"The AI tool can't make it any worse." Investigating Developers' Security Behavior with AI Assistants in a Password Storage Study

Asli Yardim
Ruhr University Bochum
Bochum, Germany
asli.yardim@rub.de

Raphael Serafini
University of Cologne
Cologne, Germany
raphael.serafini@uni-koeln.de

Nadine Jost
Ruhr University Bochum
Bochum, Germany
nadine.jost@rub.de

Anna-Marie Ortloff
University of Bonn
Bonn, Germany
ortloff@cs.uni-bonn.de

Joshua Gabriel Speckels
University of Cologne
Cologne, Germany
joshua.speckels@uni-koeln.de

Alena Naiakshina
University of Cologne
Cologne, Germany
alena.naiakshina@uni-koeln.de

Abstract

Past research showed that software developers often require explicit instructions to implement security measures. With the rapid rise of AI assistant tools such as ChatGPT, it remains unclear whether AI assistance supports or undermines secure practices, whether explicit security instructions are still essential, and how developers behave without guidance. To investigate these research questions, we conducted a qualitative lab study with 21 computer science students and a quantitative online study with 80 freelance developers. We focused on secure password storage and asked participants to implement registration logic under four conditions: without instructions, with AI assistance, with security instructions, or with both AI assistance and security instructions. Our study reveals a clear behavioral shift: In our task, many participants relied on AI-assisted code generation for security-related tasks, often prioritizing convenience over security. However, explicit security-focused instructions can redirect this behavior toward secure outcomes, demonstrating that AI tools alone are insufficient without targeted guidance.

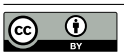
CCS Concepts

• **Human-centered computing → Empirical studies in HCI**; • **Security and privacy → Usability in security and privacy**.

Keywords

AI code assistants, secure password storage, developer security behavior, security instructions, usable security, empirical study

ACM Reference Format:
Asli Yardim, Raphael Serafini, Nadine Jost, Anna-Marie Ortloff, Joshua Gabriel Speckels, and Alena Naiakshina. 2026. "The AI tool can't make it any worse." Investigating Developers' Security Behavior with AI Assistants in a Password Storage Study. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 33 pages. https://doi.org/10.1145/3772318.3791693



CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3791693

1 Introduction

Password storage is one of the most common cryptographic tasks that developers perform [55], even though many lack formal security expertise [37, 59]. Proper implementation requires modern hash functions (e.g., Argon2, BCrypt) and unique salts to defend against offline attacks such as rainbow table lookups [9, 19, 62]. Prior work shows that without *security interventions*, e.g., explicit instructions to consider password security, developers frequently neglect these practices and rely on insecure defaults such as plaintext storage or outdated hashing algorithms [56, 57, 59]. However, these findings hold in a technological landscape where AI assistants seemed to be less prominent in developers' workflow. The rapid adoption of AI assistants such as GitHub Copilot and ChatGPT [14, 32] reshaped software development. These tools offer code suggestions, generate snippets, explain concepts, and assist with debugging. Surveys indicate that over 70% of developers already use or plan to use such tools [82, 83]. In response, some organizations now encourage or mandate their use to boost productivity and consistency [12, 51]. Although recent studies have raised concerns about the security of AI-generated code, which can be outdated, vulnerable, or incorrect [63, 67, 69, 79], developers may still adopt these suggestions without verification, especially under time pressure or due to misplaced trust [69, 77]. When major technological shifts occur, replication becomes essential because prior findings may no longer hold [42, 78]. Consequently, with AI assistance now widely adopted, revisiting pre-AI instruction results is necessary to assess whether established security interventions remain effective. This raises the question:

As AI assistants are increasingly integrated into software development workflows, how does their presence and their mandated use influence established security interventions such as explicit security instructions?

We address this question in the well-established use case of secure password storage [56–59] by conducting two complementary studies in the age of AI assistants: a qualitative lab study (Study A) with 21 computer science students, and a quantitative remote study (Study B) with 80 freelance developers. In both studies, participants implemented password storage for a user registration feature adapted from prior work by Naiakshina et al. [57, 59], which examined how developers respond to security instructions in the absence of AI assistance. In this work, participants in each study were randomly assigned to one of four instruction conditions: (1) a

non-instructed baseline group that received no additional instructions, (2) an AI-instructed group instructed to use AI assistants of their choice, (3) a security-instructed group instructed to store passwords securely, and (4) a security-and-AI-instructed group instructed both to use AI assistants and to store passwords securely. These four conditions let us measure spontaneous versus mandated AI adoption and compare this behavior to prior pre-AI security instruction studies. With this, we addressed the following research questions:

- **RQ1:** How did developers' security behavior concerning password storage change after the technological shift of popular AI assistants?
- **RQ2:** How do developers perceive the functionality and security of programming code generated by AI assistants in the context of password storage?

Building on pre-AI evidence that security instructions can improve secure password storage [57, 59] and recent findings that AI assistance can both harm and help security depending on how it is used [63, 67, 69, 79], we quantitatively test the following hypotheses in Study B:

- **H-1:** An instruction has an effect on participants' security score.
- **H-2:** Using AI assistants to solve the task has an effect on participants' security score.

Across both studies, we make three contributions. First, we show that in our password storage task, developers frequently turn to AI assistants even without explicit AI instructions, yet a functionality-first mindset appeared and security was often neglected unless explicitly requested. Second, we find that security instructions remain effective in the era of widely adopted AI assistants: In Study B, security instructions significantly improved the security of password storage implementations (H-1), and combining AI instructions with security instructions did not diminish this effect. Instead, when developers were explicitly encouraged to use AI tools while focusing on security, they achieved higher security scores than participants who did not use AI assistants (H-2). Third, our qualitative analysis of Study A and survey data from Study B reveal that participants express skepticism toward security-related AI responses, citing concerns about outdated information, data protection, and potential vulnerabilities, while still frequently copying/pasting AI-generated code, highlighting a nuanced and sometimes contradictory relationship between trust, convenience, and secure coding behavior.

