Tunisian republic
Ministry of Higher Education
and scientific research
University of Tunis El Manar
Higher Institute of Computer Science

Graduation Internship Report

Presented to give an account of my experience as part of my compulsory summer internship.

By

Ben said Joumene

Leveraging Prompt Engineering for AI-Driven Development

professional supervisor : **Tounsi Mouhamed** Project Manager

Realized within VERMEG

Academic Year 2023 - 2024

Tunisian Republic

Ministry of Higher Education and Scientific Research

University of Tunis El Manar

Higher Institute of Computer Science of El Manar

Final Year Project Report

Presented for the award of the

National Engineer's Degree in Applied and Technological Sciences

Specialization : Software Engineering and Information Systems

By

Ben Said Joumene

Leveraging Prompt Engineering for AI-Driven Development

Professional Supervisor : **Tounsi Mouhamed** Project Manager

Realized within VERMEG

Academic Year 2023 - 2024

I authorize the student to submit his or her internship report for oral presentation.

Professional Superivisor, **Mr Tounsi Mouhamed**

Signature

Dedications

I dedicate this work to :

My beloved mother For her love and support. To all my friends, the ones I value and appreciate so much. To my extended family and all the professors who helped me so much throughout my schooling. It is with deep gratitude and deep respect that I dedicate this modest work to my beloved mother, for her love, care and dedication. she has always believed in me, supported and guided me. I also want to dedicate this project to my sister, for all her advice, affection and encouragements . my brother and all my friends who always gave me a hand whenever I needed it and to all my teachers during my educational journey. To anyone who has engraved my life by a word that has oriented me to the good way etc.

Ben said Joumene

Acknowledgment

First of all, I would like to thank "GOD". Before starting any development, it seems appropriate to begin with sincere thanks to the people who have been kind enough to participate in this effect from near or far. I would like to express my immense gratitude to my Professor Ms.Haouari Bakhta .It was a real privilege and honor for me to share her exceptional knowledge but also of her extraordinary human qualities would like to thank her for her supportive and helpful advice throughout the duration of this project. I would also like to express our deepest respects and gratitudes to the management of The Higher Institute of Computer Science of Ariana .I address a very particular thanks to Mrs.Imen Ben hafaiedh. Finally, I wish to thank my friends for their support and encouragement throughout our study

Table des matières

Table des figures

Liste des tableaux

List of abbreviations

— **COT** = **C**hain **O**f **T**hough

— **LLM** = **L**arge**L**anguage **M**odels

— **NLP** = **N**atural **L**anguage **P**rocessing

General Introduction

AI is the primary driver of change in the evolving digital landscape of software development. Vermeg leads such an example of evolution. In aligning with new technologies in their development processes, like ChatGPT and GitHub Copilot, Vermeg sought to strengthen its processes in order to take projects to the best possible. This will not only work for optimizing the workflows but also help in productivity improvement but imagine an intelligent bot as a coding buddy that could provide hints, ideas, and possible solutions just in one instance. That is the beauty of both ChatGPT and GitHub Copilot, after all. Both have really revolutionized the development process in enabling teams to spend more time on what truly matters-to develop and deliver high-quality software that will meet client requirements or customer expectations. But the magic really begins when you learn the art or science of how to craft the right prompts that elicit the best results from AI tools.
Having understood the importance of the field, Vermeg tasked me with a research and development project that included an evaluation of the best strategy in Prompt Engineering. This internship is meant to identify which of these techniques can be applied to web development within the frameworks of Spring Boot and Angular. During this project, I go in-depth into exploring various techniques that would make interfacing the tools easier for the best outcome.

This report sought to bring out practical views on how Vermeg can unlock the potential of AI for development by adopting suitable prompt engineering strategies that will open up the future growth and success of the company. The report has six well-structured chapters, each serving its purpose. Chapter 1 covers the host organization, Vermeg, and the scope of the project ; in addition, it sets up the problem statement and outlines the solution with the use of AI tools and prompt engineering. Chapter 2 provides background to apply generative AI technologies, including ChatGPT, GitHub Copilot, and prompt engineering in software development. Chapter 3 presents some of the techniques used in prompt engineering, such as Zero-Shot, Few-Shot, and Chain-of-Thought prompting, and gives their use cases. Chapter 4 describes the implementation and testing of a web application using Spring Boot and Angular. This chapter focuses on the practical assessment of different prompt engineering techniques. Chapter 5 covers the actual usage of providing some tip and insight into how some of the features of GitHub CoPilot can be used for development.

Chapitre 1

General Context

Plan

Introduction

In this chapter, we will have an overview of the project. First, we will present the host organization followed by the main goal of this RD internship which involves identifying the problem and outlining the research approach to solve it. This chapter concludes with a definition of the Scrum methodology used throughout the project life cycle.

1.1 The Host Organization

Vermeg is a software provider specializing in the financial and insurance sectors. the company was founded in 1993 in Amsterdam,. today it operates in over 40 countries serving over than 550 clients globally. Vermeg focuses on banking, capital markets, asset management and digital financial services. Its products help enhance operational efficiency and ensure compliance for clients in these industries. Vermeg has a focus on innovation solutions to tackle the ever changing challenge that financial institutions face, to help them improve their customer service delivery and ensuring they meet their needs.[1]

Figure 1.1 : The Vermeg [1].

1.1.1 History and Positioning

Vermeg is a recognized company in the field of information technology, providing software solutions for the financial sector. Since its founding, Vermeg has established itself as a renowned player, particularly due to its innovative approach and its ability to adapt its solutions to the ever-changing needs of its clients.[1]

1.1.2 Business Sectors and Services

Vermeg offers solutions in several sectors :

— **Insurance :** Management of policies and claims.

— **Asset Management :** Tools for financial investments.

— **Banking Services :** Management of transactions and customer accounts. The company stands out for its ability to integrate emerging technologies, such as artificial intelligence, into its software solutions.

Vermeg is structured into four specialized business lines, including the REG department, which plays a crucial role in ensuring the regulatory compliance of financial institutions. [1]

1.2 Project Scope

In this section we will present the main problem and our approach to solve this problem.

1.2.1 Study of the Problem

Vermeg started using ChatGPT and GitHub Copilot to speed up and automate its application development workflow and allow developers to focus on more complex challenges.
While these two tools are very powerful they struggle to return accurate results when the tasks passed and code are quite complex especially github copilot.
While these tools are powerful in generating code snippets and providing contextual suggestions, they exhibit limitations when dealing with highly complex or ambiguous tasks. Especially github copilot .

1.2.1.1 Proposed Solution

To address the challenges encountered my primary goal is to conduct research on AI-driven development and prompt engineering.
This research is composed of two parts, investigating different prompt engineering techniques and developing an entire web application using these techniques to evaluate their efficiency in web development with Spring Boot and Angular, utilizing ChatGPT and GitHub Copilot.
By systematically exploring these methods, I aim to optimize the integration of AI tools into the development process, ultimately improving the quality and efficiency of outcomes.

