



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

---

# Exploring the Efficacy of ChatGPT in Generating Requirements: An Experimental Study

Bachelor of Science Thesis in Software Engineering and Management

LEILA BENCHEIKH

NIKLAS HÖGLUND

---

Department of Computer Science and Engineering  
UNIVERSITY OF GOTHENBURG  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2023



The Author grants to University of Gothenburg and Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let University of Gothenburg and Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

**Experimental investigation on the efficacy of ChatGPT in generating requirements including the freemium and premium versions**

An In-depth Investigation of ChatGPT's Performance in Generating Software Requirements: Assessing Quality Attributes, Improvement Potential, Version Variations, and AI Detection Accuracy

© LEILA BENCHEIKH, June, 2023.

© NIKLAS HÖGLUND, June, 2023.

Supervisor: JENNIFER HORKOFF

Examiner: HANS-MARTIN HEYN

University of Gothenburg  
Chalmers University of Technology  
Department of Computer Science and Engineering  
SE-412 96 Göteborg  
Sweden  
Telephone + 46 (0)31-772 1000

# Exploring the Efficacy of ChatGPT in Generating Requirements: An Experimental Study

Leila Bencheikh

*Department of Computer Science and Engineering*  
*University of Gothenburg*  
Gothenburg, Sweden  
bencheikh.leila92@gmail.com

Niklas Höglund

*Department of Computer Science and Engineering*  
*University of Gothenburg*  
Gothenburg, Sweden  
gushognia@student.gu.se

**Abstract**—This thesis explores the efficacy of ChatGPT in generating software requirements and compares its performance to human participants through an experimental study. The study addresses three main research questions (RQs), examining how ChatGPT-generated requirements align with human-written requirements, the variation in quality between different versions of ChatGPT using two additional sub-questions that look at improvement in quality from feedback and consistency of quality when the same prompt is queried multiple times, and the capacity of the Content at Scale AI detector in identifying AI-generated requirements.

Our findings reveal that ChatGPT, in both its free and premium versions, holds potential for generating software requirements, albeit it lags behind human experts in some quality attributes. However, the use of feedback mechanisms helps enhance the AI's performance. We also found that the premium version of ChatGPT outperforms its free counterpart in consistency and overall quality, except in an initial trial where the free version showed superior performance on certain attributes.

ChatGPT demonstrated remarkable time efficiency compared to human participants without compromising significantly on the quality of generated requirements. Despite this, requirements produced by experienced human practitioners were found to be more detailed and comprehensive. This suggests that AI tools like ChatGPT should supplement rather than replace human expertise in requirement engineering.

Our study also assessed the Content at Scale AI detector's proficiency in recognizing AI-generated requirements, uncovering high precision but low recall. This highlights a need for improved detectors for reliable identification of AI-generated requirements.

In summary, this thesis underscores the importance of AI tools like ChatGPT in efficient requirement generation while emphasizing the irreplaceable value of human expertise and feedback mechanisms in optimizing AI's performance. Furthermore, it identifies a crucial need for the development of enhanced AI detectors for the accurate identification of AI-generated requirements.

**Index Terms**—ChatGPT, AI-detector accuracy, requirements, experimental study.

## I. INTRODUCTION

Over the last two decades, the innovation and progress in the field of Machine Learning (ML) has been dramatic, and the field has moved on from being a topic only for research in a laboratory to have different algorithms and concepts in an extensive amount of commercial products [2]. There has been a significant increase in investment by companies in the development of ML models in recent years [1].

Large Language Models (LLM) are a type of deep learning AI that generates natural language text and has become more sophisticated in recent years [3]. Among LLMs, GPT-3 and GPT-4 have gained exponential interest from the public, as evidence by an increase in Twitter mentions compared to other popular models [4].

OpenAI is an AI research company backed by Microsoft that in the end of November 2022 released an AI chat robot they named ChatGPT that uses the LLM GPT-3, which we will use to answer our research questions since it is the most popular and advanced model of this type. GPT is a model built on top of a previous large language model from the same company called GPT-3 that at the time of its release in November 2020 was one of the most advanced models, with ten times more parameters than any previous non-sparse language model with 175 billion parameters [6]. Models like ChatGPT make logical decision based on the type of information the model was trained for. This should mean that it could easily understand logical problems such as mathematic ones that require one solution than a more sophisticated problem where the answer could be tackled from different perspectives, for instance, a political opinion [5].

The use of ChatGPT and its rise in popularity have shown potential in different fields and situations. Concretely, ChatGPT has shown its potential for example in one study where ChatGPT managed to place itself within the passing range of the United States Medical Licensing Exam [7]. It also achieved passing results on a waiver exam for the Operations Management course at the Wharton School of the University of Pennsylvania. This type of course is typically included in a master's degree on business administration at the university. ChatGPT also managed to create exam questions deemed usable and relevant for the same course [52]. These positive results of answering questions on a high educational level and creating relevant text for a specific context raise questions on what other use cases ChatGPT could have.

Requirements engineering (RE) is a critical phase in software development, and its efficacy directly impacts the quality and success of the software system. The process involves identifying, documenting, and analyzing the user's needs and requirements to develop software that meets their expectations. Traditional requirement gathering methods lean on face-to-

face interactions, surveys, and feedback mechanisms that are time-consuming, costly, and prone to errors. Recent advancements in AI and natural language processing (NLP) facilitated entrusting automated requirement gathering methods that can improve the efficiency and accuracy of the process [44].

The use of Artificial Intelligence (AI) has in recent years extended to software engineering field. ChatGPT has been utilized in various ways in software development, including bug fixing and coding. As a language model, ChatGPT has the ability to understand natural language queries and provide responses that can help developers identify and fix bugs more efficiently. In coding, ChatGPT can generate code snippets based on natural language descriptions of desired functionality, thereby reducing the time and effort required for coding. Research studies have shown promising results in using ChatGPT for software engineering tasks, including bug fixing [37] and code generation [36]. Automated Program Repair (APR) systems, on the other hand, are tools and techniques that use automated methods to fix bugs and errors in software code. Although APR systems can produce promising results, the involvement of human input can significantly enhance their capabilities, and results show that ChatGPT can be a useful tool to facilitate this process [37].

The investigation of ChatGPT's effectiveness in generating requirements is closely linked to automated program repair and debugging. Understanding a software's functionalities and intended results is crucial for successful debugging and fixing. Requirements represent these functionalities and are typically defined to specify desired behavior and features. If ChatGPT's natural language processing capabilities enable it to interpret descriptions of functionality, this raises our curiosity about whether it can generate requirements effectively. This investigation aligns with the goals of automated program repair and debugging, exploring the potential of using natural language descriptions to automate or assist in requirement formulation and improve the efficiency of software development.

Using ChatGPT in software engineering is still a relatively new field of study, but its potential for enhancing productivity and efficiency in software development is increasingly being recognized [39].

Several studies have explored the use of machine learning (ML) techniques, including Support Vector Machines (SVM) [49], bag-of-words (BoW) [42], and deep learning, in automating requirements engineering tasks such as elicitation, classification, and specification [40]. These methods have demonstrated improved accuracy and efficiency in the requirements engineering process [42]. However, there are also limitations to their use, such as the need for extensive training data [40], a limited set of requirements, and gaps in the literature on criteria selection for ML techniques [45] and the relevance and suitability of datasets [45].

Despite the potential of ML techniques in automating certain aspects of requirements engineering [45], they cannot replace the manual work needed to create sound software requirements specifications that stakeholders can use.

The aim of this study is to assess the effectiveness of

ChatGPT in generating requirements by comparing the quality of AI-generated requirements with those written by humans, furthermore, this study aims to examine the potential impact of human involvement on the results, specifically in terms of identifying any substantial improvements. To achieve this, we conducted an experimental study to evaluate the quality of requirements using IEEE's existing quality attributes as metrics before and after human involvement.

## II. RELATED WORK

To investigate the extent to which ChatGPT can generate complete requirements for software development projects, previous studies on ChatGPT capabilities, requirements elicitation and the use of AI and ML in RE are relevant.

### A. AI

In order to highlight potential benefits and limitations of using ChatGPT for requirements generation, this subsection provides an overview of some existing language models and their abilities in addition to ChatGPT.

#### 1) *LLM*:

Wei et al. demonstrated that the performance of LLMs is significantly improved as the size of the model and amount of data increase [8]. However, training LLMs is time-consuming and labor-intensive, training LLMs requires a lot of time and effort from the researchers involved in the project. Additionally, significant computational resources and energy usage are required to train LLMs due to the large amount of data and complex algorithms involved in the training process [8]. OpenAI's GPT-3, for example, has 175 billion parameters and was pre-trained on 45 TB of text data with supercomputers, costing 12 million dollars [9].

Artificial General Intelligence (AGI), also known as Strong AI or Human-Level AI, is a type of hypothetical computer system that possesses the ability to perform any intellectual task that a human can do [11]. LLMs offer a possible approach to Artificial General Intelligence (AGI) and have been explored by many organizations to promote the future development of AI including: Google by releasing Switch-Transformer [10], Baidu by releasing ERNIE 3.0 [12] and Huawei by releasing Pangu [13] in addition to the chatbot Bard released by Google after OpenAI.

To further improve the abilities of LLMs, more strategies, such as code pre-training [14], instruction tuning [15], and reinforcement learning from human feedback [17], have been used to enhance reasoning, long-range context modeling, and task generalization. As LLMs continue to advance, they offer potential applications for trustworthy AI, conversational search engines, and AGI, as demonstrated by Google's chatbot Bard and ChatGPT's success [16].

#### 2) *ChatGPT*:

ChatGPT has become a highly successful consumer application, with over 100 million monthly active visitors within two months of its launch [18]. This success is due to its ability to deliver high-quality conversations, answer follow-up questions, challenge incorrect premises, and admit its

mistakes [19]. ChatGPT has several emergent abilities, including high-quality conversation, complex reasoning, chain-of-thought (CoT), zero/few-shot learning, cross-task generalization, and code understanding/generation [20]. These abilities are made possible by the extensive training of neural network models using language models on large-scale textual data [21].

In their article, Zhou et al. discuss the potential and prospects of ChatGPT as a conversational AI model that has improved on previous models [22]. They highlight the main advantages of ChatGPT, including generalization, correction, safety, and creativity [22]:

- **Generalization:** ChatGPT uses previous conversational contexts to answer hypothetical questions, and can learn task generalization through instruction tuning.
- **Correction:** ChatGPT can admit mistakes and optimize answers according to user feedback.
- **Safety:** ChatGPT can reject unsafe questions or generate safe responses with ethical and political considerations. Supervised instruction tuning informs the model which answers are more reasonable and provides reasons with the answer.
- **Creativity:** ChatGPT excels in creative writing tasks and can polish its writing step by step, including brainstorming, story/poem generation, and speech generation.

The authors provide the preliminary details of ChatGPT, including its origin from GPT-3 and the three essential strategies used to arrive at ChatGPT [22]. These strategies include code pre-training, instruction tuning, and Reinforcement Learning from Human Feedback (RLHF) [22], which are relevant to investigate the improvement of requirement quality after human feedback.

Focusing on previous studies that have specifically explored the use of ChatGPT highlights different aspects of ChatGPT's performance, in order to provide background information on the effectiveness of ChatGPT in other areas, as it also highlights potential advantages and challenges that may arise of using ChatGPT for requirements generation.

## ***B. AI for SE***

To comprehend the application of AI in software engineering, it is crucial to provide an overview of how AI and related techniques have been utilized in SE, including the stages of the software development life cycle where AI can be implemented. Additionally, a comparison between ChatGPT and a specific LLM in the context of bug fixing is presented, offering insights into how ChatGPT's performance stacks up against other language models.

### *1) Overview on how AI is applied in SE:*

Barenkamp et al. analyzed the relevance of AI in software engineering by combining a systematic review of previous research and qualitative interviews with software developers [26]. The study identifies the automation of routine jobs, analysis of big data pools, and systematic evaluation of data in neural networks as major achievements and future potentials of AI in software engineering. The authors utilized a mixed-method approach to comprehensively assess the opportunities,

limitations, and potential risks of AI applications at the six stages of the software development life cycle. The approach involved a systematic review of more than 60 publications and qualitative expert interviews. The review evaluated experiences with AI applications at each stage of the life cycle [27], while the interviews helped to develop categories and explore issues that are not yet established. According to their results, the authors found that AI can speed up development processes, reduce development costs, and increase efficiency gains. While AI currently depends on man-made structures and is mainly reproductive, the automation of software engineering routines can enhance human developers' creative potential.

According to Barenkamp et al., the use of AI in software engineering life cycle includes the following stages: software project planning, problem analysis, software design, software implementation, software testing and integration and software maintenance [26].

AI could become an integral part of the software engineering life cycle, providing automation and optimization to different stages of the process [26]. In the planning stage, AI can assist in analyzing data, estimating project timelines [28], to predict human interactions and the environment in software project operations for optimal task assignment and team performance [29]. An example is Forecast, an AI-powered project management tool that predicts task durations, identifies risks, and suggests solutions [31].

During problem analysis, AI can be utilized to calculate risk weights and cluster risk factors in a dynamic process. In addition, hybrid technology road mapping, an AI-based planning device, can be used to assess the potentials and risks of new technologies and has been successfully applied in the analysis and selection of software projects [30].

In software design, AI can generate code, optimize algorithms, and suggest design improvements based on user behavior data [26]. Sketch2Code is an AI tool that converts hand-drawn interface designs into functional code [32].

In software implementation, AI can automate repetitive tasks like testing and debugging, and optimize performance and resource utilization [26]. Google's TensorFlow library is an example that provides tools for automating certain programming tasks [33].

In testing and integration, AI can automate testing, identify potential issues, and ensure compatibility across different systems [26]. Testim uses AI to generate test cases, identify bugs, and prioritize testing efforts [34].

In software maintenance, AI tools can support the maintenance and updating process of software in an internet environment [26]. They can also assist in the classification of user queries to direct users based on their query pattern [26]. Logz.io is an AI tool that analyzes log data to identify potential issues before they cause system failures [35].

Integrating AI into the software engineering life cycle can help software teams work more efficiently, reduce errors and costs, and deliver better software products that meet user needs [26].

### *2) Codex Vs ChatGPT in bug fixing:*

Codex is an AI system developed by OpenAI that uses deep learning to generate code [36]. Codex was trained on a massive dataset of open-source code repositories and has the ability to generate code in multiple programming languages based on natural language prompts provided by the user and dedicated deep learning-based automated program repair (APR) methods [38] on a standard benchmark dataset [36]. APR systems have been developed to suggest software patches automatically for detected errors [38]. In a research that has been conducted to analyse the automatic bug fixing performance of ChatGPT comparing it to Codex, the findings reveal that ChatGPT exhibits comparable performance to Codex in bug fixing [37]. Notably, ChatGPT outperforms APR, fixing 19 out of 40 bugs compared to only 7 [37]. When utilizing ChatGPT's dialogue feature and providing additional information about the bug in a follow-up request, its performance improves further, achieving an overall success rate of 77.5% [37]. These results demonstrate that human input can significantly enhance the capabilities of an automated APR system, with ChatGPT serving as a useful tool to facilitate this process [37].

Including the "Codex vs. ChatGPT" section exemplifies two key points in relation to the thesis. Firstly, research findings demonstrate ChatGPT's comparable bug fixing performance to Codex, a system specifically designed for code generation, additionally, that ChatGPT surpasses traditional APR systems in fixing a higher number of bugs, emphasizing its effectiveness in this task and its potential as a valuable tool within the software engineering life cycle stages.

Furthermore, the discussion on AI tools for requirement elicitation and classification piques our curiosity regarding ChatGPT's performance in generating requirements. The positive outcomes observed in bug fixing indicate that ChatGPT could potentially excel in other software engineering tasks, including requirement generation. This suggests that ChatGPT, with its language generation capabilities, holds promise for supporting various stages of the software development process.

Secondly, the study highlights that human involvement, such as providing additional bug information, enhances ChatGPT's bug fixing performance. This underscores the significance of human expertise in improving the capabilities of automated systems.

