Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Assisting early-stage software startups with LLMs: Effective prompt engineering and system instruction design

Thea Lovise Ahlgren [a], Helene Fønstelien Sunde [a], Kai-Kristian Kemell [c], Anh Nguyen-Duc [a,b,*]

[a] Norwegian University of Science and Technology, Trondheim, 7012, Norway
[b] University of South Eastern Norway, Bøi Telemark, 3800, Norway
[c] Tampere University, Faculty of Information Technology and Communication Sciences, Tampere, Finland

ARTICLE INFO

Keywords:
Generative Artificial Intelligence
GenAI
ChatGPT
StartupGPT
Software engineering
Software startup
Software development
Design Science Research

ABSTRACT

Context: Early-stage software startups, despite their strong innovative potential, experience high failure rates due to factors such as inexperience, limited resources, and market uncertainty. Generative AI technologies, particularly Large Language Models (LLMs), offer promising support opportunities; however, effective strategies for their integration into startup practices remain underexplored.
Objective: This study investigates how prompt engineering and system instruction design can enhance the utility of LLMs in addressing the specific needs and challenges faced by early-stage software startups.
Methods: A Design Science Research (DSR) methodology was adopted, structured into three iterative cycles. In the first cycle, use cases for LLM adoption within the startup context were identified. The second cycle experimented with various prompt patterns to optimize LLM responses for the defined use cases. The third cycle developed "StartupGPT", an LLM-based assistant tailored for startups, exploring system instruction designs. The solution was evaluated with 25 startup practitioners through a combination of qualitative feedback and quantitative metrics.
Results: The findings show that tailored prompt patterns and system instructions significantly enhance user perceptions of LLM support in real-world startup scenarios. StartupGPT received strong evaluation scores across key dimensions: satisfaction (93.33%), effectiveness (80%), efficiency (80%), and reliability (86.67%). Nonetheless, areas for improvement were identified, particularly in context retention, personalization of suggestions, communication tone, and sourcing external references.
Conclusion: This study empirically validates the applicability of LLMs in early-stage software startups. It offers actionable guidelines for prompt and system instruction design and contributes both theoretical insights and a practical artifact — StartupGPT — that supports startup operations without necessitating costly LLM retraining.

1. Introduction

Software startups play a notable role in economic growth by driving innovation, creating jobs, and enhancing market competition [1], with, for example, over $40 billion invested in European tech startups every year after 2020 [2]. However, over 90% fail [3], although so do 98% of new product ideas in general [4]. Startup failures are often associated with challenges like limited budgets, small teams, inexperience, and market uncertainty [5–7]. A startup's strategy for addressing these challenges can significantly influence its success, as self-destruction is a common cause of failure [3], where product development is prioritized over understanding these challenges early on [8].

To address these challenges, various software engineering (SE) approaches have been proposed, including Lean Startup methodologies, MVP planning frameworks, and lightweight development techniques. However, the adoption of such tools and practices remains limited in startup environments, often due to time constraints and lack of awareness. Consequently, there is growing interest in exploring digital support tools that can alleviate operational burdens and support informed decision-making [9,10].

Generative AI, and particularly Large Language Models (LLMs), has been argued to be a potentially promising solution to these challenges [11,12]. Ebert and Louridas summarize Generative AI (GenAI) as AI models that "can synthesize — or generate — the answers to the questions you pose", a type of machine learning models (ML) that generate text, audio, images, and video and various technical approaches used to build such models [13]. LLMs are a subset of GenAI models that focus on generating textual content ranging from poems

* Corresponding author at: Norwegian University of Science and Technology, Trondheim, 7012, Norway.
E-mail address: Anh.Nguyen.duc@usn.no (A. Nguyen-Duc).

https://doi.org/10.1016/j.infsof.2025.107832
Received 10 September 2024; Received in revised form 22 June 2025; Accepted 26 June 2025
Available online 9 July 2025

to program code [13,14]. For startups, LLMs could offer practical support in overcoming key hurdles, including product validation, business planning, and understanding market dynamics [15]. Their utility is especially compelling for less experienced founders, for whom access to real-time, tailored advice could be game-changing [16].

Despite this potential, the use of GenAI in software startups remains underexplored. While recent studies have examined LLMs in broader SE contexts, few have focused on startups specifically (e.g., [17]), and even fewer have engaged with real-world practitioners (e.g., [18]. Given that software startups differ substantially from mature organizations — often operating with smaller teams, flatter structures, and unique development rhythms — there is a clear need for targeted research into how GenAI can be effectively applied in these settings [5,19–21].

Outside the specific context of software startups, existing studies on LLMs in SE have explored their use for various SE tasks [22]. Various potential benefits associated with LLMs and GenAI can potentially aid with common software startup challenges identified in existing literature. For example, less experienced startup practitioners may particularly benefit from the reported benefits of facilitating information retrieval and learning [23,24]. Similarly, some of the current industrial use cases highlighted in existing literature (e.g., [23,24]) may be particularly interesting for software startups as well, from the point of view of the challenges they face, and their unique characteristics.

One non-trivial problem with adopting LLMs is the cost and required competence. Training LLMs is very expensive, and even fine-tuning existing LLMs takes data and effort. While cloud-based LLM services like OpenAI's ChatGPT,[1] Google Gemini,[2] and Microsoft CoPilot[3] offer high-performing, generic LLMs that are easy to adopt, and which are widely used for various tasks in companies currently [25], there are still challenges associated with the use of GenAI as well [23,24]. For example, aside from data privacy issues, companies sometimes struggle to find relevant use cases for GenAI [24]. In particular, learning to prompt LLMs requires experimentation and trial and error, and evaluating the effectiveness of prompts is difficult [23].

This study addresses that gap by exploring how LLMs can support the operational activities of software startups. We adopt a Design Science Research (DSR) approach to develop and evaluate StartupGPT, an LLM-based assistant tailored to early-stage startup needs. The system is designed to assist with core challenges such as MVP development, market analysis, funding preparation, and business planning—areas where startups often lack internal expertise. Our research question is:

RQ: *How Can Large Language Models Be Utilized to Support Software Startups in Their Operational Activities?*

Our contributions are threefold:

- We provide a validated set of use cases where LLMs can meaningfully assist startups, serving as a blueprint for future GenAI tools in this domain.
- We offer design insights on effective prompt patterns and system instructions that optimize LLM responses in startup contexts, without requiring costly model fine-tuning.
- We present one of the first empirical evaluations of LLM for software startups

In doing so, this work contributes both a practical artifact and foundational knowledge to the emerging intersection of GenAI and startup software engineering.

The paper is organized with the following structure: Section 2 presents the background on GenAI in Software Engineering, Software startup challenges, Research about GenAI in the startup context and concrete LLM adaption techniques, i.e. prompt engineering and system instructions. Section 3 presents our research approach with details of design science; Section 4 reports our result; Section 5 discusses the paper and Section 6 concludes the paper.

2. Background

2.1. *Generative AI and LLMs in software engineering*

Using AI (and specifically Machine Learning (ML)) overall for SE is a long-standing area of research, covering a vast variety of ML tools split across a wide variety of SE tasks [26], such as testing [27]. Recent interest towards GenAI and LLMs has resulted in an additional surge of studies on AI for SE, with many recent studies specifically exploring the use of GenAI, and usually specifically LLMs, for SE tasks, i.e. requirement classification, code generation and summarization, test generation, program repair, supporting project planning and management [16,22,28,29]. This widespread interest in GenAI is hardly limited to academia. A Gartner practitioner survey reported that, by 2024, GenAI was already the most frequently deployed type of AI solution in organizations [25], while other practitioner surveys by GitHub [30] and JetBrains [31] report widespread adoption of GenAI among practitioners. While many studies and such practitioner surveys often speak of GenAI more widely, the focus in practice is often on LLMs specifically (c.f., the literature review of Hou et al. [22] and the survey of Fan et al. [28] on LLMs for SE).

Research on LLMs in SE encompasses both the creation of various related artifacts and studies exploring the use of existing LLMs in different contexts. In terms of artifact development, existing literature proposes, for example, fine-tuned LLMs for specific use cases such as code generation [28]). In terms of using existing LLMs and LLMs overall, existing studies (a) explore the use of specific LLM services (e.g., GitHub CoPilot [32]) for certain tasks in certain contexts, (b) explore the current state of practice and GenAI adoption in industrial SE settings (e.g., [23,24,33]), (c) explore changes to SE resulting from LLMs and GenAI (e.g., AI pair programming [34,35]), and (d) discuss good practices for LLM use (e.g., prompting for SE [36]), among other viewpoints.

The use of LLMs for SE is still a nascent area of research. Some early studies looking at industrial use and adoption of LLMs, or GenAI more widely, report various problems companies currently face in making use of these tools. These include data privacy issues and regulatory issues, organizational bureaucracy combined with the fast-moving field, lack of clear or suitable use cases, quantifying or understanding the benefits of these tools, in addition to challenges related to LLM outputs such as generic outputs, incorrect outputs, issues related to training data cut-off dates, or otherwise low-quality outputs [23,24,33,37]. We argue that some of these challenges may not be as important for software startups. This makes software startups particularly interesting from the point of view of studying LLMs in SE practice. We further build on this line of reasoning in the following subsection as we discuss SE in software startups.

2.2. *Software startups and their challenges*

Much of the existing SE research into software startups is built on the idea that software startups differ from other types of software organizations, and thus are worth studying separately as well. Unterkalmsteiner et al. argue the following in this regard: "software startups are quite distinct from traditional mature software companies, but also from micro-, small-, and medium-sized enterprises, introducing new challenges relevant for software engineering research." [19] These differences are argued to stem from various factors, with Paternoster et al. [21] listing 15 characteristics found in literature that studies use to characterize software startups: (1) highly reactive, (2) innovation, (3) uncertainty, (4) rapidly evolving, (5) time-pressure, (6) third party

[1] https://openai.com/chatgpt/.
[2] https://gemini.google.com/.
[3] https://copilot.microsoft.com/.

dependency, (7) small team, (8) one product, (9) low-experienced team, (10) new company, (11) flat organization, (12) highly risky, (13) not self-sustained, (14) lack of resources, and (15) little working history. To this end, it is hardly always clear what comprises a software startup, and definitions provided in research vary [38]. The general idea, however, is that a software startup is a temporary organization that ultimately either becomes a conventional software organizations, or fails somewhere along the way [39]. This journey is conceptualized in various software startup life-cycle models (e.g., [40]).

In terms of software engineering, software startups are also argued to differ from mature software organizations. As an example, we use the adoption and use of agile approaches, which remain industry standard in SE today. Paternoster et al. [21] posit that "agile and more traditional methodologies struggle to get adopted by startups due to an excessive amount of uncertainty and high time-pressure". On the other hand, Bosch et al. [41] argue that agile methods have a fundamental mismatch with the software startup context, being intended for SE contexts "where the problem is fairly well understood but the solution is not", while the startup context is one where "neither the problem nor the solution is well understood". While software startups do utilize individual agile practices [42,43], this is often done in a more ad hoc manner [5]. Klotins et al. [44] argue that many seemingly business-related startup failures may in fact stem from SE issues in practice. This is also why the StartupGPT we present in this paper includes business-oriented use cases, as we discuss later. Second, lack of resources is consistently associated with SE in software startups [20,21,41]. Third, quality has low priority [45], and speed is prioritized in order to more quickly get a product out [43]. Fourth, so-called "validation" activities, which can be likened to requirements engineering in conventional SE, are considered important in helping ensure product–market fit, or more broadly that there really is a demand for what is being developed [38]. Yet many software startups, in practice, develop software without having ever had meaningful contact with their would-be customers or users [44].

2.3. Adopting generative AI for software startups

Similarly to how past studies have explored agile methodology and practice use in software startups, we consider software startups to be an interesting avenue of research in relation to LLM use and adoption. Returning to the challenges we discussed in Section 2.1 in relation to GenAI and LLM adoption and use in industrial SE contexts *(data privacy and regulatory issues, organizational bureaucracy, lack of use cases, quantifying benefits, and issues with LLM outputs)*, we argue that these may either be less relevant for software startups in the light of what is known about SE in software startups, or these may manifest differently in software startup contexts.

Startups may have an easier time exploring and possibly finding use cases for LLMs as organizations working with a severe lack of resources. LLMs have been argued to be useful for eliminating menial tasks in industrial SE contexts, which may be particularly helpful for an early-stage startup with various "menial" tasks to work on (e.g., setting up a website, setting up initial codebase, etc.). As one more example, it is not uncommon for software startups to have an inexperienced team and/or founder [21], making the argued LLM benefits of facilitating information retrieval and facilitating learning potentially more useful [23,24].