1.3 Development methodology

In this section we will define the Scrum method and it's different roles.

1.3.1 Scrum Definition

Scrum is a framework for responding to complex and changing problems, while productively and creatively delivering products of the highest possible value.
The Scrum framework enables you to work as a team to achieve continuous improvement through iterative, incremental delivery of products to satisfy your customers. Scrum is based on the theory of empirical process control and supported by 3 fundamental pillars :

— **transparency** : being honest, having nothing to hide, working together for the success of the product/project by making important aspects of the process visible to all those responsible for the results.

— **inspection :** being able to help each other and inspect Scrum artifacts and progress against a Sprint Goal to detect undesirable deviations.

— **Adaptation :** adapting to changes in general, product changes, changes in the way things are done. [2] The figure 1.2 represent Scrum metholod different steps.

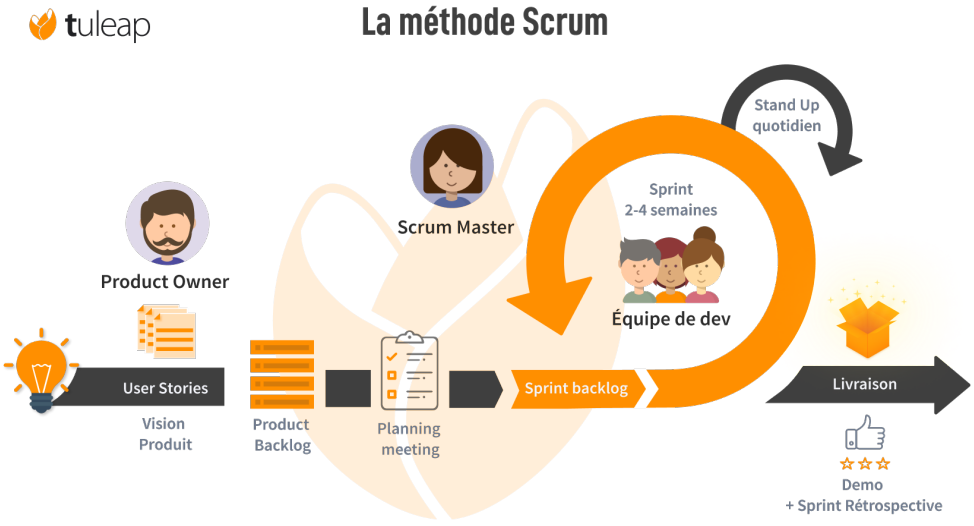

Figure 1.2 : The Scrum Method [2].

1.3.2 Roles

— **The Scrum Master**

According to the Scrum Master guide, the Scrum Master is a servant-leader for the team, the guarantor of the application of the approach.He is not, however, a project manager.

— **The Product Owner**

The role of the Product Owner is to act as a bridge between the business and technical sides of the project. He's the link between the customer and the development team.

— **The development team**

The development team is responsible for transforming expressed needs into usable functionalities.[2]

Conclusion

In this chapter we presented the problematic faced and solution proposed which focuses on prompt engineering . In the next chapter we will define the various technologies used in this research which revolve around AI-driven development.

Chapitre 2

Generative AI State-of-the-Art

Plan

Introduction

This chapter we will explore how generative AI is transforming software development , how tools like GitHub Copilot and ChatGPT enhance workflows by automating repetitive tasks, allowing developers to focus on complex challenges. We also introduce prompt engineering, a technique to improve AI outputs. These concepts prepare us for the upcoming chapters where we dive into real-world applications.

2.1 AI-Driven Development

In this section we will define AI-Driven and its different components

2.1.1 Definition

AI-driven development consists of integrating artificial intelligence into the software development life cycle. It empowers developers by reducing manual tasks, improving accuracy, and streamlining workflows, leaving AI to handle repetitive and data-driven tasks.

2.1.2 Key Components

— **Artificial Intelligence (AI) :** AI encompasses various technologies like machine learning and generative AI.

— **Machine Learning :** Machine learning enables systems to improve through experience without explicit programming.

— **Generative AI :** Generative AI focuses on creating new content, such as text, images, or music, by finding patterns in data.

— **Large Language Models (LLMs) :** LLM ssuch as ChatGPT, represent a breakthrough in generating human-like responses and text.

— **Natural Language Processing (NLP) :** NLP is a key subfield of AI it allows computers to understand and generate human language. ChatGPT exemplifies this with its ability to mimic human conversations and assist with tasks like coding and writing.

2.2 GitHub Copilot

In this section we will define GitHub Copilot and present the main Functionalities it offers.

2.2.1 Definition

GitHub Copilot is an AI-powered coding assistant created by GitHub and OpenAI. It provides real-time code suggestions, completes code blocks, and integrates perfectly. with IDEs like Visual Studio Code. Powered by OpenAI's Codex model, Copilot analyzes code, Comments, and project structure to provide contextually relevant suggestions. Supported languages include Python, JavaScript, TypeScript ; automates routine tasks in a uniform manner. Which eventually enables the developers to target complex works.
[3]

Figure 2.1 : Github Copilot .

2.2.2 Functionalities

Following are the major features of GitHub Copilot :

- **Real-time code suggestions :** Suggest lines of code or entire blocks based on what you type.
- **Auto-completion :** Automatically finishes off functions or code constructs.
- **Multi-language compatibility :** Works with multiple languages like Python, JavaScript, TypeScript, and many others.
- **Context analysis :** Grasps code, comments, and the project to deliver relevant suggestions.

- **Automation of repetitive tasks :** This saves time used on mundane tasks by quickly generating standard code.
- **Documentation support :** It can generate comments to explain code or functions.
- **Seamless integration :** Works with popular IDEs like Visual Studio Code.

These features will save the developers' time and let them focus on the tasks at hand.**ref3** [3]

2.3 ChatGPT

In this section we will define GhatGPT Copilot and present the main Functionalities it offers.

2.3.1 Definition

ChatGPT, based on OpenAI's GPT architecture, generates human-like text and supports tasks Like answering questions, writing, and coding. It is good at brainstorming, summarizing, and providing explanations. ChatGPT processes large-sized datasets to generate coherent responses by predicting the next word in a sequence..
[4]

Figure 2.2 : ChatGPT

2.3.2 Functionalities

ChatGPT generates understandable responses by predicting words in sequence, hence processing big data. Its use spans coding, creative writing, debugging, and describing complex ideas. It also integrates into APIs for automating interactions, so it is a versatile development tool.
But ChatGPT has also evolved as a series of technological enhancements are poured out for use in making human jobs easier. The additional improvements and enhancements have increased a good deal, thus allowing it to return answers precisely to each request in order to make user

interactions even smoother.

