Original Paper

Using Natural Language Prompts With AI Models for Low-Cost Assistive Software Design: Exploratory Comparative Evaluation

Francesc Antoni Bañuls-Lapuerta[1*], PhD; Vicent Marti-Miralles[1,2*], MSc; Rómulo Jacobo Gónzalez-García[1*], PhD; Gabriel Martínez-Rico[1*], Prof Dr

[1]Campus Capacitas, Valencia Catholic University Saint Vincent Martyr, Burjassot, Spain
[2]Doctorate School, Valencia Catholic University Saint Vincent Martyr, Valencia, Spain
*all authors contributed equally

Corresponding Author:

Vicent Marti-Miralles, MSc
Campus Capacitas
Valencia Catholic University Saint Vincent Martyr
Carrer de Joaquin Navarro, 37
Burjassot 46010
Spain
Phone: 96 363 74 12
Email: vimarmi@mail.ucv.es

Abstract

Background: This study investigates the capacity of 7 artificial intelligence (AI) models, 5 free and 2 paid, to generate functional software for designing low-cost, personalized assistive products.

Objective: The objective was to determine which models are most effective, accessible, and consistent in supporting nontechnical professionals in developing inclusive digital solutions and to assess the capabilities of commercially available and easy-to-access AI models to generate code from natural language interactions in the shape of a nontechnical assistive technology design process.

Methods: Each AI model was prompted using natural language, without any technical input, to create a Python program that converts an arcade gamepad into an adapted mouse-like controller. Sixteen progressively complex functions were requested through standardized prompts, delivered without additional feedback or correction. Model performance was evaluated based on the number of successfully implemented functions and the average number of prompts required.

Results: Paid models demonstrated markedly superior performance. Gemini Pro (Google) successfully implemented 14 of 16 requested functions with an average of 1.25 (SD 0.45) prompts, while ChatGPT Plus (GPT-5) achieved 11 functions with an average of 1.31 (SD 0.48) prompts. In contrast, free models produced between 0 and 4 functional outcomes, with DeepSeek and Gemini Free ranking the highest within their category. The enhanced outcomes of paid models were linked to improved contextual understanding, greater tolerance for natural language, and reduced conversational drift.

Conclusions: Paid AI models, particularly Gemini Pro and ChatGPT Plus, exhibit strong potential as tools for bridging the gap between health or education professionals and software development. They enable the creation of affordable, user-centered assistive technology without requiring advanced programming skills. Nevertheless, human oversight and foundational literacy in prompt design remain crucial to guarantee functionality, reliability, and ethical use.

JMIR Rehabil Assist Technol 2026;13:e86786; doi: 10.2196/86786

Keywords: artificial intelligence; assistive products; digital accessibility; Gemini; ChatGPT; inclusive software

Introduction

Background

Artificial intelligence (AI) can be defined as the ability of nonhuman systems, machines, or software to simulate functions inherent to human intelligence, such as perceiving, reasoning, learning, planning, and anticipating future situations [1]. The recent advancements in the field have made clear that generative AI will open new scenarios in which human-machine collaboration will enable faster and more adaptive software solutions to meet the needs of people with disabilities [2]. In these contexts, the professional programmer assumes a supervisory role, while users without technical experience can create functional prototypes through intuitive AI-powered interfaces. It is thought that, by 2030, integrated assistants, such as the so-called HyperAssistant, will be capable of accompanying individuals throughout all stages of development, bridging the gap between software conception and implementation [3].

AI is transforming the way software is conceived and produced by allowing nontechnical users to participate actively in programming. Traditionally, this process was restricted to those mastering specific languages and methodologies; however, generative models in the shape of conversational AI systems have reduced the gap between natural language and coding, fostering digital inclusion and democratizing access to digital creation [4,5]. Generative assistants increase productivity and allow users with limited skills to build functional prototypes by using natural language–oriented programming, where users express requirements in everyday language and AI translates them into executable code, bridging human thought and computational structures. This AI integration in development environments additionally enhances efficiency through real-time suggestions and code autocompletion, while warning against technological overreliance [6-8].

Design and Characteristics of Effective Prompts for Interaction With AI

The dialogue between a user and an AI system depends on how instructions or prompts, as they are known in the AI environment, are formulated. Prompts act as mediators between human intention and automatic language generation, shaping the trajectory of interaction, structuring intent, and bridging human goals with machine logic [9]. The first key element to achieving functional prompts is goal definition. Effective communication with conversational systems requires explicit purposes that limit and contain the semantic context. A prompt without a defined end, such as “tell me about economics,” produces generic responses, whereas “summarize in 200 words the effects of remote work on business productivity” provides focus and precision [10].

The second principle is linguistic specificity, referring to the fact that grammatically well-structured prompts enhance coherence. Since AI operates through probabilistic predictions, lexical clarity, thematic delimitation, and details about format or audience reduce ambiguity [11-13]. The internal structure should ideally include 5 essential elements: topic, style, tone, format, and context. It should also contain action verbs (such as analyze, compare, or synthesize) to guide specific cognitive-like operations [14,15]. This level of precision is fundamental, as prompt design directly influences the reasoning processes activated by the model [16].

Using these key principles and unifying different prompt explanations from the literature, we could divide prompts into 6 specific and differentiated types with their own goals (Table 1).

Table 1. Types of prompts used.

Prompt type	Goal	Example
Zero-shot or decision	Execute a task or produce a direct response without prior examples [17].	Explain the concept of neuroplasticity in simple terms.
Few-shot or iterative	Guide generation using prior examples [12,18].	Example 1: write a scientific abstract. Example 2: create a new one about artificial intelligence.
Chain of thought	Promote logical processes or explanations step-by-step [19].	Explain, step by step, how you arrive at the conclusion about the benefits of automation.
Sequential	Develop an idea or process step by step [11].	First, define the concept, then provide an example, and finally conclude.
Argumentative	Request a reasoned, well-supported position [11].	Argue why automation can be beneficial.
Verification	Review one’s own output and detect errors or biases [20].	Review your previous answer and indicate possible inaccuracies.

Communicative Limitations of Nonexpert Users in Interactions With AI

Several studies agree that interactions between users and AI systems present communicative limitations stemming from the spontaneous and imprecise use of natural language. Users without technical training often formulate ambiguous prompts, making it difficult for the system to interpret communicative intent [21]. Additionally, the absence of nonverbal cues and the limited adaptability of artificial discourse exacerbate these differences, revealing that the effectiveness of interaction depends on the precision and structure of the language used [22].

From a qualitative perspective, users tend to treat chatbots as human interlocutors, reproducing communicative patterns based on empathy and reciprocity [23]. Users tend to apply Grice conversational maxims when evaluating virtual assistants. These maxims include quantity (providing the

right amount of information), quality (saying only what is true and verifiable), manner (expressing oneself clearly and orderly, avoiding ambiguity), and relevance (keeping responses focused on the topic at hand) [24]. Complementarily, anthropomorphism, or trying to treat AI like a human, seems to induce a sort of relational dissonance that clearly shows a contradiction as users understand AI as nonhuman yet interact with it as if it were a human [25,26].

Generative models do not interpret implicit intentions but rather textual correlations; therefore, effectiveness depends on the specificity of the prompt [12,14]. Many users are unaware of this mechanism and rely on trial and error [27,28]. Furthermore, the absence of mental models about AI processing creates a cognitive gap between intention and response, reinforced by interfaces that simulate naturalness while not processing information cognitively naturally [29,30].