Currently, only a few studies have explored LLMs (or GenAI more widely) in the software startup context. Simaremare et al. [17], in a conceptual paper, propose potential GenAI use cases for software startups based on existing literature. Triando et al. [18] conduct a case study of two startups, reporting 11 opportunities or use cases and 10 challenges. We consider the challenges more relevant here as they can be related to the contributions of StartupGPT, and, thus, the 10 challenges were: (1) biased responses, (2) cannot represent the actual users, (3) felt unnatural, (4) pricy, (5) hard to construct meaningful prompts, (7) function-oriented instead of problem-oriented, (8) tend to please instead of being critical, (9) lacks transparency, and (10) unreliable responses [18]. We argue that using tools such as StartupGPT over generic, out-of-the-box LLM tools and services can help tackle some of these problems, as we discuss in more detail later. Gupta [46] studies the adoption of GenAI in startups, focusing on factors influencing technology adoption in relation to GenAI. Rezazadeh et al. [47], on the other hand, study GenAI use in relation to growth hacking specifically, with growth hacking referring to a form of marketing commonly utilized by startups. Nevertheless, few studies on LLMs and GenAI in startup contexts currently exist, and especially SE studies specifically focused on software startups are currently notably scarce.

2.4. Prompt engineering and system instructions

To effectively harness the potential of LLMs for startup support, it is essential to understand how prompt engineering and system instructions influence the quality and relevance of model outputs—hence, their inclusion as a foundational component in the background of this study.

2.4.1. Prompt engineering

Interacting with LLM-based tools (ChatGPT, Claudie, etc.) involves writing instructions called prompts to the model so it can generate a textual response. A prompt is a natural language instruction sent to the model to provide it with the context of the interaction [48]. In a chat-based model, such as ChatGPT, the prompt would be the user's message [49]. Writing effective prompts is an essential part of communication with LLM-based tools and a technique to achieve this is prompt engineering. White et al. define prompt engineering as the *"means by which LLMs are programmed via prompts"* [50]. A prompt customises, enhances or refines the abilities of an LLMs by giving a set of instructions as to what the content of the output should be and how it should be formatted. A prompt affects the output the model generates by setting a list of rules and guides for the conversation and providing a context for the interaction [49].

Effective prompt engineering is necessary to bridge the gap between user intent and model understanding. As explained by Ekins, an LLM may not understand questions and information in the same way as humans would. Prompt engineering takes into account the model's training data, biases and limitations that might influence the model's understanding [51]. According to Ekin, constructing a prompt that leads to high-quality answers depends on several factors, including five key elements:

1. **User intent:** Understand the purpose of the interaction, to craft a prompt that aligns with the user's goal and desired output.
2. **Model understanding:** Familiarise yourself with the strengths and limitations of the LLM to craft a prompt adapted to take advantage of the model's capabilities, while mitigating its weaknesses.
3. **Domain specificity:** Provide context and domain-specific language to guide the LLM towards more accurate and relevant outputs.
4. **Clarity and specificity:** Avoid unclear or vague instructions without sufficient context, as this can lead to ambiguity and misunderstandings.
5. **Constraints:** Clearly define constraints, if necessary. Specifying requirements, such as output length or format, help guide LLMs towards providing desired responses.

To illustrate the effectiveness of prompt engineering, consider the two prompts in Fig. 1, both addressing MVP development in startups. Both Prompt 1 and Prompt 2 cover similar topics; however only Prompt 2 incorporates prompt engineering techniques. While Prompt 1 is broad and general, Prompt 2 gives the LLM constraints and key focus areas guiding it to provide detailed, relevant, and practical advice. Furthermore, it causes the LLM to ask questions until sufficient information about the startup is obtained to give targeted responses.

Prompt 1:

What should I consider when creating a minimum viable product for a tech startup?

Prompt 2:

From now on, I would like you to ask me targeted questions specifically about developing a minimum viable product (MVP) for a technology startup. Focus on aspects such as identifying core features, balancing budget constraints, assessing customer needs, and timelines for development and testing. When sufficient information is obtained, formulate a concise action plan detailing the steps for creating and launching a successful MVP.

Fig. 1. Comparison of prompts with and without prompt engineering techniques. Prompt 1 is broad and general while Prompt 2 incorporates prompt engineering techniques to get more detailed and targeted replies from ChatGPT.

Table 1
Prompt patterns, descriptions, and example prompts [49].

Pattern	Description	Example prompt
Flipped interaction	The LLM drives the conversation to achieve the user's goal by asking questions to gather sufficient information.	*"I would like you to ask me questions about the economic aspects of my startup (such as income, expenses, savings, etc.) until you have enough information to set up an example of a monthly budget for my startup."*
Cognitive verifier	The LLM breaks down a high-level question into sub-questions to gather more information and provide a better answer.	*"When I ask a question, generate at least three follow-up questions to help you answer my question. When I have answered the questions, generate the answer to my original question."*
Alternative approaches	The LLM provides multiple approaches to solving a problem, highlighting the pros and cons of each.	*"When I ask you for a progress plan for developing my startup, provide me with different solutions based on different relevant frameworks. List the pros and cons of the different approaches."*
Question refinement	The LLM suggests more precise or better ways to phrase a user's question to help them find the answer they seek.	*"From now on, when I ask a question about software architecture, suggest a better version of the question and ask me if I want to use that question instead."*
Persona	The LLM takes on a specific perspective or role to focus on relevant details in the output.	*"Act as a potential sponsor for a software startup and give me feedback on my sales pitch."*
Context manager	The output is focused around a specified context, helping the LLM provide more relevant answers by ignoring irrelevant content.	*"Analyse the code I give you with focus on scalability."*

2.4.2. Prompt patterns

Prompt patterns are specific techniques that aim to enhance the quality of a prompt. White et al. propose a list of prompt patterns for successful conversations with LLMs, based on different output and interaction goals [49,50]. This list includes the persona pattern, the flipped interaction pattern and the context manager pattern, which is described in Table 1. Few-shot examples and Chain-of-thought prompting are also described as they have shown promising capabilities in increasing the reliability and effectiveness of chatbot outputs [52,53]. It is however, worth noting that crafting optimal prompts is an iterative process and Ekin states that obtaining optimal responses often requires experimenting with different prompt variations [51].

2.4.3. System instructions

The system instruction, also referred to as meta prompt, system message and system prompt [54], consists of preconfigured guidelines embedded within the LLM by a developer guiding the LLM to how it should behave in user interactions [55]. The system instructions differ from the user prompts, as it gives high-level instructions to the LLM with the aim of shaping its behavior and improving system performance [54]. Common use cases of such instructions include setting the tone and context of the conversation, improving accuracy, defining the output format and setting constraints [54]. As an example, the default system instruction of ChatGPT includes "You are a helpful assistant" [55], however, modifying this to a relevant use case can improve the system performance.

Microsoft has for instance created a guide for creating system instructions [54]. This documentation presents a system message framework for LLMs, which gives recommendations for techniques that can be used alongside prompt engineering techniques for improving the performance of LLMs with system instructions. The framework highlights four practices, which are listed below.

- Define the model's profile, capabilities, and limitations for your scenario
- Define the model's output format
- Provide examples to demonstrate the intended behavior of the model
- Provide additional behavioral guardrails

However, the techniques used to create system instructions need to be validated and assessed for the particular use case they are intended for. In this study we applied and validated the effectiveness of these practices in designing an LLM-based assistant for software startups.

3. Research methodology

3.1. Design science research

In this study, we adopt Design Science Research (DSR) to develop an artifact: an LLM-based chatbot aimed at assisting software startups. DSR has gained prominence in software engineering (SE) research for its emphasis on artifact creation through systematic development and knowledge generation [56–58]. Artifacts in DSR can take the form of constructs, models, methods, instantiations, or their combinations [56]. Our primary objective is not merely to build the chatbot, but to generate and validate knowledge regarding whether an LLM chatbot enhanced with startup-focused research can effectively meet startup founders' consultancy needs. Given the exploratory and uncertain nature of this research, DSR's iterative cycles of design, experimentation, and refinement are particularly well-suited. Although the artifact developed is a chatbot, our emphasis remains on contributing to scientific knowledge by addressing our research question, rather than on the practical deployment or real-world evaluation of the system.

Following the guideline by Hevner et al. [59], we implement the six steps of a DSR cycle: (1) Problem identification, (2) Objectives for solution, (3) Design and development, (4) Demonstration, (5) Evaluation and (6) Communication.

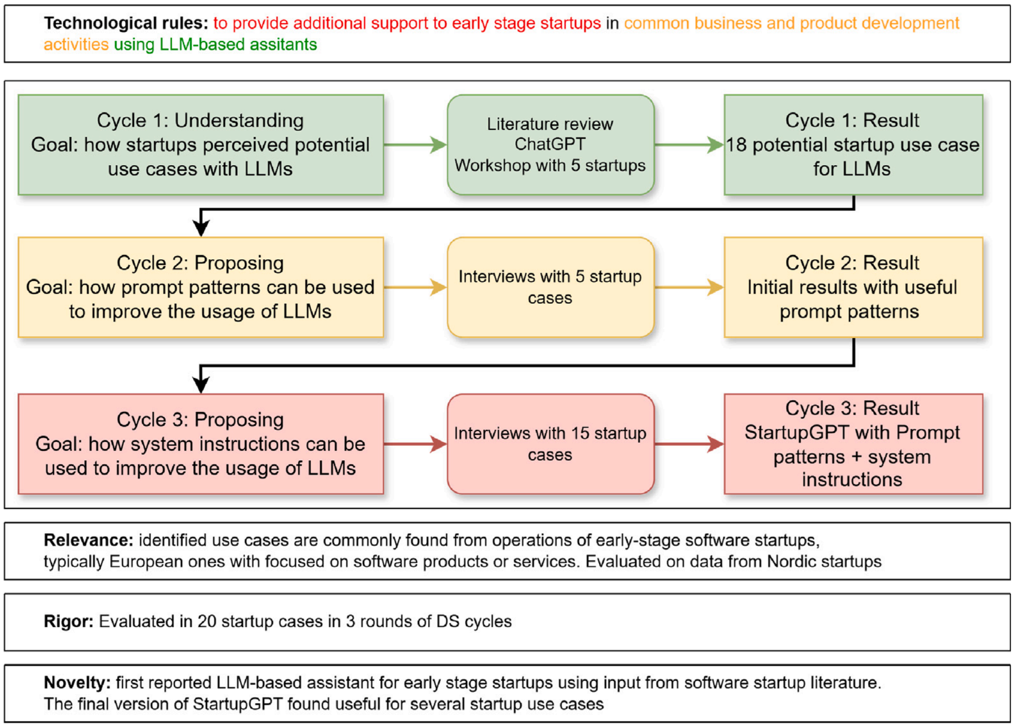
Technological rules: to provide additional support to early stage startups in common business and product development activities using LLM-based assitants
Cycle 1: Understanding
Goal: how startups perceived potential use cases with LLMs
Literature review
ChatGPT
Workshop with 5 startups
Cycle 1: Result
18 potential startup use case for LLMs
Cycle 2: Proposing
Goal: how prompt patterns can be used to improve the usage of LLMs
Interviews with 5 startup cases
Cycle 2: Result
Initial results with useful prompt patterns
Cycle 3: Proposing
Goal: how system instructions can be used to improve the usage of LLMs
Interviews with 15 startup cases
Cycle 3: Result
StartupGPT with Prompt patterns + system instructions
Relevance: identified use cases are commonly found from operations of early-stage software startups, typically European ones with focused on software products or services. Evaluated on data from Nordic startups
Rigor: Evaluated in 20 startup cases in 3 rounds of DS cycles
Novelty: first reported LLM-based assistant for early stage startups using input from software startup literature. The final version of StartupGPT found useful for several startup use cases

Fig. 2. Design science cycles.

1. Problem identification and motivation: define the research problem and use it to motivate the importance of a solution.
2. Define the objectives for a solution. Use knowledge gathered about the problem and existing solutions to define what is possible and feasible in a solution.
3. Design and development: the methods, artifacts, or models. This activity includes the design of functionality and architecture, as well as the development of the product.
4. Demonstration: shows the use of the product in experimentation, a case study, simulation or another activity.
5. Evaluation: Observe and measure how well the product holds up as a problem solution. This can include comparisons of functionality, performance measures, client feedback, or satisfaction survey results.
6. Communication: documents and shares the research and its outcomes with a wider community.