.[4]

Here are just a few of the recent enhancements :

— **Improved content generation**

— **Ability to understand conversational context**

— **Improved response accuracy**

— **Introduction of new features and plugins**

— **Enhanced user interface**

[4]

2.4 Prompt Engineering

Prompt engineering is the domain that deals with designing and perfecting prompts to guide AI models effectively to come up with the best output. Writing a well-structured, clearer prompts mean better results with AI.Different techniques apply only to specific use cases.

Conclusion

This chapter highlighted such tools as GitHub Copilot and ChatGPT while showing their contribution in AI-driven development, and we introduced prompt engineering, a key technique for Optimizing AI output. The next chapter is dedicated to testing these ideas by developing areal-world application with Spring Boot and Angular

Chapitre 3

Prompt engineering Techniques

Plan

Introduction

In this chapter, we will outline a range of prompt engineering techniques that we have collected from various research studies and findings, which served as the foundation for our work. Also we'll take a closer look at prompt engineering techniques rules for creating effective prompts. The goal to craft prompts that get accurate and useful results from AI.

3.1 Zero-Shot Prompting

The zero-shot Prompting technique consists of directly passing the task in the prompt without any additional examples or context to guide it. The model performs based on its general knowledge

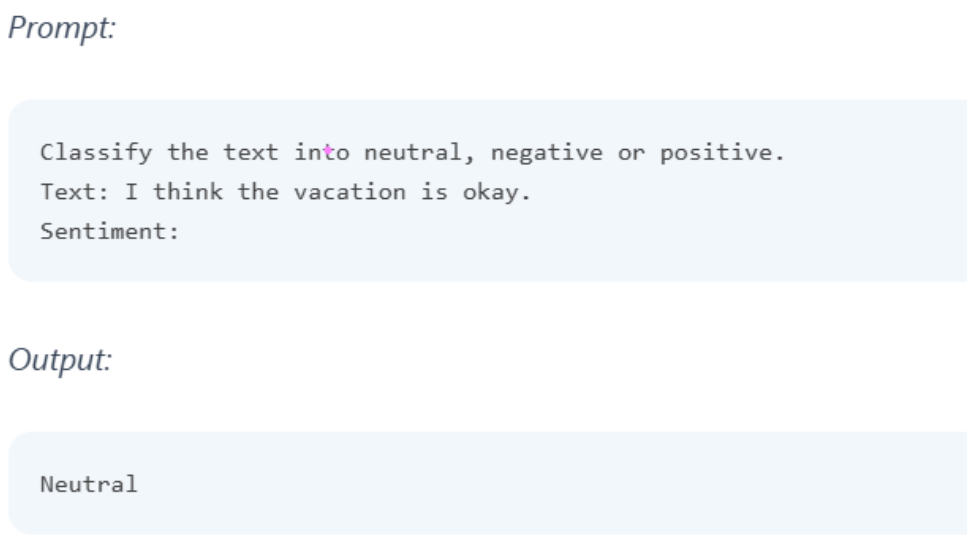

Figure 3.1 : Zero-Shot Prompting Example

3.2 Few-Shot Prompting

Few-shot prompting is a technique used for more complex tasks. A few examples are passed off the task usually in the form of input-output pairs within the prompt to help the model understand the desired output format or behaviour for handling similar requests. This technique uses in-context learning meaning the AI generalizes and learns patterns from the provided examples unlike the traditional machine learning model that needs additional training on a

dataset in-context learning allows the model to adapt to new tasks during runtime without modifying its underlying weights. By analyzing the examples in the context of the prompt the model can produce outputs that align with the demonstrated patterns [5].

These are two examples showcasing the use of few-shot prompting :

Prompt:

```
A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle
is:
```

Output:

```
When we won the game, we all started to farduddle in celebration.
```

Figure 3.2 : few-Shot Prompting Example1

Prompt:

```
This is awesome! // Negative
This is bad! // Positive
Wow that movie was rad! //Positive
What a horrible show! //
```

Output:

```
Negative
```

Figure 3.3 : few-Shot Prompting Example2

3.3 Chain-of-Thought (CoT) Prompting

This technique starts similarly to few-shot prompting by providing examples of input-output pairs to the language model but the output also includes the reasoning process used to arrive at the desired result it's the steps required to get to the output This reasoning process is called complex reasoning which encourages the model to think step by step [6]

The image Below is a CoT prompt example.

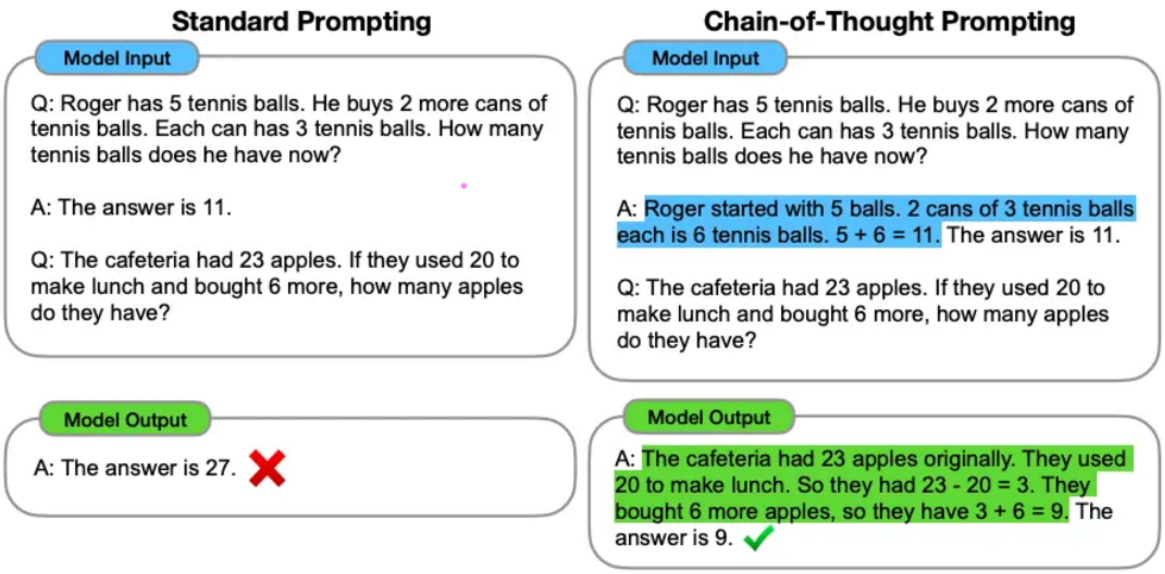

Figure 3.4 : Chain-of-Thought (CoT) Prompting Example

3.4 Zero Chain-of-Thought (CoT) Prompting

Similar to Chain-of-Thought (CoT) prompting, zero-shot Chain-of-Thought prompting encourages the language model to think by explaining its reasoning. This technique involves adding phrases like "Let's think step by step" to the original prompt [6].

The image Below is a zero CoT prompt example.

