder and decoder consists of a multi-head attention and a feed-forward neural network. The multi-head attention is the core component of Transformer, which learns to assign different weights to tokens according to their relevance. This information routing method allows the model to be better at handling long term dependency, hence, improving the performance in a wide range of NLP tasks. Another advantage of transformer is that its architecture makes it highly parallelizable and allows data to trump inductive biases. This property makes transformer well-suited for large-scale pre-training, enabling transformer-based models to become adaptable to different downstream tasks.

Since the introduction of the Transformer architecture, it has become the dominant choice in NLP due to its parallelism and learning capabilities. Generally, these transformers based pre-trained language models can be commonly classified into two types based on their training tasks: autoregressive language modelling and masked language modelling. Given a sentence, which is composed of several tokens, the objective of masked language modelling, e.g., BERT, refers to predicting the probability of a masked token given context information. The most notable example of masked language modelling is BERT, which includes masked language modelling and next sentence prediction tasks. While autoregressive language modelling, e.g., GPT, is to model the probability of the next token given previous tokens, hence, left-to-right language modelling. Different from masked language models, autoregressive models are more suitable for generative tasks.

Despite being trained on large-scale data, the GAI may not always produce output that aligns with the user's intent, which includes considerations of usefulness and truthfulness. To better align AIGC output with human preferences, reinforcement learning from human feedback (RLHF) [4] has been applied to fine-tune models in various applications such as GPT models, Typically, the whole pipeline of RLHF includes the following three steps: pre-training, reward learning, and fine-tuning with reinforcement learning. First, a language model is pre-trained on large-scale datasets as an initial language model. Since the (prompt-answer) pair might not align with human purposes, in the second step we train a reward model to encode the diversified and complex human preference. Although RLHF has shown promising results by incorporating fluency, progress in this field is impeded by a lack of publicly available benchmarks and implementation resources, leading to a perception that RL is a challenging approach for NLP.

In recent years, there have been **significant hardware advancements** that have facilitated the training of large-scale models. In the past, training a large neural network using CPUs could take several days or even weeks. However, with the emergence of more powerful computing resources, this process has been accelerated by several orders of magnitude. For instance, the NVIDIA A100 GPU achieves seven times faster during BERT-large inference compared to the V100 and 11 times faster than the T42. Additionally, Google's Tensor Processing Units (TPUs), which are designed specifically for deep learning, offer even higher computing performance compared to the current generation of A100 GPUs3. This rapid progress in computing power has significantly increased the efficiency of AI model training and opened new possibilities for developing large and complex models.

Another significant improvement is **distributed training**. In traditional machine learning, training is typically performed on a single machine using a single processor. This approach can work well for small datasets and models, but it becomes impractical when dealing with large datasets and complex models. In distributed training, the training workload is split among multiple processors or machines, allowing the model to be trained much faster. Some companies have also released frameworks that simplify the process of distributed training on deep learning stacks. These frameworks provide tools and APIs that allow developers to easily distribute their training workloads across multiple processors or machines, without having to manage the underlying infrastructure.

Cloud computing has also played a vital role in training large-scale models. Previously, models are often trained locally. Now with the cloud computing services like AWS and Azure providing access to powerful computing resources, deep learning researchers and practitioners could spin up large clusters of GPUs or TPUs as needed for training large-scale models. Overall, these advancements have enabled the development of more complex and accurate models, unlocking new possibilities in various areas of AI research and applications.

1.3 - ChatGPT: competitors, historical notes and best features

In recent years, Chatbot (GAI tool designed to simulate conversation with human), has gained much attention beyond the computer science community, where the whole society begins to be interested in the various content generation products built by large tech companies, which can automate the creation of large amounts of content in a short time. Chatbots normally use large language models (LLM) like GPT to understand and respond to user queries and inputs in a conversational manner. They can be programmed to perform a wide range of tasks, for example, providing customer support and answering frequently asked questions. In the rapidly evolving landscape of GAI, a range of companies and research organizations have developed sophisticated chatbot. Here is a comparative analysis [10] of 6 of the notable chatbot competitors as of October 2023.

Chatbot Competitors

	OpenAI ChatGPT-4 [5]	OpenAI ChatGPT [5]	Google Bard [6]	Microsoft Bing [7]	Anthropic Claude 2 [8]	Opera Aria [9]
GAI Model	GPT-4	GPT 3.5	LaMDA	GPT-4	Claude 2.1	GPT
Model Size (Parameters)	1,76 trillion	175 billion	540 billion	135 billion	130 billion	130 billion
Training dataset (words)	n.d.	45Tb	n.d.	500 billion	n.d.	n.d.
Text Input	Yes	Yes	Yes	Yes	Yes	Yes
Voice Input	Yes	Yes	Yes	Yes	Yes	Yes
Images Input	Yes	Yes	Yes	Yes	Yes	Yes
Video Input	No	No	No	No	No	Yes
Text Output	Yes	Yes	Yes	Yes	Yes	Yes
Voice Output	Yes	Yes	Yes	Yes	Yes	Yes
Images Output	Yes	No	No	Yes	No	No
Video Output	No	No	No	No	No	No
Input-Output length(token)	8192	4096	1000	4096	200.000	4096
Best Ability	Data analysis, coding, visualization	Multi-purpose, conversation	Multi-purpose, conversation consulting external sources	Multi-purpose, conversation consulting web sources	Analyse and create long documents	Convert keywords in its responses into web links
Knowledge Cutoff Dates	04/2023	01/2022	None	None	Early 2023	None
Quantitative Accuracy [10]	1°	2°	4°	3°	5°	6°
Qualitative precision [10]	1°	2°	5°	4°	3°	6°
Price	20$ month	Free	Free	Free	20$ month	Free
Release Year	2023	2022	2023	2023	2023	2023

Table 1.2- Overview of different characteristics of major Chatbot competitors.

The scale of an LLM depends on the size of ingested datasets, the amount of training compute, and the number of parameters it can support. Parameters are numerical values that determine how a neural network processes and generates natural language. The more parameters a model has, the more data it can learn from and the more complex tasks it can perform.

In comparison result in table 1.2, **ChatGPT-4 significantly outperforms its competitors in terms of accuracy and precision. This result is consistent with other comparisons available online. For this reason, combined with its widespread use, this thesis is focused on it.** ChatGPT-4 belongs to the ChatGPT model family developed by OpenAI for building chatbot, which can efficiently understand and respond to human language inputs in a meaningful way. ChatGPT has undergone several updates and improvements since its debut, becoming a powerful tool for natural language processing tasks such as language translation, question answering, and even creative writing. Now, we'll look at the evolution of ChatGPT, from its early days to the present, and see how it has grown to become one of the most advanced language models on the market.

In the early days of ChatGPT, the model was first introduced by OpenAI in 2018. At that time, the model was based on the transformer architecture and was pre-trained on a massive amount of internet text data. It was primarily used for language generation tasks such as text completion and summarization. However, the model's capabilities were limited compared to today's version. For example, it struggled with understanding context and had a hard time with more complex language tasks. Additionally, the model's training data was relatively small compared to current versions, which means that it had a harder time understanding and responding to certain types of input. Despite these limitations, the introduction of ChatGPT marked a major step forward in the field of natural language processing and set the stage for further advancements in the years to come.

The first major update to ChatGPT, came in 2019, when OpenAI released a new version of the model that was significantly more powerful than the original. This update was focused on improving the model's language understanding capabilities. One of the key changes was the expansion of the training data used to train the model. By training the model on a much larger dataset, it was able to learn a wider range of language patterns and improve its ability to understand context. Additionally, the update introduced fine-tuning capabilities, which allowed users to fine-tune the model on their own specific task or dataset. This made ChatGPT more versatile and useful for a wider range of applications. This update also introduced a new version of the model called GPT-2, which was even more powerful and accurate than the original ChatGPT.

This update was a significant step forward for ChatGPT, and it greatly expanded the model's capabilities and usefulness. With improved language understanding and fine-tuning capabilities, the updated ChatGPT was able to perform a wider range of tasks, making it a more valuable tool for researchers and developers in the field of natural language processing.

The introduction of fine-tuning capabilities allowed users to adapt the model to specific tasks or datasets. Prior to the fine-tuning capabilities, the model was pre-trained on a large dataset of internet text data, and it could only perform general language tasks such as text completion and summarization. However, with fine-tuning, users could now adapt the model to perform more specific and specialized tasks, such as language translation, question answering, and even creative writing. Fine-tuning works by using a smaller dataset that is relevant to the specific task at hand and "fine-tuning" the model's parameters on this dataset. This allows the model to learn task-specific patterns and improve its performance on that task. This process can be done quickly and easily, and it doesn't require retraining the model from scratch. The introduction of fine-tuning capabilities was a major step forward for ChatGPT, as it greatly expanded the model's usefulness and versatility. It allowed users to customize the model to their specific needs, making it a valuable tool for a wide range of applications. Additionally, it also enabled researchers and developers to conduct more targeted experiments and achieve better performance on specific tasks.

The expansion of training data refers to the process of improving the amount of text data used to train the ChatGPT model. The more data the model is trained on, the more it can learn about different language patterns and styles. This helps the model to better understand and respond to a wide range of inputs, and it also improves its ability to understand context and carry out more complex language tasks. In the early days of ChatGPT, the model was trained on a relatively small dataset of internet text data. However, with the release of later versions, the model was trained on much larger datasets, which greatly improved its performance. For example, GPT-2, which was an updated version of ChatGPT, was trained on a dataset that was several orders of magnitude larger than the original ChatGPT. This helped GPT-2 to achieve better performance on a wide range of language tasks, and it also made the model more versatile and useful for a wider range of applications. Expanding the training data is a critical step in the development of any language model, as more data leads to more accurate and versatile models. With the expansion of training data, ChatGPT was able to improve its language understanding, and it also helped the model to better understand and respond to different types of inputs.

Improvements in language understanding, refer to the advancements in the ability of the ChatGPT model to understand and interpret the meaning of text input. The more a model can understand about the input, the better it can respond or generate text. Language understanding is a critical aspect of natural language processing and it encompasses different aspects of language understanding such as understanding context, sentiment, entities, and intent. In the early days of ChatGPT, the model had a hard time understanding the context of the input, and it struggled with more complex language tasks. However, with the release of later versions, the model's language understanding capabilities were greatly improved. For example, the introduction of fine-tuning capabilities and the expansion of training data helped the model to better understand and respond to different types of inputs. Additionally, the model was also trained to understand entities and sentiment, which improved its ability to understand the meaning and context of text input. The improvements in language understanding allowed ChatGPT to perform a wider range of tasks, making it a more valuable tool for researchers and developers in the field of natural language processing. With better language understanding, ChatGPT was able to generate more accurate and natural responses, and it also made the model more versatile and useful for a wide range of applications.

Integration with other applications, refers to the ability of ChatGPT to be incorporated into other systems and software to perform various natural language processing tasks. As a powerful language model, ChatGPT can be utilized in a wide range of applications such as language translation, question answering, text summarization, and even creative writing. One way of integrating ChatGPT with other applications is by using its language generation capabilities to generate text. For example, ChatGPT can be used to generate responses in a chatbot, to automatically summarize news articles, or to generate creative writing such as stories or poetry. Another way to integrate ChatGPT is through fine-tuning the model on specific tasks or datasets that align with the needs of the application. This allows the model to perform specific tasks such as language translation, sentiment analysis, or intent recognition. The integration of ChatGPT with other applications greatly expands its capabilities and usefulness. It allows the model to be used in a wide range of industries such as customer service, content creation, and language translation. Additionally, it also makes the model more versatile and more accessible to a wider range of users. The integration of ChatGPT with other applications has also led to the development of new applications and services that leverage the model's capabilities.

The impact of ChatGPT on natural language processing (NLP) research has been significant. As one of the most advanced and powerful language models, ChatGPT has helped to push the boundaries of what is possible with NLP and has opened up new avenues of research.

One of the main ways that ChatGPT has impacted NLP research is by providing a powerful tool for language generation. The model's ability to generate natural-sounding text has led to new research in areas such as text completion, summarization, and creative writing. Additionally, its ability to fine-tune on specific tasks and understand context has led to research on other NLP tasks such as language translation, sentiment analysis, and intent recognition. ChatGPT has also had a significant impact on the field by being a powerful benchmark model. Its performance on various NLP tasks set a high bar for other models to compete with. This has led to a surge in research on developing models that can perform as well or better than ChatGPT on various NLP tasks. Furthermore, the release of GPT-3, a later version of ChatGPT, has sparked a lot of interest and research in the field of language understanding and language generation. It has also sparked research in areas like zero-shot learning and transfer learning.

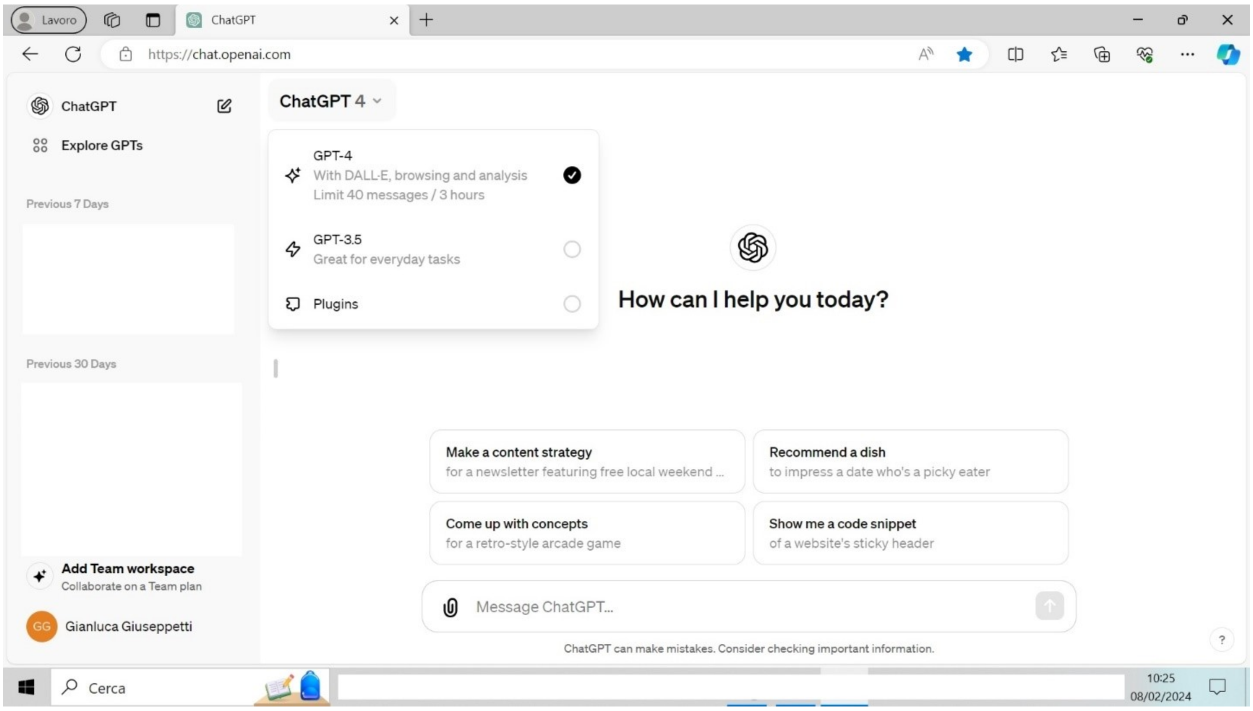
Current capabilities and future potential of ChatGPT. ChatGPT is currently one of the most advanced and powerful language models available. It can perform a wide range of natural language processing tasks such as language generation, text completion, summarization, translation, text-to-speech, and even creative writing. Its fine-tuning capabilities also allow it to be adapted to specific tasks or datasets, making it a versatile tool for a wide range of applications. In terms of future potential, ChatGPT has the potential to be used in even more industries and applications. With further improvements in language understanding, it could be used in areas such as sentiment analysis, opinion mining, and emotion recognition. Additionally, with advancements in machine learning and deep learning, ChatGPT could be used in fields such as computer vision, speech recognition, and robotic process automation.

In the Table 1.3, I show a chronological summary [5] of key events and developments of ChatGPT and OpenAI's related services from late 2022 to now.

Period	Update
Nov 30, 2022	ChatGPT was launched by San Francisco–based OpenAI (the creator of the initial GPT series of large language models; DALL·E 2, a diffusion model used to generate images; and Whisper, a speech transcription model). The service was initially free to the public and the company had plans to monetize the service later.
Dec 4, 2022	ChatGPT reached over one million users.
Jan 2023	ChatGPT reached over 100 million users, becoming the fastest-growing consumer application.
Feb 2023	Launch of ChatGPT Plus at $20 a month.
Mar 14, 2023	Release of ChatGPT-4; available via API and for ChatGPT Plus users. But premium users were limited to a cap of 100 messages every four hours, with the limit tightening to 25 messages every three hours in response to increased demand. This includes both plugins made by OpenAI, such as browse with Bing and Advanced Data Analysis , and external plugins from developers such as , Expedia, OpenTable, Zapier, Shopify, Slack, Wikipedia and Wolfram.
Mar 2023	ChatGPT Plus users got access to third-party plugins and browsing mode. In addition to its consumer-friendly "ChatGPT Plus" package, OpenAI made its ChatGPT and Whisper model API available in March 2023, providing developers with an application programming interface for AI-enabled language and speech-to-text features. ChatGPT's new API uses the same GPT-3.5-turbo AI model as the chatbot. This allows developers to add either an unmodified or modified version of ChatGPT to their applications. The ChatGPT API costs $0.002 per 1000 tokens (about 750 words), making it ten times cheaper than the GPT-3.5 models.
Apr 2023	ChatGPT blocked by Italy, China, Iran, North Korea, and Russia.
May 2023	Launch of ChatGPT iOS app with chat history syncing and voice input.
Jun 2023	Launch of ChatGPT Android app in Bangladesh, Brazil, India, and the U.S., later worldwide.
Jul 2023	OpenAI's proprietary Code Interpreter plugin made available to ChatGPT Plus subscribers. The Interpreter provides a wide range of capabilities, including data analysis and interpretation, instant data formatting, personal data scientist services, creative solutions, musical taste analysis, video editing, and file upload/download with image extraction.
Sep 2023	Announcement of DALL-E 3 integration into ChatGPT Plus and Enterprise in October. The integration uses ChatGPT to write prompts for DALL-E guided by conversation with users.
Nov 6, 2023	Introduction of GPTs for Plus and Enterprise users to create tailored versions of ChatGPT.
Nov 21, 2023	ChatGPT with voice is available to all users.
Late 2023	Introduction of AI classifier for indicating AI-written text. OpenAI cautions that the tool will "likely yield a lot of false positives and negatives, sometimes with great confidence.”
Jan 10, 2024	Introducing GPT Store and ChatGPT Team plan.
Feb 13, 2024	Memory and new controls for ChatGPT.

Table 1.3- Overview of OpenAI ChatGPT last update

The last web version of ChatGPT-4 (Figure 1.3) is a state-of-the-art language generation model that can perform a variety of tasks, including writing essays, summarizing text, answering questions, translating languages, and even basic reasoning. Among countless examples, it can write and debug computer programs, compose music, teleplays, fairy tales, and student essays, answer test questions (sometimes, depending on the test, at a level above the average human test-taker), generate business ideas, write poetry and song lyrics, translate and summarize text emulate a Linux system, simulate entire chat rooms, play games like tic-tac-toe. OpenAI also makes an app version available to install but the most recent features are first made available in the web version.



How can I help you today?

ChatGPT can make mistakes. Consider checking important information.

Figure 1.3- Last web version ChatGPT-4 screenshot.

Limitations. Sometimes writes plausible sounding but incorrect or nonsensical answers". This behaviour is common for large language models and is called "hallucination".

ChatGPT-4 has limited knowledge of events that occurred after April 2023, remembers a limited number of previous prompts in the same conversation, no interaction between current conversation and a previous, cap of 40 messages each 3 hours, occasionally archive chats are not long available to continue chatting and generating incorrect or nonsensical text and sometimes platform income in crash errors.

Current common troubleshooting in Appendices, section B.

CHAPTER 2

ChatGPT-4: a teaching tool for Computer Science education

2.1 - State-of-the-art teaching features

The integration of technological tools in education, especially in the domain of Computer Science (CS), has indeed become significantly more popular in the 21st century. This shift towards technology-enhanced learning environments in schools is driven by the recognition that such tools can offer numerous benefits over conventional teaching techniques. Innovation is fundamental to CS education because the use of technological tools improves the quality of education.

The computing education community has a rich history of pedagogical innovation [12] designed to support teachers in facilitating student learning. The appropriate use of ChatGPT-4 holds potential to revolutionize the CS education sector by reducing workloads and freeing up teachers to concentrate on delivering top-notch teaching. However, it's crucial to acknowledge that ChatGPT-4 generated content could be inaccurate, inappropriate, biased, or even out of date. The onus of using ChatGPT-4 result effectively rests on understanding these limitations and applying professional judgment. Teachers' workload remains a pressing issue, and there's a strong commitment to help educators spend less time on non-pupil facing activities. Collaborations with the education sector and experts aim to identify and leverage opportunities to enhance education and reduce workload using ChatGPT-4. Key to this is the understanding that having access to ChatGPT-4 doesn't replace the need for deep knowledge. Effective use of ChatGPT-4 requires clear writing skills and domain understanding for prompt creation, and a schema for sense-checking the results. ChatGPT-4 can simplify certain written tasks, but it can't replace the judgment and in-depth subject knowledge of a human expert.

In tables 2.1, I summarize the state-of-the-art features of ChatGPT-4 that hold the potential to significantly impact traditional teaching methods. These features are categorized in 4 macro-features: customized content creation, accessibility, professional development for educators and assessment creation and grading. For each feature, the table provides specify the feature name, a description, the educational benefits and references to the relative example number from the case study presented in the next paragraph.

Customized Didactical Material Creation [13][15][20][22]

N	Feature Name	Description	Educational Benefits	Example
1	Textbooks	Personalized textbooks focusing on programming, data structures, and advancements in CS.	Enhanced understanding, paced learning	1
2	Quizzes	Quizzes on coding logic, software design, and algorithms.	Effective assessment, understanding of learning gaps	2
3	Flashcards	Interactive flashcards for memorizing CS terms, code syntax, and algorithms.	Aids in memorization and understanding of key concepts	3
4	Interactive Lessons	Virtual coding labs, simulations, and real-time feedback systems for learning CS concepts.	Hands-on learning, real-time problem-solving support	NO

Accessibility [13][15][20][22]

N	Feature Name	Description	Educational Benefits	Example
5	Speech Recognition for Input	Enables voice input for text and code entry.	Assists students who struggle with traditional keyboards, especially in programming and coding.	2;3
6	Descriptive Text for Visual Content	Automatically generates alt text for images and descriptions for videos.	Aids students with visual impairments by describing visual content in detail.	4
7	Predictive Text and Learning Aids	Offers predictive text options and real-time feedback.	Supports students with learning disabilities in constructing sentences and understanding content.	5
8	Text-to-Speech (TTS) Conversion	Converts written text into spoken words.	Helps students with visual impairments or reading difficulties by allowing them to listen to content.	NO
9	Sign Language Interpretation	Creates virtual sign language interpreters.	Translates spoken or written language into sign language in real-time.	NO
10	Customizable Learning Interfaces	Adapts the user interface of educational software to various needs.	Enhances accessibility for students with disabilities by modifying interface elements.	NO

Professional Development for Educators [17][20][21][25]

N	Feature Name	Description	Educational Benefits	Example
11	Personalized Learning Experiences for Educators	AI systems suggest customized professional development courses based on teaching style and expertise.	Tailored professional growth, enhanced teaching skills	6
12	Facilitating Peer Collaboration and Networking	Connects educators globally for knowledge sharing and collaboration.	Expanded professional network, collaborative learning	7
13	Literature Review Assistance	Assists in literature review by suggesting relevant resources and organizing information.	Helps in conducting thorough and relevant literature reviews.	NO

Assessment Creation and Grading [17]18][20][21]				
N	**Feature Name**	**Description**	**Educational Benefits**	**Example**
14	Personalized Assessments	Creates assessments based on a student's individual progress, strengths, and weaknesses.	Effective in measuring individual student progress.	NO
15	Plagiarism Detection	Checks assignments for plagiarism to maintain academic integrity.	Helps to ensures originality of student work.	NO
16	Reducing Bias in Grading	Grades based on set criteria and data, minimizing unconscious bias.	More objective assessment of student work.	NO

Tables 2.1- Overview of state-of-the-art teaching features of ChatGPT-4

A precious source as example of application of state-of-the-art features has been Prof. Marc Alier [11] reported by my supervisor. His adept utilization of advanced tools like ChatGPT-4 in his teaching methodologies exemplifies the merger of technology and education. In the videos series "ChatGPT-4 "Oportunidad y reto para la docencia" have illuminated pathways to make world of coding and algorithms in a manner that resonates with each student's unique learning style. It's not just about imparting knowledge but creating an environment where each student is inspired to explore, discover, and innovate. "El shock del futuro". Prof. Alier, speaking about GAI and ChatGPT-4, defines the concept of future shock is a warning of how we must adapt and maintain and learn to handle changes in society and technology so that we can enjoy the benefits without feeling overwhelmed or fearful. During a person's life the world changes completely and what happens is that as change accelerates these radical changes, they happen in society many times as we live. Prof. Alier, with his video course, guided me to how to deal the changes in teaching produced using ChatGPT-4.

With reference to my experience, in the last 2 years, as substitute teacher in Mathematics and Computer Science at the Istituto Tecnico Industriale Fermi-Sacconi-Ceci in Ascoli Piceno (IT), I will describe, in the following paragraph, a personal case study on how I utilized some of the state-of-the-art teaching features of ChatGPT-4 (Table 2.1, Example column). Due to the occasional need for multiple interactions with ChatGPT-4 and common troubleshooting issues (Appendices B), especially in the first period of this version release (from February 2023 to last 2023), I have included only the most relevant chat excerpts for each example. These examples were developed following the best practices outlined in OpenAI's prompt engineering guidelines (Appendices C). Although ChatGPT-4 can understand prompts in Italian, I choose to input prompts in English and request responses in Italian, as this approach yielded better performance results.

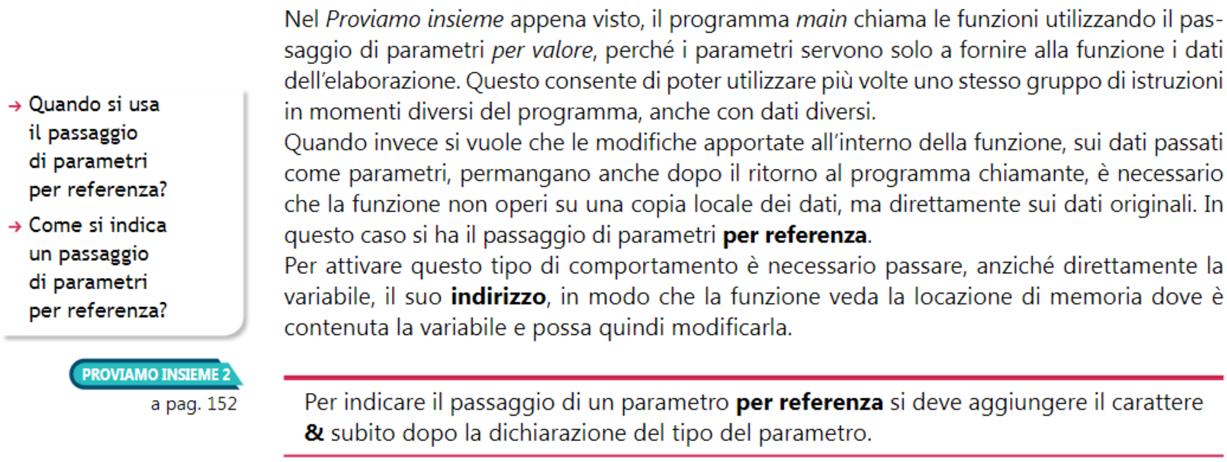
2.2 - Case study: personal experience as substitute teacher in secondary school

Inclusivity in CS education at the secondary level is not an option but a necessity. As we march into a future where technology is woven into the fabric of society, ensuring that every young mind has an equal opportunity to participate in this journey is a collective responsibility. Every student brings a unique perspective and these diverse viewpoints can unlock innovations and solutions for tomorrow's challenges.

This diversity isn't a challenge but an opportunity.

In my journey as a CS teacher, I've recognized that each student brings a unique set of attributes, skills, and challenges to the classroom. As a teacher, the responsibility to prepare didactic material that is inclusive involves ensuring that all students, regardless of their abilities, backgrounds, or learning styles, have equal access to education and are provided with the necessary tools and support to reach their fullest potential. I'm committed to taking concrete steps to ensure my teaching materials are inclusive and accessible to all. Feedback becomes my guide, my continual evolution as an educator. It's a collaborative way where each observation, critique, and appreciation from my students and peers is a stepping stone, propelling me to refine my approaches, ensuring that learning is not just inclusive but a diverse variety of talents, perspectives, and innovations that each student brings to the classroom.
I tried to approach this essential duty with the help of ChatGPT-4, especially for (special educational needs) BES and specific learning disorder (DSA), in C++ programming and Computer Science basic concepts. Below, I present 7 examples showcasing the use of ChatGPT-4's teaching features.

In the Example 1, ChatGPT-4 helped me (Figure 2.4) to produce an integration to textbook Atlas Pro.tech Vol.A, official education book, about parameter passed by reference in C++ functions. In the textbook, it's only used the notation that places the '&' symbol of at the end of the declaration type (Figure 2.2) and there is no mention of the alternative notation that places the '&' symbol at the beginning of the variable name. My students have encountered numerous examples of this alternative notation in various online sources and asked me which is the correct notation. Since both notations work correctly within the development environment, I would like to request an explanation from ChatGPT-4 regarding the motivations behind these two notations.



→ Quando si usa il passaggio di parametri per referenza?

→ Come si indica un passaggio di parametri per referenza?

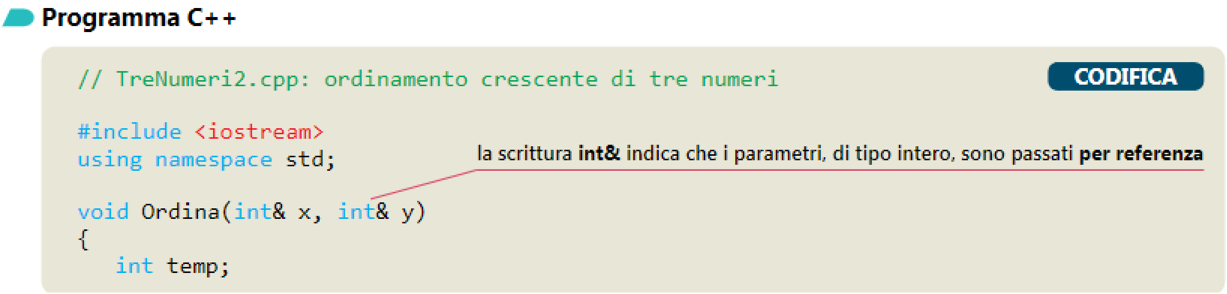
PROVIAMO INSIEME 2 a pag. 152

Nel *Proviamo insieme* appena visto, il programma *main* chiama le funzioni utilizzando il passaggio di parametri *per valore*, perché i parametri servono solo a fornire alla funzione i dati dell'elaborazione. Questo consente di poter utilizzare più volte uno stesso gruppo di istruzioni in momenti diversi del programma, anche con dati diversi.
Quando invece si vuole che le modifiche apportate all'interno della funzione, sui dati passati come parametri, permangano anche dopo il ritorno al programma chiamante, è necessario che la funzione non operi su una copia locale dei dati, ma direttamente sui dati originali. In questo caso si ha il passaggio di parametri **per referenza**.
Per attivare questo tipo di comportamento è necessario passare, anziché direttamente la variabile, il suo **indirizzo**, in modo che la funzione veda la locazione di memoria dove è contenuta la variabile e possa quindi modificarla.

Per indicare il passaggio di un parametro **per referenza** si deve aggiungere il carattere **&** subito dopo la dichiarazione del tipo del parametro.

Figure 2.2- Extract of page151 of textbook Atlas Pro.tech Vol. A

In the example at page. 152 (Figure 2.3), & char is attached at the end of the declaration type:



Programma C++

CODIFICA

```
// TreNumeri2.cpp: ordinamento crescente di tre numeri

#include <iostream>
using namespace std;

void Ordina(int& x, int& y)
{
   int temp;
```

la scrittura **int&** indica che i parametri, di tipo intero, sono passati **per referenza**

Figure 2.3- Extract of page152 of textbook Atlas Pro.tech Vol. A

GG

You

passaggio di parametri per referenza in c++

ChatGPT

ChatGPT

Il passaggio di parametri per referenza in C++ consente di passare agli argomenti di una funzione non il valore stesso, ma un riferimento a esso. Questo significa che la funzione può modificare il valore della variabile passata come argomento nella funzione chiamante. Il passaggio per referenza è utile per modificare uno o più argomenti o per ottimizzare le prestazioni evitando la copia di dati di grandi dimensioni.

