# ChatGPT-Resistant Screening Instrument for Identifying Non-Programmers

Raphael Serafini
Ruhr University Bochum
Bochum, Germany
raphael.serafini@rub.de

Clemens Otto
Ruhr University Bochum
Bochum, Germany
clemens.otto@rub.de

Stefan Albert Horstmann
Ruhr University Bochum
Bochum, Germany
stefan-albert.horstmann@rub.de

Alena Naiakshina
Ruhr University Bochum
Bochum, Germany
alena.naiakshina@rub.de

## ABSTRACT

To ensure the validity of software engineering and IT security studies with professional programmers, it is essential to identify participants without programming skills. Existing screening questions are efficient, cheating robust, and effectively differentiate programmers from non-programmers. However, the release of ChatGPT raises concerns about their continued effectiveness in identifying non-programmers. In a simulated attack, we showed that ChatGPT can easily solve existing screening questions. Therefore, we designed new ChatGPT-resistant screening questions using visual concepts and code comprehension tasks. We evaluated 28 screening questions in an online study with 121 participants involving programmers and non-programmers. Our results showed that questions using visualizations of well-known programming concepts performed best in differentiating between programmers and non-programmers. Participants prompted to use ChatGPT struggled to solve the tasks. They considered ChatGPT ineffective and changed their strategy after a few screening questions. In total, we present six ChatGPT-resistant screening questions that effectively identify non-programmers. We provide recommendations on setting up a ChatGPT-resistant screening instrument that takes less than three minutes to complete by excluding 99.47% of non-programmers while including 94.83% of programmers.

## CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**.

## KEYWORDS

chatgpt, programmer screening, developer study, study protection

**ACM Reference Format:**
Raphael Serafini, Clemens Otto, Stefan Albert Horstmann, and Alena Naiakshina. 2024. ChatGPT-Resistant Screening Instrument for Identifying Non-Programmers. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639075


*ICSE '24, April 14–20, 2024, Lisbon, Portugal*

ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639075

## 1 INTRODUCTION

Conducting empirical research with professional programmers is essential to provide insights into software development issues such as security vulnerabilities, privacy violations, or poor usability. However, recruiting professional programmers for user studies is challenging, especially if a high number of participants is needed for quantitative analysis [1, 5, 8, 18, 25]. Therefore, past research explored different crowdsourcing platforms for participant recruitment such as Clickworker [15], Qualtrics [42], or MTurk [3] [24, 48]. However, compared to end-user studies, higher compensation rates introduce the risk of participants without programming skills participating in studies with programmers [18]. While some platforms (e.g., Qualtrics [42], Prolific [41]) offer filtering options for software development experience, relying on self-reported data can threaten the validity of studies with programmers [18]. Therefore, screeners were recommended to use in line with the different recruitment strategies [17, 24, 48]. Danilova et al. [17] proposed several screening questions to assess whether participants have programming skills. In this work, we refer to the "programmer" and "programming skill" definitions used in [17]. In a follow-up study, Danilova et al. [16] extended the question set and added time limits to provide an efficient and googling-robust instrument.

With the introduction of AI-based applications such as ChatGPT [35], non-programmers are equipped with powerful tools to circumvent the proposed screening questions. ChatGPT is based on a language model developed by OpenAI [37] and enables human-like conversations in a dialogue format while allowing users to solve complex tasks without any background knowledge [11]. It can provide programming assistance with code generation, debugging, testing, and code reviews. While AI-based assistant applications were already introduced in the past [2, 4, 28], ChatGPT received high media attention [23]. This motivated even less tech-savvy users to familiarize themselves with the tool and test it for complex tasks [30, 39]. Within the first two months after its release in November 2022, it was already used by over 100 million users [39]. Therefore, we focused on ChatGPT in this work. In a simulated attack scenario, we demonstrate that the existing screening questions [16, 17] can be solved using ChatGPT despite the suggested

time limits. To address this issue, we investigated the following research question:

*Which questions can be used to screen out participants without any programming skills while meeting effectiveness, efficiency, and robustness against ChatGPT-based cheating requirements?*

We improved and extended existing screening questions fulfilling the same requirements as in [17] (effectiveness, efficiency, robustness against cheating, and language independence) and additionally addressing ChatGPT-resistance. To evaluate our screening questions, we conducted a study with 121 participants involving programmers recruited on Upwork.com and non-programmers recruited on Clickworker.com. Half of the participants were prompted to use ChatGPT to solve the tasks, while the other half received instructions without mentioning ChatGPT at all. Based on our results, we recommend six ChatGPT-resistant screening questions that can be used in software engineering and IT security studies to identify non-programmers. We also provide recommendations on setting up the screening instrument for different inclusion/exclusion criteria based on researchers' requirements.

## 2 RELATED WORK

This section summarizes past research on screening questions used for participant recruitment and ChatGPT demonstrating to be a powerful tool for solving complex tasks.

### 2.1 Screening Questions

Past research showed that recruiting professional programmers for studies is a challenging task [24, 48]. Researchers often were required to test for programming skills to ensure data quality. For example, Balebako et al. [7] used technical questions to check the participants' knowledge in a screening survey before inviting them for an interview study and a follow-up online survey. Acar et al. [1] asked participants to solve a short programming task before inviting them to a lab study. Assal and Chiasson [5] recruited professional programmers on Qualtrics for an online survey and required participants to show a baseline understanding by selecting correct security descriptions. Danilova et al. [18] added a code comprehension task to an online survey to filter out potential non-programmers. They noticed that 33 out of 129 participants failed to solve the task, although participants were recruited through a professional service with programming skills as a requirement to participate in their study.

The most related work to this study was conducted by Danilova et al. [16, 17]. In a user study with 17 computer science students and 33 professional programmers, Danilova et al. [17] explored screening questions that could be used to identify non-programmers. The best-performing tasks were tested with 52 programmers recruited on Clickworker.com. They found differences in performance compared to the ground truth of programmers, highlighting the need for effective screening questions even when using professional recruitment services. Finally, they tested the tasks in an attack scenario with 47 non-programmers, offering a €2 reward for each correctly solved task. Participants were explicitly allowed to use any source. The authors noticed that Google might be often used to solve the tasks. In a follow-up study, Danilova et al. [16] tested time limits for screening questions to address the googling issue. They first tested them with 74 computer science students by assigning three different time constraints to the groups: (1) "Base" group without any time constraints, (2) "No Limit" group encouraged to solve the tasks as quickly as possible, and (3) "Countdown" group with a strict timer. They found several tasks with time limits performed well. Similar to the previous study, an attack scenario was performed with 24 non-programmers using a "countdown" setup. The results showed that non-programmers failed to use sources like Google to solve the tasks. Finally, they recommended six screening questions with time limits to be used in surveys.

While the previous studies identified non-programmers using Google to solve the tasks to be the strongest attackers, the introduction of prominent AI-based solutions such as ChatGPT opened the field for new attack scenarios. In Section 3, we demonstrate that ChatGPT can solve all the recommended screening tasks from previous research despite using the recommended time limits.

### 2.2 ChatGPT

ChatGPT uses the advanced language model GPT-3.5, which is built upon the generative pre-trained transformer (GPT) architecture, involving a three-phase training approach [32]. While no vast specifications are published about GPT-3.5, the training concepts mentioned in the blog post of the official website [32] can be found in the literature (e.g., [44, 54]). ChatGPT is trained similarly to the previously released InstructGPT [38]. A pre-trained language model, such as GPT-3.0, is used as a starting point. In the initial phase, sample prompts are collected for which a labeler specifies the desired output behavior for this prompt. The data is used to fine-tune a supervised policy (SFT). In the second phase, several outputs are generated by the model for the same prompt. A labeler ranks the outputs from best to worst. This ranking is used to train the reward model (RM). In the final phase, the SFT is further fine-tuned using proximal policy optimization (PPO). PPO [44] is an optimization algorithm used in reinforcement learning, which updates the model's policy to maximize the reward obtained from the reward model. Phases two and three can be iterated by collecting more comparison data to update the reward model and fine-tune the policy.

Stack Overflow conducted a large user survey exploring developers' usage and perceptions of AI tools for software development [47]. The findings showed that developers have a generally positive view of AI tools and have used or plan to use them for software development in the future. Sobania et al. [45] showed that a bug-fixing rate of over 75% could be achieved in studies using ChatGPT. Further, past research found that students use ChatGPT for doing their homework tasks [9]. Therefore, researchers from non-engineering fields explored ChatGPT as an assisting tool to answer knowledge questions [13, 26, 49]. Choi et al. [13] tested ChatGPT on four law exams and found it could achieve a passing grade. Kung et al. [26] tested ChatGPT with the "United States Medical Licensing Exam" and found it performed almost well enough to pass the exam. Terwiesch [49] showed that ChatGPT could pass a "Wharton Master of Business Administration" exam. The fact that ChatGPT seems to be used for school and university homework indicates that the use of ChatGPT might increase in the near future. Due to its large

popularity and easy access via a web GUI [36], less tech-savvy users are equipped with a powerful tool.