### ***C. AI and ML for RE***

In a survey, Liu et al. highlight the importance of RE in the software development life cycle and discusses the role of AI in improving the quality of requirements [44]. The authors review recent studies on how AI techniques such as ML, classification, and NLP have advanced the field of RE. They note that applying suitable ML and NLP techniques to textual requirements can help derive further meaning and improve the performance of RE. The study suggests that the integration of AI into the RE process can help requirement engineers focus on creativity rather than mundane tasks. Liu et al. study is directly related to the topic of our research paper as they focus specifically on how AI could be used in the RE process. The authors highlight different approaches, such as natural

language processing (NLP) and machine learning (ML), and their applications in different RE tasks.

The next studies and works in this section center on research that utilizes ML techniques in RE. They showcase various ML applications in RE, such as predicting requirement changes and identifying similar requirements. Additionally, relevant works on requirement elicitation and classification are included to address our research question on generating and classifying requirements as FRs and NFRs.

Zamani et al. explore the use of ML in RE and its effectiveness in improving the RE process and artifacts [45]. A mapping study was conducted on empirical studies published between 2010 and April 2020 revealing that ML is effective in automating RE analysis tasks, reducing complexity, cost, and time. However, there are gaps in the literature, specifically in selecting criteria for ML techniques and determining the relevance and appropriateness of datasets. Change impact analysis and prediction modelling in RE are under-explored areas.

A systematic mapping study on NLP for RE has been conducted [46] but differs from this study [45] as it only focuses on NLP, while this study [45] is about ML in RE.

In their article, Zhao et al. report on the mapping study of Natural Language Processing for Requirements Engineering (NLP4RE), which applies NLP techniques to the requirements engineering process [46]. The study identified 404 primary studies published over 36 years, with most of the studies focusing on requirements analysis, quality defect detection, and requirements specification [46]. The study also extracted 130 RE-specific NLP tools and identified 231 NLP technologies, with traditional analysis techniques, general-purpose tools, and generic language lexicons being commonly used. The study provides a conceptual framework for NLP4RE, identifies research gaps and directions, and encourages more synergies between the RE and NLP communities [46].

#### *1) ML for user stories:*

MUCE (ML-based User story Creation from Existing stories) is a tool developed and built for research purposes [23]. It uses ML to generate user stories for agile software development. It is related to the research question of whether ChatGPT and similar AI technologies can generate accurate and complete requirements for software development projects because both MUCE and ChatGPT are using ML techniques to automate parts of the requirements elicitation process. The use of MUCE demonstrates the growing interest in using ML to automate the requirements elicitation process in software development.

The investigation of the use of ML in generating user stories is related to the research topic since the user stories are a type of requirements.

#### *2) RE Classification:*

This subsection focuses specifically on studies that have explored classification techniques in RE, including those related to requirement categorization.

Despite efforts to automate the classification of requirements into Functional Requirements (FR) and Non-Functional Re-

quirements (NFR) [51], current methods have not yet proven efficient enough to fully replace the manual work required to create high-quality software requirements specifications that meet stakeholders' needs [41]. Quba et al. [41], tried to combine the BoW (bag-of-words) technique which is a type of technique extracting features through vectorization [42] from text, and apply this technique separately with two ML algorithms, Support Vector Machine (SVM) or k-nearest neighbors (KNN) to see which one performs better in classifying requirements. They believe that using BoW together with SVM algorithm can help programmers for classification of requirements. The article does not make any statement if their approach is better than manual work but is relevant as it shows that there is hope that ML will be useful for requirements elicitation and that to automate the process of generating/categorize software requirements automatically is a relevant topic closely related to requirements generation.

In their paper [47], Abad et al. investigate automated classification of requirements into functional and non-functional categories, as well as sub-classifying non-functional requirements into usability, availability, or performance categories. The authors propose a pre-processing approach to standardize and normalize requirements, which improves the performance of existing classification methods. They also compare the performance of several machine learning algorithms for sub-classifying non-functional requirements and find that Binarized Naïve Bayes performs best. The study concludes that additional or better sentimental patterns and sentence structures are needed for differentiating usability requirements from other types of NFRs.

#### 3) *Automation of RE specification:*

Panichella and Ruiz address the challenge of expediting the time-to-market of high-quality software products during digital transformation in their paper [43]. They introduce Requirements-Collector, a tool that uses ML and deep learning for automating requirements specification and user feedback analysis in software development. The tool can classify requirements from audio recordings of RE meetings and textual feedback from user reviews with reliable accuracy. The study aims to automate requirements specification and user feedback collection to reduce time-to-market for software products using a virtual reality requirements engineering room. Participants can discuss requirements that are automatically specified as user stories, and user reviews are automatically analyzed. The tool incorporates virtual reality tools and interactive boards for real-time editing of requirements. The study also presents transcript and user review classifiers to support the automatic requirements collector tool. The authors show that the use of ML and DL computational mechanisms is effective in text classification problems. The tool has the potential to reduce manual tasks, improve communication, and ensure software quality from the conception phase. The authors used the Requirement-Collector tool to compare the performance of over ten machine learning models and one deep learning method in classifying transcripts and user feedback. Preliminary results suggest that deep learning methods are more

appropriate for classifying transcripts, achieving an average F-measure of 33%. The best performing ML model (SMO) achieved an F-Measure of 77% for classifying user feedback from user reviews [43].

#### 4) *Automation of both: RE elicitation and classification:*

Surana et al. introduced a conversational chatbot and text classification algorithms to automate requirements elicitation and classification together [40]. The chatbot uses several ML algorithms, including Natural Language Understanding, Natural Language Generation, Text Pre-processing, and Conditional Random Field for Named Entity Recognition to elicit requirements. The elicited requirements are saved to a file, and a text classification model is developed using the Bag-of-Words approach, Term Frequency-Inverse Document Frequency (tf-idf) [50], and Multinomial Naive Bayes (NB) [40], a popular ML algorithm used for classification tasks, particularly in Natural Language Processing (NLP) [48]. The process also includes the use of Support Vector Machine (SVM) [49], a type of supervised machine learning algorithm used for classification and regression analysis. Both algorithms were used to classify the elicited requirements into Functional and Non-functional categories after elicitation, and the chatbot was found to retrieve information from users accurately. The proposed approach has limitations, including a limited set of requirements and the need for extensive training data. However, it can identify incomplete descriptions of system requirements and accurately classify them.

### D. Conclusion

ChatGPT is a relatively new technology that has gained popularity in recent years. However, there is still a lack of research conducted on ChatGPT. While some studies have been conducted on the broader topic of natural language processing and machine learning, there are few that focus specifically on ChatGPT and none were conducted on its capabilities in generating requirements.

This research is important as it fills a gap in the current knowledge and understanding of ChatGPT's abilities in generating requirements. By exploring the potential of ChatGPT in this area, we can gain new insights into how it can be used to streamline and improve the software development process. Additionally, this research can help to identify areas where ChatGPT may fall short, allowing for further improvements and advancements in the technology.

## III. RESEARCH METHODOLOGY

For our research we chose to use an experimental study. It is an appropriate approach to explore the efficacy of ChatGPT in generating requirements because it allows for a controlled environment to evaluate ChatGPT-generated requirements and human-written requirements. It provides a more objective approach compared to other methodologies such as surveys or case studies, which may be more subjective.

Through the study, we quantitatively measure the quality and accuracy of requirements generated by ChatGPT and compare them to human-written requirements using thematic

coding. The quality attributes we focus on align with the characteristics outlined by the IEEE standard for individual requirement. As suggested by the IEEE, each stakeholder, system, and system element requirement shall possess the following nine characteristics: Necessary, Implementation Free, Unambiguous, Consistent, Complete, Singular, Feasible, Traceable, and Verifiable [53]. These attributes, as defined in the International Standard ISO/IEC/IEEE 29148, are used as criteria to assess the quality of the requirements.

This approach gives a valuable insights into the capabilities and limitations of ChatGPT in generating requirements, and inform potential improvements or applications of the technology in the field of software engineering.

#### A. Research questions and hypotheses

There are three main research questions that guide our study which are relevant to normative research as they seek to evaluate the quality of ChatGPT generated requirements and explore ways to improve them through feedback. The RQs aim to evaluate the quality of ChatGPT generated requirements and determine the ability of an AI detector to detect AI-generated text. The relevant factors for these RQs could include the criteria for defining the quality of a requirement, the type and amount of feedback given to ChatGPT and the effectiveness of the AI detector.

For each of the research questions and sub-research questions guiding the study we have defined two-sided hypothesis, dependent and independent variables, and extraneous variables as follows:

##### **RQ1: How do requirements generated by ChatGPT compare to human-written requirements in terms of the nine requirement characteristics defined by the IEEE for an individual requirement?**

- **H0:** There is no significant difference in the quality of ChatGPT-generated requirements compared to human-written requirements, in terms of meeting the quality attributes of an individual requirement as defined by the IEEE.
- **H1:** There is a significant difference in the quality of ChatGPT-generated requirements compared to human-written requirements, in terms of meeting the quality attributes of an individual requirements defined by the IEEE.

For this research question, we have identified the following dependent, independent and extraneous variables:

- **Dependent variable:** The quality of the requirement generated (measured through a quality metric, e.g. completeness, consistency, singularity, etc.)
- **Independent variable:** The source of the requirement (ChatGPT or human-written)
- **Extraneous variables:**
  - *Experience level of human writers:* This variable refers to the experience and expertise of the human participants writing the requirements. It could influence the quality of the requirements and introduce

bias into the comparison. To control for this variable, the human participants who wrote the requirements were selected based on their background and experience in software engineering with at least 1 year of professional experience in the field of SE.

- *Prompt and instructions:* This variable refers to the difficulty level of the instructions. Complex and different instructions may impact the accuracy of results. Both ChatGPT and human participants were provided with the same prompt and instructions to ensure consistency and fairness in requirement generation.
- *Quality assessment:* This variable refers to requirement quality interpretation which could introduce subjectivity and potentially impact the consistency and accuracy of the quality assessment of the requirements generated by ChatGPT and human participants. The requirements generated by both ChatGPT and human participants were evaluated using thematic coding based on quality categories from IEEE (ISO/IEC/IEEE 29148:2011) [53]. This coding allowed for the assessment of various quality attributes such as completeness, consistency, singularity, feasibility, traceability, and verifiability.
- *Statistical analysis:* The statistical analysis was conducted with a Mann-Whitney U test to compare the results gathered from quality evaluation of ChatGPT-generated requirements and human-written requirements to control for more precise values. The Mann-Whitney U test is non-parametric and can use ordinal data [54].

##### **RQ2: How does the quality of ChatGPT-generated requirements vary across different versions, such as premium and freemium?**

- **H0:** There is no significant difference in the quality of ChatGPT-generated requirements across different versions, such as premium and freemium.
- **H1:** There is a significant difference in the quality of ChatGPT-generated requirements across different versions, such as premium and freemium.

To investigate this research question, we have identified the following dependent, independent, and extraneous variables as well as two sub-research questions (sub-RQs) that serve as the foundation for answering RQ2.

- **Dependent variable:** Quality of ChatGPT-generated requirements (measured through a quality metric).
- **Independent variable:** Version of ChatGPT (premium vs. freemium)
- **Extraneous variables:**
  - *Consistency and complexity of the prompt:* Different and more complex prompts may be more challenging for ChatGPT to generate accurate requirements. To control for the potential influence of prompt characteristics on requirement quality, we ensured to the prompt and complexity of the prompt

remained consistent across all queries conducted in both versions.

- *The experience and expertise of the user querying the prompt:* To minimize the impact of user proficiency on requirement quality, we used the same user with consistent experience and expertise to query the same prompt for each ChatGPT version, the premium and freemium.
- *The presence of any technical issues or internet connectivity problems during the queries:* Technical issues or internet connectivity problems could cause errors or delays in generating requirements, which could impact the quality of the results. To mitigate the potential effects of technical issues or internet connectivity problems, we conducted the queries under controlled conditions. By ensuring a stable and reliable testing environment, we minimized disruptions that could impact the quality of the generated requirements and maintained a consistent evaluation process.

The following sub-RQs, **RQ2.1** and **RQ2.2** provide a framework for examining two factors that could influence the **RQ2** and facilitate a systematic analysis of the data.

#### **RQ2.1: How does feedback given to ChatGPT impact the improvement of requirements' qualities?**

- **H0:** There is no significant improvement in the quality of requirements generated by ChatGPT after the provided feedback.
- **H1:** There is a significant improvement in the quality of requirements generated by ChatGPT after the provided feedback.

For this sub-research question, we have identified the following dependent, independent and extraneous variables:

- **Dependent variable:** The quality of the requirement generated (measured through a quality metric).
- **Independent variable:** Feedback (presence or absence of feedback given to ChatGPT)
- **Extraneous variables:**
  - *Quality of feedback given to ChatGPT:* The quality of the feedback given to ChatGPT could influence the quality of the requirements it generates. If the feedback is unclear or incorrect, ChatGPT may not learn from it and may continue to generate low-quality requirements. To manage this variable, we ensured that the feedback given to ChatGPT was clear, accurate, and consistent as demonstrated in the *Figures 1 and 2.*

#### **RQ2.2: What is the impact of querying the same prompt multiple times on the quality of requirements generated by ChatGPT?**

- **H0:** There is no significant difference in the quality of requirements generated by ChatGPT for the same prompt when queried multiple times.

- **H1:** There is a significant difference in the quality of requirements generated by ChatGPT for the same prompt when queried multiple times.

For this sub-research question, we have identified the following dependent, independent and extraneous variables:

- **Dependent variable:** The quality of requirements generated by ChatGPT (measured through a quality metric).
- **Independent variable:** The number of times the same prompt is queried on ChatGPT.
- **Extraneous variables:**
  - *Version of ChatGPT:* Different versions of ChatGPT may have varying levels of accuracy and performance, which could impact the quality of generated requirements. To control for variations in accuracy and performance, we assigned each researcher to a specific version of ChatGPT. One researcher was in charge of the Free version based on GPT-3, while the other researcher handled the Premium version based on GPT-4. This ensured consistency in the testing environment and eliminated the potential influence of different versions on the quality of generated requirements.
  - *The length and complexity of the prompt:* We carefully maintained the length and complexity of the prompt across all queries in both versions of ChatGPT. This approach aimed to minimize any bias introduced by variations in the prompt's characteristics, ensuring that the comparison focused solely on the effect of querying the same prompt multiple times.
  - *The experience and expertise of the user querying the prompt:* Users with more experience and expertise in the domain may be better able to craft effective prompts that elicit high-quality requirements from ChatGPT. To control for the impact of user proficiency on requirement quality, we used the same experienced user for each version. Each user possessed consistent experience and expertise in the domain and was familiar with crafting effective prompts that elicited high-quality requirements from ChatGPT. By using the same user and prompt, we reduced the potential influence of user variability on the generated results.

#### **RQ3: To what extent can an AI detector differentiate between ChatGPT generated requirements and human-written requirements?**

- **H0:** The Content at Scale AI detector, specialized in detecting AI-generated text, is not able to detect requirements generated by ChatGPT.
- **H1:** The Content at Scale AI detector, specialized in detecting AI-generated text, is able to detect requirements generated by ChatGPT.

For this research question, we have identified the following dependent, independent and extraneous variables:

- **Dependent variable:** The detection accuracy of Content at Scale AI detector. (measured using precision, recall and F1-score on ChatGPT-generated requirements)
- **Independent variable:** The source of the requirement (ChatGPT-generated or human-written)
- **Extraneous variables:**
  - *Length of the requirements:* The length of the requirements could affect the detector’s ability to detect them. Shorter requirements may be more difficult to detect accurately. By using a set of 120 requirements in total. This set included contributions from 6 participants, with each participant providing 10 requirements, in addition to the requirements generated from 3 iterations of both the premium and freemium versions of ChatGPT, making a total of 60 REs from ChatGPT. The set of 120 were tested at once.

### B. Relationships between RQs

The research questions in this study explore various aspects of ChatGPT-generated requirements and their quality compared to human-written requirements, as well as the potential for improvement and detection.

**RQ1** aims to compare the quality of requirements generated by ChatGPT to those produced by humans. The evaluation process involves analyzing the collected requirements from human participants alongside those generated by ChatGPT.

**Sub-RQ2.1** investigates whether the quality of requirements improves when feedback is provided to ChatGPT. Feedback is given based on the findings from **RQ1**, with the intention of enhancing the quality of the generated requirements for both the freemium and premium versions of ChatGPT.

**Sub-RQ2.2** explores whether the quality of requirements generated by ChatGPT improves when the same prompt is queried multiple times. By repeating the prompt and analyzing the data obtained from ChatGPT’s requirement quality evaluation in **RQ1**, comparisons are made to identify any variations in quality between the freemium and premium versions.