Based on these findings, we discuss the security implications of instructions and outline future research paths.

2 Related Work

2.1 Developer Password Studies

Past research investigated the behavior of end-users regarding password creation, usage, knowledge, and policy. However, developers' password-storage mistakes present a greater risk to millions of users. Bonneau and Prebusch [44] examined 150 websites utilizing password authentication, uncovering weak practices such as plaintext storage and inadequate encryption on platforms with minimal security incentives, whereas more sensitive platforms demonstrated stronger measures. Acar et al. [91] conducted an online experiment with 307 GitHub users to study security-related tasks, including password storage. Among the participants, 145 failed to securely store user credentials in a database, exposing them to risks such as rainbow table attacks [62] or plaintext password storage. Another study [2] found developers relying on Stack Overflow produced less secure code. Wijayarathna and Arachchilage [90] examined 10 developers working with SCrypt in the Java Bouncy Castle API, highlighting usability difficulties stemming from the complexity of selecting secure parameters.

We base our study design on Naiakshina et al.' work [57–59], which is most related to our study. Exploring secure password storage practices and security instructions in a lab study with 20 computer science (CS) students, they found that none of the participants stored password security without being instructed, i.e., all non-instructed participants stored passwords in cleartext [59]. In a follow-up study, Naiakshina et al. [58] investigated the impact of task design and framing in developer studies. In contrast to the qualitative study, this extension took a quantitative approach involving 40 participants. The study revealed a positive impact of copy/paste on security, showing that all participants who copied/pasted code handed in a secure solution, contradicting the findings of [2, 27, 47]. Further, Naiakshina et al. [57] investigated the behavior of freelance developers concerning password storage. They found significantly fewer freelancers implemented security without security instructions. These results were also confirmed by a further follow-up study investigating the ecological validity of developer studies with CS students [56]. The instruction and framework treatment effects demonstrated statistical significance among developers. In contrast to previous research conducted before the introduction of popular AI assistant tools such as ChatGPT, henceforth referred to as *pre-AI studies*, we build on the security instruction intervention introduced in [59] and [57] to investigate how the presence and mandated use of AI assistants affect the effectiveness of security instructions in software development.

2.2 AI Assistants Studies

Developer-focused studies examined the real-world impact of AI assistants, particularly on security-related tasks, revealing both strengths and limitations in the quality of AI-generated code. Tony et al. [85] found participants fixed vulnerabilities more effectively without a development assistant than using one. Oh et al. [64] investigated the impact of poisoning attacks on AI code assistants (e.g., ChatGPT, IntelliCode) on code security. They found that developers using code generation tools were more likely to incorporate insecure code compared to those relying on code completion tools or none, highlighting a significant security impact. Similarly, Serafini et al. [77] demonstrated that AI assistance can negatively affect security when developers place misplaced trust in manipulated ChatGPT outputs. Perry et al. [69] conducted a qualitative investigation involving 47 students and professional developers using an AI code assistant to solve security-related tasks in Python, JavaScript, and C. They observed that solutions aided by OpenAI's codex-davinci-002 were less secure. Participants with AI assistance were more likely to perceive their solutions as secure. However, skeptical participants who crafted precise prompts produced solutions with fewer vulnerabilities. Asare et al. [5] explored GitHub Copilot's impact on

the solutions of 25 students across tasks of varying difficulty, from simple (user sign-on) to complex (transaction fulfillment), finding potential security benefits for the complex task. Further, security-focused work found that GitHub Copilot-generated code frequently contained vulnerabilities [67], and generating secure fixes required careful prompting [68, 79]. Further studies showed developers use GitHub Copilot to explore next steps and speed up execution [8], but still faced challenges in understanding and applying its output [72, 87]. While GitHub Copilot increased productivity in terms of code volume, it often required refinement [43]. Compared to previous work that mostly focused on GitHub Copilot, we did not restrict participants' choice of AI assistant. Instead, we explored which tools they preferred to solve the study task and how they behaved in security-related tasks when using these tools under varying instruction conditions.

3 Study A: Qualitative Lab Study on Password Storage with a Student Sample

3.1 Methodology

We conducted a lab experiment aimed to explore changes in developers' implementation workflow since the introduction of popular AI assistants, assessing their impact on code security and developers' behavior. Figure 1 illustrates our methodology. Table 1 provides an overview of our methodology in relation to the pre-AI study A [59]. In the following, we provide a detailed description of how we adapted the study.

3.1.1 Study Design. **Laboratory Setup.** We conducted the study in a laboratory, basing the task design on the pre-AI Study A [59], using the same programming language, tools, frameworks, and database as the pre-AI study A (e.g., Java, Eclipse IDE, PostgreSQL, Hibernate). We used the same source code provided to us, including the documented frontend, database backend, and task descriptions with application illustrations and implementation hints. All participants were provided the same setup concerning hardware and software and completed a backend registration feature for a university social network, continuing work left by a former team member. The task involved defining a user model, setting up database access, and completing controller logic. Participants were free to end the task whenever they considered their implementation complete or did not wish to continue. While GitHub Copilot requires a subscription and IDE integration, ChatGPT and Microsoft Copilot are freely accessible, with Microsoft Copilot offering login-free browser use. Students could access GitHub Copilot for free [31]. Because our goal was to understand developers' behavior rather than compare tools, participants could use whichever AI assistant they would usually use, whether browser-based or IDE-integrated. We collected data in four main steps:

(1) We recorded the screen during the implementation (RQ1).
(2) We collected the source code of participants' solutions (RQ1).
(3) We conducted a survey (RQ1, RQ2).
(4) We conducted a follow-up semi-structured interview (RQ2).