## 3 THREAT OF CHATGPT

In this section, we present a Threat Model for the six screening questions suggested by Danilova et al. (see Table 2) and demonstrate their limitation in a simulated attack scenario.

### 3.1 Threat Model

As an attacker, we define a participant motivated to solve existing screening questions [16] by using ChatGPT despite lacking programming skills. In this case, the screening questions might be copied and pasted into the ChatGPT environment to produce an output that might be used to solve the tasks. First, we passed the six recommended screening questions to ChatGPT without providing the potential answer options. This procedure was repeated ten times per question. We analyzed each produced ChatGPT output and found ChatGPT provided correct answers to all the screening questions without requiring answer options or additional background knowledge.

Second, we estimated an attacker's *typing* speed to typewrite screening questions. Dhakal et al. [19] reported an average typing speed of 51.56 words per minute ($\sigma$ = 20.20). As a word is standardized as five *characters* or keystrokes when calculating words per minute, typing speed is translated to 257.8 characters per minute (cpm) on average. Thus, we conducted our calculation assuming a typing speed of 257.8 cpm. As ChatGPT is resistant to typos in the prompts [40, 51], we do not factor in the accuracy of the typing speed. Since ChatGPT did not require multiple-choice options to answer the screening tasks correctly, we assume an attacker passes the questions without the possible answer options to the AI tool. Note that this is a low estimate, as an attacker could still reduce the number of characters to increase efficiency. For example, typing "boolean" will result in ChatGPT providing a definition of a boolean, usually offering the attacker enough information within the first two sentences to solve the screening task correctly.

Third, we estimated the time ChatGPT required to *reply* to a screening question correctly. For this, we passed each screening question to ChatGPT ten times and measured the time ChatGPT needed to provide an answer. We based our calculation on the average time it took ChatGPT to print the reply. For our calculation, we had to repeat six out of 60 measurements where ChatGPT experienced a timeout and did not provide any answers.

Finally, we included *buffer* time. The attacker requires time to read the screening question, the ChatGPT output, and to select one answer option provided. We assumed the time spent reading the question was within the time allotted to typewrite the question to ChatGPT. Further, participants could read responses while ChatGPT produces them, requiring little additional time. Ojanpää et al. [31] conducted experiments on the time needed to find a word in a list of words. Their results showed that human visual search time increases linearly with word and list length. The question "Recursive Function" has the longest answer options compared to the time limit, making it the most challenging question to find the answer in the answer options. The answer options contain 210 characters (equivalent to 42 words). The worst performing participant in [31] would have needed 3.15 seconds to find the answer. Participants would likely perform better since all five answer options tested here begin with the same 13 characters. In addition, time pressure improves visual search [53]. Thus, we added five additional seconds as an upper limit for each screening question.

**Table 1: Simulated attack on the six recommended screening questions [16] with time in seconds.**

Typing = time for typing all characters, Reply = time ChatGPT needed to provide an answer, Buffer = time to select an answer

| Task | # Characters | Typing | Reply | Buffer | Total | Limit [16] |
|---|---|---|---|---|---|---|
| Boolean | 62 | 14.43 | 3.3 | 5 | 22.73 | 30 |
| Recursive Function | 73 | 16.99 | 4.4 | 5 | 26.39 | 30 |
| Websites | 75 | 17.46 | 4.9 | 5 | 27.36 | 30 |
| IDEs known | 52 | 12.10 | 5.5 | 5 | 22.6 | 60 |
| Array - read index | 94 | 21.88 | 3.1 | 5 | 30.04 | 60 |
| Prime check - purpose | 227 | 52.84 | 2.2 | 5 | 60.12 | 180 |

### 3.2 Simulated Attack on Existing Screening Questions

Using the threat model described above, we calculated the time an attacker using ChatGPT would need to solve the six multiple-choice tasks recommended by Danilova et al. [16]. We calculated the time an attacker would need to type the question into ChatGPT based on the number of characters in the question and the typing speed - without the screening question-answer options. We added the time that ChatGPT needs to answer the question and a buffer of five seconds. Table 1 displays the results of our analysis. The calculated times demonstrated that the existing screening questions could be solved with ChatGPT within the time limits suggested in [16] with a minimum difference of 2.64 and a maximum difference of 119.88 seconds. The results show that the existing screening questions might not withstand an attacker lacking programming knowledge using ChatGPT.

## 4 METHODOLOGY

We designed and evaluated ChatGPT-resistant screening questions to identify participants without programming skills. To test our screening questions, we compared the correctness rates of programmers to non-programmers. To test screeners' ChatGPT-resistance, we prompted half of the participants to use ChatGPT and compared their correctness rates to those not prompted for ChatGPT. Thus, we tested four groups: (1) Programmers not prompted to use ChatGPT (P-NC), (2) Programmers prompted to use ChatGPT (P-C), (3) Non-Programmers not prompted to use ChatGPT (NP-NC), and (4) Non-Programmers prompted to use ChatGPT (NP-C). While we defined P-NC as the control group, the NP-NC and NP-C groups simulated an attack scenario similar to Danilova et al. [16]. Participants were asked to use a laptop or PC to participate in our study. If prompted for ChatGPT usage, they were required to have a ChatGPT account.

At the beginning of the survey, participants were informed about the need to answer a series of programming screening questions within time limits. A total of 28 screening questions were asked, randomized in order and response options. The next question was only displayed when participants pressed "Next," allowing breaks between each question. Requiring participants to answer all 28

questions without breaks would likely have resulted in increased pressure and worse performance. After participants completed the screening questions, a follow-up survey asked for tool usage, specifically if participants used Google, ChatGPT, or other tools. We asked if they were already familiar with and had used ChatGPT before this study. If participants indicated to have used ChatGPT for the screening questions, we asked for their strategy, if they felt it was helpful, the number of questions it was used for, and their reasons for any strategy changes. Finally, we collected programmer-specific, non-programmer-specific, and general demographic information.

## 4.1 Instrument Requirements

We adapted the requirements for screening questions of Danilova et al. [16] by ChatGPT-resistance:

**1. Effectiveness**: The screening instrument should differentiate programmers from non-programmers by addressing the risk of retaining non-programmers (guessing correctly) and excluding non-programmers (one-time inattention).

**2. Efficiency**: The time required to complete the screening instrument should be as short as possible. Long screening processes were refused by participants [1]. Time limits should be set high enough to prevent non-programmers from finding the correct answer while avoiding pressure on programmers.

**3. Cheating Robustness**: The instrument should be designed so that it becomes difficult for participants without programming skills to come by the answers, for instance, by using online search engines or ChatGPT.

**4. Language Independence**: The screening instrument should be based on well-known programming concepts and code syntax rather than focusing on a specific programming language. Similar to Danilova et al. [16], we focused on imperative programming languages, such as C, C++, Java, or Python.

## 4.2 Screening Questions

We tested 18 knowledge and 10 code comprehension screening questions (see Table 2). While knowledge questions required participants to demonstrate their programming knowledge (e.g., of popular IDEs), comprehension questions required them to understand programming output. We adopted the recommended questions from Danilova et al. [16, 17]. We also included some questions that were not recommended but might be promising according to our applied ChatGPT-resistance strategies. Additionally, we developed new knowledge questions inspired by data structures and algorithms frequently asked in programming interviews [12]. We also created further comprehension questions inspired by challenging mathematical problems, such as [22], that might be promising in being ChatGPT resistant.

To prevent participants from copying and pasting screening questions into ChatGPT, the tasks could be displayed as images, requiring manual typewriting. Therefore, we shifted from a text-based to a visual format for knowledge questions. Instead of a lengthy text question part, we showed a visually illustrated programming-related concept. In particular, we replaced the description of a concept with the most basic visual representation (e.g., data structures) or a concrete example (e.g., IDE interface). Further, we obfuscated

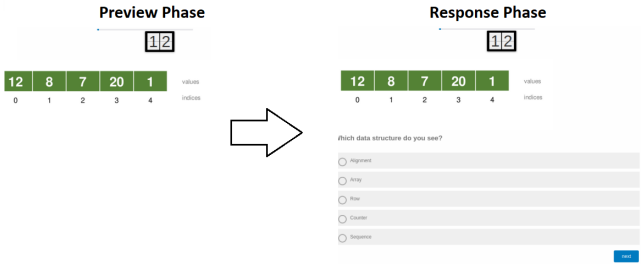


**Figure 1: Preview and Response Phases for Q9**

or removed any text not required to answer the questions correctly and utilized images to represent static concepts and graphics interchange format (GIF) for dynamic concepts. We avoided extended answer options. Instead, we used multiple-choice answer options with a maximum length of one or two words. Thus, non-programmers could not create meaningful ChatGPT prompts from images or simply copy&paste question text. Further, visual questions and one-word answer parts could be more accessible and quicker for participants with programming skills to process.

