The Application of ChatGPT for Identification of Microservices

Tatjana Stojanović

Department of Software Engineering
University of Belgrade
Belgrade, Serbia
tatjana.stojanovic@fon.bg.ac.rs
[0000-0001-7191-6444]

Saša D. Lazarević

Department of Software Engineering
University of Belgrade
Belgrade, Serbia
sasa.lazarevic@fon.bg.ac.rs
[0000-0002-5588-4195]

Abstract—Identification and definition of microservices is one of the most important aspects of systems based on the microservices architecture. There are many approaches for proper identification of microservices and their boundaries. However, these approaches are usually carried out by software architects, meaning that the overall success of the system design depends on their skills, abilities and understanding of the entire system. This comprehensive task can be quite demanding, which leaves room for oversights and errors. Luckily, new technologies emerge daily, as well as new uses for those technologies, which make such difficult tasks a little bit easier. In this paper, utilization of the popular ChatGPT large language model for analyzing software requirements and identifying microservices is explored. Three different examples are presented along with recommended solutions, showing that usage of GPT for analyzing software requirement can be useful, but used with caution because of its drawbacks.

Keywords - chatgpt, microservices architecture, artificial intelligence, large language model

I. INTRODUCTION

Microservices architecture is a distributed software system architecture which is consisted of highly cohesive and loosely coupled services. [1][2] Microservices architecture can be very complex for designing and implementing. One of the first challenges faced when designing systems with this architecture is defining boundaries of microservices. Poorly designed microservices can have a negative impact on system performance [3], which may lead to high interdependence, system not being fault tolerant, overlapping of responsibilities and other. Determining boundaries of microservices demands knowledge and understanding of the system, which can be difficult having in mind that these systems are usually very complex. Studies from the practice shows that this process is prone to error, and usually needs several iterations in order to get it right. This work is usually done by software architects and domain experts, meanings the quality of the solution depends of their skill and understanding of the customer's needs. One of the technologies that appeared recently and left deep impact is ChatGPT. New ways to utilize this technology are discovered daily. Capabilities of ChatGPT are explored in different areas of software engineering such as debugging[4], generating[5] and testing code, improving code quality[6], refactoring, software design[6], etc. In order to maximize effectiveness, different prompt engineering techniques and prompt catalogues are suggested [6]. Reference [7] examines ChatGPT's capabilities in the software development cycle, where in requirement analysis, it was able to successfully separate functional and non-functional requirements and generate detailed use case specification. In domain modeling, it was able to identify domain concepts, its attributes and relationship. In design modeling, it provided a well-detailed design class and sequence diagram.

The goal of this paper is to explore could large language models, such as ChatGPT, be used to analyze requirements and provide usable designs for microservices architecture. Here it will be explored whether GPT can be used for identification of microservices and for making suitable recommendations depending on system needs and given instructions.

ChatGPT is a large language model (LLM) developed by OpenAI. [8]. Currently ChatGPT is based on GPT-3.5 architecture (Generative Pre-trained Transformer) which is a natural language processing model that is trained on an enormous amount of text data. It is available free use since the 22nd November of 2022. ChatGPT has reached 100 million users in about two months. [9]

Large language models are probabilistic models which are designed to learn statistical pattern in natural language. These models are pre-trained on a large amount of available text data. Their architecture is based on artifical neural networks, specifically on Transformer model which was invented in 2017. Transformer models have proved to be good in natural language contextual understanding.[10] Transformer-based models employ an encoder-decoder archtecture. While encoder endcodes text in a numerical representation, the decoder takes in such representation and decodes it back into text. The Generative Pre-trained Transformer (GPT) is a decoder, which generates a new text.

In order to determine capabilities of GPTs to understand and identify microservices based on description of needed functionalities, description of three different systems will be provided to the chatbot. Afterwards, each solution will be analyzed.

II. EXAMPLES

In this section, several examples will be presented. For each example several additional instructions will be given to the Chat GPT, and results will be presented and analyzed.