**RQ2** examines the variation in the quality of ChatGPT-generated requirements across different versions, including premium and freemium. The results from **RQ2.1** and **RQ2.2** are analyzed to determine if there are significant differences in quality among these versions.

In **RQ3**, the focus is on assessing the performance of the Content at Scale AI detector, specialized in detecting AI-generated text, in identifying requirements generated by ChatGPT. The data collected in **RQ1** and **RQ2** are utilized to evaluate the detector’s effectiveness in detecting ChatGPT-generated requirements.

### C. Participation and data collection

- **Participants:** The participants were recruited from software engineering students that have professional experience within Software Engineering program as teaching assistants and professionals with at least one year of experience in writing software requirements.

- **Background:** The participants have knowledge of software development and have experience in software requirements engineering, including how to write clear, concise, and complete requirements following IEEE requirement quality.

The experiment consists of twelve phases as follows:

#### 1) *Phase 1 - Meeting participants:*

The experiment involved a mix of software developers and individuals with experience in software development and requirement elicitation. Eleven human participants in total took part of the experiment, five of the participants are professionals in the industry within the field of software engineering, one researcher expert in RE, while five are bachelor students that have experience in writing software requirements specifications and experience as teaching assistants within Software Engineering and Management Bachelor’s program offered by the University of Gothenburg. In addition, the experiment involves using both versions of ChatGPT, the AI chat-bot model, the freemium (Model GPT-3) and the premium (Model GPT-4).

The participants were asked to write requirements for a hypothetical software project that gives an overview of a domain problem description that needs solving. The hypothetical software project is an application designed to handle emergencies and disasters in Canada originally which we adapted to be about Sweden that can be read in appendix A.

The hypothetical software project is one of the domain problem descriptions written by Jennifer Horkoff, for the DIT046 course at Gothenburg University, who agreed upon using them. The task itself was designed to be completed within a time frame of less than 30 minutes to ensure that the task was feasible and could be comfortably accomplished within a reasonable duration, out of consideration of our participants professional engagements or studies.

**Pilot experiment:** We conducted a pilot experiment with one participant prior to the main study. The purpose of the pilot experiment was to test the feasibility and clarity of the experiment design, task instructions, and any necessary clarifications. By involving a single participant, we could make sure that the task is feasible within the 30 minutes time frame before proceeding with the main study.

**Initial Meeting:** Prior to the start of the experiment, a meeting was organized with each participant. During this meeting, we provided a detailed explanation of the experiment, including the purpose, objectives, and tasks involved. Participants had the opportunity to ask questions and seek clarification to ensure a clear understanding of their roles and responsibilities. The initial meeting conducted with each participant lasted approximately 10 minutes. During this meeting, we provided an overview of the experiment, including the task, objectives, and expectations. Participants had the opportunity to ask questions and seek clarification to ensure a clear understanding of the requirements generation task.

**Participant Form:** To gather relevant background information and ensure consistency in data collection, participants were asked to complete a form. This form included sections

where participants could provide details about their occupation, experience in software engineering, and confidence in writing requirements. Additionally, the form included a description of the problem, along with explanations differentiating functional requirements (FR) and non-functional requirements (NFR). Furthermore, the form included a list of quality attributes (Appendix B) and the rationale for their importance. Participants were asked to note the time of starting the task. The form also provided a structured format for writing requirements, with separate sections for the requirement description and rationale (Appendix C).

Providing participants with a reminder of the quality attributes and the difference between FR and NFR is important to ensure that everyone involved in the experiment is on the same page. This helps maintain consistency in the evaluation process, which contribute to the overall quality and reliability of the generated requirements in the experiment.

**Ethical Considerations:** Participants were informed about the purpose of the experiment and their consent was obtained before the experiment started. The data collected during the experiment was anonymized and kept confidential. Participants were free to withdraw from the experiment at any time without penalty.

2) *Phase 2 - Prompt ChatGPT freemium version:*

In *Phase 2* of the experiment, we continued our investigation by interacting with the free version of ChatGPT, which is powered by the GPT-3 language model. The objective was to explore the capabilities of ChatGPT in generating qualitative requirements and distinguishing between functional requirements (FRs) and non-functional requirements (NFRs) for the same software project.

Similar to *Phase 1*, we provided the necessary prompts and instructions related to the project to ChatGPT. The model then generated a set of 10 requirements based on the input it received. Our focus was not only on the overall quality of the generated requirements but also on ChatGPT’s ability to accurately identify and classify them as either FRs or NFRs.

Table I presents the prompt used to interact with ChatGPT and the corresponding responses generated by the model. The symbols used to differentiate the user’s input from ChatGPT output:

- >: User prompt
- #: ChatGPT response

The requirements gathered from these interactions are provided in appendix D for reference.

By evaluating the outputs of ChatGPT, we aimed to assess its performance in requirements generation and its understanding of the distinction between FRs and NFRs. This analysis provides insights into the effectiveness and potential limitations of the free version of ChatGPT in meeting the quality criteria for software requirements.

3) *Phase 3 - Prompt ChatGPT premium version:*

In *Phase 3* of the experiment, we extended our investigation by engaging with the premium version of ChatGPT, which utilizes an advanced language model, including the recently released GPT-4. This version of ChatGPT is known for its

<p>&gt;Hi ChatGPT I have a task for you.</p> <p># Hello! I'm happy to help with any task you have. Please let me know what you need assistance with.</p> <p>&gt;I want you to write 7 functional requirement and 3 non-functional requirements based on a problem description and the qualities of a good requirement for a software system that I will soon type to you. A functional requirement defines what the system must do in order to work correctly with the desired functionality and how it responds to certain inputs within the system. In other words, the features or capabilities of the system. On the other hand, a non-functional requirement is used to describe system quality which could be tied to the performance or fault tolerance for example to improve the quality of the software for the end user. In other words, how well something is done within the system.</p> <p># Sure, I'm happy to help you with that task. Please provide me with the problem description and the qualities of a good requirement for a software system. I will use that information to write the 7 functional requirements and 3 non-functional requirements you need.</p> <p>&gt;The 9 qualities of a good requirement that I want you to keep in mind when writing requirements based on the problem description I soon will write to you are the following.</p> <ol style="list-style-type: none"> <li>1. Necessary: The requirement is essential for the capability of the system and should be realistic to implement from the constraints of the product.</li> <li>2. Implementation Free: Implementation independent, which means not saying how a system should do a requirement from a technical standpoint but still be necessary.</li> <li>3. Unambiguous: The requirements should only be able to be interpreted one way</li> <li>4. Consistent: Not conflicting with other ones or be the same thing with different words</li> <li>5. Complete: The requirement is measurable and containing all the information needed for it to be implemented correctly</li> <li>6. Singular: The requirement should not be a combination of many requirements</li> <li>7. Feasible: Should be realistic for the program (based on context)</li> <li>8. Traceable: Can you understand why the requirements is there (In this case the problem description)</li> <li>9. Verifiable: The final entity should be testable. For example, good is hard to test</li> </ol> <p># Thank you for providing the qualities of a good requirement. Please provide me with the problem description so that I can begin writing the functional and non-functional requirements.</p> <p>&gt;When writing the requirement I want to have a clear and concise description and the reason behind the requirement as well written in the form of,  Requirement Number: Id starting from 1  Description: Requirement  Rational: Reason behind requirement</p> <p># Thank you for providing the format. Please provide me with the problem description so that I can begin writing the functional and non-functional requirements.</p>
--

Table I: Prompt used to interact with ChatGPT and responses provided

enhanced capabilities in generating creative and high-quality outputs.

The purpose was to assess the potential improvements in ChatGPT’s performance in generating high-quality requirements and distinguishing between FRs and NFRs for the same software project.

Just like in *Phase 2*, we provided the same prompts and instructions related to the software project to ChatGPT. The premium version of ChatGPT then generated a new set of 10 requirements based on the provided input. Our emphasis was not only on the overall quality of the requirements but also on ChatGPT’s accuracy in correctly categorizing them as FRs or NFRs.

By comparing the results from *Phase 2* (free version) and *Phase 3* (premium version), we aimed to gain insights into the potential impact of using the premium version of ChatGPT on the quality and classification of the generated requirements. This comparison will provide valuable information on the added benefits and relevance of upgrading to the premium version of ChatGPT for requirements generation tasks.

4) *Phase 4 - Evaluation of Requirement Quality:*

In *Phase 4* of the experiment, we evaluated the quality of the requirements provided by both the participants and ChatGPT (Free and Premium Versions) gathered in the *Phases 1, 2 and 3*, using thematic coding based on the quality categories from the IEEE standard ISO/IEC/IEEE 29148:2011 for individual requirement [53]. In addition to the coding process, we employed a scoring method to determine whether the quality of each requirement was verified or not. In addition, we have collected data regarding time spent by the human participants in writing the form provided -including answering background questions, reading description and writing REs. We also timed both ChatGPT version so see interaction time for generating REs.

**RE quality evaluation:** For each quality attribute, including: Necessary, Implementation Free, Unambiguous, Consistent, Complete, Singular, Feasible, Traceable and Verifiable we assigned a score of 1 if the requirement fully met the quality criteria, indicating that it was verified. Conversely, a score of 0 was assigned if the requirement did not meet the quality criteria, indicating that it was not verified as shown in *Table II*. To simplify the visualization in the table, each attribute was represented by a number for brevity, as the table exceeded the document’s size as follows: 1) Necessary, 2) Implementation Free, 3) Unambiguous, 4) Consistent, 5) Complete, 6) Singular, 7) Feasible, 8) Traceable and 9) Verifiable.

RE ID	Type	Description	Rationale	Quality attributes										
				1	2	3	4	5	6	7	8	9		
101	FR	The system shall provide real-time info about the recurrent disaster, for instance, the location, and level of the emergency.	If a user uses the app in those critical times, they can get up-to-date accurate info about the whereabouts of the disaster, so they can avoid the area.	1	1	1	1	1	1	1	1	1	1	1

Table II: An example of a requirement evaluation

By using the scoring method, we were able to quantitatively assess the level of quality for each requirement attribute. The results of the scoring method for each attribute were recorded in a table for both, FRs and NFRs, allowing us to analyze the quality of requirements generated by the participants and ChatGPT based on each attribute.

We have conducted the evaluation of requirements quality based on the IEEE 29148:2011 standard in a collaborative manner, taking into consideration the subjective nature of judging certain quality aspects such as unambiguousness, completeness, and verifiability. Initially, all requirements were gathered in a shared document. One researcher performed the initial evaluation, while the other researcher independently reviewed the results. The goal was to ensure a comprehensive and unbiased evaluation process. In instances where there were disagreements or differences in evaluation results, both researchers met to discuss and reevaluate the requirements together. This collaborative approach allowed for the resolution of any discrepancies and the achievement of consensus on the evaluated quality attributes. To further enhance the accuracy of the evaluation, a portion of the results (20%) was independently reviewed by Jennifer Horkoff, an expert in Requirements Engineering. Her feedback was considered valuable, and any necessary adjustments were made based on her recommendations. Jennifer Horkoff confirmed that the evaluation process was on the right track, even though a few values may have changed as a result of her review. Overall, the evaluation of requirements quality was performed collaboratively, with regular discussions and expert input, to ensure accuracy and reliability in accordance with the IEEE 29148:2011 standard.

**Statistical analysis:** We used the Mann-Whitney U Test to compare scores of human written requirements and ChatGPT generated requirements for both version.

The Mann-Whitney U test, also known as the Wilcoxon rank-sum test, was chosen for this study to compare the quality of requirements generated by ChatGPT with human-written requirements. This non-parametric test is suitable for our study because it does not assume a normal distribution of data and can be used to analyze ordinal or continuous data [55].

The Mann-Whitney U test works by comparing the ranks of two independent groups, in our case, the requirements generated by ChatGPT and human-written requirements. It evaluates whether there is a significant difference in the distribution of ranks between the two groups. The test calculates a U statistic, which represents the sum of ranks for one group relative to the other, and it also provides a p-value that indicates the significance of the observed difference [56].

By using the Mann-Whitney U test, we can determine whether there is a statistically significant difference in the quality of requirements between ChatGPT-generated and human-written requirements, specifically in terms of meeting the quality attributes defined by the IEEE for an individual requirement. This test allows us to make objective comparisons and draw conclusions about the performance of ChatGPT in generating requirements [55].

**Time spent to formulate REs:** In addition to REs quality evaluation, we collected data on the time spent by human participants to write requirements as illustrated in the graph in *Figure 6*, and the time taken by ChatGPT to generate requirements.

5) *Phase 5 - Repeated Prompt on Freemium ChatGPT:*

In *Phase 5* of the experiment, we aimed to investigate the potential improvement of requirements generated by the freemium version of ChatGPT through the utilization of machine learning (ML) techniques.

To examine this, we repeated the prompt, which described the hypothetical software project, two more times on the freemium version of ChatGPT. By doing so, we sought to assess whether the ML algorithms embedded in ChatGPT would enhance the quality of the generated requirements with each iteration.

The analysis of the results from this phase will provide insights into the effectiveness of ML techniques employed by ChatGPT in refining the requirements over successive interactions. It will also contribute to our understanding of the capability of the freemium version to leverage ML algorithms for improvements in requirement generation.

#### 6) *Phase 6 - Repeated Prompt on Premium ChatGPT:*

Similarly to *Phase 5*, in this phase of the experiment, we focused on evaluating the performance of the premium version of ChatGPT in generating requirements for the same software project.

The objective was to investigate whether the enhanced features and capabilities of the premium version would lead to improved requirement generation compared to the freemium version. By utilizing the premium version, which may have access to advanced language models, more computing resources, or additional training data, we aimed to assess any potential differences in requirement quality and accuracy.

#### 7) *Phase 7 - Feedback to ChatGPT freemium on quality:*

In *Phase 7* of the experiment, we focused on providing feedback to the ChatGPT free version regarding the quality of its generated requirements. This phase was based on the evaluation of the results obtained in *Phase 4*, where we assessed the quality of requirements provided by ChatGPT.

To provide feedback, we utilized a specific prompt that allowed us to address the missing quality attributes in each requirement. The prompt was designed to highlight the areas where the generated requirements could be improved in terms of necessary attributes such as singularity, feasibility, and verifiability. The prompt used for providing feedback is depicted in *Figure 1*, where it is clearly presented for reference. The prompt served as a guiding framework to communicate to ChatGPT the specific quality attributes that were lacking in its generated requirements.

The process of providing feedback to the ChatGPT serves as a valuable step in understanding the interplay between AI and human involvement in improving the quality of requirements. It contributes to our overall analysis of the experiment's outcomes and sheds light on the potential benefits of leveraging human intervention in the requirements generation process.

#### 8) *Phase 8 - Feedback to ChatGPT premium on quality:*

In *Phase 8* of the experiment, our focus shifted to providing feedback to the ChatGPT premium version regarding the quality of its generated requirements. This phase followed a similar approach to *Phase 7*, after we evaluated the results

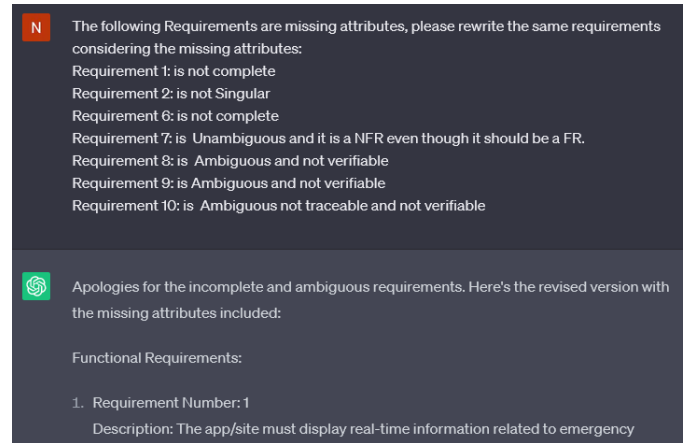


Figure 1: Prompt for Feedback: Missing Quality Attributes in ChatGPT Freemium Requirements

obtained in *Phase 4* specifically for the ChatGPT premium version.

*Figure 2* illustrates the prompt that was used during this feedback phase, providing a visual representation for reference. By utilizing this prompt, we were able to effectively communicate the identified quality attribute deficiencies to the ChatGPT premium version.