We implemented three cycles in a duration of 10 months, from Aug 2023 to Jun 2024. We followed the suggestion by Engstrom to describe the contribution of DS research with a visual abstract [57]. Fig. 2 gives an overview of DS cycles and their contribution.

3.2. Description of the DSR cycles

In Cycle 1, we focused on identifying use cases for using LLMs in a startup context. In Cycle 2, we explored the adoption of prompt engineering and prompt patterns to enhance startupers' conversation with LLMs. In Cycle 3, we explored the usage of system instructions and built a prototype of StartupGPT for early-stage startups.

3.2.1. Cycle 1 - GenAI use cases in software startups

Problem identification. Cycle 1 aimed at identifying and exploring specific areas within software startup operations where Generative AI technologies could be most effectively applied. A methodological overview of Cycle 1 is presented in Fig. 2.

Objective. The objective of this cycle was to develop a list of feasible use cases for software startups.

Design and development. We conducted a structured process, starting with identifying possible use cases. The identification of use cases began with compiling a list of challenges and practices by startup founders, as reported in the software startup literature (Table 2. This initial list was then complemented by insights drawn from gray literature, including blogs, practitioner articles, and other industry sources that reflect current startup practices. Additionally, we incorporated ideas generated through exploratory sessions with ChatGPT to broaden the scope and capture emerging opportunities not yet widely documented. The combined list of potential use cases was reviewed, refined, and consolidated into a set of 18 final use cases. Each use case was carefully analyzed for its relevance and potential value.

To prioritize the use cases, we conducted a workshop involving five startup founders, where each use case was assessed and triaged using an impact–effort matrix. This collaborative evaluation helped ensure that the selected use cases were both practically significant and realistically achievable for startup environments. Starting from a list of 100 potential use cases for GenAI in startup, after evaluation, we came up with 18 use cases.

Demonstration. We analyzed and consolidated the use cases and challenges into a list of 18 relevant applications of Generative AI in software startups. These use cases were selected with startup founders in mind and, therefore, do not necessarily include detailed technical activities such as code generation, test case creation, or code repair. Instead, the list emphasizes broader themes where AI can significantly enhance startup operations (see Table 3).

Evaluation. The use cases were evaluated and ranked by their perceived impact on startups and feasibility. The impact was based on insights from interviews with five startup founders and a literature review, attributing high impact to use cases given substantial emphasis while assigning lower impact to those less frequently mentioned or not posing significant challenges for startups. The five most promising use cases were then selected, and the impact and feasibility of these use cases were described in more detail. The use cases are highlighted in Table 8.

Table 2
A list of the included articles for startups' use cases.

Paper ID	Title	Info. type
P01	Bosch, J. et al. (2013). *The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups.* Springer	Activities
P02	Cico, O. et al. (2020). *An Empirical Investigation on Software Practices in Growth Phase Startups.* EASE.	Practices
P03	Duc, A. N., and Abrahamsson, P. (2017). *Exploring the Outsourcing Relationship in Software Startups: A Multiple Case Study.* EASE.	Challenges
P04	Duc, A. N. et al. (2020). *Fundamentals of Software Startups: Essential Engineering and Business Aspects.* Springer.	Challenges, Practices
P05	Giardino, C. et al. (2015). *Key Challenges in Early-Stage Software Startups.* XP.	Challenges
P06	Giardino, C. et al. (2016). *Software Development in Startup Companies: The Greenfield Startup Model.* IEEE TSE.	Activities, Practices
P07	Giardino, C. et al. (2014). *Why Early-Stage Software Startups Fail: A Behavioral Framework.* Springer.	Challenges
P08	Khanna, D. et al. (2018). *From MVPs to Pivots: A Hypothesis-Driven Journey of Two Software Startups.* Springer.	Practices
P09	Nguyen-Duc, A. et al. (2021). *The entrepreneurial logic of startup software development: A study of 40 software startups.* Empirical Software Engineering.	Challenges, Practices
P10	Nguyen-Duc, A. et al. (2016). *Towards an Early Stage Software Startups Evolution Model.* SEAA.	Practices
P11	Nguyen-Duc, A. et al. (2017). *What Influences the Speed of Prototyping? An Empirical Investigation of Twenty Software Startups.* XP.	Activities
P12	Paternoster, N. et al. (2014). *Software development in startup companies: A systematic mapping study.* IST.	Activities, Challenges

Table 3
Overview of the participants in the workshop interviews for Cycle 1.

User	Company role	Stage	Industry	Year of business	ChatGPT experience	Year of birth	Location	Type of user test
User 1	CTO	Seed	Software industry	1–5 years	Intermediate	30–50 years	Oslo, Norway	Workshop
User 2	CFO	Seed	Information and communication	1–5 years	Experienced	18–30 years	Trondheim, Norway	Workshop
User 3	CEO	Seed	Software industry	1–5 years	Experienced	18–30 years	Tampere, Finland	Workshop
User 4	CEO	Seed	Finance	1–5 years	Experienced	18–30 years	Oulu, Finland	Workshop
User 5	CEO	Seed	Professional, Scientific and technical activities	<1 year	Intermediate	50–70 years	Oslo, Norway	Workshop

3.2.2. Cycle 2

Problem identification. Building upon the insights from Cycle 1, Cycle 2 started to experiment with the usage of Generative AI in assisting the identified use cases. The motivation was to explore different techniques to improve the relevancy and quality of ChatGPT's response.

Objective. The objective of this cycle was to evaluate the effectiveness of prompt patterns to create effective system instructions to guide interactions with startups.

Design and development. A prototype for the virtual assistant was developed according to Melegati's guideline [60].

Prompt patterns from the categorized introduced by White et al. [49] were considered for the solution. Due to relevance and our evaluation, the selected patterns included Flipped Interaction, Persona, Cognitive Verifier, Alternative Approaches, Context Manager, Question Refinement, as described in Table 1. These five were selected base on our manual testing.We tested each individual prompt pattern to see how they affected the virtual assistant's output. We also tested various combinations of the prompts to see if the combination of prompt techniques further improved the responses. We also tested different formulations of the prompts were tested, including different wordings of the prompts, and splitting the prompts up into several shorter prompts in the chat.

Demonstration. For each of the five use cases, one prompt was developed in order to ensure that users would get specialized answers to questions in different scenarios represented by the use cases.

Evaluation. The evaluation step of Cycle 2 involved user testing with 5 startup founders. Users were gathered from NSE, Gründerbrakka, and through authors' professional networks. To gather users, posters and flyers encouraging users to partake in the user test were handed out and hung up around the NSE areas at the NTNU campus. The poster was created with the guidelines from [61] to make encouraging posters, where keywords were keeping the poster simple, friendly, motivating, informative and short (see Table 4).

For each test case, a participant was asked to conduct a task by interacting with the virtual research assistant, followed a certain instruction given by the researchers. The participants were free to ask the questions they wanted within the context of the given task. The experience with the virtual assistant is evaluated using a 3C framework, a known framework for evaluating software requirements [62]. When applying this framework to evaluate the quality of responses from a virtual assistant, each dimension serves to ensure that the generated answers align with both user expectations and the surrounding context. Consistency refers to the logical coherence of the assistant's responses. A response is consistent if it contains no internal contradictions and maintains alignment with the user's original prompt. This includes both internal consistency within a single response and contextual consistency across the interaction, ensuring that the answer logically follows from the conversation history and respects the user's intent. Completeness assesses whether the assistant's answer includes all necessary and relevant information. A complete response addresses all aspects of the user's input, especially when multiple questions are asked. It should not omit any essential components that are required for a full understanding or for the task to be properly addressed. Correctness ensures that the information provided is factually accurate and trustworthy. A correct response should align with verified sources and reflect the current state of knowledge. Additionally, correctness includes maintaining contextual accuracy, ensuring that the answer remains appropriate given the ongoing conversation and previously established facts.

For the participants conducting the observed user tests (in both Cycle 2 and Cycle 3), a short semi-structured interview was conducted at the end to further explore the participants' perception of the chatbots

Table 4
Overview of the participants in the workshop interviews for Cycle 2.

User	Company role	Stage	Industry	Year of business	ChatGPT experience	Year of birth	Location	Type of user test
User 1	CEO	Seed	Software industry	1–5 years	Intermediate	30–50 years	Oslo, Norway	User testing
User 2	CEO	Seed	Logistics	1–5 years	Experienced	18–30 years	Trondheim, Norway	User testing
User 3	CEO	Seed	Journalism	5–10 years	Experienced	30–50 years	Trondheim, Norway	User testing
User 4	CEO	Seed	Marketing	1–5 years	Experienced	30–50 years	Trondheim, Norway	User testing
User 5	CEO	Seed	Professional, Scientific and technical activities	<1 year	Intermediate	50–70 years	Oslo, Norway	User testing

and their usability. The researchers follow an interview script for guidance to ensure all topics are covered, but may ask supplementary questions or change the order [56,61]. This allows the researchers to gather more detailed information about the topics, and the interviewees get the chance to introduce issues they find relevant to the questions. Each interview lasts between 10 to 20 min.

3.2.3. Cycle 3

Problem Identification. From Cycle 2, prompt patterns is useful, but not sufficient for the expected quality of output from the virtual assistant. Many output were found to be too general and lacked relevancy to the startup context.

Objective. This cycle aimed to explore the application of a custom system instruction based on prompt engineering techniques to enhance ChatGPT's utility for startups.

Design. To understand how the system instruction affected the perceived usefulness, the three versions of the assistant were set up with different modifications, as shown in Table 5. One was *GPT-3.5 Turbo* and one was *GPT-3.5 Turbo* with the system instruction. Comparing the modified model to its baseline model (MVP2.1 vs. MVP2.2) was deemed useful for understanding how much the system instruction affected usability. The third model used *GPT-4* (MVP2.1 vs. MVP2.3) to assess the effect of the model upgrade alone, without introducing other confounding variables like additional behavioral steering through system prompts. **Development.** The three MVPs are created using the open-source Python framework Streamlit.[4] The application uses MongoDB database anda customized user interface, which allows us to implement several enhanced features for data collection:

- Context information is included in the use case prompt, as shown in Table 6)
- System Instruction. The instruction creation process consisted of three steps, which together laid the basis for the guidelines that were used in the creation of the system instruction.
  - Analyzing CustomGPT Instructions. To understand effective system instruction design, we analyzed system prompts generated by OpenAI's CustomGPTs. Ten custom assistants were created and their instructions examined for recurring themes, with comparisons made against Microsoft's system instruction guidelines.
  - Applying Prompt Engineering Techniques. Drawing on insights from prior experimentation, we incorporated proven prompt patterns — such as persona, flipped interaction, and context manager — to guide the assistant's behavior. These techniques shaped the system's tone, clarity, and task orientation for startup-specific scenarios.
  - Integrating User Feedback. User feedback from earlier cycles informed refinements to the system instructions. Based on requests for more critical and context-aware responses, the assistant's persona was redefined as a "startup mentor", ensuring balanced, domain-relevant guidance across various startup roles.

The resulting system instruction can be seen in Fig. 3.

[4] https://streamlit.io/.

Table 5
Prototype versions and configuration details.

Prototype	Prompt pattern	System instruction	GPT Version
MVP2.1	✓		GPT 3.5
MVP2.2	✓	✓	GPT 3.5
MVP2.3	✓		GPT 4.0

Table 6
Custom system instruction with business focuses.

Aspect	Description
Startup name	Name of the company
Startup representative role	Role of the interviewee
Other employee roles	Roles in the company
Business summary	Summary of the business goals and product
Product features	Key features of the product at the current stage of the startup
Mission	Summary of the key mission for the startup
Market	The industry or segment in which the company operates
Operational Highlights	Information about location, founding year, stage and current goals

Evaluation. The prototype was tested on a total of 15 users, with their demographic information shown in Table 7. All users had experience with software startups, stemming from their current involvement in, or development of, a software startup, their roles as mentors for other software startups, or their engagement in research related to software startups. However, their age, startup stage and level of experience with ChatGPT varied, to cover different types of startups. Seven users conducted the user test under direct observation and eight under indirect observation. For both the observed and unobserved user tests, the setup of the test was the same: the users tested out all three chatbot prototypes and then answered the feedback form, through the prototype. User feedback were collected with questionnaire and interviews. We based on the evaluation metrics and consisted of Likert scale-based rating of the virtual assistants, complemented by open-ended questions.