In C++, il passaggio per referenza viene realizzato in due modi principali:

1. **Utilizzando i puntatori**
2. **Utilizzando le referenze**

Utilizzo dei Puntatori: per passare un parametro per referenza utilizzando i puntatori, è necessario passare l'indirizzo della variabile utilizzando l'operatore `&` e il parametro della funzione deve essere un puntatore che può conservare questo indirizzo. Ecco un esempio:

```
#include <iostream>
using namespace std;

void incrementa(int *valore) {
    (*valore)++;
}

int main() {
    int a = 5;
    incrementa(&a); // Passa l'indirizzo di a
    cout << a; // Stampa 6
    return 0;
```

} In questo esempio, `incrementa` riceve un puntatore a un intero. Modificando il valore puntato da `valore` (cioè `*valore++`), si modifica il valore della variabile `a` nel `main`.

Utilizzo delle Referenze: il passaggio per referenza può essere realizzato più direttamente in C++ utilizzando le referenze. Una referenza è essenzialmente un alias per un'altra variabile. Ecco come si può fare:

```
#include <iostream>
using namespace std;

void incrementa(int &valore) {
    valore++;
}

int main() {
    int a = 5;
    incrementa(a); // Passa a per referenza
    cout << a; // Stampa 6
    return 0;
```

} In questo caso, la funzione `incrementa` riceve una referenza a un intero. Qualsiasi modifica a `valore` all'interno della funzione `incrementa` influisce direttamente sulla variabile `a` nel `main`, senza necessità di dereferenziazione esplicita.

Considerazioni

- **Efficienza**: Il passaggio per referenza evita la copia di oggetti di grandi dimensioni, migliorando l'efficienza del programma.
- **Modifiche**: Se non si desidera che la funzione modifichi il parametro, è possibile passare la referenza o il puntatore come `const` per garantire la sicurezza.
- **Utilizzo**: Le referenze sono generalmente preferite per la loro sintassi più semplice e per il fatto che non possono essere `nullptr`, rendendo il codice più sicuro e più facile da comprendere rispetto all'uso dei puntatori.

Scegliere tra puntatori e referenze dipende dalle esigenze specifiche del programma e dalle preferenze dello sviluppatore, ma l'uso delle referenze è spesso considerato uno stile più moderno e sicuro in C++.

GG

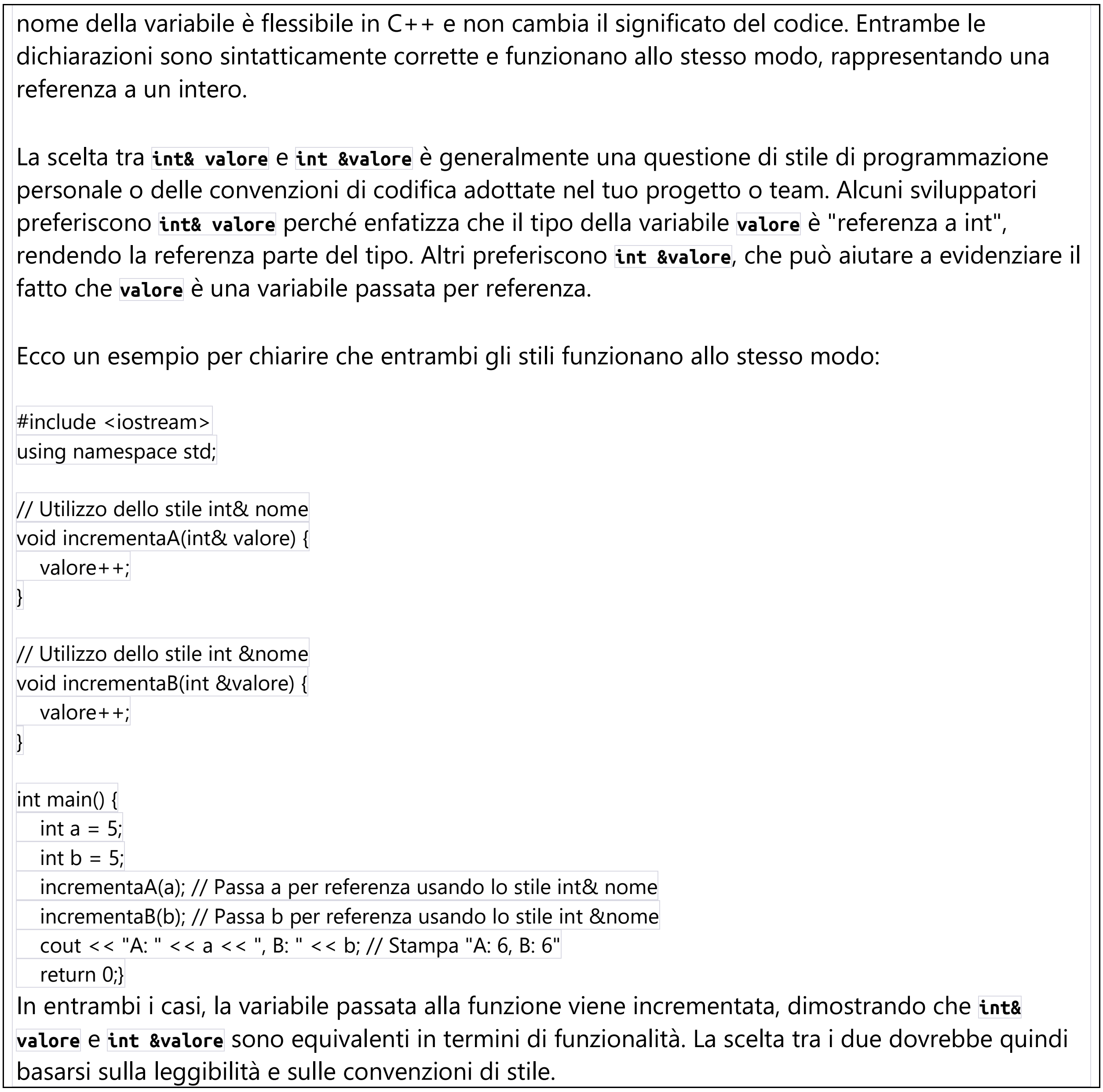
You

invece di int &valore posso usare int& valore

ChatGPT

ChatGPT

Sì, puoi usare `int& valore` al posto di `int &valore` quando definisci una funzione che accetta un parametro per referenza in C++. La posizione dell'`&` (operatore di referenza) rispetto al tipo e al



nome della variabile è flessibile in C++ e non cambia il significato del codice. Entrambe le dichiarazioni sono sintatticamente corrette e funzionano allo stesso modo, rappresentando una referenza a un intero.

La scelta tra `int& valore` e `int &valore` è generalmente una questione di stile di programmazione personale o delle convenzioni di codifica adottate nel tuo progetto o team. Alcuni sviluppatori preferiscono `int& valore` perché enfatizza che il tipo della variabile `valore` è "referenza a int", rendendo la referenza parte del tipo. Altri preferiscono `int &valore`, che può aiutare a evidenziare il fatto che `valore` è una variabile passata per referenza.

Ecco un esempio per chiarire che entrambi gli stili funzionano allo stesso modo:

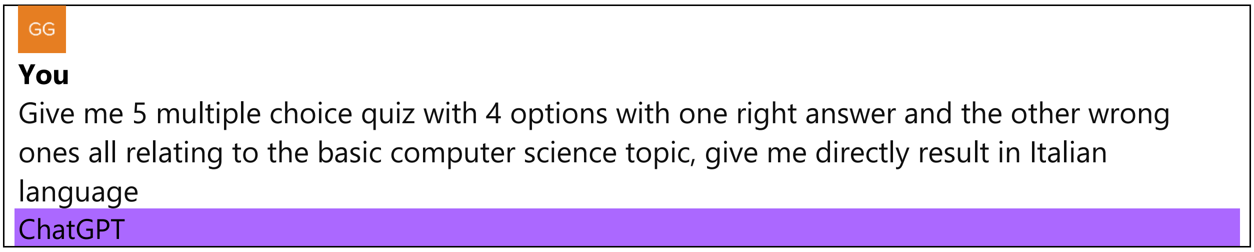
```
#include <iostream>
using namespace std;

// Utilizzo dello stile int& nome
void incrementaA(int& valore) {
    valore++;
}

// Utilizzo dello stile int &nome
void incrementaB(int &valore) {
    valore++;
}

int main() {
    int a = 5;
    int b = 5;
    incrementaA(a); // Passa a per referenza usando lo stile int& nome
    incrementaB(b); // Passa b per referenza usando lo stile int &nome
    cout << "A: " << a << ", B: " << b; // Stampa "A: 6, B: 6"
    return 0;}
```

In entrambi i casi, la variabile passata alla funzione viene incrementata, dimostrando che `int& valore` e `int &valore` sono equivalenti in termini di funzionalità. La scelta tra i due dovrebbe quindi basarsi sulla leggibilità e sulle convenzioni di stile.

Figure 2.4- ChatGPT-4 chat prompt and result of Example 1.

After a revision of explanation on official C++ manual available on web, I used this explanation to integrate official textbook.

In the Example 2, ChatGPT-4 helped me (Figure 2.5) to produce personalized quizzes aligned with each student's learning progression, manage complexity and format of the questions make them suitable for BES students.

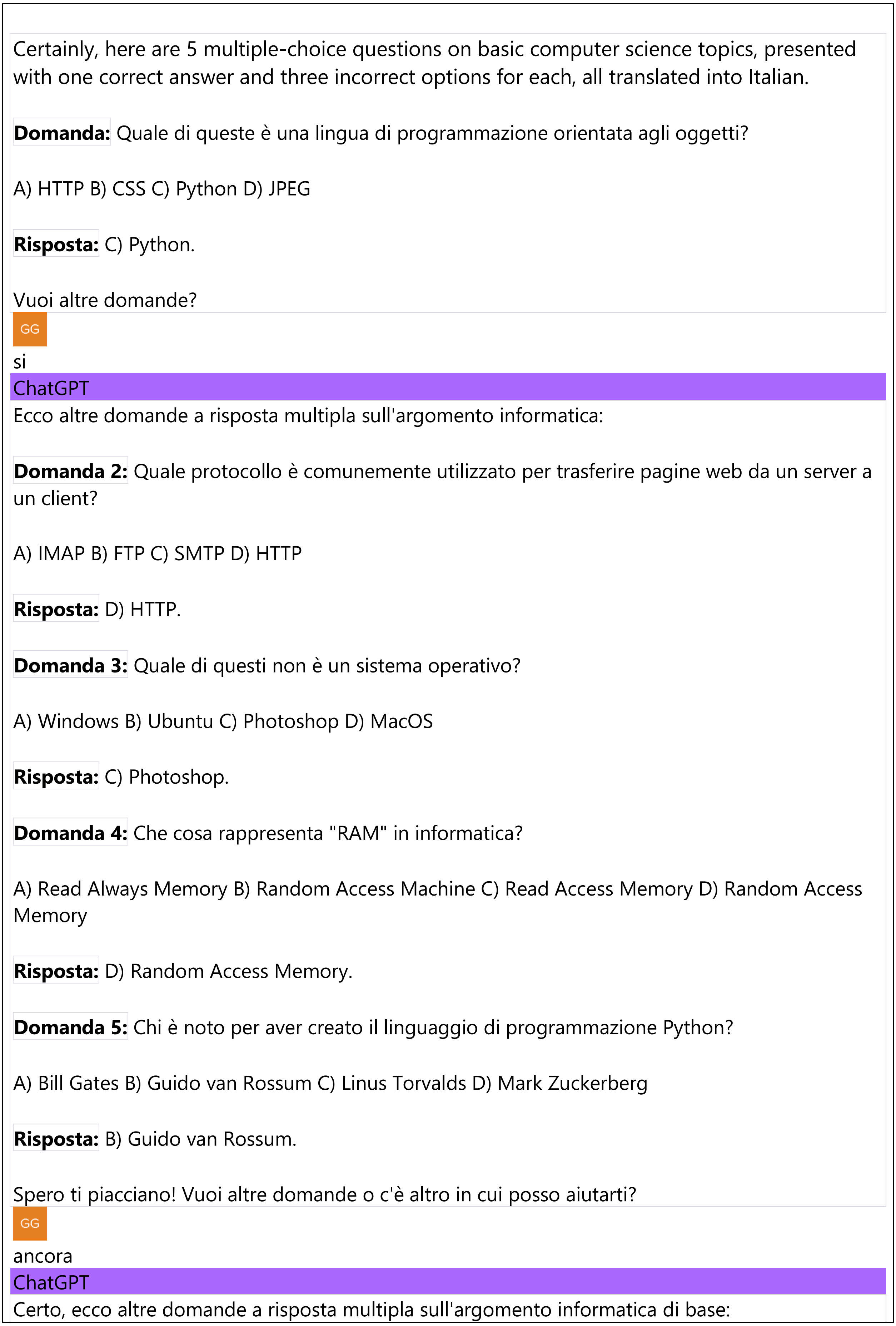


GG

You

Give me 5 multiple choice quiz with 4 options with one right answer and the other wrong ones all relating to the basic computer science topic, give me directly result in Italian language

ChatGPT



Certainly, here are 5 multiple-choice questions on basic computer science topics, presented with one correct answer and three incorrect options for each, all translated into Italian.

Domanda: Quale di queste è una lingua di programmazione orientata agli oggetti?

A) HTTP B) CSS C) Python D) JPEG

Risposta: C) Python.

Vuoi altre domande?

GG

si

ChatGPT

Ecco altre domande a risposta multipla sull'argomento informatica:

Domanda 2: Quale protocollo è comunemente utilizzato per trasferire pagine web da un server a un client?

A) IMAP B) FTP C) SMTP D) HTTP

Risposta: D) HTTP.

Domanda 3: Quale di questi non è un sistema operativo?

A) Windows B) Ubuntu C) Photoshop D) MacOS

Risposta: C) Photoshop.

Domanda 4: Che cosa rappresenta "RAM" in informatica?

A) Read Always Memory B) Random Access Machine C) Read Access Memory D) Random Access Memory

Risposta: D) Random Access Memory.

Domanda 5: Chi è noto per aver creato il linguaggio di programmazione Python?

A) Bill Gates B) Guido van Rossum C) Linus Torvalds D) Mark Zuckerberg

Risposta: B) Guido van Rossum.

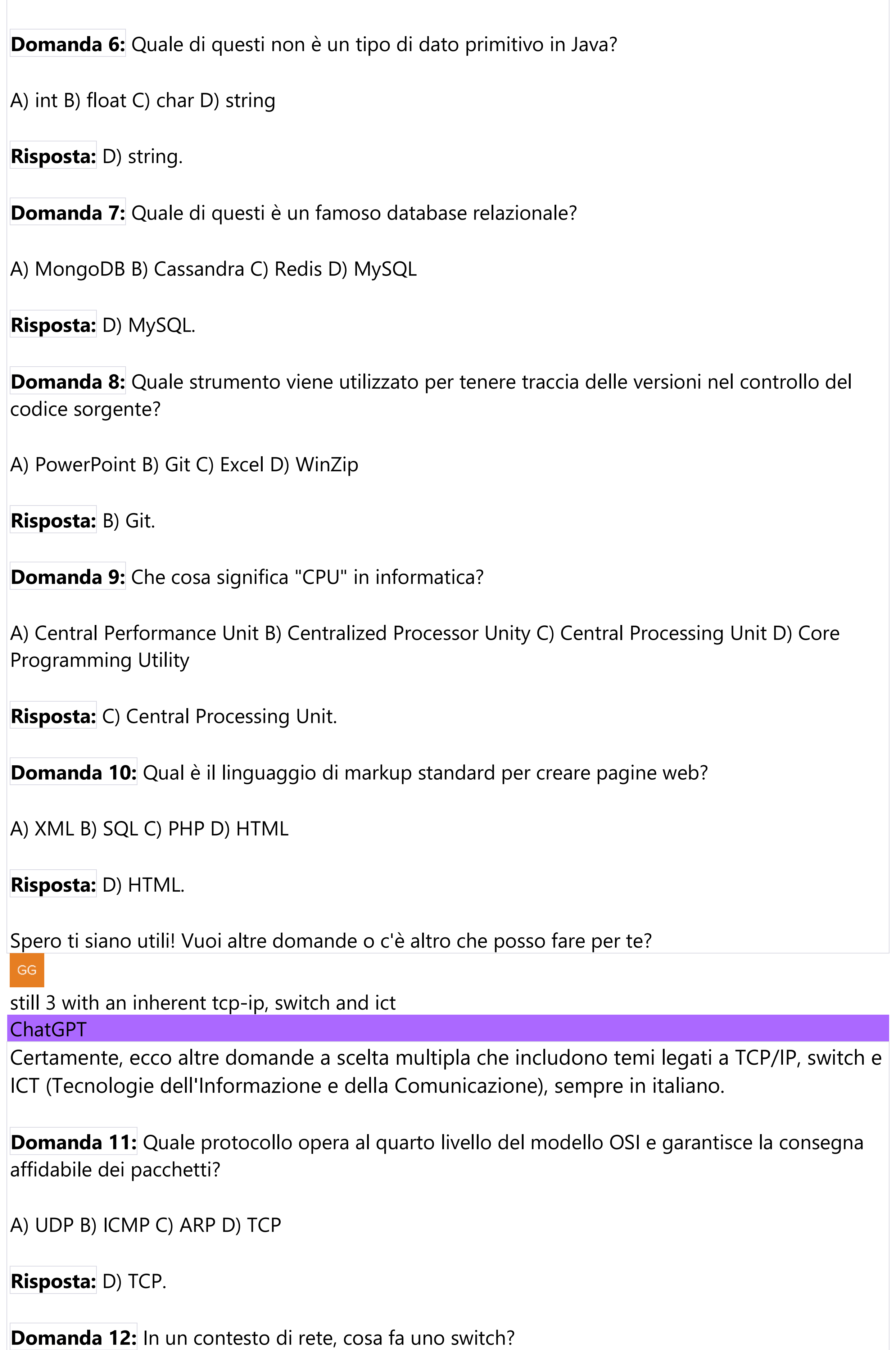
Spero ti piacciano! Vuoi altre domande o c'è altro in cui posso aiutarti?

GG

ancora

ChatGPT

Certo, ecco altre domande a risposta multipla sull'argomento informatica di base:



Domanda 6: Quale di questi non è un tipo di dato primitivo in Java?

A) int B) float C) char D) string

Risposta: D) string.

Domanda 7: Quale di questi è un famoso database relazionale?

A) MongoDB B) Cassandra C) Redis D) MySQL

Risposta: D) MySQL.

Domanda 8: Quale strumento viene utilizzato per tenere traccia delle versioni nel controllo del codice sorgente?

A) PowerPoint B) Git C) Excel D) WinZip

Risposta: B) Git.

Domanda 9: Che cosa significa "CPU" in informatica?

A) Central Performance Unit B) Centralized Processor Unity C) Central Processing Unit D) Core Programming Utility

Risposta: C) Central Processing Unit.

Domanda 10: Qual è il linguaggio di markup standard per creare pagine web?

A) XML B) SQL C) PHP D) HTML

Risposta: D) HTML.

Spero ti siano utili! Vuoi altre domande o c'è altro che posso fare per te?

still 3 with an inherent tcp-ip, switch and ict

ChatGPT

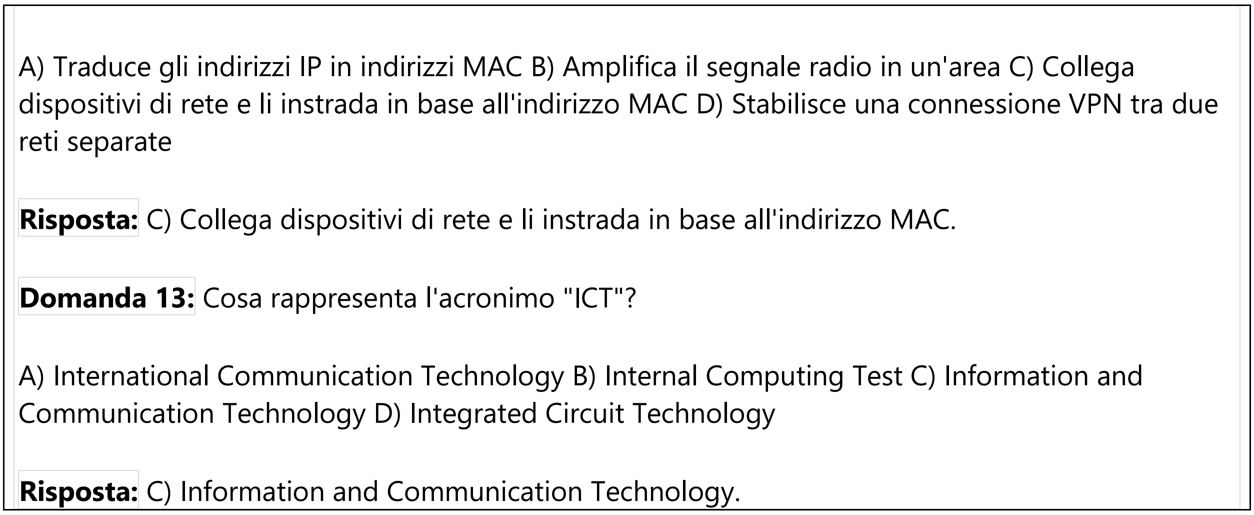
Certamente, ecco altre domande a scelta multipla che includono temi legati a TCP/IP, switch e ICT (Tecnologie dell'Informazione e della Comunicazione), sempre in italiano.

Domanda 11: Quale protocollo opera al quarto livello del modello OSI e garantisce la consegna affidabile dei pacchetti?

A) UDP B) ICMP C) ARP D) TCP

Risposta: D) TCP.

Domanda 12: In un contesto di rete, cosa fa uno switch?



A) Traduce gli indirizzi IP in indirizzi MAC B) Amplifica il segnale radio in un'area C) Collega dispositivi di rete e li instrada in base all'indirizzo MAC D) Stabilisce una connessione VPN tra due reti separate

Risposta: C) Collega dispositivi di rete e li instrada in base all'indirizzo MAC.

Domanda 13: Cosa rappresenta l'acronimo "ICT"?

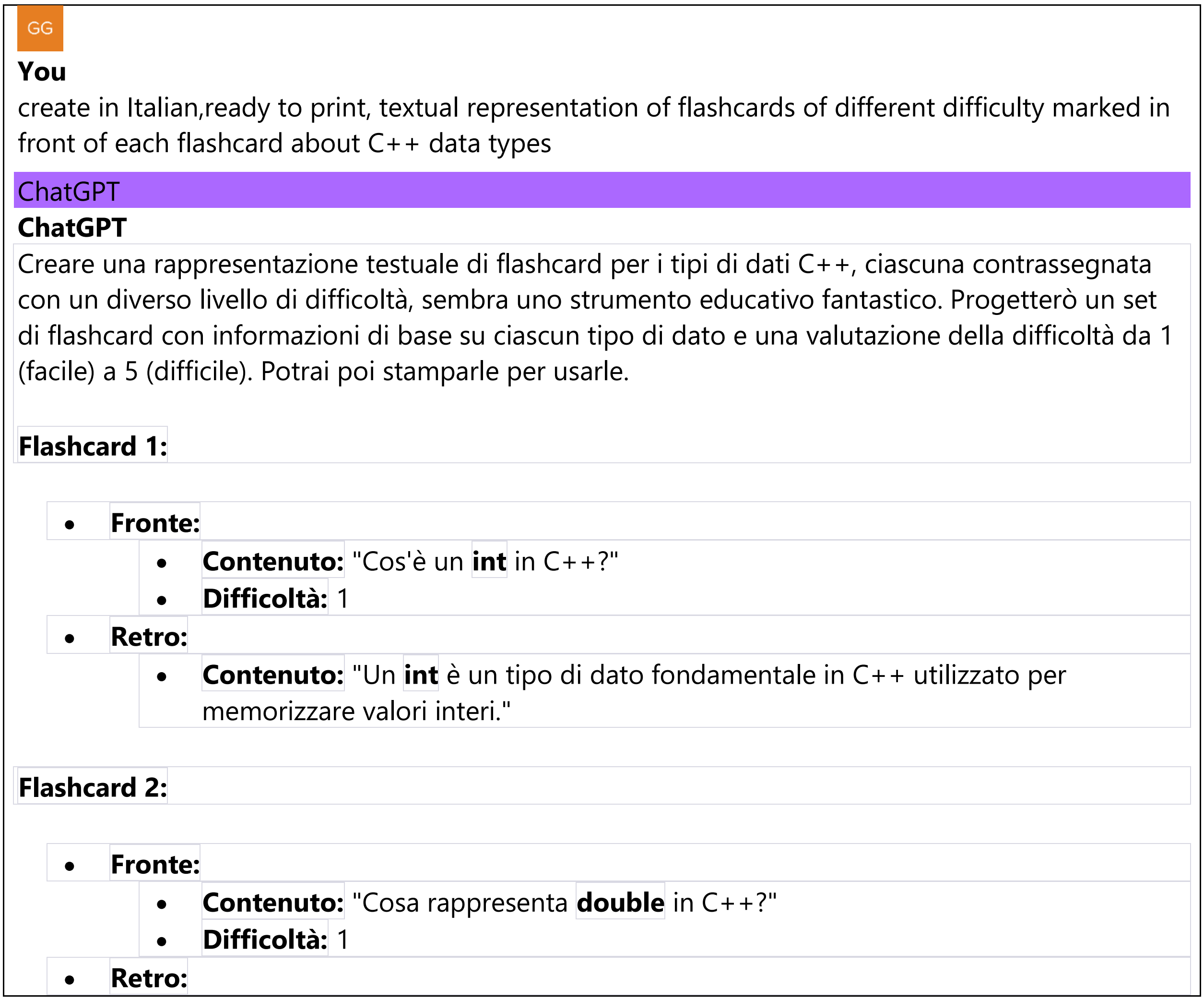
A) International Communication Technology B) Internal Computing Test C) Information and Communication Technology D) Integrated Circuit Technology

Risposta: C) Information and Communication Technology.

Figure 2.5- ChatGPT-4 chat prompt and result of Example 2.

After a revision of result and other similar chat, I validated quiz and prepared to use.

In the Example 3, ChatGPT-4 helped me (Figure 2.6) to produce personalized flashcards about data types in C++ aligned with each student's learning progression, for students with special needs useful to prepare oral questions.



You

create in Italian,ready to print, textual representation of flashcards of different difficulty marked in front of each flashcard about C++ data types

ChatGPT

Creare una rappresentazione testuale di flashcard per i tipi di dati C++, ciascuna contrassegnata con un diverso livello di difficoltà, sembra uno strumento educativo fantastico. Progetterò un set di flashcard con informazioni di base su ciascun tipo di dato e una valutazione della difficoltà da 1 (facile) a 5 (difficile). Potrai poi stamparle per usarle.

Flashcard 1:

- **Fronte:**
  - **Contenuto:** "Cos'è un **int** in C++?"
  - **Difficoltà:** 1
- **Retro:**
  - **Contenuto:** "Un **int** è un tipo di dato fondamentale in C++ utilizzato per memorizzare valori interi."

Flashcard 2:

- **Fronte:**
  - **Contenuto:** "Cosa rappresenta **double** in C++?"
  - **Difficoltà:** 1
- **Retro:**

    - **Contenuto:** "**double** è usato per numeri in virgola mobile con doppia precisione."

Flashcard 3:

- **Fronte:**
    - **Contenuto:** "Spiega **char** in C++."
    - **Difficoltà:** 1
- **Retro:**
    - **Contenuto:** "**char** rappresenta un singolo carattere."

Flashcard 4:

- **Fronte:**
    - **Contenuto:** "Definisci **bool** in C++."
    - **Difficoltà:** 1
- **Retro:**
    - **Contenuto:** "**bool** è un tipo di dato utilizzato per rappresentare valori booleani: **true** o **false**."

Flashcard 5:

- **Fronte:**
    - **Contenuto:** "Qual è lo scopo di **float** in C++?"
    - **Difficoltà:** 2
- **Retro:**
    - **Contenuto:** "**float** è usato per numeri in virgola mobile a precisione singola."

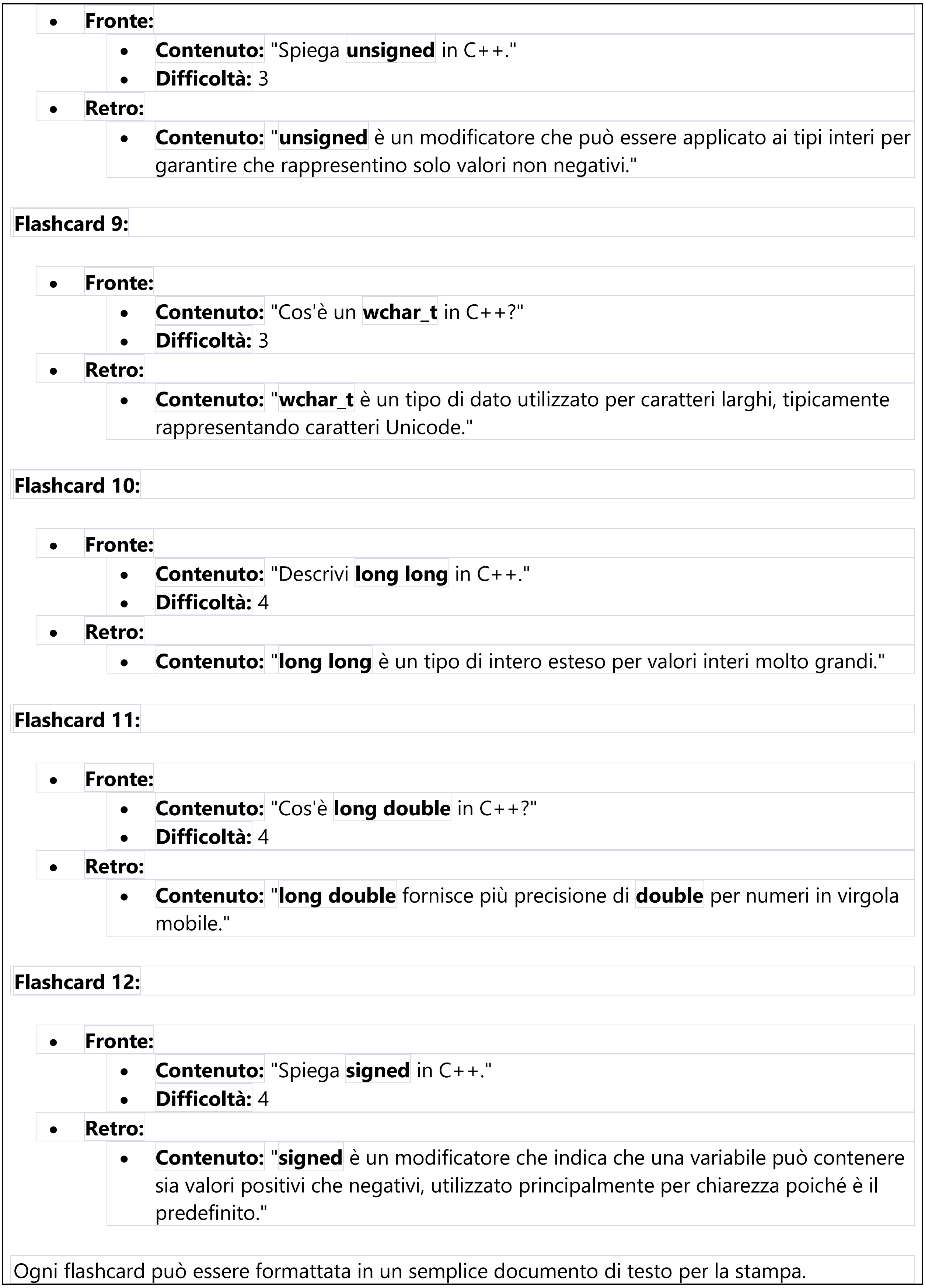
Flashcard 6:

- **Fronte:**
    - **Contenuto:** "Descrivi **long** in C++."
    - **Difficoltà:** 2
- **Retro:**
    - **Contenuto:** "**long** è una dimensione estesa di intero che può memorizzare valori più grandi di **int**."

Flashcard 7:

- **Fronte:**
    - **Contenuto:** "Cosa significa **short** in C++?"
    - **Difficoltà:** 2
- **Retro:**
    - **Contenuto:** "**short** è un tipo di dato intero più piccolo, utilizzato quando la conservazione della memoria è importante."

Flashcard 8:



- **Fronte:**
    - **Contenuto:** "Spiega **unsigned** in C++."
    - **Difficoltà:** 3
- **Retro:**
    - **Contenuto:** "**unsigned** è un modificatore che può essere applicato ai tipi interi per garantire che rappresentino solo valori non negativi."

Flashcard 9:

- **Fronte:**
    - **Contenuto:** "Cos'è un **wchar_t** in C++?"
    - **Difficoltà:** 3
- **Retro:**
    - **Contenuto:** "**wchar_t** è un tipo di dato utilizzato per caratteri larghi, tipicamente rappresentando caratteri Unicode."

Flashcard 10:

- **Fronte:**
    - **Contenuto:** "Descrivi **long long** in C++."
    - **Difficoltà:** 4
- **Retro:**
    - **Contenuto:** "**long long** è un tipo di intero esteso per valori interi molto grandi."

Flashcard 11:

- **Fronte:**
    - **Contenuto:** "Cos'è **long double** in C++?"
    - **Difficoltà:** 4
- **Retro:**
    - **Contenuto:** "**long double** fornisce più precisione di **double** per numeri in virgola mobile."