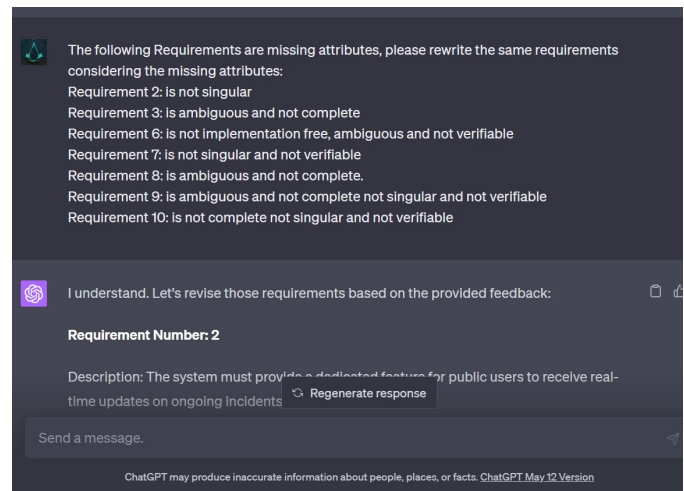


Figure 2: Prompt for Feedback: Missing Quality Attributes in ChatGPT Premium Requirements

#### 9) *Phase 9 - Quality after feedback:*

*Phase 9* involved re-evaluating the quality of the requirements provided by ChatGPT for both versions, following the feedback provided in *Phase 8*. This re-evaluation was conducted using thematic coding based on quality categories from IEEE, as explained in *Phase 4*. The purpose of this phase was to assess whether the feedback provided to ChatGPT had an impact on the quality of the generated requirements. By applying the same thematic coding approach, we could compare the updated requirements with the initial evaluation

and analyze any changes or improvements that occurred as a result of the feedback.

#### 10) *Phase 10 - Quality after repeated prompts:*

*Phase 10* involved evaluating the quality of requirements generated by ChatGPT for both the free and premium versions. This evaluation was conducted after using the same prompt two additional times. The purpose was to assess if repeated interactions with the prompt resulted in any changes in requirement quality.

#### 11) *Phase 11: Using Content at scale AI Detector:*

In *Phase 11* of the experiment, we focused on evaluating the set of requirements generated by both the ChatGPT free version and premium version in *Phases 2, 3, 5 and 6*, as well as the requirements written by six human participants gathered in *Phase 1*. To ensure fairness in the comparison, we selected these human participants (P1, P2, P6, P7, P8 and P11) based on their average to good performance in requirement quality.

#### 12) *Phase 12: Analysing the results from AI detector:*

The evaluation method we used to calculate precision, recall, and F1 score provides a quantitative measure of the performance of the Content at Scale AI detector in detecting AI-generated requirements. These metrics are widely used in information retrieval and machine learning tasks to evaluate the effectiveness of classifiers or detectors.

The precision metric measures the accuracy of the detector in correctly identifying AI-generated requirements among all the requirements classified as AI-generated. A high precision indicates a low rate of false positives, meaning that the detector has a low tendency to incorrectly label human-written requirements as AI-generated.

The recall metric, on the other hand, assesses the detector's ability to capture all the AI-generated requirements in the dataset. It measures the proportion of correctly identified AI-generated requirements out of all the actual AI-generated requirements. A high recall indicates a low rate of false negatives, suggesting that the detector can effectively identify most of the AI-generated requirements.

The F1 score combines both precision and recall into a single metric that balances the trade-off between them. It provides a comprehensive evaluation of the detector's performance, taking into account both false positives and false negatives. A higher F1 score indicates a better overall performance of the detector in accurately identifying AI-generated requirements.

Since we had a total of 60 requirements generated by each version of ChatGPT in all three iterations (10 requirements each), we decided to include an equal number of requirements from the human participants to maintain a balanced dataset. This resulted in a set of 60 human-written requirements in total.

To assess the quality and effectiveness of the generated requirements, we utilized the Content at Scale AI detector. This detector is a tool designed to detect AI-generated content. By subjecting the set of requirements to this detector, we aimed to evaluate the ability of ChatGPT in generating requirements that can be distinguished from human-written requirements.

## D. Data collection

The data collection primarily consists of:

*Qualitative data:* by gathering textual responses from the participants and ChatGPT both versions in the form of requirements. In addition, data gathered from the AI detector to verify whether ChatGPT generated requirements are detected as AI generated.

*Quantitative data:* gathered by checking how many quality attributes were met for both, human-written and ChatGPT generated requirements, using Mann-Whitney U Test to compare scores, and precision recall and F1-score to measure the accuracy of AI detector.

We have defined the exact commands used for interaction with ChatGPT to generate the requirements already in *Table I*, in addition all requirements generated are available in appendix F.

The following data has been collected during the experiment:

- a) Background information about participants in *Phase 1*
- b) Requirements written by the participants in *Phase 1*.
- c) Requirements generated by freemium version of ChatGPT in *Phase 2*.
- d) Requirements generated by premium version of ChatGPT in *Phase 3*.
- e) Requirements generated by both version of ChatGPT after feedback in *Phase 9*.
- f) Requirements generated by both versions of ChatGPT in phase 10 after 2 iterations.
- g) Results of the AI detector in *Phase 12*.

## E. Data analysis

**Thematic analysis** [24] is a good way to conduct the experiment because it allows for an in-depth exploration of the data and identification of patterns and themes that may not be immediately apparent. We already have a set of predefined themes, which are the quality attributes of the software requirements: necessary, implementation free, unambiguous, consistent, complete, singular, feasible, traceable and verifiable, see appendix B for a short description of each quality attribute. We will also use another theme for categorising Functional Requirements (FR) and Non-Functional Requirements (NFR) [51].

We conducted thematic analysis on the generated requirements from ChatGPT and the requirements we have gathered from participants in order to apply the themes help us measure the quality of the requirements. A software requirement is considered "good" if it is: necessary, implementation free, unambiguous, consistent, complete, singular, feasible, traceable and verifiable [25].

Furthermore, the use of a template for requirement quality analysis, allows for a standardized and objective evaluation of the quality of the requirements. The template includes predefined qualities used as a reference, and scores assigned based on whether these qualities are present or not in each

requirement. The scores used were 1 and 0 that correspond respectively to quality verified in the requirement and not verified

We used the **Mann-Whitney U test** in this study to compare the quality of requirements generated by ChatGPT with human-written requirements. The choice of the significance level ( $p$ -value) in hypothesis testing is a subjective decision influenced by various factors. In our study, we selected a significance level of 0.05 (95% confidence level), which is a common convention in many research disciplines. This level of significance allows for a 5% chance of making a Type I error, where we reject the null hypothesis when it is actually true. The significance level serves as a criteria value to ensure generalizability and provide relatively high confidence that the results are not due to chance.

Setting a higher significance value could risk drawing false conclusions about the applicability of using ChatGPT in requirements engineering and quality of generated requirements. It also increases the likelihood of making a Type I error.

On the other hand, choosing a lower significance level could lead to a higher risk of Type II errors, where we fail to reject a false null hypothesis. This could falsely indicate limitations of ChatGPT and impede advancements in utilizing ChatGPT for requirements engineering. Therefore, we believe that 0.05 strikes a balance, limiting both false positives and false negatives, and providing results that are reliable for decision-making. Overall, the choice of the 0.05 significance level ensures a reasonable trade-off between the risks of Type I and Type II errors, allowing for meaningful conclusions while minimizing the potential for erroneous interpretations.

#### F. Limitations

We have defined several validity threats and limitations for our experiment, as described below:

- 1) **Internal validity threats:** These are threats to the ability of the experiment to draw accurate conclusions about cause-and-effect relationships. The following internal validity threats might be present in our experiment:
  - *Selection bias:* The participants in the study are software engineering students and professionals who have experience in writing software requirements. This may limit the generalizability of the results to other populations.
  - *Maturation:* The quality of the requirements generated by ChatGPT may change over time due to factors such as learning and updates, which could affect the results.
  - *Instrumentation:* The quality metrics used to evaluate the requirements may not be sensitive enough to capture all aspects of requirement quality.
- 2) **External validity threats:** These are threats to the ability of the experiment to generalize the results to other settings or populations. The following external validity threats might be present in our experiment:
  - *Artificial task:* The requirements generated in the study may not be representative of real-world re-

quirements, which could limit the generalizability of the results.

- *Use of ChatGPT:* The versions of ChatGPT used in the study may not be representative of other versions or AI models, which could limit the generalizability of the results.

- 3) **Construct validity threats:** These are threats to the extent to which the measures used in the study accurately measure the constructs they are intended to measure. The following construct validity threats might present in our experiment:

- *Quality metrics:* The quality metrics used to evaluate the requirements may not accurately capture all aspects of requirement quality, leading to incomplete or biased results.
- *Feedback quality:* The quality of the feedback given to ChatGPT may not accurately reflect the quality of the requirements generated, which could affect the results.
- *Detection accuracy:* The Content at Scale AI detector used to evaluate the requirements may not accurately detect AI-generated text, leading to inaccurate results.

## IV. RESULTS

In this section, we present the results of our experiment, focusing on the data collected from the participants, as well as the performance of both the freemium and premium versions of ChatGPT in generating requirements. Additionally, we examine the effectiveness of the AI detector in detecting requirements. Our aim is to address the research questions (RQs) that guided this study.

To ensure clarity and organization, we first present the results in a phase-wise manner, highlighting the phases where significant data were obtained. This allows for a comprehensive understanding of the experimental progress and provides insights into the connections between the RQs. Subsequently, we proceed to answer the RQs based on the accumulated findings.

### A. Phase 1 - Participants

Our experiment included a total of 11 participants who contributed by providing their inputs. Each participant was requested to write a set of 10 requirements, consisting of 7 FRs and 3 NFRs (**Appendix E**).

*Table III* provides an overview of the demographic information of the participants who took part in our experiment. The table presents key details about each participant, including their current occupation, experience in their current occupation, experience in Software Engineering, and their self-assessed confidence in writing requirements.

The participant information reveals the diverse background and experiences of the individuals involved in the study. We had a total of 11 participants, including 4 students, 6 practitioners in the field of Software , and a researcher in REs.

Participants	Occupation	Experience in current occupation	Experience in SE	Confidence in writing REs (0-5)
P1	Student	2-5 years	2-5 years	3
P2	Practitioner	>5 years	5-10 years	4
P3	Practitioner	<2 years	5-10 years	4
P4	Practitioner	>5 years	2-5 years	5
P5	Student	2-5 years	2-5 years	3
P6	Practitioner	>5 years	>10 years	4
P7	Researcher	>5 years	>10 years	4
P8	Practitioner	<2 years	2-5 years	3
P9	Practitioner	<2 years	<2 years	2
P10	Student	2-5 years	5-10 years	4
P11	Student	2-5 years	<2 years	4

Table III: Demographics of the participants that contributed in the experiment

Their occupations varied, providing a mix of perspectives and expertise.

In terms of experience in their current occupation, we had participants with different ranges. Some participants had been students or practitioners for 2-5 years, while others had more than 5 years of experience. Similarly, when it came to their experience in Software Engineering, participants had varying durations, with some having 2-5 years of experience, others having 5-10 years, and a few having over 10 years of experience. Most of our participants have over 2 years experience in SE as represented in *Figure 3*.

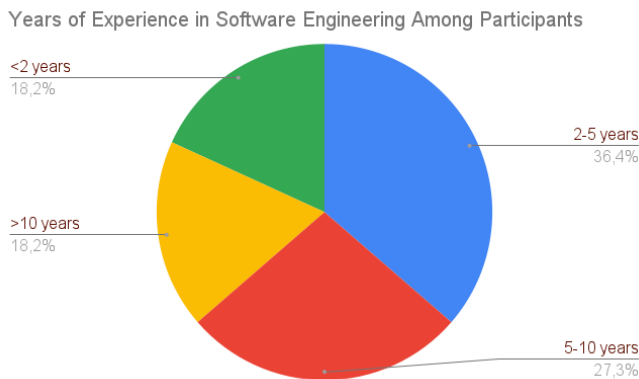


Figure 3: Distribution of Years of Experience in Software Engineering Among Participants graph

Participants' confidence in writing requirements varied among the group, with ratings ranging from 2 to 5 on a scale of 0 to 5. A rating of 0 indicates no confidence in writing requirements, while a rating of 5 represents high confidence in this skill. The values indicate different levels of self-assessed proficiency. It's worth noting that the confidence levels did not necessarily correlate with their years of experience, suggesting that factors other than experience may influence their confidence in this particular task.

It is noteworthy to mention that all the the students participating in the experiment have background as Teaching Assistants in the Software Engineering and Management program at the University of Gothenburg. They have gained practical experience in requirements engineering through their

involvement in academic software projects. This hands-on exposure to software development and requirement gathering has enhanced their understanding of the process and their ability to apply theoretical knowledge in practical settings.

Overall, the participant group represented a mix of backgrounds, experiences, and confidence levels, contributing to the diversity of perspectives in the study.

#### B. Phase 4 - Evaluation of Requirement Quality

##### 1) RE quality evaluation:

*Figure 4* provides an overview of the result analysis, focusing on the evaluation of requirement quality based on specific attributes. It visually presents the number of requirements for each participant that met a particular quality attribute. *Appendix G* provides a illustration of the same results based on quality attributes met in the requirements collected.

The results of the evaluation indicate the number of requirements that met specific quality attributes for each participant. Overall, most participants achieved high scores across various quality attributes, demonstrating a good level of requirement quality.

Based on the summarized description of the results, participants generally scored well across most of the quality attributes. Attributes such as "Necessary," "Implementation Free," "Unambiguous," and "Consistent" received high scores, indicating that participants were able to meet these criteria effectively in their requirements.

On the other hand, attributes such as "Complete," "Singular," "Feasible," "Traceable," and "Verifiable" received relatively lower scores. This suggests that participants faced challenges in fully addressing these aspects in their requirements, potentially indicating areas for improvement.

There were a few anomalies observed in the results. For participants P1, P2, P6, P10, and P11, there were cases where requirements categorized as FRs should have been classified as NFRs. This indicates a misclassification or misinterpretation of the requirement type.

In the evaluation of the ChatGPT free version -based on GPT-3 model- group, the scores were generally high across most of the quality attributes. Attributes such as "Necessary," "Implementation Free," "Consistent," and "Feasible" received perfect scores, indicating that ChatGPT free version performed well in these areas.

There were relatively lower scores for attributes such as "Unambiguous," "Complete," "Singular," "Traceable," and "Verifiable." This suggests that ChatGPT-free may have encountered challenges in fully meeting these quality criteria.

In comparison to the participant group, ChatGPT-free generally performed well, achieving high scores across most of the quality attributes. The scores for attributes such as "Necessary," "Implementation Free," "Consistent," and "Feasible" were comparable or even higher than the scores of the participants.

However, there were some quality attributes where ChatGPT-free scored lower than the participants. These attributes include "Unambiguous," "Complete," "Singular,"

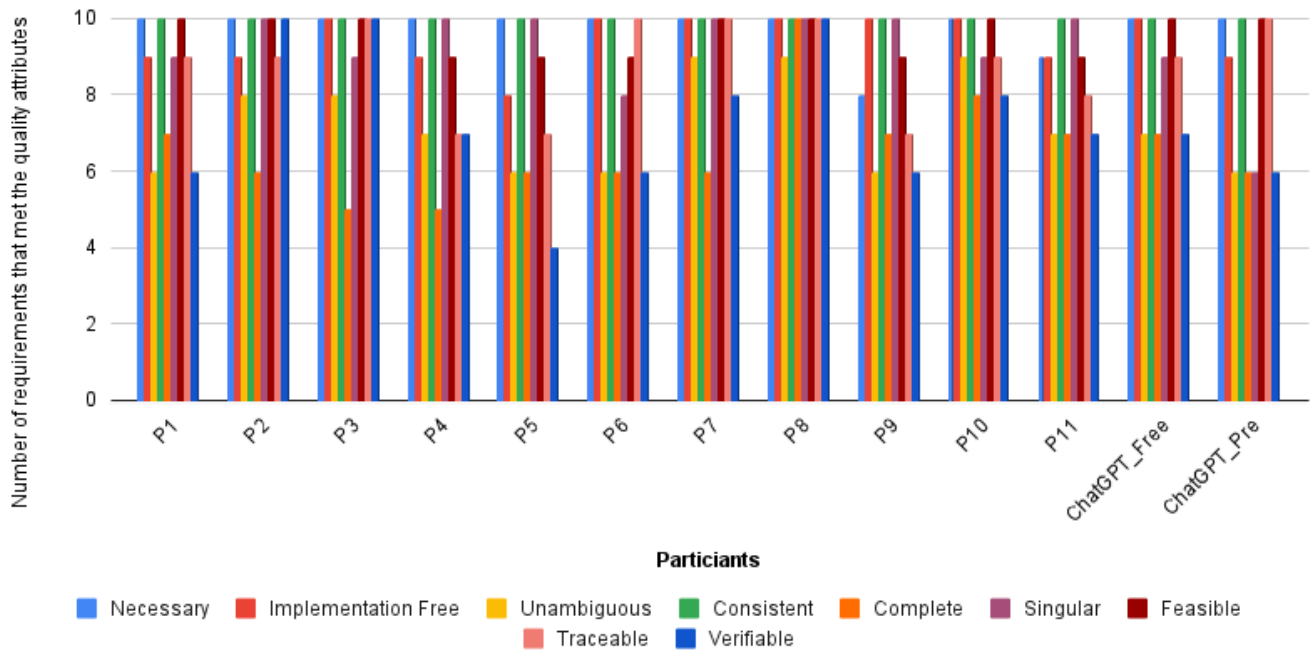


Figure 4: Overview on Results of Requirement Quality Evaluation Based on Quality Attributes including FRs and NFRs

“Traceable,” and “Verifiable.” The participants generally outperformed ChatGPT-free in these areas.