In summary, most users lack the metalinguistic skills required to design effective prompts, and bridging this gap demands a form of communicative literacy oriented toward AI.

AI in Education: Barriers, Assistive Products, and Opportunities for Digital Inclusion

At this point, it is essential to analyze how AI can be integrated into educational and disability support contexts, expanding opportunities for student participation and learning. This link between technology and inclusion requires examining the barriers that still limit its effective use in classrooms. The analysis of possibilities for students with disabilities within the school setting must address the obstacles that hinder their access and active participation, emphasizing the importance of identifying factors that prevent the functional use of educational materials [31]. Students with disabilities face multiple barriers—including negative attitudes, lack of appropriate resources, and insufficient technological accessibility—which restrict their access to inclusive and equitable education [32].

Since the COVID-19 pandemic, information and communication technology has played a central role both in education and society [33]. However, the lack of technological accessibility remains a major barrier, as current learning materials heavily depend on digital tools. This not only affects curricular access but also the acquisition of digital skills essential for adulthood. Studies and hiring companies, such as iHire, highlight that 76% of people seek employment online, illustrating the importance of digital literacy for future labor inclusion [34,35].

To address these barriers, assistive products (APs) are key tools for ensuring the educational and social participation of students with disabilities. According to ISO 9999:2022, APs are products that optimize functioning and reduce disability, serving as intermediaries between personal abilities and environmental demands. The previous version of this standard (ISO 9999:2016) specified 3 key functions: facilitating participation, supporting or substituting body functions, and preventing limitations or restrictions in participation. APs associated with information and communication technology include both hardware and software, and the ISO 9999:2023 standard emphasizes their mixed nature, explicitly stating "including software" and recognizing their educational value within category e130 of the International Classification of Functioning [36-38].

Society and public administrations are responsible for ensuring that children with disabilities have access to the APs necessary for their personal and social development [39-41]. However, the World Health Organization warns that high costs and technical complexity limit access, particularly in educational contexts [42]. Although technological advances and globalization have partially reduced prices, technical barriers remain in specific contexts [43].

While many education and health professionals are familiar with physical APs, they often lack expertise in software-based solutions, which usually require technical support from IT specialists. This dependency increases costs and delays implementation. In this context, AI represents a major opportunity: its ability to generate and modify code accessibly allows the creation or adaptation of low-cost technological products, narrowing the gap between technical design and real student needs [44]. Although AI-generated software may not reach the technical refinement of professionally supervised development, it offers an effective, economical, and flexible alternative to enhance accessibility and educational participation for students with disabilities, particularly in low-resource or time-constrained contexts.

Methods

Overview

The software development process was carried out iteratively with the assistance of 8 different AI models, 6 of which were free and 2 were paid (Table 2). These models were selected to generate results applicable to contexts where cost may be a limiting factor. The chatbots were asked to program a software solution to convert an arcade fighting-style controller into a computer control device similar to an adapted mouse. Communication with the AIs was conducted using predetermined prompts prior to testing.

The objective was not only to obtain a functional program but also to significantly reduce the feedback provided to each AI and subsequently analyze each model's ability to solve the problem as efficiently as possible, assessing whether it could produce a complete and functional solution. This approach enabled the comparison of performance, accuracy, and coherence across different platforms within the same experimental context, providing objective data on their real effectiveness in supporting software development through natural language instructions.

Table 2. AIs used, models, and price.

Company	AI[a]	Model	Price
OpenAI	ChatGPT (Free)	GPT-4.1 mini	N/A[b]
OpenAI	ChatGPT (Pro)	GPT-5	€23 (US $26.36)/month
Google	Gemini (Free)	Gemini 2.5 Pro	N/A
Google	Gemini (Paid)	Gemini 2.5 Pro	€21.99 (US $25.20)/month
Anthropic	Claude (Free)	Sonnet 4.5	N/A
DeepSeek	DeepSeek	DeepSeek V3.2	N/A
Microsoft	Copilot	o3 mini	N/A

[a]AI: artificial intelligence.
[b]N/A: not applicable.

To ensure experimental validity and avoid prior learning or contamination from conversational memory, each evaluation was conducted in entirely new conversations and, in the case of free plans, through newly created accounts. This strategy is grounded in empirical evidence showing that conversational history can induce bias or interference between tasks, thereby affecting model responses [45]. Similarly, literature on data contamination in language models indicates that any prior exposure to the evaluated information alters the reliability of results [46,47]. Even slight reformulations or lingering contextual information can modify the direction and quality of the generated output [48]. Therefore, initiating each session in a clean environment constitutes a methodologically necessary measure to control carryover bias, prevent contextual leakage, and ensure independence between trials, thereby preserving the external comparability of performance and accuracy across the evaluated models.

Ethical Considerations

The Research Ethics Committee of Valencia Catholic University Saint Vincent Martyr approved the study (UCV/2023-2024/010).

Test Protocol

The first step in initiating the coding process was to define the functions that could be beneficial and link each of them to the prompts that would be sent to the different AIs (Table 3). The definition of these prompts and the design of the conversation were based on two key concepts:

1. Iterative execution: the initial prompt is sent; the generated code is tested for functionality; and if it works successfully, the next prompt is sent with the intention of iteratively adding new features to the designed program.
2. Error handling: no feedback is provided, only the message "it doesn't work, fix it." This approach is based on the understanding that a person without technical qualifications might not comprehend the underlying problems in the code and would likely just ask the AI to fix the errors encountered. Such a person might be able to inform the AI that, for example, the joystick is not working but not describe more complex interaction failures or interpret Python error messages.

Table 3. Function desired and chosen prompt.

Function	Prompt
1. Move mouse with joystick	I have no knowledge of programming.
	I have a generic fighting style Game Pad that has a single joystick and several buttons.
	I need you to code a program in Python that allows the joystick on this device to generate the movements of the computer mouse.
2. Single right and left click	I need you to code left and click buttons like the mouse has.
3. Allow button reassignment	I need you to code in an app that allows the program to reassign keys in the gamepad.
4. Double left click from a single button	I need you to add an extra button that performs a double click in single click.
5. Toggle hold for left click	I need you to add an extra button that holds left click when pressed and releases when pressed again.
6. Scroll page up and down	I need you to add two extra buttons for scrolling up and down.
7. Increase and decrease volume	I need you to add extra buttons for volume up and down.
8. Assign a program to a button	I need you to add an extra button that opens Google Chrome when pressed.
9. Perform an automatic click after 3 seconds of mouse inactivity	Make it so that when the mouse automatically clicks once when standing still for 3 seconds.
10. Allow double-click with a configurable delay	I need you to add a delay that allows two clicks made within 3 seconds to register as a double click.
11. Type the user's name with a button	I need you to add a button that writes the name Pelayo.

Function	Prompt
12. Pause and unpause	I need you to add a button that toggles between pause and play.
13. Ignore movement during the first second of joystick input	I need you to add a delay that ignores the first seconds of movement in any direction and then starts moving.
14. Automatically send a help email	I need you to add an extra button that automatically sends an email saying “Help” to francesc.banuls@ucv.es.
15. Generate an app that allows modifying sensitivity and delay time for automatic clicking	I need you to code in an app that allows me to change sensitivity of the joystick and how long the mouse waits after being still to automatically click.
16. Create a program and installer	I need you to compile the program made into an app by coding installer for this app.