Prompt:

```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

Output:

```
Neutral
```

Figure 3.5 : Chain-of-Thought (CoT) Prompting Example

3.5 Self-consistency

Simply asking a model the same question more than once and accepting the majority response as the final response is known as self-consistency. It is a follow-up to CoT prompting, and when combined, it has greater impact.

It has been demonstrated that self-consistency enhances performance on tasks involving math, common sense, and symbolic reasoning. Self-consistency was found to be able to enhance outcomes even in cases when standard CoT was determined to be ineffective [7]. The Image below is a detailed example of a well structured Self-consistency prompt.

Prompt:

```
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done,
there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted.
So, they must have planted 21 - 15 = 6 trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74
chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops
did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of
lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does
he have now?
A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then he got 2 more from dad, so
in total he has 7 + 2 = 9 toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from
monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total 4 * 5 =
20 computers were added. There were 9 computers in the beginning, so now there are 9 + 20 = 29 computers.
The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many
golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has 58 - 23 = 35 balls. On
Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: She bought 5 bagels for $3 each. This means she spent $15. She has $8 left.
Q: When I was 6 my sister was half my age. Now I'm 70 how old is my sister?
A:
```

Figure 3.6 : self-consistency prompt input

Output 1:

```
When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is 70 - 3 = 67. The answer is 67.
```

Output 2:

```
When the narrator was 6, his sister was half his age, which is 3. Now that the narrator is 70, his sister would
be 70 - 3 = 67 years old. The answer is 67.
```

Output 3:

```
When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is 70/2 = 35. The answer is 35.
```

Figure 3.7 : self-consistency prompt output

3.6 ReAct prompt

This method guides the LLMs to generate the best result by an iterative approach each iteration is composed of two steps reasoning step and an action step explaining the react Re for reasoning and act for action The reasoning and action taken in the next iteration depend only on the observation of the current action so the model takes actions and modifies its subsequent action depending on the observation of the results The model improves its responses with each iteration based on real-time observations. This approach enables the model to communicate with outside sources therefore producing a more correct and accurate response [8].

he image below is an example of a ReAct prompt illustrating the iterative process of thinking, acting, and observing.

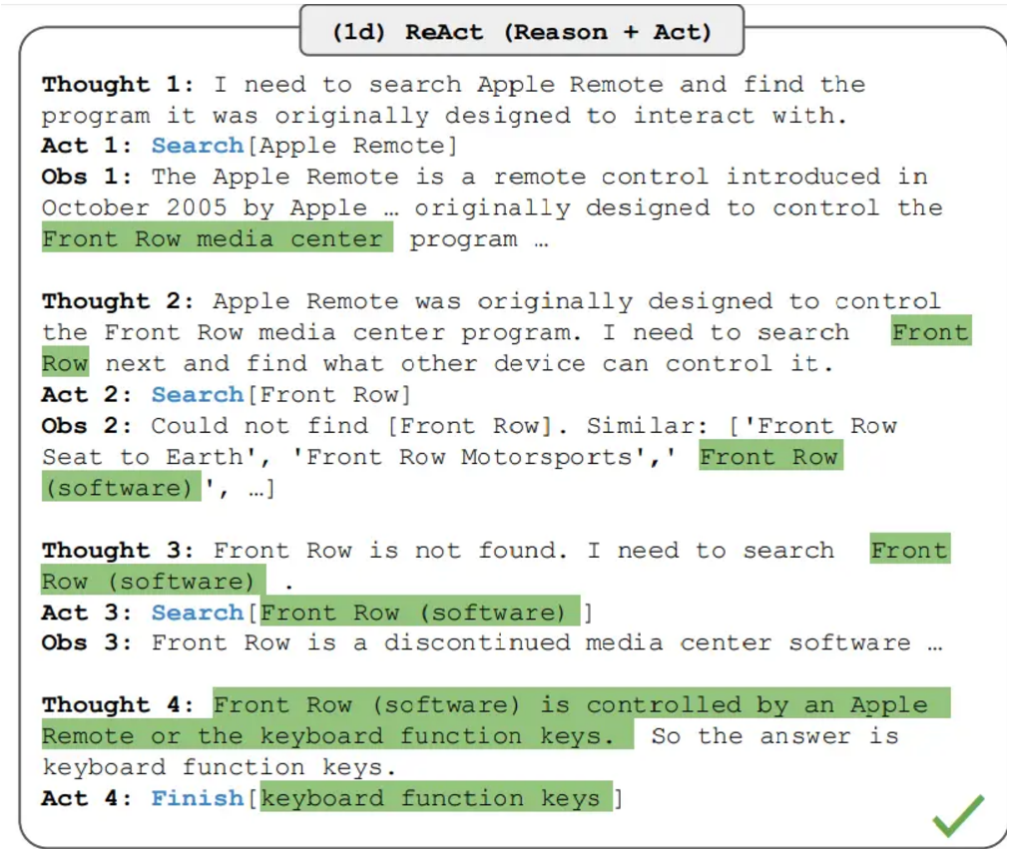

Figure 3.8 : React prompt example

3.7 Reflexion prompting

Reflexion prompting is a mechanism to enhance the output of agents that are language-based through reflective feedback structures and mechanisms of learning from interactions and experiences.

This draws primarily from a paradigm proposed in the work by Shinn et al where linguistic feedback whether in free form or scala is channeled into meaningful reinforcement that guides the agent in refining its decision-making processes [9].

The figure 3.9 is the architecture of a reflexion based model

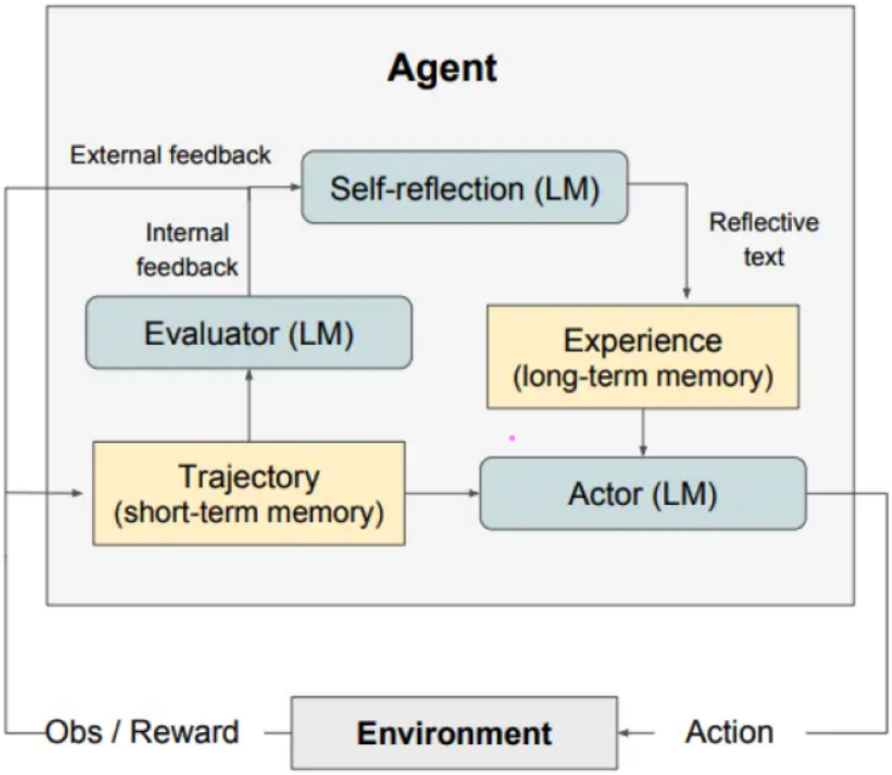

Figure 3.9 : Reflexion Model Architecture

3.7.1 Key Components of Reflexion Prompting

— Actor : This component generates actions on the basis of the agent's observations. It uses frameworks such as Chain-of-Thought (CoT) and ReAct to generate responses. The actor's actions and observations, recorded as a trajectory, are essentially the experience of the agent in carrying out a task.

— Evaluator : It scores the actor's actions. It scores the outcomes of the generated trajectories and tells the agent how well it has performed those actions. A specific example is that this scoring can take many different forms, depending on the specific task at hand, because there may be many different notions of how good the actions are.

— Self-Reflection : This generates feedback that should spur self-improvement. It takes into consideration the rewards received and the trajectory of actions, which therefore provides a basis for understanding what future tasks may demand. Through reflecting on past experiences and their outcomes, an agent can adjust its strategies toward enhancing its future performance in subsequent trials.

3.7.2 Process Overview :

The steps followed in general in the Reflexion process are as follows : Define the task.

— Generate an action trajectory.

— Evaluate the results of the actions taken.

— Reflect on the performance using the feedback generated.

— Generate the next trajectory based on insights from the reflection.

Conclusion

In this chapter we presented a wide range of prompt engineering technique in the next chapter we will test the effeminacy of these technique for developing a web application with GitHub Copilot.

Chapitre 4

Evaluating Prompt Engineering Techniques through Web Application Development

Plan

Introduction

In this chapter we will discuss the testing phase of this project, during which we tested different prompt engineering techniques by implementing a web application.
We will present the application's goal and architecture, as well as the class diagram.
Our focus will primarily be on the GitHub Copilot testing phase.

4.1 Purpose of the Application

For this project, I built a data managing system application for managing 3 data entities **Jurisdiction**, **Returns**, and **References** these tables are interconnected through one-to-many relationships. The web application was developed using spring boot for the back-end and angular for the front-end

The application provides a dashboard to perform CRUD Operations including Create, Read, Update, and Delete it also ensures seamless data navigation and integrity.

— Jurisdiction Table :

Represents legal or geographical entities such as states, countries or regions.

— Ret Table for returns :

Serves as the central table connecting jurisdictions to their associated records or transactions. It acts as a bridge between Jurisdiction and references.

— Ref Table for references :

Contains reference or supplementary data related to the entries in the Ret table.

The web application was developed using **Spring Boot** for the back-end and **Angular** for the front-end.

The application provides a **dashboard** to perform **CRUD operations** (Create, Read, Update, Delete) while ensuring seamless data navigation and integrity.

4.2 Application Features

— **Dashboard** :A centralized interface composed of separate pages for managing the different tables.

— **Data Relationships** : Related entities are automatically linked for example once selecting a return you can check the references related to it.

— **CRUD Operations** : For adding, updating and deleting data entries for all three entities.

— **CSV Import** : Upload functionality to batch-import data for any of the tables converting rows into table entries.

— **Search and Filter** : Search and filter options based on the entity's name or Id.

— **API Communication** : RESTful APIs for seamless frontend-backend communication.

4.3 Technologies Used

In this section we will present the different technologies and frameworks used for creating the web application.

4.3.1 Spring Boot

— The Java Spring Framework is a popular open-source, enterprise-level framework for building stand-alone production applications that run on the Java Virtual Machine (JVM).

— Spring Boot a tool built on the Spring Framework simplifies the development of web applications and microservices through three core features :

autoconfiguration : an opinionated approach to configuration and the ability to create stand-alone applications.

These capabilities enable developers to set up Spring-based applications with minimal configuration,and Spring Boot applications can also be optimized to run on the Open Liberty runtime. features like automatic server configuration and security management [10].

Figure 4.1 : Spring Boot

4.3.2 Angular

— Angular is a development platform built on TypeScript .It provides a component-based framework for building scalable web applications a collection of well-integrated libraries for features like routing, forms management ,client-server communication and a suite of developer tools for developing, building, testing and updating code. Designed to scale from single developer projects to enterprise level applications.

— Angular is supported by 1.7 million ecosystem from developers, library authors and content creators [11]. [11].

Figure 4.2 : Angular

4.3.3 MySQL