Flashcard 12:

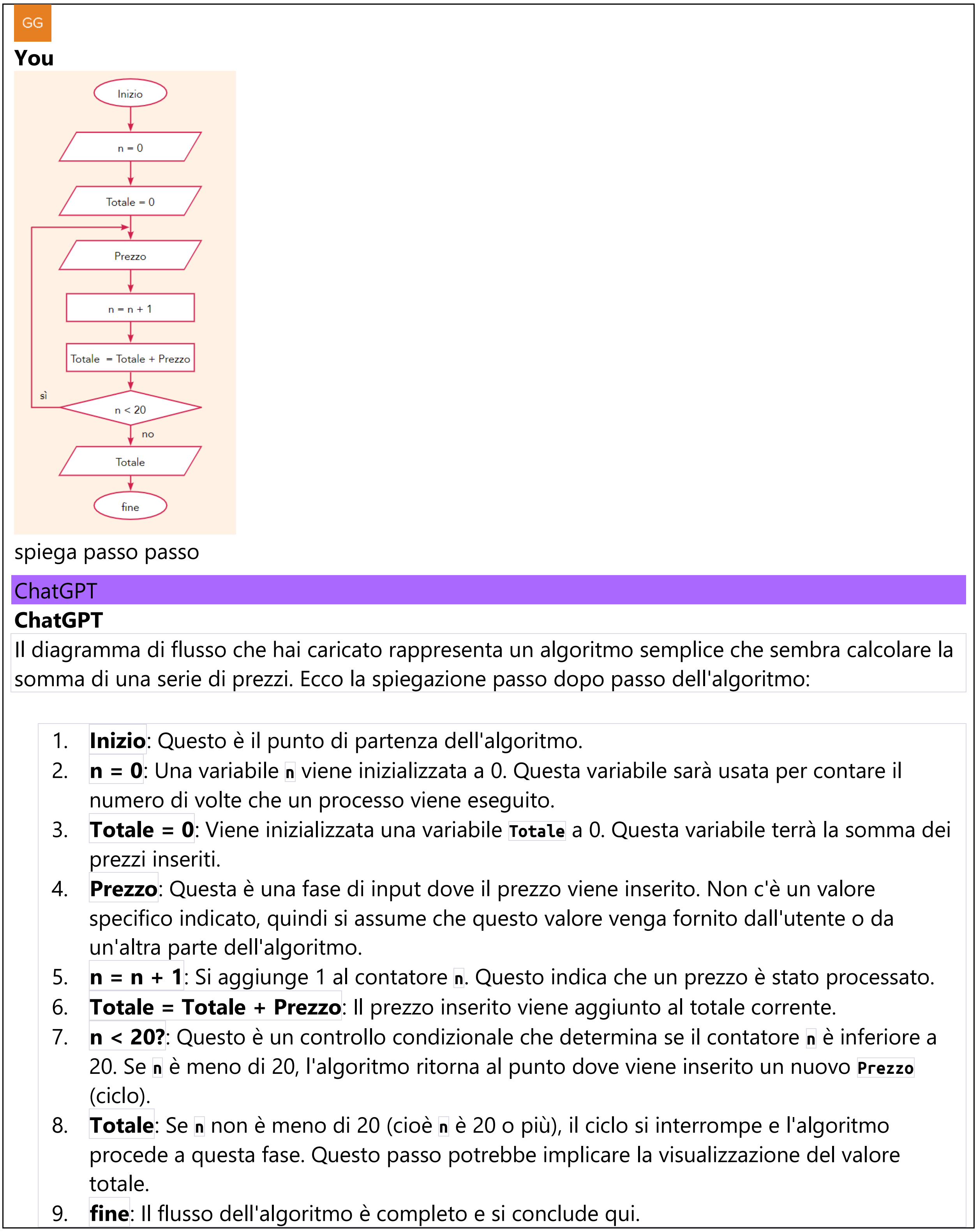
- **Fronte:**
    - **Contenuto:** "Spiega **signed** in C++."
    - **Difficoltà:** 4
- **Retro:**
    - **Contenuto:** "**signed** è un modificatore che indica che una variabile può contenere sia valori positivi che negativi, utilizzato principalmente per chiarezza poiché è il predefinito."

Ogni flashcard può essere formattata in un semplice documento di testo per la stampa.

Figure 2.6- ChatGPT-4 chat prompt and result of Example 3.

After a revision of result and other similar chat, I validated flashcards and prepared to print.

In the Example 4, ChatGPT-4 helped me (Figure 2.7) to produce an explanation of a flowchart take from education book: Atlas Pro.tech Vol.1.



You

spiega passo passo

ChatGPT

Il diagramma di flusso che hai caricato rappresenta un algoritmo semplice che sembra calcolare la somma di una serie di prezzi. Ecco la spiegazione passo dopo passo dell'algoritmo:

1. **Inizio**: Questo è il punto di partenza dell'algoritmo.
2. **n = 0**: Una variabile `n` viene inizializzata a 0. Questa variabile sarà usata per contare il numero di volte che un processo viene eseguito.
3. **Totale = 0**: Viene inizializzata una variabile `Totale` a 0. Questa variabile terrà la somma dei prezzi inseriti.
4. **Prezzo**: Questa è una fase di input dove il prezzo viene inserito. Non c'è un valore specifico indicato, quindi si assume che questo valore venga fornito dall'utente o da un'altra parte dell'algoritmo.
5. **n = n + 1**: Si aggiunge 1 al contatore `n`. Questo indica che un prezzo è stato processato.
6. **Totale = Totale + Prezzo**: Il prezzo inserito viene aggiunto al totale corrente.
7. **n < 20?**: Questo è un controllo condizionale che determina se il contatore `n` è inferiore a 20. Se `n` è meno di 20, l'algoritmo ritorna al punto dove viene inserito un nuovo `Prezzo` (ciclo).
8. **Totale**: Se `n` non è meno di 20 (cioè `n` è 20 o più), il ciclo si interrompe e l'algoritmo procede a questa fase. Questo passo potrebbe implicare la visualizzazione del valore totale.
9. **fine**: Il flusso dell'algoritmo è completo e si conclude qui.

In sostanza, questo algoritmo calcola la somma di 20 prezzi inseriti, mantenendo un conteggio di quanti prezzi sono stati processati e termina quando questo conteggio raggiunge 20.

Figure 2.7- ChatGPT-4 chat prompt and result of Example 4.

After a revision of result, I validated flowchart explanation and prepared to use.

In Example 5, ChatGPT-4 helped me (Figure 2.9) to write a simple source code in C++ about understand array of structs (Figure 2.8) and a basic explanation, useful for BES students, needed to generate a solution.

Realizzare un programma in C++ che gestisca una biblioteca. Per ogni Libro si vuole memorizzare mediante una struttura:

- Titolo;
- Autore;
- Genere (Romanzo, Giallo, Manuale…..);
- Prezzo
- Anno pubblicazione

Dopo aver caricato i Libri in un array v[ ] a dieci posizioni, il programma deve presentare un menu di gestione con le seguenti opzioni:

1. Visualizzazione del vettore caricato mediante richiamo della funzione stampa( );
2. Stampare tutti i dati relativi al libro più vecchio (anno di pubblicazione minore);
3. Stampare tutti i dati dei libri di un Genere immesso da tastiera dall’utente;

Terminare il programma.

Figure 2.8- Array of struct program definition of Example 5.

GG

You

Realizzare un programma in C++ che gestisca una biblioteca. Per ogni Libro si vuole memorizzare mediante una struttura: - Titolo; - Autore; - Genere (Romanzo, Giallo, Manuale…..); - Prezzo - Anno pubblicazione Dopo aver caricato i Libri in un vettore v[ ] a dieci posizioni, il programma deve presentare un menu di gestione con le seguenti opzioni: 1. Visualizzazione del vettore caricato mediante richiamo della funzione stampa( ); 2. Stampare tutti i dati relativi al libro più vecchio (anno di pubblicazione minore); 3. Stampare tutti i dati dei libri di un Genere immesso da tastiera dall’utente; 4. Terminare il programma

ChatGPT

Ecco un esempio di programma in C++ che soddisfa le tue richieste. Questo programma definisce una struttura `Libro` per memorizzare le informazioni su ogni libro. Implementa inoltre un menu per permettere all'utente di visualizzare tutti i libri, trovare il libro più vecchio, visualizzare i libri di un genere specifico, o uscire dal programma.

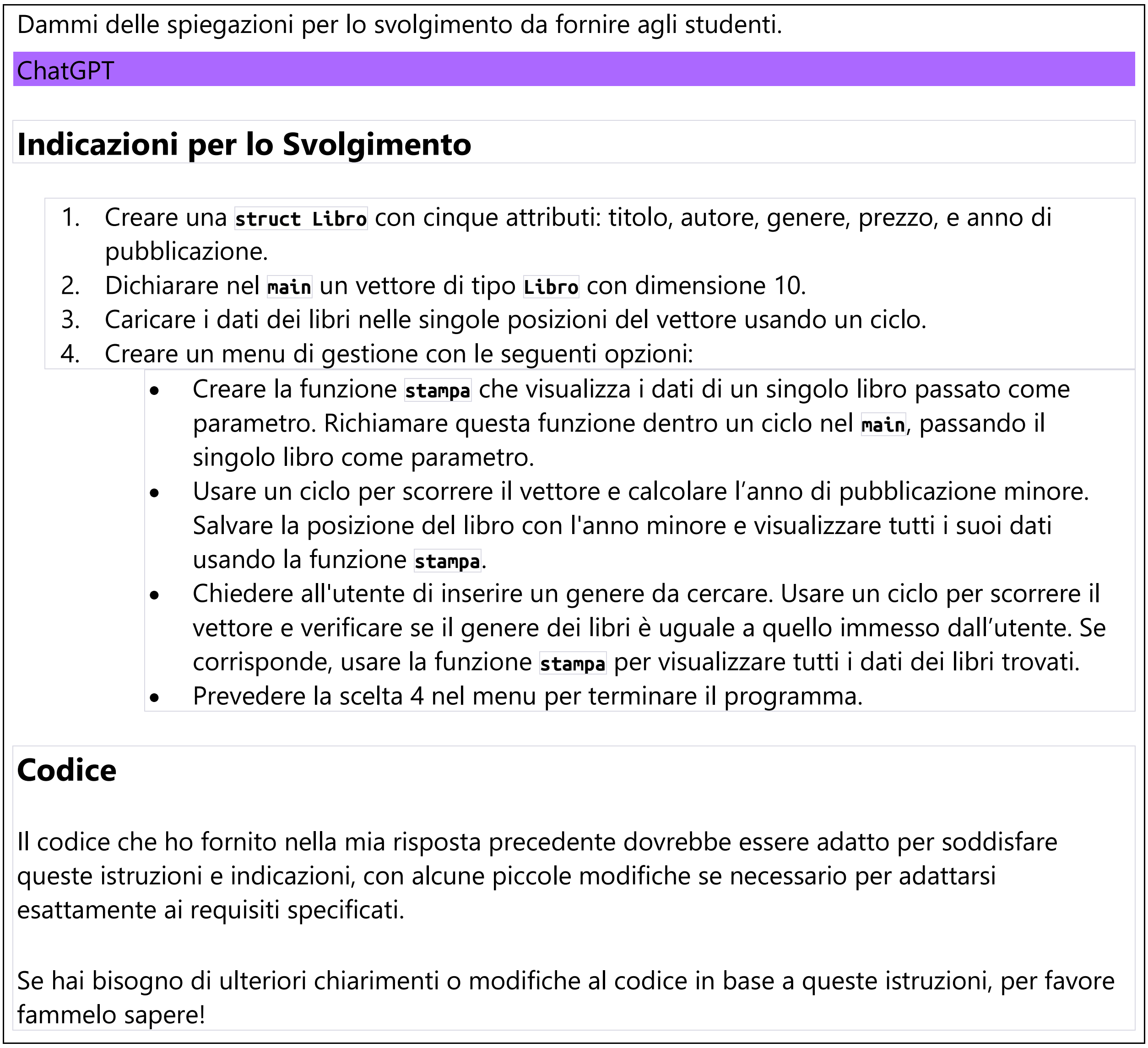
cppCopy code

```cpp
#include <iostream>
#include <string>
#include <vector>

struct Libro {
    std::string titolo;
    std::string autore;
    std::string genere;
    double prezzo;
    int annoPubblicazione;
};

void stampa(const std::vector<Libro>& libri) {
    for (const auto& libro : libri) {
        std::cout << "Titolo: " << libro.titolo << std::endl;
        std::cout << "Autore: " << libro.autore << std::endl;
        std::cout << "Genere: " << libro.genere << std::endl;
        std::cout << "Prezzo: " << libro.prezzo << std::endl;
        std::cout << "Anno di Pubblicazione: " << libro.annoPubblicazione << std::endl;
        std::cout << "------------------------------------" << std::endl;
    }
}

int main() {
    std::vector<Libro> v(10); // Crea un vettore di 10 libri

    // Popola il vettore con i libri
    // (in un'applicazione reale, potresti voler ottenere questi dati in un modo diverso, ad esempio da
un file o un database)
    for (int i = 0; i < 10; i++) {
        std::cout << "Inserisci il titolo del libro " << i+1 << ": ";
        std::getline(std::cin, v[i].titolo);
        std::cout << "Inserisci l'autore del libro " << i+1 << ": ";
        std::getline(std::cin, v[i].autore);
        std::cout << "Inserisci il genere del libro " << i+1 << ": ";
        std::getline(std::cin, v[i].genere);
        std::cout << "Inserisci il prezzo del libro " << i+1 << ": ";
        std::cin >> v[i].prezzo;
        std::cout << "Inserisci l'anno di pubblicazione del libro " << i+1 << ": ";
        std::cin >> v[i].annoPubblicazione;
        std::cin.ignore(); // Pulisce il buffer di input
    }

    int scelta;
    do {
        std::cout << "\nMenu:\n";
        std::cout << "1. Visualizza tutti i libri\n";
        std::cout << "2. Visualizza il libro più vecchio\n";
        std::cout << "3. Visualizza libri di un genere specifico\n";
        std::cout << "4. Esci\n";
        std::cout << "Scegli un'opzione: ";
        std::cin >> scelta;
        std::cin.ignore(); // Pulisce il buffer di input

        switch (scelta) {
            case 1:
                stampa(v);
                break;
        case 2:
          {
            Libro piuVecchio = v[0];
            for (const auto& libro : v) {
              if (libro.annoPubblicazione < piuVecchio.annoPubblicazione) {
                piuVecchio = libro;
              }
            }
            std::cout << "\nIl libro più vecchio è:\n";
            std::cout << "Titolo: " << piuVecchio.titolo << std::endl;
            std::cout << "Autore: " << piuVecchio.autore << std::endl;
            std::cout << "Genere: " << piuVecchio.genere << std::endl;
            std::cout << "Prezzo: " << piuVecchio.prezzo << std::endl;
            std::cout << "Anno di Pubblicazione: " << piuVecchio.annoPubblicazione << std::endl;
          }
          break;
        case 3:
          {
            std::string genere;
            std::cout << "Inserisci il genere: ";
            std::getline(std::cin, genere);
            std::cout << "\nLibri del genere " << genere << ":\n";
            for (const auto& libro : v) {
              if (libro.genere == genere) {
                std::cout << "Titolo: " << libro.titolo << std::endl;
                std::cout << "Autore: " << libro.autore << std::endl;
                std::cout << "Prezzo: " << libro.prezzo << std::endl;
                std::cout << "Anno di Pubblicazione: " << libro.annoPubblicazione << std::endl;
                std::cout << "-------------------------------------" << std::endl;
              }
            }
          }
          break;
        case 4:
          std::cout << "Arrivederci!\n";
          break;
        default:
          std::cout << "Scelta non valida. Riprova.\n";
          break;
      }
  } while (scelta != 4);

  return 0;
}
```

Nota: questo è un esempio di base e potrebbe richiedere ulteriori migliorie e personalizzazioni per adattarsi alle tue esigenze specifiche. Ad esempio, potrebbe essere utile gestire errori di input, come valori non validi o stringhe vuote, e aggiungere ulteriori funzionalità per una gestione più completa della biblioteca.

GG

You



Dammi delle spiegazioni per lo svolgimento da fornire agli studenti.

ChatGPT

Indicazioni per lo Svolgimento

1. Creare una `struct Libro` con cinque attributi: titolo, autore, genere, prezzo, e anno di pubblicazione.
2. Dichiarare nel `main` un vettore di tipo `Libro` con dimensione 10.
3. Caricare i dati dei libri nelle singole posizioni del vettore usando un ciclo.
4. Creare un menu di gestione con le seguenti opzioni:
    - Creare la funzione `stampa` che visualizza i dati di un singolo libro passato come parametro. Richiamare questa funzione dentro un ciclo nel `main`, passando il singolo libro come parametro.
    - Usare un ciclo per scorrere il vettore e calcolare l'anno di pubblicazione minore. Salvare la posizione del libro con l'anno minore e visualizzare tutti i suoi dati usando la funzione `stampa`.
    - Chiedere all'utente di inserire un genere da cercare. Usare un ciclo per scorrere il vettore e verificare se il genere dei libri è uguale a quello immesso dall'utente. Se corrisponde, usare la funzione `stampa` per visualizzare tutti i dati dei libri trovati.
    - Prevedere la scelta 4 nel menu per terminare il programma.

Codice

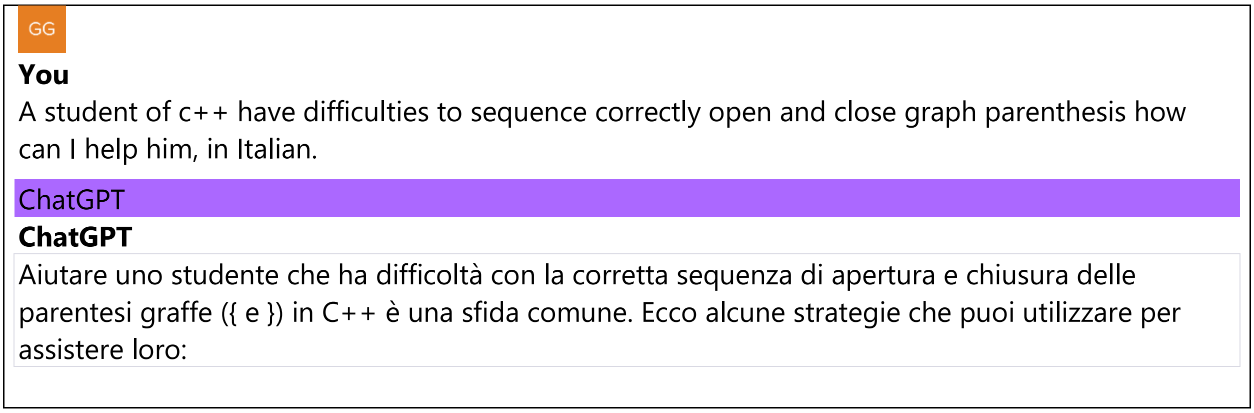
Il codice che ho fornito nella mia risposta precedente dovrebbe essere adatto per soddisfare queste istruzioni e indicazioni, con alcune piccole modifiche se necessario per adattarsi esattamente ai requisiti specificati.

Se hai bisogno di ulteriori chiarimenti o modifiche al codice in base a queste istruzioni, per favore fammelo sapere!

Figure 2.9- ChatGPT-4 chat prompt and result of Example 5.

After a revision of result and testing in Visual Studio Code environment, I validated source code and used for my lesson.

In the Example 6, ChatGPT-4 helped me (Figure 2.10) to try a solution for a BES student to correctly manage graph parenthesis in C++ code.



GG

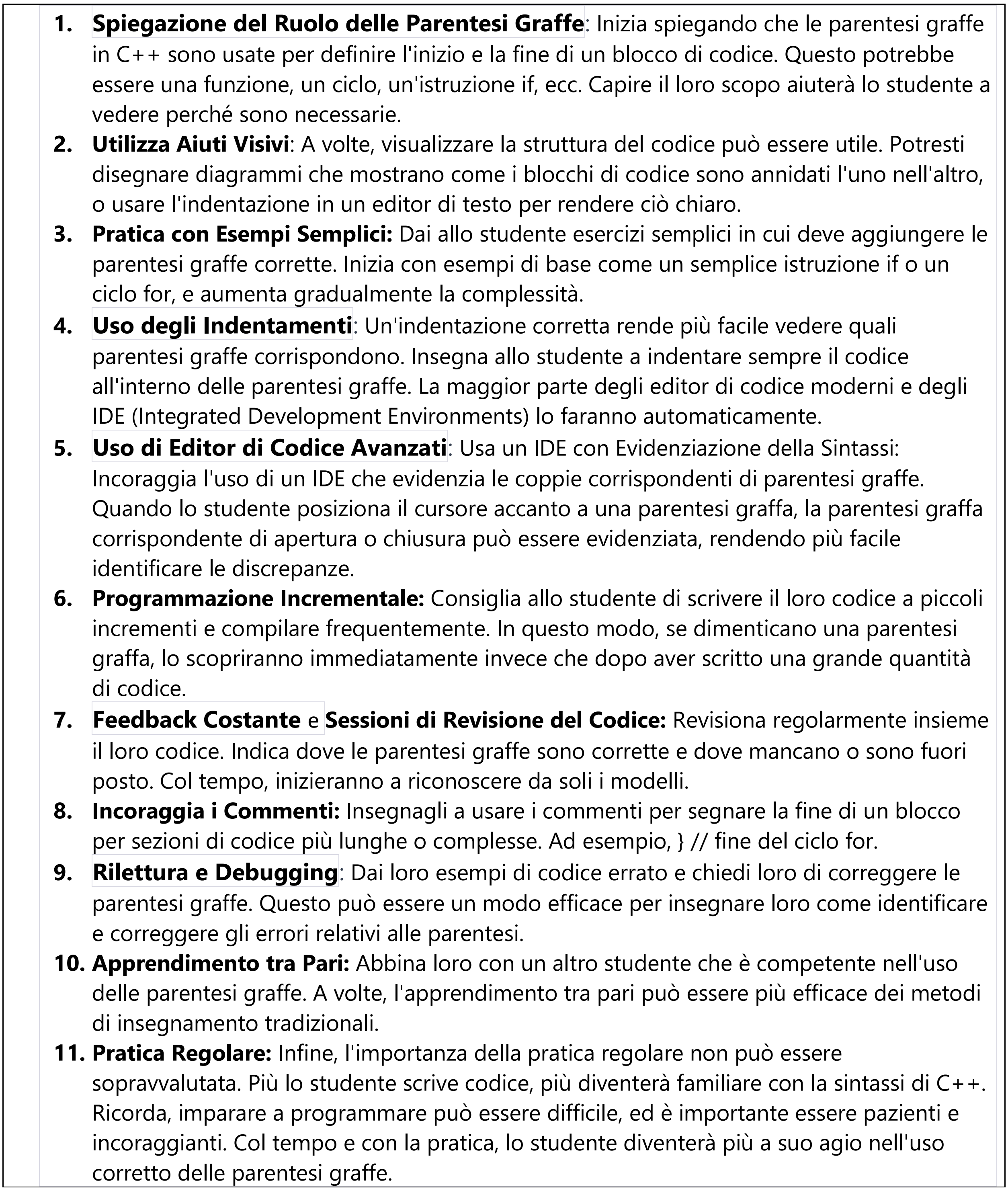
You

A student of c++ have difficulties to sequence correctly open and close graph parenthesis how can I help him, in Italian.

ChatGPT

ChatGPT

Aiutare uno studente che ha difficoltà con la corretta sequenza di apertura e chiusura delle parentesi graffe ({ e }) in C++ è una sfida comune. Ecco alcune strategie che puoi utilizzare per assistere loro:



1. **Spiegazione del Ruolo delle Parentesi Graffe**: Inizia spiegando che le parentesi graffe in C++ sono usate per definire l'inizio e la fine di un blocco di codice. Questo potrebbe essere una funzione, un ciclo, un'istruzione if, ecc. Capire il loro scopo aiuterà lo studente a vedere perché sono necessarie.
2. **Utilizza Aiuti Visivi**: A volte, visualizzare la struttura del codice può essere utile. Potresti disegnare diagrammi che mostrano come i blocchi di codice sono annidati l'uno nell'altro, o usare l'indentazione in un editor di testo per rendere ciò chiaro.
3. **Pratica con Esempi Semplici:** Dai allo studente esercizi semplici in cui deve aggiungere le parentesi graffe corrette. Inizia con esempi di base come un semplice istruzione if o un ciclo for, e aumenta gradualmente la complessità.
4. **Uso degli Indentamenti**: Un'indentazione corretta rende più facile vedere quali parentesi graffe corrispondono. Insegna allo studente a indentare sempre il codice all'interno delle parentesi graffe. La maggior parte degli editor di codice moderni e degli IDE (Integrated Development Environments) lo faranno automaticamente.
5. **Uso di Editor di Codice Avanzati**: Usa un IDE con Evidenziazione della Sintassi: Incoraggia l'uso di un IDE che evidenzia le coppie corrispondenti di parentesi graffe. Quando lo studente posiziona il cursore accanto a una parentesi graffa, la parentesi graffa corrispondente di apertura o chiusura può essere evidenziata, rendendo più facile identificare le discrepanze.
6. **Programmazione Incrementale:** Consiglia allo studente di scrivere il loro codice a piccoli incrementi e compilare frequentemente. In questo modo, se dimenticano una parentesi graffa, lo scopriranno immediatamente invece che dopo aver scritto una grande quantità di codice.
7. **Feedback Costante** e **Sessioni di Revisione del Codice:** Revisiona regolarmente insieme il loro codice. Indica dove le parentesi graffe sono corrette e dove mancano o sono fuori posto. Col tempo, inizieranno a riconoscere da soli i modelli.
8. **Incoraggia i Commenti:** Insegnagli a usare i commenti per segnare la fine di un blocco per sezioni di codice più lunghe o complesse. Ad esempio, } // fine del ciclo for.
9. **Rilettura e Debugging**: Dai loro esempi di codice errato e chiedi loro di correggere le parentesi graffe. Questo può essere un modo efficace per insegnare loro come identificare e correggere gli errori relativi alle parentesi.
10. **Apprendimento tra Pari:** Abbina loro con un altro studente che è competente nell'uso delle parentesi graffe. A volte, l'apprendimento tra pari può essere più efficace dei metodi di insegnamento tradizionali.
11. **Pratica Regolare:** Infine, l'importanza della pratica regolare non può essere sopravvalutata. Più lo studente scrive codice, più diventerà familiare con la sintassi di C++. Ricorda, imparare a programmare può essere difficile, ed è importante essere pazienti e incoraggianti. Col tempo e con la pratica, lo studente diventerà più a suo agio nell'uso corretto delle parentesi graffe.

Figure 2.10- ChatGPT-4 chat prompt and result of Example 6.

After a revision of result, I validated explanation and prepared to use.

In the Example 7, I created ChatGPT-4 link (Figure 2.11) of result chat of example 5 and sent to my fellow Prof. to share knowledge.

https://chat.openai.com/share/a37ae26e-64aa-4552-a2d8-850b3a370a97

Figure 2.11- ChatGPT-4 chat public link of Example 7.

CHAPTER 3

ChatGPT-4: a learning tool for Computer Science education

3.1 - State-of-the-art learning features

Integrating technology in CS education transforms learning into a more accessible, collaborative, and engaging process, significantly enhancing students' learning capabilities and motivation [12]. Technology facilitates access to vast amounts of information, allowing students to explore resources beyond the traditional classroom setting. This accessibility supports a more diverse and enriched learning experience. Technological tools have revolutionized how students collaborate. Platforms that support group work, peer review, and collective research efforts foster a sense of community and shared purpose in learning. This collaborative spirit extends to virtual environments where students can work together regardless of physical distances. Technology offers unique advantages for students requiring special education. Adaptive and assistive technologies can tailor learning experiences to meet individual needs, ensuring that education is accessible to everyone.

ChatGPT-4 can significantly impact traditional learning methods in Computer Science education, thanks to the state-of-the-art learning features grouped in 5 macro-features: interactive tutoring, personalized interactive learning, project ideas and problem solving, programming language learning, documentation support and research assistant. In the tables below I present each feature with relative description, educational benefits, relative applicative example number in the cases study.

Personalized Interactive Tutoring [20][23][27][28][29]

N	Feature Name	Description	Educational Benefits	Example
1	24/7 Availability and Accessibility	Ensures help is available anytime, beneficial for different time zones and geographical locations.	Provides flexible learning opportunities, especially for students with irregular schedules.	ALL
2	Instant Feedback and Adaptive Learning	Provides immediate responses and adapts to individual learning styles for personalized education.	Enhances understanding through immediate correction and personalized learning paths.	1.2.1 1.2.2
3	Multiple Explanation Methods	Offers explanations in different forms (e.g., step-by-step, simulation) to cater to diverse learning styles.	Increases comprehension by catering to individual learning preferences.	2.1.1

4	Interactive Exercises and Assessments	Includes varied interactive exercises and assessments to reinforce learning tailored to student's level.	Engages students actively in their learning process with varied and relevant material.	2.1.1
5	Integration with Educational Resources	Seamlessly integrates with existing educational resources and platforms, enhancing the learning experience.	Offers a comprehensive learning experience by connecting various educational materials.	2.1.1
6	Community and Collaboration Features	Enables community building and collaboration, allowing students to learn from and support each other.	Promotes collaborative learning and peer support, enhancing the overall learning experience.	NO

Project Ideas and Problem Solving [20][23][27][28][29]

N	Feature Name	Description	Educational Benefits	Example
7	Idea Generation Tools	Uses AI to suggest project ideas based on trends, interests, and successful projects.	Encourages creativity and innovation in project development.	1.1.1 1.2.1
8	Problem-Solving Assistance	Guides students through problem-solving processes in complex subjects.	Helps develop critical thinking and analytical skills in problem-solving.	1.1.1 1.1.2
9	Resource Recommendation	Recommends relevant resources such as articles, tutorials, or case studies.	Expands knowledge and exposes students to different methodologies.	1.1.1

Programming Language Learning [20][23][27][28][29]

N	Feature Name	Description	Educational Benefits	Example
10	Code Writing Assistance	Guides students in writing code with syntax and logic suggestions in various languages.	Improves proficiency in writing and understanding code.	1.2.1 1.3.1 2.3.1
11	Code Understanding Support	Provides explanations of code segments to enhance comprehension of programming concepts.	Aids in grasping programming concepts and practices.	1.2.2 1.3.1 2.3.1
12	Debugging Tools	Helps identify and fix errors in code, teaching efficient problem-solving in coding.	Develops critical problem-solving skills in a technical context.	1.3.1 1.3.2
13	Multi-Language Learning	Offers resources and exercises for learning different programming languages at various levels.	Prepares students for working with a variety of programming languages.	3.1
14	Code Translation	Translates code between languages, offering examples and direct translations.	Facilitates easier transition between different coding languages.	3.1

Documentational Support [20][23][27][28][29]

N	Feature Name	Description	Educational Benefits	Example
15	Developer Guide Generation	Compiles developer guides detailing software architecture,	Provides comprehensive insights into software	2.2.1

		dependencies, and development guidelines.	development for future modification or development.	
16	Automated Comment Generation	Generates comments in code explaining functions, methods, and logic for better understanding.	Helps students learn effective code documentation techniques.	2.2.2
17	User Manual Creation	Automatically creates user manuals based on the code to guide end-users.	Facilitates the creation of user-friendly guides for software end-users.	2.2.3
18	Code Quality Analysis	Assesses code quality and suggests improvements, aligned with best coding practices.	Promotes the maintenance of high-quality code in line with professional standards.	1.3.2

Research Assistant [20][23][27][28][29]

N	Feature Name	Description	Educational Benefits	Example
19	Idea Generation for Research	Suggests research topics and questions using AI, based on trends and academic literature.	Helps develop research topics and questions, enhancing creativity in research.	1.1.1 2.1.1
20	Citation and Bibliography Management	Manages citations and bibliographies, streamlining the referencing process.	Ensures proper citation and referencing, crucial for academic integrity.	NO

Table 3.1- Overview of state-of-the-art learning features of ChatGPT-4

As good example of some state-of-the art features (figure 3.1), I present a recent work [19] evaluating the accuracy and performance of code generation models of ChatGPT-4 on typical introductory level programming problems.

Algorithms early. In a traditional Computer Science course, the initial focus usually begins with syntax and basic programming principles, and it can take time for students to become proficient in these fundamentals. If code generation models can be utilized to handle low-level implementation tasks, this may enable students to start focusing on higher-level algorithms earlier. This approach has some similarity with the use of block-based environments for novices, which abstract away the complexities of syntax and allow students to focus on algorithmic issues. As a result, teaching could initially concentrate more on algorithms and problem-solving aspects, using automatic code generation for implementation, and defer in-depth and nuanced discussions of syntax until later.

Specification-focused tasks. A similar shift could be seen away from implementation and towards specification. Novices are typically presented with very carefully specified problems with clear and unambiguous problem statements. Such detailed specifications provide excellent context for code generation models to generate correct code solutions. New types of problems could task students with developing the specifications themselves, perhaps by inferring patterns from provided test suites. The student-written specifications could then be provided to code models for the implementation step, and iteratively refined.