Based on this premise, and to reduce inconsistencies, it was considered that modern AIs increasingly incorporate more integrated self-feedback mechanisms, independent from human input, and that models should be capable of performing such functions [49]. All testing was conducted by the authors, in the context of an autonomous reference center for disability in Valencia, Spain, during 4 days, from October 10, 2025, to October 14, 2025. Subsequently, all testing was conducted with the approval of the university’s ethics committee with code UCV/2023-2024/010.

Additionally, all testing was conducted using a Kubii USB Arcade Controller reference ODARCADE, connected via USB to a Windows 11 computer using Python 3.13 and having installed the libraries pygame and pyautogui.

The selection and design of the prompt constitute a key methodological element in this study, as its structure determines the comparative validity among the different AI models analyzed. Its uniformity follows the principle of instructional consistency, emphasizing that coherence in the formulation of commands is essential to ensure comparability between conversational systems [10]. Complementarily, the structure of the prompt conditions the type of cognitive processing activated by the model; therefore, keeping it constant allows the isolation of the intrinsic effect of each architecture [11]. Minimal wording differences can change both performance and the relative ranking of models, reinforcing the need to use a standardized prompt. Accordingly, the prompts were designed based on 3 theoretical principles [12,28,50].

The first is defining a nontechnical user role, representative of professionals without programming training. Contextualizing the interlocutor’s identity allows for adjusting response complexity, and that assigning an explicit role guides the model’s generation toward a semantically coherent and functional framework. This role also demands that all models are accessed from their web-found standard versions without adjusting models or using specific more technical or specially designed models for programming. This can be seen in the use of Copilot instead of specific instances coded into GitHub or other programming platforms [10,11].

The second is presenting a sequence of actions formulated in natural language and logical order. Procedural prompts activate step-by-step reasoning that promotes coherent and structured results, facilitating the translation of human descriptions into computational processes (as the de facto interaction mode with chatbots), although such prompts may carry a higher risk of drift [11].

The third is detailing functional requirements, such as cursor control or command execution. Precision in parameters or specific conditions reduces ambiguity and improves consistency of the output, making the prompt a cognitive interface between human language and automated execution [12,28].

These 3 concepts serve as guiding principles to mediate between the natural language of a nontechnical professional and the AIs. To more faithfully emulate the real process of software design by a person without advanced technical knowledge, 2 conditions were established to determine when to terminate the programming attempts.

The first one is that 3 consecutive nonfunctional generations either fail to execute, only partially implement, or lose communication with the gamepad. It is understood that if 3 consecutive iterations cannot fix inherited errors without feedback, it is unlikely that subsequent ones will do so.

Prompts are considered successful when they can correctly implement the full capabilities described in Table 1 for each prompt without compromising existing functionality. They are considered partially successful when they either maintain previous capabilities but only partially implement new functions or when they correctly implement new functions but lose previous ones. Finally, prompts are considered unsuccessful when they either fail to execute or lose communication with the gamepad. Partially successful prompts are marked as such to better understand the performance of different models but are functionally considered unsuccessful as they did not achieve the desired function. As such, 3 unsuccessful or partially successful prompts (with 2 tries each) stop interactions with a model.

The other way in which interactions with models are stopped is when reaching the daily prompt limit in free plans, to better reflect that the prompt limit is one of the most significant constraints of these models.

Goals

The experimental design was developed with the explicit understanding that current AI systems are inherently nondeterministic and that the replication of results will be inconsistent [51]. Consequently, the study objectives reflect this fact:

- General goal 1: compare the effectiveness of the most widely used AI solutions as software programmers for creating personalized assistive products using nontechnical natural language.
    - Specific goal 1: describe the relationship between cost and effectiveness of the models used.
    - Specific goal 2: analyze the ease of use, consistency, and accuracy of each model.

Results

Overview

The 8 AI solutions initially proposed were used to progressively design robust code solutions, incorporating functions of increasing complexity inspired by APs with similar objectives.

The results can be divided according to the cost of the alternatives, distinguishing between those that are free of charge and those requiring a subscription. The latter provided more consistent results and better adaptation to the project's functional requirements, demonstrating a greater understanding of the requested functions. However, it is important to note that these results should be interpreted within the framework of the specific characteristics of the study and that even under identical conditions, different outcomes could be obtained due to the nondeterministic nature of these technologies.

Paid Alternatives

The subscription-based alternatives delivered consistent results, generating code that can generally be considered successful in meeting the required functions. This category includes ChatGPT Pro, based on the GPT-5 model, and Gemini Pro, which uses Google's advanced Gemini Pro-2.5 model.

Regarding Gemini Pro, the AI solution developed by Google proved to be the most robust option. It successfully implemented 14 out of the 16 required functions within a single application, without any rollback of previously implemented features during the design process and generated functional code with an average of 1.25 (SD 0.45) prompts. Gemini was even partially able to anticipate the implementation of future functions in earlier items: item 2 was successfully integrated within item 1; item 6 was partially included in item 3 by adding a scrolling mode, although without the 2 specific keys intended for that function; and item 15 was partially addressed in item 3 through a joystick sensitivity slider. The only functions not achieved were items 14 and 16. Item 14 could be considered partially achieved, as the AI managed to open the email app and compose a message addressed to the correct recipient but was unable to send it. All partially achieved items are considered not achieved when counting successful items implemented. Regarding item 16, Gemini could not autonomously generate an installable app, and the alternative solutions proposed in Python were also nonfunctional.

One thing to note is that both items (14 and 16) did not work across the board for security reasons, as AIs are not capable of generating programs that have such deep access to your operative system. Seeing how the AIs approached a task they knew beforehand was impossible to fully be able to perform is interesting and shows the AIs' problem-solving skill navigating natural language generated prompts, which may be difficult or impossible to fulfill.

Complementarily, OpenAI's paid version (ChatGPT Pro using GPT-5) also demonstrated solid performance, successfully implementing 11 out of the 16 required functions and producing functional code with an average of 1.31 (SD 0.48) prompts. GPT-5 showed the same limitation as Gemini Pro with respect to sending the email in item 14 and was likewise unable to compile an executable file or provide stable Python-based compilation solutions. Additionally, OpenAI's AI had difficulties implementing joystick controls in item 1, which caused the device to move the cursor only downward and to the right. This error persisted in subsequent iterations, affecting later results such as items 13 and 15, which were only partially achieved due to the lack of precise joystick control. ChatGPT also displayed inconsistency, producing multiple files for different functions instead of integrating them into a single app and rolling back previously implemented features without justification or explicit request.

Free Alternatives

The free alternatives, in contrast to the subscription-based ones, produced notably less consistent results. In some cases, this was due to the limitations of the models themselves, which were unable to meet the functional requirements set for the task. In other cases, restrictions on the number of interactions significantly limited development, constituting a key distinction. It cannot be asserted that greater time investment would have yielded results equivalent to the paid versions, since the possibility of submitting additional prompts was not available. Therefore, in this study, the maximum number of interactions allowed by each model in its free version was established as a methodological limitation. Conversely, the models that failed to generate functional solutions can, for the time being, be considered inoperative for the intended function.

The free AI with the best performance was DeepSeek. This model generated the first 3 items in the initial prompt and even implemented item 2 without it being explicitly requested in item 1. However, the AI was unable to technically complete items 4, 5, and 6, implementing item 5 through a bind function that only allowed the assignment of a single key and caused the program to crash when attempting to add more combinations. This model generated functional items with an average of 1.88 (SD 0.34) prompts, achieving 4 out of the 16 required functions.