— MySQL is a widely used open-sourced database management system that helps organize and manage data using Structured Query Language (SQL). Known for being fast, reliable and user-friendly. It's a popular choice for web applications and businesses of all sizes. MySQL allows multiple users to access and work with data simultaneously and it's designed to handle everything from small projects to large, complex databases. Its flexibility and compatibility with different operating systems and programming languages making it a go-to option for developers around the world [12].

Figure 4.3 : MySQL

4.4 Design and Development Process

In this section we will present the Application architecture and the relation between the different items as well as the Class diagram

4.4.1 Application Architecture

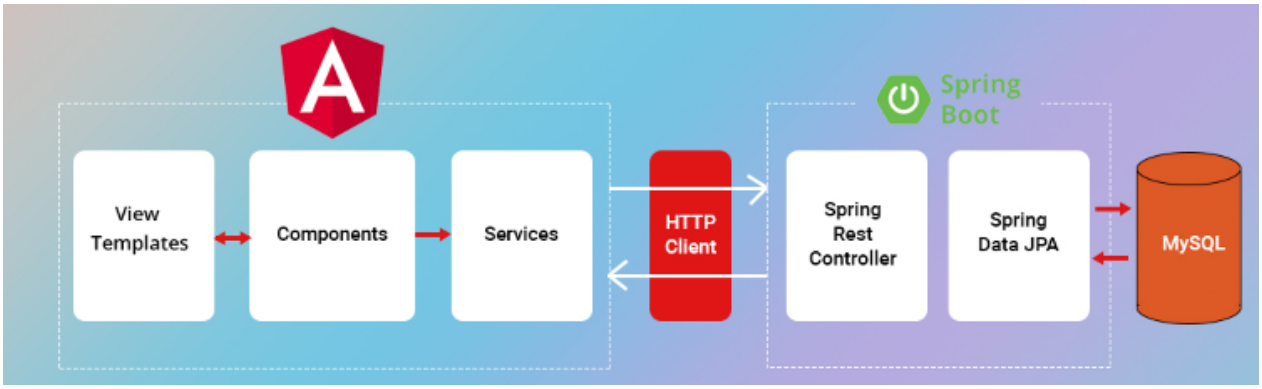

Figure 4.4 : Application Architecture

The above image represent the architecture of our application and how the different front end and backend components are connected and communicate

The architecture of the web application uses a separation of concerns there are 3 main components front end and backend .

The backend is responsible for managing business logic, data persistence, and API exposure. It has three key components :

— Controller Layer : Handles HTTP requests and maps them to appropriate service methods.

— Service Layer : This layer is responsible for the business logic, processing data and interacting.

— Repository Layer :This layer handles the interaction with the database. The Spring Data JPA is used to interact with the database to execute the CRUD operation for Jurisdiction, Ret, and Ref entities.

The front-end is developed using Angular a framework that allows the creation of dynamic components. The different components interact with the backend to fetch, display and manipulate data.

Front-end handles the UI for data tables and forms display, and also provides the possibility to import.csv files for bulk data entry.

The data models in this application are the Jurisdiction, Ref and Ref entities. These models are linked through one-to-many relationships. Each entity is mapped to a corresponding table in the database using JPA annotations.

The relationships are configured such that the Ret entity has a foreign key reference to Jurisdiction, and the Ref entity has a foreign key reference to Ret. This architecture ensures that the application can perform efficient queries and manage data integrity while supporting the necessary CRUD operations.

4.4.2 Data Models and Relationships

The application has three main data models : **Jurisdiction**, **Returns (Ret)**, and **References (Ref)**. These models are linked via one-to-many relationships :

— **Returns** has a foreign key referencing **Jurisdiction**.

— **References** has a foreign key referencing **Returns**.

This setup ensures efficient queries and data integrity while supporting CRUD operations.

4.4.3 Class diagram

The Image below represent the Class Diagram of our application

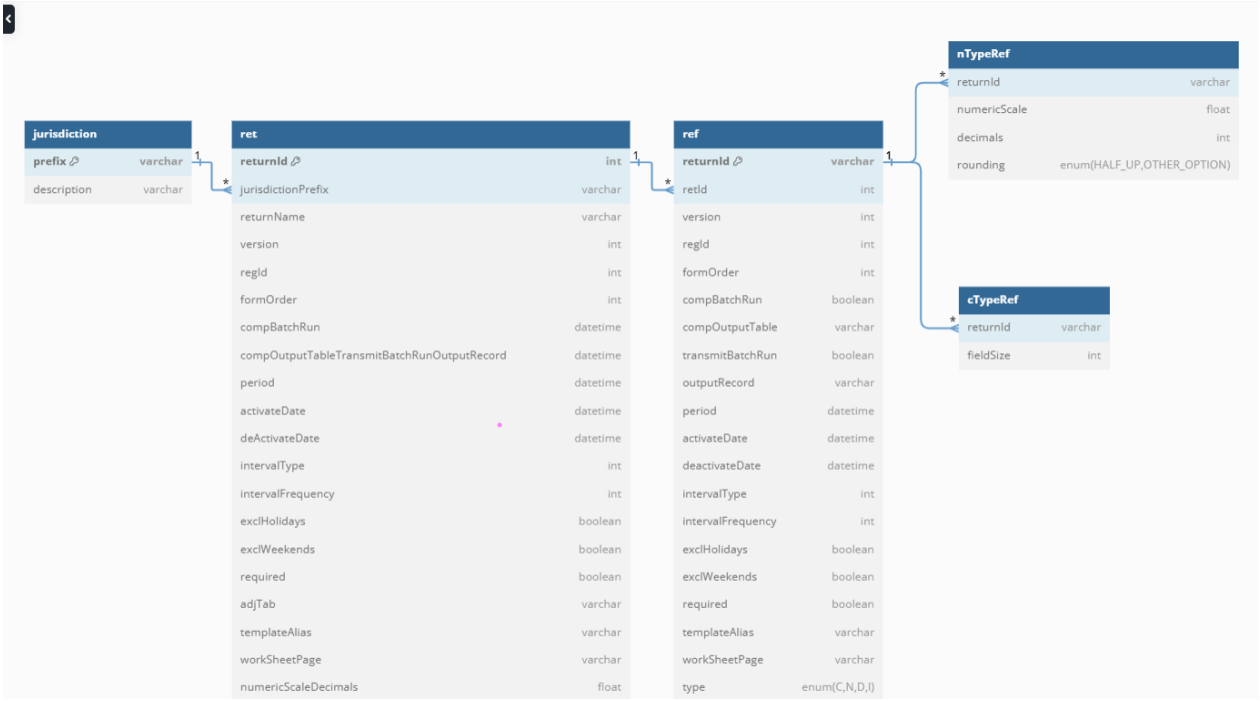

Figure 4.5 : Class Diagram

4.5 Role of Prompt Engineering in Development

Prompt engineering played a significant role in this project. **GitHub Copilot** and **ChatGPT** were utilized for generating code :

— **Backend Code Generation** : Prompts were used to produce Spring Boot REST API controllers, service classes, repositories, and entity classes. Github copilot had provisioned with structured templates which minimized efforts in manually coding.

— **Frontend Development** : In the preparation of Angular components, routing and user interface, chatGPT provided assistance which smoothed the process between backend to frontend.

4.6 Prompt engineering with GitHub Copilot

In this section we will showcase examples of various prompt engineering techniques applied with GitHub Copilot.Most of the examples highlight the development of unit tests and integration tests. . While several techniques were also used during the backend development process we only mentioned those implemented for the unit testing.

4.6.1 Creating Repository Unit Tests Using Role-Playing Technique

— Tests Created : Ref, Ret, and Jurisdiction tests.

— Prompts Used : One with the persona pattern and one without.

GitHub Copilot generated an entire suite of tests covering all CRUD operations without needing specific instructions within the prompts.

— Evaluation metrics :

Efficiency : The entire process took only 5 seconds. Code Quality : The generated code was very clean and efficient. Prompt Requirements : Only one prompt was needed to get a good result, though an additional prompt was required for comments.
while the persona pattern did not influence the outcome, GitHub Copilot demonstrated its ability to generate high-quality, comprehensive unit tests rapidly with minimal prompting.

4.6.2 Creating Integration Tests

To evaluate GitHub Copilot's ability to understand the relationships between different entities, I asked it to generate integration tests without passing relationship details in the prompt.
Three integration tests were created :

— Inheritance Test : Tested the inheritance between Ref, NRef, and CRef (with CRef and NRef extending Ref).

— One-to-Many Relation Test (Jurisdiction and Rets) : Tested the one-to-many relationship between Jurisdiction and Rets.

— One-to-Many Relation Test (Rets and Refs) : Tested the one-to-many relationship between Rets and Refs.

The table below Showcase the different results obtained during this test :

Tableau 4.1 : Summary of Relationship Analysis Results

Metric	Without Relationship Details	With Relationship Details
Inheritance Test	Failed to understand relationships	Generated efficient test
Jurisdiction and Rets	Failed to understand relationships	Generated efficient test
Rets and Refs	Failed to understand relationships	Generated efficient test

— **First Test :** Without mentioning the relationship, GitHub Copilot failed to determine the relationship between Ref, NRef, and CRef, resulting in significant hallucinations.

— **Second Test** : Once I passed the relationship details in the prompt, it generated a very efficient test.

4.7 Creating service

To evaluate the copilot's ability to learn and improve its output i employed different prompt engineering approaches :

— Ask for Improvement : For each output, I asked Copilot if there were any further improvements it could suggest.

— Reflection : Used for error detection. I also explored improvements unrelated to errors.

— Question Refinement : Whenever I asked a question I requested Copilot to suggest a better version of the question that would help it provide a more accurate response.

— Alternative Approaches : Whenever I asked for a solution to a problem I also asked Copilot to provide at two alternative approaches and compare their advantages and disadvantages.

Before each prompt I asked Copilot to generate the necessary components for a service and then create the service class.

These Are the different metrics i focus on for each part :

— All CRUD and filter methods in the repository

— Exception handling

— Logging statements

— Comments

— Transaction management

— Creativity

4.7.1 Reflection

Reflexion Technique was used for Ret service Creation.

First Prompt :

“After generating a response, review it and list any potential errors or areas for improvement“