This approach captured web activity, search history, engagement with security-related searches, and attempts at security implementations via OBS [61] and ManicTime [49]. We also examined participants' use of AI assistants, focusing on prompting methods and interactions with AI-generated responses. Since this study was conducted in Germany, participants could choose their language preference (German or English) for all materials (see Appendix A).

Security and AI Instruction Interventions. In the pre-AI study A, half of the participants were instructed to store user password securely (see Table 1). Since we wanted to explore the interplay between AI assistance and security instructions, we introduced two additional instructions, resulting in four groups differing in their instruction scenario (see Table 2). First, to understand how participants behave without any instructions, the Non-Instructed (**N**) group was asked to implement the registration logic without any instruction intervention. This condition allowed us to observe whether participants would independently choose to use AI assistant tools and consider secure implementations without being explicitly instructed to do so. Second, the AI-Instructed group (**A**) was instructed to use any AI tool of their choice to assess how AI instructions affect developers' password security practices. Third, the Security-Instructed group (**S**) was instructed to store the passwords securely. This condition allowed us to observe whether participants would independently choose to use AI assistant tools in a security-instructed task and to compare their behavior with the pre-AI study A. Finally, the Security- and AI-Instructed group (**SA**) was instructed to securely store user passwords while also using any AI tool of their choice to investigate developers' password storage practices and their perceptions of AI suggested code. To ensure ecological validity, all participants were told they are allowed to use "any kind of source available on the Internet that may be helpful or has valuable information."

Survey. The survey involved a pre- and post-survey. In the pre-survey, participants were asked about the anticipated task difficulty after reading the task description. We did not assess participants' security expertise in the pre-survey, to avoid priming. For the same reason, only participants in groups **A** and **SA** received questions about their usage of AI tools in programming, as introducing AI-related concepts to the N and S groups immediately before the task could have influenced their likelihood of using AI. The post-survey for all participants assessed task difficulty and security behavior, as in the pre-AI study A.

Follow-up Interviews. In the pre-AI study A, follow-up interviews covered participants' security solutions and concepts like hashing or salting. We extended this with a third section focusing on AI assistants. Questions included participants' AI tool experience, usage during the study, and perceptions. Depending on the instruction scenario, we asked participants additional questions (e.g., about whether they would have used AI tools if not been prompted to do so). All interviews were conducted by the same researcher (R2) and recorded.

We conducted a pilot study with one CS student to test the functionality of our laboratory setup. The participant was not instructed and opened the post-survey before fulfilling the task and, therefore, was primed for security upon reading security-related questions. Based on this, we introduced verbal instructions advising post-survey completion after the task to prevent security priming.

3.1.2 Participants. We recruited participants similar to those in the pre-AI study A, limiting recruitment to the local area due to the lab-based nature of the study, which was conducted from August to

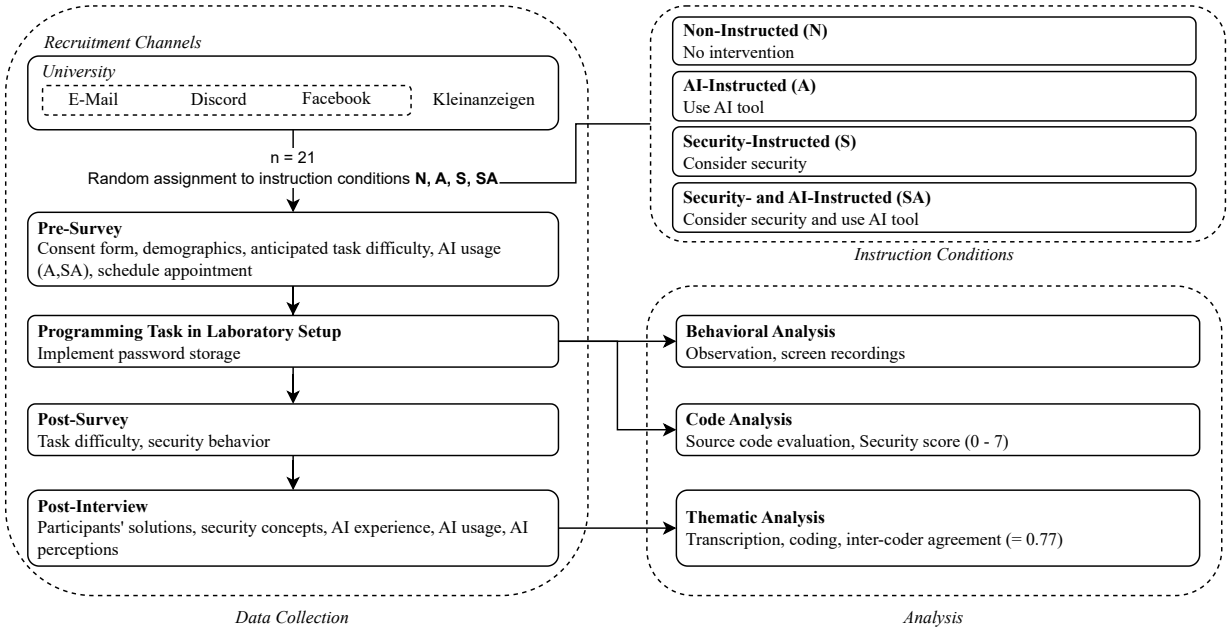


Figure 1: Methodology Overview of Study A.