The second-best performance can be attributed to GPT Free. The free version of GPT successfully produced a functional prototype on the first attempt for the first 3 prompts but maintained the inconsistency observed in its paid counterpart. From the fourth prompt onward, it began producing nonfunctional code, leading to discontinuation

after the seventh attempt. On average, it generated functional prototypes with 1.81 prompts and achieved 3 out of the 16 intended functions.

After GPT, in terms of performance, is Gemini Pro Free. Although it theoretically uses the same model as the paid version, its results differed appreciably. The free version produced correct and functional outputs up to the permitted prompt limit, forcing the process to stop prematurely. It successfully implemented the first 2 items and partially the third, generating a program that ran natively in the browser. Subsequently, time-based restrictions prevented continuation. On average, it produced functional prototypes with 1.88 prompts and achieved 2 out of the 16 complete functions.

The second-to-last worst-performing model is Claude. The Sonnet 4.5 model managed to implement only the first of the planned functions. After failing to execute functions 3, 4, and 5, its use was discontinued. Claude generated functional prototypes with an average of 1.94 (SD 0.25) prompts, achieving 1 out of the 16 proposed functions.

Finally, the individually worst-performing model is Copilot. Using its Deep Think model, Microsoft's AI failed to generate any executable Python files within the first 3 prompts, leading to the termination of testing after the fourth attempt. It did not produce any functional prototypes, with an average of 2 prompts and a total of 0 out of 16 functions achieved.

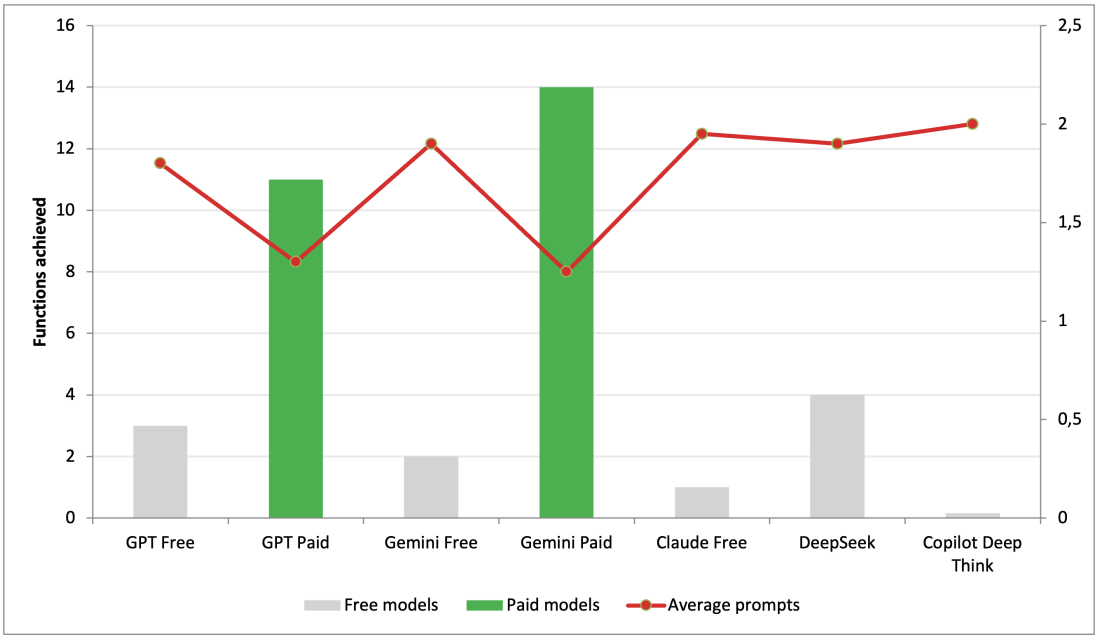
All models and a color-coded list of the items successfully implemented by each, along with the number of prompts required, are presented in Table 4. Additionally, prompts achieved and average prompts per model are presented in Figure 1.

Table 4. Functions achieved and prompts needed.

Required function	GPT Free	GPT Paid	Gemini Free	Gemini Paid	Claude Free	DeepSeek	Copilot Deep Think
1	✓[a]	×[b]	✓	✓	✓	✓	×
2	✓	✓	×	✓	×	✓[c]	×
3	✓	✓	✓[c]	×	×	✓	×
4	×	✓	×	✓	×	×	×
5	×	✓	×	✓	×	✓	×
6	×	✓	×	✓[c]	×	×	×
7	×	✓	×	✓	×	×	×
8	×	✓	×	✓	×	×	×
9	×	✓	×	✓	×	×	×
10	×	✓	×	×	×	×	×
11	×	✓	×	✓	×	×	×
12	×	✓	×	✓	×	×	×
13	×	×	×	✓	×	×	×
14	×	×	×	×	×	×	×
15	×	×	×	✓[c]	×	×	×
16	×	×	×	×	×	×	×

[a]Items achieved.
[b]Items not achieved or partially achieved. Partially achieved items are counted as unsuccessful.
[c]Functions already implemented or partially already implemented in previous interactions without direct prompts for their inclusion.

Figure 1. Prompts achieved and average prompts per model.



Discussion

Principal Findings

The results obtained show that the paid AI alternatives produced robust code, achieving 14 out of 16 functions with Gemini Pro and 11 out of 16 functions with ChatGPT Plus, respectively. The free-to-use alternatives, on the other hand, managed to integrate a variable, though consistently lower, number of functions ranging from none to 4 successfully implemented. The discussion of the results obtained can therefore be divided into 3 key areas (Figure 1).

Superiority of Paid AIs

The use of paid AI systems offers several advantages, among which the most significant are access to more advanced models and the elimination or substantial expansion of the number of prompts allowed within a given period. These advantages naturally make paid AI solutions more reliable for performing complex tasks. The main improvements can be summarized in 3 key points.

Paid versions typically operate on newer, more advanced architectures with higher parameter capacity and improved optimization strategies and training based on larger and higher-quality datasets. In literature, larger and better-trained models consistently achieve superior results in both code generation and natural language understanding tasks. For instance, GPT-4 performs substantially better in programming tasks than earlier versions [52].

Due to these technical enhancements, paid models generally demonstrate a stronger grasp of natural language and communicative context, which increases the likelihood of successful interactions in this specific case scenario. They are also better at maintaining coherence across sequential instructions, retaining previously defined functions, and being better at avoiding contradictions and hallucinations. This allows them to integrate multiple features coherently and progressively with fewer rollbacks or loss of content [53-55].

The removal or significant reduction of restrictions on the number of prompts permitted per time interval is particularly relevant when considered alongside the highlighted advantages. Older free models often have weaker memory capabilities and greater difficulty minimizing inconsistencies, rollbacks, and errors across different interaction sessions [56]. Increasing the interaction limits from, for example, 5 per session in the free version of Gemini to 100, or from 10 every 5 hours in GPT-4.1 mini to 160 every 3 hours in GPT-5, not only eases time constraints but also reduces errors and can substantially improve code quality [57,58].

In summary, the differences in model capacity, contextual understanding, memory, and reasoning largely explain why paid AIs may achieve significantly more functional results than their free counterparts. This outcome is expected, as paying for a service naturally entails the expectation of better performance. However, another relevant point of discussion lies in comparing paid AIs with one another and questioning why Google's AI seems to have outperformed GPT in this specific scenario, despite the latter's longer development history and its recent update to version 5.

