

STEPPING UP THE PROMPT GAME: REQUIREMENTS ENGINEERING AS A CATALYST FOR EFFECTIVE CODE GENERATION

Favour Kio

July 18th, 2023

CS 846 – Requirements Engineering

PRESENTATION OVERVIEW

- Introduction
- Review Approach
- Literature review
 - Requirements Engineering
 - Prompt Engineering
- Leveraging Requirements Engineering for Prompt Engineering
- Conclusion
- Future works
- References

A QUICK INTRODUCTION, SHALL WE?

MOTIVATION AND BACKGROUND

- The story of Ken, the Software developer
- ChatGPT == New Stack overflow.
- Tool for support instead.
- Thesis: Prompt Engineering with LLMs for code generation.



Retrieved from: <https://www.istockphoto.com/illustrations/happy-software-developer>

INTO THE LITERATURE REVIEW THEN

KEY WORDS USED IN SEARCH

- "Requirements engineering" OR "Requirement engineering" OR "Requirements Engineer" OR "Requirement engineer"
- "Prompts engineering" OR "Prompts Engineer" OR "Prompt engineering" OR "Prompt Engineer"
- LLMS OR "generative AI" OR "large language models" OR "large language model" OR "prompt engineer" OR "prompt engineers" OR "prompt engineering" OR "generative code" OR "code generation"
- "Requirements engineering" OR "requirements specification" OR "requirement engineering"

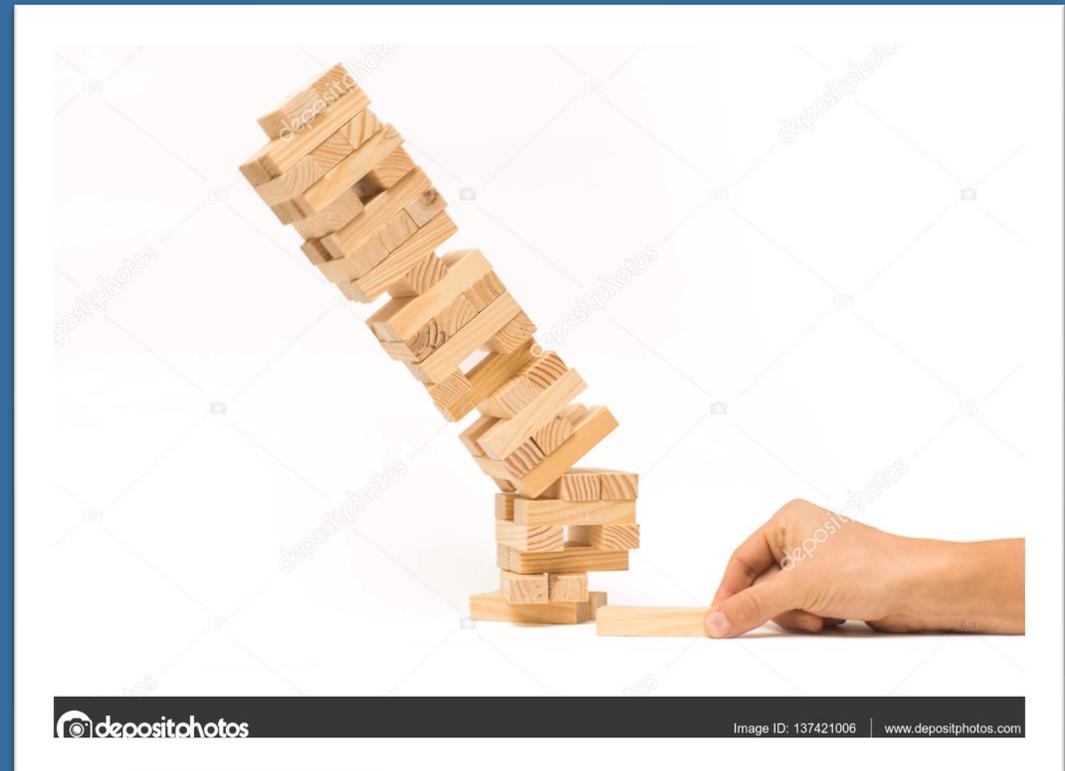
REVIEW APPROACH

- Key words used
- Number of meaningful results – 36
- Number of papers read – 20
- Scope of research – LLMs/code Generation

REQUIREMENTS ENGINEERING

HOW CRUCIAL IS RE?