During the evaluation process, there was a specific case where ChatGPT generated a non-functional requirement that was mistakenly classified as a functional requirement. This occurrence highlighted a discrepancy in the classification of the requirement.

These differences suggest that while ChatGPT-free showed promising results in generating requirements, there are still some aspects where human participants demonstrated better performance. This highlights the need for further evaluation and improvement to enhance ChatGPT’s ability to meet all quality attributes at the same level as human-generated requirements.

In comparison to the participant group, ChatGPT Premium (ChatGPT-Pre) exhibited mixed results in terms of meeting the quality attributes of the requirements. On a positive note, ChatGPT-Pre performed well in attributes such as “Necessary,” “Implementation Free,” “Consistent,” “Feasible,” and “Traceable,” scoring relatively high. However, there were certain areas where ChatGPT-Pre showed lower scores compared to the participants. These attributes include “Unambiguous,” “Complete,” “Singular,” and “Verifiable,” where ChatGPT-Pre scored lower than the participants.

Figure 5 illustrates the comparison of ChatGPT-Free and ChatGPT-Pre in terms of various quality attributes during Phase 1. The results indicate that ChatGPT-Free generally outperformed ChatGPT-Pre in several quality attributes, including implementation free, unambiguousness, completeness, and singularity. On the other hand, both groups exhibited consistency

Performance of the freemium and premium versions of ChatGPT in Generating Qualitative Requirements

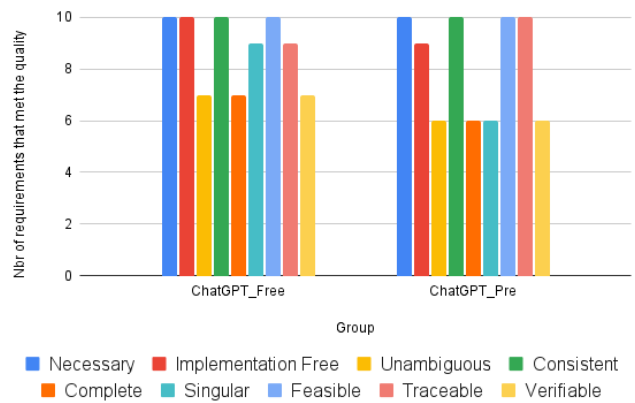


Figure 5: ChatGPT performance: Comparison between the freemium and premium versions

and achieved high scores for the necessary attribute, feasibility, traceability, and verifiability.

This suggests that both versions of ChatGPT demonstrate competence in certain quality aspects, while ChatGPT-Free shows an advantage in specific areas.

2) **Statistical analysis:**

The results of using Mann-Whitney U Test to compare scores of human written requirements and ChatGPT generated requirements for both version suggests the following:

**Free version of ChatGPT:** The total scoring of the attributes can be seen in the appendix D. *Table IV* shows the p-value for all the attributes when the variance was not equal to zero. The two quality attributes necessary and consistent both had zero variance between human written and ChatGPT generated requirements which means there is no difference in the data values for this quality attribute. When comparing the human written requirements to the ones generated by the free version of ChatGPT there is no significant statistical difference in any of the other quality attributes except for singular. Singular got a p-value  $< 0.05$  which is our chosen significance value. This indicates we can reject the null hypothesis for the quality attribute singular, but we can not reject it for any of the other quality attributes.

	W	df	p
Implementation Free	19.000		0.703
Unambiguous	12.000		0.258
Complete	19.000		0.688
Singular	2.500		0.012
Feasible	18.500		0.675
Traceable	11.000		0.209
Verifiable	18.000		0.625

Table IV: P-values when comparing generated requirement from the free version of ChatGPT with human written requirements.

**Premium version of ChatGPT:** *Table V* shows the four quality attributes that did not have zero variance. Similiar to the free version we can not reject the null hypothesis for RQ1 when using the paid version of ChatGPT except for the quality attribute singular. The p-value for singular  $< 0.05$  when comparing to human written requirements. This means we can reject the null hypothesis when using the paid version of ChatGPT for the quality attribute singular but can not reject it for any other quality attribute.

	W	df	p
Implementation Free	19.000		0.703
Unambiguous	21.000		0.793
Complete	15.500		0.467
Singular	5.000		0.030

Table V: P-values when comparing generated requirement from the paid version of ChatGPT with human written requirements.

The data presented in *Tables IV* and *V* indicate that, for the **majority** of attributes related to the IEEE requirement quality of an individual requirement, there is no significant difference between human-written requirements and those generated by either the free or premium version of ChatGPT. However, a significant difference was observed for the attribute "Singular" in both cases.

It is important to note that the chosen analysis has its limits. First, the Mann-Whitney U Test is a non-parametric test used to compare two independent groups, making it suitable for our study design. However, it does not provide information

about the magnitude of the differences observed. Additionally, since the judgment of quality aspects of requirements can be subjective, the statistical analysis may not fully capture all nuances and subjective interpretations.

### 3) Time spent to formulate REs:

The graph in *Figure 6* illustrates the time spent by human participants to write requirements.

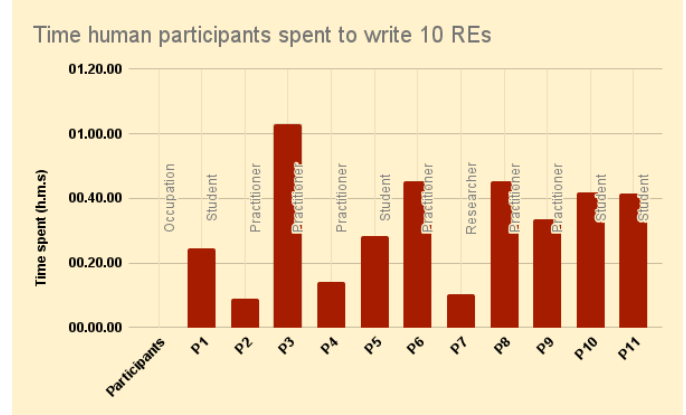


Figure 6: Time human participants spent to write 10 REs

*Duration spent to write 10 REs:* The time taken by participants to write 10 requirements varies. The range is from 8 minutes and 51 seconds to 1 hour, 3 minutes, and 1 second. The average time spent is around 30 minutes.

*Time differences:* The duration spent to write the requirements does not seem to correlate directly with participants' occupation or experience level. For example, some participants with less experience took less time compared to those with more experience.

During the entire interaction, including prompts and responses, the total time taken by ChatGPT-Free was 2 minutes, while ChatGPT-Premium took approximately 1 minute as shown in *Table VI* summarizing the time durations.

	ChatGPT-Free	ChatGPT-Premium
<b>Duration</b>	2 minutes	1 minute

Table VI: Total duration of interaction with ChatGPT both versions to generate REs

The results suggest, in terms of time efficiency, AI-based models offer an advantage by reducing the time required for requirement generation compared to humans.

### C. Phase 9: Re-evaluating the quality of the requirements provided by ChatGPT for both versions after feedback

Upon analyzing the results illustrated in *Figure 7*, it is evident that there were notable improvements in the quality of requirements generated by both versions of ChatGPT after receiving feedback.

For the ChatGPT-Free version, the requirements showed significant enhancements across multiple quality attributes. The scores for necessary, implementation free, consistent, and

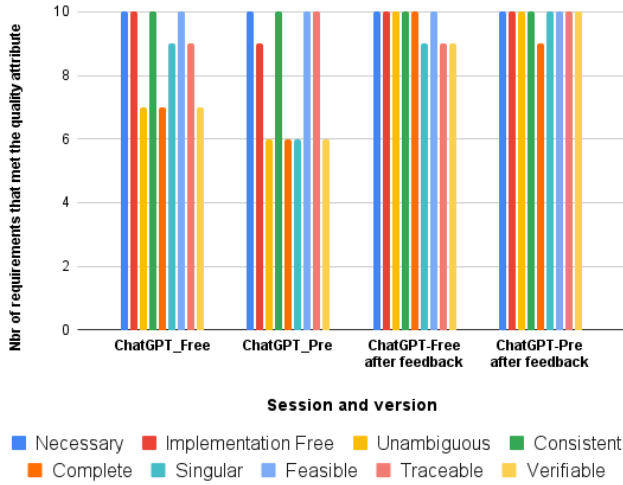


Figure 7: Changes in Requirement Quality after Feedback: Comparison of ChatGPT Versions including all the quality attributes

feasible requirements remained consistently high throughout the evaluation process. However, there was a noticeable improvement in unambiguous, complete, singular, traceable, and verifiable requirements after the feedback was provided.

Similarly, for the ChatGPT-Pre version, there was an overall improvement in the quality of requirements. The scores for necessary, implementation free, consistent, feasible, traceable, and verifiable requirements remained consistently high. Notably, there were substantial improvements in unambiguous, complete, singular, and consistent requirements after incorporating the feedback.

Scoped overview on improvements of performance after feedback - changed values

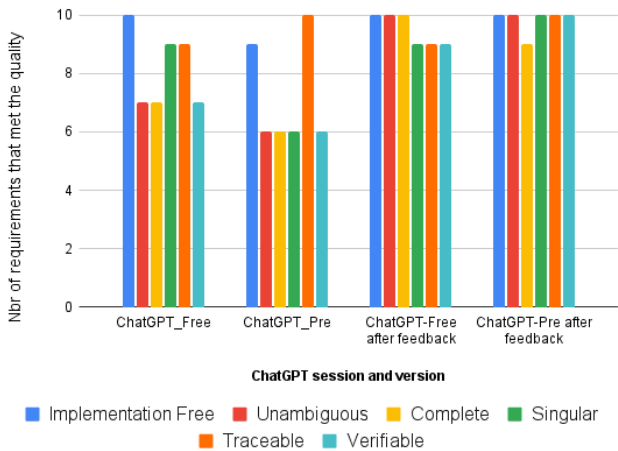


Figure 8: Changes in Requirement Quality after Feedback: Comparison of ChatGPT Versions in Specific Attributes

The data provided in Figure 8 represents the evaluation of requirement quality using specific quality attributes. The focus is on the areas where changes were observed, either for the better or worse.

For the attribute "Implementation Free," the ChatGPT\_Free initially scored 10, indicating a high level of implementation freedom. However, in the ChatGPT\_Pre, the score was slightly lower at 9. After receiving feedback, both improved their scores, with ChatGPT-Free reaching a perfect score of 10, while ChatGPT-Pre maintained its score of 10.

Similarly, in the "Unambiguous" attribute, the ChatGPT\_Free had a score of 7 initially, which was lower than the ChatGPT\_Pre's score of 6. However, after feedback, both improved significantly, with ChatGPT-Free achieving a perfect score of 10, and ChatGPT-Pre improving to 10.

In terms of completeness, the ChatGPT\_Free scored 7 initially, while the ChatGPT\_Pre scored 6. After feedback, both made progress, with ChatGPT-Free reaching a score of 10, and ChatGPT-Pre improving to 9.

For the attributes "Singular," "Traceable," and "Verifiable," both initially showed lower scores, indicating room for improvement. However, after feedback, significant improvements were observed, with both reaching perfect scores of 10 in these areas.

It is worth mentioning that when we provided feedback to ChatGPT about a requirement, pointing out that a functional requirement should actually be a non-functional requirement and highlighting missing qualities, ChatGPT generated an NFR with enhanced quality, as we observed.

The feedback provided to both versions of ChatGPT seemed overall to have a positive impact on the generated requirements, resulting in higher scores across various quality attributes. The improvements indicate the ability of ChatGPT to learn from feedback and adjust its outputs accordingly.

Attribute	Free Before	Free After	Change in %
Unambiguous	7	10	42.9%
Complete	7	10	42.9%
Singular	9	9	-
Traceable	9	9	-
Verifiable	7	9	28.6%

Table VII: Attribute scores of the Free version of ChatGPT before and after feedback

Attribute	Premium Before	Premium After	Change in %
Impl. Free	9	10	11.1%
Unambiguous	6	10	66.7%
Complete	6	9	50%
Singular	6	10	66.7%
Verifiable	6	10	66.7%

Table VIII: Attribute scores of the Premium version of ChatGPT before and after feedback

The percentage increase in scores, as shown in the Tables VII and VIII demonstrates the effectiveness of providing feedback to ChatGPT in improving requirement qualities for both versions.

In terms of improvement, the Free version performed better in the attributes of Unambiguousness and Completeness, with a change of 42.9% for both attributes. The Verifiability attribute also improved by 28.6%. On the other hand, the Premium version demonstrated higher improvement percentages in attributes such as Unambiguousness, Completeness, Singular, and Verifiability, with changes of 66.7% for all these attributes. The Implementation Freedom attribute showed a relatively smaller improvement of 11.1%.

Overall, the Premium version showed higher improvement percentages in most attributes compared to the Free version. However, it is important to note that both versions displayed positive changes, indicating that the feedback had a beneficial impact on the quality of the generated requirements.

These results suggest that the iterative process of providing feedback to ChatGPT can contribute to the enhancement of requirement quality, especially the premium version, highlighting the potential of leveraging human interaction to improve the performance of AI-based requirement generation systems.

#### D. Phase 10 - Quality after repeated prompts

Overview of changes in REs quality in terms of iteration

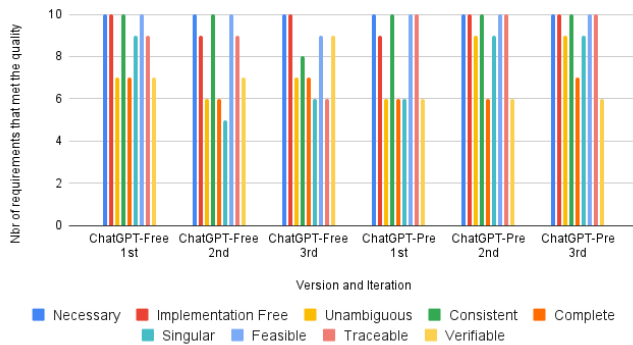


Figure 9: Overview on Changes in Requirement Quality for ChatGPT both Versions in three iterations

The results illustrated in Figure 9 show variations in requirement quality across different attributes for both ChatGPT versions. For the free version, the quality of requirements showed some fluctuations over the three iterations. The premium version also exhibited variations in quality, but to a lesser extent.

##### 1) ChatGPT Free Version iterations:

The graph presented in Figure 10 illustrates the results of the evaluation conducted on the ChatGPT free version across multiple iterations. The data focuses on various quality attributes of the generated requirements, including Necessary, Implementation Free, Unambiguous, Consistent, Complete, Singular, Feasible, Traceable, and Verifiable. Each iteration of the ChatGPT free version is represented by a different data point on the graph. The graph provides insights into the performance and improvement of the ChatGPT free version over the course of multiple iterations.