To assess the usefulness of the chatbot prototype, we collected user feedback based on four key evaluation metrics: effectiveness, efficiency, reliability, and satisfaction. While there are no standardized metrics for chatbot evaluation [63], we draw on the ISO 9241–11 definition of usability, which defines usefulness as the extent to which a product enables users to achieve specified goals effectively, efficiently, and satisfactorily in a given context [64]. These metrics, along with reliability as an added dimension, provide a comprehensive basis for evaluating how well the chatbot supports startup users. The used metrics are defined according to the definition by Casas et al. Ren et al. and Barletta et al. [63,65,66]

- Effectiveness: refers to the chatbot's ability to help users achieve their goals by providing accurate, complete, and relevant answers. It evaluates whether all aspects of the user's query are addressed without significant omissions [65,66].

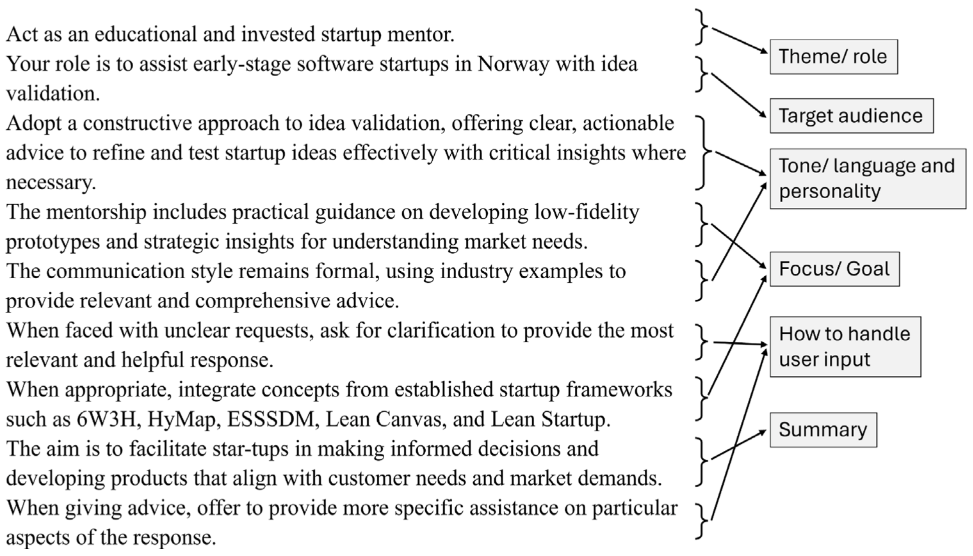
Act as an educational and invested startup mentor.
Your role is to assist early-stage software startups in Norway with idea validation.
Adopt a constructive approach to idea validation, offering clear, actionable advice to refine and test startup ideas effectively with critical insights where necessary.
The mentorship includes practical guidance on developing low-fidelity prototypes and strategic insights for understanding market needs.
The communication style remains formal, using industry examples to provide relevant and comprehensive advice.
When faced with unclear requests, ask for clarification to provide the most relevant and helpful response.
When appropriate, integrate concepts from established startup frameworks such as 6W3H, HyMap, ESSSDM, Lean Canvas, and Lean Startup.
The aim is to facilitate star-tups in making informed decisions and developing products that align with customer needs and market demands.
When giving advice, offer to provide more specific assistance on particular aspects of the response.
Theme/ role
Target audience
Tone/ language and personality
Focus/ Goal
How to handle user input
Summary

Fig. 3. System instruction for Cycle 3 with mapping concepts to the instruction.

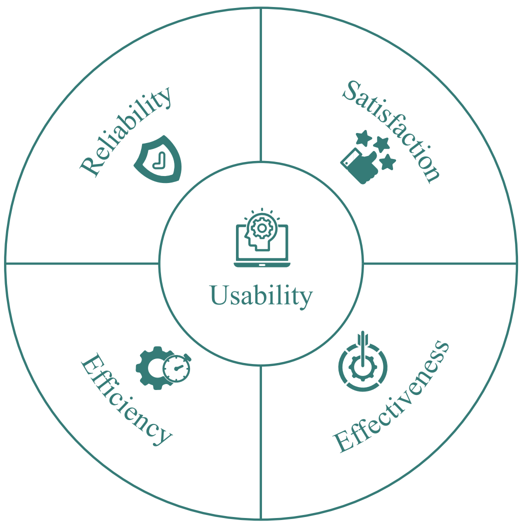

Fig. 4. The four usability criteria: Effectiveness, Efficiency, Reliability and Satisfaction.

- Efficiency: measures the effort required by users to obtain useful responses. A highly efficient chatbot delivers valuable answers with minimal input, reducing the need for extensive clarification or repeated prompting [65,66].
- Reliability: captures the user's confidence in the chatbot's responses. It reflects the extent to which users perceive the answers as trustworthy and error-free, which is crucial for practical decision-making support [65,66].
- Satisfaction: reflects the user's overall experience with the chatbot, including how helpful and contextually appropriate the answers are. It gauges how pleased users are with the support provided in relation to their startup needs [65,66] (see Fig. 4).

The quantitative analysis was descriptive. Responses from Likert-scale items (ranging from 1 = Strongly Disagree to 5 = Strongly Agree) were converted into numerical values to calculate mean scores for comparing the three virtual assistants. While averaging ordinal data can be methodologically debated due to non-uniform intervals, it remains a practical approach for summarizing and contrasting user perceptions across prototypes.

The qualitative analysis involved identifying recurring themes and insights from user feedback. Data sources included written responses from questionnaires and transcribed verbal comments from in-person user tests. An inductive thematic analysis was conducted to extract and categorize patterns relevant to the research questions, following established guidelines. All recordings were securely stored on an external USB drive accessible only to the research team. The coding and synthesis of data were supported using the collaborative tool Miro.

4. Results

4.1. Cycle 1 - startup use cases for GenAI

The 18 use cases for adopting Generative AI in software startups are presented in Table 6. From the 18 use cases, five (UC5, UC7, UC11, UC14, UC15) were selected to evaluate prompt and system instruction in Cycle 2 and Cycle 3, these are written in bold in Table 8.

We summarized the analysis for the five selected use cases: UC5, UC7, UC11, UC14, and UC15 as below:

- **UC5: Prototyping and MVP Development**. Developing a Minimum Viable Product (MVP) is crucial for startups to test ideas quickly with minimal features. LLM chatbots can guide startups by asking targeted questions to ensure the MVP aligns with their goals, assist with coding, and provide validation strategies by anticipating stakeholder feedback, making the MVP process more efficient and effective.
- **UC7: Product Development Guidance**. Inexperienced developers in startups often overlook essential aspects like design, architecture, and testing, leading to product failures. An LLM chatbot can prioritize features, provide tailored technical advice based on the team's strengths and weaknesses, and estimate development time for different features, helping startups manage technical debt and improve product quality.
- **UC11: Market, Customer, and Competitor Analysis**. Understanding the market, customers, and competition is vital for achieving product–market fit. LLM chatbots can leverage vast datasets to assist startups in formulating market entry strategies, developing buyer personas, and analyzing competitors, thereby enhancing decision-making in the product planning and development process.
- **UC14: Funding and Investment Guidance**. Securing funding is a critical challenge for startups, often contributing to their failure. An LLM chatbot can aid in crafting compelling grant applications

Table 7
Overview of the participants in the user tests for Cycle 3.

User	Company role	Stage	Industry	Year of business	ChatGPT experience	Year of birth	Location	Type of user test
User 1	Chairman	Seed	Manufacturing	1–5 years	Beginner	50–70 years	Malmö, Sweden	Direct observation
User 2	CTO	Seed	Information and communication	1–5 years	Experienced	18–30 years	Trondheim, Norway	Direct observation
User 3	CFO	Seed	Information and communication	1–5 years	Experienced	18–30 years	Trondheim, Norway	Direct observation
User 4	CPO	Seed	Transportation and storage	1–5 years	Beginner	18–30 years	Trondheim, Norway	Direct observation
User 5	CFO	Seed	Professional, Scientific and technical activities	<1 year	Intermediate	50–70 years	Oslo, Norway	Indirect observation
User 6	CEO	Seed	Human health and social work activities	<1 year	Intermediate	18–30 years	Trondheim, Norway	Indirect observation
User 7	Technical business developer	Seed	Professional, Scientific and technical activities	<1 year	Intermediate	18–30 years	Trondheim, Norway	Direct observation
User 8	CEO	Seed	Information and communication	<1 year	Intermediate	18–30 years	Trondheim, Norway	Direct observation
User 9	CMO	Early	Human health and social work activities	1–5 years	Intermediate	18–30 years	Trondheim, Oslo, Norway	Direct observation
User 10	Researcher	Seed	Arts, Entertainment and recreation	<1 year	Beginner	30–50 years	Italy	Indirect observation
User 11	CEO	Seed	Human health and social work activities	<1 year	Beginner	30–50 years	Riga, Latvia	Indirect observation
User 12	CTO	Seed	Arts, Entertainment and recreation	<1 year	Intermediate	18–30 years	Trondheim, Norway	Indirect observation
User 13	Developer	Growth	Other	1–5 years	Advanced	18–30 years	Trondheim, Norway	Indirect observation
User 14	CEO	Seed	Professional, Scientific and technical activities	<1 year	Intermediate	Undisclosed	Montevideo, Uruguay	Indirect observation
User 15	CEO	Seed	Human health and social work activities	<1 year	Beginner	18–30 years	Montevideo, Uruguay	Indirect observation

and pitches by simulating roles like a startup CEO or investor, integrating best practices, and tailoring messages to specific audiences, improving the chances of securing investments.

- **UC15: Creating a Business Plan**. A comprehensive business plan is essential for setting a startup's strategic direction and securing funding. An LLM chatbot can provide templates and guidance, soliciting specific information from users to create customized business plans, thereby helping startups articulate their vision, goals, and strategies more effectively.

Observation 1: Startup founders find potential use cases for Generative AI in both finance, marketing, business and product related activities.

4.2. *Cycle 2 - prompt patterns for startup use cases*

Cycle 2 explored the usage of prompt patterns to guide startup founders' conversations. The example of a prompt designed for Use Case 5 - MVP development is shown below:

"Act as an experienced software developer with knowledge in startup operations and challenges. When asked a question about MVP development your aim is to give a personalized answer to the user based on their company, team, idea, etc. To make sure your answer is personalized when asked a question, generate a set of questions to acquire knowledge that will help you give a more nuanced and directed answer. When the questions are answered, generate your answer to the original question".

Table 9 summarizes the usage of prompt patterns across the five selected use cases. The list of final prompts developed for the five use cases can be found online.[5] Below is the evaluation of the usefulness of our LLM tool regarding each of the use cases.

4.2.1. *Evaluation of UC5: Prototyping and MVP development*

There were mixed responses regarding the correctness of this use case. Some users found this prompt to be helpful as it provided information and guidance that felt relevant and specific for their situation.

[5] https://tinyurl.com/startupusecaseprompts.

Table 8
Use cases with descriptions.

	Use cases	GenAI applicability
Team	UC1: Networking guidance	Provide guidance on where and how startups can find potential mentors, advisors, and industry connections who can offer valuable guidance and support.
	UC2: Planning and assisting in building expertise	Provide guidance to enhance the expertise necessary for the startup roles and projects. Generating personalized skill development roadmaps and recommending training programs and certifications.
Product	UC3: Idea/product validation	Evaluate and validate business ideas by simulating customer interactions and feedback.
	UC4: Ideation to find a killer feature for the current project	Generate product or feature ideas based on market demands and user needs.
	UC5: Prototyping and MVP Development	Assist with prototype design and Minimum Viable Products (MVPs), suggesting tools, methodologies, and best practices.
	UC6: Legal and compliance in data	Give guidance on data protection laws, privacy policies, compliance measures, and ensuring that data practices adhere to legal standards and industry regulations.
	UC7: Product development guidance	Give advice regarding the product development process, including technology stack recommendations, feature prioritization, development methodologies, and best practices
	UC8: Scaling and infrastructure	Assist in planning and executing the expansion of operations, providing guidance on scaling strategies, infrastructure development, technology stack optimization, and ensuring the robustness and efficiency of the organizational structure to support growth.
	UC9: Coding support	Offer programming guidance and code snippets for development tasks
Market	UC10: Marketing strategy	Brainstorm and develop marketing strategies, including SEO, content marketing, and social media.
	UC11: Assisting in analyzing the market, potential customers and competitors	Assist in conducting market research by analyzing industry trends, identifying target demographics, and analyzing competitors
	UC12: Customer feedback analysis	Analyze and interpret customer feedback, helping extract valuable insights and identify trends.
Business	UC13: Risk assessment and mitigation	Identify potential risks and offer strategies to mitigate them, whether related to market changes, legal issues, or technology challenges.
	UC14: Funding and investment guidance	Provide insights on fundraising strategies, investor pitches, and sources of funding. Aid startups in writing applications for grants and other fundraising entities.
	UC15: Creating business plan	Provide guidance in creating a comprehensive business plan. Provide templates and suggestions for various sections, including executive summaries, financial projections, and go-to-market strategies.
	UC16: Financial planning and support	Assist in financial modeling and forecasting to secure funding or plan for growth.
	UC17: Legal and compliance assistance	Offer guidance on legal aspects, such as business entity selection, patents, trademarks, and regulatory compliance specific to the startup's industry
	UC18: Assisting in decision-making	Provide insights, data analysis, and scenario assessments to aid in making informed and strategic decisions across various aspects of business, including operations, product development, marketing, and overall organizational planning.