By Following the pattern of the first prompt I took the suggested improvements from the generated response and used them as input for a second prompt asking the same question.

The second prompt

"Apply the improvements you mentioned, then review your response and list any potential errors or further improvements."

The desired output was obtained after 3 iterations making it 4 prompt in total.

4.7.2 Question Refinement Analysis

GitHub Copilot was used to generate the Ref class. This class represent a challenge to GitHub Copilot with it’s complex structure .The main issue noticed is that GitHub Copilot does not maintain conversational context effectively.

— **Initial Prompt :** The first prompt was :

"Whenever I ask a question, suggest a better version of the question that would help you provide a more accurate answer."

The response was an acknowledgment without actionable improvement.

— **Second Prompt :** Asked Copilot :

"create a ref service," but it ignored the initial refinement request and generated a basic CRUD service."

— **Third Prompt :** Refined the question again to :

"Can you generate a service class for the ref entity ?

This prompt yielded a service with CRUD operations but lacked additional desired features.

— **Fourth Prompt :** To further improve, the prompt was :

"Can you generate a service with all CRUD operations ?"

This time, the output included exceptions and comments.

It took a total of three refined prompts to achieve the desired result, highlighting Copilot’s limited ability to consider previous requests. These results demonstrate that GitHub Copilot has limited context persistence, as it failed to effectively incorporate the initial request into subsequent responses. It required multiple refinements to generate the desired code.

4.8 Result Analysis

It became strikingly clear during the testing period that GitHub Copilot falls drastically short in maintaining and following conversational context. It would typically not merge any dwelling details or clarifications requested from earlier prompts into subsequent responses when writing code. For example, it demanded multiple refinement prompts for creating a service class until the desired output was achieved because it could not recall or apply the context from earlier prompts. This lack of context persistence shines a spotlight on the difficulty of exploiting advanced prompt engineering techniques with GitHub Copilot.

Conclusion

One thing that became very apparent during the testing period is the fact that GitHub Copilot does not fare very well at all in keeping a flow and context of the conversation. Details that were requested with it previously, for instance dwelling details or clarifications from earlier prompts, are not merged into subsequent responses when it has to write code. For instance, with the prompt to create a service class, it required multiple refinements in prompting to finally get what was desired because there and then, it was impossible for it to remember or apply the context from earlier prompts. This inability to maintain context is actually indicative of the difficulty one would have in exploiting advanced prompt engineering techniques with GitHub Copilot.

Chapitre 5

Practical Use of GitHub Copilot

Plan

Introduction

In this chapter, we are going to look at the practical use of GitHub Copilot, focusing on its real-time code suggestions, Copilot Chat, and adaptation to coding styles. We also present methods for enhance development efficiency and code quality. Key Features GitHub Copilot is offering real-time code completions based on context and provides ready-to-use code suggestions for the current task. It also learns from the user's habits to adapt to their coding style.

5.1 Key Features

While writing code, Copilot generates context-specific proposals that the developer can accept, modify, or reject. This enables the developers to speed up writing functions, loops, and conditions, seamlessly integrating these suggestions into the ongoing code.

This proactive assistance saves the developers precious time, in particular, on repetitive or complicated tasks, and reduces possible errors. Also, Copilot's suggestions go far beyond simple auto complete ; it provides sophisticated code solutions based on best practices and widely used models in the open-source community.

Practical Use of GitHub Copilot by avoiding the necessity to toggle back and forth between the IDE and other resources. Copilot Chat becomes a real personal assistant, and can explain challenging concepts and debugging code for the right path toward perfect solutions.

5.2 Using Copilot Chat in the IDE

Copilot Chat allows developers to ask questions from within the IDE in order to solve programming issues in real time. It is much more interactive and answers natural language questions about the code being worked on by explaining the proposed solution.

What makes Copilot Chat unique is that it can embed deeply into the developer environment. By capturing the general context of the project, such as active files and related functions, it gives highly relevant and context-specific responses. This feature changes how developers interact with their code by eliminating the need to switch between the IDE and external resources. Copilot Chat becomes a true personal assistant, able to explain complex concepts, debug code, and guide users toward optimal solutions

5.3 Optimizing the Use of GitHub Copilot

5.3.1 Tips and Best Practices for Better Efficiency

Utilizing full capability of github copilot to enable the team to actually take benefit of it following best practises is something that will be required.Here are a few Suggestions :

- **Writing Effective Comments :**
  - **Using Natural Language :** Writing requests in the form of descriptive comments helps improve code generation.
  - **Clarity :** Formulating clear and detailed comments ensures better direction for Copilot's suggestions.
  - **Use Detailed Instructions :** Comments can include details about parameters, return types, and expected behaviors.
- **Evaluate Suggestions :** Carefully review each suggestion before accepting it to ensure it meets the project's needs and adjust it if necessary.
- **Context Integration :** Place comments near relevant code to help Copilot better understand the context.
- **Use Multi-line Comments for Complex Code :** For multi-step functions or complex algorithms, use multi-line comments to guide Copilot in generating more accurate code.
- **Use Examples :** Provide clear examples of input and output to help Copilot understand expectations and generate more precise code suggestions.

5.3.2 Keyboard Shortcuts and Advanced Features

To make the most of GitHub Copilot, using keyboard shortcuts and advanced features is essential :

- **Display Suggestions :** Use **Ctrl + Space** to show Copilot's suggestions in your IDE.
- **Accept a Suggestion :** Tab accepts the highlighted suggestion.
- **Reject a Suggestion :** Press Esc to reject the current suggestion and go into manual editing.
- **Display Alternative Suggestions :** To cycle through alternatives, use Alt + [or Alt +].

5.3.3 Improving Code Quality and Documentation