Two chats are made for each example – in one the whole description of the system was given at once and GPT was instructed to identify potential microservices, and in the other the description of the system was given partially, in pieces. The goal of making these two chats was to determine whether the way of conducting the conversation will affect the result. Also, for the same purpose, additional descriptions involved not only providing additional functionality but also changing some of the existing ones. In order to assess GPTs overall understanding of the system several questions were asked: to identify the interdependency between microservices, to identify the most critical service and service which will have to be scaled.

A. Example 1: Online car parts store

In the first chat, ChatGPT was given the whole description about the system for online car parts store:

The software system is used for online car parts store. The customers can select the model of their car, and after that all available parts for the chosen car are displayed in categories. The available quantity of parts is updated automatically after successful order or it can be updated manually by authorized administrator user. In order to create an order or purchase car parts, customers must create a new user using an email, or they can log in using an existing user. When creating a user, customer must provide personal information, including first and last name, telephone number, and residence address. After the user has been created, a verification email containing a verification link is sent to the customer. The customer user is verified after the customer clicks on the verification link in the sent email. In order to purchase parts, customer must have a verified user in the system, and must create a purchase order. An order contains chosen car parts and quantity of chosen parts. Personal customer information is loaded from customer's user, and customer is given an option to ship the order to the residence address, or to specify another shipping address. After choosing an address, total price of the order is displayed and customer is given an option to use a discount by providing a discount code. If the customer provides it, the discount code is verified to check if it has expired or if it has already been used. If the code is valid, the price of the order is reduced by the amount specified by the discount code. Lastly, the customer chooses the payment method. They can choose to pay in cash on delivery, or to pay online using a credit or debit bank card. If they choose to pay using a bank card, the customer is redirected to the online payment page, where they are required to specify card information, after which the system communicates with the bank to realize the payment. The system has two types of users: customer users and administrator (employee) users. Both log in the system using the same login form. Administrator users can modify the quantity of existing car parts or they can add new car parts and new categories of parts, and also modify orders which have not yet been realized. While shopping for parts, customers can use shopping cart, where all chosen parts are placed. After choosing all desired parts, customer can review their shopping cart and modify it is necessary, after which they are forwarded to the page for creating of the order. Every customer can get a recommendation discount, when they recommend the application to the new user. In this case, it is necessary to track the list of the recommended users for given customer, since after certain number, the customer who gave recommendations will get an increase in the discount amount.

From this description, following services were identified by GPT[1]:

1. User Service: This microservice handles the creation, authentication, and verification of user accounts. It is responsible for storing and managing user data, sending verification emails, and tracking recommended users.
2. Inventory Service: This microservice is responsible for managing the inventory of car parts available in the online store. It handles the addition, modification, and deletion of car parts and categories of parts, as well as updating the quantity of parts available after successful orders.
3. Order Service: This microservice handles the creation, modification, and processing of purchase orders. It retrieves personal information from the user service, calculates the total price of the order, and verifies discount codes. It also handles payment processing through communication with the bank.
4. Cart Service: This microservice manages the shopping cart functionality of the online store. It allows customers to add and modify their chosen parts before creating an order.
5. Recommendation Service: This microservice tracks recommended users for each customer and calculates the discount amount based on the number of recommended users.
6. Authentication Service: This microservice handles authentication and authorization of both customer and administrator users. It is responsible for ensuring that only authorized users can access and modify data in the system.
7. Shipping Service: This microservice handles the selection and verification of shipping addresses for orders.
8. Notification Service: This microservice handles sending notifications to customers about order status updates, shipping information, and other relevant information.
9. Discount Service: This microservice manages the creation and verification of discount codes for customers.

In further conversation it was determined that the User Service is the most likely to be changed, while the most crucial part is the Order Service, and the Inventory Service

[1] Names and description of the services are generated by ChatGPT.

is the most likely to be scaled. ChatGPT was instructed to determine interdependencies between these microservices, in order to establish overall understanding of the system. At Fig. 1, identified microservices and their relationship is shown.