- Knowing what to build is the first step of any system development (Berry, 1998)
- It is the most difficult part of building a software. (Brooks, 1987)
- No other part of the work so cripples the resulting system if done wrong. (Brooks, 1987)
- The process of arriving at a set of features that need to be developed is referred to as requirements engineering (RE) (Berry, 2012)
- We know that the cost of fixing errors in requirements goes up exponentially as we progress across the project life cycle.
- According to a study by NASA, the cost to fix requirements during the requirements generation phase is minimal but can go up to 1500x in the operations phase. (Ray, 2023)



Retrieved from: <https://depositphotos.com/201291458/stock-photo-board-game-jenga-tower-of.html>

July 18, 2023

9

RE IN MODEL-DRIVEN DEVELOPMENT (Loniewski, et al., 2010)

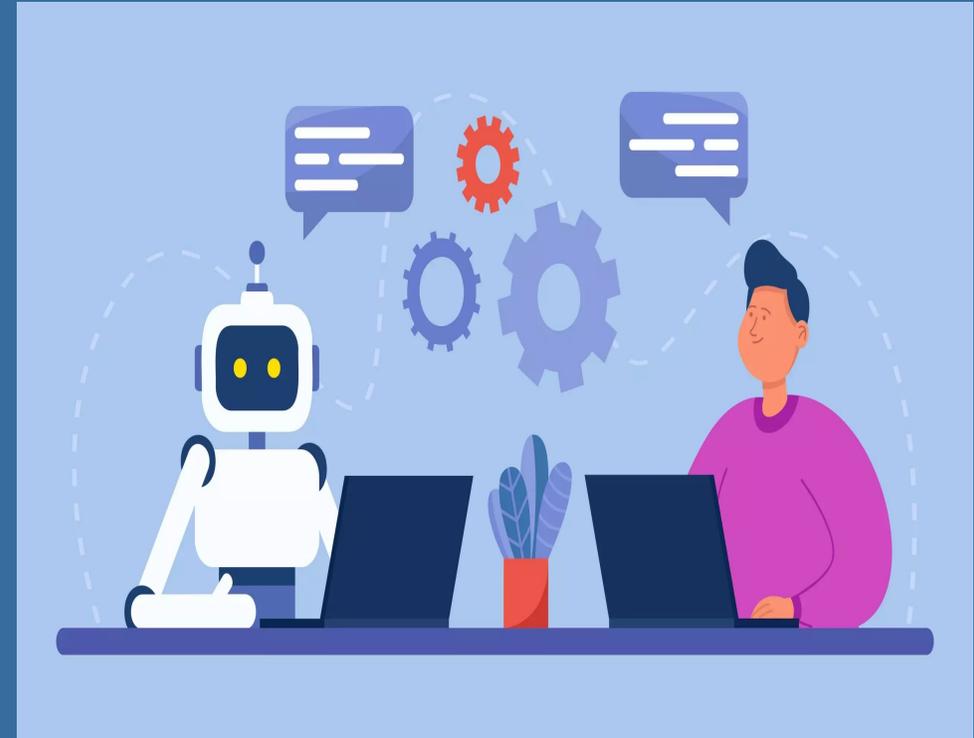
- There are no empirical studies on the benefits of the use of RE techniques in the improvement of productivity, efficiency and software development process quality and effectiveness.
- Properly managed, well-documented and easily understandable software requirements definitions have a great impact on final product quality.
- 62% of those that apply some RE techniques in the MDD approach used models as a means to represent the requirements.