Explaining algorithmic concepts clearly. It is well known that the outputs produced by large language models are very sensitive to their inputs. In fact, "prompt engineering," where effective prompts are crafted, has emerged as a distinct (and nascent) skill when working with these models. For example, when using ChatGPT-4 to solve probability and statistics problems, engineering the prompt to include explicit hints on the strategy for solving a problem is extremely effective. Denny et al. found that prompt engineering strategies which described algorithmic steps were effective for solving programming tasks for which ChatGPT-4 initially generated solutions that were incorrect. Developers are more successful working with ChatGPT-4 when they decompose larger programming statements into smaller tasks and then explicitly prompt ChatGPT-4 for each of the subtasks. It is likely that students will need to develop new skills to communicate effectively with these models. A key skill will be the ability to describe the computational steps they wish to achieve in natural language as a way of guiding the model to produce valid outputs.

A focus on refactoring. Students sometimes have trouble getting started on programming assignments, sometimes referred to as the programmer's writer's block. ChatGPT-4 can help students overcome this barrier by immediately providing starter code, enabling them to build upon existing code rather than starting from scratch with a blank code editor. This approach may require a shift in focus towards tasks such as rewriting, refactoring, and debugging code, but it provides the opportunity to help students maintain momentum in a realistic setting where the ability to evaluate, rewrite, and extend code is often more important than writing every line of code from scratch.

A focus on software testing. Students sometimes submit programs that they do not fully understand. Code generation tools are likely to exacerbate this issue, given the ease with which students can generate large blocks of opaque and untested code. Buck and Stucki argue that students should be gradually exposed to complexity, and that one must understand the fundamental building blocks before creating something larger that uses them. An explicit focus on testing may help students to develop an understanding of the code they are generating with tools like ChatGPT-4. This is certainly not a new pedagogical approach, but AI code generation may impart momentum in this area. Edwards, an advocate for test-driven development (TDD), asserted in 2003 that it can help students "read and comprehend source code, envision how a sequence of statements will behave, and predict how a change to the code will result in a change in behaviour". However, it does not seem that computing educators fully embraced TDD or related pedagogies on a wide scale. Since that time, TDD has become an important programming methodology in the field. A major tenet of TDD is that even when the code base is small and functionality incomplete, the code that does exist runs without error. Since the team develops tests continuously, new code must pass a greater number of tests, leading to increased confidence in the project as it proceeds. The relative ratio of new-code-being-tested to old-code-already-tested decreases over time. This practice may be upended by tools like ChatGPT-4 where students are regenerating large sections of code and therefore invalidating previous tests. A possible solution is incremental testing in which students are rewarded for effort spent writing software tests throughout the development of a project. This regular, early and incremental feedback builds on previous test-driven development practices and may help students better understand the code that they are writing, and the code being generated for them. Large industry shifts combined with the availability of tools like ChatGPT-4 necessitate a renewed focus on test-driven development, not as a luxury, but as a core competency for this new era of teaching computing.

Code generation tools powered by ChatGPT-4 can correctly and reliably solve many typical introductory programming problems. This raises interesting questions for educators including: Just how good are these tools? Can students with no programming knowledge, but who are armed with these tools, pass typical programming-centric assessments that are common in introductory courses? Do we need a different approach? To explore the performance of ChatGPT-4 in the context of introductory programming, we prompted Codex with real exam questions and compared its performance to that of students taking the same exams. We also prompted Codex to solve several variants of a well-known Computer Science-level programming problem (the "Rainfall problem") and examined both the correctness and the variety of solutions produced. My AI wants to know if its grade will be rounded up. We took all questions from two basic Python programming exams that had already been taken by students and provided them as input (verbatim) to Codex. The exam questions involved common Python datatypes including strings, tuples, lists, and dictionaries. They ranged in complexity from simple calculations, such as computing the sum of a series of simulated dice rolls, to more complex data manipulations such as extracting a sorted list of the keys that are mapped to the maximum value in a dictionary. To evaluate the code generated, we executed it against the same set of test cases that were used in assessing the student exams. This follows a similar evaluation approach employed by the Codex developers. If the Codex output differed from the expected output with only a trivial formatting error (for example, a missing comma or period) we made the appropriate correction, much as a student would if using Codex to complete an exam. To contextualize the performance of the Codex model, we calculated the score for its responses in the same way as for real students using the same question weights and accumulated penalties for incorrect submissions. Codex scored 15.7/20 (78.5%) on Exam 1 and 19.5/25 (78.0%) on Exam 2. Figure 1 plots the scores (scaled to a maximum of 100) of 71 students enrolled in the course in 2020 who completed both exams. Codex's score is marked with a red asterisk. Averaging both Exam 1 and Exam 2 performance, Codex ranks 17 amongst the 71 students, placing it within the top quartile of class performance. We observed that some of the Codex answers contained trivial formatting errors. We also observed that Codex performed poorly with problems that disallowed the use of certain language features (e.g. using split() to tokenize a string). Codex often did not produce code that avoided using these restricted features, and thus the model (in these cases) often did not pass the auto-grader. Codex also performed poorly when asked to produce formatted ASCII output such as patterns of characters forming geometric shapes, especially where the requirements were not specified in the problem description but had to be inferred from the provided example inputs and outputs.

Yes, We definitely wrote this code myself. To understand the amount of variation in the responses, we provided Codex with seven variants of the problem description for the well-studied Rainfall problem (which averages values in a collection) a total of 50 times each, generating 350 responses. Each response was executed against 10 test cases (a total of 3500 evaluations). Across all variants, Codex had an average score close to 50%. Codex performed poorly on cases where no valid values were provided as input (e.g. where the collection to be averaged was empty). We also examined the number of source lines of code for all Rainfall variants, excluding blank and comment lines. In addition, we classified the general algorithmic approach employed in the solutions as an indicator of algorithmic variation. We found that Codex provides a diverse range of responses to the same input prompt. Depending on the prompt, the resulting programs used varied programmatic structures, while ultimately favouring expected methods for each problem variation (i.e., for-loops for processing lists, and while-loops for processing standard input).

Figure 3.1- Web article [19] on accuracy and performance of code generation models of ChatGPT-4.

As student at this University, in the following paragraphs, I present 3 personal case study on how I used some state-of-the-art learning tool features of ChatGPT-4. (Table 3.1, Example column). As specified in Chapter 2, due to the occasional need for multiple interactions with ChatGPT-4 and common troubleshooting issues (Appendices B), especially in the first period of this version release (from February 2023 to last 2023), I have included only the most relevant chat excerpts for each example. These examples were developed following the best practices outlined in OpenAI's prompt engineering guidelines (Appendices C).

3.2 - Case study 1: self-tested code for React app in MVVM architecture pattern.

In the fast-paced world of software development, the ability to rapidly produce reliable and efficient code is invaluable. With the advent of advanced language models like ChatGPT-4, developers now have a powerful tool at their disposal. Among those leading the charge to integrate this technology into the software development lifecycle is Xu Hao. He has pioneered a method that not only optimizes the process of code generation but also ensures that the produced code is self-tested and robust.

The article [30] instruction reads: "Xu Hao shows how he drives ChatGPT-4 to produce useful self-tested code. His initial prompt primes the LLM with an implementation strategy (chain of thought prompting). His prompt also asks for an implementation plan rather than code (general knowledge prompting). Once he has the plan, he uses it to refine the implementation and generate useful sections of code."

I started with this prompt (Figure 3.2) that sets the context for the application and how the article wants the code to be structured.

```
The current system is an online whiteboard system. Tech stack: typescript, react, redux,
konvajs and react-konva. And vitest, react testing library for model, view model and related
hooks, cypress component tests for view. All codes should be written in the tech stack
mentioned above. Requirements should be implemented as react components in the MVVM
architecture pattern.

There are 2 types of view model in the system.

        1.  Shared view model. View model that represents states shared among local and
            remote users.

        2.  Local view model. View model that represents states only applicable to local
            user

Here is the common implementation strategy:

        1.  Shared view model is implemented as Redux store slice. Tested in vitest.

        2.  Local view model is implemented as React component props or states (by useState
            hook), unless for global local view model, which is also implemented as Redux
            store slice. Tested in vitest.

        3.  Hooks are used as the major view helpers to retrieve data from shared view
            model. For most the case, it will use ‘createSelector’ and ‘useSelector’ for
            memorization. Tested in vitest and react testing library.

        4.  Don’t dispatch action directly to change the states of shared view model, use an
            encapsulated view model interface instead. In the interface, each redux action
            is mapped to a method. Tested in vitest.
```

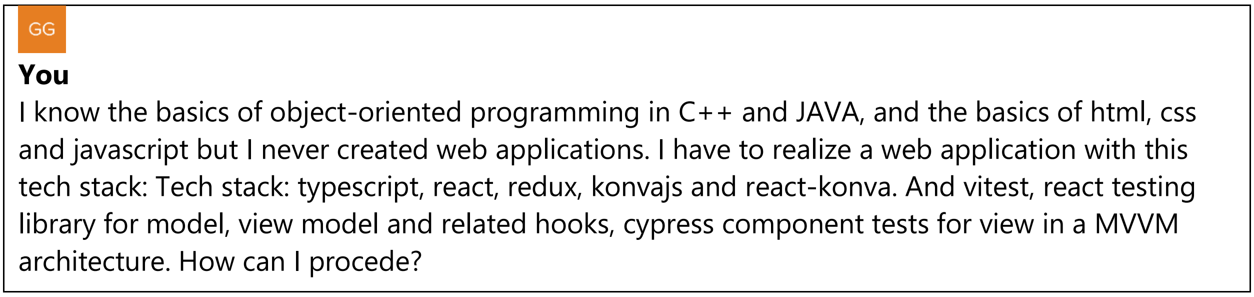
```
       5.  View is consisting of konva shapes and implemented as react component via react-
           konva. Tested in cypress component tests.

Here are certain patterns should be followed when implement and test the component.

       1.  When write test, use describe instead of test.

       2.  Data-driven tests are preferred.

       3.  When test the view component, fake view model via the view model interface.

Awareness Layer

Requirement:

Display other users’ awareness info (cursor, name and online information) on the
whiteboard.

AC1: Don’t display local user.

AC2: When remote user changes cursor location, display the change in animation.

Provide an overall solution following the guidance mentioned above. Hint, keep all
awareness information in a Konva layer, and an awareness info component to render cursor,
and name. Don’t generate code. Describe the solution and breaking the solution down as a
task list based on the guidance mentioned above. And we will refer this task list as our
master plan.”
```

Figure 3.2- First ChatGPT-4 chat prompt.

After, following the methodology outlined in the article, I proceeded in 3 steps. For each step, I present an example of ChatGPT-4 interaction that assisted me in developing the web application.

1) I asked to ChatGPT-4 information about Tech stack and Model-View-ViewModel (MVVM) architecture and development environment.
2) I asked to ChatGPT-4 to give me project folders and files scheme and source code for each file.
3) After completed project I asked to ChatGPT-4 to solve every compiler and run error.

In the example 1.1.1 (case study 1, step 1, example 1), I asked ChatGPT-4 using plugin Advanced Data Analysis (Figure 3,3) to guide me on how to precede to develop desired app starting from a description of his architecture.



GG

You

I know the basics of object-oriented programming in C++ and JAVA, and the basics of html, css and javascript but I never created web applications. I have to realize a web application with this tech stack: Tech stack: typescript, react, redux, konvajs and react-konva. And vitest, react testing library for model, view model and related hooks, cypress component tests for view in a MVVM architecture. How can I procede?

ChatGPT

ChatGPT

Starting a web application project with a tech stack involving TypeScript, React, Redux, KonvaJS, React-Konva, Vitest, React Testing Library, and Cypress in an MVVM architecture requires a structured approach. Since you already have basic knowledge of HTML, CSS, JavaScript, and object-oriented programming in C++ and Java, you're well-positioned to tackle this. Here's a step-by-step guide to get you started:

1. Understand the Technologies and Architecture

TypeScript

- TypeScript is a superset of JavaScript that adds static types. Learn about types, interfaces, and how TypeScript improves JavaScript development.
- **Resources**: TypeScript Handbook

2. **React**

- Learn about React's component-based architecture, JSX, state, and props.
- **Resources**: React Official Docs

3. **Redux**

- Redux is a state management library. Learn actions, reducers, and the store.
- **Resources**: Redux Essentials

4. **KonvaJS and React-Konva**

- KonvaJS is a drawing library, and React-Konva helps use Konva with React.
- **Resources**: KonvaJS, React-Konva

5. **Testing Libraries (Vitest, React Testing Library, Cypress)**

- Learn how to write tests using these libraries.
- **Resources**: Vitest, React Testing Library, Cypress

2. Set Up Your Development Environment

- Install Node.js and npm (Node Package Manager).
- Set up an IDE or editor that supports TypeScript and React (like Visual Studio Code).
- Use Create React App with the TypeScript template or set up manually with Webpack or Vite.
- `npx create-react-app my-app --template typescript`.
- Use npm or yarn to install Redux, Konva, React-Konva, testing libraries.
- Example: `npm install redux react-redux konva react-konva @testing-library/react vitest cypress`

3. Learn and Implement Each Technology

- **React & TypeScript**: Start by building simple React components using TypeScript.
- **Redux**: Implement Redux for state management. Start with simple use cases and gradually move to more complex scenarios.

- **KonvaJS & React-Konva**: Experiment by creating simple canvas drawings and then integrate them into your React components.
- **Vitest & React Testing Library**: Write unit tests for your components and hooks.
- **Cypress**: Develop end-to-end tests for your application.

4. Design the Application Using MVVM

- **Model**: Represents the data and business logic of your application.
- **View**: UI components (React components).
- **ViewModel**: An abstraction of the view exposing public properties and commands.

5. Start Building Your Application

- Begin with a basic layout and incrementally add features.
- Keep the MVVM structure in mind while developing.
- Regularly test your application using Vitest, React Testing Library, and Cypress.

6. Implement Features Using React, Redux, and KonvaJS

- Implement interactive features using React and Redux.
- Use KonvaJS for any canvas-based features.

7. Testing and Debugging

- Write tests as you develop.
- Debug using browser tools and IDE features.

8. Refine and Iterate

- Continuously test, refactor, and enhance your application.
- Seek feedback and make improvements.

9. Deployment

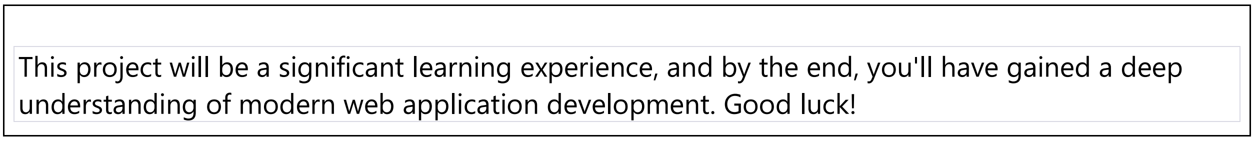
- Once your application is ready, deploy it. You might use services like Netlify, Vercel, or AWS.

Resources

- Official documentation for each technology.
- Online courses and tutorials (e.g., Udemy, Coursera, freeCodeCamp).
- Community forums and Stack Overflow for specific issues.

Additional Tips

- Follow the MVVM architecture strictly - keep the model, view, and view model distinct and clean.
- Read the official documentation of each library/tool you are using - it’s the best way to understand how they work.
- Look for sample projects or tutorials using the specific tech stack.

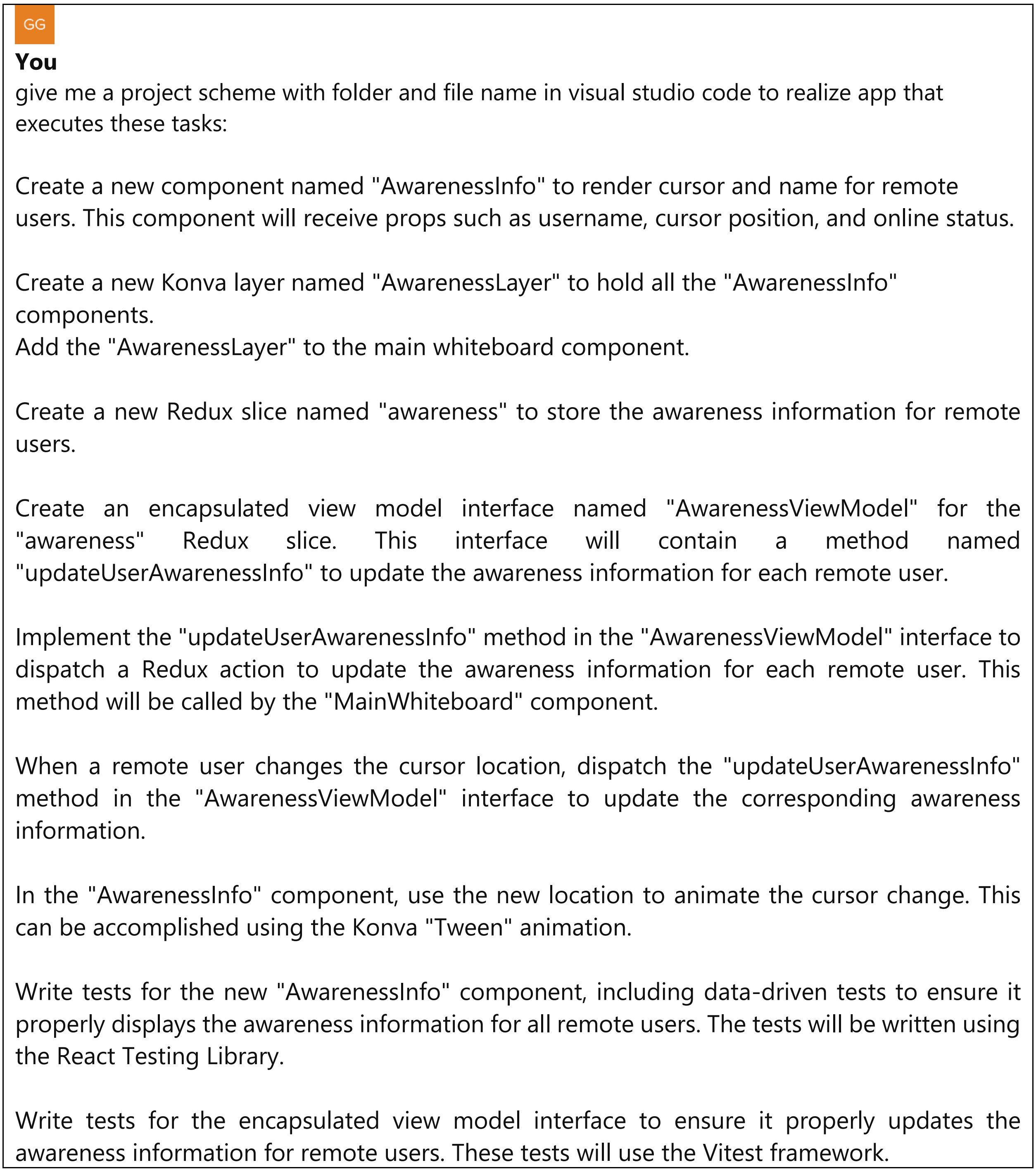


This project will be a significant learning experience, and by the end, you'll have gained a deep understanding of modern web application development. Good luck!

Figure 3.3- ChatGPT-4 chat prompt and result of Example 1.1.1.

after further chats and reviewing feedback on official documentation available online, I decided to proceed as recommended by CharGPT-4.

In the Example 1.2.1, I asked to ChatGPT-4 (Figure 3.4) to guide me to define project folders and files scheme.



You

give me a project scheme with folder and file name in visual studio code to realize app that executes these tasks:

Create a new component named "AwarenessInfo" to render cursor and name for remote users. This component will receive props such as username, cursor position, and online status.

Create a new Konva layer named "AwarenessLayer" to hold all the "AwarenessInfo" components.
Add the "AwarenessLayer" to the main whiteboard component.

Create a new Redux slice named "awareness" to store the awareness information for remote users.

Create an encapsulated view model interface named "AwarenessViewModel" for the "awareness" Redux slice. This interface will contain a method named "updateUserAwarenessInfo" to update the awareness information for each remote user.

Implement the "updateUserAwarenessInfo" method in the "AwarenessViewModel" interface to dispatch a Redux action to update the awareness information for each remote user. This method will be called by the "MainWhiteboard" component.

When a remote user changes the cursor location, dispatch the "updateUserAwarenessInfo" method in the "AwarenessViewModel" interface to update the corresponding awareness information.

In the "AwarenessInfo" component, use the new location to animate the cursor change. This can be accomplished using the Konva "Tween" animation.

Write tests for the new "AwarenessInfo" component, including data-driven tests to ensure it properly displays the awareness information for all remote users. The tests will be written using the React Testing Library.

Write tests for the encapsulated view model interface to ensure it properly updates the awareness information for remote users. These tests will use the Vitest framework.

Write end-to-end Cypress tests to ensure the awareness layer is properly displayed and updated when remote users change their cursor location. The tests will simulate user interactions with the whiteboard and verify the state of the "AwarenessLayer" and "AwarenessInfo" components.

ChatGPT

ChatGPT

For a Visual Studio Code project that encompasses all these tasks, a well-organized directory and file structure is essential. The following is a proposed project scheme that aligns with your requirements:

```
/your-app-name
|-- /src
    |-- /components
        |-- /AwarenessInfo
            |-- AwarenessInfo.jsx
            |-- AwarenessInfo.styles.css
        |-- /AwarenessLayer
            |-- AwarenessLayer.jsx
            |-- AwarenessLayer.styles.css
        |-- /MainWhiteboard
            |-- MainWhiteboard.jsx
            |-- MainWhiteboard.styles.css
    |-- /redux
        |-- /awarenessSlice
            |-- awarenessSlice.js
            |-- awarenessActions.js
            |-- awarenessReducer.js
    |-- /viewModels
        |-- AwarenessViewModel.js
    |-- /tests
        |-- /components
            |-- AwarenessInfo.test.jsx
            |-- AwarenessLayer.test.jsx
            |-- MainWhiteboard.test.jsx
        |-- /redux
            |-- awarenessSlice.test.js
        |-- /viewModels
            |-- AwarenessViewModel.test.js
        |-- /e2e
            |-- awarenessLayer.e2e.js
|-- /public
    |-- index.html
|-- package.json
|-- README.md
```

Here's a brief explanation of each directory and file:

- `src`: This is the main source directory for your application.
    - `components`: This directory contains all React components.
        - `AwarenessInfo`: This sub-directory holds the `AwarenessInfo` component and its associated styles.
        - `AwarenessLayer`: This sub-directory holds the `AwarenessLayer` component and its associated styles.

        - `MainWhiteboard`: This sub-directory contains the main whiteboard component.
    - `redux`: This directory contains all Redux-related files.
        - `awarenessSlice`: This sub-directory includes files for the Redux slice managing awareness.
    - `viewModels`: This directory contains view models, like the `AwarenessViewModel`.
- `tests`: This directory is for all your test files.
    - `components`: Tests for React components.
    - `redux`: Tests for Redux slices.
    - `viewModels`: Tests for view models.
    - `e2e`: End-to-end tests using Cypress.
- `public`: This directory holds static files like `index.html`.
- `package.json`: This file manages project dependencies and scripts.
- `README.md`: A Markdown file for documenting your project.

This structure is modular and scalable, making it suitable for a project with multiple components and complex state management.

Figure 3.4- ChatGPT-4 chat prompt and result of Example 1.2.1.

in this case, I decided to follow the instructions and subsequently evaluate the quality of the result.

In the example 1.2.2, I asked to ChatGPT-4 (Figure 3.5) to give me source code for first file.

GG

You

give me code inside MainWhiteboard.test.tsx

ChatGPT

ChatGPT

Sure, let's create a test file `MainWhiteboard.test.tsx` for the `MainWhiteboard` component. We'll use the `react-testing-library` and `jest` to create unit tests for this component. In these tests, we'll check if the `MainWhiteboard` component properly renders and updates the `AwarenessLayer` when the Redux store changes.

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import { Provider } from 'react-redux';
import { configureStore } from '@reduxjs/toolkit';
import awarenessReducer from '../redux/slices/awarenessSlice';
import MainWhiteboard from '../components/MainWhiteboard';

describe('MainWhiteboard Component', () => {
  let store: any;

  beforeEach(() => {
    store = configureStore({
      reducer: {
        awareness: awarenessReducer,
      },
    });
  });

  it('renders the MainWhiteboard', () => {
    const { getByTestId } = render(
      <Provider store={store}>
        <MainWhiteboard />
      </Provider>
    );

    expect(getByTestId('main-whiteboard')).toBeInTheDocument();
  });

  it('updates AwarenessLayer on Redux store change', () => {
    const { getByTestId } = render(
      <Provider store={store}>
        <MainWhiteboard />
      </Provider>
    );

    fireEvent.click(getByTestId('main-whiteboard'));
    expect(store.getState().awareness).not.toEqual({});
  });
});
```

In this code:

- We import necessary libraries and modules, including `react`, `react-redux`, `@testing-library/react`, and `configureStore` from `@reduxjs/toolkit`.
- We set up a test Redux store with the `awareness` reducer before each test.
- In the first test (`renders the MainWhiteboard`), we render the `MainWhiteboard` component and check that it is in the document. For this test to work, you need to add `data-testid="main-whiteboard"` to the top-level element of the `MainWhiteboard` component.
- In the second test (`updates AwarenessLayer on Redux store change`), we simulate a click on the `MainWhiteboard` component (you need to handle this click to update the Redux store in your `MainWhiteboard` component). We then check that the `awareness` state in the Redux store is not empty. For this test to work, you need to add functionality to the `MainWhiteboard` component to update the `awareness` state in the Redux store when it is clicked.

Please adjust these tests according to the specifics of your `MainWhiteboard` component and your application's requirements.

Figure 3.5- ChatGPT-4 chat prompt and result of Example 1.2.2.

I subsequently asked for the source codes of all the files included in the project. obviously, once I had finished writing the source code, the program didn't work immediately and so I tried to correct all the compilation errors, always asking ChatGPT-4 to give me instructions on how to resolve the errors.

In the example 1.3.1, I asked to ChatGPT-4 (Figure 3.6) to help me to solve compiler error.

GG

You

this is my AwarenessViewModel.ts code import { updateUserAwarenessInfo } from '../redux/slices/awarenessSlice'; import { AppDispatch } from '../redux/store'; export interface AwarenessViewModel { updateUserAwarenessInfo: (id: string, name: string, cursorPosition: { x: number, y: number }, online: boolean) => void; } export const createAwarenessViewModel = (dispatch: AppDispatch): AwarenessViewModel => ({ updateUserAwarenessInfo: (id, name, cursorPosition, online) => dispatch(updateUserAwarenessInfo({ id, name, cursorPosition, online })), });

ChatGPT

ChatGPT

The `AwarenessViewModel` you've defined looks great! Now, you just need to adjust the `MainWhiteboard.tsx` code to use this view model.

Here is the updated code for `MainWhiteboard.tsx`:

```
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { Stage } from 'react-konva';
import { RootState } from '../redux/store';
import { createAwarenessViewModel, AwarenessViewModel } from
'../viewModels/AwarenessViewModel';
import AwarenessLayer from './AwarenessLayer';

// Assume this is your local data
const localData = [
  {
    id: 'user1',
    name: 'Alice',
    cursorPosition: { x: 50, y: 50 },
    online: true,
  },
  {
    id: 'user2',
    name: 'Bob',
    cursorPosition: { x: 100, y: 100 },
    online: true,
  },
  // other user data...
];

const MainWhiteboard: React.FC = () => {
  const dispatch = useDispatch();
  const awarenessState = useSelector((state: RootState) => state.awareness);
  const viewModel: AwarenessViewModel = createAwarenessViewModel(dispatch);

  useEffect(() => {
    // Update the awareness information for each user
    localData.forEach(({ id, name, cursorPosition, online }) => {
      viewModel.updateUserAwarenessInfo(id, name, cursorPosition, online);
    });
  }, []);

  return (
    <Stage width={window.innerWidth} height={window.innerHeight}>
```

```
      <AwarenessLayer awarenessState={awarenessState} />
      {/* Other layers/components go here... */}
    </Stage>
  );
};