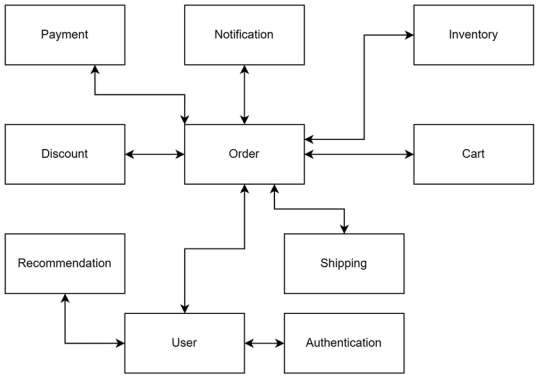

Fig. 1. Online car parts store I

As it can be seen, the Order microservice is placed in the center of the architecture, with many other services depending on it, and also having many dependencies. This might be seen as problematic and as a flaw in the design, and is determined by the nature of the system. However, it is what was provided in the description of the system, and ChatGPT provided faithful model to the description. Whether it is suitable description of the system, and whether changes need to be made to it, is up to the software architect to analyze. By asking further questions, and providing more information about the system, these potential flaws can be eliminated.

In the second chat GPT was asked to identify and define microservices based on the part of the full description. For the given description, GPT has identified six microservices: User Service, Catalog Service, Order Service, Payment Service, Admin Service and Notification Service, which was not immediately identified, but later in the conversation. Afterwards, ChatGPT was given an additional description of the system and was asked whether more microservices will be necessary. Two more services were identified for implementing these changes:

1. Recommendation Service,
2. Payment Gateway Service: This microservice would integrate with payment gateway APIs to enable customers to pay online using a credit or debit bank card. It would handle the payment transaction and update the order status once the payment has been completed.

The User Service is identified as critical and highly dependent microservice, as it serves as a central point of authentication and user information. The Order Service is also likely to be highly dependent, as it manages the core functionality of the system, while the Payment Service, Payment Gateway Service, and Recommendation Service are more specialized and have narrower dependencies. The most likely parts of the system to be changed in the future are the discount and recommendation system and the user interface. Discount and recommendation may need to be updated to include new rules or conditions and recommendations based on user behavior or business needs.

Conducting several chats with ChatGPT, it did reference services which were not previously identified, but turned out to be necessary. At Fig. 2, identified microservices and their relationships are shown.

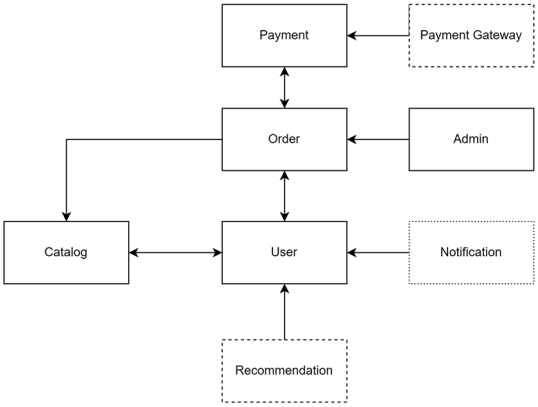

Fig. 2. Online car parts store II

It is obvious from Fig. 2 that Order microservice in this case is less dependent and has less other services depending on it, which might be seen as better solution. Hence, it can be concluded that the way the system was described to ChatGPT and the level of details provided affects the quality of the solution provided.

These two recommended designs are somewhat similar, while there are some differences in identified services. Giving more detailed instructions about the desired design, one can influence the complexity and quality of the solution. Instructions such as “make as many microservices as possible”, or “try not to make more than 5 microservices” can guide ChatGPT in providing a more desirable solution.