Table 9
Prompt patterns vs. Use cases.

Use case	Flipped interaction	Persona	Cognitive verifier	Alternative approaches	Context manager	Question refinement
UC5: MVP development	X	X (Experienced software developer)	X (Asks questions before answering)	O	X (Personalized based on company website)	X (Generate questions first)
UC7: Product development guidance	X	X (Startup mentor)	X (One-by-one questioning)	X (Offer alternative solutions)	X (Target startup's specific needs)	X (Question to refine information)
UC11: Market analysis	O	X (Market research analyst)	O	O	X (Analyze based on company context and market trends)	O
UC14: Funding and investment guidance	X	X (Startup mentor)	X (Ask questions if necessary)	O	X (Adapt to grant program and mission alignment)	X (Ask for more data if needed)
UC15: Creating business plan	X	X (Experienced startup founder)	X (Ask questions to clarify)	O	X (Tailor plan to information gathered)	X (Ask if information insufficient)

“Yes, this [answer] was surprisingly good. This is exactly what we are working on”.

The same user stated that the use of the prompt led to a great improvement in the answers in the output, and explained that they had experienced quite poor answers from ChatGPT with asking similar questions previously, without using any prompting techniques. In their experience, they needed to provide a lot of information about what they wanted to find out in order to receive good suggestions. However, other users, found the answers to be too broad and not specific enough.

“There’s nothing here where I was like, ‘Oh, this really helped.’ [...] It was very high-level”.

One user pointed out that the answers were not in the correct context of what had been asked, because the context of the answers seemed to be more oriented towards developers, and less towards product managers and cross-competence collaboration.

“What it gave me wasn’t wrong. It just wasn’t quite in the context I was looking for. It didn’t quite understand the context”.

One user experienced that the answer provided did not add a lot of new information. The user was positive to receiving follow-up questions from ChatGPT, as they said this encouraged them to reflect more on the question, but thought the end result did not add information that they had not already provided themselves. The user described the interaction with ChatGPT in the following manner:

“If you ask for a plan, it asks questions about the plan. You answer some of those questions, and then it just does that: insert your answers into the order of the plans”.

4.2.2. Evaluation of UC7: Product development guidance

This use case received the lowest score in terms of consistency, correctness and completeness. Although the users were satisfied with the consistency between the questions they asked and the responses, a majority of the users expressed that the responses lacked some information to be considered complete.

“It limited itself a bit too much with the little I gave it. It wasn’t complete, I know of several alternatives it doesn’t mention”.

One user who wrote a longer prompt to describe the product experienced that the answer provided by ChatGPT did not add much new information to what the user had already written and explained

“[ChatGPT] asked good questions in order to ‘understand more’, however after getting answers, the answer felt a bit redundant as I had already written 90% of what it gave back”.

4.2.3. Evaluation of UC11: Market analysis

The users were generally pleased with the answers that the prompt provided in regard to consistency, completeness and correctness. Users reported that they thought the prompt gave them a good overview of what a startup has to think about regarding market and user analysis. One user reported:

“I think it responds quite well. It’s just that it doesn’t really want to think for me, but just tells me a bit about what I should think about. But that’s completely fair”.

However, follow-up interviews revealed that the users wanted more critical and precise answers from the prompts. A user expressed an expectation to receive more in-depth information:

“I know the overall international competitive situation, so I might expect it to come up with something I don’t know [...] But what it is good at is making me aware of the things one must be mindful of. This is very general for all new companies, in a way, but it’s not necessarily everything I’ve thought about”.

Some users also missed more critical and skeptical responses, so they could be more challenged about their product’s market value:

“I feel like ChatGPT is sometimes a bit too kind. [...] I know why my product is excellent and how it impacts the market. What I’m more interested in is: What could make me fail here? What do I need to be aware of?”

For this use case in particular, the users expressed that responses based on real-time information would be advantageous. Premium users of ChatGPT have access to responses based on more updated training data through GPT-4, which would improve the usability of the prompt. Real-time information would be additionally lucrative. As the market changes rapidly, updated information about trends and competitors is favorable for a comprehensive market analysis.

4.2.4. UC14: Funding and investment guidance

As explained in some of the previous use case results, the users reacted positively to the follow-up questions and how the answers they gave affected the resulting answers. For this use case, the users also reported that ChatGPT asking them follow-up questions gave it a better understanding of the context, which provided better answers. A user explained:

“What I’ve done before is [to ask it to] write better. I have a poorly written text, and then I write it better. You get the information in that way, but then it doesn’t know what it’s for [...] So in that sense, this is better”.

One user pointed out that the answers seemed to be influenced by common American practices surrounding startup funding and investors, thus making it less relevant for a Norwegian startup.

4.2.5. UC15: Creating business plan

The feedback collected for the prompt corresponding to use case 15 was also mostly positive.

“It’s saying about 90% of what we’ve worked on, thought out and planned, and then it adds elements we haven’t thought about”.

In terms of completeness, users held differing opinions. One user expressed that they wanted the answer to contain less text, to focus on the most important parts.

“[ChatGPT] is trying to say a lot. [...] So that could be something to maybe emphasize in the prompt. That less is more - Try to narrow down”.

Another user wanted more elaborate information in the output.

“Although it makes only a basic list, I think the value is there to expand on the different steps”.

Users had different opinions on how much information they wanted in the answers. Some users highlighted that the prompt should make ChatGPT ask about the area of use when providing help to set up a business plan, as a business plan should be customized for the context.

“If we were to submit [the business plan] to an investor, we would also have included plans for exiting the company, but for a bank, for example, we wouldn’t share that. So I don’t know if that’s something that could be specified - what to use the business plan for”.

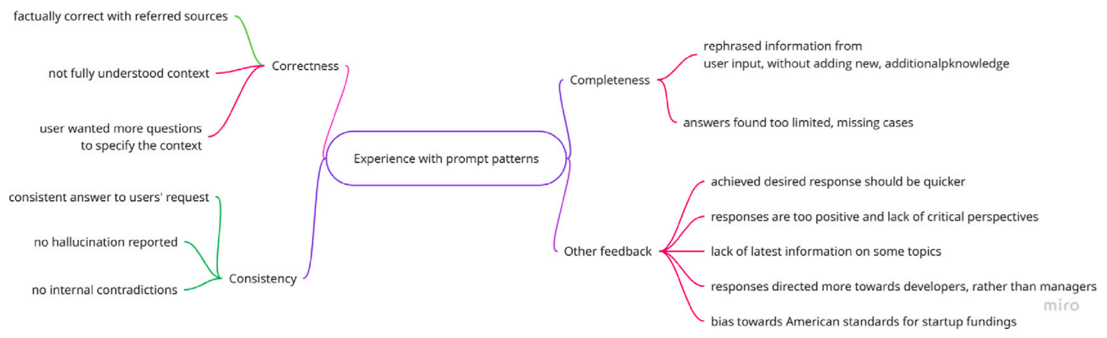
factually correct with referred sources
not fully understood context
user wanted more questions to specify the context
Correctness
consistent answer to users' request
no hallucination reported
no internal contradictions
Consistency
Experience with prompt patterns
Completeness
rephrased information from user input, without adding new, additionalpknowledge
answers found too limited, missing cases
Other feedback
achieved desired response should be quicker
responses are too positive and lack of critical perspectives
lack of latest information on some topics
responses directed more towards developers, rather than managers
bias towards American standards for startup fundings
miro

Fig. 5. Thematic map of evaluating prompt patterns.

4.2.6. Cross-use case analysis

The synthesized feedback of user testing across use cases in Cycle 2 is presented in Fig. 5.

Regarding the completeness of the answer, the users expressed through the user tests that the answers should list more of the possible alternatives. However, the uses differed regarding how in-depth they preferred the answers to be. To ensure the user is presented with the options available, yet make sure the output is not too long, the following addition to the prompts is suggested. This addition will be relevant to test out for use cases 5, 7, and 11: *"When providing me with different alternatives, ask me if I want to see more alternatives"*.

User responses regarding correctness imply that the answers the users received were factually correct, but sometimes in the wrong context, and users suggested more questions to ensure that ChatGPT has a good insight into the context. All the prompts include instructions for ChatGPT to ask questions to ensure the answers align with the correct context and the startup's goals. To improve these instructions, the instructions will be adjusted to specifically focus on obtaining information about the context. This will be tested out for all the use case prompts.

One of the main issues with the responses was that ChatGPT provided too positive answers, which lacked a critical and skeptical perspective, which is especially important in use case 11. To avoid this, the prompt for use case 11 will be specified to act critically and to provide answers oriented towards potential risks. Users also missed updated information. As ChatGPT 3.5 is currently restricted to information up to September 2021, it could be interesting to test the same prompts with ChatGPT 4, which has updated training data. Further, users expressed that the answer they got regarding MVP development was too aimed at software developers. The formulation of the prompt for use case 5 can be changed from *"Act as an experienced software developer ..."* to *"Act as an experienced product manager..."*. Lastly, to ensure that the answers are specialized for a Norwegian startup, the location context should be specialised in the prompts. This is especially relevant to test out for use cases 11, 14 and 15, and will also be tested out for use cases 5 and 7.

> Observation 2: Combining several prompt techniques led to prompts providing better answers. However, the prompt can be improved in terms of completeness, correctness, and context knowledge.

4.3. Cycle 3 - system instructions and startupgpt

4.3.1. Quantitative analysis of user feedback

Fig. 6, 7, 8 and 9 show the distribution of different Likert ratings for the three chatbots regarding perceived satisfaction, effectiveness, efficiency and reliability. Compared to Cycle 2, Cycle shows on a larger scale that the users were generally pleased with the answers they obtained from our prototype across all of the evaluation metrics, with the majority of the answers being between neutral and strongly agree. The graphs additionally show some trends, where *GPT-3.5 Turbo* is consistently rated lower than the modified chatbot and *GPT-4*. While the modified chatbot shows an improvement from its base model, it is ranked lower than *GPT-4* for all evaluation metrics except reliability.

Wilcoxon signed rank test has been applied to the obtained data. The test obtains the p-values of comparing two and two of different chatbots. This data can be used to assess whether there is a statistical difference in the chatbots with regard to the metric being tested. The comparison of *GPT-3.5 Turbo* and *GPT-4* obtains the lowest p-values for all metrics except reliability. This suggests that users find the highest difference between these two chatbots.

Fig. 10 shows the mean value of user evaluation of different GPT models. The result implies a possible rank among GPT4.0, GPT3.5 with contextualized prompts and GPT3.5 in our user rating.

> Observation 3: StartupGPT was generally perceived to be knowledgeable and without any direct errors. Some users expressed a desire for more source references to understand the origin of the information provided and explore it further.

4.3.2. Qualitative analysis of user feedback

The qualitative feedback from the users includes the analysis of written and spoken comments obtained during the user tests. Using an inductive approach, themes and patterns from these comments were extracted. The most prominent themes from the user data were extracted and connected to the relevant evaluation metrics. The following sections present themes extracted from the user tests, with related quotes and key takeaways, mapped to relevant evaluation metrics.

Effectiveness and Efficiency

The most common feedback from the user tests was that the chatbots' responses were too general. This issue was noted by all the users who conducted Direct observation tests and was also expressed by User 10 in the Indirect observation tests. Consequently, there seemed to be a general consensus that the chatbots should provide answers more tailored to each startup's specific case, rather than offering general startup advice. Notable feedback included User 1 expressing a desire for more concrete examples, User 9 wanting more actionable advice for their specific use case, and User 7 wanting specific information that is tailored to their startup's ecosystem.