export default MainWhiteboard;
```

In this updated version of `MainWhiteboard.tsx`:

- We create the view model by calling `createAwarenessViewModel(dispatch)`.
- In the `useEffect` hook, we call `updateUserAwarenessInfo` method with each property of the user info object from the `localData` array.

This way, we're using the view model you've defined to update the awareness information. Remember to adjust the import paths (`../redux/store`, `../viewModels/AwarenessViewModel`, `./AwarenessLayer`) according to your project structure.

Figure 3.6- ChatGPT-4 chat prompt and result of Example 1.3.1.

Finally, after numerous chats, I managed to compile the project but unfortunately the result was not as expected and I was faced with run errors.

In the example 1.3.2, I asked to ChatGPT-4 (Figure 3.7) to help me to solve run error and execute correctly self-test with Vitest and Cypress tools.

GG

You

when I execute app I view title but I view only blank page

ChatGPT

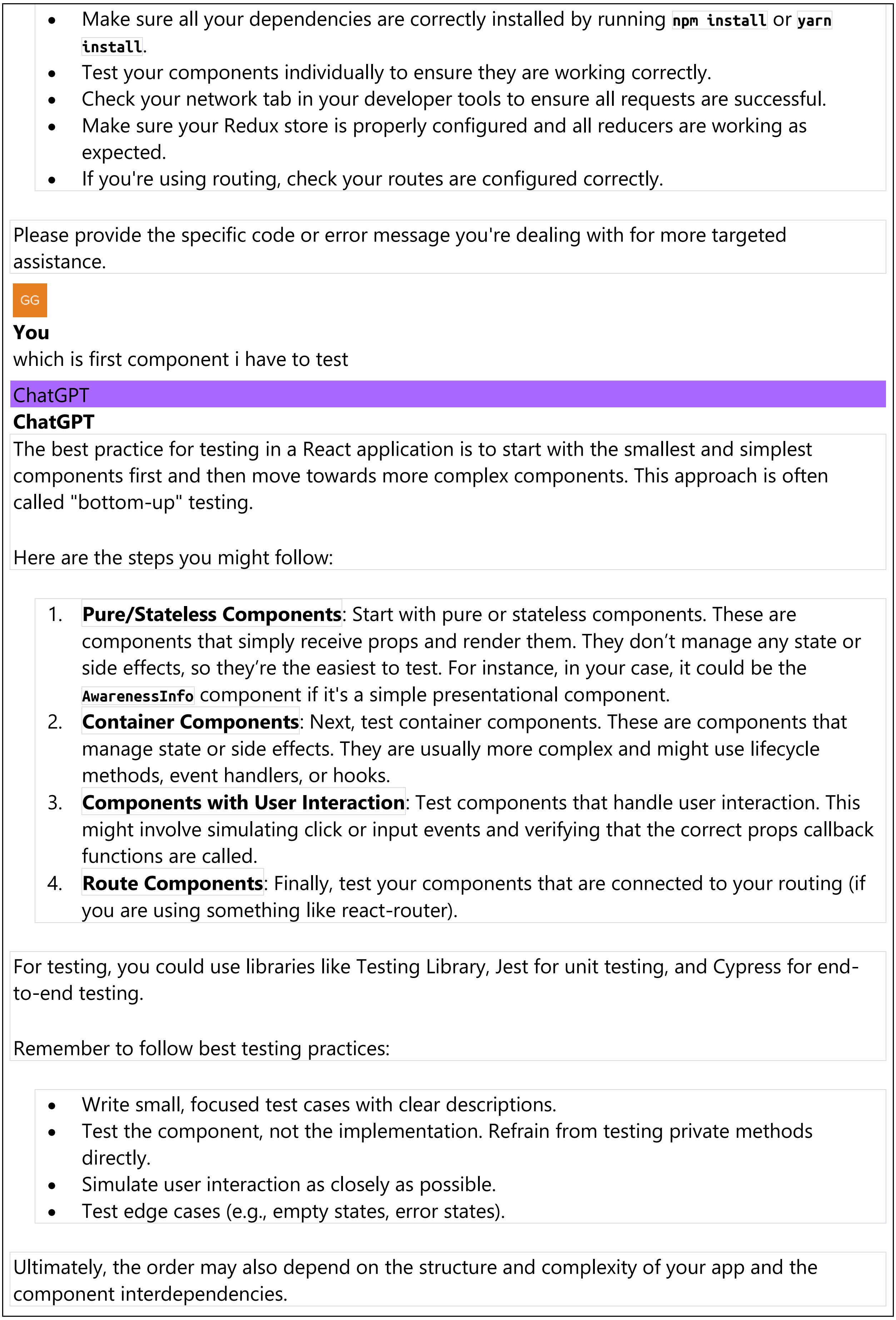
ChatGPT

There can be a number of reasons why your app might display a blank page:

1. **React component rendering error**: Your React component could have an error causing it to fail to render.
2. **JavaScript errors**: There may be an error in your JavaScript code which is causing the app to fail to load.
3. **Network issues**: If your app relies on loading data from a server or a database, there may be a network issue preventing the data from being loaded.
4. **Missing dependencies**: If your app relies on certain dependencies which are not present or not correctly installed, this can cause the app to fail to load.
5. **Incorrect routing**: If you are using something like React Router and your routes are not set up correctly, this can cause a blank page to appear.
6. **Issues with the Redux store**: If your app uses Redux and there's an issue with the way the store has been set up, this could cause the app to not render correctly.

To diagnose the problem, you can:

- Check the JavaScript console in your web browser's developer tools for any error messages.



- Make sure all your dependencies are correctly installed by running `npm install` or `yarn install`.
- Test your components individually to ensure they are working correctly.
- Check your network tab in your developer tools to ensure all requests are successful.
- Make sure your Redux store is properly configured and all reducers are working as expected.
- If you're using routing, check your routes are configured correctly.

Please provide the specific code or error message you're dealing with for more targeted assistance.

GG

You

which is first component i have to test

ChatGPT

ChatGPT

The best practice for testing in a React application is to start with the smallest and simplest components first and then move towards more complex components. This approach is often called "bottom-up" testing.

Here are the steps you might follow:

1. **Pure/Stateless Components**: Start with pure or stateless components. These are components that simply receive props and render them. They don't manage any state or side effects, so they're the easiest to test. For instance, in your case, it could be the `AwarenessInfo` component if it's a simple presentational component.
2. **Container Components**: Next, test container components. These are components that manage state or side effects. They are usually more complex and might use lifecycle methods, event handlers, or hooks.
3. **Components with User Interaction**: Test components that handle user interaction. This might involve simulating click or input events and verifying that the correct props callback functions are called.
4. **Route Components**: Finally, test your components that are connected to your routing (if you are using something like react-router).

For testing, you could use libraries like Testing Library, Jest for unit testing, and Cypress for end-to-end testing.

Remember to follow best testing practices:

- Write small, focused test cases with clear descriptions.
- Test the component, not the implementation. Refrain from testing private methods directly.
- Simulate user interaction as closely as possible.
- Test edge cases (e.g., empty states, error states).

Ultimately, the order may also depend on the structure and complexity of your app and the component interdependencies.

Figure 3.7- ChatGPT-4 chat prompt and result of Example 1.3.2.

After some corrections, I finished project. I display result (Figure 3.8): real time GPS position monitor of Alice and Bob.

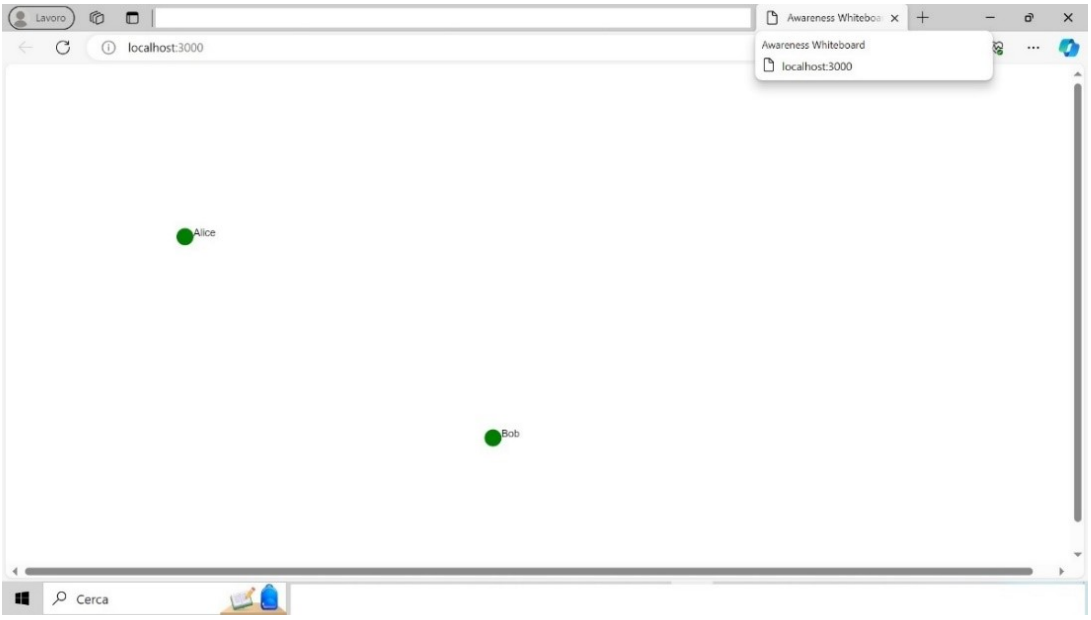


Figure 3.8- Web App interface: Alice and Bob GPS real time position.

It's difficult to evaluate exactly how much time I saved and the quality of the final result compared to a classic approach but this project increased my knowledge on potentiality of ChatGPT-4 and project tech stack arguments. For a comprehensive evaluation, see the article "Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors" [23].

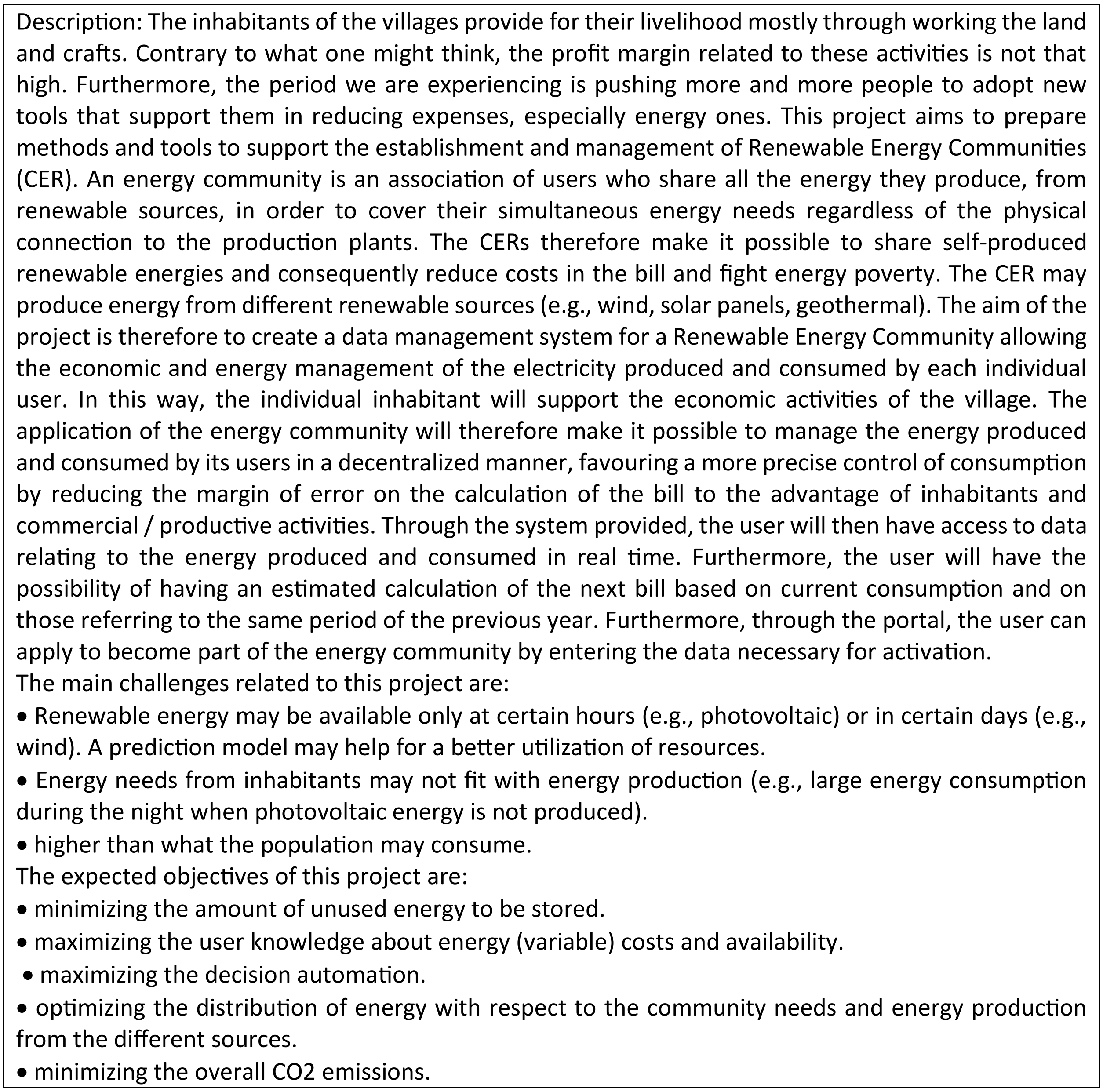
This integration between human expertise and ChatGPT-4 capability promises a future where software development is not just accelerated but also enhanced in quality.

3.3 - Case study 2: Software Architectures Project

After promising experience of previous case study, I experimented ChatGPT-4 robust capability to process, analyse, and generate source code can be exploited to augment data exploration, complex logic model creation, create friendly user interface and project documentation.

My software architectures project proposes a study of an existent project "Renewable Energy Communities" and offers an update "Renewable Energy Communities 2.0", finalized to a produce a simulation of an alternative end-to-end solution.

The existent project is one of possible solution of 2022-2023 software architectures course project track. (Figure 3.9)



Description: The inhabitants of the villages provide for their livelihood mostly through working the land and crafts. Contrary to what one might think, the profit margin related to these activities is not that high. Furthermore, the period we are experiencing is pushing more and more people to adopt new tools that support them in reducing expenses, especially energy ones. This project aims to prepare methods and tools to support the establishment and management of Renewable Energy Communities (CER). An energy community is an association of users who share all the energy they produce, from renewable sources, in order to cover their simultaneous energy needs regardless of the physical connection to the production plants. The CERs therefore make it possible to share self-produced renewable energies and consequently reduce costs in the bill and fight energy poverty. The CER may produce energy from different renewable sources (e.g., wind, solar panels, geothermal). The aim of the project is therefore to create a data management system for a Renewable Energy Community allowing the economic and energy management of the electricity produced and consumed by each individual user. In this way, the individual inhabitant will support the economic activities of the village. The application of the energy community will therefore make it possible to manage the energy produced and consumed by its users in a decentralized manner, favouring a more precise control of consumption by reducing the margin of error on the calculation of the bill to the advantage of inhabitants and commercial / productive activities. Through the system provided, the user will then have access to data relating to the energy produced and consumed in real time. Furthermore, the user will have the possibility of having an estimated calculation of the next bill based on current consumption and on those referring to the same period of the previous year. Furthermore, through the portal, the user can apply to become part of the energy community by entering the data necessary for activation.

The main challenges related to this project are:

- Renewable energy may be available only at certain hours (e.g., photovoltaic) or in certain days (e.g., wind). A prediction model may help for a better utilization of resources.
- Energy needs from inhabitants may not fit with energy production (e.g., large energy consumption during the night when photovoltaic energy is not produced).
- higher than what the population may consume.

The expected objectives of this project are:

- minimizing the amount of unused energy to be stored.
- maximizing the user knowledge about energy (variable) costs and availability.
- maximizing the decision automation.
- optimizing the distribution of energy with respect to the community needs and energy production from the different sources.
- minimizing the overall CO2 emissions.

Figure 3.9 - 2022-2023 software architectures course project track.

The original project consists of a system composed of 2 subsystems. The prediction system and the physical system. The prediction subsystem is a model used for prediction of energy production and consumption of the households based on historical data. It also includes a model for predicting the surplus of the energy. If the predicted surplus is negative, meaning the energy production predicted for the household for one month is less than their consumption at the same period, a connection is established to the grid for buying energy. In case the surplus is positive, meaning that the household's energy production within a certain time is more than their own needs, another connection is made with the grid for selling the energy. The energy production predictor uses the data of weather conditions in Spain for making predictions of energy produced by solar panels. Having the information of the number of solar panels per household, their energy production of the households is predicted. The energy consumption predictor uses features like size of the household as inputs. The system also predicts the bill to pay /receipt of money made for the users. The physical subsystem receives the actual data from the households every month. Every household contains a smart meter that sends consumption data of the household to this subsystem. Additionally, the solar panels installed in the houses also send data on the actual energy production of the house periodically. This subsystem provides patterns of energy production and consumption of the households.

My project, starting from exploration of data collection, LSTM (Long Short-Term Memory networks) deep learning modelling, predictions and visualization of original project, focus on prediction system module and extends prediction to solar and wind energy production and consumption, based on real weather data, substitute LSTM prediction model with Transformer prediction model and provides an intuitive interface for users to visualize and compare predicted energy production with predicted energy consumption. To explore most of ChatGPT-4 potentiality, I extensively used it for three types of information: technical information, graph generator for project documentation and source code useful for web app implementation.

Due to the potential for inaccurate information generated by ChatGPT-4, for each type of information, I illustrated which rigorous validation process I used to ensure reliability and accuracy of each type of information.

1. To validate **advanced technical information**, I followed below steps:
    a. I used multiple ChatGPT-4 chat to obtain advanced technical information.
    b. To ensure reliability and accuracy of advanced technical information generated from ChatGPT-4 I asked it to provide me all kind of information sources about technical information generated, URLs or bibliography references by the chat.
    c. I founded these references on Internet, I verified them and listed in paragraph References.
    d. Together other information founded on trusted websites and my skills and knowledge I ensured reliability and accuracy and where necessary I modified it before embedding it to the documentation.

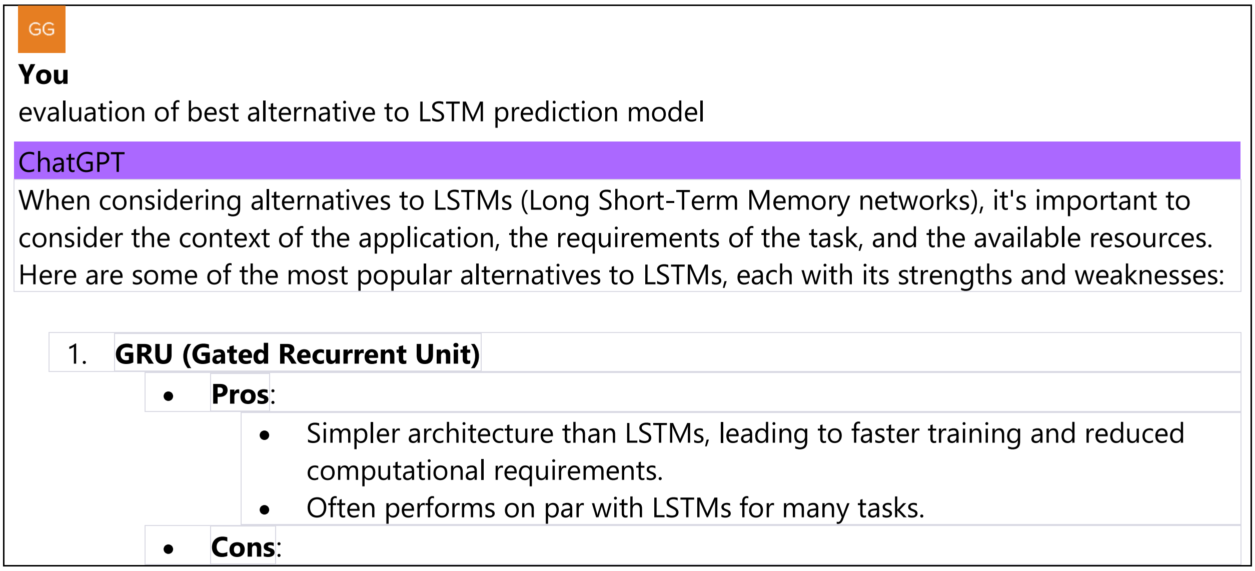
2. To validate **graph used to realize UML static and dynamic architecture view**, I followed below steps:

   a. To obtain required complex diagram I started from a chat in which I asked to ChatGPT-4 to supply me a source code in python and library pvgraphviz.
   b. After downloaded the source code, I edited python code in my development framework.
   c. I modified source code and executed more time until reaching the wanted result, adding the title before embedding it into the documentation.

3. To validate **complex source code**, I followed below steps:
   a. To obtain required complex source code from ChatGPT-4 I used multiple chat.
   b. To ensure its correctness and quality, I downloaded the source code directly into the development framework and I reviewed it using my skills and knowledge.
   c. I ensured that the code met the required standards and specifications by personally checking it within the context of my development environment. This include running tests, reviewing the code for errors or potential improvements.

Now I report one example of ChatGPT-4 chat for each step that helped me to realize web application and project documentation.

In the example 2.1.1 (case study 2, information type 1, example 1), ChatGPT-4 helped me (Figure 3.10) to substitute LSTM with Transformer neural network architecture, so I asked to ChatGPT-4 advanced technical information about LSTM and other known neural artificial architecture useful as documentation support.



You
evaluation of best alternative to LSTM prediction model

ChatGPT
When considering alternatives to LSTMs (Long Short-Term Memory networks), it's important to consider the context of the application, the requirements of the task, and the available resources. Here are some of the most popular alternatives to LSTMs, each with its strengths and weaknesses:

1. **GRU (Gated Recurrent Unit)**
   - **Pros**:
     - Simpler architecture than LSTMs, leading to faster training and reduced computational requirements.
     - Often performs on par with LSTMs for many tasks.
   - **Cons**:

    - May not capture long-term dependencies as effectively as LSTMs in some cases.

2. **Bidirectional LSTMs/GRUs**
    - **Pros**:
        - Processes sequences in both forward and backward directions, allowing the model to have information from both past and future states.
        - Often outperforms unidirectional LSTMs in tasks where future context is useful, such as sequence labeling.
    - **Cons**:
        - Increases the number of parameters, potentially leading to longer training times and a higher risk of overfitting.
3. **Transformer-based Models (e.g., BERT, GPT, T5)**
    - **Pros**:
        - Self-attention mechanism allows for capturing long-range dependencies effectively.
        - Achieves state-of-the-art results on many NLP tasks.
        - Pre-trained models available for transfer learning.
    - **Cons**:
        - Can be computationally intensive, especially for long sequences.
        - Requires more memory and computational resources compared to LSTMs and GRUs.
4. **1D Convolutional Neural Networks (CNNs)**
    - **Pros**:
        - Can capture local patterns in sequences effectively.
        - Often faster to train than recurrent models.
        - Suitable for tasks like sequence classification.
    - **Cons**:
        - May not capture long-term dependencies as effectively as recurrent models.
5. **Simple RNN**
    - **Pros**:
        - Simpler architecture than LSTMs and GRUs.
    - **Cons**:
        - Susceptible to the vanishing gradient problem, making them less effective at capturing long-term dependencies.
        - Often outperformed by LSTMs and GRUs.
6. **Attention Mechanisms (without full Transformers)**
    - **Pros**:
        - Allows the model to focus on specific parts of the sequence when producing an output.
        - Can enhance the performance of LSTMs and GRUs.
    - **Cons**:
        - Additional complexity over the base RNN models.
7. **TCN (Temporal Convolutional Network)**
    - **Pros**:
        - Uses dilated convolutions to capture long-range dependencies.
        - Has shown competitive performance with RNNs for sequence modeling.
    - **Cons**:

- Less commonly used than RNNs and Transformers, so fewer resources and pre-trained models are available.

8. **Hybrid Models**
   - **Pros**:
     - Combines the strengths of different architectures (e.g., CNNs with RNNs).
     - Can be tailored to specific tasks.
   - **Cons**:
     - Increased complexity and potential risk of overfitting.

Conclusion: The best alternative to LSTMs largely depends on the specific problem at hand. For instance, for sequence-to-sequence translation, attention mechanisms have been particularly effective. For tasks with very long sequences, TCNs might be more suitable. For many NLP tasks, Transformer-based models like BERT and GPT have set new benchmarks. However, in scenarios with limited computational resources or when working with small datasets, simpler models like GRUs might be more appropriate. As always, empirical testing on the specific task is the best way to determine the most suitable model.

GG

You

give me web article references for each model.

ChatGPT

ChatGPT

I did a quick search for more information and here's what I found.

For an in-depth understanding of the Transformer model, I recommend checking out an article on Machine Learning Mastery. This article provides a comprehensive overview of the Transformer model, including its encoder-decoder structure, how it implements self-attention, and compares it to recurrent and convolutional layers. It also covers the intricacies of the Transformer architecture, explaining each component's function in the model. This resource is particularly useful for those looking to deepen their understanding of Transformer models in machine learning.

The Transformer Model - MachineLearningMastery.com

For other neural network models, I would need to conduct further searches to provide appropriate references.

GG

You

conduct further searches

ChatGPT

ChatGPT

For an in-depth understanding of Temporal Convolutional Networks (TCNs), I found a useful article on Nature's Scientific Reports. This article discusses a seasonal-trend decomposition-based 2-dimensional temporal convolution dense network (2DTCDN) for improving long-term multivariate time series forecasting. It delves into the challenges and methodologies of decomposing time series data, highlighting how different components like trends, seasonality, and residuals are addressed. The article also outlines the problem statement for multivariate time series forecasting and explains the methodology of the STL-2DTCDN model. This resource is particularly insightful for those interested in advanced techniques in time series analysis and forecasting using TCNs.

Improving long-term multivariate time series forecasting with a seasonal-trend decomposition-based 2-dimensional temporal convolution dense network | Scientific Reports (nature.com)

I will continue to search for relevant articles for the other neural network models.
For Echo State Networks (ESNs), there are a couple of informative articles that delve into their design and applications. The first one, titled "A Review of Designs and Applications of Echo State Networks," provides a comprehensive overview of ESNs, including their basic models, designs of reservoirs, and various structural modifications. It discusses how ESNs have evolved since their inception, highlighting their efficiency in handling chaotic time series and complex dynamical systems.

[2012.02974] A Review of Designs and Applications of Echo State Networks (arxiv.org)

Another insightful article, "Echo State Networks with Self-Normalizing Activations on the Hyper-Sphere," explores a specific model of ESNs characterized by a self-normalizing activation function. This model exhibits nonlinear behaviors while maintaining superior memory capacity, a feature crucial for many neural network applications. The article also addresses the universal function approximation property of ESNs, demonstrating their broad applicability in various computational tasks.

Echo State Networks with Self-Normalizing Activations on the Hyper-Sphere | Scientific Reports (nature.com)

These articles provide a deep dive into the world of Echo State Networks, exploring both their theoretical foundations and practical applications.

GG

You

continue

ChatGPT

ChatGPT

For a comprehensive understanding of Bidirectional Recurrent Neural Networks (BRNNs), the Paperspace Blog offers a detailed guide. This article explains the architecture of BRNNs, highlighting their unique feature of processing input data in both forward and backward directions. It covers the mathematical foundations of how BRNNs function, including the update mechanisms for forward and backward states. The guide also discusses various applications of BRNNs, such as handwriting recognition, speech recognition, and natural language processing. Additionally, it provides a practical example of using BRNNs for sentiment analysis, demonstrating how to build, train, and test a Bidirectional RNN model using the Keras library.

A Guide to Bidirectional RNNs With Keras | Paperspace Blog

This resource is particularly valuable for those looking to gain a deeper understanding of BRNNs and their practical applications in machine learning and data processing tasks.

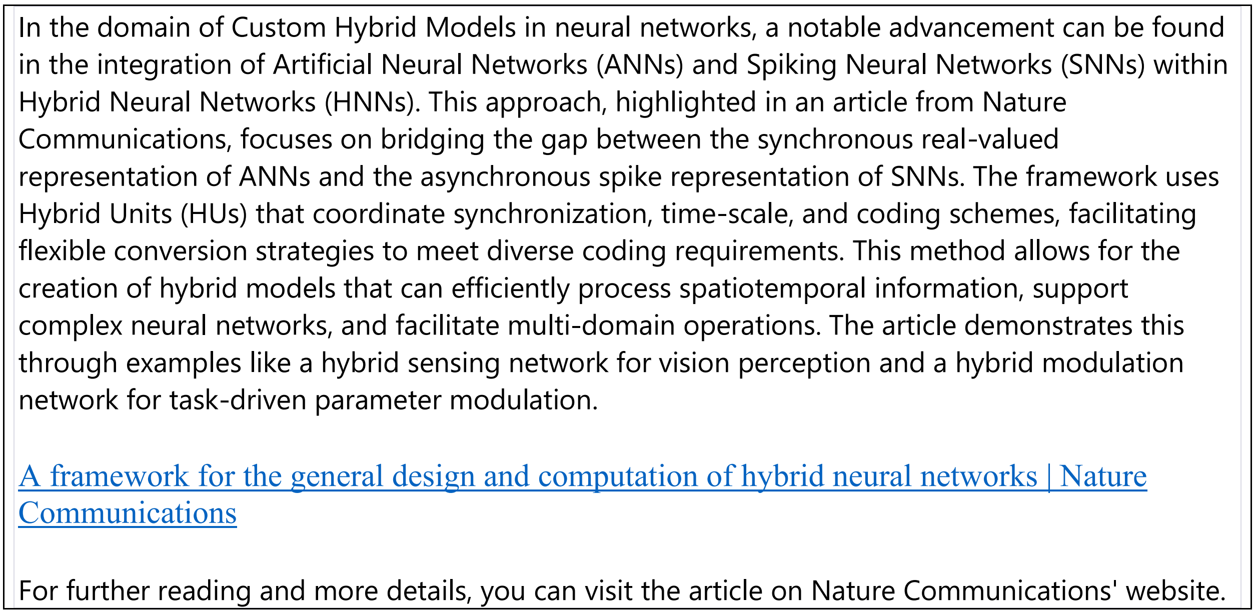
GG

You

continue

ChatGPT

ChatGPT



In the domain of Custom Hybrid Models in neural networks, a notable advancement can be found in the integration of Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) within Hybrid Neural Networks (HNNs). This approach, highlighted in an article from Nature Communications, focuses on bridging the gap between the synchronous real-valued representation of ANNs and the asynchronous spike representation of SNNs. The framework uses Hybrid Units (HUs) that coordinate synchronization, time-scale, and coding schemes, facilitating flexible conversion strategies to meet diverse coding requirements. This method allows for the creation of hybrid models that can efficiently process spatiotemporal information, support complex neural networks, and facilitate multi-domain operations. The article demonstrates this through examples like a hybrid sensing network for vision perception and a hybrid modulation network for task-driven parameter modulation.

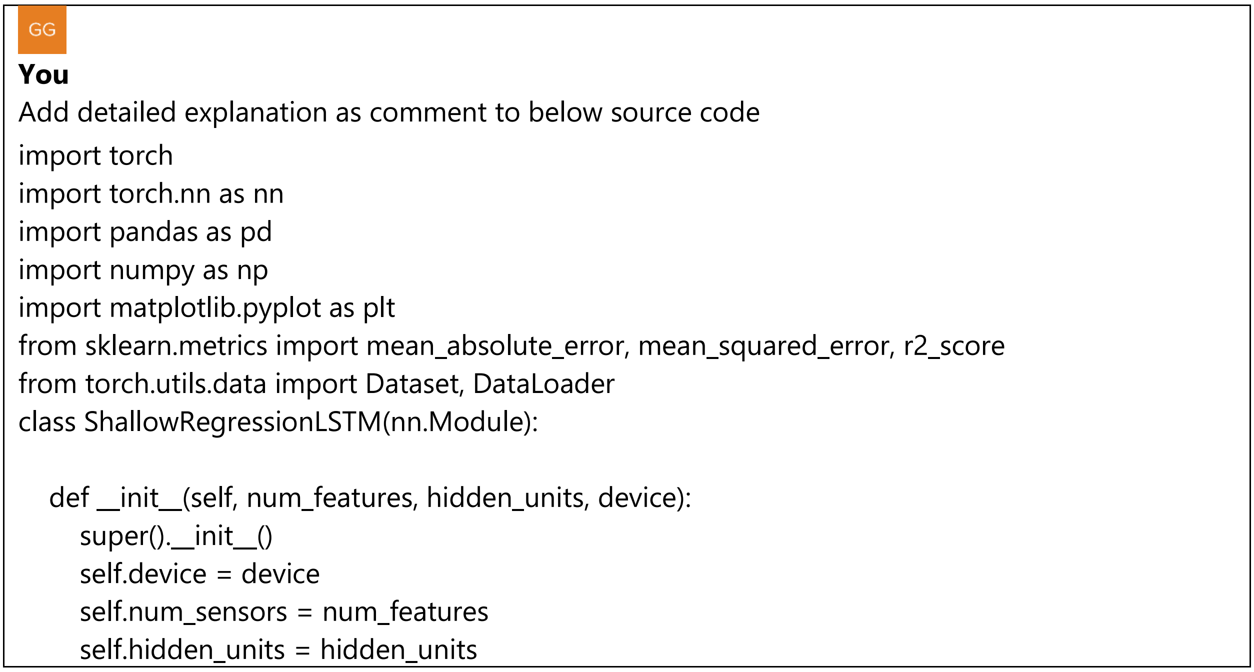
A framework for the general design and computation of hybrid neural networks | Nature Communications

For further reading and more details, you can visit the article on Nature Communications' website.

Figure 3.10- ChatGPT-4 chat prompt and result of Example 2.1.1.

As specified in step 1.c and 1.d, I found these references on Internet, I verified them and together other information founded on trusted websites and my skills and knowledge I ensured reliability and accuracy before embedding it to the documentation.

In the example 2.1.2, ChatGPT-4 helped me (Figure 3.11) to add source code comment, enhancing software quality. Well-commented source code is easier to maintain, improve the readability, help developers to communicate their intent to others, provide hints for debugging or note potential improvements, is useful for educational.



You

Add detailed explanation as comment to below source code

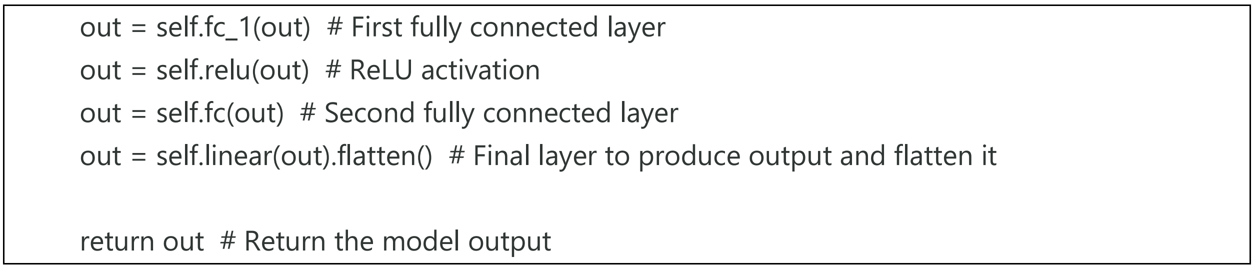
```
import torch
import torch.nn as nn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from torch.utils.data import Dataset, DataLoader
class ShallowRegressionLSTM(nn.Module):

    def __init__(self, num_features, hidden_units, device):
        super().__init__()
        self.device = device
        self.num_sensors = num_features
        self.hidden_units = hidden_units
        self.num_layers = 1

        self.lstm = nn.LSTM(
            input_size=self.num_sensors,
            hidden_size=self.hidden_units,
            batch_first=True,
            dropout= 0,
            num_layers=self.num_layers
        )
        self.fc_1 =  nn.Linear(self.hidden_units, 32)
        self.fc = nn.Linear(32, 16)
        self.relu = nn.ReLU()
        self.linear = nn.Linear(in_features=16, out_features=1)

    def forward(self, x):
        batch_size = x.shape[0]
        h0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)
        c0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)

        _, (hn, _) = self.lstm(x, (h0, c0))
        out = self.relu(hn[0])
        out = self.fc_1(out)
        out = self.relu(out) #relu
        out = self.fc(out) #Fin
        out = self.linear(out).flatten()
        return out
```

ChatGPT

ChatGPT

Certainly! Below is a detailed explanation of each part of the provided source code, which implements a shallow LSTM regression model in PyTorch.