B. Example 2: A veterinary clinic

In the first chat, GPT is given the full description:

The software system is used for a veterinary clinic. The system allows scheduling pet examinations and interventions, as well as documenting them. The examinations are scheduled by customers through a form on a website. They should fill all the needed data and choose available date and time. It is expected that there will be short periods with high frequency of requests, and periods with rather low frequency. After the examination or intervention has been conducted, it is documented by the vet who conducted it. Vet can choose a pet if it has already been recorded, or can add it in the system if it does not exist in it. A vet does this by logging in with a dedicated user and inserting data describing the procedure through the dedicated form. Another type of users are admin users who manage the content of the website, providing and managing the available services which the clinic provides. They also manage the promotional part of the system, which consists of a blog with ed-

ucational and professional articles and a gallery of videos and pictures of the clinic and pets who are treated there. One more part of the system is the chat which customers can use to contact available vet who can then answer their questions. Scheduling the examinations and using the chat doesn't require customers to have a registered user. However, customers should provide an email through which they will get updates on their questions asked in the chat. The vets respond to questions in chat using their dedicated user. Only one vet is on duty at the time for responding to the customer's questions in the chat, and all the questions should be forwarded to them.

Services identified by ChatGPT are[2]:

1. Appointment Service: This microservice handles the scheduling of pet examinations and interventions. It provides functionality for customers to submit appointment requests through the website form, choose available dates and times, and store the relevant appointment data.
2. User Service: The User Service manages user authentication and authorization. It handles user registration, login, and user profile management. It differentiates between customer users, vet users, and admin users.
3. Pet Service: This microservice handles the management of pet data. It allows vets to add new pets to the system if they are not already registered. It also provides functionality for vets to associate pets with examination and intervention records.
4. Documentation Service: This microservice is responsible for documenting examinations and interventions. It allows vets to log in, access the dedicated form, and insert data describing the procedure. It stores and manages the examination and intervention records.
5. Admin Service: The Admin Service provides functionality for admin users to manage the content of the website. This includes managing the available services provided by the clinic, such as adding, editing, or removing services.
6. Promotional Service: This microservice handles the promotional part of the system. It manages the blog with educational and professional articles, as well as the gallery of videos and pictures related to the clinic and the pets treated there.
7. Chat Service: The Chat Service enables customers to contact available vets and ask questions through the chat feature. It allows customers to initiate chats without requiring registered users. The service also ensures that questions are forwarded to the vet on duty, who can respond using their dedicated user.

The scheduling and the pet management services are identified as the core services of the system, while the promotional content management service is pointed out as the service which is likely to undergo frequent changes. At Fig. 3, identified microservices and their relationships are shown.

[2] Names and description of the services are generated by ChatGPT.

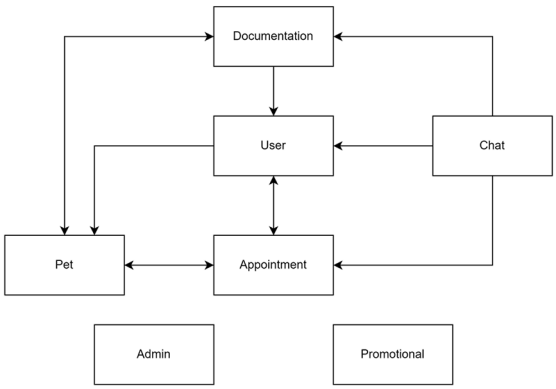

Fig. 3. The online system for the veterinary clinic I

As it can be seen at Fig. 3, the services are little less dependent on each other than in the previous example. However, there are two services (Admin and Promotional) which can be seen as isolated, which might be a sign that the model is not suitable for the system and that further changes are needed.

Given the partial description, followed by additional details these microservices are identified: Appointment scheduling microservice, Pet management microservice, Examination and intervention microservice, User management microservice, Content management microservice.

Additional microservices are: Promotional management microservice and Chat management microservice.

At Fig. 4, identified microservices and their relationships are shown.