We divided the knowledge questions into two phases, as shown in Figure 1. In the *Preview Phase*, the visualization of a programming concept and a countdown were displayed without showing the according question and answer options. Thus, non-programmers were challenged to extract useful information, while programmers benefited from this extra time to think of possible answers. We set the time limits for the Preview Phase at 7-15 seconds, depending on the complexity of the image. Participants were forwarded to the next page after the expiration of the Preview Phase. In the follow-up *Response Phase*, a question about the visualization and the answer options were displayed. For the answer options, we included programming-related concepts (if possible). After a time limit of 12-15 seconds, depending on the number of answer options, participants were automatically forwarded to the next page regardless of whether an answer was chosen. Danilova et al. [16] showed that most participants solved their knowledge questions after ten seconds. These questions had a time limit of 30 seconds. Since we expected our visualizations to be easier to solve, we set lower time limits and tested them in a pilot study. Similar to [16], we chose five answer options for knowledge questions except in cases where more than one answer had to be selected (e.g., Q7 Languages).

Additionally, we developed short code comprehension questions. In contrast to Danilova et al. [16], we did not use a C-like pseudocode for our questions but a more simplified version similar to Python. Thus, the code syntax required fewer parentheses, making it more readable. We also used "while loops" instead of "for loops," since the syntax and semantics of while loops are more standardized across different programming languages (e.g., C in contrast to Python). This is because while loops are a more basic control flow construct than for loops. In addition, we omitted variable types in most code snippets since they were not needed to answer the questions. To achieve ChatGPT-resistance, we applied two strategies to comprehension questions (see Table 2). First, we wanted to mislead ChatGPT into producing wrong answers. For example, Q22 Backwards shows a function with a reversed string output. While ChatGPT recognizes the purpose of this function, it

**Table 2: Overview of the screening questions (Q1-Q28)**

| ID | Recommended Questions from [16] | Abbreviation | Answer Options | Category | Chat-GPT Resistance Strategy |
|---|---|---|---|---|---|
| Q1 | Which programming-related Webpage do you see? | Websites | Multiple Choice (5) | Knowledge (Information Source) | Visualization |
| Q2 | What concept about functions is implemented here? | Recursive | Multiple Choice (5) | Knowledge (Basic Knowledge) | Visualization |
| Q3 | What type of tool do you see? | IDE | Multiple Choice (5) | Knowledge (IDE Recognition) | Visualization |
| Q4 | Select the boolean value | Detect - Bool | Multiple Choice (12) | Knowledge (Programming Elements) | Visualization |
| Q5 | What does this code print? (0-indexed) | Array Index | Multiple Choice (8) | Comprehension (Array Output) | Code Bloating (Unreachable Code) |
| Q6 | This code tests if a number ... | Prime | Multiple Choice (8) | Comprehension (Prime Check) | Code Bloating (If-Else Nestings) |
| **ID** | **Additional Questions** | **Abbreviation** | **Answer Options** | **Category** | **Chat-GPT Resistance Strategy** |
| Q7 | Select 2 programming languages (logos) [16] | Languages | Multiple Choice (15) | Knowledge (Language Recognition) | Visualization |
| Q8 | What does the black box represent? [17] | Compiler | Multiple Choice (5) | Knowledge (Basic Knowledge) | Visualization |
| Q9 | Which data structure do you see? | Array | Multiple Choice (5) | Knowledge (Data Structure) | Visualization |
| Q10 | Which data structure do you see? | Stack | Multiple Choice (5) | Knowledge (Data Structure) | Visualization |
| Q11 | Which data structure do you see? | Queue | Multiple Choice (5) | Knowledge (Data Structure) | Visualization |
| Q12 | Which data structure do you see? | Linked List | Multiple Choice (5) | Knowledge (Data Structure) | Visualization |
| Q13 | Which data structure do you see? | Tree | Multiple Choice (5) | Knowledge (Data Structure) | Visualization |
| Q14 | What algorithm do you see? | DFS | Multiple Choice (5) | Knowledge (Algorithm) | Visualization |
| Q15 | What algorithm do you see? | BFS | Multiple Choice (5) | Knowledge (Algorithm) | Visualization |
| Q16 | What is the run time? [17] | Run Time | Multiple Choice (5) | Knowledge (Algorithm) | Visualization |
| Q17 | Select the string | Detect - String | Multiple Choice (12) | Knowledge (Programming Elements) | Visualization |
| Q18 | Select the parameter [16] | Detect - Parameter | Multiple Choice (12) | Knowledge (Programming Elements) | Visualization |
| Q19 | Select the function name | Detect - FunctionName | Multiple Choice (12) | Knowledge (Programming Elements) | Visualization |
| Q20 | Select the loop | Detect - Loop | Multiple Choice (12) | Knowledge (Programming Elements) | Visualization |
| Q21 | What does this code print? (0-indexed) [16] | Conditions | Multiple Choice (8) | Comprehension (If-Else Nestings) | Code Bloating (If-Else Nestings) |
| Q22 | What does this code print? (0-indexed) [17] | Backwards | Multiple Choice (8) | Comprehension (Reverse String) | Misleading ChatGPT |
| Q23 | What does this code print? (0-indexed) [16] | Palindrome | Multiple Choice (8) | Comprehension (Palindrome Check) | Misleading ChatGPT |
| Q24 | What does this code print? [16] | StringCount | Multiple Choice (8) | Comprehension (Compare two Strings) | Code Bloating (Unreachable Code) |
| Q25 | What problem for the function does the code in the red box create? [16] | EndlessLoop | Multiple Choice (8) | Comprehension (Fix Endless Loop) | Code Bloating (Unreachable Code) |
| Q26 | What does this code print? [22] | Euler Or | Multiple Choice (8) | Comprehension (Complex Formula) | Misleading ChatGPT |
| Q27 | What does this code print? | CrazyCalc | Multiple Choice (8) | Comprehension (Complex Formula) | Misleading ChatGPT |
| Q28 | Reorder the code snippets to calculate the frequency of „a“ in the string. | Reordering | Reordering (7) | Comprehension (Reordering Code Lines) | Drag&Drop Reordering |

frequently produces the wrong output if a complex term is used as an input, e.g., for the input "scissors," ChatGPT produces the correct answer in only 10% of all prompts. Second, to avoid typewriting in a reasonable time, screening questions' programming code was extended by statements that did not affect the purpose or output of the code, i.e., code bloating by adding unreachable code or if-else nesting. We used this approach when we could not modify the comprehension questions so that they mislead ChatGPT. Since we expected comprehension questions to be more challenging than knowledge questions for programmers, we increased the number of answer options from five to eight, reducing the chance of non-programmers guessing correctly. Based on our pilot study results, we set the time limit to 60 seconds. The detailed screening questions and a description of how code comprehension questions were designed to be unlikely solved by ChatGPT can be found in the supplementary material. In the following, we summarize the tested screening questions:

**Recommended questions (Q1-Q6).** We visualized Danilova et al.'s recommended know-how screening questions [16] instead of using text. In addition, we redesigned both recommended comprehension questions to explore code bloating and if-else nesting. The screening questions required identifying recursion, prominent developer websites, IDEs, a compiler, and a boolean. In addition, participants were asked to determine the purpose of a prime check function and the output of an array at a specified index.

**Promising questions (Q7-Q8).** We additionally tested two knowledge questions of Danilova et al. [16, 17] beyond the six recommended since we expected them to perform better when visualized.

**Data structures (Q9-Q13).** Data structures are fundamental programming concepts that are often asked during coding interviews [12]. We asked participants to identify commonly used representations of data structures. These data structures involved an Array, a Stack, a Queue, a Linked List, and a Tree. If necessary, we modified the visual representation to ensure that they could not be found using a Google search. For example, we used a vertical instead of a horizontal representation for a Queue.

**Algorithms (Q14-Q16).** Algorithms are another important programming concept, which is frequently asked during coding interviews [12]. We asked participants to identify a visual representation of depth-first-search (DFS) and breadth-first-search (BFS). In addition, we asked participants to indicate the runtime of iterating a predefined array since participants seemed familiar with arrays in the study by Danilova et al. [16]. We chose GIFs for these visualizations to clearly illustrate the step-by-step process of algorithms. In addition, we aligned them with standard visualizations for easy recognition by developers, essential for identifying the algorithm from the image alone during the preview phase.

**Programming elements (Q17-Q20).** Another recommended question [16] required identifying a boolean type from a list of values. Similarly, we asked participants to identify a boolean, a string, the function parameter, the function name, and a loop in a provided code snippet.