Table 1: Comparison of our study design with the pre-AI study A of Naiakshina et al. [59]

	Pre-AI Study A [59]	This Study
Independent Variables (IV)	IV1: Security instruction (yes/no) IV2: Framework (JSF/Spring)	IV1: Security instruction (yes/no) IV2: AI instruction (yes/no)
Survey	Task difficulty Security behavior	Task difficulty Security behavior Usage of AI tools
Interview	Task solution Security concepts	Task solution Security concepts AI tool experience, usage, and perception
Recruitment	University	University, Facebook, Kleinanzeigen
Evaluation	Code analysis Interviews	Code analysis Interviews Screen Recording

November in 2023. Channels included our university's Computer Science student council (via e-mail and Discord [20]), university Facebook groups, nearby student councils, and Kleinanzeigen [46] (similar to Craigslist [17]). We stated in the study invitation that participation required familiarity with Java and Eclipse. We conducted a screening survey via Qualtrics [73]. Eligible participants, aged 18 or above and studying Computer Science, scheduled experiment appointments via Calendly [13]. Out of 59 interested participants, 32 scheduled appointments. The final sample included participants across four groups: Non-instructed (N, n=5), security-instructed (S, n=6), AI-instructed (A, n=5), and security-AI-instructed (SA, n=5).

We report demographics and experience across the four groups in Table 3. We recruited 21 full-time CS students (16 BSc, 4 MSc), ages 20–39 (μ = 24.5), 18 men, one woman, two undisclosed. Six worked part-time as developers; two were research assistants. The tool experience of participants was measured using four 7-point Likert scales capturing familiarity with Java, PostgreSQL, Hibernate, and the Eclipse IDE (1 = not familiar at all, 7 = very familiar). Scores were summed for comparability with the pre-AI study A, resulting in a possible range of 4–28. Their average tool experience was μ = 12 (vs. 13.6 in the pre-AI study A). The study was advertised as lasting up to 8 hours , for which participants were compensated 115 €.[1] Actual completion times ranged from 1:52 to 6:48 hours.

[1]Unlike the pre-AI study A providing lunch and 100 € compensation, we asked participants to bring their own lunch due to university restrictions and increased compensation by 15 € to cover expenses.

Table 2: Instructions Depending on the Groups.

		AI Instruction	
		No	Yes
Security Instruction	**No**	**Non-Instructed (N)**	**AI-Instructed (A):** *"'Please use an AI Tool of your choice."*
	Yes	**Security-Instructed (S):** *"Please ensure that the user password is stored securely."*	**Security- and AI-Instructed (SA):** *"'Please use an AI Tool of your choice. Please ensure that the user password is stored securely."*

Table 3: Demographics of the 21 Participants.

Gender	Man: 18, Woman: 1, Not Disclosed: 2
Age	min = 20, max = 39, μ = 24.5, md = 23, σ = 4.7
Study program	BSc: 16, MSc: 4, Not Specified: 1
Tool Experience with Java, PostgreSQL, Hibernate, and Eclipse IDE (Not familiar at all (4) – Very familiar(28))	
N	μ = 12.4, md = 11, σ = 4.
A	μ = 13.6, md = 15, σ = 3.4
S	μ = 8.8, md = 9, σ = 2.1
SA	μ = 13.8, md = 13, σ = 4.8
Total	μ = 12, md = 11, σ = 4

3.1.3 Evaluation. **Code Analysis.** We assessed functionality and security using the same method as the pre-AI study A. Solutions were functional if user data from the login interface was successfully stored in the database. To assess the security of participants' solutions, we used a security scale. The score considered various factors, including the choice of the hashing algorithm, the iteration count, and the method of salt generation. The score ranged from 0 to 7, with 7 being the highest possible score:

- The end-user password is salted (+1) and hashed (+1)
- The derived length of the hash is at least 160 bits long (+1)
- The iteration count for key stretching is at least 1 000 (+0.5) or 10 000 (+1) for PBKDF2 and at least 2^{10} = 1 024 for bcrypt (+1)
- A memory-hard hashing function is used (+1)
- The salt value is generated randomly (+1)
- The salt is at least 32 bits in length (+1)

In the pre-AI study A, solutions scoring 6 points met industry-standard security due to default framework values [35, 36], while a score of 7 (academic standard) required memory-hard hashing functions like Argon2 [9, 38, 66]. We conducted additional analysis on participants' solutions, focusing on *attempted security*. This assessment considered participants to have made an attempt at security if participants used a search engine or AI assistant to gather information on terms such as *security*, *secure storage*, *hashing*, *encryption*, or equivalent terms.

Screen Recording. We analyzed screen recordings in 10-second intervals, focusing on participants' security implementations, AI tool usage, and coding strategies (e.g., *copy/paste*, *debugging*). Observation duration was adjusted based on task complexity. AI assistant use was identified via ManicTime logs [49], with timestamps and interaction types documented (see Appendix A.6.2).

Interviews. We transcribed 21 interviews and analyzed the data using thematic analysis following Boyatzis [10]. After familiarizing themselves with the transcripts, researchers R1 and R2 independently developed initial codebooks, which they merged through discussion. They jointly coded a random subset of 4 interviews and individually coded the remaining 17. New codes were added to the unified codebook as they emerged. Each researcher (R1, R3) coded half of the dataset in the first round. In the second round, each coded the remaining transcripts and recoded them from the most recent to the earliest. This procedure helped to ensure that transcripts coded earlier in time were reviewed with the full set of concepts developed during coding. Themes were derived from recurring patterns and interconnections, such as distrust in AI security suggestions due to concerns about data protection or outdated information (*"[...] ChatGPT, for example, only has knowledge from five years ago."* — [N2]). Inter-coder agreement was reached through two rounds of discussion and measured using Cohen's kappa (κ) [15], resulting in a high agreement of 0.77 [28]. The codebook can be found in Appendix A.7.