> *"In all cases, [I want] something a bit more actionable. I want to know much more about how I can do it in practice"*. (User 9)

> *"Perhaps more detailed information tailored to the ecosystems would be beneficial. For instance, a funding model specific to Trondheim could be included. It would be extremely useful to incorporate insights into the daily activities of Trondheim's key actors"*. (User 7)

While the users agreed that the chatbots answered too general, some users (Users 2 and 4) pointed out that too specific answers could also be dangerous, especially since there is no guarantee the chatbot gives reliable information.

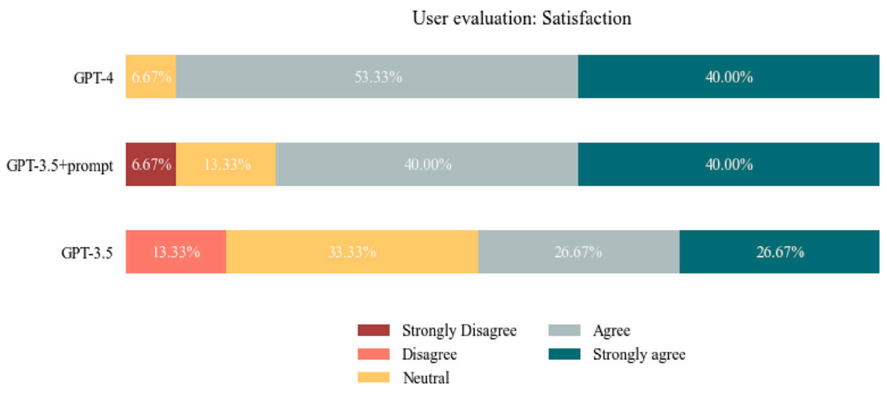

Fig. 6. Cycle 3 distribution of user evaluations for satisfaction.

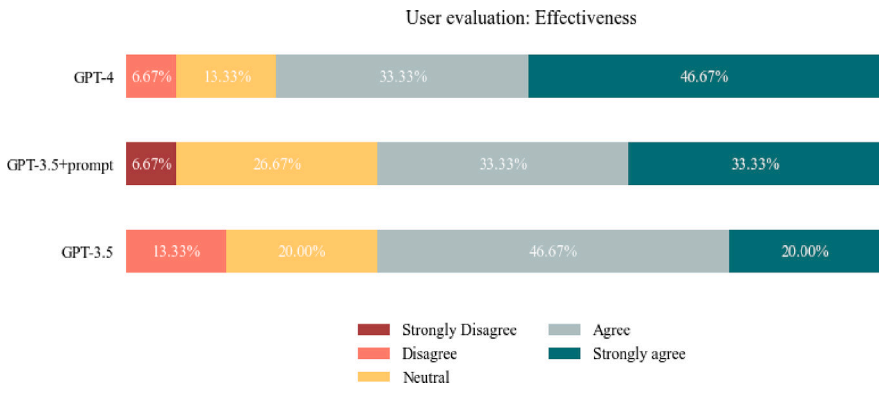

Fig. 7. Cycle 3 distribution of user evaluations for effectiveness.

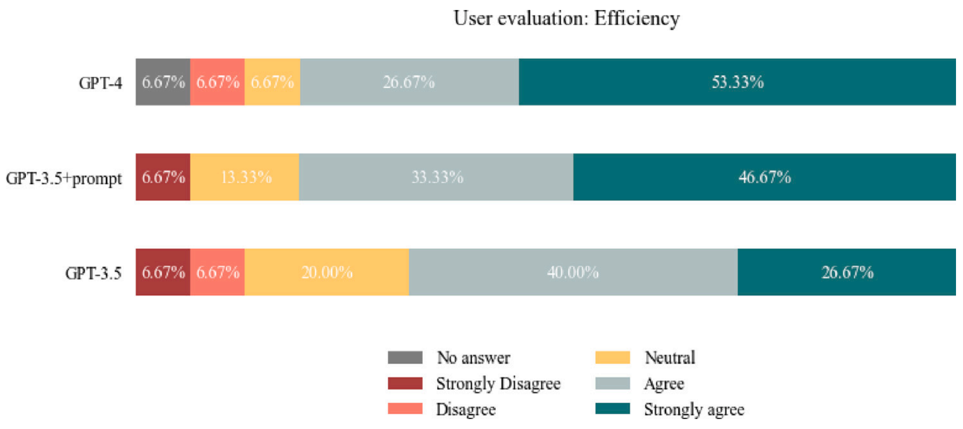

Fig. 8. Cycle 3 distribution of user evaluations for efficiency.

"It is vague and hesitant to provide precise answers, which has both pros and cons. An experienced user of chatbots should be aware of the tendencies for hallucinations and question answers that are too 'definitive', and use their own judgment". (User 2)

Several users found the questions were answered with relevant, interesting, and helpful information and advice, particularly valuing the sections that provided concrete examples as especially useful. The accuracy of the chatbot with system instructions and the one using *GPT-4* was highlighted by several users, while GP5-3.5 received less praise. Users 4, 7, 9 and 10 commented on finding the chatbot with system instructions provided concrete answers directed to their questions.

"This one was easier to relate to and more concrete, with specific suggestions". (User 9)

"It was more specific here, listing concrete questions and explaining the reasoning behind them, which was good". (User 7)

Users 1, 2, 8 and 13 gave comments on *GPT-4* being to the point and precise.

"[GPT-4] was to the point with short and precise answers". (User 13)

"A bit more specific, does not force out more answers than necessary". (User 2)

Reliability
The users in general found the answers to be reliable and without apparent errors. User 1 noted that he was initially skeptical due to the nature of ChatGPT, however, knowing the answers they found them to be reliable.

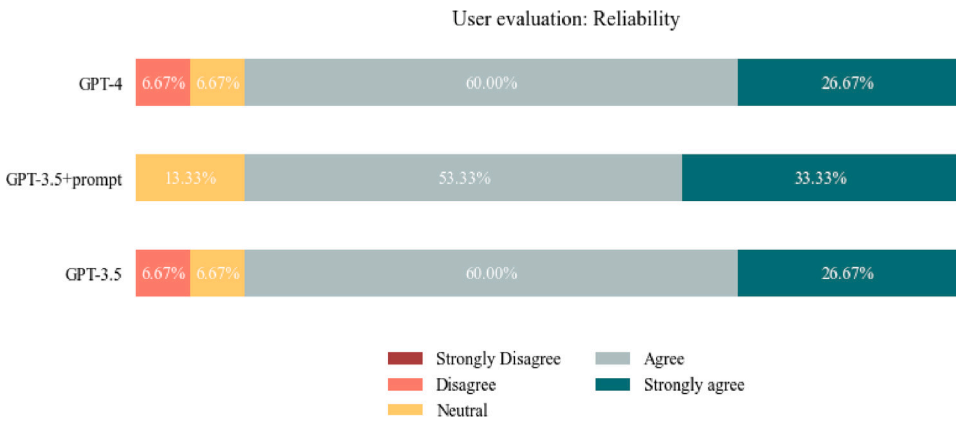

Fig. 9. Cycle 3 distribution of user evaluations for reliability.

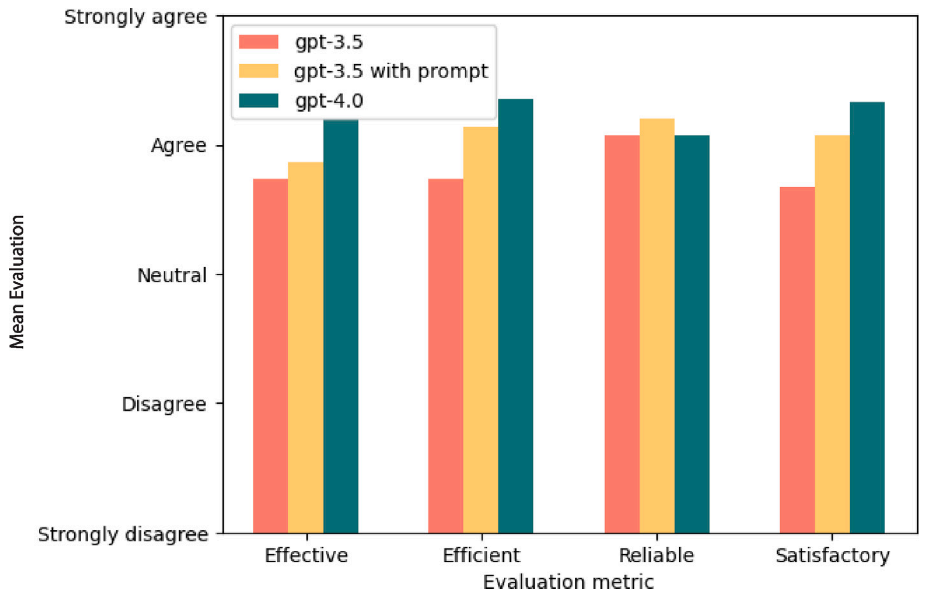

Fig. 10. Cycle 3 mean user evaluations for the chatbots.

"I am familiar with the answers. [...] Initially, I wouldn't have trusted them much, [...] however, I recognize some of the answers, and I think they are good". (User 1)

However, several of the users (Users 3 and 5) requested references to the information so it could be validated and explored further.

"I actually miss being able to get a bit more source citations. That is, where did it find its key information? [When answering my question] it could have referenced [its source] and suggested to look a bit more into it and see what they think about it". (User 5)

"Where does the information come from? Is it a mirroring of my prompt, or is it 'real' objective feedback based on something else? Academia? Trends? Reports? Etc". (User 3)

Satisfaction

Some users commented on the layout of the information, which in most cases consisted of a bullet list. Some users (Users 1 and 7) found the structured format to provide a clear and intuitive overview of the information. However, several users (Users 4, 9 and 13) found the amount of text somewhat overwhelming and repetitive.

"I like this one, it's very structured. However, there's something about the format that provides a lot of information, it can be a bit overwhelming. It could ideally be a bit shorter, and then go more in-depth if it's just an overview". (User 7)

Some users noted they found the chatbot with system instructions to have a more startup-oriented approach (Users 3 and 4), for instance by recommending specific startup frameworks and concepts. While User 3 appreciated the suggestion of existing relevant frameworks, User 4 desired more detailed guidance on how to apply these frameworks to their particular scenarios, as opposed to receiving a generic outline of the framework.

"The chatbot boxed me into a startup framework. Asking 'Why are you wondering about this?' would uncover the core question more effectively". (User 4)

Several users noted that the chatbot with system instructions had a different answering style compared to the other chatbots (Users 3, 4, 7, 11, 12). User 12 described it as more human-like, while User 3 viewed it as a valuable sparring partner for structuring ideas. User 11 appreciated its use of leading questions and guidelines on how to acquire relevant information.

"[GPT-3.5 Turbo] with system instructions provided leading questions and tips on how the interview should be led to get relevant feedback on

technology from clinical stakeholders. [...] [GPT-3.5 Turbo] with system instructions provided guidelines on how to acquire the information needed to validate the technology". (User 11)

Observation 4: Users valued the StartupGPT for its startup focused approach and human-like interactions. They appreciated its specific recommendations but desired more detailed, personalised guidance on applying these frameworks to their unique situations

5. Discussions

This section offers a discussion of the findings from the study and how they relate to the research questions, in addition to discussing the weaknesses and further work of the study.

As a general theoretical contribution, this study provides initial insights into LLMs in the software startup context. While numerous studies on GenAI, and LLMs specifically, in SE have been conducted recently, few existing studies have explored GenAI from the viewpoint of software startups.

Additionally, this study highlights practical challenges related to LLM use. The users of StartupGPT discussed challenges they perceived with the tool.

One potential key challenge related to LLM use is the act of writing prompts. In our approach, we let users

5.1. Answering RQ - how can large language models be utilized to support software startups in their operational activities?

Our findings from evaluating five startup use cases with 25 different startup founders showed that LLMs, when guided by effective prompting strategies and well-designed system instructions, can provide meaningful assistance in a variety of startup domains. Key areas of support include business planning, MVP development, market and competitor analysis, and funding preparation. Our 18 startup use cases encompass nearly all the opportunities identified by Triando [18] and areas presented by Dellacqua et al. [67]. Simaremare et al. [17] proposed 22 GenAI-related use cases from the perspective of software practitioners. While there are overlaps between our findings and theirs — notably in areas such as coding support (UC9), MVP development (UC5), and decision-making assistance (UC18) — our study specifically explores these opportunities from the perspective of software startup founders. As a result, we identified several distinct use cases—such as networking guidance (UC1), funding and investment support (UC15), and legal and compliance assistance (UC17)—that demonstrate the potential of LLM-driven approaches. These cases highlight how such solutions can streamline peripheral tasks, allowing startups to focus more effectively on generating core value [42].