**Comprehension Questions (Q21-Q28).** For the comprehension questions, participants were tasked to determine the return value of a function presented as a code snippet, such as the value of an array at a given index, the return value of a block of nested if-else statements, a reversed string, or a palindrome check. The questions also asked to spot the mistake in a bubble sort function. Most of the chosen comprehension questions were modified versions of screening questions investigated in [16]. We reduced the complexity of the code snippets by using a python-style pseudo code and utilized our approaches in providing ChatGPT-resistance. In addition, we wanted to test comprehension questions which included more complex logical or arithmetic formula with which ChatGPT was known to struggle. Further, to test a different type of comprehension questions, we tasked participants to reorder code

lines by dragging and dropping them in the correct order for constructing a function that counts the occurrence of "a" in an input string.

## 4.3 Pilot Study

To test our study design, we conducted a pilot study with 12 participants recruited from personal contacts. Four non-programmers, including students from non-programming disciplines, solved only a few screening questions but did not report any comprehension issues. The programming groups consisted of four computer science students and four professional developers. We found that the time limits for some knowledge questions were too tight for both the preview and response phases. Thus, we increased the preview time from 7 seconds to 15 seconds for images containing a lot of information (e.g., Q7 Languages) and the response time from 12 seconds to 15 seconds for questions with more than five answer choices (e.g., Q3 IDE). In addition, we observed that the time limits for code comprehension questions were initially set too tight at 40 seconds, resulting in some unanswered questions. However, participants indicated that they missed only a few seconds. Thus, we increased the time limits to at least 60 seconds for all comprehension questions.

## 4.4 Participants

Based on previous research recommendations [24, 43], we recruited professional programmers for both programmer groups (P-C, P-NC) on Upwork.com. Upwork tracks records and allowed us to filter participants based on programmer characteristics. We ensured that all participants recruited on Upwork were likely to have programming skills without relying on a screening procedure as follows. We selected freelancers who listed popular imperative programming languages in their profiles and set Talent Quality to at least *Top Rated* [52]. Top Rated freelancers have a customer satisfaction score of at least 90%, a completed first project on Upwork over three months ago, an earning of at least $1000 in the last 12 months, an active status during the previous three months, and at least 100 billed hours. We invited randomly selected developers to our online survey. Of the recruited 63 professional programmers, we excluded seven participants from the P-C group because they did not try to use ChatGPT and additional five participants from the P-NC group due to failing attention checks and incomplete responses, leaving us with 26 participants for the P-NC group and 25 participants for the P-C group. The attention check question was included after the first 14 screening questions and followed the design of our knowledge questions, showing a visual representation of a neural net. Participants were asked to select the answer "machine learning." Participants' countries of residence and programming experience varied in both programmer groups (see Table 3). Participants recruited on Upwork received $20 for participating in our study.

Like Danilova et al. [16], we recruited participants without programming skills (NP-C, NP-NC) from Clickworker.com. We excluded participants who indicated having any programming skills at the beginning of the survey. In addition, we also asked for their programming self-assessment at the end of the study, implementing a double verification process. This helped us identify potential bots and inattentive or fraudulent participants. For the unprompted group (NP-NC), 194 participants started the survey. We excluded 126 participants due to not meeting the inclusion criteria, failing the attention check, or indicating using ChatGPT (20/126). From the remaining 68 participants, we removed 31 bots, leaving us with 37 participants. For the ChatGPT-prompted group (NP-C), 269 participants started the survey. We excluded 198 participants due to not meeting the inclusion criteria or failing the attention check. From the remaining 71 participants, we removed 34 bots and four participants indicating not having used ChatGPT, leaving us with 33 participants. Most Clickworker participants were from Germany, likely due to Clickworker being a German company (see Table 3). We followed the payout scheme of Danilova et al. [16] to incentives non-programmers to solve the questions. However, we increased the base pay because we expected participants to answer very few questions correctly, and we wanted to ensure that the payout reflected the minimum wage. In addition, we reduced the reward for each correct screening question because we tested more questions with lower time limits compared to Danilova et al. Participants received €5 as compensation for completing the survey and an additional €0.5 for each question answered correctly, resulting in a maximum payment of €19.

## 4.5 ChatGPT Experience

To avoid ChatGPT priming of non-prompted participants, we asked for ChatGPT experience in the follow-up survey. Almost all participants have already heard of ChatGPT (96% in P-NC, 100% in P-C, 89% in NP-NC, and 97% in NP-C). Before participating in our study, ChatGPT was reported to be used by 84.6% in the P-NC group, 92% in the P-C group, 29.7% in the NP-NC group, and 84.8% in the NP-C group. Programmers reported sending over 50 ChatGPT prompts, while non-programmers fewer than 19 prompts. Interestingly, 28% of the programmers (P-C) and 9.1% of the non-programmers (NP-C) reported having a "ChatGPT Plus" subscription.

## 4.6 Evaluation

We followed the approach of Danilova et al. [16, 17] to compare the success rates of correctly answering the screening questions between two groups using the Fisher's exact test [21, p. 816]. We considered questions with at least a 90% correctness rate for programmers and, at most, 40% correctness rate for non-programmers. Since we designed our screening questions with shorter time limits than [16], we chose thresholds that enabled us to ask more questions in the same time frame by allowing developers to make some mistakes while keeping the screening procedure short and straightforward. Thus, we set the proportions p1 and p2 for the Fisher's exact test at 0.9 and 0.4, respectively. We conducted a power analysis using G*Power [20], which indicated a sample size of 24 participants for each group using the two-sided Fisher's exact test. Using Bonferroni-Holm [21, p. 428], we corrected the significance level by a factor of two since we tested group P-NC twice for each screening question.

We used inductive coding for the qualitative analysis [50] of the open-ended questions in our follow-up survey. After two researchers independently coded answers, they merged their sets of codes. Due to the short and straightforward answers, all the discrepancies could be resolved by discussion. While the primary purpose

**Table 3: Demographics of the Participants (n = 121)**

| | *Programmers (Upwork)* | | *Non-Programmers (Clickworker)* | |
|---|---|---|---|---|
| | **Programmers Non-ChatGPT (P-NC) (n=26)** | **Programmers ChatGPT (P-C) (n=25)** | **Non-Programmers Non-ChatGPT (NP-NC) (n=37)** | **Non-Programmers ChatGPT (NP-C) (n=33)** |
| **Prompt** | "Please solve the tasks!" | "Please use ChatGPT for solving these tasks if possible." | "Please solve the tasks!" | "Please use ChatGPT for solving these tasks if possible." |
| **Gender** | male: 24, female: 2, other: 0 | male: 25, female: 0, other: 0 | male: 24, female: 12, other: 1 | male: 21, female: 11, other: 1 |
| **Age** | min: 22, max: 51, mean: 34.04, md: 32.0, sd: 8.64 | min: 18, max: 47, mean: 28.44, md: 27, sd: 6.98 | min: 22, max: 72, mean: 41.7, md: 41, sd: 11.73 | min: 23, max: 52, mean: 35.88, md: 36, sd: 8.87 |
| **Country of Residence** | Pakistan: 4, Serbia: 4, India: 3, Vietnam: 3, Others: 12 | Pakistan: 6, Bangladesh: 3, Nigeria: 2, Turkey: 2, India: 2, Others: 10 | Germany: 20, Italy: 2, USA: 4, India: 2, Others: 9 | Germany: 15, USA: 4, India: 2, Brazil: 2, Kenya: 2, Others: 8 |
| **Highest Education** | Bachelor: 13, Master: 9, Others: 4 | Bachelor: 17, Master: 2, Others: 6 | Bachelor: 9, Master: 13, Others: 15 | Bachelor: 12, Master: 4, Others: 17 |
| **ChatGPT Experience** | Yes: 22, No: 4 | Yes: 23, No: 2 | Yes: 11, No: 26 | Yes: 28, No: 5 |
| **Self-rated Programming Experience** | Beginner: 0, Advanced Beginner: 0, Intermediate: 5, Advanced: 15, Expert: 6 | Beginner: 1, Advanced Beginner: 0, Intermediate: 9, Advanced: 11, Expert: 4 | No experience at all: 37 | No experience at all: 33 |
| **General Programming Experience [years]** | min: 3, max: 35, mean: 13.19, md: 10.0, sd: 8.69 | min: 1, max: 25, mean: 7.72, md: 7, sd: 5.58 | - | - |
| **Programming Experience in the Job [years]** | min: 2, max: 20, mean: 8.19, md: 6.0, sd: 4.78 | min: 0, max: 13, mean: 4.16, md: 3, sd: 3.56 | - | - |

of qualitative analysis is to explore a phenomenon in depth, we additionally report how many participants mentioned approaches, concepts, and themes to indicate their frequency and distribution.

## 5 ETHICS

Our project was examined and approved by the university's institutional review board (IRB). Participants received a consent form at the beginning of the study describing the scope of the study, data use, and retention policies. We complied with the General Data Protection Regulation (GDPR). Participants were informed that they could withdraw their data during or after the study without facing any negative consequences. In addition, participants were asked to save the consent form for their use.