3.1.4 Ethics. Our university's institutional review board (IRB) approved the project. Participants were informed in the recruitment invitation that their screen would be recorded during the programming task. They also received a consent form in the screening survey outlining all data practices, including the use of screen recordings (OBS), application-usage logs (ManicTime), and audio recordings for the interview. These recordings were stored locally during the session and subsequently uploaded in encrypted form to our university's cloud storage. We adhered to the General Data Protection Regulation (GDPR), allowing participants to ask questions and withdraw at any time without consequences. No directly identifying information were collected beyond what was necessary for scheduling and compensation. Personal identifiers were kept separately in a pseudonymised and encrypted manner to allow data deletion upon request. At the end of the interview, all participants were debriefed and informed about the study's purpose. In particular, participants in conditions without AI instructions were told that the study examined developers' use of AI tools and participants without security instructions were informed that the study examined secure coding behavior.

3.1.5 Limitations.

Study Design. As with any laboratory study, being observed and working in a controlled environment might have influenced participants' behavior, e.g., by increasing caution. The long task duration (up to eight hours) may also have introduced fatigue or

time pressure, affecting persistence with secure implementations. Further, although we based our setup and scenario on the pre-AI studies for comparability, it might only partially reflected current development workflow.

AI Underuse. Although our instructions permitted any kind of source available on the Internet, the explicit reference to websites in the task description adopted from the pre-AI study might not have been interpreted as including AI tools. Additionally, the fact that all screen activity was recorded may have further discouraged their use.

IDE Setup. We used Eclipse IDE, as in the previous study [59], limiting our study to free AI assistants compatible with the IDE. While no official GitHub Copilot plugin existed for Eclipse at the time of our study, unofficial integrations with positive community feedback [22, 29, 30] provided a workaround. We acknowledge that using a different IDE or allowing participants to use their own environments might have impacted their behavior.

Coding. Iterative qualitative coding carries the risk that early transcripts were not reviewed with the full set of concepts in mind, meaning relevant phenomena may have been overlooked before a code was formally introduced. As a result, an evolving codebook can reduce comparability across transcripts and may affect inter-coder reliability. We sought to mitigate this by conducting a structured second pass in which each researcher recoded the other's transcripts from the most recent to the earliest, so that transcripts coded early in the process were revisited after substantial parts of the codebook had already been developed.

Impact. The introduction of AI tools and their impact on software security presents a complex problem that is difficult to study due to their rapid evolution. However, empirical security research shows that behavioral patterns, particularly habits and routines, change more slowly than the tools themselves [6, 40, 70]. Moreover, despite improvements in newer LLM versions [33, 34, 54], recent work demonstrates that security-relevant hallucinations, incomplete reasoning, and outdated or insecure code suggestions persist across model updates [69, 77]. Therefore, with this work, we take a first step toward understanding developers' behavior by comparing insights from a security study with developers conducted before the emergence of AI tools and after the introduction of popular AI tools. We raise awareness of both the potential and the risks of using AI tools for security critical tasks such as secure password storage, an insight that remains valid despite the evolving capabilities of AI assistants.

4 Study A: Results of the Qualitative Lab Study on Password Storage with a Student Sample

We present solutions and statements by participants by labeling them with N, A, S, and SA from 1 to 6 according to the instruction scenario. While our primary focus is on qualitative results, we also report how many participants reported statements where necessary to indicate a distribution. To support statements, we list participants who stated the respective themes. If more than five participants mentioned a theme, we list a random subset of participants by including at least one representative from each group. In the interviews, all but SA1 reported using ChatGPT before the study. Among these participants, 13 participants used ChatGPT during the study as well. Additionally, 6 participants used Microsoft Copilot. A full Table of participant's AI usage can be found in the Appendix A.1. Expected task difficulty in Study A fell on the difficult to neutral end of the scale for all groups. Mean expected difficulty ranged between 2.6 and 3.2 (N: μ = 2.6, A: μ = 3.0, S: μ = 2.7, SA: μ = 3.2), with corresponding medians of 2.0 – 3.0, indicating broadly similar expectations before the task. Experienced task difficulty followed the same pattern: The AI-instructed groups showed higher variability (A: σ = 2.3, SA: σ = 1.5) compared to the non-AI-instructed groups (N: σ = 0.6, S: σ = 0.6), while median values remained within a narrow range (N: md = 2.0, A: md = 4.0, S: md = 2.0, SA: md = 4.0).

4.1 Password Security Behavior (RQ1)

Table 4 provides a summary of the participants' final solutions, outlining the functionality, security, source, and task completion time. Eight of 21 participants submitted a functional solution: One from S (S6), three from A (A1, A3, A4), and four from SA (SA1, SA3, SA4, SA5) successfully stored the password in the database. None of the five N group participants stored any user data. The main implementation issues were database-related (e.g., Tomcat Server setup). We analyzed the screen recordings to assess participants' resources and strategies for implementing a secure solution. Similar to the pre-AI study A, all our participants who implemented security measures, adopted a copy/paste strategy. Compared to the pre-AI study A where all participants used rather blog posts, our participants obtained code snippets primarily from ChatGPT (SA1, SA3, SA4, SA5), with A4 using Microsoft Copilot, and A3 and S6 using a blog post.