```
# Importing necessary libraries
import torch
import torch.nn as nn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from torch.utils.data import Dataset, DataLoader

# Define the ShallowRegressionLSTM class that inherits from PyTorch's nn.Module
class ShallowRegressionLSTM(nn.Module):
    """
    A shallow LSTM model for regression tasks. Inherits from PyTorch's nn.Module.
    Designed to predict a continuous value from a sequence of input features.
    """
    def __init__(self, num_features, hidden_units, device):
        """
        Initializes the model.
        :param num_features: Number of input features.
        :param hidden_units: Number of hidden units in the LSTM layer.
        :param device: The device (CPU or GPU) the model should run on.
        """
        super().__init__()  # Initialize the parent class (nn.Module)
        self.device = device  # Device configuration (CPU or GPU)
        self.num_sensors = num_features  # Number of input features
        self.hidden_units = hidden_units  # Number of hidden units in the LSTM
        self.num_layers = 1  # Number of LSTM layers

        # Define the LSTM layer with specified input size, hidden size, and number of layers
        self.lstm = nn.LSTM(
            input_size=self.num_sensors,
            hidden_size=self.hidden_units,
            batch_first=True,  # Specifies that the input tensor's first dimension is the batch size
            dropout=0,  # Dropout rate (not using dropout here)
            num_layers=self.num_layers  # Number of LSTM layers
        )

        # Fully connected layers and ReLU activation to map LSTM outputs to desired output size
        self.fc_1 = nn.Linear(self.hidden_units, 32)  # First fully connected layer
        self.fc = nn.Linear(32, 16)  # Second fully connected layer
        self.relu = nn.ReLU()  # ReLU activation function
        self.linear = nn.Linear(in_features=16, out_features=1)  # Final layer to produce output

    def forward(self, x):
        """
        Defines the forward pass of the model.
        :param x: Input tensor containing features.
        """
        batch_size = x.shape[0]  # Determine batch size

        # Initialize hidden and cell states to zeros
        h0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)
        c0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)

        # Pass input through LSTM layer
        _, (hn, _) = self.lstm(x, (h0, c0))  # hn contains the final hidden state of the LSTM

        # Sequentially pass the output through ReLU and fully connected layers
        out = self.relu(hn[0])  # Apply ReLU activation to the last hidden state
```

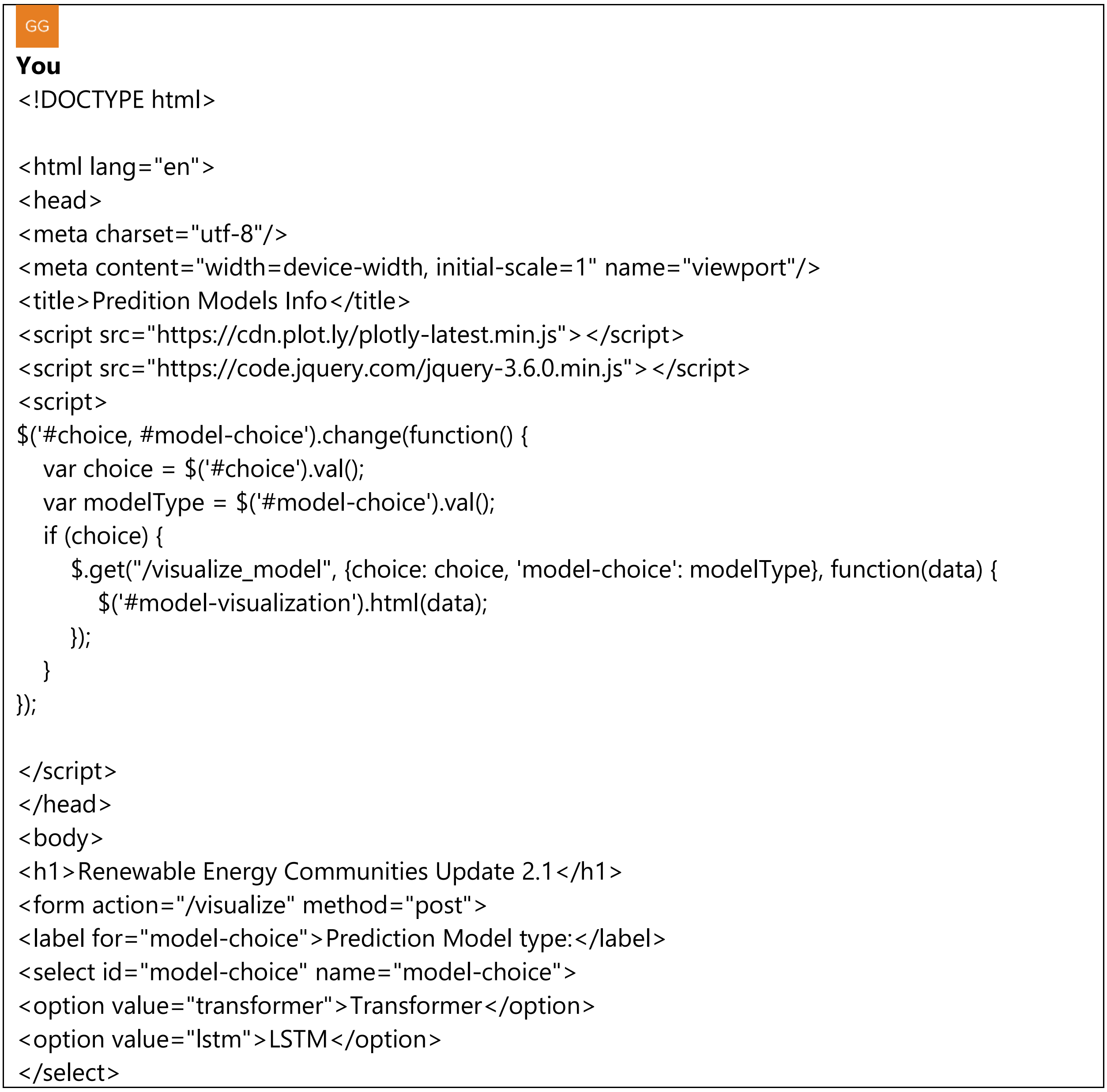


```
        out = self.fc_1(out)  # First fully connected layer
        out = self.relu(out)  # ReLU activation
        out = self.fc(out)  # Second fully connected layer
        out = self.linear(out).flatten()  # Final layer to produce output and flatten it

        return out  # Return the model output
```

Figure 3.11- ChatGPT-4 chat prompt and result of Example 2.1.2.

As specified in step 1.c and 1.d, I verified them using my skills and knowledge, I ensured reliability and accuracy before embedding it into source code.

In the example 2.1.3, ChatGPT-4 helped me (Figure 3.12) to produce user manual of Web Interface starting from interface source code.



You

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<title>Predition Models Info</title>
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
$('#choice, #model-choice').change(function() {
   var choice = $('#choice').val();
   var modelType = $('#model-choice').val();
   if (choice) {
      $.get("/visualize_model", {choice: choice, 'model-choice': modelType}, function(data) {
         $('#model-visualization').html(data);
      });
   }
});

</script>
</head>
<body>
<h1>Renewable Energy Communities Update 2.1</h1>
<form action="/visualize" method="post">
<label for="model-choice">Prediction Model type:</label>
<select id="model-choice" name="model-choice">
<option value="transformer">Transformer</option>
<option value="lstm">LSTM</option>
</select>
```

```
<label for="choice">Energy type:</label>
<select id="choice" name="choice">
<option value="">Choose Energy Type</option>
<option value="solar">Solar</option>
<option value="wind">Wind</option>
<option value="both">Both</option>
</select>
<input type="submit" value="Visualize"/>
</form>
<div id="model-visualization"></div>
<script>
$(document).ready(function() {
    // Initially disable the "Visualize" button
    $('input[type="submit"]').prop('disabled', true);

    // Enable the "Visualize" button only if a valid energy type is selected
    $('#choice').change(function() {
        if ($(this).val()) {
            $('input[type="submit"]').prop('disabled', false);
        } else {
            $('input[type="submit"]').prop('disabled', true);
        }
    });
});
</script>
<script>
$('#choice, #model-choice').change(function() {
    var choice = $('#choice').val();
    var modelType = $('#model-choice').val();
    if (choice) {
        $.get("/visualize_model", {choice: choice, 'model-choice': modelType}, function(data) {
            $('#model-visualization').html(data);
        });
    }
});

</script>
<script>
$(document).ready(function() {
    // Function to check the conditions for enabling the "Visualize" button
    function checkConditions() {
        var modelChoice = $('#model-choice').val();
        var energyChoice = $('#choice').val();

        if (modelChoice === 'transformer' && energyChoice && energyChoice !== modelChoice) {
            $('input[type="submit"]').prop('disabled', false);
        } else {
            $('input[type="submit"]').prop('disabled', true);
        }
    }
    // Initially disable the "Visualize" button
    $('input[type="submit"]').prop('disabled', true);

    // Attach event listeners to both dropdowns
    $('#model-choice, #choice').change(function() {
        checkConditions();
    });
});
</script>
</body>
</html>
```

ChatGPT

ChatGPT

This HTML document outlines an interactive web page focused on visualizing prediction models for renewable energy communities. Let's break down its components and functionality:

Head Section

- **Character Set and Viewport**: The document specifies a character set of UTF-8 and sets the viewport to make sure the page is mobile-responsive.
- **Title**: The title of the page is "Prediction Models Info".
- **External Scripts**: Includes Plotly.js for data visualization, and jQuery for simplifying DOM manipulation and AJAX calls.

Body Section

Main Content

- **Heading**: Displays a heading "Renewable Energy Communities Update 2.1".
- **Form**: Allows the user to select a prediction model type (Transformer or LSTM) and an energy type (Solar, Wind, or Both) before submitting to visualize the model. The form action is "/visualize" which presumably triggers server-side processing to visualize the selected model and energy type combination.

JavaScript Functionality

1. **Dropdown Change Event**: The script listens for changes in the dropdown selections. Upon change, it sends a GET request to "/visualize_model" with the selected values of **choice** (energy type) and **model-choice** (model type) as parameters. The server response is then displayed within the **#model-visualization** div. This functionality allows for dynamic updating of the visualization without needing to reload the page or submit the form.
2. **Visualize Button Enablement**: Initially, the "Visualize" button is disabled. It is enabled only if a valid energy type is selected. This prevents form submission without a meaningful selection.
3. **Conditions for Enabling the "Visualize" Button**: An additional script checks specific conditions before enabling the "Visualize" button. It requires that the model choice is 'transformer', an energy choice is made, and the energy choice is not equal to the model choice. However, this logic seems to have a flaw since the energy choice (**solar**, **wind**, **both**) can never be equal to the model choice (**transformer**, **lstm**), making the last condition redundant or possibly incorrect.

GG

You

give me functionality as user manual like

ChatGPT

ChatGPT

The provided HTML document is designed for a web application that visualizes prediction models for renewable energy communities, focusing on solar and wind energy. Below is a user manual-style breakdown of its functionality:

Renewable Energy Communities Update 2.1

Overview:

This web application allows users to visualize prediction models related to renewable energy usage and forecasting, specifically focusing on solar and wind energy. It provides an interactive interface where users can select the type of energy and the prediction model they are interested in for visualization.

How to Use:

1. **Select Prediction Model Type:**
    - Navigate to the "Prediction Model type" dropdown menu.
    - Choose between "Transformer" and "LSTM" as the model type for predictions. These models offer different approaches to analyzing and forecasting energy data.
2. **Choose Energy Type:**
    - Go to the "Energy type" dropdown menu located next to the model type selector.
    - You have three options to choose from: "Solar", "Wind", or "Both". This choice determines the type of energy data that will be visualized using the selected prediction model.
3. **Visualization:**
    - After selecting both the prediction model type and the energy type, the "Visualize" button will become enabled. Initially, this button is disabled to prevent invalid requests.
    - Click on the "Visualize" button to proceed with the visualization. The application will then generate and display a visual representation of the prediction model based on your selections.

Additional Features:

- **Dynamic Visualization Update:**
    - The application is designed to update the visualization dynamically based on your selections. When you change either the model type or the energy type, the visualization will automatically update to reflect your new choices without needing to click the "Visualize" button again.
- **Form Validation:**
    - To ensure that a valid visualization request is made, the "Visualize" button remains disabled until a valid set of choices is made. This behavior is controlled through JavaScript to enhance user experience by preventing unnecessary or invalid requests.
    - Specifically, the "Visualize" button will only be enabled if:
        - A valid energy type is selected, and
        - For some configurations, additional conditions must be met based on the model type and energy type selected. These are implemented to ensure compatibility and relevance of the visualization.

Summary:
This user manual provides a guide on how to interact with the Renewable Energy Communities Update 2.1 web application. By following the steps outlined, users can effectively explore various renewable energy prediction models and visualize the potential outcomes for solar and wind energy projects.

Figure 3.12- ChatGPT-4 chat prompt and result of Example 2.1.3.

As specified in step 1.c and 1.d, I verified reliability and accuracy of ChatGPT-4 result before embedding it to the documentation.

In the example 2.2.1, ChatGPT-4 helped me (Figure 3.13) to design graph used to realize UML static and dynamic architecture view using python pygraphviz chosen thanks to a previous chat, so I asked to ChatGPT-4 how to produce one of UML graph.

GG

You

give me component diagram

ChatGPT

ChatGPT

A component diagram is a type of UML diagram that visualizes the organization and wiring of physical or logical components in a system. These components can be software modules, middleware, libraries, etc., and the component diagram helps in visualizing how they interact and depend on one another.

To create a component diagram for the provided files, we need to identify the primary components and their dependencies or interactions. From our review of the provided files, we can identify the following components:

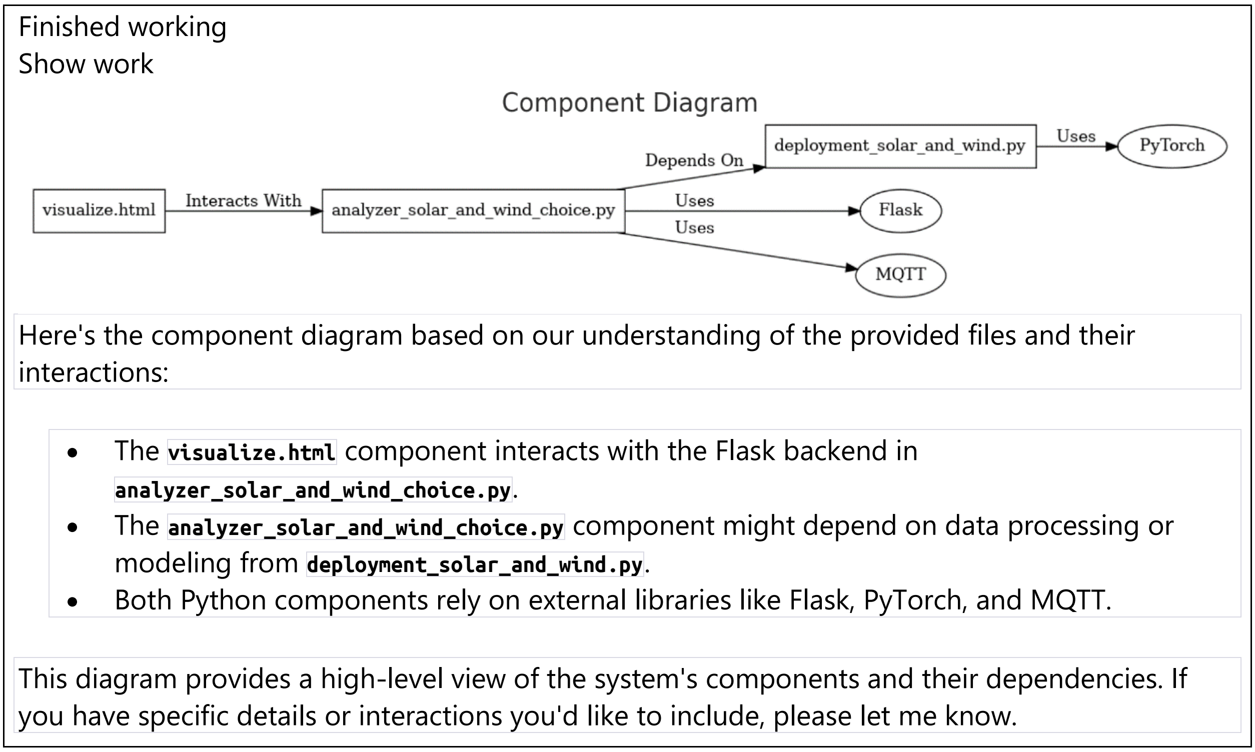
1. `visualize.html`: A web front-end component.
2. `analyzer_solar_and_wind_choice.py`: Contains the backend logic for data analysis and visualization.
3. `deployment_solar_and_wind.py`: Handles data processing and interactions with machine learning models.

There are also external dependencies and libraries such as Flask, PyTorch, MQTT, etc., which can be considered as components.

Let's create a hypothetical component diagram based on the above understanding:

1. The `visualize.html` interacts with the Flask backend in `analyzer_solar_and_wind_choice.py`.
2. `analyzer_solar_and_wind_choice.py` might depend on data processing or modeling from `deployment_solar_and_wind.py`.
3. Both Python scripts rely on external libraries like Flask, PyTorch, etc.

Let's visualize this in a component diagram.



Finished working
Show work

Here's the component diagram based on our understanding of the provided files and their interactions:

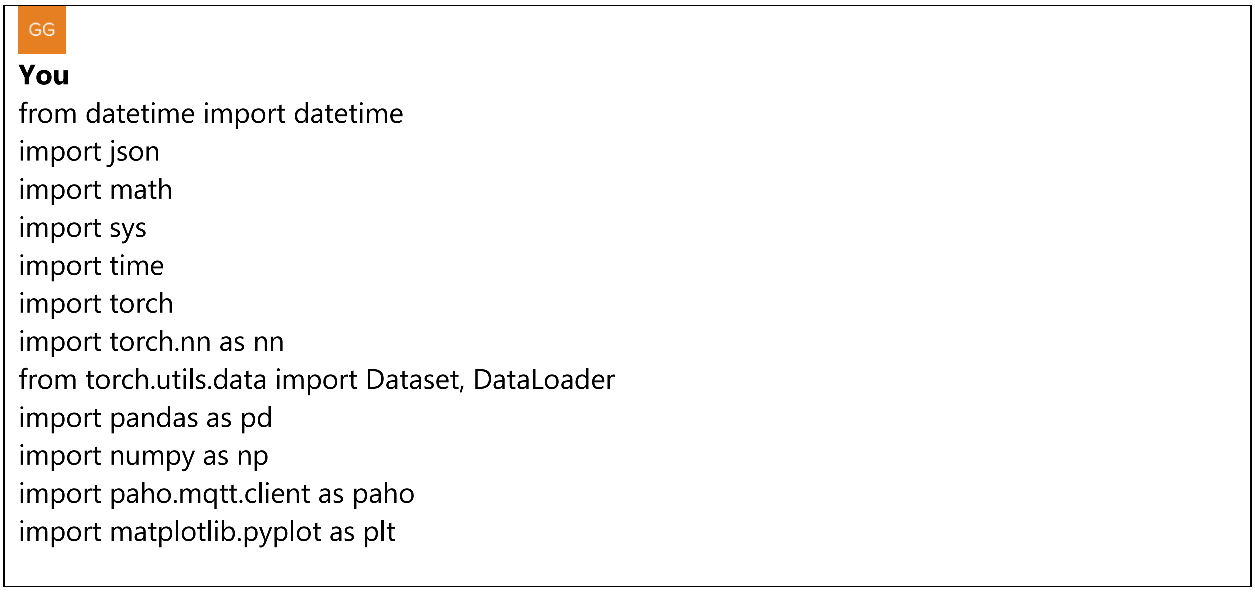
- The `visualize.html` component interacts with the Flask backend in `analyzer_solar_and_wind_choice.py`.
- The `analyzer_solar_and_wind_choice.py` component might depend on data processing or modeling from `deployment_solar_and_wind.py`.
- Both Python components rely on external libraries like Flask, PyTorch, and MQTT.

This diagram provides a high-level view of the system's components and their dependencies. If you have specific details or interactions you'd like to include, please let me know.

Figure 3.13- ChatGPT-4 chat prompt and result of Example 2.2.1.

As specified in step 2.b and 2.c, after downloaded the source code, I edited python code in my development framework, I modified source code and executed more time until reaching the wanted result, adding the title before embedding it into the documentation.

In the example 2.3.1, I ChatGPT-4 helped me (Figure 3.14) to write complex source code as substitute LSTM prediction procedures with Transformer model, so I asked to ChatGPT-4 how to proceed for the substitution.



You

```
from datetime import datetime
import json
import math
import sys
import time
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
import pandas as pd
import numpy as np
import paho.mqtt.client as paho
import matplotlib.pyplot as plt
```

```
class weatherDataset(Dataset):
    """
    The Dataset implements the Dataset Pytorch class, including itemgetter, len and intialization.
    Additionally, dates can be requested using the get_date class method.
    """

    def __init__(self, csv_file,sequence_len,device):
        self.device = device
        dataset = pd.read_csv(csv_file, header=0)
        self.time = dataset["time"]
        dataset = dataset.drop(columns="time")
        dataset = dataset.values
        self.sequence_length = sequence_len
        dataset_tens = torch.tensor(dataset, dtype=torch.float)
        self.norm_data = dataset_tens[:,2:].clone().detach()
        self.solar = dataset_tens[:,0].clone().detach()
        self.wind = dataset_tens[:,1].clone().detach()

    def __len__(self):
        return len(self.norm_data)

    def get_date(self,row):
        return self.time[row]

    def __getitem__(self, i):
        if i >= self.sequence_length - 1:
            i_start = i - self.sequence_length + 1
            x = self.norm_data[i_start:(i + 1)]
        else:
            padding = self.norm_data[0].repeat(self.sequence_length - i - 1, 1)
            x = self.norm_data[0:(i + 1)]
            x = torch.cat((padding, x), 0)

        predictionSolar = self.solar[i]
        predictionWind = self.wind[i]
        sample = [x.to(self.device),predictionSolar.to(self.device),predictionWind.to(self.device)]
        return sample

class ShallowRegressionLSTM(nn.Module):
    """
    The ShallowRegressionLSTM implements the nn.Module class from Pytorch.
    Number of features are the amount of input features for the LSTM, Hidden Units describes the
    size of the Networks layers and the device can be changed between CPU and GPU (when Cuda
is available)
    This LSTM is a Shallow LSTM implementing only a single Network (num_layers=1)
    """

    def __init__(self, num_features, hidden_units, device):
        """Initialize LSTM with number of features, hidden units and the selcted device"""
        super().__init__()
        self.device = device
        self.num_sensors = num_features  # this is the number of features
        self.hidden_units = hidden_units
        self.num_layers = 1

        self.lstm = nn.LSTM(
            input_size=self.num_sensors,
            hidden_size=self.hidden_units,
            batch_first=True,
            dropout= 0.5,
            num_layers=self.num_layers
        )
        self.fc_1 =  nn.Linear(self.hidden_units, 32) #fully connected 1
        self.fc = nn.Linear(32, 16) #fully connected last layer
        self.relu = nn.ReLU()
        self.linear = nn.Linear(in_features=16, out_features=1)

    def forward(self, x):
        """push the values through the system without performing a backpropagation"""

        batch_size = x.shape[0]
        h0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)
        c0 = torch.zeros(self.num_layers, batch_size, self.hidden_units).requires_grad_().to(self.device)

        _, (hn, _) = self.lstm(x, (h0, c0))
        out = self.relu(hn[0])  # First dim of Hn is num_layers, which is set to 1 above.
        out = self.fc_1(out) #first Dense
        out = self.relu(out) #relu
        out = self.fc(out) #Fin
        out = self.linear(out).flatten()
        return out


def deploy_model(model:ShallowRegressionLSTM,row:int,s_l):
    """Selects the device and instantiates and loads the model for prediction from a selected file.
    Afterwards, a starting data is picked and the model can be deployed to do the prediction"""
    SEQUENCE_LEN = s_l
    device = torch.device("cpu")
    dataset_test = weatherDataset("Testing_Set.csv",SEQUENCE_LEN,device)
    date = dataset_test.get_date(row)
    datetime_object = datetime.strptime(date, '%Y-%m-%d %H:%M:%S%z')
    [x,y_sol,y_win] = dataset_test[row]
    model.eval()
    prediction = model(torch.unsqueeze(x,dim=0))
    return datetime_object.strftime("%m/%d/%Y,
%H:%M:%S"),prediction.detach().numpy()[0],y_sol.detach().numpy(),y_win.detach().numpy()

def paho_client():
    """Instatiate a PAHO client witht the methods used from the PAHO website, using localhost as
the host"""
```

```
    # CLIENT PAHO
    port = 1883
    username = 'mosquittoBroker'
    password = 'se4gd'
    client_id = f'Solar_Client'
    client = paho.Client(client_id)
    client.username_pw_set(username, password)
    if client.connect("localhost",1883,60)!=0:
        print("Could not connect to MQTT Broker!")
        sys.exit(-1)
    else:
        print("connected")
    client.publish("Agriculture/solar",1)
    return client

def energyProduction(i,model,amountOfSolarPanels=10):
    """Calculate the Produced Energy from the amount of solar panels and the model prediction"""
    SOLAR_PANEL_PRODUCTION = 400*amountOfSolarPanels
    [date,prediction_sol,y_sol,_y_win] = deploy_model(model,i,48)
    if(prediction_sol<0):
        prediction_sol=0
    return [date,prediction_sol*SOLAR_PANEL_PRODUCTION]

def energyConsumption(date:datetime,sizeInSqm=0,people=4):
    """Calculate the Consumed Energy from the amount of solar panels and the model prediction"""
    date = datetime.strptime(date,"%m/%d/%Y, %H:%M:%S")
    # https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8847370/
    AVERAGE_MONTHLY = 4000000/12
    if(people>1):
        AVERAGE_MONTHLY+=(AVERAGE_MONTHLY//2)*(people-1)
    def pdf(x):
        """Normalization"""
        mean = np.mean(x)
        std = np.std(x)
        y_out = (std * np.sqrt(2 * np.pi)) * np.exp( - (x - mean)**2 / (2 * std**2))
        return y_out
    start = (AVERAGE_MONTHLY//2)-(AVERAGE_MONTHLY//2*(people-1))
    end = AVERAGE_MONTHLY+(AVERAGE_MONTHLY//2*(people-1))
    x = np.arange(start, end, (end-start)/12)
    x = np.roll(x,6)
    x = pdf(x)
    month = date.month
    dailyEnergy = x[month-1]/30
    hourlyEnergy_x = (dailyEnergy/24)
    x = date.hour-1
    y = hourlyEnergy_x*(-math.sin(x*(math.pi/6))+1)
    return y
def main():
    # can be picked at random and is the starting point of the simulation using the testing set
    i=3938
    simulation_len = len(pd.read_csv("Testing_Set.csv", header=0))
    solar_model = ShallowRegressionLSTM(30,32,"cpu")
    stateDict_sol = torch.load("solar_model_bigger.pt",map_location=torch.device('cpu'))
    solar_model.load_state_dict(stateDict_sol)
    solar_model.eval()
    client = paho_client()
    arr_con = []
    arr_prod = []
    # Starting the true loop with the paaho client that continues as long as the city runs
    while(True):
        if(i>=simulation_len-1):
                i=0
        time.sleep(1)
        [dateForl,solar] = energyProduction(i,solar_model)
        consumption = energyConsumption(dateForl)
        print(f"Date:{dateForl}, Production: {solar}, Consumption:{consumption}")
        solar_publish = json.dumps({"production":solar,"timestamp":dateForl})
        consumption_publish = json.dumps({"consumption":solar,"timestamp":dateForl})
        arr_con.append(consumption)
        arr_prod.append(solar)
        client.publish("prediction/production",solar_publish)
        client.publish("prediction/consumption",consumption_publish)
        i=i+1

    # plotting details - commented for later reuse
    # x = np.arange(len(arr_con))
    # plt.plot(x,arr_con,label="consumption")
    # plt.plot(x,arr_prod,label="production")
    # plt.xlabel("time of the day in hours")
    # plt.ylabel("energy")
    # plt.legend()
    # plt.show()
    main()
```

ChatGPT

ChatGPT

This script is an energy simulation that uses two trained LSTM models (Long Short-Term Memory models) to predict solar and wind energy production based on a dataset. The predictions are then compared with energy consumption, and results are published to an MQTT broker (using Paho).

Here's a breakdown of the code:

1. **Dependencies**: The code starts by importing necessary libraries.
2. **weatherDataset Class**: This class inherits from PyTorch's `Dataset` class and is used to load and process a CSV dataset containing weather and energy data.
3. **ShallowRegressionLSTM Class**: This class defines an LSTM neural network architecture for regression. The LSTM is used to predict solar or wind energy based on input weather features.
4. **deploy_model function**: This function takes a trained model and a row index from the dataset and returns a prediction for solar or wind energy for that specific time.

5. **paho_client function**: Initializes a PAHO MQTT client for publishing data. The MQTT broker is expected to run on `localhost` with the given credentials.
6. **energyProduction function**: Uses the trained solar and wind models to predict energy production for a given row in the dataset.
7. **energyConsumption function**: Estimates the energy consumption based on the date and some other parameters. The consumption is calculated based on average monthly consumption values and a periodic function simulating daily variations.
8. **main function**: The main function performs the following tasks:
   - Loads the trained models for solar and wind prediction.
   - Connects to the MQTT broker.
   - Iteratively (in a `while` loop) predicts solar and wind production for each hour, calculates the expected consumption, and publishes these values to the MQTT broker.
   - The loop runs indefinitely, and when it reaches the end of the dataset, it wraps around and starts from the beginning again.

The simulation can be stopped by sending an interrupt signal (like Ctrl+C).

This script does not include any error handling mechanisms, and it assumes that the necessary files (`Testing_Set.csv`, `solar_model_bigger.pt`, and `wind_model_bigger.pt`) are available in the same directory. The MQTT broker is also assumed to be running and available at `localhost`. If any of these assumptions are not met, the script will throw an error and exit.

The main purpose of this script is to simulate energy production and consumption over time and to publish these values to an MQTT broker, presumably for monitoring or further analysis.

GG

You

use transformers instead of lstm

ChatGPT

ChatGPT

Transforming the architecture from LSTM to Transformers involves a considerable change in the model's design and training methodology. Transformers have become state-of-the-art for many sequence-based tasks due to their self-attention mechanisms, which allow them to consider the entire input sequence when making predictions.

Here's how you can replace the LSTM with a Transformer in the given code:

1. **Update the Neural Network Architecture**:

Replace the `ShallowRegressionLSTM` class with a `TransformerRegressor` class:

```
import torch.nn.functional as F

class TransformerRegressor(nn.Module):
    def __init__(self, num_features, hidden_units, device, num_heads=4, num_layers=2):
        super().__init__()
        self.device = device
        self.embedding = nn.Linear(num_features, hidden_units)
        self.transformer = nn.Transformer(
            d_model=hidden_units,
            nhead=num_heads,
            num_encoder_layers=num_layers,
            num_decoder_layers=num_layers
        )
        self.fc_1 = nn.Linear(hidden_units, 32)  # fully connected 1
        self.fc = nn.Linear(32, 16)  # fully connected last layer
        self.relu = nn.ReLU()
        self.linear = nn.Linear(in_features=16, out_features=1)

    def forward(self, x):
        # Embed the input
        embedded = F.relu(self.embedding(x))
        # As we're only interested in encoding and not decoding,
        # we use the transformer encoder only
        out = self.transformer.encoder(embedded)
        out = self.relu(out[:, -1, :])  # Take the output from the last position
        out = self.fc_1(out)  # first Dense
        out = self.relu(out)  # relu
        out = self.fc(out)  # Fin
        out = self.linear(out).flatten()
        return out
```

2. **Training the Transformer Model**:

If you've already trained the LSTM models and saved their weights, you'll need to train the Transformer models from scratch or fine-tune them using the previous dataset. This involves setting up a training loop, defining a loss function, and optimizing the model parameters.

3. **Using the Transformer in the main function**:

Replace the initialization of the LSTM models with the Transformer:

```
solar_model = TransformerRegressor(30, 32, "cpu")
stateDict_sol = torch.load("solar_model_transformer.pt", map_location=torch.device('cpu'))
solar_model.load_state_dict(stateDict_sol)
solar_model.eval()

wind_model = TransformerRegressor(30, 32, "cpu")
stateDict_win = torch.load("wind_model_transformer.pt", map_location=torch.device('cpu'))
wind_model.load_state_dict(stateDict_win)
wind_model.eval()
```

Remember to update the model filenames (`solar_model_transformer.pt` and `wind_model_transformer.pt`) accordingly.

4. **Potential Changes in Hyperparameters**:

When transitioning from LSTM to Transformer models, you might need to adjust hyperparameters such as the learning rate, batch size, or the number of training epochs to get optimal performance. This might require some experimentation.

5. **Model Size and Computational Requirements**:

Transformers are typically more computationally intensive than LSTMs, so you might notice an increase in training time and memory usage. However, Transformers often outperform LSTMs, especially on larger datasets.

Remember, the provided modifications give a basic Transformer architecture for regression. Depending on the complexity of your dataset and the specific problem you're addressing, you might need to further tune or adjust the architecture and training process for best results.

Figure 3.14- ChatGPT-4 chat prompt and result of Example 2.3.1.

As specified in step 3.b and 3.c, to ensure source code correctness and quality, I downloaded the source code directly into the development framework and I reviewed it using my skills and I ensured that the code met the required standards and specifications. This include reviewing the code for errors or potential improvements, create and run all the necessary tests. I display result reported in Figure 3.15 and Figure 3.16 in which user can visualize all data necessary to evaluate in real time all own energy values.