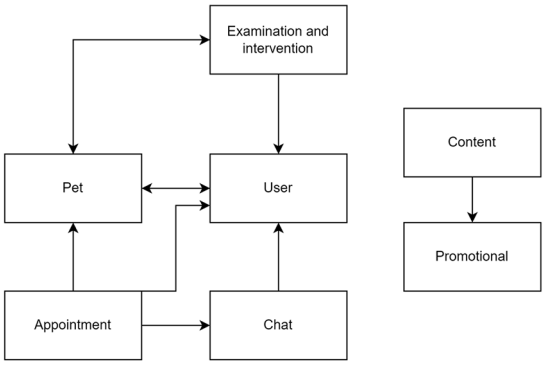

Fig. 4. The online system for the veterinary clinic II

In this iteration, there are no services which are isolated, but there are two services (Content and Promotional) which are connected with each other, but not with the rest of the system. Whether this is true for the described system is for the architect to analyze.

C. Example 3: An online music library and player

In the first chat, GPT is given the full description:

The software system is used for online music library and player. There are two types of users: regular and ad-

min users. Admin users can add new songs, albums and artists to the library and modify existing ones, as well as create public playlists. Regular users should create new account in order to log in if they do not already have existing account. In order to create the new account, users should provide personal information as well as email address. After that, users are required to set up two-factor authentication using their mobile telephone. Once the account has been created, users are able to search the music library and find wanted songs. In the search, they can use various filters, such as genre, artist, decade, album, etc. Users can create their own custom playlists, and can also mark certain songs as their favorites. They can also rate songs, albums and artists, as well as leave written reviews. These reviews are in a form of a blog, where other users can comment on reviews and mark them as useful or not useful. The song player has customization options, such as volume adjustment, equalizer options, options to repeat songs and to randomize playlists, change the songs order, etc. In order to use the services of this system, regular customers should provide a payment method. They can use free two weeks trial period, after which they should select one of the payment subscription plans. These plans include monthly and yearly payment, and if a user stops paying or decides to stop using the application, their access to the application should be restricted. There is also an option for users to get subscription from their mobile services provider through their mobile service contract. In this case, the system should be able to check with the mobile service provider whether the user's contract allows for the such kind of subscription.

Seven microservices were identified, as it follows:

1. Account Management Microservice: This microservice handles the creation and management of user accounts. It includes the functionality to store and retrieve personal information and email addresses of the users. It also handles the two-factor authentication process using mobile phones.
2. Music Library Microservice: This microservice manages the songs, albums, artists, and playlists in the music library. It provides the functionality to add new songs, albums, and artists, modify existing ones, and create public playlists. It also provides the functionality to search for songs using filters such as genre, artist, decade, album, etc.
3. Playlist Management Microservice: This microservice handles the creation and management of custom playlists by users. It provides functionality to add, modify, and delete playlists. It also includes the functionality for users to mark songs as their favorites.
4. Review Microservice: This microservice handles the rating and review functionality of the application. It includes the functionality for users to rate songs, albums, and artists, and to leave written reviews. Other users can comment on reviews and mark them as useful or not useful.
5. Player Microservice: This microservice handles the functionality of the song player. It includes the customization options such as volume adjustment, equalizer options, options to repeat songs and to randomize playlists, change the songs order, etc.
6. Payment Management Microservice: This microservice handles the payment functionality of the application. It includes the functionality to store and retrieve payment information of users. It also handles the free two weeks trial period and the selection of payment subscription plans. It includes the functionality to restrict access to the application if the user stops paying or decides to stop using the application.
7. Mobile Subscription Microservice: This microservice handles the functionality to check with the mobile service provider whether the user's contract allows for the subscription to the application. It includes the functionality to retrieve information about the user's mobile service contract.
8. The User management microservice and the Music library microservice are likely to be the most crucial for the system. The Music library microservice and the Playlist microservice are likely to be the most likely to be scaled, as the system grows and more songs and users are added. The Music library microservice may need to be scaled horizontally to handle the increased load, while the Playlist microservice may need to be scaled vertically to handle the increased number of playlists. The Rating and review microservice and the Player customization microservice are the most likely to be changed as they involve user preferences, which can change over time.