None of the N group submitted solutions with security measures (see Table 4). In the S group, S6 achieved the lowest score (2), and S1 attempted secure password storage but failed. Of our 10 participants instructed for AI assistants, six (A3, A4, SA1, SA3, SA4, SA5) successfully implemented a secure solution with password hashing. Four of these participants (SA1, SA3, SA4, SA5) were specifically instructed for secure password storage *and* AI assistant usage. The security scores for security measures varied between 2 to 7, with SA4 achieving the lowest score of 2 in their group. A3 and SA1 implemented security measures but did not achieve industry standard security of 6 points, with 5 and 5.5 points respectively. By contrast, SA5 achieved industry standard with 6 points. A4 employed a memory-hard hash function, Argon2, and salted the password, achieving the maximum security score possible - 7, i.e., academic standard.

We compared security scores between our participants and those from the pre-AI study A [59]. In the pre-AI study A, only participants instructed for security considered security, with three of the 10 JSF participants implementing security measures, achieving a mean security score of 4.5/7, and one reaching industry standard of 6 points. In our study, 7 of the 21 JSF participants implemented security measures with a mean security score of 4.64 out of 7. Excluding participant S6 and A3, who did not use AI assistance for their implementation of security measures, resulted in a mean of 5.1 out of 7. One participant reached industry standard (6 points).

Additionally, while none of the pre-AI participants received a full score, one of our participants reached academic standard (7 points). However, despite AI assistance, scores improved by only 0.5 points, with most participants submitting solutions still below the industry standard.

4.1.1 Participants' Behavior Under No Instructions. Two participants who were not explicitly instructed to use AI assistants (N1, N3) opted to use AI assistance during the task. However, they still did not solve the task functionally. Further, some participants seemed to feel using AI in a study might be prohibited. Instead of asking for permission, they attempted to use AI on personal devices. For example, N3 opened ChatGPT on their personal tablet. Recognizing this incident, we communicated that all activities should be displayed on the PC screen. This participant visited sites that accessed ChatGPT before eventually landing on ChatGPT's official website and logging in. By contrast, N4 believed it was not in the interest of the study to use AI assistants, so they did not use them. However, they stated if it had not been a study, they would have used an AI assistant, *"[s]imply because having a different perspective is always helpful, and if it didn't work before, the AI tool can't make it any worse."* — [N4].

Participants cited various reasons for lack of awareness regarding the necessity of a secure solution, including prioritizing functionality first (N3) and a simple lack of awareness that a secure solution was required (N1, N2).

4.1.2 Participants' Behavior Under AI Instructions. We asked participants who were instructed to use AI whether they would have used it if not been told to do so. Among the participants who would have used it (A1, A2) and those who would not have used it (A3, A4, A5), reasons were based on similar aspects, such as habits and experience. Reasons for the former were based on positive experiences and the habit of using AI. In contrast, reasons for the latter were based on negative (or the lack of) experiences and the habit of *"programming like this for a very long time, and it's still an old habit not to use AI."* — [A5]. A3 would have assumed it was not the purpose of the study and, therefore, would not have used AI if not told to do so. This participant did not use any AI assistant to specifically secure their solution and employed a less secure hash function by copying/pasting code snippets from a blog post [7] found through a Bing search. The blog post recommended the algorithm PBKDF2WithHmacSHA1, known for having a hash length of less than 160 bits. A2 did not attempt to implement a secure solution, *"due to the task, [as] it did not appear to be required"* — [A2]. A2 noted they focused on fulfilling task requirements before security (functionality first), while two other participants (A1, A5) showed a lack of awareness that a secure solution was required. A1, who did not implement a secure solution, stated during the interview that they were unaware of security. However, our video recordings showed they specifically prompted ChatGPT on password validation and received a mention of BCrypt in between functional suggestions of the same response. Interestingly, they incorporated all suggestions except for the recommended security measure. While it appears that they read the response, given their incorporation of the functional suggestions, they actively opted not to address security without an explicit instruction. A1 further mentioned that they would have also stored the password securely if it had been a task in a company instead. These observations align with findings in the pre-AI study A, where participants relied on requirements.

A4 achieved the highest study score, with 7 points for security. They proactively searched for hashing, salting, and peppering information, indicating their familiarity with these security concepts. They found Argon2 through multiple sources, including Microsoft Copilot, Stack Overflow [84], and OWASP [66]. A4 read OWASP's Password Storage Cheat Sheet, which recommended using Argon2id for password hashing. However, the OWASP site [66] did not provide a code snippet, and A4 referenced a Stack Overflow thread about Argon2 hashing [84]. Subsequently, they requested Microsoft Copilot to generate a code snippet using the query "java argon2id" and copied/pasted the resulting code snippet into their solution. To implement their solution, A4 explored additional links suggested by the same response from Microsoft Copilot, focusing on aspects such as binding the dependency of the Argon2 algorithm [4]. Finally, A4 relied on Microsoft Copilot to implement the code of their solution after consulting various online platforms, including OWASP and Stack Overflow.

4.1.3 Participants' Behavior Under Security Instructions. As in the non-instructed group (see Section 4.1.1), two participants (S2, S5) reported they could not address security, as they did not progress to functional solutions. Another participant (S3) reported they were not actively instructed for security but received security-related responses from the corresponding AI assistants (ChatGPT) during the study. Participants without explicit AI instructions (S1, S2, S3) nevertheless chose to use AI tools during the task. However, their solutions were still not functional. In addition, some participants stated to be uncertain whether using AI was even permitted in the study. Similarly to A3 (see Section 4.1.2), S6 copied/pasted code from Stack Overflow thread [81]. The participant demonstrated awareness of hashing, evident by their creation of a function named GetPasswordHash(), even before encountering the concept anywhere. However, the Stack Overflow thread from which the participant copied/pasted code did not address salting. Consequently, S6 failed to implement a salt in their code, resulting in the lowest security score of 2 in our study.