## 6 LIMITATIONS

The following study limitations need to be considered when interpreting the results. First, the recruitment of non-programmers from Clickworker lacks effective filtering mechanisms, which raises the possibility of fraudulent participants, including programmers posing as non-programmers. Although efforts (e.g., double verification) have been made to detect such participants, some individuals may still have been undetected. The correctness rates of six NP-C participants were above 50%, indicating they might have some programming skills, although stating otherwise. Second, for the first seven participants in the P-NC group, the Qualtrics servers experienced some latency issues. This resulted in participants experiencing delays in displaying the images while the timer was already running. After we noticed this issue, we stopped the study until the latency issues were resolved by Qualtrics (fixed within 24 hours). Luckily for us, this issue only affected P-NC participants leading even to the underreported effectiveness of our screening questions. Third, differences in compensation between programmers and non-programmers could result in inequalities in motivation, potentially influencing the study's outcome. Fourth, since prompted groups used ChatGPT only for a few questions and non-prompted participants did not use ChatGPT at all, we could not determine the impact on each question. Finally, while we provide first insights into how to challenge AI-based tools such as ChatGPT, other existing tools and screening questions not considered in this work have to be explored by future research.

**Table 4: Percentages of correct answers in each group.**

| ID | Question Name | P-NC | P-C | NP-NC | NP-C |
|---|---|---|---|---|---|
| Q9 | Array | 100.0% | 88.0% | 24.3% | 42.4% |
| Q7 | Languages | 100.0% | 88.0% | 35.1% | 39.4% |
| Q3 | IDE | 96.2% | 88.0% | 27.0% | 39.4% |
| Q2 | Recursive | 96.2% | 92.0% | 16.2% | 33.3% |
| Q8 | Compiler | 96.2% | 100.0% | 13.5% | 27.3% |
| Q13 | Tree | 92.3% | 84.0% | 29.7% | 63.6% |
| Q10 | Stack | 92.3% | 92.0% | 37.8% | 51.5% |
| Q20 | Detect - Loop | 92.3% | 96.0% | 13.5% | 30.3% |
| Q19 | Detect - FunctionName | 92.3% | 76.0% | 24.3% | 18.2% |
| Q1 | Websites | 92.3% | 96.0% | 18.9% | 24.2% |
| Q11 | Queue | 92.3% | 88.0% | 16.2% | 21.2% |
| Q14 | DFS | 84.6% | 64.0% | 27.0% | 39.4% |
| Q18 | Detect - Parameter | 84.6% | 80.0% | 2.7% | 15.2% |
| Q5 | Array Index | 84.6% | 64.0% | 2.7% | 9.1% |
| Q22 | Backwards | 84.6% | 52.0% | 5.4% | 6.1% |
| Q12 | Linked List | 80.8% | 88.0% | 24.3% | 24.2% |
| Q25 | EndlessLoop | 76.9% | 60.0% | 24.3% | 24.2% |
| Q17 | Detect - String | 73.1% | 64.0% | 5.4% | 30.3% |
| Q24 | StringCount | 73.1% | 56.0% | 18.9% | 21.2% |
| Q16 | Run Time | 69.2% | 64.0% | 8.1% | 12.1% |
| Q15 | BFS | 65.4% | 68.0% | 21.6% | 12.1% |
| Q28 | Reordering | 65.4% | 44.0% | 2.7% | 9.1% |
| Q23 | Palindrome | 57.7% | 32.0% | 21.6% | 21.2% |
| Q21 | Conditions | 46.2% | 20.0% | 0.0% | 12.1% |
| Q4 | Detect - Bool | 42.3% | 48.0% | 8.1% | 24.2% |
| Q6 | Prime | 42.3% | 52.0% | 2.7% | 12.1% |
| Q27 | CrazyCalc | 38.5% | 12.0% | 8.1% | 9.1% |
| Q26 | Euler Or | 15.4% | 12.0% | 32.4% | 18.2% |

## 7 RESULTS

In this section, we present the evaluation of our screening questions and participants' strategies to solve the tasks.

**Table 5: Percentages of correct answers in each non-prompted group. Pre = Preview Phase, Res = Response Phase**

| Topic | P-NC | NP-NC | Time Limit |
|---|---|---|---|
| Array | 100.0% | 24.3% | [Pre]: 7 s, [Res] 12 s |
| Languages | 100.0% | 35.1% | [Pre]: 15 s, [Res] 12 s |
| Compiler | 96.2% | 13.5% | [Pre]: 7 s, [Res] 12 s |
| Recursive | 96.2% | 16.2% | [Pre]: 7 s, [Res] 12 s |
| IDE | 96.2% | 27.0% | [Pre]: 7 s, [Res] 15 s |
| Detect - Loop | 92.3% | 13.5% | [Pre]: 15 s, [Res] 12 s |
| Queue | 92.3% | 16.2% | [Pre]: 7 s, [Res] 12 s |
| Websites | 92.3% | 18.9% | [Pre]: 7 s, [Res] 12 s |
| Detect - FunctionName | 92.3% | 24.3% | [Pre]: 15 s, [Res] 12 s |
| Tree | 92.3% | 29.7% | [Pre]: 7 s, [Res] 12 s |

## 7.1 Effectiveness of screening questions

Table 4 shows the correctness rates of every group for each of the 28 screening questions. We consider questions to be effective if non-prompted programmers (P-NC) achieved a correctness rate of at least 92.3% and non-programmers (NP-NC) a rate of at most 35.1%. This range included all questions that fall within the scope of our power analysis. We chose to use stricter thresholds compared to our power analysis to keep the number of questions required to screen participants effectively small. Ten knowledge questions fulfilled our requirements (see Table 5). Notably, this includes three of the questions recommended by Danilova et al. [16] (Q1 Websites, Q2 Recursive, and Q3 IDE). Our results suggested that using ChatGPT was more difficult with visual concept questions than with existing screening questions. Except for question Q26 (Euler Or), the programmers without ChatGPT (P-NC) were always more successful than both groups of non-programmers with and without ChatGPT (NP-C, NP-NC). Concerning the comprehension questions, the probability of guessing the answer correctly is 12.5% due to eight answer options. Most non-programmer correctness rates varied around this value. The highest success rate of 84.6% for comprehension questions was achieved by Array Index (Q5) and Backwards (Q22).

To evaluate the effectiveness of our screening questions, we used Fisher's exact test to investigate the differences in correctness rates between the programmer group without ChatGPT (P-NC) and both non-programmer groups (NP-NC, NP-C). We only considered questions with correctness rates falling within the scope of our power analysis. All p-values (<0.001) were below the significance level. We refer to the supplementary material for additional statistics, such as odds ratios.

## 7.2 Participants' time pressure perceptions

Participants were asked if they felt time pressure. Table 6 shows how many participants in each group felt a certain level of time pressure. The majority of all groups felt some degree of time pressure. Non-programmers using ChatGPT (NP-C) had the highest percentage (63.6%) of feeling pressured. In contrast, programmers without ChatGPT had the smallest percentage (7.7%) of high pressure, and 42.3% felt no pressure. Overall, more programmers felt no pressure compared to non-programmers. ChatGPT's usage increased the time pressure for programmers and non-programmers. Except for NP-C, about half of the participants felt less pressured. We also asked participants how they felt about the knowledge questions under time pressure. The majority of programmers (65.4%) who did not use ChatGPT stated that they did not feel pressure at all. About one-third felt little pressure, and none of these participants felt very high or extreme time pressure. From the other three groups, some participants felt very high or even extreme time pressure. Thus, a trend towards more pressure for non-programmers or when using ChatGPT was observed. We also asked participants how they felt about comprehension questions under time pressure. Time pressure increased significantly for all groups. Most ChatGPT-prompted participants felt great or extreme pressure.