Changes in Requirement Quality for ChatGPT Free Version in terms of iterations

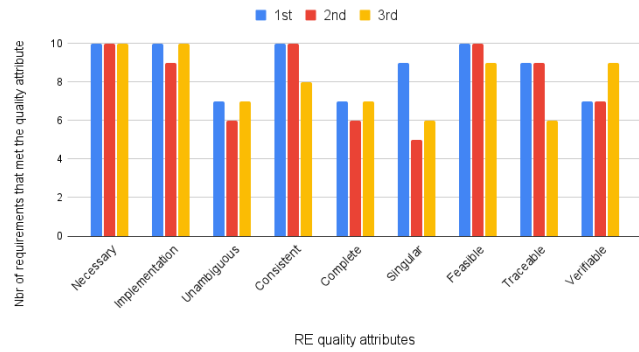


Figure 10: Changes in Requirement Quality for ChatGPT Free Version in three iterations

Across the three iterations of the ChatGPT free version, there were both consistent aspects and changes observed in the quality attributes of the generated requirements. The necessary attribute remained consistent, with all requirements meeting this criterion in each iteration, indicating that essential elements were included.

**Implementation free:** the first iteration had the highest number of requirements (10) satisfying this attribute. However, in the second iteration, one requirement received a lower score in this aspect, and in the third iteration, the number of implementation free requirements decreased slightly to 9.

**Unambiguousness:** saw a decrease in the second and third iterations, with fewer requirements (6 and 7, respectively) being unambiguous compared to the previous iterations.

**Consistency:** was maintained throughout all iterations, indicating that the generated requirements were coherent and aligned with the project context.

**Completeness:** decreased slightly in the second and third iterations, with one less requirement considered complete compared to the previous iterations.

**Singularity:** decreased across the iterations, with fewer requirements focusing on individual functionalities.

**Feasibility:** showed slight variations, with the second iteration having one requirement with a lower score in this aspect.

**Traceability and verifiability:** had minor changes, but overall, most requirements remained traceable and verifiable across the iterations.

The analysis reveals variations in requirement quality across different iterations of ChatGPT free version. While some attributes remained consistent, others showed fluctuations, such as unambiguousness, completeness, singularity, and traceability.

##### 2) ChatGPT Premium Version iterations:

The analysis of the premium ChatGPT-generated requirements for each iteration revealed consistent patterns and some variations in different quality attributes as shown in Figure 11. In the first iteration, all requirements met the necessary quality attribute, indicating their inclusion of essential elements. Most

Changes in requirement Quality for ChatGPT premium version in terms of iterations

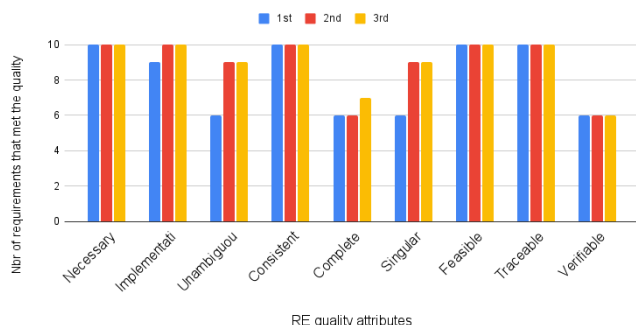


Figure 11: Changes in Requirement Quality for ChatGPT Premium Version in three iterations

requirements were implementation free, not specifying specific implementation details. The majority of requirements were unambiguous and maintained consistency with the project context. However, there was a slight decrease in the completeness of the requirements, with one requirement lacking certain aspects. The attribute of singularity also decreased, as fewer requirements focused on individual functionalities. Nonetheless, the requirements remained feasible, traceable, and verifiable.

Moving to the second iteration, the necessary attribute remained consistent, with all requirements meeting this criterion. All requirements continued to be implementation free, and the attribute of unambiguousness showed an improvement, with more requirements being clear and precise. The requirements maintained consistency, completeness, and feasibility. There was an increase in the attribute of singularity, with more requirements focusing on individual functionalities. The traceability and verifiability attributes remained consistent, ensuring easy tracking of the requirements' origin and their objective testability.

In the third iteration, the necessary attribute remained consistent, and all requirements continued to be implementation free and unambiguous. The requirements maintained consistency and feasibility, with a slight increase in completeness and singularity compared to the previous iterations. However, the attribute of traceability showed a slight decrease, with fewer requirements being easily traceable. The verifiability attribute remained consistent, indicating that the requirements could be objectively tested.

The premium ChatGPT-generated requirements demonstrated a consistent to better level of quality across the iterations, meeting the necessary criteria and exhibiting implementation freedom, unambiguousness, consistency, completeness, singularity, feasibility, traceability, and verifiability.

In summary, in terms of specific attributes, the free version demonstrated improvements in the Feasible and Verifiable categories over the iterations. However, there were inconsistencies in the Singular and Unambiguous attributes. The premium

version generally maintained a high level of quality across most attributes, with slight improvements in Unambiguous and Singular categories over the iterations.

### E. Phase 12: Analysing the results from AI detector

We have defined the following value categories to evaluate the accuracy of AI detector Content at Scale in detecting AI generated requirements:

- **True Positives (TP):** The number of AI-generated requirements correctly detected by the detector.
- **False Positives (FP):** The number of human-written requirements incorrectly detected as AI-generated by the detector.
- **True Negatives (TN):** The number of human written requirements were correctly classified as human-written by the detector.
- **False Negatives (FN):** The number of AI-generated requirements incorrectly classified as human-written by the detector.

Given that each version of ChatGPT generated a total of 60 requirements across all three iterations (with each iteration producing 10 requirements), we elected to include the same number of requirements from the human participants in order to keep the dataset balanced. Consequently, this approach led to a total compilation of 120 requirements.

Out of the total 60 AI-generated requirements, the detector correctly identified 8 requirements as AI-generated, resulting in true positives (TP).

However, there was 1 false positive (FP), where a human-written requirement was mistakenly classified as AI-generated out of 60 by the detector.

There were 59 true negatives (TN), where human-written requirements correctly classified as human-written by the detector.

Furthermore, there were 53 false negatives (FN), where AI-generated requirements were incorrectly classified as human-written by the detector. This means that the detector failed to recognize these requirements as being generated by ChatGPT.

	Human-RE	AI-RE	TP	FP	FN
<b>Count</b>	60	60	8	1	53

Table IX: Values extracted to evaluate AI Content Detector: Content at Scale accuracy

These results as summarised in *Table IX* highlight both the strengths and limitations of the Content at Scale AI detector in accurately distinguishing between AI-generated and human-written requirements. While it demonstrated true positive rate, there were some instances of misclassifications, leading to false positives and false negatives.

Metric	TP	FP	FN	Precision	Recall	F1 Score
<b>Value</b>	8	1	53	0.8889	0.1333	0.2319

Table X: Performance Metrics for Content at Scale AI Detector in Detecting AI-Generated Requirements

The precision is approximately 0.889, indicating that out of all the requirements classified as AI-generated, around 88.9% were actually AI-generated.

The recall is approximately 0.133, suggesting that the detector only captured around 13.3% of all the AI-generated requirements.

The F1 score, which combines precision and recall, is approximately 0.231. It provides a balanced measure of the model's performance, taking into account both false positives and false negatives.

This suggests that the detector has difficulty accurately identifying AI-generated requirements and tends to miss a significant number of them.

## F. Answers to Research Questions

### **RQ1: How do requirements generated by ChatGPT compare to human-written requirements in terms of the nine requirement characteristics defined by the IEEE for an individual requirement?**

Based on the collected data and analysis in *Phase 4*, it's clear that ChatGPT, in both its free and premium versions, performs reasonably well in generating software requirements, matching the quality of human-written requirements in several IEEE requirement quality attributes. However, it falls short in the "Singular" attribute and is generally outperformed by experienced human practitioners and researchers, particularly in terms of unambiguousness and completeness.

A Mann-Whitney U Test was used to compare scores of generated and human written requirements. The collected data in *Tables IV* and *V* show that, for most attributes of the IEEE requirement quality of an individual requirement, there is no significant difference between human-written requirements and those generated by either the free or premium version of ChatGPT.

In other words, the null hypothesis (H0) cannot be rejected for **most** quality attributes. However, in both cases, there is a significant difference for the attribute "Singular," leading to the rejection of H0 and the acceptance of H1 for this attribute.

### **RQ2: How does the quality of ChatGPT-generated requirements vary across different versions, such as premium and freemium?**

To address this research question, we conducted an investigation into the performance of both ChatGPT versions after receiving feedback (Sub RQ2.1) in terms of requirement quality. Additionally, we examined the impact of querying the same prompt multiple times (Sub RQ2.2) on the generated requirements.

#### **• RQ2.1: How does feedback given to ChatGPT impact the improvement of requirements' qualities?**

To test the hypotheses, we evaluated the results obtained from the feedback given to ChatGPT in *Phase 9* of the experiment, we analyzed the results obtained after providing feedback to both the free and paid versions of ChatGPT.

The data showed that both the free and premium versions of ChatGPT demonstrated improvement in various quality attributes after receiving feedback, as shown in *Tables VII* & *VIII*. The scores increased in multiple attributes.

Based on these observations, we can reject the null hypothesis (H0) and accept the alternative hypothesis (H1). The feedback provided to ChatGPT resulted in a significant improvement in the quality of requirements generated by the system. This suggests that providing feedback can effectively enhance the requirement qualities of ChatGPT.

These findings highlight the value of incorporating feedback mechanisms in ChatGPT for requirement generation and emphasize the potential for improvement through human interaction.

#### **• RQ2.2: What is the impact of querying the same prompt multiple times on the quality of requirements generated by ChatGPT?**

Based on the results obtained from the analysis detailed in *Phase 10*, we observed variations in the quality of requirements generated by ChatGPT for the same prompt when queried multiple times. These variations were evident in both the ChatGPT free version and the premium version.

The ChatGPT free version showed fluctuations in several quality attributes across different iterations, including unambiguousness, completeness, singularity, and traceability. While some attributes remained consistent, others demonstrated improvements or slight variations.

On the other hand, the premium ChatGPT-generated requirements consistently exhibited a better level of quality across the iterations. They consistently met the necessary criteria and demonstrated higher scores in attributes such as implementation freedom, unambiguousness, consistency, completeness, singularity, feasibility, traceability, and verifiability.

Based on these findings, we can reject the null hypothesis (H0) and conclude that there is a significant difference in the quality of requirements generated by ChatGPT for the same prompt when queried multiple times. The improvements and variations observed in the requirement quality indicate that repetitive querying of the same prompt can influence the quality of the generated requirements.

In addition to the data collected and the conclusions of RQ2.1 and RQ2.2, we conducted a comparative analysis of the performance between the premium and freemium versions. This analysis involved examining the results presented in a graph in *Phase 4* (*Figure 5*) and incorporating feedback from *Phase 9*, which is depicted in *Figure 7*. By considering these additional findings, we can draw conclusions for RQ2 regarding the hypotheses.

The results of RQ2.1 indicate that both the free and premium versions of ChatGPT demonstrated improvements in various quality attributes after receiving feedback. This is evident in *Phase 9*, as depicted in *Figure 8*, and supported by the data presented in *Tables VII* and *VIII*. Notably, the premium version of ChatGPT exhibited superior perfor-

mance compared to the free version, consistently showing enhancements across all quality attributes. Specifically, the premium version achieved higher scores in completeness and verifiability, suggesting its greater potential for enhancing the quality of generated requirements compared to the free version. These findings affirm the hypothesis H1, as there is a significant difference in the quality of requirements generated by ChatGPT across different versions.

Additionally, the analysis in RQ2.2 revealed variations in the quality of requirements generated by ChatGPT when the same prompt was queried multiple times. The ChatGPT free version exhibited fluctuations in several quality attributes, while the premium version consistently exhibited a better level of quality across iterations. Based on these findings, we can reject the null hypothesis (H0) and conclude that there is a significant difference in the quality of requirements generated by ChatGPT across different versions. The improvements and variations observed indicate that repetitive querying of the same prompt and the use of different versions influence the quality of the generated requirements.

In conclusion, to RQ2, the results support the alternative hypothesis (H1), suggesting a significant difference in the quality of requirements generated by ChatGPT across different versions.

### **RQ3: To what extent can an AI detector differentiate between ChatGPT generated requirements and human-written requirements?**

The results analysed in *phase 12* indicate that the Content at Scale AI detector has a relatively high precision, meaning it correctly identified the detected AI REs as AI-generated requirements. However, the recall is low, suggesting that the detector fails to capture a significant number of AI-generated requirements. Consequently, the F1 score, which considers both precision and recall, is relatively low. We fail to reject the null hypothesis (H0) and conclude that the Content at Scale AI detector is **not** able to effectively detect requirements generated by ChatGPT.

## V. DISCUSSION

### *1) Our findings and Related work:*

The findings of this research substantiate and extend the work of Wei et al. [8], which indicated that the performance of Large Language Models (LLMs) like ChatGPT is significantly improved with an increase in model size and data quantity. Our study similarly found that ChatGPT, both free and premium versions, is competent at generating software requirements, although it still falls behind experienced humans in certain quality attributes.

Our work also resonates with strategies explored by Zhou et al. [22] and others [14], [15], [17] to further improve LLMs. One of these strategies is reinforcement learning from human feedback, which, in our study, was found to significantly improve the quality of requirements generated by ChatGPT. This validates the potential of integrating feedback mechanisms to optimize the performance of AI tools in software

requirement engineering, in line with the idea of leveraging LLMs for a variety of applications such as trustworthy AI and conversational search engines [16].

Consistent with the work of Barenkamp et al. [26], our study underscores the utility of AI in different stages of the software engineering life cycle. We found that ChatGPT significantly enhances time efficiency in generating requirements compared to humans, without substantially compromising the quality. This suggests that AI tools can contribute to working more efficiently and reducing errors, further corroborating the potential benefits of integrating AI into the software engineering life cycle as highlighted by Barenkamp et al.

In relation to Abad et al. [47] and Surana et al. [40], who explored automated classification of requirements and requirements elicitation respectively, our study demonstrates the potential of ChatGPT in performing similar tasks with proficient efficiency. Nevertheless, our results also highlight that the intricacy and quality of requirements generated by experienced humans remain unparalleled, indicating the continued relevance of human expertise.

Finally, regarding our third RQ related to AI detectors, our findings mirror the challenge faced by many in accurately identifying AI-generated text. Despite the Content at Scale AI detector demonstrating high precision, its low recall indicates the need for enhanced AI detectors for more reliable identification of AI-generated requirements. This aligns with the ongoing quest for improved AI detection mechanisms in the field.

In conclusion, this research reinforces the potential of AI tools such as ChatGPT in generating software requirements and the value of human feedback in enhancing AI's performance. The findings also echo the broader literature on the utility of AI in software engineering and the ongoing challenges in accurately identifying AI-generated content.

### *2) Alternative methodology and future work:*

An alternative methodology that could have been employed in this research is the experimental comparison using mixed-methods. Rather than exclusively relying on quantitative evaluation, integrating qualitative methods such as interviews or focus groups with the participants could offer a more in-depth understanding of their experiences and perceptions. This would provide a broader perspective on the effectiveness of ChatGPT in generating requirements and their interaction with it. Integrating a mixed-methods approach would offer a more comprehensive understanding of ChatGPT's performance. It would capture the nuances of participant experiences and the subjective factors influencing the performance of ChatGPT, enhancing the richness and depth of the insights gained.

The study could be extended by comparing ChatGPT's efficacy with other AI-based tools in generating software requirements. This would provide a comprehensive view of the AI landscape in the context of software requirement engineering and help identify the most promising tools. Comparing ChatGPT with other AI tools would offer a competitive analysis and assist stakeholders in making informed decisions about

which AI tool to incorporate in their software requirement engineering process.

## VI. CONCLUSIONS

The research offers key insights into ChatGPT's potential, across both free and premium versions, for generating software requirements. It was found that while ChatGPT can match human-written requirements on several IEEE quality attributes, it lags behind experienced humans in aspects like unambiguity, completeness, and singularity.

Regardless, significant potential exists in leveraging AI tools like ChatGPT to augment human expertise in requirement generation. Feedback mechanisms can help improve the AI's performance, contributing to higher quality requirements.

Further investigation demonstrated that querying the same prompt multiple times influenced the quality of requirements generated by ChatGPT. The free version exhibited fluctuations in quality attributes across different iterations, while the premium version consistently delivered higher-quality requirements. These variations highlight the importance of considering repetitive querying and the selection of different versions when using ChatGPT for requirement generation.