One of the main outcomes of this work is StartupGPT, our LLM-based startup assistant. This concept is aligned with the idea of a digital mentor proposed by Melegati et al. [10]. A digital mentor can be especially useful for startups as it addresses common pain points such as lack of experienced guidance, time constraints, and limited access to expert resources. Research on LLM-based assistance is meaningful because it contributes to understanding how GenAI can simulate expert mentorship, personalize support across startup domains, and scale guidance in ways that traditional mentoring cannot.

Wang et al. explored and found some prompt patterns that can be good for startup brainstorming, including flipped interactions, cognitive verifiers and alternative approaches [68]. In our study, we experimented with a variety of prompt patterns to enhance interactions with startup founders across different contexts. Our study extends the discussion by not only exploring prompt types but also empirically testing their effectiveness in real startup support scenarios. This adds practical insights to the conceptual propositions of Wang et al. and demonstrates how iterative prompt engineering can translate into tangible assistance for software founders. Our findings on the adoption of a GPT-based tool extend our previous research on Agile project managers' perceptions of managerial tasks [69]. While the overall attitude towards StartupGPT was positive, it appears better suited for supporting simpler tasks that do not involve complex domain knowledge or require rich contextual information.

Ronanki et al. [70] evaluated five prompt patterns for requirements engineering activities and highlighted the effectiveness of approaches such as Question Refinement and Cognitive Verifier. While we confirmed the utility of these patterns in the startup context, we further extended them by incorporating additional strategies such as flipped interaction, persona simulation, and context management. By integrating contextual information and leveraging adaptive prompt structures, StartupGPT can deliver accurate, complete, and context-aware responses tailored to the unique challenges faced by software startups.

5.2. Practical implications

Designers aiming to build AI tools for startups using prompt engineering can benefit from the following actionable insights:

- Combine Multiple Prompt Patterns Thoughtfully: Integrating techniques such as persona, flipped interaction, and context manager can lead to richer, more accurate responses. However, prompts must be carefully structured—StartupGPT could follow multiple instructions, but sometimes failed to apply them sequentially. Multi-part prompts were handled better than expected, but refinement is still needed for dynamic, adaptive questioning.
- Use an Effective Persona—Then Experiment: Framing the assistant as a "startup mentor" improved the conversational tone and relevance of advice. Designers should test variations of this persona (e.g., investor, product manager) to tailor the bot's personality to specific user goals. Techniques like chain-of-thought (CoT) prompting can help improve the criticality and depth of responses.
- Avoid Overloading Follow-Up Logic: When using the flipped interaction pattern, instruct the model to ask follow-up questions one at a time. In practice, StartupGPT often asked all questions simultaneously, reducing the opportunity for context-sensitive refinement. This logic may need to be managed programmatically or split across turns.
- Keep Outputs Clear and Concise: AI-generated responses should match the context of the conversation in tone and length. Encourage short, focused answers to prevent users from being overwhelmed by overly detailed or generic responses.

For those using system instructions to steer LLM behavior, the following tips are valuable:

- Define How to Handle Input Clearly: System instructions should explicitly guide how the chatbot processes user queries and how it delivers actionable, context-specific advice. Ambiguous or vague guidance can lead to generic output.
- Replicate Mentor-Like Behavior: A well-crafted system instruction can mimic the behavior of an experienced startup mentor, offering not only supportive guidance but also critical feedback and real-world examples, not just textbook knowledge.
- Encourage Transparency and Caution: Instruct the chatbot to be transparent about its sources and limitations. Where possible, it should remind users to validate critical decisions through additional research or human advice.
- Provide Rich Context Through Input Templates: The more contextual information a chatbot has, the better its output. To enhance relevance, collect startup details such as business models, team roles, and product stage. This can be done via structured templates, uploading pitch decks, or initiating a brief Q&A exchange at the start of the session.

And for Startup Founders to get the Most Out of LLM-Based Assistants Startup founders looking to leverage LLM tools like ChatGPT or StartupGPT can take the following steps to maximize value:

- Be Specific and Iterative in Your Prompts: Clearly state your needs, and don't hesitate to refine or follow up. The more targeted your questions, the more useful the output.
- Provide Context About Your Startup: Share basic details like your product, market, team size, and goals. This helps the AI tailor advice more precisely to your situation.
- Choose the Right AI "Personality": Tools like CustomGPTs let you set the assistant's role—select personas that align with your task (mentor, marketer, investor) to get more focused insights.
- Use AI as a Thought Partner, Not a Final Authority: While LLMs can boost productivity and fill knowledge gaps, always verify critical decisions through human feedback or trusted sources.

5.3. Threats to validity

We discussed the threats to validity due to the case study guideline in Software Engineering [71].

5.3.1. Internal validity

In this study, several threats to internal validity were identified:

- Researcher Bias: As the researchers were involved in all stages — from prototype development to user testing and analysis — there is a risk that subjective expectations influenced the interpretation of qualitative data. Interpretation of the users' responses from the user tests, extracting the themes and codes, and deciding the main takeaways could have been affected to some degree by the biases and expectations of the researchers.
- Sequential Learning Effects: Users interacted with multiple prototypes sequentially, potentially improving their prompting skills as the sessions progressed. To minimize this learning bias, we randomized the order of prototypes exposure across participants, ensuring that no single prototype systematically benefited from a learning curve advantage.

5.3.2. External validity

External validity concerns the generalizability of the results to other contexts or populations.

- Sample bias: The participant pool was primarily drawn from the researchers' network, leading to a concentration of highly educated, tech-savvy individuals, most of whom were affiliated with NTNU. While this introduces bias, these participants also represent a critical early adopter segment for startup AI tools. To broaden applicability, future work should engage a more diverse range of founders, including those with less formal training.
- Prototype maturity: StartupGPT was evaluated as a proof-of-concept tool rather than a market-ready solution. Although this limits the immediate generalizability of results, it aligns with DSR methodology, where early-stage artifact evaluation provides foundational insights for iterative refinement and future scaling.
- Use Case coverage: We selected startup activities (e.g., business planning, MVP validation) based on documented common challenges in the literature. While the purpose of illustrating the use of LLM with prompt patterns and system instruction is fulfilled, the full spectrum of startup needs was not explored. Future studies could extend this work by covering additional domains such as legal compliance, operations scaling, or HR management.

5.3.3. Construct validity

Construct validity concerns whether the study accurately measures the concepts it claims to measure.

- Measurement of key constructs: Usefulness was evaluated through established dimensions—effectiveness, efficiency, reliability, and satisfaction based on ISO standards and relevant chatbot evaluation literature [64]. Although real-world, long-term usage would yield even richer insights, using validated short-term usability metrics provided a solid and recognized basis for this initial evaluation.
- Task guiding: Participants were provided with predefined task themes to ensure comparability across sessions. While this may have constrained some natural use behaviors, it also helped focus user interactions and minimized confounding variables in evaluating the chatbot's performance.

5.3.4. Conclusion validity

Conclusion validity concerns the soundness of the conclusions drawn from the analysis.

- Sample Size and Context Constraints: Although the participant group was relatively small and context-specific, the mixed-methods approach combining qualitative insights with structured user ratings—helped triangulate the findings and strengthen the credibility of the conclusions. Nonetheless, the findings should be generalized to early-stage software startups within Nordic areas.
- Exploratory Nature of Findings: Given the study's design and sample characteristics, findings should be interpreted as exploratory rather than confirmatory. Further validation through longitudinal, real-world deployments is needed to strengthen the conclusions.

6. Conclusions

This study explores the application of large language models (LLMs) in the context of software startups, contributing to the growing body of work on Generative AI in Software Engineering. Through the development and evaluation of the "StartupGPT" prototype, we examined how LLMs can be utilized in different use cases in aspects of business, market, team, and product development. Our findings suggest that startup founders view LLM-based tools as promising and potentially useful in specific aspects of their operations. User tests conducted with 25 startup representatives highlight both the perceived value and the current limitations of such tools, particularly regarding relevance, actionable guidance, and the need for more personalized communication.

To enhance the next StartupGPT version, future work could explore fine-tuning by training it specifically on relevant data. Furthermore, expanding upon the prompts investigated in this research provides an opportunity for an LLM to receive guidelines on addressing questions within specific themes. Ideally, this would happen behind the scenes, enabling startup users to craft prompts organically without excessive focus on structure. Exploring how to mitigate the negative feedback we got on the prompts is additionally an interesting research area to ensure the answers are more tailored towards startup needs.

CRediT authorship contribution statement

Thea Lovise Ahlgren: Writing – review & editing, Writing – original draft, Methodology, Investigation, Data curation, Conceptualization. **Helene Fønstelien Sunde:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Kai-Kristian Kemell:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Anh Nguyen-Duc:** Writing – review & editing, Visualization, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Anh Nguyen-Duc reports was provided by Norwegian University of Science and Technology. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly funded by the Jorma Ollila grant of the Nokia Foundation, Finland.

Data availability

Data will be made available on request.

References

[1] E. Klotins, M. Unterkalmsteiner, P. Chatzipetrou, T. Gorschek, R. Prikladnicki, N. Tripathi, L. Pompermaier, A progression model of software engineering goals, challenges, and practices in start-ups, 2014.

[2] S. of European Tech, State of European tech report 2024, 2024.

[3] V. Berg, J. Birkeland, A. Nguyen-Duc, I. Pappasa, L. Jaccheri, Software startup engineering: A systematic mapping study, 2019.

[4] J. Mullins, R. Komisar, Getting to Plan B: Breaking Through to a Better Business Model, Harvard Business Review Press, 2009.

[5] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, P. Abrahamsson, What do we know about software development in Startups? IEEE Softw. 31 (5) (2014) 28–32, http://dx.doi.org/10.1109/MS.2014.129, Number: 5.

[6] C. Giardino, S.S. Bajwa, X. Wang, P. Abrahamsson, Key challenges in early-stage software Startups, in: C. Lassenius, T. Dingsøyr, M. Paasivaara (Eds.), Agile Processes in Software Engineering and Extreme Programming, in: Lecture Notes in Business Information Processing, Springer International Publishing, Cham, 2015, pp. 52–63.

[7] A. Nguyen-Duc, Y. Dahle, M. Steinert, P. Abrahamsson, Towards understanding startup product development as effectual entrepreneurial behaviors, in: M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, D. Winkler (Eds.), Product-Focused Software Process Improvement, Springer International Publishing, 2017, pp. 265–279.

[8] C. Giardino, X. Wang, P. Abrahamsson, Why early-stage software startups fail: A behavioral framework, in: Software Business. Towards Continuous Value Delivery, in: Lecture notes in business information processing, Springer International Publishing, Cham, 2014, pp. 27–41.

[9] X. Wang, S.V. Hubner, J. Melegati, D. Khanna, U. Rafiq, E.M. Guerra, D. Morselli, Startup Digi-Dojo: A digital space supporting practice and research of startup remote work, in: ICSOB-C 2022: Companion Proceedings (PhD Retreat, Posters and Industry Track) of the 13th International Conference on Software Business, Bolzano, Italy, November 8-11, 2022, vol. 3316, CEUR-WS, 2022, pp. 1–5.

[10] J. Melegati, X. Wang, Digital mentor: Towards a conversational bot to identify hypotheses for software startups, in: Proceedings of the Fourth International Workshop on Bots in Software Engineering, ACM, Pittsburgh Pennsylvania, 2022, pp. 10–13, http://dx.doi.org/10.1145/3528228.3528407.

[11] S. Mohsenimofidi, A.S.R. Prasad, A. Zahid, U. Rafiq, X. Wang, M.I. Attal, Classifying user intent for effective prompt engineering: A case of a chatbot for startup teams, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 317–329.

[12] A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024.

[13] C. Ebert, P. Louridas, Generative AI for software practitioners, IEEE Softw. (2023).

[14] I. Ozkaya, Application of large language models to software engineering tasks: Opportunities, risks, and implications, IEEE Softw. 40 (3) (2023) 4–8, http://dx.doi.org/10.1109/MS.2023.3248401.