4.1.4 Participants' Behavior Under Security and AI Instructions. Similar to the AI-instructed group (see Section 4.1.2), participants in the security- and AI-instructed group were asked whether they would have used AI if they had not been instructed to do so. Among those who indicated they would (SA2, SA4) and those who indicated they would not (SA1, SA3, SA5), the reasons were likewise based on similar aspects, such as habits and prior experience. Participants' behavior indicated that avoidance of third-party libraries was rather driven by integration difficulties than uncertainty about whether libraries were permitted. Screen recordings showed that participants actively attempted to import secure libraries (e.g., BCrypt) and abandoned them after encountering import or configuration errors, subsequently prompting the AI for Java alternatives that required no setup. For example, upon encountering a response from ChatGPT which included explanations of the security concepts HTTPS, password hashing, and salting, SA3 inquired ChatGPT "How can one hash a password in Java using a random salt?" Despite ChatGPT's initial suggestion to use BCrypt, SA3 preferred to avoid using an external library. They asked ChatGPT "Can I do

Table 4: Evaluation of the Functionality and Security of Participants' Solutions. N=Non-Instructed, S=Security-Instructed, A=AI-Instructed, SA=Security-AI-Instructed.

(*)Italic indicates attempted security solutions, which are not part of the final submissions but were attempted during the task.

	Time	**Functionality**	**Source**	**Security**					
					Hashing			**Salt**	
	(hh:mm)	Storage working		Hashfunction (at most +2)	Length(bits) (+1 if ≥ 160)	Iteration count (at most +1)	Used library (at most +2)	Length (bits) (+1 if ≥ 32)	**Total** (7)
N1	06:44	✗							0
N2	06:52	✗							0
N3	06:11	✗							0
N4	06:36	✗							0
N5	03:24	✗							0
A1	02:54	✓							0
A2	07:04	✗							0
A3	03:14	✓	Baeldung	PBKDF2WithHmacSHA1	128	65 536	javax.crypto	128	5
A4	03:51	✓	Microsoft Copilot	Argon2	256	10	Spring	128	7
A5	06:48	✗							0
S1	03:22	✗	*ChatGPT*	*BCrypt*	*192*	*1 024*	*mindrot*	*128*	*6**
S2	03:19	✗							0
S3	04:05	✗							0
S4	05:48	✗							0
S5	06:40	✗							0
S6	05:45	✓	Stack Overflow	SHA-256	256	1	java.security	–	2
SA1	03:15	✓	ChatGPT	PBKDF2WithHmacSHA256	256	1 000	javax.crypto	128	5.5
SA2	05:42	✗							0
SA3	01:52	✓	ChatGPT	SHA-256	256	1	java.security	128	5
SA4	03:34	✓	ChatGPT	SHA-256	256	1	java.security	–	2
SA5	03:52	✓	ChatGPT	BCrypt	192	1 024	mindrot	128	6

this in pure Java(20), without relying on external libraries?" Subsequently, ChatGPT provided a code snippet featuring SHA-256 and a salt function, which SA3 copied and pasted. Similarly, in response to SA4's prompt about password hashing, ChatGPT recommended BCrypt with a salt and multiple iterations, accompanied by a secure code snippet. When facing challenges to import BCrypt, SA4 sought guidance by prompting "I cannot add libraries. How would I hash a password" and, similar to SA3, SA4 received a SHA-256 example instead. However, this code snippet did not include a salt. We assume that since SA4 specifically prompted how to *hash* a password, the response focused solely only hashing. Our screen analysis indicated that the switch to an insecure solution was rather driven by import friction than by permission constraints. Further, SA1 implemented PBKDF2WithHmacSHA256 via a code snippet generated by ChatGPT with a hash length of 256. Upon a non-security-related prompt, SA2 received a recommendation from ChatGPT to use hashing and salting for passwords. However, SA2 proceeded to try to solve the task functionally without implementing the suggested security measures. Like A1, SA3 indicated they would have ensured secure storage in a company setting, which once more reflects the pre-AI study A findings on reliance on requirements.

SA1 and SA5 issued similar ChatGPT prompts for secure password storage but received different recommendations (PBKDF2 vs. BCrypt), likely due to differences in provided context. Both solutions achieved comparable security scores (5.5, 6). Note that the missing 0.5 points of SA1 resulted from a modification of the participant. Despite lacking prior knowledge of salting, both participants ultimately implemented salting by reusing ChatGPT's secure examples – SA1 after being prompted by ChatGPT's insistence, and SA5 by directly copying the generated code. Although unfamiliar with salting during the interview, SA5 successfully implemented it by copying a secure ChatGPT-generated snippet.

RQ1: Participants' Security Behavior – Summary

The decision to use AI assistance during the task, whether mandated or voluntary, was shaped by participants' habits and prior experience. A functionality-first mindset dominated across all groups, including those receiving security instructions. However, combining security instructions with AI instructions yielded the most effective results: All but one participant in this condition submitted both functional and securely hashed solutions. Similar to the pre-AI Study A, our findings confirm the reliance on copy/paste for submissions, but this reliance has shifted toward AI assistance rather than Stack Overflow or tutorials. All but three participants used AI assistants, effectively incorporating code snippets primarily from ChatGPT and Microsoft Copilot. However, our results also indicate a tendency to prioritize convenience over security, with participants favoring easily copyable solutions, even when they were insecure. More concerningly, participants adapted their solutions to be less secure to avoid using third-party libraries. Combining multiple resources yielded the best results, emphasizing the importance of leveraging diverse resources.