Differences Between Gemini Pro and GPT-5

Although it is not possible to access the internal technical details of Gemini Pro, or of any major AI model, the existing literature published by the companies behind these models appears to support the results obtained in this specific case scenario, pointing to 3 key aspects that may explain why Gemini stood out as the most robust alternative for generating code from natural language given the restrictions and tasks asked.

On the one hand, Gemini may have better alignment between intention and execution. According to the Google Gemini team, the training of Gemini Pro placed greater emphasis on the correspondence between high-level instructions and structured code generation, giving it an advantage in translating complex requirements into functional code. The model achieved scores above 74% on specialized programming benchmarks [59].

This may also lead to much greater stability and coherence in long sessions. In tasks involving multiple functions (such as the 16 required in this study), the ability to maintain coherence across many iterations is crucial. If Gemini Pro manages internal session states more effectively, it becomes less prone to "forget" previously implemented functions or to inconsistently rename elements [60].

On the other hand, and especially relevant to the study, prompt optimization may be more tolerant of natural language. A distinct training approach regarding the model's handling of natural language results in greater adaptability to intricate instructions, improved contextual understanding, and enhanced problem-solving abilities. This makes Gemini Pro more resilient to inaccuracies or ambiguities in prompts written in nontechnical natural language, whereas ChatGPT Plus may tend to require more carefully formulated instructions to produce reliable outcomes [59,60].

Limitations and Strengths of Free AIs

As previously mentioned, free AI systems may present significant limitations related to the use of simpler and more restricted models, limited numbers of prompts, and, in some cases, the restriction to process multimodal information. These constraints make free models more challenging to use or less adaptable to users' needs. The standardized subscription fee of US $20, as implemented by companies such as OpenAI and Google, remains possibly too expensive for low-resource contexts in many parts of the world.

In recent months, both companies have begun introducing country-based pricing, adjusting subscription costs to local purchasing power. OpenAI (2025) implemented multicurrency billing to reduce conversion costs, and in India, the Plus plan is now offered in rupees, alongside a lower-cost alternative named ChatGPT Go [61]. This strategy echoes *The Economist*'s Big Mac Index, which illustrates how multinational companies adjust prices according to economic context. AI can enhance accessibility in education, but its impact is constrained by the economic barriers preventing access to paid tools [62]. However, if AI design fails to account for income inequalities within a single country, despite global cost-adjustment strategies, it may reinforce digital exclusion. Thus, exploring the use and functionality of free versions and comparing them with paid alternatives as a potential source of social inequality becomes a social necessity to ensure equitable access to knowledge and accessibility [63,64].

In our test, among the free options, Gemini Free managed to implement 2 functions with relative stability during the first iteration, approaching a third before reaching the interaction limit. Considering the performance of its paid counterpart under fewer restrictions, it is likely that free versions could offer solid, no-cost solutions, albeit requiring a significant time investment, provided the model seems to be able to maintain internal coherence across sessions [59].

In contrast, GPT Free and DeepSeek achieved a greater number of items but lost track of the ongoing programming process, producing unstable code that would require the user to identify errors and explain them to the chatbot. While this may be possible, it is difficult to fully assess how much feedback a nontechnical user may be able to provide, and as such, it is difficult to give certain conclusions about these models. Professionals with intermediate technical knowledge could use these 2 models effectively to develop software solutions, but nontechnical users may be unable to [65,66].

Finally, Claude and Copilot, in their free versions, showed no real ability to generate operational code. In the case of Claude, this is understandable, since its website explicitly states that only the Pro version provides such capabilities [67]. The second model, Copilot, simply appears unable to generate functional, plug-and-play code. This is also due to the several factors, such as copilot being used in the web version (not optimized to do such tasks) without any specific model or implementation chosen [68]. In both case scenarios, using Claude Pro with programming capabilities and a version of Copilot optimized for programming may generate more competent results.

These findings may point to the fact that not all free versions are equivalent: some models are sufficiently robust for basic tasks, while others are less capable with the task chosen, the prompt limit, the version used, and many other factors contributing to their success.

One key limitation to have in mind is that only 1 run has been carried out with each model to mimic a real case scenario of a nontechnical user using these models. Given the nondeterministic nature of these models, more different runs around the same task may produce different results.

Observations and Practical Implications

The observations, limitations, and implications for practice derived from this study can be summarized in 10 key points that integrate the main insights gained throughout the research. Four main points stand out as the implications of the results obtained.

Some AI models, such as ChatGPT (both 4.1 mini and 5), may produce inconsistent results, hallucinating or forgetting its progress. This can be seen by these models including renaming functions without apparent reason, changing project names, generating new programs instead of adding features to the existing one, or rolling back previously correct functions. This suggests that OpenAI models can generate the intended products but require a technical skill level beyond that of nonspecialist professionals, significantly complicating the process of obtaining functional prototypes for nontechnical users in this specific case scenario [66,69].

This behavior is sometimes categorized as AI "drift" and means that, during extended sessions, models may change behavior, forget previously defined functions, or misinterpret prompts that are conceptually equivalent. Maintaining a clear "narrative thread" and consistent prompt formulation is therefore essential. Natural language alone may not suffice, and the risk of accumulating errors that render the code nonfunctional remains high [70,71].

In this specific test, Gemini showed greater consistency interpreting natural-language instructions and required fewer technical reformulations or corrections. It also showed a lower risk of drift. In contexts where users represent the target population, these traits may reduce the barrier to entry and increase the likelihood of producing functional prototypes. The Pro version is notably more efficient and reliable, but the free version can achieve similar outcomes with sufficient time investment if the time between tries does not induce AI drift [72].

While AIs can generate code that is operational and useful, it must be remembered that its efficiency (execution time, memory usage, and design simplicity) is generally suboptimal. For this reason, professional supervision, or ideally full implementation, by a qualified software engineer remains preferable. Studies, such as EffiBench, have shown that AI-generated code tends to be, on average, less efficient than optimized human solutions [73].

The results obtained in this domain (low-cost AP adaptations with 16 functions with no feedback and in a single run) are constrained to the models and prompts used and to the absence of user feedback, maybe reflecting the condition of a nontechnical user. Other models or differently designed prompts might yield different outcomes. The practical implications are as follows:

1. For real-world AP adaptations: Collaboration with software professionals should always be prioritized when budgets allow. If it is not feasible, both paid models tested may be the most effective choice, as they perform better in programming tasks than other paid alternatives. When access to paid tools is possible, they remain the most robust option. However, free models such as Gemini Free, GPT Free, or even DeepSeek can provide functional results in exchange for greater time investment.
2. For low-budget or educational contexts: Free versions, particularly Gemini Free or GPT Free, can serve as valuable tools for initial prototyping, though additional iterations, revisions, and acceptance of functional limitations will be necessary [74].
3. Effective prompt design and problem segmentation: Breaking the task into smaller subfunctions and crafting clear, stepwise prompts improves the success rate, especially in free models.
4. Continuous monitoring and automated testing: Implementing automated unit tests (eg, by prompting the AI to generate its own test cases) may help detect errors or inconsistencies in the generated code.
5. Caution with long sessions or evolving prompts: Avoid overly long or evolving prompts within a single session and restart the interaction when the model shows signs of drift to preserve consistency.