GitHub Copilot goes a long way in improving both code quality and documentation.

Code Quality :

- **Code Structure :** The suggestions given by Copilot, generally maintaining established coding conventions, improve the readability and consistency of the code.
- **Error Reduction :** By making suggestions based on proven practices, Copilot minimizes errors and enhances the reliability of the code.

Documentation :

- **Automatic Generation :** Copilot can automatically generate comments and function descriptions, enriching your code documentation and making it easier to understand.

Conclusion

GitHub Copilot streamlines development with its powerful features but requires an understanding of the limitations of its use and consideration of different strategies we covered in this chapter to fully exploit its potential.

General Conclusion

The research represents a milestone in the exploitation of artificial intelligence in developing software. Through an in-depth analysis of the potentials of tools like GitHub Copilot and ChatGPT, it was possible to catch glimpses of great potentials and their critical limitations. This work attempts to make maximum use of the techniques put forward by prompt engineering in offering improvements in development efficiency with the smoothing workflows within the production application environment using Spring Boot and Angular.

These results really highlight the transformative role of AI in software development, from automating routine activities to making intelligent code suggestions, or even improving the productivity of developers. The other side of this journey unfolds into the critical challenges that remain : GitHub Copilot is restricted with respect to the context it considers-for example, being consistent and fitting into a complex scenario with many refinements.

From a technical point of view, the development of the complete web application provided quite a real playground for proposed approaches of prompt engineering. Further, this pragmatic approach allowed us to gauge zero-shot and chain-of-thought prompting, reflexion, and ReAct methods against their ability to enhance AI-driven code generation. While certain techniques were very promising, others pointed out further refinements and adaptations that needed to be done for the specific project requirements.

This project has turned out to be a personal eye-opener, with the fast changes in the world of AI-driven development. It further underlined the fact that while working with state-of-the-art technologies, critical thinking has to be hand in glove with adaptability. Professionally, it showed me a balance between human capabilities and functionalities of AI to provide an optimal outcome.

In the wake of such contributions, the work has its limitations, too. For example, only certain tools and frameworks were tested ; hence, the findings may not generalize to other contexts. Future work could also focus on the contextual understanding of AI tools and the implementation of more sophisticated prompt engineering strategies. These techniques can also be extended to other domains to provide more insights with wider applicability.

The conclusion of this project is that a very strong foundation is laid for the employment of AI in software development, and both GitHub Copilot and ChatGPT have the potential to hugely boost the development process once correctly guided by methodologies that define its limitations. In

the future, technology will continue to change. Undoubtedly, the introduction of artificial intelligence into the field of software engineering has brought new possibilities and rethought conventional limits to innovation.

Bibliographie

[1] Vermeg , 2023 , « Vermeg Official Website , »adresse : https://www.vermeg.com/fr/ , 12/8/2024.

[2] Scrum.org , 2021 , « What is Scrum ? , »adresse : https://www.scrum.org/resources/what-scrum-module , 12/8/2024.

[3] G. Documentation , 2024 , « What is GitHub Copilot ? , »adresse : https://docs.github.com/en/copilot/about-github-copilot/what-is-github-copilot , 20/8/2024.

[4] OpenAI , 2024 , « ChatGPT - OpenAI , »adresse : https://openai.com/index/chatgpt/ , 7/8/2024.

[5] H. Touvron, T. Lavril, G. Izacard et al. , 2023 , « LLaMA : Open and Efficient Foundation Language Models , »adresse : https://arxiv.org/abs/2302.13971 , 15/8/2024.

[6] J. Wei, X. Wang, D. Schuurmans et al. , 2022 , « Chain-of-Thought Prompting Elicits Reasoning in Large Language Models , »adresse : https://arxiv.org/abs/2201.11903 , 25/8/2024.

[7] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi et D. Zhou , 2022 , « Self-Consistency Improves Chain of Thought Reasoning in Language Models , »adresse : https://arxiv.org/abs/2203.11171 , 14/8/2024.

[8] S. Yao, H. Zhao, D. Yu et al. , 2022 , « ReAct : Synergizing Reasoning and Acting in Language Models , »adresse : https://arxiv.org/abs/2210.03629 , 8/9/2024.

[9] N. Shinn, K. Labash et A. Kashyap , 2023 , « Reflexion : Language Agents with Verbal Reinforcement Learning , »adresse : https://arxiv.org/abs/2303.11366 , 18/8/2024.

[10] Spring , 2024 , « Spring Boot , »adresse : https://spring.io/projects/spring-boot , 5/8/2024.

[11] Angular , 2024 , « Angular Overview , »adresse : https://angular.dev/overview , 22/8/2024.

[12] Oracle , 2024 , « What is MySQL ? , »adresse : https://www.oracle.com/ca-fr/mysql/what-is-mysql/ , 27/8/2024.

ملخّص

انن َ ُنرسِتَريناقش هذا التقرير دمج الأدوات التي تعتمد على الذكاء الاصطناعي، مثل رهَتصة وِتسُب رِدت، في تطوير تطبيقات الويب باستخدام إطاري ضرِذ ت و انُلَر. يقيّم المشروع بعض التقنيات في هندسة البرمجة، مع التركيز على مدى جودتها في تحسين سير عمل التطوير. تم بناء تطبيق لوحة تحكم يدعم رضو وتصور البيانات لاختبار هذه الأساليب. وقد أظهرت النتائج الإمكانات الكبيرة للذكاء الاصطناعي في تحسين الإنتاجية من خلال تبسيط مهام الترميز، ولكن أيضاًا بعض نقاط الضعف فيما يتعلق بالسياق في المطالبة التكرارية. تؤكد النتائج على أهمية هندسة الموجهات الفعالة في الاستفادة من أدوات الذكاء الاصطناعي في إيجاد حلول برمجية مبتكرة.

كلمات مفاتيح :

التطوير القائم على الذكاء الاصطناعي، والهندسة الفورية، وِتسُب رِدت، وِتسُب رِدت، وَتصة، وتطبيق الويب.

Abstract

This report discusses the integration of AI-driven tools, such as ChatGPT and GitHub Copilot, into the development of web applications with Spring Boot and Angular frameworks. The project assesses some of the techniques in Prompt Engineering, focusing on how good they are in optimizing development workflows. A dashboard application was built that supported CRUD and data visualization to test these methods. Results have shown the great potentials of AI in improving productivity by simplifying coding tasks, but also some weaknesses regarding context in iterative prompting. The findings emphasize the importance of effective prompt engineering in leveraging AI tools for innovative software solutions.

Keywords : AI-driven development, prompt engineering, GitHub Copilot, ChatGPT, web application.

2 نهج أبو الريحان الباروني 2080 أريانة الهاتف : 164 706 71 الفاكس : 698 706 71 البريد الالكتروني : isi@isim.rnu.tn

2, Abou Raihane Bayrouni 2080 l'Ariana **Tél** : 71 706 164 **Fax** : 71 706 698 **Email** : isi@isim.rnu.tn