RE + AI

- Bencomo asked if we could make better software by combining human and artificial intelligence (Bencomo et al., 2021)
- Users come from different stakeholder groups and the requirements of each group are expressed as different sets of multiple goals. (Bencomo et al., 2021)
- User requirements are the forces that change the shape of the ideas explored by AI. (Bencomo et al., 2021)
- The more an AI tool understands human requirements, and human reasoning, the more they can produce more efficient solutions
- Unfortunately, in many cases, the requirements are incomplete, and the projects are under-specified. Imprecise requirements and specifications cause not only problems in the development, but they also result in inaccurate cost estimations

RE + AI + AGILE (Kuhlen et al., 2016)

- If the effort of a requirement is not estimated correctly, the effort for a successful implementation will be difficult to assess at the end.
- The success of software development depends on four variables: time, functionality, resources and technical debts
- Often, developers reduce the quality of implementation to keep up with deadlines
- The use of a code generator could offer potentials to improve the processes of requirements engineering.
- It could shift effort in development to the requirements engineering process
- They do not show the savings which are realized by the code generator.



Retrieved from: <https://www.wellable.co/blog/is-chatgpt-the-new-health-coach-of-2023/>

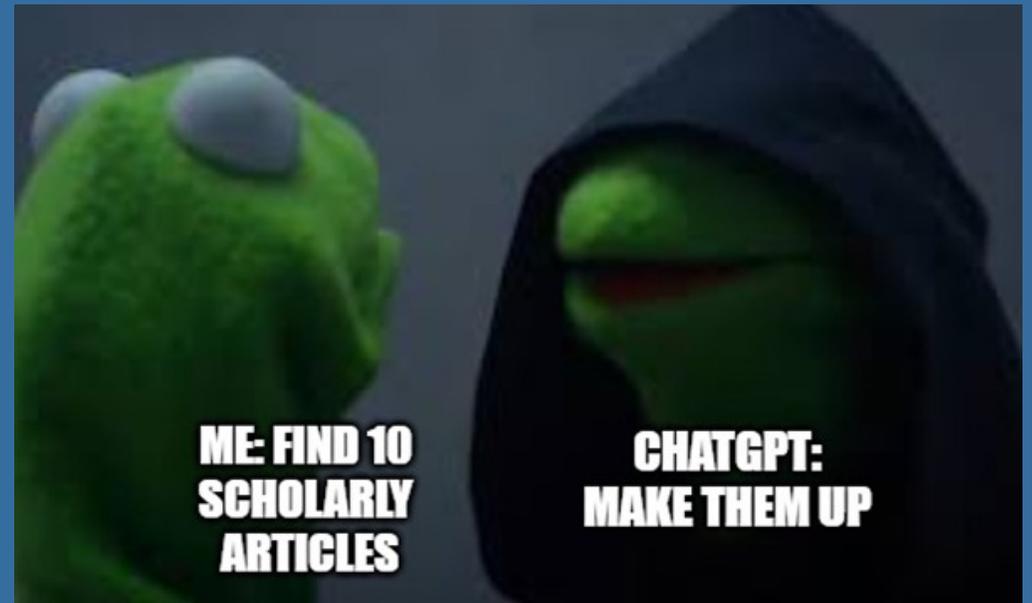
PROMPTS ENGINEERING

LARGE LANGUAGE MODELS(LLMs)

- Large Language Models (LLMs) have experienced a surge in popularity and adoption across various scenarios. (Liu et al., 2023)
- Chat-enabled AI tools such as ChatGPT and GitHub Copilot can aid a range of common software development and engineering tasks (Zhang et al., 2023)
- More focus was placed on ChatGPT for this study
- The utilization of ChatGPT has substantially enhanced productivity in numerous industries, allowing for quicker and more efficient processing of natural language tasks and beyond. (Liu et al., 2023)

LLMS ARE NOT PERFECT

- From the SE perspective, ChatGPT is viewed as an unprecedented example of a chatbot that can produce well-articulated responses to complex queries.
- However, it remains an unexplored area in terms of its potential and perils in the context of software development processes (Ahmad et al., 2023)
- The tendency of ChatGPT to "hallucinate" confidently and enthusiastically when generating incorrect output is quite dramatic. (White et al., 2023)



PROMPT ENGINEERING

- A prompt is a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing its latent capabilities. (White et al., 2023)
- **Prompt engineering** is the means by which LLMs are programmed via prompts.
- In a chat-based environment, a prompt is a chat message that the user sends to an LLM, such as ChatGPT. (White et al., 2023)
- Strategies or formats of asking Dad
- **Prompt patterns** play an essential role in providing the foundation for prompt engineering. (Zhou et al., 2022)