At Fig. 5, identified microservices and their relationships are shown.

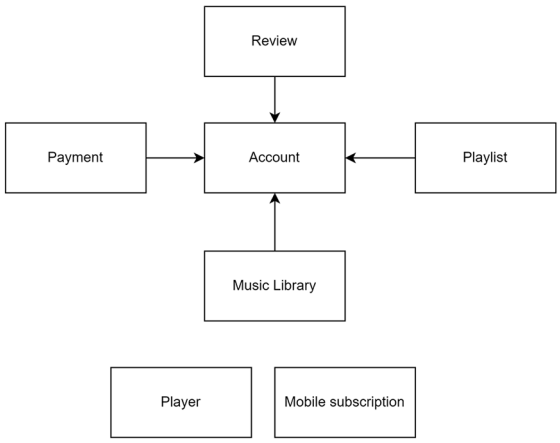

Fig. 5 The online music library and player I

At Fig. 5, a combination of previous examples can be seen. There is Account service which is placed in the center of the system, and there are two services which might be considered isolated. These might be signs that further analysis is needed.

Given the partial description, identified microservices are: User management microservice, Music library microservice, Playlist microservice, Favorites microservice.

Provided additional information, three more micros-

ervices were identified: Rating and review microservice, Player customization microservice, Payment management microservice.

At Fig. 6, identified microservices and their relationships are shown.

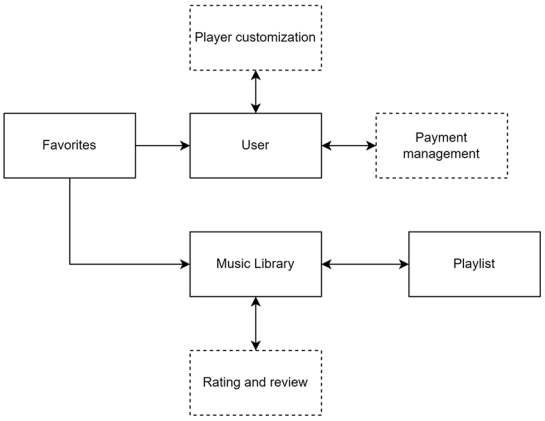

Fig. 6 The online music library and player II

After providing additional information, previous potential problems are resolved, since newly identified services connect the whole system and make it more granulated.

III. DISCUSSION

After analyzing previous examples and proposed solutions, it can be concluded that LLM GPT shows good understanding of microservices architecture and can provide solutions which make logical sense based on the given description of the system. Based on this, it can be used as an additional tool when defining microservices architecture. Being an LLM, GPT is capable of maintaining big amount of information about the system in the given context, while also allowing the changes and extensions of user requirements. Also, given additional instructions, GPT has proven to be able to adjust solution to the new requirements, and to steer solution to the desired one. This, of course, implies that the user of ChatGPT is competent and capable to recognize which solution is better, and to guide ChatGPT to it. Another important dimension is that the user should be careful to provide all of the relevant information. Otherwise, the proposed solution might not be as suitable to the actual system and might cause later problems.

Another positive side of using ChatGPT, is that it is capable of analyzing different characteristics of the solution, while showing the understanding of the most important features of the microservices architecture. Furthermore, it can predict which of the microservices are susceptible to change, which service provides the core functionality of the system and which service will probably have to be scaled. All that is presented above, along with the presented examples show that Chat GPT can be a useful tool for software architects. Having in mind its capability of understanding many details and complex topics, it is a good tool for determining context boundaries. Also, it is possible to influence the solution with providing additional details about the system, system requirements or other. Besides, as shown Chat GPT can recognize coupling between microservices and suggest improvements. Also, the number of microservices can be manipulated, while maintaining the most important principles of microservices architecture.