Conclusions

In conclusion, the comparative analysis of different AI models applied to the generation of functional software for designing personalized APs using natural language reveals clear differences in performance, usability, and consistency. Paid models, such as Gemini Pro and ChatGPT Plus, demonstrated greater efficiency and reliability, achieving 14 and 11 out of 16 required functions, respectively, while free alternatives ranged between 0 and 4. These results confirm that the technical advancements of newer and more sophisticated models, such as expanded context windows, improved multimodal communication, enhanced natural language comprehension, reduced drift, and fewer prompt limitations, directly influence the functional quality of the generated code.

Regarding the relationship between cost and effectiveness, Gemini Pro may stand out as the most balanced option, in this specific case scenario offering higher precision and coherence at a cost equivalent to other paid alternatives. For low-resource contexts, DeepSeek emerges as a viable free alternative that, despite its limitations, can produce acceptable results given sufficient time and technical supervision, while the free versions of Gemini can deliver solid outputs with adequate time investment.

In terms of ease of use and consistency, Gemini demonstrated greater tolerance for nontechnical natural language and more stable performance in long sessions, making it particularly suitable for users without programming knowledge, an essential aspect when the goal is to promote inclusive and low-cost design of APs.

Overall, the findings suggest that AI can serve as an effective bridge between health care or educational professionals and programming, enabling the creation of personalized assistive solutions without the need for advanced software development expertise. Nevertheless, the results also highlight the critical need for collaboration with qualified IT professionals, given the importance of human technical oversight, careful prompt design, and continuous testing to prevent drift and ensure the generation of stable software alternatives.

Future research could focus on exploring how AI models handle problem-solving from visual inputs (eg, images sent to chatbots), their capacity for self-generated prompt design, and the evaluation of emerging or alternative models.

Funding

This study was carried out within the framework of the research project PID2022-142309OB-100 funded by the Ministry of Science and Innovation, State Research Agency and in collaboration with the Capacitas Campus of the Universidad Católica de Valéncia San Vicente Mártir (UCV).

Conflicts of Interest

None declared.

References

1. Xu Y, Liu X, Cao X, et al. Artificial intelligence: a powerful paradigm for scientific research. Innovation (Camb). Oct 28, 2021;2(4):100179. [doi: 10.1016/j.xinn.2021.100179] [Medline: 34877560]
2. Sauvola J, Tarkoma S, Klemettinen M, Riekki J, Doermann D. Future of software development with generative AI. Autom Softw Eng. May 2024;31(1):26. [doi: 10.1007/s10515-024-00426-z]
3. Qiu K, Puccinelli N, Ciniselli M, Di Grazia L. From today's code to tomorrow's symphony: the AI transformation of developer's routine by 2030. ACM Trans Softw Eng Methodol. Jun 30, 2025;34(5):1-17. [doi: 10.1145/3709353]
4. Alenezi M, Akour M. AI-driven innovations in software engineering: a review of current practices and future directions. Appl Sci (Basel). 2025;15(3):1344. [doi: 10.3390/app15031344]
5. Yang EW, Waldrup B, Velazquez-Villarreal E. Conversational AI agent for precision oncology: AI-HOPE-WNT integrates clinical and genomic data to investigate WNT pathway dysregulation in colorectal cancer. Front Artif Intell. 2025;8:1624797. [doi: 10.3389/frai.2025.1624797] [Medline: 40860720]
6. Yu L. Paradigm shift on coding productivity using GenAI. Presented at: Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering (EASE '25); Jun 17-20, 2025; Istanbul, Türkiye. [doi: 10.1145/3756681.3757081]
7. Sergeyuk A, Titov S, Izadi M. In-IDE human-AI experience in the era of large language models; a literature review. Presented at: IDE '24: Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments; Apr 20, 2024; Lisbon, Portugal. [doi: 10.1145/3643796.3648463]
8. Beheshti A. Natural language-oriented programming (NLOP): towards democratizing software creation. Presented at: 2024 IEEE International Conference on Software Services Engineering (SSE); Jul 7-13, 2024; Shenzhen, China. [doi: 10.1109/SSE62657.2024.00047]
9. Dalsgaard P. Thinking through prompting: cognitive mediation in human–AI interaction. Presented at: Proceedings of the European Conference on Cognitive Ergonomics (ECCE '25); Oct 7-10, 2025; Tallinn, Estonia. 2025.[doi: 10.1145/3746175.3747192]
10. McTear M, Callejas Z, Griol D. The Conversational Interface Talking to Smart Devices. Springer International Publishing; 2016. [doi: 10.1007/978-3-319-32967-3]
11. Morales-Chan M. Explorando el potencial de chat GPT: una clasificación de prompts efectivos para la enseñanza [Report in Spanish]. Universidad Galileo; 2023. URL: https://biblioteca.galileo.edu/tesario/handle/123456789/1348 [Accessed 2026-03-07]
12. Brown TB, Mann B, Ryder N, et al. Language models are few-shot learners. Presented at: Proceedings of the 34rd International Conference on Neural Information Processing Systems (NeurIPS 2020); Dec 6-12, 2020; Vancouver, Canada. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf [Accessed 2026-03-07]
13. Serban I, Sordoni A, Lowe R, et al. A hierarchical latent variable encoder-decoder model for generating dialogues. Presented at: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017); Feb 4-9, 2017; San Francisco, California, USA. 2017.[doi: 10.1609/aaai.v31i1.10983]
14. Dathathri S, Madotto A, Lan Z, Fung P, Neubig G. Plug and play language models: a simple approach to controlled text generation. Presented at: Findings of the Association for Computational Linguistics: EMNLP 2021; Nov 7-11, 2021; Punta Cana, Dominican Republic. [doi: 10.18653/v1/2021.findings-emnlp.334]
15. Korzynski P, Mazurek G, Krzypkowska P, Kurasinski A. Artificial intelligence prompt engineering as a new digital competence: analysis of generative AI technologies such as ChatGPT. Entrepren Bus Econ Rev. 2023;11(3):25-37. [doi: 10.15678/EBER.2023.110302]
16. Kumar H, Musabirov I, Shi J, et al. Exploring the design of prompts for applying GPT-3 based chatbots: a mental wellbeing case study on Mechanical Turk. arXiv. Preprint posted online on Sep 22, 2022. [doi: 10.48550/arXiv.2209.11344]
17. Zhou K, Zhang J, Liu X, Sun M. A systematic survey of prompt engineering in large language models: techniques and applications. arXiv. Preprint posted online on Feb 5, 2024. [doi: 10.48550/arXiv.2402.07927]