KEY PROMPT PATTERNS

THE 2 GENERAL PATTERNS (White et al., 2023)

- First prompt pattern is to directly ask an LLM to provide some information or generate some code.
- Second prompt pattern is to dictate rules for the LLM to follow going forward, such as conforming to certain coding standards.
- The second type of prompt, customizes future interactions with the LLM by defining additional rules to follow or information to use when performing subsequent tasks

THE CODE CLUSTERING PATTERN (Schmidt et al)

- The goal of this pattern is to separate and cluster code into functions, classes, etc.
- Unless the LLM is told otherwise, its code will solve the problem at hand and often does not solve structuring problems, such as separating pure and impure functions, that it has not been asked to solve.
- "Whenever I ask you to write code, I want you to write code in a way that separates functions with side-effects, such as file system, database, or network access, from the functions without side-effects."
- This is one of the most basic patterns that can dramatically improve the perceived quality of LLM- produced code

THE INTERMEDIATE ABSTRACTION PATTERN (Schmidt et al)

- LLMs will generate code that is very procedural and directly translates the goals into code.
- As the LLM is asked to continually add features to the code, it may produce longer and longer functions with little to no separation of concepts into modules, functions, or other abstractions that will facilitate long-term maintainability.
- "Whenever I ask you to write code, I want you to separate the business logic as much as possible from any underlying 3rd-party libraries. Whenever business logic uses a 3rd-party library, please write an intermediate abstraction that the business logic uses instead so that the 3rd-party library could be replaced with an alternate library if needed."

THE PRINCIPLED CODE PATTERN (Schmidt et al)

- The goal of the pattern is to use well-known names for coding principles to describe the desired code structure without having to explicitly describe each individual design rule.
- "From now on, whenever you write, refactor, or review code, make sure it adheres to SOLID design principles."
- Only well-known named descriptions of code qualities from prior to when the LLM was trained.

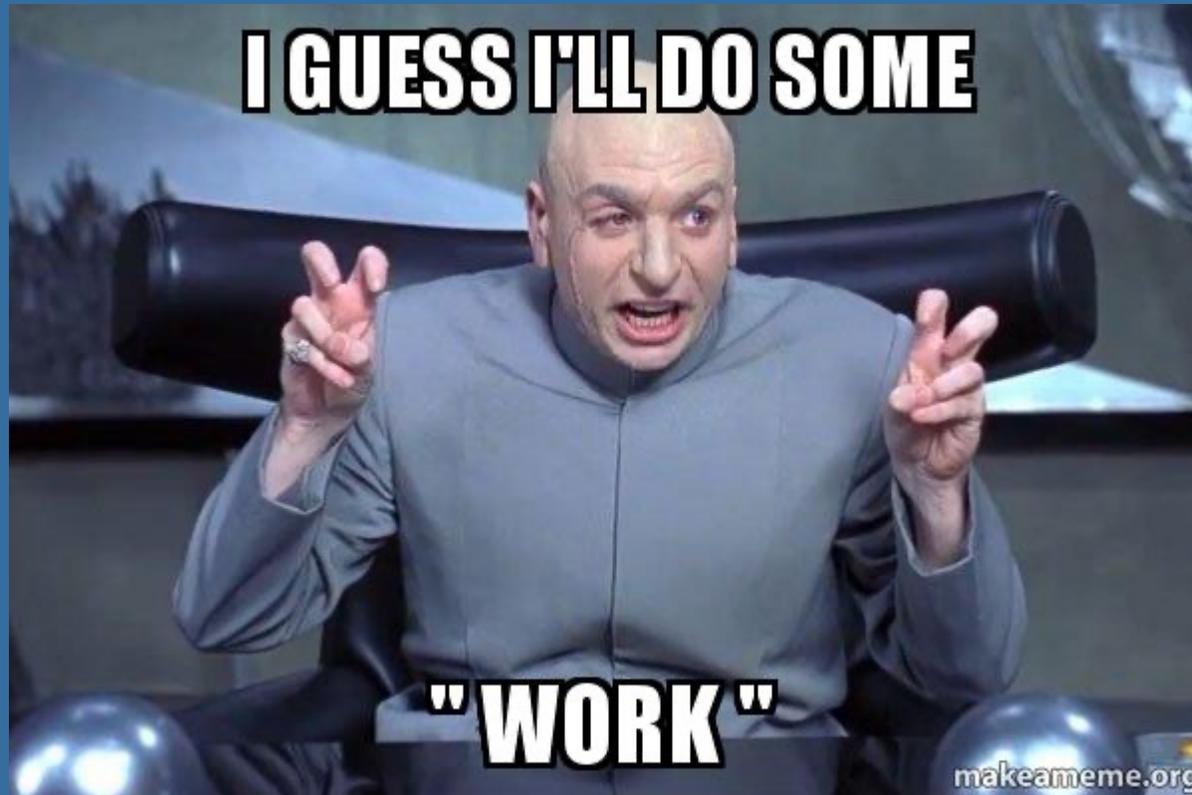
CHAIN OF THOUGHT (Zhou et al., 2022)