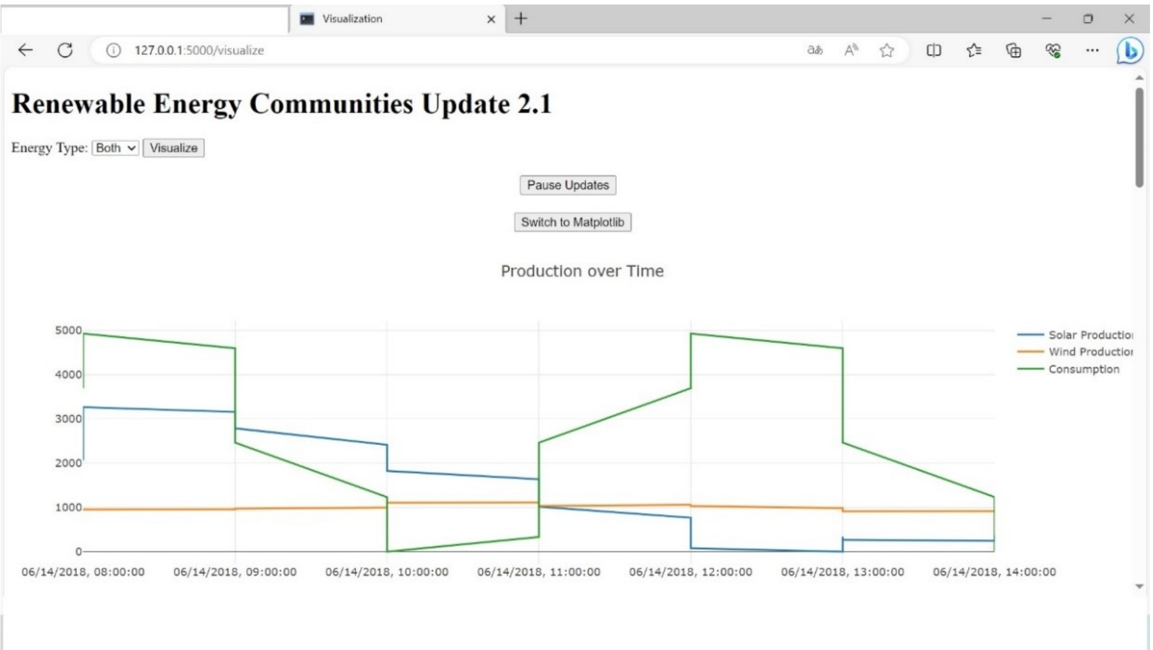
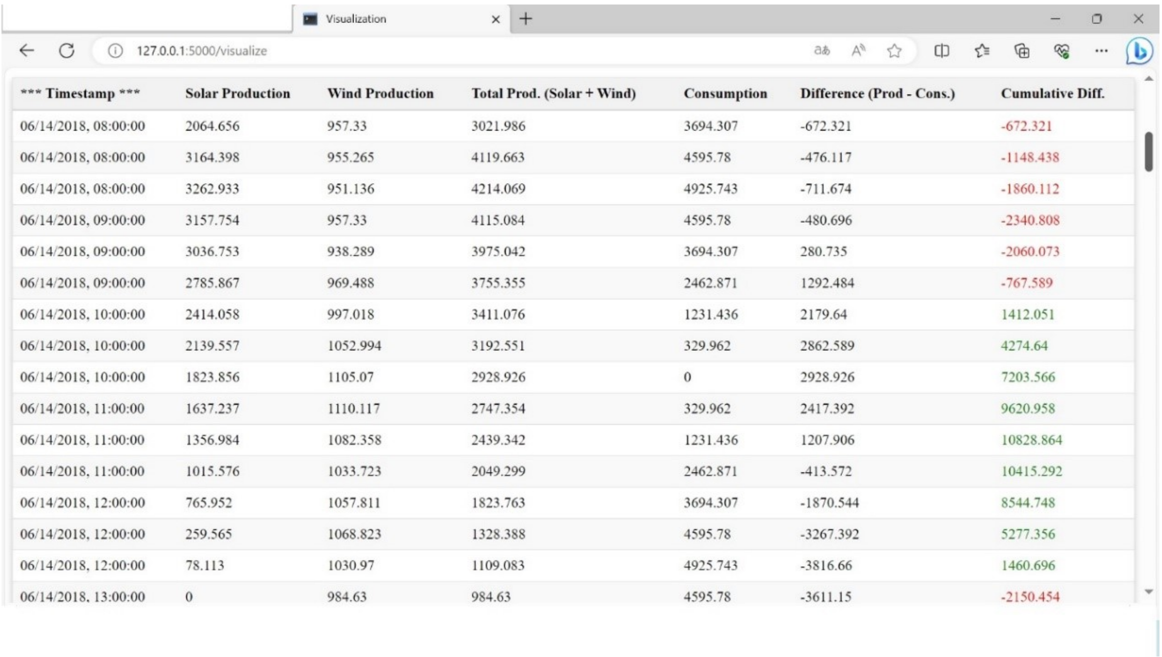


Figure 3.15: User interface to interact with wind and/or solar prediction/production and prediction/consumption.



*** Timestamp ***	Solar Production	Wind Production	Total Prod. (Solar + Wind)	Consumption	Difference (Prod - Cons.)	Cumulative Diff.
06/14/2018, 08:00:00	2064.656	957.33	3021.986	3694.307	-672.321	-672.321
06/14/2018, 08:00:00	3164.398	955.265	4119.663	4595.78	-476.117	-1148.438
06/14/2018, 08:00:00	3262.933	951.136	4214.069	4925.743	-711.674	-1860.112
06/14/2018, 09:00:00	3157.754	957.33	4115.084	4595.78	-480.696	-2340.808
06/14/2018, 09:00:00	3036.753	938.289	3975.042	3694.307	280.735	-2060.073
06/14/2018, 09:00:00	2785.867	969.488	3755.355	2462.871	1292.484	-767.589
06/14/2018, 10:00:00	2414.058	997.018	3411.076	1231.436	2179.64	1412.051
06/14/2018, 10:00:00	2139.557	1052.994	3192.551	329.962	2862.589	4274.64
06/14/2018, 10:00:00	1823.856	1105.07	2928.926	0	2928.926	7203.566
06/14/2018, 11:00:00	1637.237	1110.117	2747.354	329.962	2417.392	9620.958
06/14/2018, 11:00:00	1356.984	1082.358	2439.342	1231.436	1207.906	10828.864
06/14/2018, 11:00:00	1015.576	1033.723	2049.299	2462.871	-413.572	10415.292
06/14/2018, 12:00:00	765.952	1057.811	1823.763	3694.307	-1870.544	8544.748
06/14/2018, 12:00:00	259.565	1068.823	1328.388	4595.78	-3267.392	5277.356
06/14/2018, 12:00:00	78.113	1030.97	1109.083	4925.743	-3816.66	1460.696
06/14/2018, 13:00:00	0	984.63	984.63	4595.78	-3611.15	-2150.454

Figure 3.16: Surplus/deficit prediction/production result table

3.4 - Case study 3: how-to manage smartphone cameras in Android app.

In this third case study, I decide to test ChatGPT-4 feature of knowledge of less common programming languages and language conversion. As project I choose a common Android app feature: manage smartphone cameras. At the web address Getting Started with CameraX (android.com), I found source code written in Kotlin language, but I desire to translate in Java language that I Know so I can modify procedure at my pleasure.

In the example 3.1 (case study 3, example 1), ChatGPT-4 helped me (Figure 3.17) to translate main app procedure.

GG

You

convert kotlin code to java

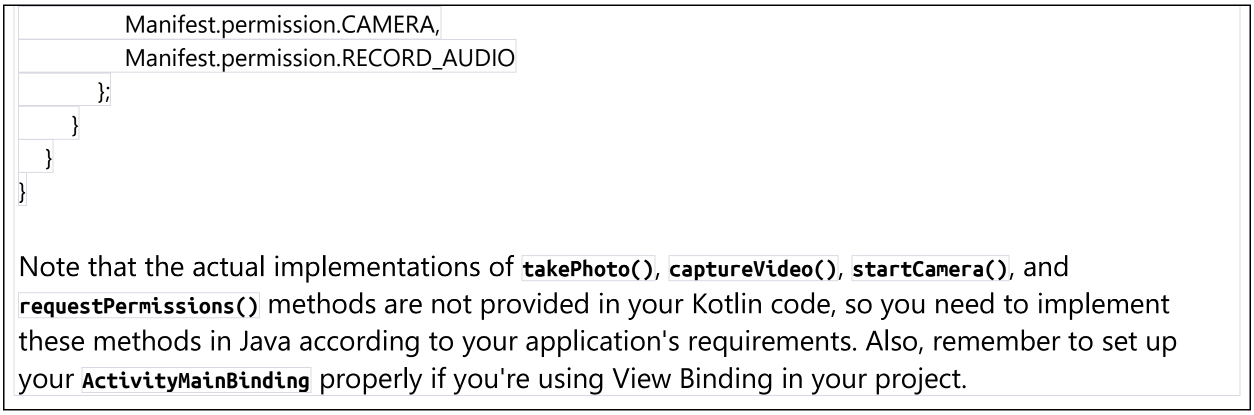
```
package com.android.example.cameraxapp
import android.Manifest
import android.content.ContentValues
import android.content.pm.PackageManager
import android.os.Build
import android.os.Bundle
import android.provider.MediaStore
import androidx.appcompat.app.AppCompatActivity
import androidx.camera.core.ImageCapture
import androidx.camera.video.Recorder
import androidx.camera.video.Recording
import androidx.camera.video.VideoCapture
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.android.example.cameraxapp.databinding.ActivityMainBinding
import java.util.concurrent.ExecutorService
import java.util.concurrent.Executors
import android.widget.Toast
import androidx.activity.result.contract.ActivityResultContracts
import androidx.camera.lifecycle.ProcessCameraProvider
import androidx.camera.core.Preview
import androidx.camera.core.CameraSelector
import android.util.Log
import androidx.camera.core.ImageAnalysis
import androidx.camera.core.ImageCaptureException
import androidx.camera.core.ImageProxy
import androidx.camera.video.FallbackStrategy
import androidx.camera.video.MediaStoreOutputOptions
import androidx.camera.video.Quality
import androidx.camera.video.QualitySelector
import androidx.camera.video.VideoRecordEvent
import androidx.core.content.PermissionChecker
import java.nio.ByteBuffer
import java.text.SimpleDateFormat
import java.util.Locale
typealias LumaListener = (luma: Double) -> Unit
class MainActivity : AppCompatActivity() {
private lateinit var viewBinding: ActivityMainBinding
private var imageCapture: ImageCapture? = null
private var videoCapture: VideoCapture<Recorder>? = null

private var recording: Recording? = null
private lateinit var cameraExecutor: ExecutorService
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
viewBinding = ActivityMainBinding.inflate(layoutInflater)
setContentView(viewBinding.root)
// Request camera permissions
if (allPermissionsGranted()) {
startCamera() }
else { requestPermissions() }
// Set up the listeners for take photo and video capture buttons
viewBinding.imageCaptureButton.setOnClickListener { takePhoto() }
viewBinding.videoCaptureButton.setOnClickListener { captureVideo() }
cameraExecutor = Executors.newSingleThreadExecutor() }
private fun takePhoto() {}
private fun captureVideo() {}
private fun startCamera() {}
private fun requestPermissions() {}
private fun allPermissionsGranted() = REQUIRED_PERMISSIONS.all {
ContextCompat.checkSelfPermission( baseContext, it) ==
PackageManager.PERMISSION_GRANTED } override fun onDestroy() {
super.onDestroy() cameraExecutor.shutdown() }
companion object {
private const val TAG = "CameraXApp"
private const val FILENAME_FORMAT = "yyyy-MM-dd-HH-mm-ss-SSS"
private val REQUIRED_PERMISSIONS = mutableListOf (
Manifest.permission.CAMERA, Manifest.permission.RECORD_AUDIO ).apply {
if (Build.VERSION.SDK_INT <= Build.VERSION_CODES.P) {
add(Manifest.permission.WRITE_EXTERNAL_STORAGE) } }.toTypedArray()
}
}
```

ChatGPT

ChatGPT

To convert the Kotlin code for an Android app using CameraX to Java, we need to adapt several aspects like property declarations, nullability handling, extension functions (like `allPermissionsGranted()`), and the way we define constants and companion objects. Here's how the Kotlin code can be translated into Java for the `MainActivity` class of the `com.android.example.cameraxapp` package:

```
package com.android.example.cameraxapp;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.ImageCapture;
import androidx.camera.video.Recorder;
import androidx.camera.video.Recording;
import androidx.camera.video.VideoCapture;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import com.android.example.cameraxapp.databinding.ActivityMainBinding;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding viewBinding;

    private ImageCapture imageCapture = null;
    private VideoCapture<Recorder> videoCapture = null;
    private Recording recording = null;

    private ExecutorService cameraExecutor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        viewBinding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(viewBinding.getRoot());

        // Request camera permissions
        if (allPermissionsGranted()) {
            startCamera();
        } else {
            requestPermissions();
        }

        // Set up the listeners for take photo and video capture buttons
        viewBinding.getImageCaptureButton().setOnClickListener(view -> takePhoto());
        viewBinding.getVideoCaptureButton().setOnClickListener(view -> captureVideo());

        cameraExecutor = Executors.newSingleThreadExecutor();
    }

    private void takePhoto() {
        // Implementation
    }

    private void captureVideo() {
        // Implementation
    }

    private void startCamera() {
        // Implementation
    }

    private void requestPermissions() {
        // Implementation
    }

    private boolean allPermissionsGranted() {
        for (String permission : REQUIRED_PERMISSIONS) {
            if (ContextCompat.checkSelfPermission(this, permission) != PackageManager.PERMISSION_GRANTED) {
                return false;
            }
        }
        return true;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        cameraExecutor.shutdown();
    }

    private static final String TAG = "CameraXApp";
    private static final String FILENAME_FORMAT = "yyyy-MM-dd-HH-mm-ss-SSS";
    private static final String[] REQUIRED_PERMISSIONS;

    static {
        if (Build.VERSION.SDK_INT <= Build.VERSION_CODES.P) {
            REQUIRED_PERMISSIONS = new String[]{
                Manifest.permission.CAMERA,
                Manifest.permission.RECORD_AUDIO,
                Manifest.permission.WRITE_EXTERNAL_STORAGE
            };
        } else {
            REQUIRED_PERMISSIONS = new String[]{
```



```
                Manifest.permission.CAMERA,
                Manifest.permission.RECORD_AUDIO
            };
        }
    }
}
```

Note that the actual implementations of `takePhoto()`, `captureVideo()`, `startCamera()`, and `requestPermissions()` methods are not provided in your Kotlin code, so you need to implement these methods in Java according to your application's requirements. Also, remember to set up your `ActivityMainBinding` properly if you're using View Binding in your project.

Figure 3.17- ChatGPT-4 chat prompt and result of Example 3.1.

The result java source code translated from ChatGPT-4 inserted in default java app has correctly worked after some modification. The same for the translation of the rest of the entire app source code.

CHAPTER 4

Conclusion

4.1 - GAI tools open questions: ethic, safety, privacy and copyright concerns.

The rapid GAI evolution promises enhanced capabilities for generating content that is both realistic and of high quality. In the previous 2 chapters, I showed how GAI tools as ChatGPT-4 has potentiality to revolutionize CS education in different way but incorporating GAI tools to enhance the teaching and learning experience, should be done thoughtfully, balancing the benefits with potential risks [24] and always maintaining educational standards and ethical norms. In tables 4.1, I summarized all the open questions to solve, grouped in 4 macro concern: ethic, safety, privacy and copyright.

Ethic [36]	
Concern	**Description**
Environment Protection	Large-scale GAI models require substantial computational power and energy, potentially contributing to carbon emissions and global warming.
Limited Transparency	GAI can generate realistic-looking content that is difficult to distinguish from human-created content, raising concerns about authenticity, consent, and misinformation.
Bias and Discrimination	GAI systems can inherit biases present in their training data or in the people who create them, leading to unfair or discriminatory outcomes, especially for underrepresented or marginalized groups.
Responsibility	Determining responsibility for harm or decisions made by a GAI system is challenging. It's unclear whether the developer, the user, or the machine itself should be held accountable.

Safety [37]	
Concern	**Description**
Loss Control	There are concerns about the loss of control over highly autonomous GAI systems, especially if they make decisions that are not understood or anticipated by human overseers.
Security	GAI systems can be vulnerable to attacks, including adversarial attacks, that can manipulate their output.
Reliability and Robustness	GAI must be reliable and robust to errors and unforeseen inputs. Failures can lead to significant consequences, especially in critical applications like healthcare or transportation.

Privacy [38]	
Concern	**Description**
Data Privacy	GAI requires large amounts of data, raising concerns about data privacy and consent. Individuals might not be aware of how their data is being used.
Surveillance	GAI can be used for surveillance, tracking, and profiling, leading to an invasion of privacy and potential misuse by authorities or corporations.
Anonymity	GAI can potentially de-anonymize data, revealing sensitive information about individuals even from anonymized datasets.

Copyright [39]	
Concern	**Description**
Ownership of AI-Generated Content	Authorship: Challenges in determining the author of AI-generated content. Traditional laws focus on human authorship, but AI challenges this. Legal Recognition: Varying approaches in different jurisdictions to recognize AI-generated content.
Usage of Training Data	Infringement: Issues arising from using copyrighted material for GAI training without permission. Fair Use: Debate over the fair use of copyrighted data in GAI training.
Licensing and Permissions	AI Outputs: Complications in licensing AI-generated content due to unclear ownership. Training Data Licensing: The need for clear licenses and permissions for using existing content in GAI training.
Moral Rights	Integrity and Attribution: Questions about the rights to integrity and attribution for GAI-created works, challenging traditional concepts of moral rights.
Legal Adaptations	New Laws: The need for new laws and regulations for AI-related copyright issues. International Standards: The necessity of international standards and agreements due to the global nature of GAI.

Tables 4.1- Overview of all GAI open questions to solve.

Current GAI regulations in major country in Appendices A.
Current ChatGPT-4 terms of use and legal issues in Appendices D.

4.2 - Future scenarios and final considerations

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." - Weiser (1991)

About future scenarios and final consideration, I start from considerations presented in the article [31] outlined in the Figure 4.1.

> Matt Welsh claims they herald the “end of programming” and believes there is major upheaval ahead for which few are prepared, as most of the classic computer science will become irrelevant. The immediate and long-term impacts on novice programmers, students, and their educators is a conversation fewer are having. Experts uniquely appreciate this new technology only because they already understand the underlying fundamentals. Novices may be able to generate code by communicating natural language descriptions to the models, but natural language is ambiguous and the ability to read and comprehend code is arguably more important now than ever. Code literacy skills are essential to critically analyse what is being produced to ensure alignment between one’s intentions and the generated code. Without the skills to read, test and verify that code does what is intended, users risk becoming mere consumers of the generated content, relying on blind faith more than developed expertise. Although professional developers may indeed spend less time in the future writing ‘low-level’ code, we argue that writing code remains a valuable way for novices to learn the fundamental concepts essential for code literacy. Reading and writing code are complementary and related skills, and developing both has been a longstanding goal for students learning programming. Although we do expect to see some shift in emphasis, even in introductory courses, towards modifying code generated by GAI tools, the ability to edit such outputs and compose code in today’s high-level languages will likely remain a fundamental skill for students of computing. This aligns with Yellin’s recent viewpoint that as programs increase in complexity, natural language becomes too imprecise an instrument with which to specify them. At some point, editing code directly is more effective than issuing clarifying instructions in natural language. Tools like ChatGPT-4, harnessed correctly, have the potential to be valuable assistants for this learning. We see these tools being used to explain concepts to a broad and diverse range of learners, generate exemplar code to illustrate those concepts, and to generate useful learning resources that are contextualized to the interests of individuals. We also see new pedagogies emerging that leverage code generation tools, including explicit teaching of effective ways to communicate with the tools, tasks that focus on problem specification rather than implementation, and a renewed focus on test driven development. These tools will also be leveraged by professional programmers to increase productivity and even their non-coding managers to better understand the code their developers are writing. Educators need to adapt quickly as the use of these tools proliferates. Many existing assessment approaches may no longer be effective, and students must be taught to use these tools responsibly to support their learning rather than becoming over-reliant on them. There is also a growing need to teach new topics that focus on the wider societal implications of GAI models. These tools raise important and challenging legal, ethical, economic, and other considerations that are currently (re)shaping the society that our students will soon enter as graduates. As educators one of our most important, and challenging, jobs are to prepare students for such rapidly changing landscapes. We believe it is imperative to get ahead of the use of these tools, incorporate them into our classrooms from the very beginning, and teach students to use them responsibly via modern pedagogies. In short, we must adopt or perish. However, adoption is not a compromise or a concession; rather it is a path to a new flourishing.

Figure 4.1- CACM Article by Matt Welsh.

More specifically, about use of GAI tools in CS education issues and potential challenges [19][21][25], including misuse by students, I adopt considerations presented by Prof. Nardelli of Roma Tor Vergata University, director of the National Laboratory "Computer Science and School" of CINI and former president of Informatics Europe in his public interview [40] and public blog [41] reported in Figure 4.2.

> GAI tools can indeed lead to increased productivity for teachers if, through them, they are able to reduce the time spent on repetitive tasks, thereby allowing them to focus more on caring for students in greater need. Among these routine activities, we can include, for example: generating texts for exercises and exams, generating presentations from texts (summarization), generating texts from detailed outlines, providing explanations for simple clarification requests, etc. Of course, it is crucial to always be aware that they have margins of error that require always checking what they produce (and, therefore, to have a good understanding of the subject matter being dealt with) and that the ultimate responsibility always lies with the individual.
>
> In general, however, unsupervised use of AI tools by students for school activities should be avoided. Some have suggested, for example, that students could use them at home to get a first assessment of their homework. This idea is somewhat bizarre, as they might then also use them to produce the homework itself! But above all, given the potential for these tools to generate incorrect outputs, it is precisely the students who need this function the most who are least able to recognize any errors they might make. Obviously, students will still have access to these tools through their smartphones, so it will be necessary to resist prohibitive temptations in favor of educating and spreading knowledge, conducted, for example, through collective use and critical discussion in class. On the other hand, this is probably good news; the fact that anyone can use these tools to produce a well-written text will likely lead to a resurgence and greater presence of direct human relationships and oral dialogue.
>
> For certain school tracks, particularly technical institutes for computer science, curricula will need to be adjusted (and the necessary professional development for teachers developed) to include appropriate scientific and technological training in this sector, which will significantly influence the labor market in the near future. Given that those who choose this technical path are generally oriented towards entering the job market directly upon completion, it is appropriate that preparation in this area becomes part of their training.
>
> In conclusion, it is essential that AI tools used in the educational sector are controlled/monitored by the public sector to avoid the risks associated with purely commercial products, which lack independent checks on how they were developed, with what data they were trained, what security checks they were subjected to, how they use and manipulate the data provided, and what potential gender-based or other forms of discrimination they may perpetuate. All these aspects have a significant social impact and thus require the highest level of scrutiny. It should also be remembered that this technology is still experimental, and every interaction with it contributes to its development, provided entirely for free. This would be acceptable only for a tool owned by the community, whose improvements benefit everyone.

Figure 4.2- Prof. Nardelli of Roma Tor Vergata University public interview

GAI ethics, safety, privacy and copyright are multidimensional challenges that require a collaborative approach involving stakeholders from various sectors, including government, academia, industry, and civil society, to navigate the complex landscape and ensure the beneficial and equitable use of GAI. Ultimately, it is crucial to strike a balance between leveraging the benefits of GAI and safeguarding core human values. In the table 4.2, I summarized some possible solutions and best practices [35].

Strategy	Description
Emphasizing Education	Provide comprehensive training to employees on ethical implications, risks, and data usage guidelines in GAI. Create a culture that encourages questions and understanding of GAI's complex nature.
Investing in Robust Data Security	Protect data assets using advanced security techniques like encryption and anonymization. Employ digital twins for testing to identify and address security flaws without risking the actual system.
Encouraging Independent Verification	Encourage users to independently verify GAI outputs to prevent the spread of inaccurate or misleading information, maintaining trust in the technology.
Keeping Up with the GAI Landscape	Stay informed about the evolving AI landscape to ensure GAI use aligns with ethical best practices and new regulations.
Implementing Clear Usage Policies	Develop a well-defined acceptable use policy for GAI, referencing frameworks like NIST's AI Risk Management Framework or the EU's Ethics Guidelines for Trustworthy AI. Regularly review and update the policy.
Use of Individually Trained GAI	Utilize individually trained GAI to minimize bias and hallucinations and to protect corporate and consumer data.

Tables 4.2- Overview of GAI open questions possible solution and best practices.

I'm agree with final considerations presented in the same article [35] reported below.

> The impact of this technology on society will largely depend on our ability to responsibly navigate its concerns landscape. We are operating in a domain where legal regulations are still catching up, but that doesn't mean organizations are helpless. Quite the contrary, they are in a unique position to shape the future by adopting proactive measures. By establishing robust ethical use policies, companies can take the helm in safeguarding the rights of their customers and employees. They can champion privacy by enforcing stringent data management techniques, and combat biases by ensuring diversity in their GAI training data. By emphasizing transparency in their GAI processes and acknowledging the environmental footprint of these technologies, organizations not only protect their reputation but also foster trust and pave the way for responsible GAI usage. Let's remember that GAI, like any technology, is a tool in our hands. It's our responsibility to ensure that this tool is used to create a world where innovation and ethics not only co-exist but thrive together.
>
> In conclusion, navigating the ethics of GAI is not a destination but a journey - a continual process of learning, adapting, and evolving.

BIBLIOGRAPHY

[1]. File:AI-ML-DL.svg - Wikimedia Commons

[2]. A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT (arxiv.org) by YIHAN CAO, Lehigh University & Carnegie Mellon University, SIYU LI, YIXIN LIU, ZHILING YAN, YUTONG DAI, LICHAO SUN, Lehigh University, USA PHILIP S. YU, University of Illinois at Chicago.

[3]. Language Models with Transformers by Chenguang Wang Mu Li Alexander J. Smol

[4]. The Story of RLHF: Origins, Motivations, Techniques, and Modern Applications (substack.com) by Cameron R. Wolfe, Rice University Ph. D.

[5]. Research index (openai.com) by OpenAI Inc.

[6]. Guida di Bard (google.com) by Google LLC

[7]. Overview of Microsoft Search in Bing | Microsoft Learn by Microsoft Corporation

[8]. Research \ Anthropic by Antrophic PBC

[9]. Aria browser AI | Free AI with real-time web access | Opera Browser by Otello Corporation

[10]. ChatGPT v Bard v Bing v Claude 2 v Aria v human-expert. How good are AI chatbots at scientific writing? By Edisa Lozić1 and Benjamin Štular.

[11]. wasabi.essi.upc.edu by Prof. Marc Alier, Politecnical University of Catalonia.

[12]. a-review-of-technological-tools-in-teaching-and-learning-computer-science-7731.pdf (ejmste.com) by Qurat-ul-Ain 1, Farah Shahid 1, Muhammad Aleem 1*, Muhammad Arshad Islam 1, Muhammad Azhar Iqbal 1, Muhammad Murtaza Yousaf 2 1 Department of Computer Science, Capital University of Science and Technology Islamabad 44000, PAKISTAN 2 Punjab University College of Information Technology, University of the Punjab Lahore, PAKISTAN

[13]. Chat GPT a scuola, ecco come potrebbe cambiare la didattica - Agenda Digitale by Prof. Paolo Ferri, Università degli studi Milano Bicocca

[14]. Teaching with AI (openai.com) by OpenAI Inc.

[15]. The Educational Affordances and Challenges of ChatGPT: State of the Field | TechTrends (springer.com) by Helen Crompton, Diane Burke Old Dominion University, Norfolk, USA

[16]. ChatGPT impacts in programming education: A... | F1000Research by Christos-Nikolaos Anagnostopoulos of University of the Aegean, Greece.

[17]. Examining Science Education in ChatGPT: An Exploratory Study of Generative Artificial Intelligence | Journal of Science Education and Technology (springer.com) by Grant Cooper Curtin University, WA, 6102, Bentley, Australia

[18]. Exploring the Use of ChatGPT as a Tool for Learning and Assessment in Undergraduate Computer Science Curriculum: Opportunities and Challenges (arxiv.org) by Basit Qureshi Department of Computer Science, Prince Sultan University, Saudi Arabia

[19]. [2308.04309] Generative AI in Computing Education: Perspectives of Students and Instructors (arxiv.org) by Cynthia Zastudil, Magdalena Rogalska, Christine Kapp, Jennifer Vaughn, Stephen MacNeil Computer & Information Sciences Temple University Philadelphia, PA

[20]. [2306.02608] Computing Education in the Era of Generative AI (arxiv.org) by Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly University of Auckland New Zealand, James Prather, Brent N. Reeves Abilene Christian University, Texas USA, Brett A. Becker, Eddie Antonio Santos University College Dublin Ireland, Arto Hellas, Juho Leinonen, Sami Sarsa Aalto University Espoo Finland.

[21]. [2310.00658] The Robots are Here: Navigating the Generative AI Revolution in Computing Education (arxiv.org) by James Prather, Brent N. Reeves Abilene Christian University, Texas USA, Paul Denny, Juho Leinonen, Andrew Luxton-Reilly University of Auckland New Zealand, Brett A. Becker University College Dublin Ireland, Ibrahim Albluwi Princess Sumaya University for Technology Jordan, Michelle Craig, Andrew Petersen University of Toronto Canada, Hieke Keuning Utrecht University Heidelberglaan Netherlands, Natalie Kiesler DIPF Leibniz Institute for Research and Information in Education Rostocker Germany, Tobias Kohn TU Wien Austria, Stephen MacNeil Temple University Philadelphia USA, Raymond Pettit University of Virginia Virginia USA, Jaromir Savelka Carnegie Mellon University Pennsylvania USA

[22]. [2311.10199] Generative AI in Undergraduate Information Technology Education - Insights from nine courses (arxiv.org) by Anh Nguyen Duc, Tor Lønnestad, Ingrid Sundbø, Marius Rohde Johannessen, Veralia Gabriela, Salah Uddin Ahmed and Rania El-Gazzar

[23]. [2306.17156] Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human TutorsThis article is a full version of the poster (extended abstract) from ICER'23 [1]. (arxiv.org) by Tung Phung MPI-SWS, Victor-Alexandru Pădurean MPI-SWS, José Cambronero Microsoft, Sumit Gulwani Microsoft, Tobias Kohn TU Wien, Rupak Majumdar MPI-SWS Adish Singla MPI-SWS, Gustavo Soares Microsoft

[24]. Frontiers | The use of ChatGPT in teaching and learning: a systematic review through SWOT analysis approach (frontiersin.org) by Duong Thi Thuy Mai Can Van Da Nguyen Van Hanh University of Science and Technology, Hanoi, Vietnam

[25]. Education Sciences | Free Full-Text | Challenges and Opportunities of Generative AI for Higher Education as Explained by ChatGPT (mdpi.com) by Rosario Michel-Villarreal University of Leeds, UK, Eliseo Vilalta-Perdomo, David Ernesto Salinas-Navarro Aston University, Birmingham, UK, Ricardo Thiery-Aguilera School of Engineering and Sciences, Tecnológico de Monterrey, Monterrey, Mexico Flor Silvestre Gerardou Falmouth University, UK

[26]. Challenges for Computer Science Education Arising from new AI Systems like ChatGPT by Robert Mischak University of Applied Sciences - Styria, Robert Pucher University of Applied Sciences – Austria

[27]. ChatGPT for Teaching and Learning: An Experience from Data Science Education (arxiv.org) by YONG ZHENG, Illinois Institute of Technology, USA

[28]. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning - ScienceDirect by Ramazan Yilmaz, Fatma Gizem Karaoglan Yilmaz Bartin University, Turkey

[29]. Students' Voices on Generative AI: Perceptions, Benefits, and Challenges in Higher Education by Cecilia Ka Yuk Chan, Wenjie Hu University of Hong Kong, Pokfulam, Hong Kong

[30]. An example of LLM prompting for programming (martinfowler.com) by Martin Flower

[31]. The End of Programming | January 2023 | Communications of the ACM by Matt Welsh

[32]. Generative AI and the future of education - UNESCO Digital Library by UNESCO. Assistant Director-General for Education, 2018- (Giannini, Stefania)

[33]. A Global Overview of Generative AI Regulations (pillsburylaw.com) by Nancy A. Fischer, Elizabeth Vella Moeller, The Honorable Jerry McNerney, Amaris Trozzo, Steven Farmer, Jenny (Jia) Sheng, Jack Ko, Ph.D., Jenny Y. Liu, Chunbin Xu, Fred Ji, Wenjun Cai

[34]. Artificial intelligence act (europa.eu) by EPRS | European Parliamentary Research Service

[35]. Generative AI - Ethical concerns and how to solve them (linkedin.com) by Mark Reineke MSc Emirati Arabi Uniti.

[36]. AI in education: A guide to the safe and responsible use of AI in schools - Insights - Colin Biggers & Paisley (cbp.com.au) by Kristen Lopes and Gabriella Mills.

[37]. Sharing Information with ChatGPT Creates Privacy Risks (g2.com) by Brandon Summers-Miller.

[38]. Copyright in generative deep learning | Data & Policy | Cambridge Core by Cambridge University Press – May 2022

[39]. Generative AI: the US Copyright class action against OpenAI - Kluwer Copyright Blog (kluweriplaw.com) by Gianluca Campus Università di Milano

[40]. Il ciclone ChatGPT e la scuola: 5 punti chiave - Startmag by Prof. Enrico Nardelli Università di Roma Tor Vergata.

[41]. Link&Think: Intelligenza artificiale e scuola: facciamo chiarezza (link-and-think.blogspot.com) by Prof. Enrico Nardelli Università di Roma Tor Vergata.

[42]. OpenAI: OpenAI gets partial win in authors US copyright lawsuit - The Economic Times (indiatimes.com) by The Economic Times tech.

[43]. The Intercept, Raw Story and AlterNet sue OpenAI for copyright infringement | Technology | The Guardian by Nick Robins The Guardian

APPENDICES

A - GAI: regulations in major country

Below, the current (November 2023) developments of legislations (Figure A.1) [33] on GAI in the United States (U.S.), Europe (EU), the United Kingdom (UK) and the People's Republic of China (China or the PRC).

United States. Congressional leaders are intensifying efforts to develop legislation directing agency regulation of AI technology. In June, Senate Majority Leader Chuck Schumer (D-NY) publicly announced the SAFE Innovation Framework, which sets priorities for AI legislation, focusing on: security, accountability, protecting our foundations and explain ability. The framework's goal is to deliver security without compromising innovation. Although passage in 2023 is uncertain, we can expect that Congress will continue to introduce legislation, hold hearings and deploy the AI Forums throughout the rest of the session, giving industry several opportunities to engage with their representatives and senators before legislation is enacted.