18. Fagbohun O, Harrison RM, Dereventsov A. An empirical categorization of prompting techniques for large language models: a practitioner's guide. J Artif Intell Mach Learn Data Sci. ;1(4):1-11. [doi: 10.51219/JAIMLD/Oluwole-Fagbohun/15]
19. Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. Presented at: Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022); Nov 28 to Dec 9, 2022; New Orleans, Louisiana, USA. URL: https://openreview.net/pdf?id=_VjQlMeSB_J [Accessed 2026-03-07]
20. MadaanA, TandomN, GuptaP. Self-refine: iterative refinement with self-feedback. Presented at: Advances in Neural Information Processing Systems 36; Dec 10-16, 2023; New Orleans, Louisiana, USA. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html [Accessed 2026-03-18]
21. Anam RK. Prompt engineering and the effectiveness of large language models in enhancing human productivity. arXiv. Preprint posted online on May 10, 2025. [doi: 10.48550/arXiv.2507.18638]
22. Xu Y, Thomas T, Yu CL, Pan EZ. What makes children perceive or not perceive minds in generative AI? Comput Hum Behav Artif Hum. May 2025;4:100135. [doi: 10.1016/j.chbah.2025.100135]
23. Brandtzaeg PB, Skjuve M, Følstad A. My AI friend: how users of a social chatbot understand their human–AI friendship. Hum Commun Res. Jun 29, 2022;48(3):404-429. [doi: 10.1093/hcr/hqac008]
24. Panfili L, Duman S, Nave A, Ridgeway KP, Eversole N, Sarikaya R. Human-AI interactions through a Gricean lens. Proc Ling Soc Amer. ;6(1):288. [doi: 10.3765/plsa.v6i1.4971]
25. Zheng Q, Tang Y, Liu Y, Liu W, Huang Y. UX research on conversational human-AI interaction: a literature review of the ACM digital library. Presented at: CHI '22: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems; Apr 29 to May 5, 2022; New Orleans, Louisiana, USA. Apr 29, 2022.[doi: 10.1145/3491102.3501855]
26. Gülay E, Picco E, Glerean E, Coupette C. Relational dissonance in human–AI interactions: the case of knowledge work. arXiv. Preprint posted online on Sep 19, 2025. [doi: 10.48550/arXiv.2509.15836]
27. Kim Y, Lee J, Kim S, Park J, Kim J. Understanding users' dissatisfaction with ChatGPT responses: types, resolving tactics, and the effect of knowledge level. Presented at: Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI 2024); Mar 18-21, 2024; Greenville, South Carolina, USA. [doi: 10.1145/3640543.3645148]
28. Hernández Caralt M, Sekulić I, Carevic F, et al. Stupid robot, I want to speak to a human!" User frustration detection in task-oriented dialog systems. Presented at: Proceedings of the 31st International Conference on Computational Linguistics: Industry Track; Jan 19-24, 2025; Abu Dhabi, UAE. 2025.URL: https://aclanthology.org/2025.coling-industry.23/ [Accessed 2026-03-07]
29. Eiband M, Schneider H, Bilandzic M, Fazekas-Con J, Haug M, Hussmann H. Bringing transparency design into practice. Presented at: Proceedings of the 23rd International Conference on Intelligent User Interfaces (IUI 2018); Mar 7-11, 2018; Tokyo, Japan. URL: https://dl.acm.org/doi/proceedings/10.1145/3172944 [doi: 10.1145/3172944.3172961]
30. Degachi C, Freire SK, Niforatos E, Kortuem G. Understanding mental models of generative conversational search and the effect of interface transparency. arXiv. Preprint posted online on Jun 4, 2025. [doi: 10.48550/arXiv.2506.03807]
31. Jardinez MJ, Natividad LR. The advantages and challenges of inclusive education: striving for equity in the classroom. Shanlax Int J Educ. 2024;12(2):57-65. [doi: 10.34293/education.v12i2.7182]
32. Bani Odeh K, Lach LM. Barriers to, and facilitators of, education for children with disabilities worldwide: a descriptive review. Front Public Health. 2023;11:1294849. [doi: 10.3389/fpubh.2023.1294849] [Medline: 38292375]
33. Pérez Echeverría MP, Cabellos B, Pozo JI. The use of ICT in classrooms: the effect of the pandemic. Educ Inf Technol. Jul 2025;30(10):14069-14093. [doi: 10.1007/s10639-024-13124-w]
34. Alanazi AS, Benlaria H. Understanding the landscape: assistive technology and work challenges for people with disabilities in Saudi Arabia. Humanit Soc Sci Commun. 2024;11(1):1608. [doi: 10.1057/s41599-024-04023-z]
35. The state of online recruiting 2024. iHire. URL: https://www.ihire.com/resourcecenter/employer/pages/the-state-of-online-recruiting-2024 [Accessed 2025-10-22]
36. ISO 9999:2022—assistive products for persons with disability: classification and terminology. International Organization for Standardization (ISO); 2022. URL: https://cdn.standards.iteh.ai/samples/72464/3f3608ed0bff4545bd53c02373f8cdbb/ISO-9999-2022.pdf [Accessed 2026-03-07]
37. ISO 9999:2016—assistive products for persons with disability: classification and terminology. International Organization for Standardization (ISO); 2016. URL: https://cdn.standards.iteh.ai/samples/60547/4694b28419324e45810538491357903c/ISO-9999-2016.pdf [Accessed 2026-03-07]
38. International Classification of Functioning, Disability and Health (ICF). World Health Organization (WHO); 2001. URL: https://www.who.int/standards/classifications/international-classification-of-functioning-disability-and-health [Accessed 2026-03-07]