- Chain-of-thought reasoning has been shown to dramatically improve the ability of LLMs to complete complex reasoning tasks, such as solving math problems that require multiple steps.
- One of the most influential recent works of prompt engineering was the discovery that LLMs could be made to give chain-of-thoughts simply by prepending “Let’s think step by step.” to the beginning of the LLM’s response.
- APE produces the prompt: “Let’s work this out in a step-by-step way to be sure we have the right answer.” This generated prompt further improves performance from 78.7 to 82.0 on MultiArith and from 40.7 to 43.0 on GSM8K.

THE QUESTION REFINEMENT PATTERN (Schmidt et al)

- The goal of this pattern is to ensure an LLM always suggests potentially better or more refined questions PE could ask instead of their original question.
- In addition, an LLM may help PE find the information or achieve their goal in fewer interactions than if they employed conventional "trial and error" prompting.
- “Within scope X, suggest a better version of the question to use instead”
- It helps bridge the gap between the PE’s knowledge and the LLM’s understanding, thereby yielding more efficient and accurate interactions.

- "Requirements engineering" OR "Requirement engineering" OR "Requirements Engineer" OR "Requirement engineer"
- "Prompts engineering" OR "Prompts Engineer" OR "Prompt engineering" OR "Prompt Engineer"



MY ATTEMPT AT BRINGING THEM TOGETHER

THE KEY THINGS “KEN” SHOULD KNOW

- An LLM’s output is only as good as the prompt it is given!
- A strong focus on D requirements is needed.
- Prompt Engineers need to collaborate with stakeholders and key personnel to ensure their interpretation of the requirements are correct – They should ideally be part of the RE team.
- PE who understand exactly what is required of them would design their prompts differently.
- “One of the factors that seems to influence an individual's effectiveness in requirements engineering activities is knowledge of the problem being solved, i.e., domain knowledge.”
(Berry, 2012)

LEVERAGING THE QUESTION REFINEMENT PATTERN + CHAIN OF THOUGHT

- From now on, whenever I ask you to generate any code or screen in any programming language, before answering me:
 1. Identify missing requirements I may have overlooked in my prompt such as code vulnerabilities or security or bugs or user-centric requirements or error handling.
 2. Suggest a better version of the question to use as a prompt that incorporates the newly identified requirements and ask me if I would like to use your question instead.
 3. If I decline, answer the initial prompt, but if I accept, answer the most recent prompt you've given me.

G Give me a register function in javascript



Certainly! Here's an example of a registration function in JavaScript:



```
javascript Copy code  
  
function register(username, password) {  
  // Simulating registration logic  
  // Here, you can add code to store the username and password in a database  
  console.log(`Registered user: ${username}`);  
  return true; // Registration successful  
}
```

In this example, the `register` function takes two parameters: `username` and `password`. You can add code within this function to handle the registration process, such as storing the username and password in a database or performing any other necessary operations.