Also, at a May 16 hearing of the Senate Judiciary Subcommittee on Privacy, Technology and the Law, several senators indicated support for the creation of a new federal agency dedicated to regulating AI, which could include licensing activities for AI technology. There were also calls for creating an international AI regulatory body modeled after the International Atomic Energy Agency.

On May 23, the White House revealed three new steps advancing the research, development and deployment of AI technology nationwide. In addition, the Office of Science and Technology Policy (OSTP) completed a public comment period soliciting input to develop a comprehensive National AI Strategy, focused on promoting fairness and transparency in AI while maximizing AI benefits. The results of the public feedback will be made public and move OSTP to the next stage of developing the National AI Strategy.

U.S. federal agencies are also engaging with AI as it intersects with their respective jurisdictional and legislative authority, often issuing guidance to regulated entities, explaining how the agency will apply existing law to any AI violations. For example, the Federal Trade Commission (FTC) has been active in regulating deceptive and unfair practices attributed to AI, particularly enforcing statutes such as the Fair Credit Reporting Act, Equal Credit Opportunity Act and FTC Act.

European Union. The EU has also made steady progress in shaping its proposed AI law, known as the "AI Act," which has entered the final stage of the legislative process. The aim is to agree to a final version of the law by the end of 2023, after which there will likely be a 24-month transition period before it applies.

The proposed AI Act classifies AI usage based on risk levels, prohibiting certain uses (for example, real-time biometric identification surveillance systems used in public places and subliminal techniques which may cause harm) and imposing stringent monitoring and disclosure requirements for high-risk applications compared to lower-risk ones.

The EU's objective is to ensure that AI developed and utilized within Europe aligns with the region's values and rights, including ensuring human oversight, safety, privacy, transparency, non-discrimination, and social and environmental well-being. The proposed penalties for breach could be as high as 7% of global revenue or €40 million.

Further, on September 28, 2022, the European Commission proposed a new law, known as the "AI Liability Directive," aimed at adapting non-contractual civil liability rules to AI. The proposed law (which is closely tied to the AI Act) aims to establish uniform rules for damages caused by AI systems, providing broader protection for victims and fostering the AI sector by increasing guarantees. It will address specific difficulties of proof linked with AI, requiring EU Member States to empower national courts to order the disclosure of relevant evidence about specific high-risk AI systems. The proposed law will impact both users and developers of AI systems, providing clarity for developers about accountability in the event of an AI system failure, and facilitating compensation recovery for victims of crimes associated with AI systems. The negotiations on the new law are ongoing, and it is not yet clear when it will be adopted.

United Kingdom. On March 29, 2023, the UK government released a white paper outlining its pro-innovation approach to AI regulation. Rather than creating new laws or a separate AI regulator, as things stand, existing sectoral regulators will be empowered to regulate AI in their respective sectors. The focus is on enhancing existing regimes to cover AI and avoiding heavy-handed legislation that could hinder innovation.

The proposed regulatory framework outlined in the white paper defines AI based on two key characteristics, namely adaptivity and autonomy. The white paper holds that by defining AI with reference to these characteristics and designing the regulatory framework to address the challenges created by these characteristics, UK lawmakers can future proof the framework against unanticipated new technologies.

The white paper also sets out five "values-focused cross-sectorial" principles that regulators should adhere to when addressing the risks associated with AI. The principles include: (i) safety, security and robustness, (ii) appropriate transparency and explain ability, (iii) fairness, (iv) accountability and governance, and (v) contestability and redress.

The principles build on, and reflect the UK government's commitment to, the Organisation for Economic Cooperation and Development's (OECD) values-based AI principles, which promote the ethical use of AI. The aim of the principles-based approach is to allow the framework to be agile and proportionate. While not legally binding at the outset, the UK government anticipates that the principles may become enforceable in the future, depending on the evolving landscape of AI technology and its societal impact.

In addition to these principles, the UK government plans to establish central functions to support regulators in their AI oversight roles and make sure the regulatory framework operates proportionately and supports innovation. The white paper is silent on which specific entity or entities will undertake these central functions.

The white paper accompanies the investment of £2 million by the UK government to fund a new sandbox to enable AI innovators to test new AI products prior to market launch. The sandbox will enable businesses to test how AI regulations could apply to their products.

Following publication of the white paper, the UK government will continue to work with businesses and regulators as it starts to establish the central functions identified. The UK government will publish an AI regulatory roadmap alongside its response to the consultation on this white paper. In the long term, 12 months or more after the publication of the white paper, the UK government plans to implement all central functions, support regulators in applying cross-sectoral principles, publish a draft AI risk register, develop a regulatory sandbox, and release a monitoring and evaluation report to assess the framework's performance.

China. On July 13, 2023, the Cyberspace Administration of China (CAC) issued the final version of the *Interim Administrative Measures for Generative Artificial Intelligence Service* (PRC AI Regulations). GAI services provided to the public within the PRC fall within the scope of these regulations, which primarily address content generation using AI technology (GAI Services). The PRC AI Regulations explicitly exclude from their scope non-public service providers, such as industry organizations, enterprises, academic and research institutions, and public cultural institutions engaged in research, development and application of GAI technology.

The PRC AI Regulations introduce significant obligations for providers of GAI Services. These requirements include monitoring and controlling content generated by their services. Providers are required to promptly remove any illegal content, take actions against users engaged in illegal activities, and report to the authorities. Additionally, the providers must mark generated content with appropriate labels and use legitimate sources for data training while respecting intellectual property rights and obtaining consent for personal information processing. Reiterating the existing cybersecurity and personal privacy rules in China, the PRC AI Regulations mandate protecting the users' personal information, prohibiting illegal collection and sharing of identifiable data.

China is also likely to adopt an industry-oriented regulatory model, with different governmental departments regulating GAI Services within their specific fields. Industry-specific AI regulations and classification guidelines are expected to be introduced.

The PRC AI Regulations are the latest addition to the *Administrative Provisions on Algorithm Recommendation for Internet Information Services* (Algorithm Provisions, effective as of March 1, 2022) and *Administrative Provisions on Deep Synthesis of Internet Information Services* (Deep Synthesis Provisions, effective as of January 10, 2023).

The Algorithm Provisions apply to any entity that uses algorithm recommendation technologies (including without limitation technologies for generation and synthesis, personalized push, sorting and selection, retrieval and filtering, scheduling decision-making) to provide internet information services within mainland China. The Algorithm Provisions, among others, require an algorithm recommendation service provider (which could cover a GAI services provider) with a public opinion attribute or social mobilization ability to carry out a safety assessment in accordance with the application regulations and complete online record-filing formalities within 10 working days from the date it starts to provide services.

The Deep Synthesis Provisions regulate the provision of internet information services in mainland China by applying "deep synthesis technologies," which is defined as "technologies that use generative sequencing algorithms, such as deep learning and virtual reality, to create text, images, audio, video, virtual scenes, or other information." The Deep Synthesis Provisions set out a comprehensive set of responsibilities for deep synthesis service providers and technical supporters concerning data security and personal information protection, transparency, content management and labeling, technical security, etc. A GAI service provider is required to add a mark on content (pictures, videos and other content generated by GAI Services) according to the Deep Synthesis Provisions.

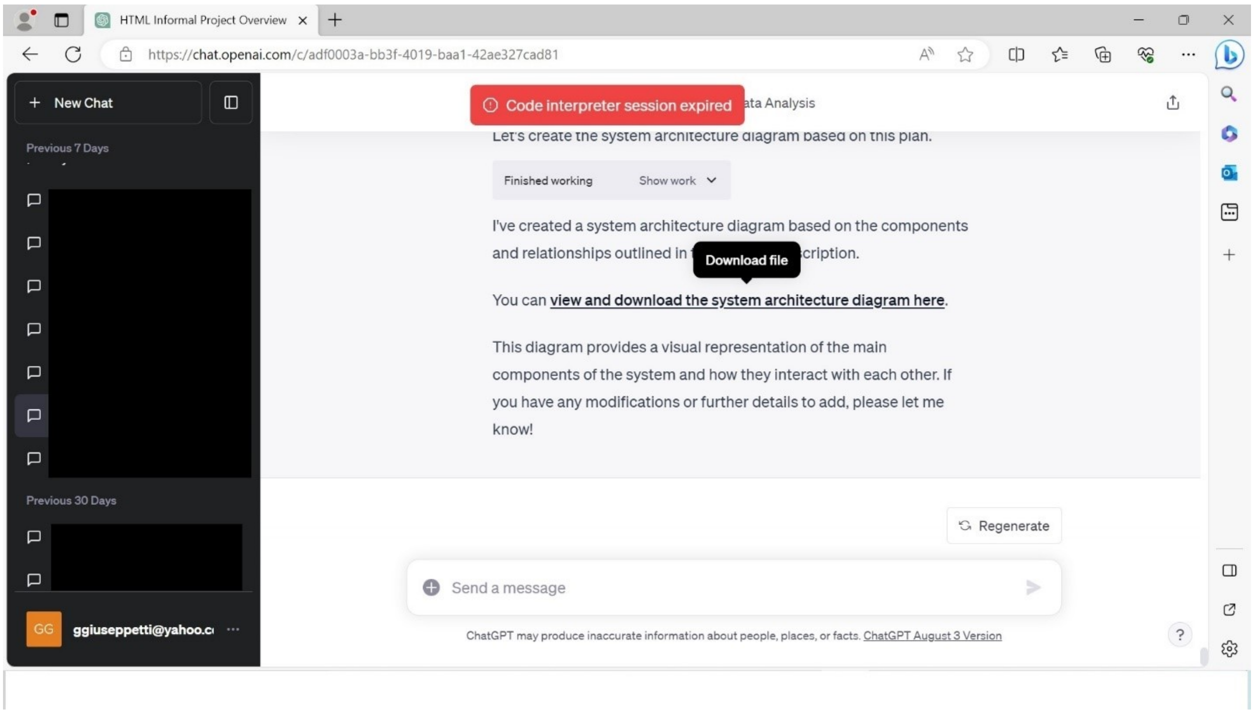
Overall, together with the existing cybersecurity and data privacy rules in China, the PRC AI Regulations aim to establish a framework for responsible and transparent use of GAI Services, imposing significant responsibilities while offering some flexibility to the service providers. The Chinese authorities have placed more emphasis on industrial policies encouraging AI innovations and massive industrial applications rather than restricting the development of AI technologies, which is also reflected in the PRC AI Regulations.

Conclusion

The global landscape of AI governance features diverse strategies. The EU focuses on sector-specific regulation, the United States opts for a decentralized approach featuring federal guidance with local adaptations, and China prioritizes consumer transparency and global AI standards dominance. Companies will need to develop global positions on AI ethics and compliance for their products in order to comply with new regulations.

Figure A.1 – GAI regulations in major country.

B - ChatGPT-4: common troubleshooting examples



[redacted]

[redacted]

Let's create the system architecture diagram based on this plan.

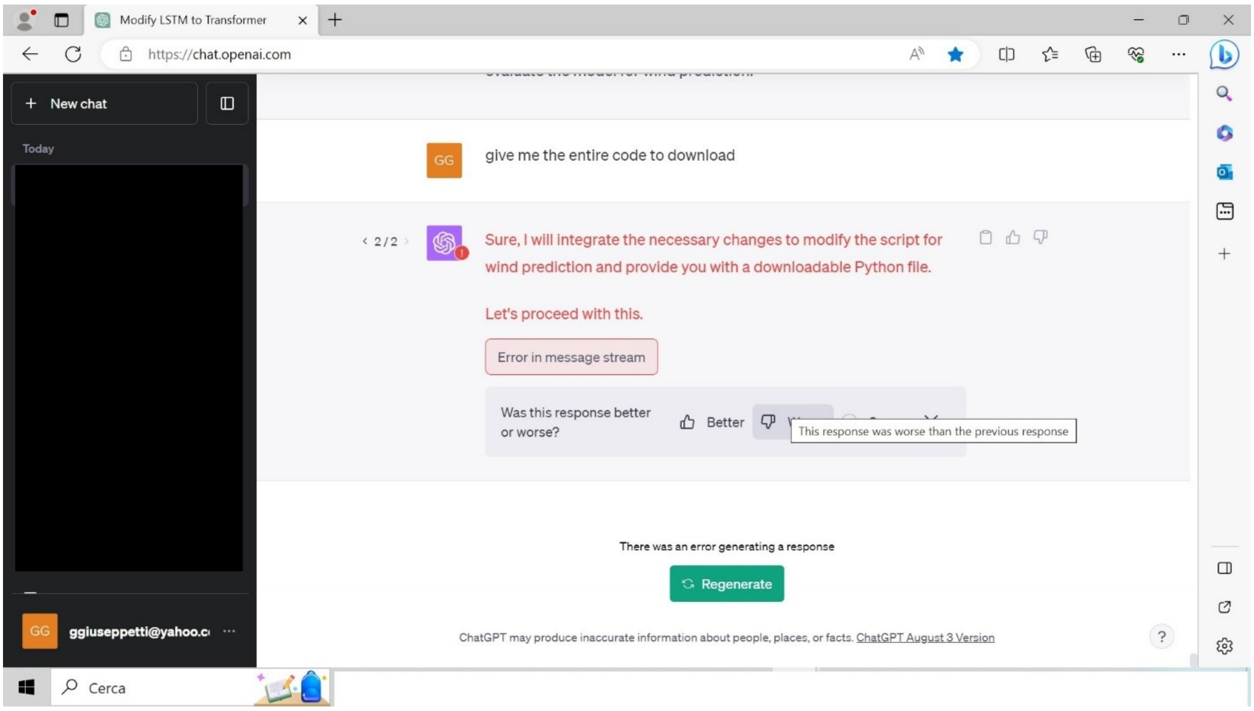
I've created a system architecture diagram based on the components and relationships outlined in cription.

You can **view and download the system architecture diagram here**.

This diagram provides a visual representation of the main components of the system and how they interact with each other. If you have any modifications or further details to add, please let me know!

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure B.1 - The above screenshot shows a message of timed out. From this moment chat has expired. To continue is necessary to recreate knowledge base acquired in another chat.



[redacted]

give me the entire code to download

Sure, I will integrate the necessary changes to modify the script for wind prediction and provide you with a downloadable Python file.

Let's proceed with this.

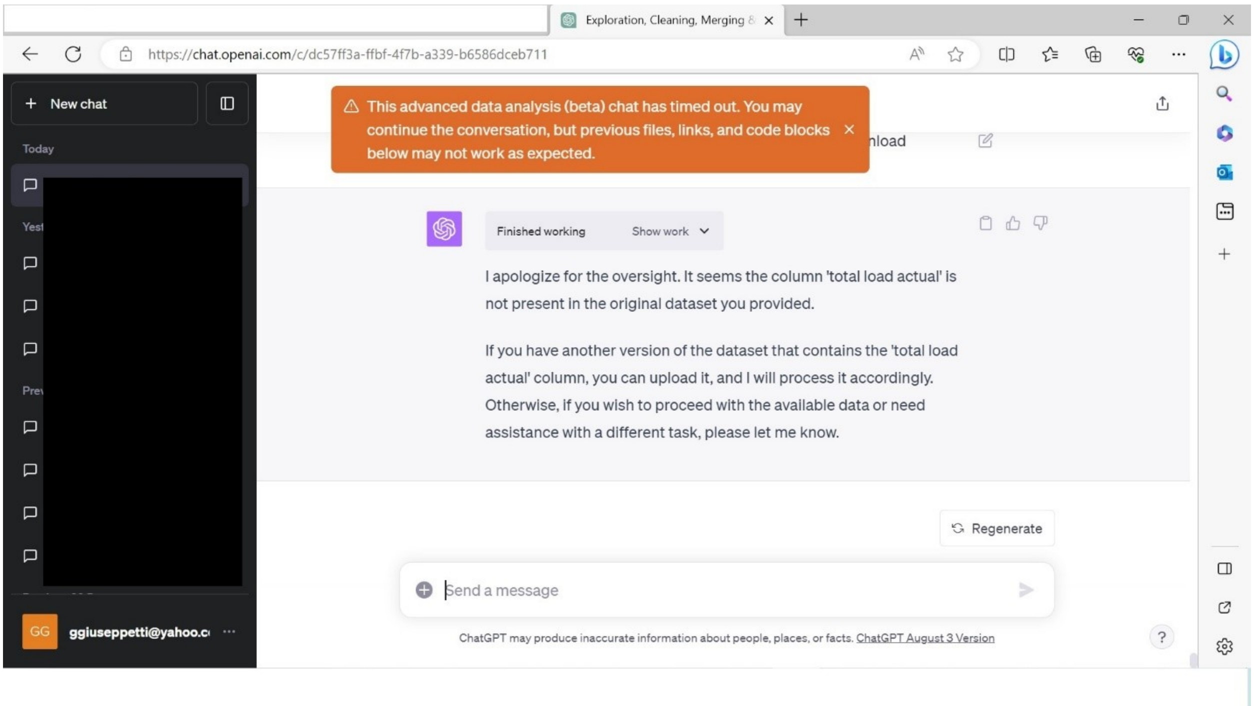
Was this response better or worse?

This response was worse than the previous response

There was an error generating a response

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure B.2 - The above screenshot shows a message of timed out. From this moment chat has expired. To continue is necessary to recreate knowledge base acquired in another chat.



[redacted]

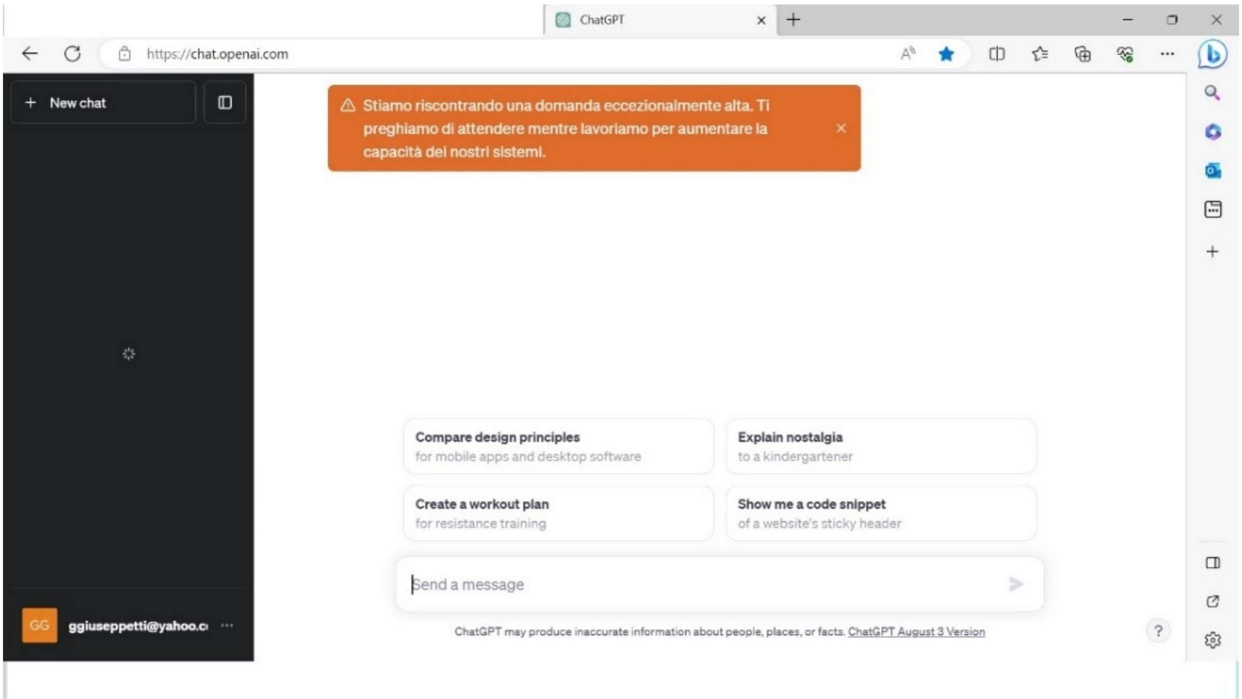
This advanced data analysis (beta) chat has timed out. You may continue the conversation, but previous files, links, and code blocks below may not work as expected.

I apologize for the oversight. It seems the column 'total load actual' is not present in the original dataset you provided.

If you have another version of the dataset that contains the 'total load actual' column, you can upload it, and I will process it accordingly. Otherwise, if you wish to proceed with the available data or need assistance with a different task, please let me know.

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure B.3 - The above screenshot shows a message of timed out. To continue is necessary to recreate partial knowledge base acquired..



Stiamo riscontrando una domanda eccezionalmente alta. Ti preghiamo di attendere mentre lavoriamo per aumentare la capacità dei nostri sistemi.

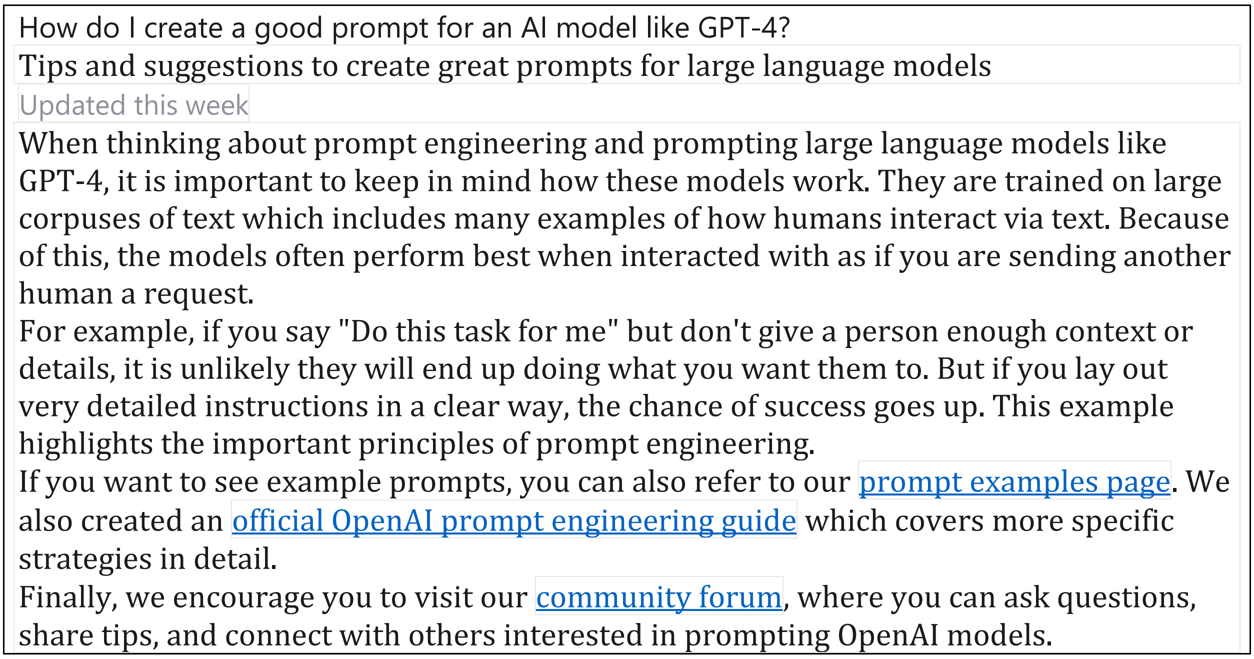
ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure B.4- The above screenshot shows a message of service unavailable. In this case, I can only wait.

N.B. The response from ChatGPT-4 is non-deterministic, so each result is just one run. If you put the same prompt in again, we'll get a different response.

C - ChatGPT-4: OpenAI prompt engineering guidelines

Below (Figure C.1), first current (February 2024) guideline. Full guidelines are available at https://help.openai.com/en/collections/3675942-prompt-engineering



How do I create a good prompt for an AI model like GPT-4?

Tips and suggestions to create great prompts for large language models

Updated this week

When thinking about prompt engineering and prompting large language models like GPT-4, it is important to keep in mind how these models work. They are trained on large corpuses of text which includes many examples of how humans interact via text. Because of this, the models often perform best when interacted with as if you are sending another human a request.

For example, if you say "Do this task for me" but don't give a person enough context or details, it is unlikely they will end up doing what you want them to. But if you lay out very detailed instructions in a clear way, the chance of success goes up. This example highlights the important principles of prompt engineering.

If you want to see example prompts, you can also refer to our prompt examples page. We also created an official OpenAI prompt engineering guide which covers more specific strategies in detail.

Finally, we encourage you to visit our community forum, where you can ask questions, share tips, and connect with others interested in prompting OpenAI models.

Figure C.1- First current (February 2024) prompt engineering OpenAI ChatGPT-4 guideline.

D - ChatGPT-4: terms of use, privacy policy and current legal issues

Current (January 2024) Term of Use, there are interesting specifications about Content, Confidentiality, Security and Data Protection as below reported.

Your Content. You may provide input to the Services ("Input"), and receive output generated and returned by the Services based on the Input ("Output"). Input and Output are collectively "Content." As between the parties and to the extent permitted by applicable law, you own all Input. Subject to your compliance with these Terms, OpenAI hereby assigns to you all its right, title and interest in and to Output. This means you can use Content for any purpose, including commercial purposes such as sale or publication, if you comply with these Terms. OpenAI may use Content to provide and maintain the Services, comply with applicable law, and enforce our policies. You are responsible for Content, including for ensuring that it does not violate any applicable law or these Terms.

Similarity of Content. Due to the nature of machine learning, Output may not be unique across users and the Services may generate the same or similar output for OpenAI or a third party. For example, you may provide input to a model such as "What color is the sky?" and receive output such as "The sky is blue." Other users may also ask similar questions and receive the same response. Responses that are requested by and generated for other users are not considered your Content.

Use of Content to Improve Services. We do not use Content that you provide to or receive from our API ("API Content") to develop or improve our Services. We may use Content from Services other than our API ("Non-API Content") to help develop and improve our Services. You can read more here about how Non-API Content may be used to improve model performance. If you do not want your Non-API Content used to improve Services, you can opt out by filling out this form. Please note that in some cases this may limit the ability of our Services to better address your specific use case.

Accuracy. Artificial intelligence and machine learning are rapidly evolving fields of study. We are constantly working to improve our Services to make them more accurate, reliable, safe and beneficial. Given the probabilistic nature of machine learning, use of our Services may in some situations result in incorrect Output that does not accurately reflect real people, places, or facts. You should evaluate the accuracy of any Output as appropriate for your use case, including by using human review of the Output.

Confidentiality. You may be given access to Confidential Information of OpenAI, its affiliates and other third parties. You may use Confidential Information only as needed to use the Services as permitted under these Terms. You may not disclose Confidential Information to any third party, and you will protect Confidential Information in the same manner that you protect your own confidential information of a similar nature, using at least reasonable care. Confidential Information means nonpublic information that OpenAI or its affiliates or third parties designate as confidential or should reasonably be considered confidential under the circumstances, including software, specifications, and other nonpublic business information. Confidential Information does not include information that: (i) is or becomes generally available to the public through no fault of yours; (ii) you already possess without any confidentiality obligations when you received it under these Terms; (iii) is rightfully disclosed to you by a third party without any confidentiality obligations; or (iv) you independently developed without using Confidential Information. You may disclose Confidential Information when required by law or the valid order of a court or other governmental authority if you give reasonable prior written notice to OpenAI and use reasonable efforts to limit the scope of disclosure, including assisting us with challenging the disclosure requirement, in each case where possible.

Security. You must implement reasonable and appropriate measures designed to help secure your access to and use of the Services. If you discover any vulnerabilities or breaches related to your use of the Services, you must promptly contact OpenAI and provide details of the vulnerability or breach.

Processing of Personal Data. If you use the Services to process personal data, you must provide legally adequate privacy notices and obtain necessary consents for the processing of such data, and you represent to us that you are processing such data in accordance with applicable law. If you will be using the OpenAI API for the processing of “personal data” as defined in the GDPR or “Personal Information” as defined in CCPA, please fill out this form to request to execute our Data Processing Addendum.

In the Privacy Policy, there are interesting specifications reported below.

We collect personal information relating to you (“Personal Information”) as follows:

Personal Information You Provide: We collect Personal Information if you create an account to use our Services or communicate with us as follows:

Account Information: When you create an account with us, we will collect information associated with your account, including your name, contact information, account credentials, payment card information, and transaction history, (collectively, “Account Information”).

User Content: When you use our Services, we collect Personal Information that is included in the input, file uploads, or feedback that you provide to our Services (“Content”).

Communication Information: If you communicate with us, we collect your name, contact information, and the contents of any messages you send (“Communication Information”).

Social Media Information: We have pages on social media sites like Instagram, Facebook, Medium, Twitter, YouTube and LinkedIn. When you interact with our social media pages, we will collect Personal Information that you elect to provide to us, such as your contact details (collectively, “Social Information”). In addition, the companies that host our social media pages may provide us with aggregate information and analytics about our social media activity.

Personal Information We Receive Automatically From Your Use of the Services: When you visit, use, or interact with the Services, we receive the following information about your visit, use, or interactions (“Technical Information”):

Log Data: Information that your browser automatically sends when you use our Services. Log data includes your Internet Protocol address, browser type and settings, the date and time of your request, and how you interact with our website.

Usage Data: We may automatically collect information about your use of the Services, such as the types of content that you view or engage with, the features you use and the actions you take, as well as your time zone, country, the dates and times of access, user agent and version, type of computer or mobile device, and your computer connection.

Device Information: Includes name of the device, operating system, device identifiers, and browser you are using. Information collected may depend on the type of device you use and its settings.

Cookies: We use cookies to operate and administer our Services, and improve your experience. A “cookie” is a piece of information sent to your browser by a website you visit. You can set your browser to accept all cookies, to reject all cookies, or to notify you whenever a cookie is offered so that you can decide each time whether to accept it. However, refusing a cookie may in some cases preclude you from using, or negatively affect the display or function of, a website or certain areas or features of a website. For more details on cookies, please visit All About Cookies.

Analytics: We may use a variety of online analytics products that use cookies to help us analyze how users use our Services and enhance your experience when you use the Services.

How we use personal information

We may use Personal Information for the following purposes:
To provide, administer, maintain and/or analyze the Services;
To improve our Services and conduct research;
To communicate with you;
To develop new programs and services;
To prevent fraud, criminal activity, or misuses of our Services, and to protect the security of our IT systems, architecture, and networks;

To carry out business transfers; and
To comply with legal obligations and legal process and to protect our rights, privacy, safety, or property, and/or that of our affiliates, you, or other third parties.

Aggregated or De-Identified Information. We may aggregate or de-identify Personal Information so that it may no longer be used to identify you and use such information to analyze the effectiveness of our Services, to improve and add features to our Services, to conduct research and for other similar purposes. In addition, from time to time, we may analyze the general behavior and characteristics of users of our Services and share aggregated information like general user statistics with third parties, publish such aggregated information or make such aggregated information generally available. We may collect aggregated information through the Services, through cookies, and through other means described in this Privacy Policy. We will maintain and use de-identified information in anonymous or de-identified form and we will not attempt to reidentify the information, unless required by law.

As noted above, we may use Content you provide us to improve our Services, for example to train the models that power ChatGPT. See here for instructions on how you can opt out of our use of your Content to train our models.

3. Disclosure of personal information

In certain circumstances we may provide your Personal Information to third parties without further notice to you, unless required by the law:

Vendors and Service Providers: To assist us in meeting business operations needs and to perform certain services and functions, we may provide Personal Information to vendors and service providers, including providers of hosting services, cloud services, and other information technology services providers, email communication software, and web analytics services, among others. Pursuant to our instructions, these parties will access, process, or store Personal Information only in the course of performing their duties to us.

Business Transfers: If we are involved in strategic transactions, reorganization, bankruptcy, receivership, or transition of service to another provider (collectively, a “Transaction”), your Personal Information and other information may be disclosed in the diligence process with counterparties and others assisting with the Transaction and transferred to a successor or affiliate as part of that Transaction along with other assets.

Legal Requirements: We may share your Personal Information, including information about your interaction with our Services, with government authorities, industry peers, or other third parties (i) if required to do so by law or in the good faith belief that such action is necessary to comply with a legal obligation, (ii) to protect and defend our rights or property, (iii) if we determine, in our sole discretion, that there is a violation of our terms, policies, or the law; (iv) to detect or prevent fraud or other illegal activity; (v) to protect the safety, security, and integrity of our products, employees, or users, or the public, or (vi) to protect against legal liability.

Affiliates: We may disclose Personal Information to our affiliates, meaning an entity that controls, is controlled by, or is under common control with OpenAI. Our affiliates may use the Personal Information we share in a manner consistent with this Privacy Policy.

Figure D.1- Current ChatGPT-4 term of use (January 2024).

Current legal Issues [39]. Because of different regulations in the country, at this moment in the US, two class actions were filed against OpenAI, one mainly focused on alleged data breach and only based on alleged copyright infringements. To complete this picture, we need to consider that in the US the US Federal Trade Commission (“FTC”) has opened an investigation into OpenAI aimed at verifying whether it has violated US consumer protection law. In the UK the High Court of

Justice of England and Wales is dealing with a copyright case between Getty Images (US) Inc. and others v. Stability Al Ltd.

On February 12, 2024, OpenAI obtain a partial victory [42] in dispute with 2 high-profile authors copyright class action.

On February 29, 2024, The Intercept, Raw Story and AlterNet sue OpenAI for copyright infringement [43].