**Table 6: Perceived time pressure**

| | P-NC | P-C | NP-NC | NP-C |
|---|---|---|---|---|
| **Time Pressure - General** | | | | |
| I did not feel pressured at all | 42.3% | 16% | 21.6% | 3% |
| I felt a little pressured | 50% | 52% | 54.1% | 33.3% |
| I felt very pressured | 7.7% | 32% | 24.3% | 63.6% |
| **Time Pressure - Knowledge Questions** | | | | |
| I did not feel pressured at all | 65.4% | 60% | 16.2% | 9.1 |
| I felt a little pressured | 19.2% | 16% | 51.4% | 18.2 |
| I felt somewhat pressured | 15.4% | 20% | 18.9% | 21.2 |
| I felt very pressured | - | 4% | 13.5% | 24.2 |
| I felt extremely pressured | - | - | - | 27.3 |
| **Time Pressure - Comprehension Questions** | | | | |
| I did not feel pressured at all | 23.1% | 16% | 18.9% | 6.1% |
| I felt a little pressured | 30.8% | 12% | 24.3% | 9.1% |
| I felt somewhat pressured | 26.9% | 16% | 24.3% | 15.2% |
| I felt very pressured | 15.4% | 40% | 29.7% | 33.3% |
| I felt extremely pressured | 3.8% | 16% | 2.7% | 36.4% |

**Table 7: Participants using programming skills to solve the questions**

| Number of solved questions | P-NC | P-C |
|---|---|---|
| 28 (all) | 96.2% | 48% |
| 21-27 | 0% | 0% |
| 11-20 | 3.8% | 4% |
| 6-10 | 0% | 16% |
| 1-5 | 0% | 24% |
| 0 | 0% | 8% |

## 7.3 Screening with Programming Skills

Table 7 shows how many participants of both programmer groups indicated using their programming skills to solve the questions. While almost all participants (96.2%) from the P-NC group indicated to have used their programming skills, 48% of the programmers prompted for ChatGPT (P-C) reported using their programming skills to solve less than ten questions.

## 7.4 Screening with ChatGPT

Table 8 shows that most of the prompted programmers and non-programmers indicated to have used ChatGPT for only 1-5 questions. 72% of participants in group P-C and 60.6% of participants in group NP-C made fewer than five attempts to solve the screening questions using ChatGPT. The number of participants who

**Table 8: Participants using ChatGPT to solve the questions**

| Number of solved questions | P-C | NP-C |
|---|---|---|
| 28 (all) | 16% | 24.2% |
| 21-27 | 0% | 0% |
| 11-20 | 4% | 3% |
| 6-10 | 8% | 12.1% |
| 1-5 | 72% | 60.6% |
| 0 | 0% | 0% |

**Table 9: Strategies used with ChatGPT**

| Primary Strategy | P-C | NP-C |
|---|---|---|
| General lack of time | 16% | 45.5% |
| Typewriting | 20% | 21.1% |
| Using keywords | 16% | 18.2% |
| Attempts to copy code | 12% | 9.1% |
| Information extraction | 20% | 3% |
| Answer verification | 8% | 0% |
| Advanced prompting | 4% | 0% |
| No strategy mentioned | 4% | 3% |

attempted to answer 6-10 questions with ChatGPT decreased; similarly, for 11-20 and 21-27 questions, only 16% prompted programmers and 24.2% prompted non-programmers attempted to solve all questions using ChatGPT.

**ChatGPT - Strategy:** Participants were asked to briefly describe their approach when using ChatGPT. We received 58 responses from 25 prompted programmers and 33 non-programmers. An overview of the responses can be found in Table 9. Nineteen participants did not specify a strategy but mentioned a lack of time, such as the inability to formulate a reasonable prompt in time. Twelve participants reported trying to type the questions into the ChatGPT environment for questions displayed as images. Six of them acknowledged that there was not enough time to typewrite the whole question. Ten participants described trying to simplify the content of the question instead of typewriting every character. Programmers reported describing the visuals or the concept they recognized and asked follow-up questions. They also described trying to understand the input and context and then transferring it to ChatGPT. Many non-programmers noted they passed keywords or main phrases to ChatGPT. Participants also mentioned trying to keep the input as short as possible to save time. Six participants indicated they planned to copy the questions, but it did not work. Some of them noted that they either changed their strategy to typewriting or didn't try typewriting at all. Another six participants attempted to use a tool to extract some information from the image. Five of which were programmers. One participant tried to enter an image link as input to ChatGPT. Another used a text extractor and wanted to let ChatGPT evaluate the generated text. Participants also mentioned using the tool Capture2Text [14] and Optical Character Recognition (OCR) for text extraction to copy it to ChatGPT. The non-programmers utilized OCR in combination with ChatGPT to answer the screening questions. Two programmers stated that they only used ChatGPT to verify their answers. One of them claimed they trusted their programming skills instead of ChatGPT if there was insufficient time. Another programmer also explicitly stated that they instructed ChatGPT to give short answers (advanced prompting) to speed up the process. Most non-programmers did not know about advanced prompting (84.8%), while 44% of the programmers used it.

**Table 10: Participants' perceived support of ChatGPT**

| ChatGPT supported me | P-C | NP-C |
|---|---|---|
| Strongly disagree | 56% | 39.4% |
| Somewhat disagree | - | 9.1% |
| Neither agree nor disagree | 8% | 24.2% |
| Somewhat agree | 36% | 21.2% |
| Strongly agree | - | 6.1% |

**Table 11: Tools used to answer questions except ChatGPT**

| Other tools used | P-NC | P-C | NP-NC | NP-C |
|---|---|---|---|---|
| No Tool / ChatGPT | 100% | 56% | 94.6% | 69.7% |
| Google | - | 28% | 5.4% | 27.3% |
| Others | - | 16% | - | 3% |

**ChatGPT - Strategy Switch:** If participants indicated to have switched their strategy from using ChatGPT to other options, we asked them why they had changed it. We received 43 responses. Most participants (32) switched due to a lack of time. Twelve participants said they could not transfer the questions to ChatGPT in time. One participant noted that it took too long to wait for the answer from ChatGPT. Two programmers pointed out that OCR tools could not properly extract the information from images and that there was not enough time for syntax correction, which led to ChatGPT being unable to answer. Eleven participants switched strategies because they could not copy-paste the question text since it was displayed as an image. Two non-programmers changed their approach because the ChatGPT answers confused them, and two programmers stated that some of the questions were easy enough to solve with their programming skills.

**ChatGPT - Support:** The level of perceived support ChatGPT provided in answering the screening questions is shown in Table 10. More than half of the programmers indicated that ChatGPT did not assist well, while 36% considered ChatGPT somewhat useful. 27.3% of the non-programmers considered ChatGPT at least somewhat helpful in answering the screening questions, while 48.5% did not perceive ChatGPT as useful.

## 7.5 Screening with other tools

Table 11 shows which tools other than ChatGPT were used to solve the tasks. Groups not prompted to use ChatGPT often indicated not using tools to solve the screeners. Over half of the participants who used ChatGPT did not use other tools. Still, almost a third of them also used Google as the most popular alternative besides Bing AI [29], PowerToys [27], an IDE, and other OCR tools or plugins. Eighteen participants using Google indicated to have searched for key concepts related to the questions or contextual information. Some participants mentioned time constraints or lack of speed as reasons for relying more on Google than ChatGPT. Only a few non-programmers (5.4%) reported to have used Google. However, these participants had a high rate of incorrect responses (on average, 25 of 28 questions were answered incorrectly), suggesting that Google was not an effective tool. One of the two participants using Bing AI said that Bing AI was faster than ChatGPT.

# 8 DISCUSSION AND RECOMMENDATIONS

Our results suggested that many prompted programmers and non-programmers gave up using ChatGPT after five screening questions. A possible explanation for the limited use of ChatGPT is that most participants did not perceive it as valuable support. A large proportion of both programmers (56%) and non-programmers (39.4%) strongly disagreed with the statement that ChatGPT was helpful. Only a small percentage of programmers (36%) and non-programmers (27.3%) found ChatGPT to be at least somewhat helpful. Of the 27.3% non-programmers who attempted to answer all questions with ChatGPT, only five of the 28 tested questions were answered correctly on average. Many participants mentioned trying to copy and paste the questions as a strategy to find the correct answer. However, this was not possible since our questions consisted of code screenshots or visual images. Participants tried to typewrite the questions or transfer keywords to ChatGPT. However, a lack of time was mentioned, especially while transferring information. Comparing the overall time pressure of programmers with and without ChatGPT for our knowledge questions, the groups that (at least partially) used ChatGPT felt more pressured. Using ChatGPT resulted in an even more considerable time pressure increase for non-programmers than programmers. We assume that considering a suitable strategy while transmitting the information to ChatGPT caused a lot of pressure. This effect may be even more vital for non-programmers because they were less experienced with ChatGPT. However, both groups, P-C and NP-C, behaved very similarly in terms of perceived support of ChatGPT, the number of questions attempted with ChatGPT, and the usage of other tools, suggesting that the difference in familiarity with ChatGPT did not affect the performance when trying to solve the screening questions. Overall, most programmers switched to their programming skills, while non-programmers resorted to guessing to solve the screening questions.