For example, you can use an AJAX request to send the data to a server endpoint that handles user registration:

```
javascript Copy code  
  
function register(username, Regenerate response  
  // Simulating registration logic
```

Send a message ▶

INITIAL PROMPT

KI

From now on, whenever I ask you to generate any code or screen in any programming language, before answering me:

1. Identify missing requirements I may have overlooked in my prompt such as code vulnerabilities or security or bugs or user-centric requirements or error handling.
2. Suggest a better version of the question to use as a prompt that incorporates the newly identified requirements and ask me if I would like to use your question instead.
3. If I decline, answer the initial prompt, but if I accept, answer the most recent prompt you've given me.



Understood! I will follow the process you described for generating code or screens in any programming language. I will identify missing requirements, suggest improvements to the question, and give you the option to use the improved question instead. If you decline the improved question, I will answer your original prompt. If you accept the improved question, I will answer the most recent prompt based on the improved requirements. Please go ahead and provide me with your initial prompt or question.

KI

Give me a register function in javascript



Sure! Before I proceed, I want to clarify the requirements for the register function. Here's what I understand so far:

Register Function Requirements:

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Ver.](#)

KI

Give me a register function in javascript



Sure! Before I proceed, I want to clarify the requirements for the register function. Here's what I understand so far:

Register Function Requirements:

1. The register function should be implemented in JavaScript.
2. It should handle user registration.
3. However, there are some missing requirements in your prompt. Could you please provide more details about the specific functionality and behavior you expect from the register function? For example, should it collect user information like username, email, and password? Should it validate the input? Should it interact with a database or API to store the user's data?

Once you provide the missing requirements, I can suggest improvements to the question or proceed with answering based on the information provided.

KI

Let it take username and email, validate the inputs and interacts with local storage



Thank you for providing the additional requirements. Based on your updated specifications, here's an improved version of the prompt:

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Ver.](#)

Let it take username and email, validate the inputs and interacts with local storage

Thank you for providing the additional requirements. Based on your updated specifications, here's an improved version of the prompt:

Improved Prompt:

Write a JavaScript `register` function that takes a `username` and an `email` as inputs. The function should validate the inputs and interact with the local storage to store the user's data.

Please let me know if you would like to proceed with this improved prompt, or if you have any further modifications.

Proceed

Great! Here's an implementation of the `register` function in JavaScript that fulfills the requirements you mentioned:

```
javascript Copy code  
  
function register(username, email) {  
  // Validate inputs  
  if (!username || !email) {  
    console.log('Please provide both a username and an email.');    return;  
  }  
  
  // Validate email format  
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;  
  if (!emailRegex.test(email)) {  
    console.log('Please provide a valid email address.');    return;  
  }  
  
  // Store user data in local storage  
  const userData = {  
    username: username,  
    email: email  
  };  
  localStorage.setItem('userData', JSON.stringify(userData));  
}
```

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Version](#)

KI Proceed

Great! Here's an implementation of the `register` function in JavaScript that fulfills the requirements you mentioned:

```
javascript Copy code  
  
function register(username, email) {  
  // Validate inputs  
  if (!username || !email) {  
    console.log('Please provide both a username and an email.');    return;  
  }  
  
  // Validate email format  
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;  
  if (!emailRegex.test(email)) {  
    console.log('Please provide a valid email address.');    return;  
  }  
  
  // Store user data in local storage  
  const userData = {  
    username: username,  
    email: email  
  };  
  localStorage.setItem('userData', JSON.stringify(userData));  
}
```

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Version](#)

6 Give me a register function in javascript



Certainly! Here's an example of a registration function in JavaScript:



```
javascript Copy code
function register(username, password) {
  // Simulating registration logic
  // Here, you can add code to store the username and password in a database
  console.log('Registered user: ${username}');
  return true; // Registration successful
}
```

In this example, the `register` function takes two parameters: `username` and `password`. You can add code within this function to handle the registration process, such as storing the username and password in a database or performing any other necessary operations.

For example, you can use an AJAX request to send the data to a server endpoint that handles user registration:

```
javascript Copy code
function register(username, password) {
  // Simulating registration logic
  // Here, you can add code to store the username and password in a database
  console.log('Registered user: ${username}');
  return true; // Registration successful
}
```

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT May 24 Version](#)

```
// Validate inputs
if (!username || !email) {
  console.log('Please provide both a username and an email.');
```

```
return;
}
```

```
// Validate email format
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailRegex.test(email)) {
  console.log('Please provide a valid email address.');
```

```
return;
}
```

```
// Store user data in local storage
const userData = {
  username: username,
  email: email
};
localStorage.setItem('userData', JSON.stringify(userData));

console.log('User registered successfully!');
```

A DRAMATIC IMPROVEMENT!