39. Tomás V. La discapacidad como elemento de discriminación positiva. In: Los Derechos de Los Niños, Responsabilidad de Todos [Book in Spanish]. Universidad de Murcia; 2007:213-218. URL: https://dialnet.unirioja.es/servlet/articulo?codigo=2306106 [Accessed 2026-03-07]
40. Vila-Merino ES, Rascón-Gómez T, Calderón-Almendros I. Discapacidad, estigma y sufrimiento en las escuelas. Narrativas emergentes por el derecho a la educación inclusiva [Article in Spanish]. Educ XX1. 2024;27(1):353-371. [doi: 10.5944/educxx1.36753]
41. de Beco G. The right to inclusive education according to Article 24 of the UN Convention on the Rights of Persons with Disabilities: background, requirements and (remaining) questions. Neth Q Hum Rights. Sep 2014;32(3):263-287. [doi: 10.1177/016934411403200304]
42. Assistive technology. World Health Organization (WHO). 2024. URL: https://www.who.int/news-room/fact-sheets/detail/assistive-technology [Accessed 2025-10-22]
43. Fteiha M, Al-Rashaida M, Elsori D, Khalil A, Al Bustami G. Obstacles for using assistive technology in centres of special needs in the UAE. Disabil Rehabil Assist Technol. Nov 2024;19(8):2934-2944. [doi: 10.1080/17483107.2024.2323698] [Medline: 38436086]
44. Fernández-Batanero JM, Montenegro-Rueda M, Fernández-Cerero J, García-Martínez I. Assistive technology for the inclusion of students with disabilities: a systematic review. Education Tech Research Dev. 2022;70(5):1911-1930. [doi: 10.1007/s11423-022-10127-7]
45. Gupta A, Sheth I, Raina V, Gales M, Fritz M. LLM task interference: an initial study on the impact of task-switch in conversational history. Presented at: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing; Nov 12-16, 2024; Miami, Florida, USA. [doi: 10.18653/v1/2024.emnlp-main.811]
46. Golchin S, Surdeanu M. Time travel in LLMs: tracing data contamination in large language models. arXiv. Preprint posted online on Aug 16, 2023. [doi: 10.48550/arXiv.2308.08493]
47. Cheng Y, Chang Y, Wu Y. A survey on data contamination for large language models. arXiv. Preprint posted online on Feb 20, 2025. [doi: 10.48550/arXiv.2502.14425]
48. Yang S, Chiang WL, Zheng L, Gonzalez JE, Stoica I. Rethinking benchmark and contamination for language models with rephrased samples. arXiv. Preprint posted online on Nov 8, 2023. [doi: 10.48550/arXiv.2311.04850]
49. Chen Z, Cox D, Gan C, et al. Principle-driven self-alignment of language models from scratch with minimal human supervision. Presented at: 37th Conference on Neural Information Processing Systems (NeurIPS 2023); Dec 10-16, 2023; New Orleans, Louisiana, USA. [doi: 10.52202/075280-0115]
50. Mizrahi M, Kaplan G, Malkin D, Dror R, Shahaf D, Stanovsky G. State of what art? A call for multi-prompt LLM evaluation. Trans Assoc Comput Linguist. Aug 2024;12:933-949. [doi: 10.1162/tacl_a_00681]
51. Ouyang S, Zhang JM, Harman M, Wang M. An empirical study of the non-determinism of ChatGPT in code generation. ACM Trans Softw Eng Methodol. 2023;34(2):1-28. [doi: 10.1145/3697010]
52. Moussiades L, Zografos G, Papakostas G. GPT-4 vs. GPT-3.5 as coding assistants. Research Square. Preprint posted online on Feb 7, 2024. [doi: 10.21203/rs.3.rs-3920214/v1]
53. Shuvo UA, Dip SA, Vaskar NR, Al Islam ABMA. Assessing ChatGPT's code generation capabilities with short vs long context programming problems. Presented at: Proceedings of the 11th International Conference on Networking, Systems, and Security (NSysS 2024); Dec 19-21, 2024; Khulna, Karak, Bangladesh. 2024.[doi: 10.1145/3704522.3704535]
54. OpenAI. GPT-4 technical report. arXiv. Preprint posted online on Mar 15, 2023. [doi: 10.48550/arXiv.2303.08774]
55. Hou Y, Zhan Z, Zhang R. Benchmarking GPT-5 for biomedical natural language processing. arXiv. Preprint posted online on Aug 28, 2025. [doi: 10.48550/arXiv.2509.04462]
56. Wang C, Sun J vince. Unable to forget: proactive Interference reveals working memory limits in LLMs beyond context length. arXiv. Preprint posted online on Jul 9, 2025. [doi: 10.48550/arXiv.2506.08184]
57. GPT-5.3 and GPT-5.4 in ChatGPT. OpenAI Help Center. 2025. URL: https://help.openai.com/en/articles/11909943-gpt-5-in-chatgpt [Accessed 2025-10-20]
58. Gemini Apps limits & upgrades for Google AI subscribers. Gemini Apps Help. 2025. URL: https://support.google.com/gemini/answer/16275805?hl=en [Accessed 2025-10-15]
59. Anil R, Borgeaud S, Alayrac JB, et al. Gemini: a family of highly capable multimodal models. Google DeepMind; 2023. URL: https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf [Accessed 2026-03-07]
60. Alsajri A, Salman HA, Steiti A. Generative models in natural language processing: a comparative study of ChatGPT and Gemini. Babylonian J Artif Intell. 2024;2024:134-145. [doi: 10.58496/BJAI/2024/015]
61. OpenAI goes local in India, ChatGPT now available in INR: check ChatGPT Plus, Pro prices here. India Today. 2025. URL: https://www.indiatoday.in/technology/news/story/openai-goes-local-in-india-chatgpt-now-available-in-inr-check-chatgpt-plus-pro-prices-here-2770840-2025-08-13 [Accessed 2025-10-22]

62. Melo-López VA, Basantes-Andrade A, Gudiño-Mejía CB, Hernández-Martínez E. The impact of artificial intelligence on inclusive education: a systematic review. Educ Sci. 2025;15(5):539. [doi: 10.3390/educsci15050539]
63. Umucu E. Artificial intelligence and health equity for people with disabilities: an integrated framework for disability-inclusive AI design. Inquiry. 2025;62:469580251365472. [doi: 10.1177/00469580251365472] [Medline: 40847466]
64. Daepp MIG, Counts S. The emerging generative artificial intelligence divide in the United States. Presented at: Proceedings of the Nineteenth International AAAI Conference on Web and Social Media (ICWSM 2025); Jun 23-26, 2025; Toronto, Canada. [doi: 10.1609/icwsm.v19i1.35825]
65. Laban P, Hayashi H, Zhou Y, Neville J. LLMs get lost in multi-turn conversation. arXiv. Preprint posted online on May 9, 2025. [doi: 10.48550/arXiv.2505.06120]
66. Liu Y, Le-Cong T, Widyasari R, et al. Refining ChatGPT-generated code: characterizing and mitigating code quality issues. ACM Trans Softw Eng Methodol. Jun 30, 2024;33(5):1-26. [doi: 10.1145/3643674]
67. Claude AI. 2025. URL: https://claude.ai/upgrade [Accessed 2025-10-21]
68. Zhang B, Liang P, Zhou X, Ahmad A, Waseem M. Practices and challenges of using GitHub Copilot: an empirical study. Presented at: The 35th International Conference on Software Engineering and Knowledge Engineering (SEKE 2023); Jul 10-12, 2023. [doi: 10.18293/SEKE2023-077]
69. Bartsch H, Jorgensen O, Rosati D, Hoelscher-Obermaier J, Pfau J. Self-consistency of large language models under ambiguity. Presented at: Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP; Dec 7, 2023; Singapore. [doi: 10.18653/v1/2023.blackboxnlp-1.7]
70. Donge V, Rossi RA, Lai VD, Yoon DS, Hakkani-Tür D, Bui T. Drift no more? Context equilibria in multi-turn LLM interactions. arXiv. Preprint posted online on Oct 9, 2025. [doi: 10.48550/arXiv.2510.07777]
71. Luo Y, Yang Z, Meng F, Li Y, Zhou J, Zhang Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. IEEE Trans Audio Speech Lang Process. 2023;33:3786-3776. [doi: 10.1109/TASLPRO.2025.3606231]
72. Comanici G, Bieber E, Schaeckerman M, et al. Gemini 2.5: pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Google DeepMind; 2025. URL: https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf [Accessed 2026-03-07]
73. Huang D, Qing Y, Shang W, Cui H, Zhang JM. Effibench: benchmarking the efficiency of automatically generated code. Presented at: Proceedings of the 38th International Conference on Neural Information Processing Systems (NeurIPS 2024); Dec 9-17, 2024; Vancouver, BC, Canada. URL: https://papers.nips.cc/paper_files/paper/2024/file/15807b6e09d691fe5e96cdecde6d7b80-Paper-Datasets_and_Benchmarks_Track.pdf [Accessed 2026-03-07]
74. Rakotonirina NC, Hamdy M, Campos JA, et al. From tools to teammates: evaluating LLMs in multi-session coding interactions. Presented at: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Jul 27 to Aug 1, 2025; Vienna, Austria. [doi: 10.18653/v1/2025.acl-long.964]

Abbreviations

AI: artificial intelligence
AP: assistive product

Edited by Sarah Munce; peer-reviewed by Ashraf Elnashar, Jin Liu, Mohammad Al-Agil; submitted 30.Oct.2025; accepted 19.Feb.2026; published 24.Mar.2026

Please cite as:
Bañuls-Lapuerta FA, Marti-Miralles V, Gónzalez-García RJ, Martínez-Rico G
Using Natural Language Prompts With AI Models for Low-Cost Assistive Software Design: Exploratory Comparative Evaluation
JMIR Rehabil Assist Technol 2026;13:e86786
URL: https://rehab.jmir.org/2026/1/e86786
doi: 10.2196/86786