## 8.1 Six ChatGPT-resistant Screening Questions

Based on our results, we recommend six screening questions (see Table 12) fulfilling the correctness threshold of 92.3% for programmers (P-NC) and 33.3% for non-programmers from our attack scenario. We opted for stricter thresholds compared to our power analysis to minimize the questions required to screen participants effectively. Thus, we excluded Q3 and Q7 since they would require more questions for screening, reduce the number of non-programmers rejected, or reduce the number of programmers included. Setting less strict thresholds would come at the cost of either effectiveness or efficiency. We refer to the NP-C group since non-programmers using ChatGPT almost always performed better than those not prompted for ChatGPT (NP-NC). All recommended screening questions are knowledge questions, visualizing well-known programming concepts. These questions performed well even with short time limits, and most programmers felt no time pressure while answering them. Some of the recommended questions (e.g., Q8 Compiler) do not contain any text, making it challenging to formulate a meaningful prompt to send to ChatGPT. Other questions (e.g., Q19 Detect - FunctionName) included too much code to be typed. We tested the ChatGPT plugin ChatOCR to extract any text from the image, which we could use as a prompt for ChatGPT. However, the extraction process took much longer than the time limit, and the text output was syntactically incorrect. This resulted in ChatGPT being unable to provide an answer. While knowledge questions may not necessarily require programming skills, unlike comprehension questions, they can be used to screen participants for programming studies effectively. We suggest extending the pool of knowledge questions by visualizing other well-known concepts.

**Table 12: Six recommended ChatGPT-resistant screening questions for programming skills. Pre = Preview Phase, Res = Response Phase**

| Topic | P-NC | NP-C | Time Limit |
|---|---|---|---|
| Compiler | 96.2% | 27.3% | [Pre]: 7 s, [Res]: 12 s |
| Recursive | 96.2% | 33.3% | [Pre]: 7 s, [Res]: 12 s |
| Detect - FunctionName | 92.3% | 18.2% | [Pre]: 15 s, [Res]: 12 s |
| Queue | 92.3% | 21.2% | [Pre]: 7 s, [Res]: 12 s |
| Websites | 92.3% | 24.2% | [Pre]: 7 s, [Res]: 12 s |
| Detect - Loop | 92.3% | 30.3% | [Pre]: 15 s, [Res]: 12 s |

Similar to previous findings [16], comprehension questions were ineffective in identifying non-programmers. In addition, programmers felt much more time pressure compared to knowledge questions. The original prime question [16] might have performed well since computer science students were recruited as participants who are more familiar with such concepts. Our results suggested that this question might be less effective beyond the university context. For future studies, we suggest testing more comprehension questions for which ChatGPT likely provides the wrong answer. This would allow for increasing the time limit and remove any need for code bloating, which might have put too much time pressure on programmers. However, this might result in a longer screening time.

## 8.2 Setup of the screening instrument

Based on researchers' requirements, the six recommended questions can be used with different levels for including programmers and excluding non-programmers. For the following setups, we average the correctness rates of the six recommended questions considering programmers (P-NC) and non-programmers (NP-C). Table 13 illustrates the inclusion of programmers and the exclusion of non-programmers in the study. The first column specifies the minimum number of questions that must be answered correctly out of the total number of questions asked. The number of questions in a screening instrument and the accuracy of the answers required depends on the goal of participant recruitment. If recruiting programmers is challenging, the screening goal may be to include as many programmers as possible while allowing more non-programmers.

If at least four out of six questions are answered correctly, it is possible to include 99.55% of programmers while excluding 95.83% of non-programmers. The difference between the two groups would also be greatest in this case. The included non-programmers might be identified by using additional measures. Another objective might be to minimize the inclusion of non-programmers in the study, even if it excludes a higher number of programmers. Increasing the number of consecutive questions makes it increasingly difficult for non-programmers to provide correct answers consistently. With five correct questions out of six, 99.4% of the non-programmers can

be excluded, while 94.83% of the programmers are still included. Asking all six questions without breaks would take, at most, 130 seconds.

While Danilova et al. [17] designed their screening questions to be used in online survey studies specifically, we recommend deploying a screening procedure in all studies where there is a lack of direct interaction between participants and researchers.

## 8.3 OCR-Protection

After this study was performed, ChatGPT-4 was released on September 24th 2023 [33], offering image processing by advancing beyond the OCR features of the Link Reader plugin. This improvement allows it to accurately and swiftly decode text and programming code from images, a task at which the Link Reader was less proficient. However, using the GPT-4 model is tied to a paid monthly subscription and a limit of 40 messages every three hours.

In the knowledge question preview phase, the six recommended screening questions remain effective for identifying non-programmers, as they have limited time to capture and submit screenshots. Leaving out the question-and-answer choices from these screenshots often leads to verbose or incorrect responses from ChatGPT. For instance, in the case of Q11 Queue, omitting the answer options leads ChatGPT to misinterpret a queue as a stack, thus not benefiting attackers in answering knowledge questions correctly during the preview phase and potentially misleading them. Ambiguity in visual concepts might plead to incorrect responses from ChatGPT in the preview phase. Therefore, non-programmers would require the response phase to capture all necessary information to answer the knowledge question accurately and quickly. However, the cumulative time for taking a screenshot, saving it, uploading it to ChatGPT, waiting for its response, locating the answer in the output, and then finding the correct response option in the screening process is likely to exceed the time limit of the response phase. Also for the comprehension questions, users must submit detailed screenshots encompassing the image, question, and answer options to get accurate responses from ChatGPT. ChatGPT typically provides correct answers but tends to include extensive explanations, requiring users to either sift through the text for the correct answer or use specific prompts for succinct responses. Thus, we also expect our recommended comprehension questions to be effective within the suggested time frames. Since December 1st 2023, GPT-4 only analyzes the first frame of a GIF [34]. Displaying a static image as a GIF while leaving the first frame empty results in ChatGPT being unable to detect its content. To prevent a non-programmer from taking a screenshot of the later frames, the GIF could be designed in a way that each frame only includes partial information about the displayed concept, question, or response options, making it impossible to transfer the image to ChatGPT. Still, more research is needed to explore the effectiveness of the screeners for the paid version of ChatGPT.

As ChatGPT and other large language models (LLMs) evolve, future research might focus on methods to deliberately mislead AI into extracting incorrect information and providing wrong or no answers to non-programmers. One promising method is perturbation, a technique that subtly alters images. While humans can still understand them, AI might misread words or characters or interpret them as opposites, thus distorting the question [10, 46]. This technique might obscure programming code, question text, and answer options. To counteract an LLM's improved ability to interpret visual concepts in knowledge questions, a similar approach could be used. Instead of editing existing text in an image, additional information could be embedded in an image invisible to humans but detectable by an AI. This additional information could distract the AI from the correct answer, overwhelm it with irrelevant content, or overwrite the associated prompt, changing the AI's behavior. While some research exists (e.g., [6]), these approaches were not extensively tested with GPT-4 but applied to models with similar capabilities. Future research might also consider other LLMs.

**Table 13: No. of questions for including programmers and excluding non-programmers for different screening setups**

| Correct/Total | P-NC | NP-C | Difference |
|---|---|---|---|
| 2/2: | 87.61% | 6.63% | 80.89% |
| 3/3: | 82.00% | 1.71% | 80.29% |
| 3/4: | 97.75% | 5.51% | 92.24% |
| 4/4: | 76.75% | 0.44% | 76.31% |
| 4/5: | 96.40% | 1.75% | 94.65% |
| 5/5: | 71.84% | 0.11% | 71.73% |
| 4/6: | 99.55% | 4.17% | 95.38% |
| 5/6: | 94.83% | 0.53% | 94.30% |
| 6/6: | 67.24% | 0.03% | 67.21% |

# 9 CONCLUSION

In software engineering and IT security studies with professional programmers, it is essential to filter out participants without programming skills to improve the validity of the study results. While online platforms provide a convenient way of recruiting programmers, researchers should not rely on self-reported programming skills [24, 48]. While screening questions for developer studies exist, they could be easily solved using ChatGPT. Thus, we developed ChatGPT-resistant screening questions consisting of visualizations of well-known programming concepts. These illustrations are easily recognized by participants with programming skills, while participants without programming skills struggled using ChatGPT to answer the screening questions correctly. In addition, most participants prompted to use ChatGPT felt pressured and quickly switched to other strategies for solving the tasks. We recommended six screening questions for identifying non-programmers while being ChatGPT-resistant. We discussed their effective setup as a screening instrument. For future studies on screening questions for developer studies, we suggest further investigating visual concepts, as these seem to perform well. While this study was conducted only a few months after the introduction of ChatGPT, we expect the popularity and usage of AI-based tools to increase in the future. Therefore, we call for more research on the implications of such tools.

# ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

# REFERENCES


[1] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2016. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP)*. 289–305. https://doi.org/10.1109/SP.2016.25

[2] Amazon. n.d.. Alexa. Retrieved July 2023 from https://developer.amazon.com/

[3] Amazon. n.d.. Amazon Mechanical Turk. Retrieved July 2023 from https://www.mturk.com/

[4] Apple Inc. n.d.. Siri. Retrieved July 2023 from https://www.apple.com/siri/

[5] Hala Assal and Sonia Chiasson. 2019. 'Think Secure from the Beginning': A Survey with Software Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300519

[6] Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. 2023. Abusing Images and Sounds for Indirect Instruction Injection in Multi-Modal LLMs. arXiv:2307.10490 [cs.CR]

[7] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason Hong, and Lorrie Cranor. 2014. The Privacy and Security Behaviors of Smartphone App Developers. https://doi.org/10.14722/usec.2014.23006

[8] Sebastian Baltes and Stephan Diehl. 2016. Worse Than Spam: Issues In Sampling Software Developers. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Ciudad Real, Spain) *(ESEM '16)*. Association for Computing Machinery, New York, NY, USA, Article 52, 6 pages. https://doi.org/10.1145/2961111.2962628

[9] Chetana Belagere. 2023. Students have started using ChatGPT to cheat in assignments, tests. How are professors catching them? https://thesouthfirst.com/karnataka/students-have-started-using-chatgpt-to-cheat-in-tests-exams-how-are-professors-catching-them/.

[10] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad Characters: Imperceptible NLP Attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. 1987–2004. https://doi.org/10.1109/SP46214.2022.9833641

[11] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712 [cs.CL]

[12] M. Chinmoy. 2016. *Cracking the Coding Interview: 189 Programming Questions and Solutions*. Blurb.

[13] Jonathan H Choi, Kristin E Hickman, Amy Monahan, and Daniel Schwarcz. 2023. ChatGPT goes to law school. *Available at SSRN* (2023).

[14] Christopher Brochtrup. n.d.. Capture2Text. Retrieved July 2023 from https://capture2text.sourceforge.net/

[15] Clickworker GmbH. n.d.. Clickworker. Retrieved July 2023 from https://www.clickworker.com/

[16] Anastasia Danilova, Stefan Horstmann, Matthew Smith, and Alena Naiakshina. 2022. Testing Time Limits in Screener Questions for Online Surveys with Programmers. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 2080–2090. https://doi.org/10.1145/3510003.3510223

[17] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do You Really Code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *Proceedings of the 43rd International Conference on Software Engineering* (Madrid, Spain) *(ICSE '21)*. IEEE Press, 537–548. https://doi.org/10.1109/ICSE43902.2021.00057

[18] Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2020. One Size Does Not Fit All: A Grounded Theory and Online Survey Study of Developer Preferences for Security Warning Types. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 136–148. https://doi.org/10.1145/3377811.3380387

[19] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. 2018. Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174220

[20] Franz Faul, Edgar Erdfelder, Axel Buchner, and Albert-Georg Lang. 2009. Statistical power analyses using G*Power 3.1: Tests for correlation and regression analyses. *Behavior Research Methods : BRM* 41, 4 (2009), 1149–1160. https://madoc.bib.uni-mannheim.de/26116/

[21] Andy Field, Jeremy Miles, and Zoë Field. 2012. *Discovering statistics using R*. Sage publications.

[22] Colin Hughes. n.d.. Retrieved July 2023 from https://projecteuler.net/problem=1

[23] Jem, Bartholomew and Dhrumil Mehta. 2023. How the media is covering ChatGPT. Retrieved July 2023 from https://www.cjr.org/tow_center/media-coverage-chatgpt.php

[24] Harjot Kaur, Sabrina Amft, Daniel Votipka, Yasemin Acar, and Sascha Fahl. 2022. Where to Recruit for Security Development Studies: Comparing Six Software Developer Samples. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 4041–4058. https://www.usenix.org/conference/usenixsecurity22/presentation/kaur

[25] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. 2017. "I Have No Idea What i'm Doing": On the Usability of Deploying HTTPS. In *Proceedings of the 26th USENIX Conference on Security Symposium* (Vancouver, BC, Canada) *(SEC'17)*. USENIX Association, USA, 1339–1356.

[26] Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. 2023. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLoS digital health* 2, 2 (2023), e0000198.

[27] Microsoft. 2023. Microsoft PowerToys. Retrieved July 2023 from https://learn.microsoft.com/en-us/windows/powertoys/

[28] Microsoft. n.d.. Cortana. Retrieved July 2023 from https://www.microsoft.com/en-us/cortana

[29] Microsoft. n.d.. Microsoft Bing. Retrieved July 2023 from https://www.bing.com/?/ai

[30] Nerdynav. 2023. 97+ ChatGPT Statistics & User Numbers in July 2023 (New Data). Retrieved July 2023 from https://nerdynav.com/chatgpt-statistics/

[31] Helena Ojanpää, Risto Näsänen, and Ilpo Kojo. 2002. Eye movements in the visual search of word lists. *Vision Research* 42, 12 (2002), 1499–1512. https://doi.org/10.1016/S0042-6989(02)00077-9

[32] OpenAI. 2022. ChatGPT: Optimizing language models for dialogue. Retrieved July 2023 from https://openai.com/blog/chatgpt/

[33] OpenAI. 2023. ChatGPT — Release Notes. Retrieved November 2023 from https://help.openai.com/en/articles/6825453-chatgpt-release-notes

[34] OpenAI. 2023. Image inputs for ChatGPT - FAQ. Retrieved January 2024 from https://help.openai.com/en/articles/8400551-image-inputs-for-chatgpt-faq

[35] OpenAI. n.d.. ChatGPT. Retrieved July 2023 from https://chat.apps.openai.com/

[36] OpenAI. n.d.. ChatGPT GUI. Retrieved July 2023 from https://chat.openai.com/

[37] OpenAI. n.d.. OpenAI. Retrieved July 2023 from https://openai.com/

[38] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. https://doi.org/10.48550/ARXIV.2203.02155 https://arxiv.org/abs/2203.02155.

[39] Martine Paris. 2023. ChatGPT Hits 100 Million Users, Google Invests In AI Bot And CatGPT Goes Viral. https://www.forbes.com/sites/martineparis/2023/02/03/chatgpt-hits-100-million-microsoft-unleashes-ai-bots-and-catgpt-goes-viral/.

[40] Maria Carolina Penteado and Fábio Perez. 2023. Evaluating GPT-3.5 and GPT-4 on Grammatical Error Correction for Brazilian Portuguese. arXiv:2306.15788 [cs.CL]

[41] Prolific. n.d.. Prolific. Retrieved July 2023 from https://www.prolific.co

[42] Qualtrics. n.d.. Qualtrics. Retrieved July 2023 from https://www.qualtrics.com

[43] Joni Salminen, Soon-gyo Jung, and Bernard J. Jansen. 2021. Suggestions for Online User Studies. In *HCI International 2021 - Late Breaking Papers: Design and User Experience*, Constantine Stephanidis, Marcelo M. Soares, Elizabeth Rosenzweig, Aaron Marcus, Sakae Yamamoto, Hirohiko Mori, Pei-Luen Patrick Rau, Gabriele Meiselwitz, Xiaowen Fang, and Abbas Moallem (Eds.). Springer International Publishing, Cham, 127–146.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[45] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. https://doi.org/10.48550/ARXIV.2301.08653 https://arxiv.org/abs/2301.08653.

[46] Congzheng Song and Vitaly Shmatikov. 2018. Fooling OCR Systems with Adversarial Text Images. arXiv:1802.05385 [cs.LG]

[47] StackOverflow. 2023. 2023 Developer Survey. Retrieved July 2023 from https://survey.stackoverflow.co/2023/#sentiment-and-usage-ai-selectt

[48] Mohammad Tahaei and Kami Vaniea. 2022. Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 590, 15 pages. https://doi.org/10.1145/3491102.3501957

[49] Christian Terwiesch. 2023. Would Chat GPT3 Get a Wharton MBA? A Prediction Based on Its Performance in the Operations Management Course. https://mackinstitute.wharton.upenn.edu/wp-content/uploads/2023/01/Christian-Terwiesch-Chat-GTP.pdf

[50] David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27, 2 (2006), 237–246. https://doi.org/10.1177/1098214005283748 arXiv:https://doi.org/10.1177/1098214005283748

[51] Ahmed Tlili, Boulus Shehata, Michael Agyemang Adarkwah, Aras Bozkurt, Daniel T Hickey, Ronghuai Huang, and Brighter Agyemang. 2023. What if the Devil Is My Guardian Angel: ChatGPT as a Case Study of Using Chatbots in

Education. *Smart Learning Environments* 10, 1 (2023), 15.

[52] Upwork Global Inc. n.d.. Top Rated. Retrieved July 2023 from https://support.upwork.com/hc/en-us/articles/211068468-Top-Rated

[53] Qianxiang Zhou, Chao Yin, and Zhongqi Liu. 2020. The Effect of Time Pressure and Task Difficulty on Human Search. In *Engineering Psychology and Cognitive Ergonomics. Mental Workload, Human Physiology, and Human Energy: 17th International Conference, EPCE 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I* (Copenhagen, Denmark). Springer-Verlag, Berlin, Heidelberg, 111–124. https://doi.org/10.1007/978-3-030-49044-7_11

[54] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593 [cs.CL]