Differences were observed between the free and premium versions of ChatGPT in consistency and overall quality. The premium version consistently performed better and showed more considerable improvement post feedback, implying its superior potential for enhancing requirement generation. Interestingly, the free version outperformed the premium one in certain attributes during the initial trial.

Regarding time efficiency, both versions of ChatGPT excelled at generating requirements faster than humans, with the premium version being particularly swift. This speed advantage does not significantly detract from the quality of the generated requirements, making AI tools valuable for efficient requirement generation.

Nonetheless, the intricacy and quality of requirements generated by experienced human practitioners remain unparalleled. This suggests the value of using AI tools like ChatGPT as supplements to human expertise in requirement engineering, to benefit from AI's speed and efficiency while retaining the depth of human analysis.

Additionally, the study assessed the efficacy of the Content at Scale AI detector in identifying AI-generated requirements. The detector demonstrated high precision but low recall, pinpointing a need for improved detectors for more reliable identification of AI-generated requirements.

In conclusion, these findings highlight the value of AI tools such as ChatGPT for time-efficient requirement generation, emphasizing the need for human expertise and feedback mechanisms to optimize AI's performance. They also underline the necessity for enhanced AI detectors for accurate identification of AI-generated requirements.

## ACKNOWLEDGEMENT

We would like to take a moment to express our heartfelt gratitude to Jenniffer Horkoff, our amazing supervisor, who

deserves a standing ovation for her exceptional guidance throughout this journey. Jenniffer, you not only provided us with the assignment's domain problem and detailed feedback but also showered us with your expertise to review our results. Your unwavering support and infectious enthusiasm made this research adventure even more thrilling. We are forever indebted to you!

To our families, who have been our rocks and pillars of support, we owe a debt of gratitude that cannot be measured. Thank you for your understanding, patience, and love, even when we were buried in piles of data or lost in the intricacies of statistical analyses. Your unwavering belief in us propelled us forward, and we are immensely grateful for your constant encouragement.

We would also like to extend our sincere appreciation to the university and its staff for creating an environment conducive to learning and exploration. Your commitment to excellence and dedication to fostering academic growth have been instrumental in our research endeavors.

Last but not least, we express our heartfelt thanks to the participants who graciously dedicated their valuable time and contributed to this study. Your involvement and willingness to share your insights and experiences have enriched our research and made it possible to draw meaningful conclusions.

To all those who have played a part in our research journey, whether big or small, we extend our deepest appreciation. Without your support, encouragement, and valuable contributions, this research would not have been possible.

## REFERENCES

- [1] Market Research Reports & Consulting — GlobalData UK Ltd. (n.d.). Artificial Intelligence (AI) Market Size, Share, Trends, Analysis and Forecast by Product/Service (Specialized AI Applications, AI Hardware, AI Platforms, AI Consulting and Support Services), Enterprise Size Band, Vertical and Region, 2021-2026. [online] Available at: <https://www.globaldata.com/store/report/artificial-intelligence-market-analysis/>.
- [2] M.I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspective, and prospects", *Science*, vol 349, no. 6245, pp. 255-266, 2015.
- [3] OmegaVp, "Large language models keep getting larger: What surprises will gpt-4 bring?" Available at <https://www.omegavp.com/blog/large-language-models-keep-getting-larger-what-surprises-will-gpt-4-bring/>. (Accessed: 16.02.2023).
- [4] S. Frieder et al., "Mathematical capabilities of chatgpt." arXiv preprint arXiv:2301.13867, 2023.
- [5] OpenAI (2022). ChatGPT: Optimizing Language Models for Dialogue. [online] OpenAI. Available at: <https://openai.com/blog/chatgpt>.
- [6] T. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [7] T. K. Kung et al., "Performance of chatgpt on usml: Potential for ai-assisted medical education using large language models," 2022, Manuscript submitted for publication.
- [8] Wei J, Tay Y, Bommasani R, et al., 2022b. Emergent abilities of large language models. <https://arxiv.org/abs/2206.07682>
- [9] Brown TB, Mann B, Ryder N, et al., 2020. Language models are few-shot learners. *Proc 34th Int Conf on Neural Information Processing Systems*, p.1877–1901.
- [10] Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1), 5232-5270.
- [11] B. Shehata, A. Tlili and R. Huang, "An Analysis of International Conference Proceedings on Artificial General Intelligence (AGI) from 2008 to 2020: A Data-Mining Mapping Analysis," 2021 8th International Conference on ICT & Accessibility (ICTA), Tunis, Tunisia, 2021, pp. 1-6, doi: 10.1109/ICTA54582.2021.9809429.

- [12] Sun, Y., Wang, S., Feng, S., Ding, S., Pang, C., Shang, J., ... and Wang, H. (2021). Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. arXiv preprint arXiv:2107.02137.
- [13] Zeng W, Ren XZ, Su T, et al., 2021. PanGu: large-scale autoregressive pretrained Chinese language models with auto-parallel computation. <https://arxiv.org/abs/2104.12369>
- [14] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... and Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.
- [15] Wei J, Bosma M, Zhao VY, et al., 2021. Fine tuned language models are zero-shot learners. 10th Int Conf on Learning Representations.
- [16] OpenAI (2022). ChatGPT: Optimizing Language Models for Dialogue. [online] OpenAI. Available at: <https://openai.com/blog/chatgpt>. [Accessed on Apr. 08, 2023].
- [17] Stiennon N, Ouyang L, Wu J, et al., 2020. Learning to summarize from human feedback. Proc 34th Int Conf on Neural Information Processing Systems, p.3008–3021.
- [18] Hu, K. (2023). ChatGPT sets record for fastest-growing user base - analyst note. Reuters. [online] 2 Feb. Available at: (<https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/>).
- [19] Schulman J, Zoph B, Kim C, et al., OpenAI (2022). ChatGPT: Optimizing Language Models for Dialogue. [online] OpenAI. Available at: <https://openai.com/blog/chatgpt>. [Accessed on Apr. 04, 2023]
- [20] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022b). Chain of thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903.
- [21] Ashish, V. (2017). Attention is all you need. Advances in neural information processing systems, 30, I.Proc 31st Int Conf on Neural Information Processing Systems, p.6000–6010
- [22] Zhou, J., Ke, P., Qiu, X. et al. ChatGPT: potential, prospects, and limitations. Front Inform Technol Electron Eng (2023).
- [23] D. Bajaj, A. Goel, S.C. Gupta and H. Batra, "MUCE: a multilingual use case model extractor using GPT-3", *International Journal of Information Technology*, vol.14(1), pp. 1543-1554, 2022.
- [24] V. Braun and V. Clarke, "Using thematic analysis in psychology." *Qualitative research in psychology*, vol 3(2), pp. 77-101, 2006.
- [25] J. Horkoff. DIT046 Lecture 4: Requirements Concepts, Writing Requirements, Requirements Quality [University of Gothenburg, PowerPoint Slides](2022)
- [26] Barenkamp, M., Rebstadt, J. Thomas, O. Applications of AI in classical software engineering. *AI Perspect* 2, 1 (2020). <https://doi.org/10.1186/s42467-020-00005-4>
- [27] Nayan B. Ruparelia. 2010. Software development lifecycle models. *SIGSOFT Softw. Eng. Notes* 35, 3 (May 2010), 8–13. <https://doi.org/10.1145/1764810.1764814>
- [28] Han W, Jiang H, Lu T, Zhang X, Li W. An optimized resolution for software project planning with improved max–min ant system algorithm. *Int J Multimedia Ubiquitous Eng.* 2015;10(6):25–38.
- [29] Athavale S, Balaraman V. Human behavioral modeling for enhanced software Project Management. In: 7th International Conference on Software Engineering; 2013. p. 15–7. sn.
- [30] Elzamly A, Hussin B, Abu-Naser SS, Doheir M. Classification of software risks with discriminant analysis techniques in software planning development process; 2015.
- [31] Forecast (n.d.). Platform Overview — Forecast. [online] [www.forecast.app](https://www.forecast.app). Available at: <https://www.forecast.app/platform>.
- [32] markdefalco (n.d.). Sketch2Code. [online] [learn.microsoft.com](https://learn.microsoft.com/en-us/shows/ai-show/sketch2code). Available at: <https://learn.microsoft.com/en-us/shows/ai-show/sketch2code> [Accessed 14 Apr. 2023].
- [33] TensorFlow (2019). TensorFlow. [online] TensorFlow. Available at: <https://www.tensorflow.org/>.
- [34] Fast authoring of AI-stabilized end-to-end tests—codeless, coded, or both. (n.d.). Automated Functional Testing - Software Testing Tool - Testim.io. [online] Available at: <https://www.testim.io/>.
- [35] Logz.io. (n.d.). How does Logz.io help me troubleshoot production faster? [online] Available at: <https://logz.io/solutions/accelerate-cloud-monitoring/> [Accessed 14 Apr. 2023].
- [36] M. Chen et al., "Evaluating large language models trained on code.", 2021, arXiv preprint arXiv:2107.03374
- [37] S. Dominik et al., "An analysis of the automatic bug fixing performance of chatgpt.", 2023, arXiv preprint arXiv:2301.08653.
- [38] C. Le Goues, T. Nguyen, S. Forrest and W. Weimer, "GenProg: A Generic Method for Automatic Software Repair," in *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 54-72, Jan.-Feb. 2012, doi: 10.1109/TSE.2011.104.
- [39] J. Sajed., S. Rafi, T. D. LaToza, K. Moran, and W. Lam, "ChatGPT and Software Testing Education: Promises Perils.", 2023, arXiv preprint arXiv:2302.03287
- [40] C. S. Rajender Kumar Surana, Shriya, D. B. Gupta and S. P. Shankar, "Intelligent Chatbot for Requirements Elicitation and Classification," 2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), Bangalore, India, 2019, pp. 866-870, doi: 10.1109/RTEICT46194.2019.9016907.
- [41] G. Y. Quba, H. Al Qaisi, A. Althunibat and S. AlZu'bi, "Software Requirements Classification using Machine Learning algorithm's," 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 2021, pp. 685-690, doi: 10.1109/ICIT52682.2021.9491688.
- [42] W. A. Qader, M. M. Ameen and B. I. Ahmed, "An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges," 2019 International Engineering Conference (IEC), Erbil, Iraq, 2019, pp. 200-204, doi: 10.1109/IEC47844.2019.8950616.
- [43] S. Panichella and M. Ruiz, "Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback," 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 2020, pp. 404-407
- [44] K. Liu, S. Reddivari and K. Reddivari, "Artificial Intelligence in Software Requirements Engineering: State-of-the-Art," 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), San Diego, CA, USA, 2022, pp. 106-111
- [45] K. Zamani, D. Zowghi and C. Arora, "Machine Learning in Requirements Engineering: A Mapping Study," 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), Notre Dame, IN, USA, 2021, pp. 116-125, doi: 10.1109/REW53955.2021.00023.
- [46] 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.
- [47] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe and K. Schneider, "What Works Better? A Study of Classifying Requirements," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017, pp. 496-501, doi: 10.1109/RE.2017.36.
- [48] K. V. Vineetha and P. Samuel, "A Multinomial Naïve Bayes Classifier for identifying Actors and Use Cases from Software Requirement Specification documents," 2022 2nd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2022, pp. 1-5, doi: 10.1109/CONIT55038.2022.9848290.
- [49] A. C. Braun, U. Weidner and S. Hinz, "Support vector machines, import vector machines and relevance vector machines for hyperspectral classification — A comparison," 2011 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Lisbon, Portugal, 2011, pp. 1-4, doi: 10.1109/WHISPERS.2011.6080861.
- [50] A. A. Putri Ratna, A. Kaltum, L. Santiar, H. Khairunnissa, I. Ibrahim and P. D. Purnamasari, "Term Frequency-Inverse Document Frequency Answer Categorization with Support Vector Machine on Automatic Short Essay Grading System with Latent Semantic Analysis for Japanese Language," 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), Batam, Indonesia, 2019, pp. 293-298, doi: 10.1109/ICECOS47637.2019.8984530.
- [51] D. Dave and V. Anu, "Identifying Functional and Non-functional Software Requirements From User App Reviews," 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRON-ICS), Toronto, ON, Canada, 2022, pp. 1-6, doi: 10.1109/IEMTRON-ICS55184.2022.9795770.
- [52] C. Terwiesch "Would Chat GPT3 Get a Wharton MBA? A Prediction Based on Its Performance in the Operations Management Course.", Mack Institute for Innovation Management at the Wharton School, University of Pennsylvania, 2023, Retrieved from: <https://mackinstitute.wharton.upenn.edu/wp-content/uploads/2023/01/Christian-Terwiesch-Chat-GTP-1.24.pdf> [Date accessed: February 6th, 2023]
- [53] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering," in *ISO/IEC/IEEE 29148:2011(E)*, vol., no., pp.1-94, 1 Dec. 2011, doi: 10.1109/IEEESTD.2011.6146379.

- [54] P. E. McKnight and J. Najab Mann-Whitney U Test. The Corsini encyclopedia of psychology, 2010.
- [55] S. M. Karadimitriou, E. Marshall and C. Knox. "Mann-whitney U test". Sheffield Hallam University, 2018.
- [56] R. J. Tallarida and R. B. Murray."Mann-Whitney" in *Manual of Pharmacologic Calculations with Computer Programs* 2nd ed. New York: Springer, 1987, pp. 149-153.

# Appendices

## Appendix A - Problem description - MSB emergency app

In Sweden, the responsibility for disaster management lies primarily with the municipalities and their emergency services (räddningstjänst), while MSB (Myndigheten för samhällsskydd och beredskap) is the national authority responsible for supporting the municipalities and emergency services in their efforts to prevent, prepare for, and respond to emergencies. MSB's goal is to ensure the health and safety of people during emergencies, and to have a successful response to emergency events.

Sweden has experienced public emergencies such as wildfires, floods, and landslides, as examples: [https://en.wikipedia.org/wiki/2018\\_Sweden\\_wildfires](https://en.wikipedia.org/wiki/2018_Sweden_wildfires)  
<https://floodlist.com/europe/central-sweden-floods-august-2021>

These disasters, along with explosions, chemical spills, etc., would be the type of emergency covered by the organization.

In such events, people need access to accurate information, allowing them to provide residents with life-saving instructions. The overall objective is to have a successful response to emergency events. To this end, MSB, in collaboration with municipalities and emergency services, is developing an app/site that works in the field and in the office to coordinate disaster management. Face-to-face communication is preferred, but this is not always possible, thus there is a need for an app/site that works in the field and in the office to coordinate disaster management.

This app/site will facilitate effective communication, including roles and tasks, but also account for information confidentiality (e.g., to protect privacy and avoid looting). The app/site will not only be used by emergency responders but will also be available to the public, allowing them to access real-time information and instructions during an emergency.

When an incident happens, the local municipalities and emergency services take the lead in responding to the situation. A formal chain of command is established, and the incident commander is assigned, often the first responder until someone more senior arrives. Emergency workers need to coordinate with MSB to get support and help with coordination. Someone must organize the entire effort to monitor and manage resources (e.g., fire trucks, pumps, ambulances), objectives, and outcomes. People receive work assignments and can request support or additional resources. Workers report the progress of assigned tasks. Responders need to brief others and the public, communicate hazards, acknowledge messages, and ask for further information.

MSB's app/site needs to facilitate effective communication, including roles and tasks. The coordination and communication between the municipalities, emergency services, and MSB are essential to ensure an effective and timely response. The ultimate goal is to ensure that the Swedish society is resilient and able to withstand and recover from any crisis or disaster.