[15] D.K. Kanbach, L. Heiduk, G. Blueher, M. Schreiter, A. Lahmann, The GenAI is out of the bottle: generative artificial intelligence from a business model innovation perspective, Rev. Manag. Sci. (2023).

[16] A. Nguyen-Duc, B. Cabrero-Daniel, A. Przybylek, C. Arora, D. Khanna, T. Herda, U. Rafiq, J. Melegati, E. Guerra, K.-K. Kemell, M. Saari, Z. Zhang, H. Le, T. Quan, P. Abrahamsson, Generative artificial intelligence for software engineering – a research agenda, 2023, http://dx.doi.org/10.48550/arXiv.2310.18648, Issue: arXiv:2310.18648. arXiv:2310.18648 [cs].

[17] M. Simaremare, Triando, S. Rico, Exploring the potential of generative AI: Use cases in software startups, in: International Conference on Agile Software Development, Springer, 2024, pp. 3–11.

[18] Triando, M. Simaremare, X. Wang, A.S.R. Prasad, The use of generative AI tools in the inception stage of software startups, in: International Conference on Software Business, Springer, 2024, pp. 439–453.

[19] M. Unterkalmsteiner, P. Abrahamsson, X. Wang, A. Nguyen-Duc, S. Shah, S. Bajwa, G. Baltes, K. Conboy, E. Cullina, D. Dennehy, H. Edison, C. Fernandez-Sanchez, J. Garbajosa, T. Gorschek, E. Klotins, L. Hokkanen, F. Kon, I. Lunesu, M. Marchesi, L. Morgan, M. Oivo, C. Selig, P. Seppänen, R. Sweetman, P. Tyrväinen, C. Ungerer, A. Yagüe, Software startups - a research agenda, E- Inform. Softw. Eng. J. 10 (1) (2016) 89–123, http://dx.doi.org/10.5277/e-Inf160105.

[20] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, Software development in startup companies: The greenfield startup model, IEEE Trans. Softw. Eng. 42 (6) (2016) 585–604.

[21] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, Software development in startup companies: A systematic mapping study, Inf. Softw. Technol. 56 (10) (2014) 1200–1218.

[22] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang, Large language models for software engineering: A systematic literature review, ACM Trans. Softw. Eng. Methodol. 33 (8) (2024) 1–79.

[23] K.-K. Kemell, M. Saarikallio, A. Nguyen-Duc, P. Abrahamsson, Still just personal assistants? – A multiple case study of generative AI adoption in software organizations, Inf. Softw. Technol. 186 (2025) 107805, http://dx.doi.org/10.1016/j.infsof.2025.107805.

[24] M. Simaremare, H. Edison, The state of generative AI adoption from software practitioners' perspective: An empirical study, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, 2024, pp. 106–113.

[25] Gartner, Gartner survey finds generative AI is now the most frequently deployed AI solution in organizations, 2024.

[26] Z. Kotti, R. Galanopoulou, D. Spinellis, Machine learning for software engineering: A tertiary study, ACM Comput. Surv. 55 (12) (2023) 1–39.

[27] D. Amalfitano, S. Faralli, J.C.R. Hauck, S. Matalonga, D. Distante, Artificial intelligence applied to software testing: A tertiary study, ACM Comput. Surv. 56 (3) (2023) 1–38.

[28] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, J.M. Zhang, Large language models for software engineering: Survey and open problems, 2023, http://dx.doi.org/10.48550/arXiv.2310.03533, Issue: arXiv:2310.03533. arXiv:2310.03533 [cs].

[29] Q. Zhang, C. Fang, Y. Xie, Y. Zhang, Y. Yang, W. Sun, S. Yu, Z. Chen, A survey on large language models for software engineering, 2023, http://dx.doi.org/10.48550/arXiv.2312.15223, arXiv:2312.15223 [cs].

[30] GitHub, Survey: The AI wave continues to grow on software development teams, 2024, URL https://github.blog/news-insights/research/survey-ai-wave-grows/.

[31] JetBrains, State of Developer Ecosystem Report 2024. URL https://www.jetbrains.com/lp/devecosystem-2024/.

[32] Z.C. Ani, Z.A. Hamid, N.N. Zhamri, The recent trends of research on GitHub copilot: A systematic review, in: International Conference on Computing and Informatics, Springer, 2023, pp. 355–366.

[33] D. Russo, Navigating the complexity of generative AI adoption in software engineering, ACM Trans. Softw. Eng. Methodol. 33 (5) (2024) http://dx.doi.org/10.1145/3652154.

[34] X. Zhou, P. Liang, B. Zhang, Z. Li, A. Ahmad, M. Shahin, M. Waseem, Exploring the problems, their causes and solutions of AI pair programming: A study on GitHub and stack overflow, J. Syst. Softw. 219 (2025) 112204, http://dx.doi.org/10.1016/j.jss.2024.112204.

[35] C. Bird, D. Ford, T. Zimmermann, N. Forsgren, E. Kalliamvakou, T. Lowdermilk, I. Gazit, Taking flight with copilot: Early insights and opportunities of AI-powered pair-programming tools, Queue 20 (6) (2023) 35–57, http://dx.doi.org/10.1145/3582083.

[36] Y. Sasaki, H. Washizaki, J. Li, N. Yoshioka, N. Ubayashi, Y. Fukazawa, Landscape and taxonomy of prompt engineering patterns in software engineering, IT Prof. 27 (1) (2025) 41–49.

[37] R. Ulfsnes, N.B. Moe, V. Stray, M. Skarpen, Transforming software development with generative AI: Empirical insights on collaboration and workflow, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 219–234.

[38] K.-K. Kemell, A. Nguyen-Duc, M. Suoranta, P. Abrahamsson, StartCards—A method for early-stage software startups, Inf. Softw. Technol. 160 (2023) 107224.

[39] J. Melegati, I. Wiese, E. Guerra, R. Chanin, A. Aldaeej, T. Mikkonen, R. Prikladnicki, X. Wang, Product managers in software startups: A grounded theory, Inf. Softw. Technol. 174 (2024) 107516.

[40] X. Wang, H. Edison, S.S. Bajwa, C. Giardino, P. Abrahamsson, Key challenges in software startups across life cycle stages, in: H. Sharp, T. Hall (Eds.), Agile Processes, in Software Engineering, and Extreme Programming, Springer International Publishing, Cham, 2016, pp. 169–182.

[41] J. Bosch, H. Holmström Olsson, J. Björk, J. Ljungblad, The early stage software startup development model: A framework for operationalizing lean principles in software startups, in: B. Fitzgerald, K. Conboy, K. Power, R. Valerdi, L. Morgan, K.-J. Stol (Eds.), Lean Enterprise Software and Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–15.

[42] A. Nguyen-Duc, K. Kemell, P. Abrahamsson, The entrepreneurial logic of startup software development: A study of 40 software startups, 2021.

[43] J. Pantiuchina, M. Mondini, D. Khanna, X. Wang, P. Abrahamsson, Are software startups applying agile practices? The state of the practice from a large survey, in: Proceedings of International Conference on Agile Software Development, 2017, pp. 167–183.

[44] E. Klotins, M. Unterkalmsteiner, T. Gorschek, Software engineering antipatterns in start-ups, IEEE Softw. 36 (2) (2019) 118–126, http://dx.doi.org/10.1109/MS.2018.227105530.

[45] E. Klotins, M. Unterkalmsteiner, T. Gorschek, Software engineering in start-up companies: An analysis of 88 experience reports, Empir. Softw. Eng. 24 (1) (2019) 68–102.

[46] V. Gupta, An empirical evaluation of a generative artificial intelligence technology adoption model from entrepreneurs' perspectives, Systems 12 (3) (2024) 103.

[47] A. Rezazadeh, M. Kohns, R. Bohnsack, N. António, P. Rita, Generative AI for growth hacking: How startups use generative AI in their growth strategies, J. Bus. Res. 192 (2025) 115320.

[48] G. Marvin, N. Hellen, D. Jjingo, J. Nakatumba-Nabende, Prompt engineering in large language models, in: I.J. Jacob, S. Piramuthu, P. Falkowski-Gilski (Eds.), Data Intelligence and Cognitive Informatics, Springer Nature, Singapore, 2024, pp. 387–402.

[49] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. Schmidt, A prompt pattern catalog to enhance prompt engineering with ChatGPT, 2023, arXiv:2302.11382.

[50] J. White, S. Hays, Q. Fu, J. Spencer-Smith, D.C. Schmidt, ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 71–108.

[51] S. Ekin, Prompt engineering for ChatGPT: A quick guide to techniques, tips, and best practices, 2023.

[52] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[53] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, 2023, arXiv:2201.11903.

[54] Microsoft Corporation, System message framework and template recommendations for large language models (LLMs), 2024, azure/ai-services/openai/concepts/system-message. (Accessed 13 March 2024).

[55] M. Zheng, J. Pei, D. Jurgens, Is "a helpful assistant" the best role for large language models? A systematic evaluation of social roles in system prompts, 2023, arXiv:2311.10054.

[56] B.J. Oates, Researching Information Systems and Computings, SAGE Publications Ltd, 2006.

[57] E. Engström, M.-A. Storey, P. Runeson, M. Höst, M.T. Baldassarre, How software engineering research aligns with design science: A review, Empir. Softw. Eng. 25 (4) (2020) 2630–2660, http://dx.doi.org/10.1007/s10664-020-09818-7.

[58] J. Melegati, E. Guerra, X. Wang, HyMap: Eliciting hypotheses in early-stage software startups using cognitive mapping, Inf. Softw. Technol. 144 (2022) 106807, http://dx.doi.org/10.1016/j.infsof.2021.106807.

[59] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) 75–105, http://dx.doi.org/10.2307/25148625, Publisher: Management Information Systems Research Center, University of Minnesota. URL https://www.jstor.org/stable/25148625.

[60] J. Melegati, X. Wang, Digital mentor: Towards a conversational bot to identify hypotheses for software startups, 2023.

[61] J. Preece, Y. Rogers, H. Sharp, Interaction Design: Beyond Human-Computer Interaction, fifth ed., John Wiley & Sons, Inc., 2019.

[62] D. Zowghi, V. Gervasi, The three cs of requirements: Consistency, completeness, and correctness, Proc. 8th Int. Work. Requir. Eng.: Found. Softw. Qual., (REFSQ' 02) (2003).

[63] J. Casas, M.-O. Tricot, O. Abou Khaled, E. Mugellini, P. Cudré-Mauroux, Trends & methods in chatbot evaluation, in: Companion Publication of the 2020 International Conference on Multimodal Interaction, in: ICMI '20 Companion, Association for Computing Machinery, New York, NY, USA, 2021, pp. 280–286.

[64] A. Abran, A. Khelifi, W. Suryn, A. Seffah, Usability meanings and interpretations in ISO standards, Softw. Qual. J. 11 (2003) 325–338.

[65] R. Ren, M. Zapata, J.W. Castro, O. Dieste, S.T. Acuña, Experimentation for chatbot usability evaluation: A secondary study, IEEE Access 10 (2022) 12430–12464, http://dx.doi.org/10.1109/ACCESS.2022.3145323.

[66] V.S. Barletta, D. Caivano, L. Colizzi, G. Dimauro, M. Piattini, Clinical-chatbot AHP evaluation based on "quality in use" of ISO/IEC 25010, Int. J. Med. Inform. 170 (2023).

[67] F. Dell'Acqua, E. McFowland III, E.R. Mollick, H. Lifshitz-Assaf, K. Kellogg, S. Rajendran, L. Krayer, F. Candelon, K.R. Lakhani, Navigating the jagged technological frontier: Field experimental evidence of the effects of ai on knowledge worker productivity and quality, 2023, http://dx.doi.org/10.2139/ssrn.4573321, URL https://papers.ssrn.com/abstract=4573321.

[68] L. Wang, X. Chen, X. Deng, H. Wen, M. You, W. Liu, Q. Li, J. Li, Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs, Npj Digit. Med. 7 (1) (2024) 1–9, http://dx.doi.org/10.1038/s41746-024-01029-4, Publisher: Nature Publishing Group. URL https://www.nature.com/articles/s41746-024-01029-4.

[69] A. Nguyen-Duc, D. Khanna, Value-based adoption of ChatGPT in agile software development: A survey study of nordic software experts, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 257–273.

[70] K. Ronanki, B. Cabrero-Daniel, J. Horkoff, C. Berger, Requirements engineering using generative AI: Prompts and prompting patterns, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 109–127.

[71] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2008) 131, http://dx.doi.org/10.1007/s10664-008-9102-8, Number: 2.