IV. CONCLUSION

With growing capabilities of GPTs, people are trying to utilize it in terms of automatization of many tasks. In this paper, utilization of this tool in terms of analyzing software requirements and identifying microservices which could be used to implement that system is presented. By providing additional instructions, user can quickly get many different solutions and have them automatically analyzed by GPT itself. The solutions provided by ChatGPT showed to be appropriate solutions for the given problem which made logical sense based on the given description. As seen, providing additional information or instructions can lead GPT to a different solution. Of course, the provided description must be correct and detailed, otherwise ChatGPT can overlook some important parts of the system, or some additional services may come up later in conversation.

One of the important things to note is that the solutions provided are only as good as provided input. From the analysis of the given examples, it is obvious that there are some details in the model which might be seen as problematic and not true to the system. Such things include one big service on which other services depend and the existence of the isolated services. It is up to the software architect to draw conclusions from this analysis and determine if model needs to be changed and in what direction. Hence, the user communicating with the tool should be well familiar with the concepts of microservices architecture and software design generally. The more experienced the person communicating with ChatGPT and the more they are familiar with the domain of the system in question and the system itself at all, the better the ending solution will be. This is the reason why it is highly unlikely that AI tools such as ChatGPT will ever be able to replace actual people when it comes to the design of software systems. Rather, it should be seen as a helping tool, one out of the many, that should help get to the optimal solution and to get to it faster. Using ChatGPT for identifying microservices can, without a doubt, be useful, in order not to overlook some important details, to save time working on initial solution and it can be practical to use for analyzing different parts of the system. The utilization of new technologies is not something that should be shied away from, but rather used to get better solutions.

For further research, it would be interesting to see if ChatGPT is capable of adapting existing solution to changes and determining whether existing solution have poorly designed architecture and how it could be improved.

REFERENCES

[1] S. Newman, Building Microservices - Designing fine-grained systems, O'Reilly Media, Inc., 2015.

[2] J. &. F. M. Lewis, "Microservices a definition of this new architectural term," 25 3 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html.

[3] M. R. a. R. H. Dharmendra Shadija, "Microservices: Granularity vs. Performance," in In Companion Proceedings of the10th International Conference on Utility and Cloud Computing (UCC '17 Companion). Association for Computing Machinery, New York, NY, USA, 215–220. , 2017.

[4] Haque, Md & Li, Shuai. (2023). The Potential Use of ChatGPT for Debugging and Bug Fixing. EAI Endorsed Transactions on AI and Robotics. 2. e4. 10.4108/airo.v2i1.3276.

[5] Liu, Chao & Xuanlin, Bao & Zhang, Hongyu & Zhang, Neng & Hu, Haibo & Zhang, Xiaohong & Yan, Meng. (2023). Improving ChatGPT Prompt for Code Generation.

[6] 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, abs/2303.07839.

[7] Kim, Dae-Kyoo. (2023). Using ChatGPT to Develop Software Systems: Alert to Software Engineers?. 10.13140/RG.2.2.26388.78725.

[8] "Open AI," 30 11 2022. [Online]. Available: https://openai.com/blog/chatgpt. [Accessed 10 5 2023].

[9] K. HU, "ChatGPT sets record for fastest-growing user base - analyst note,"2/2/2023. [Online]. Available: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/.

[10] M. Ramponi, "AssemblyAI: The Full Story of Large Language Models and RLHF," 3 5 2023. [Online]. Available: https://www.assemblyai.com/blog/the-full-story-of-large-language-models-and-rlhf/. [Accessed 10 5 2023].

[11] "Identifying domain-model boundaries for each microservice | Microsoft Learn," 13 4 2022. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/identify-microservice-domain-model-boundaries. [Accessed 10 5 2023].

[12] M. Fowler, "Microservices and the First Law of Distributed Objects," 13 8 2014. [Online]. Available: https://martinfowler.com/articles/distributed-objects-microservices.html. [Accessed 10 5 2023].