**Problem description used with modifications from case assignments  
written by Jennifer Horkoff for the course DIT046/DAT356**

### Example of stakeholders

MSB employees

The public  
Emergency services - Police, Fire Department, Health care  
Volunteer groups that will help in the case of an emergency.  
The municipality

## Appendix B - Requirements' quality attributes

The Characteristics of a good requirement are:

- 1. Necessary:** The requirement is essential for the capability of the system and should be realistic to implement from the constraints of the product.
- 2. Implementation Free:** Implementation independent, which means not saying how a system should do a requirement from a technical standpoint but still be necessary.
- 3. Unambiguous:** The requirements should only be able to be interpreted one way
- 4. Consistent:** Not conflicting with other ones or be the same thing with different words
- 5. Complete:** The requirement is measurable and containing all the information needed for it to be implemented correctly
- 6. Singular:** The requirement should not be a combination of many requirements
- 7. Feasible:** Should be realistic for the program (based on context)
- 8. Traceable:** Can you understand why the requirements is there (In this case the problem description)
- 9. Verifiable:** The final entity should be testable. For example, good is hard to test

## Appendix C - Template to be used for writing/generating requirements

Title/ID	A unique identifier for the requirement
Description	A clear and concise description of the requirement
Rational	Reason behind the requirement

## Appendix D - Total Attribute Scores After Analysis

Group	Necessary	Implementation Free	Unambiguous	Consistent	Complete	Singular	Feasible	Traceable	Verifiable
P1	10	9	6	10	7	9	10	9	6
P2	10	9	8	10	6	10	10	9	10
P3	10	10	8	10	5	9	10	10	10
P4	10	9	7	10	5	10	9	7	7
P5	10	8	6	10	6	10	9	7	4
P6	10	10	6	10	6	8	9	10	6
P7	10	10	9	10	6	10	10	10	8
P8	10	10	9	10	10	10	10	10	10
P9	8	10	6	10	7	10	9	7	6
P10	10	10	9	10	8	9	10	9	8
P11	9	9	7	10	7	10	9	8	7
ChatGPT_Freeze	10	10	7	10	7	9	10	9	7
ChatGPT_Freeze	10	9	6	10	6	5	10	9	7
ChatGPT_Freeze	10	10	7	8	7	6	9	6	9
ChatGPT_Pre	10	9	6	10	6	6	10	10	6
ChatGPT_Pre	10	10	9	10	6	9	10	10	6
ChatGPT_Pre	10	10	9	10	7	9	10	10	6

## Appendix E - Written Requirements From Participants

101	FR	The system shall provide real-time info about the current disaster, for instance, the location, and level of the emergency.
102	FR	The system shall provide various types of clear instructions to everyone when there is an emergency around.
103	FR	The system shall users to chat/contact each other along with MSB personnel, so they can coordinate more and provide further updates on the emergency and what to do in unique situations.
104	FR	The system should treat the user's data with confidentiality and be secure with the data projected.
105	FR	The system shall allow users to assign roles and tasks.

106	FR	The system should have a 3D or some sort of map that is constantly updated, so users, in general, can see the current state of emergencies around them or in their region.
107	FR	The system shall always notify every user with an alarming notification, whenever an incident occurs.
108	NFR	The system shall have a user-friendly interface.
109	NFR	The system shall automatically assign roles according to that person current location.
110	NFR	The system shall be online 24/7 and can handle a <b>large number</b> of users.
201	FR	The app shall allow the definition of roles and authorities in the case of emergency.
202	FR	The roles allowed to be defined within the app are: - Workers - Responders - Civilians
203	FR	The app shall inform the users when an emergency happens.
204	FR	The message sent by the app shall include the instructions and the roles these instructions are intended to.
205	FR	Upon receiving the message, the user shall be able to request support and/or additional information
206	FR	Workers shall be allowed to report the progress of their work assignments.
207	FR	The app shall allow video communication between the users, when possible.
208	NFR	The app shall allow the definition of backup depending in case of non availability.
209	NFR	The app shall allow the definition of a command chain to handle the emergency.
210	NFR	The app shall move to the next person in the command chain if no response is received within 15 seconds.
301	FR	User management system that supports different roles
302	FR	Allow users with the required privileges to signal an emergency in a particular area
303	FR	Send emergency broadcasts via the mobile phone and radio network
304	FR	Asset management system for the users with appropriate privileges
305	FR	Allow responders to provide status updates from the field to those coordinating the operation.

306	FR	Emergency categorization and description for responders and local authorities
307	FR	Task management system for ongoing field operations
308	NFR	High availability (99.9%) under stress (1000 requests per second)
309	NFR	Low latency (<5ms) under stress (10000 requests per second)
310	NFR	Correctness (0% false positives for emergencies categorized as life-threatening)
401	FR	The app shall be able to check the accuracy of the information
402	FR	a chain of command shall be defined with the different roles
403	FR	If no response is received from a specific role, the next role within the chain of command shall take over.
404	FR	A task shall contain a description of the actions, a deadline and the objective that shall be achieved.
405	FR	the app shall allow for face to face communication
406	FR	if video communication is not possible, the app shall allow for only voice communication as back up mechanism
407	FR	Users shall be able to request support
408	NFR	appropriate cryptography mechanisms shall be used to maintain the privacy of the data within the app
409	NFR	The response time delay shall not exceed 3 seconds.
410	NFR	The app shall use relevant colors to deferenciate between the tasks severities and urgencies
501	FR	The app/site should be able to alert users that live in the disaster region.
502	FR	The app/site should be able to alert users of the size of the dissaster and the danger of it
503	FR	The app should be able to inform the users of the nature of the dissaster (volcano, hurricane, flood etc.)
504	FR	The app should be able to provide an alertwithin a timeframe of the upcoming dissaster before it happens.
505	FR	The app should be able to present the way the dessaster is developing on a map.
506	FR	The app should be able to ring an alarm to the usrrers in a human language .
507	FR	The app should be able to navigate people away from the dissaster.

508	NFR	The system should be able to provide an alert within milliseconds to thousands of users without crashing.
509	NFR	The system should have below 1% > down time.
510	NFR	The app should be very secure.
601	FR	The app shall provide real time information and instructions to the public
602	FR	The app shall provide communication tools like messaging, voice call, video call, broadcasting among different stakeholder
603	FR	The app shall able to assign different roles to different emergency workers
604	FR	The app shall able to create and assign tasks
605	FR	The app shall able to manage different resources like fire trucks, ambulances etc.
606	FR	The app shall ensure information confidentiality and protect privacy.
607	FR	The app shall able to help emergency workers to request for additional resources and support
608	NFR	The app shall work all the times during emergency.
609	NFR	The app shall be easier to navigate and shall have user friendly interface.
610	NFR	The app shall able to handle high volume of users without delays, breaking down or performance issues.
701	FR	The app shall hold accurate information about emergencies.
702	FR	The app shall coordinate the chain of commands in between responders.
703	FR	The app shall distribute task assignments and instructions to the adequate users.
704	FR	The app shall provide the option to request resources.
705	FR	The app shall facilitate direct communication.
706	FR	The app shall provide different user roles.
707	FR	The app shall provide the municipality with overview information.
708	NFR	The information on the disaster must be updated at least every 5 minutes.
709	NFR	The chain of command needs to have correct users assigned to respective roles.
710	NFR	The app needs to be able to work with limited bandwidth.

801	FR	The system shall allow responders to create an account.
802	FR	The system shall allow responders to create news posts.
803	FR	The system shall allow users without an account to view news posts.
804	FR	The system shall display which service/organization a responder belongs to
805	FR	The system shall allow an incident commander to establish an incident report.
806	FR	The system shall allow responders to be assigned to an incident report.
807	FR	The system shall allow responders to report actions to an incident report.
808	NFR	The system shall display a new post on the public feed within 3 seconds 99% of the system up-time.
809	NFR	The system shall log in a user within 1 second 99% of the system up-time.
810	NFR	The system shall correctly update an incident report when 20 responders post updates within a 1 minute time frame.
901	FR	Be able to push out notifications to all phones
902	FR	Different levels of users
903	FR	Personal password
904	FR	The app needs to have access to the camera function if one is available
905	FR	The app needs to have access to video functions if available
906	FR	The app needs to have access to the users location via GPS
907	FR	The app should notify through both sound and vibration
908	NFR	The app needs to be able to handle 10000000 simultaneous users.
909	NFR	The app needs to load within 3 second of pushing the button no matter the amount of users
910	NFR	The app should function in areas of very low reception.
1001	FR	The system must implemented for various platforms and hardware systems so that it can be used from computers, mobile devices and emergency communication equipment.

1002	FR	Depending on stakeholders role, the system must implement quality of service and message priority, so that MSB employees and emergency services communication is highly prioritized and delivered with Level 2 assurance.
1003	FR	The system must implement accessible user interface design so that information can be properly presented via various types of devices.
1004	FR	The system must utilize integrated user authentication system so that user level and role can defined with clear hierarchy.
1005	FR	The system must implement various formats of messages and data so that emergency can send text, audio or video message.
1006	FR	The system must expose public emergency broadcast channel.
1007	FR	The system must be able to access and utilize existing standardized postcode, GPS data and zoning data.
1008	NFR	The system must respond to all requests within 300ms with minimum of 99.999999% reliability.
1009	NFR	The system must be accessible via all designed communication channels in 99.999999% of the time and maximum downtime of 0.005% during any 24 hour period.
1010	NFR	The system must implement user access and access control functions so that only authorized users can receive or send information depending on their assigned users.
1101	FR	The system shall display instructions to follow to the public about ongoing events in their location
1102	FR	The system shall allow professionals to create accounts
1103	FR	The systems shall encrypt all communications between professionals
1104	FR	The systems shall send tasks to professionnals that have to confirm reception
1105	FR	The system shall provide a map with the position of the different professional engaged in the field.
1106	FR	The system shall allow professional to request for additional resources
1107	FR	The system shall allow professionals to update the status of their assigned task
1108	NFR	The system should be available as much as possible

1109	NFR	The system should resist to emergency event
1110	NFR	The system should be easy to use

## Appendix F - Generated Requirements From ChatGPT

### Free Version

GPT-F101	<b>FR</b>	The app/site must be able to display real-time information related to emergency situations.
GPT-F102	<b>FR</b>	The app/site must allow users to report their location and request assistance during an emergency.
GPT-F103	<b>FR</b>	The app/site must enable communication between emergency responders and MSB.
GPT-F104	<b>FR</b>	The app/site must have a formal chain of command that assigns an incident commander during an emergency.
GPT-F105	<b>FR</b>	The app/site must be able to monitor and manage resources such as fire trucks, pumps, and ambulances.
GPT-F106	<b>FR</b>	The app/site must provide an overview of work assignments and progress of assigned tasks.
GPT-F107	<b>FR</b>	The app/site must ensure information confidentiality to protect privacy and avoid looting.
GPT-F108	<b>NFR</b>	The app/site must be scalable to accommodate a large number of users during an emergency.
GPT-F109	<b>NFR</b>	The app/site must have a fast response time to ensure timely communication during an emergency.
GPT-F110	<b>NFR</b>	The app/site must be accessible to people with disabilities.
GPT-F201	<b>FR</b>	The app/site should provide real-time access to accurate information about ongoing emergencies, including updates on the situation, instructions, and safety guidelines.
GPT-F202	<b>FR</b>	The app/site should allow emergency responders to communicate and coordinate their actions in the field and the office, enabling effective collaboration and information sharing.
GPT-F203	<b>FR</b>	The app/site should support the establishment of a formal chain of command and enable the assignment of incident commanders to lead the response efforts.
GPT-F204	<b>FR</b>	The app/site should enable emergency workers to request and allocate resources, such as fire trucks, ambulances, and other necessary equipment, to address the incident effectively.
GPT-F205	<b>FR</b>	The app/site should provide a mechanism for emergency workers to report the progress of assigned tasks, allowing supervisors and incident commanders to track the status of operations.
GPT-F206	<b>FR</b>	The app/site should enable emergency responders to communicate hazards and issue alerts to the public, ensuring timely dissemination of critical information for public safety.
GPT-F207	<b>FR</b>	The app/site should prioritize information confidentiality and incorporate measures to protect privacy and prevent looting during emergencies.

GPT-F208	<b>NFR</b>	The app/site should be user-friendly, with an intuitive interface and easy navigation, to ensure that both emergency responders and the public can use it effectively in high-stress situations.
GPT-F209	<b>NFR</b>	The app/site should have a high level of availability and reliability to ensure uninterrupted access to essential services, even during peak usage periods or in situations with limited network connectivity.
GPT-F210	<b>NFR</b>	The app/site should be scalable to accommodate a large user base and handle increased traffic during emergency events without compromising performance or responsiveness.
GPT-F301	<b>FR</b>	The app should provide real-time updates and information to both emergency responders and the general public during an emergency event, ensuring they have accurate and up-to-date instructions and guidance.
GPT-F302	<b>FR</b>	The app should facilitate effective communication between different stakeholders (municipalities, emergency services, MSB) by providing role-based communication channels, allowing for seamless coordination and collaboration.
GPT-F303	<b>FR</b>	The app should enable emergency workers to manage and track available resources (e.g., fire trucks, pumps, ambulances) to ensure their efficient allocation and utilization during an emergency response.
GPT-F304	<b>FR</b>	The app should allow incident commanders to assign tasks to responders, who can report the progress of their assigned tasks in real-time, providing visibility and accountability in the response efforts.
GPT-F305	<b>FR</b>	The app should facilitate the communication of hazards and risks to responders and the public, enabling the dissemination of critical safety information and warnings.
GPT-F306	<b>FR</b>	The app should support the documentation and sharing of incident briefings, ensuring that all stakeholders are well-informed about the situation and the actions taken.
GPT-F307	<b>FR</b>	The app should be accessible to the general public, allowing them to access real-time information and instructions during an emergency, promoting public safety and awareness.
GPT-F308	<b>NFR</b>	The app should have a user-friendly interface and intuitive navigation to ensure ease of use for both emergency responders and the public, especially in high-stress situations.
GPT-F309	<b>NFR</b>	The app should be capable of handling a large volume of users and data during peak periods of emergencies, ensuring reliable performance and scalability to support widespread usage.
GPT-F310	<b>NFR</b>	The app should prioritize the security and privacy of user information and communications, implementing robust encryption and authentication measures to protect sensitive data and prevent unauthorized access.

#### Premium Version

<b>GPT-P101</b>	<b>FR</b>	The system must have user authentication to differentiate between emergency responders and the public.
<b>GPT-P102</b>	<b>FR</b>	The system must provide real-time incident updates and instructions to public users.
<b>GPT-P103</b>	<b>FR</b>	The system must facilitate role-based task assignment among emergency responders.
<b>GPT-P104</b>	<b>FR</b>	The system must allow emergency responders to report the progress of assigned tasks.
<b>GPT-P105</b>	<b>FR</b>	The system must enable communication between emergency responders and MSB for resource requests and support.

GPT-P106	FR	The system must provide a mechanism to communicate hazards to all users.
GPT-P107	FR	The system should have a feature for acknowledging messages and asking for further information.
GPT-P108	NFR	The system must be available 24/7 with minimal downtime.
GPT-P109	NFR	The system should be user-friendly and intuitive.
GPT-P110	NFR	The system should maintain data confidentiality and privacy.
GPT-P201	FR	The system should provide real-time updates about ongoing disasters and emergencies.
GPT-P202	FR	The system should facilitate secure and confidential communication between emergency services and MSB.
GPT-P203	FR	The system must allow emergency responders to report on the progress of assigned tasks.
GPT-P204	FR	The system should allow the incident commander to assign roles and tasks to emergency workers.
GPT-P205	FR	The system must facilitate requests for support or additional resources during an emergency.
GPT-P206	FR	The system should provide a feature for acknowledging messages and requests.
GPT-P207	FR	The system must allow the briefing of emergency situations to both responders and the public.
GPT-P208	NFR	The system must be capable of handling high traffic during times of emergencies without degradation of performance.
GPT-P209	NFR	The system should be easily accessible and user-friendly for both emergency workers and the general public.
GPT-P210	NFR	The system should provide secure and encrypted communication channels.
GPT-P301	FR	The system must be able to send real-time alerts and notifications to users in the event of an emergency.
GPT-P302	FR	The system must provide a feature to assign and track roles and tasks among emergency workers.
GPT-P303	FR	The system must allow emergency workers to report the progress of assigned tasks.
GPT-P304	FR	The system should support secure communication channels for emergency responders and MSB coordinators.
GPT-P305	FR	The system must facilitate real-time public briefings about the emergencies.
GPT-P306	FR	The system must be able to receive requests for support or additional resources from emergency workers.
GPT-P307	FR	The system should provide a feature for incident commanders to establish and communicate the formal chain of command.
GPT-P308	NFR	The system must maintain high availability and reliability during high traffic periods such as during emergencies.

<b>GPT-P309</b>	NFR	The system must adhere to stringent data protection regulations to protect users' privacy.
<b>GPT-P310</b>	NFR	The system must have a user-friendly interface that's easily navigable even in stressful situations.

## Appendix G - Collected Requirements From Participants and ChatGPT in Terms of Quality