CONCLUSION

- This research presented a literature review to find out whether aspects of RE can be applied to Prompt Engineering.
- The study finds that from an initial standpoint, a Prompt Engineer who acknowledges and takes part in Requirement Engineering activities would design better prompts.
- Although not conclusive, there is some support to involving ChatGPT as a part of the prompt design process

“if you test the wrong generated code against the wrong set of requirements, it could look like the right generated code – until it fails in production, where the cost of fixing it would be significantly more”

- Kio

FUTURE WORKS

- Empirical research to see exactly how helpful domain knowledge is in PE
- How efficient ChatGPT/LLMs are at detecting bad requirements through PE
- How effective are LLMs at creating requirements specifications themselves?
- How much does prompt engineering save in agile?

REFERENCES

- D. M. Berry and B. Lawrence, "Requirements Engineering," in IEEE Software, vol. 15, no. 2, pp. 26-29, March-April 1998, doi: 10.1109/MS.1998.663780.
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., & Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. ACM Computing Surveys (CSUR), 54(3), 1-41.
- Ray, A. T. (2023). STANDARDIZATION OF ENGINEERING REQUIREMENTS USING LARGE LANGUAGE MODELS (Doctoral dissertation, Georgia Institute of Technology).
- Loniewski, G., Insfran, E., Abrahão, S. (2010). A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds) Model Driven Engineering Languages and Systems. MODELS 2010. Lecture Notes in Computer Science, vol 6395. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16129-2_16
- Kuhlen, D., & Speck, A. (2016). The potentials of a code generator which faces the stress ratio of requirements engineering processes in agile development projects. Modellierung 2016-Workshopband.

- Niknafs, A., & Berry, D. M. (2012, September). The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation. In 2012 20th IEEE International Requirements Engineering Conference (RE) (pp. 181-190). IEEE.
- Brooks, F. P., & Bullet, N. S. (1987). Essence and accidents of software engineering. IEEE computer, 20(4), 10-19.
- Bencomo, N., Guo, J. L., Harrison, R., Heyn, H. M., & Menzies, T. (2021). The secret to better ai and better software (is requirements engineering). IEEE Software, 39(1), 105-110. The Secret to Better AI and Better Software (Is Requirements Engineering)
- Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Aktar, M. S., & Mikkonen, T. (2023, June). Towards human-bot collaborative software architecting with chatgpt. In Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering (pp. 279-285).
- Doglio, F. (2022, December 13). The Rise of ChatGPT and the Fall of the Software Developer—Is This the Beginning of the End? Medium. <https://medium.com/geekculture/the-rise-of-chatgpt-and-the-fall-of-the-software-developer-is-this-the-beginning-of-the-end-a48485e6da1b>
- Ma, W., Liu, S., Wang, W., Hu, Q., Liu, Y., Zhang, C., ... & Liu, Y. (2023). The Scope of ChatGPT in Software Engineering: A Thorough Investigation. arXiv preprint arXiv:2305.12138.
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., ... & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382.

- White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2023). Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839.
- Zhang, J., Chen, Y., Niu, N., & Liu, C. (2023). A Preliminary Evaluation of ChatGPT in Requirements Information Retrieval. arXiv preprint arXiv:2304.12562.
- Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., ... & Liu, Y. (2023). Jailbreaking chatgpt via prompt engineering: An empirical study. arXiv preprint arXiv:2305.13860.
- Oppenlaender, J., Linder, R., & Silvennoinen, J. (2023). Prompting ai art: An investigation into the creative skill of prompt engineering. arXiv preprint arXiv:2303.13534.
- Schmidt, D. C., Spencer-Smith, J., Fu, Q., & White, J. Cataloging Prompt Patterns to Enhance the Discipline of Prompt Engineering.
- Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). Large language models are human-level prompt engineers. arXiv preprint arXiv:2211.01910.

THANK YOU!