4.2 Participants' Perceptions of the Functionality and Security of AI-Generated Code (RQ2)

In the following, we present our participants' perceptions of functional and security-related generations by AI assistants from the follow-up interviews.

4.2.1 Participant Perceptions of Security in AI Responses. Almost all participants expressed skepticism towards security-related AI suggestions. Fourteen participants reported they do not trust security-related suggestions by AI assistants (N2, N5, A2, A3, A4, A5, S1, S2, S3, S4, S5, S6, SA1, SA2), mentioning various reasons for their distrust, such as outdated information (*"[...] AI is limited in terms of the data, which is not so recent, i.e., it has been around for months. So, security [...] is always being updated."* — [S1]), data protection violation (*"Of course [...] data protection, which is something you should be more careful about."* — [SA1]), and vulnerabilities in suggestions (*"You don't have to adopt all of [the suggestions], because there could also be a few things that are missing at the security-relevant location that the AI bot has probably skipped"* — [S2]). S6 and SA2 placed greater trust in statements made by individuals with experience, such as security experts. At the same time, N5 believed ChatGPT would not be able to propose a secure solution and did not use any AI assistant during the study. Interestingly, despite their distrust, A4 and SA1 copied/pasted code generated by AI assistants and implemented a secure solution.

The four participants (N3, A1, SA4, SA5) who indicated to trust security-related suggestions by AI assistants do so only with caution, except for SA4, stating that they trust security-related AI suggestions due to their lack of experience in security concepts. SA5 highlighted *"[...] you always have to question the security aspects in particular. Whether what is said there, why it is actually correct, so I'm still a bit cautious."* — [SA5]. Similarly, despite trusting security-related suggestions, A1 mentioned caution when prompting ChatGPT due to data privacy: *"I wouldn't use it for, for example, I wouldn't give ChatGPT any code from my employer or anything like that, because there were also security concerns because ChatGPT stores everything"* — [A1]. Further, N3 noted they *"use [...] website[s] because ChatGPT has a limited knowledge from 2021"* — [N3]. However, they put their trust in security-related suggestions up to 2021. SA1 was not familiar with AI assistants before the study but believed they would implement security more extensively in the future due to AI assistance (*"I think [I would implement security] maybe a bit more extensively for sure."* — [SA1]).

4.2.2 Participant Strategies in Assessing Functionality and Security in AI Responses. Eleven participants (e.g., N3, A1, S2, SA4) emphasized the importance of checking the responses of AI assistants for functionality and security. Interestingly, most of the participants in groups A and SA believed assessing AI responses before adopting them is crucial. Nine participants (e.g., A5, S1, SA1) indicated relying on their knowledge to evaluate the correctness of AI responses, highlighting the importance of using personal expertise (e.g.,*"[The responses] were objectively wrong. [...] But I don't think I would have known that if I hadn't dealt with programming in general beforehand."* — [SA4]). Two participants highlighted the relevance of assessing responses by researching information from external websites (e.g., *"I had a further look on another website"* — [A3]). A1 claimed both strategies to be essential: *"You also have to have an idea yourself, maybe do some research, make sure [it's correct], especially when it comes to critical things."* — [A1]). S6 and SA4 stated they would still prefer consulting a security expert. If no security expert is available, SA4 would consult AI assistants. Similarly, A4 indicated if they would lack sufficient security knowledge for a task, they would opt for an AI assistant.

Despite participants highlighting the importance of assessing AI responses, only N3 and SA3 checked each code snippet before adopting it. Most participants (e.g., A1, S1, SA4) copied/pasted code partially without assessment. SA2 copied/pasted their programming code without any evaluation.

RQ2: Participants' Perceptions – Summary

Participants expressed varying trust in AI-generated code based on potential errors and the need for additional assessment. Specifically regarding security suggestions, participants were concerned about outdated information, data protection, and potential vulnerabilities. They still implemented code generated by AI assistants, revealing a nuanced relationship between distrust and practical reliance on AI-generated solutions. Participants with limited security experience preferred AI assistance when lacking sufficient knowledge for a task.

5 Study B: Quantitative Online Study on Password Storage with Freelance Developers

5.1 Methodology

Study A indicated that developers do not implement secure password storage without explicit instructions and that combining security and AI instructions yields the most secure outcomes. To explore whether these findings are generalizable, we conducted a follow-up Study B with freelance developers and tested how different instruction conditions influence their security scores. Additionally, Study A revealed that many participants relied on copy/paste use of AI-generated code. Prior research has shown that such code is frequently outdated, insecure, or vulnerable if adopted uncritically [64, 67, 69, 79, 85], although in some cases careful prompting can support more secure implementations [5, 75]. Accordingly, Study B also investigated the effect of actual AI use on security scores. With this, we tested the following hypotheses:

- **H-1:** An instruction has an effect on participants' security score.
- **H-2:** Using AI assistants to solve the task has an effect on participants' security score.

Figure 2 provides an overview of our methodology for study B. The recruiting material, surveys, task description, and codebook can be found in the Appendix B.

5.1.1 Study Design. **Online Setup.** Similar to Naiakshina et al.'s work [57], where freelance developers were recruited to explore the effect of security instructions on secure password storage pre-AI, we conducted a follow-up online study with freelancers, where participants were hired to work on the task. However, deception needs to be treated with care in empirical research and should only be

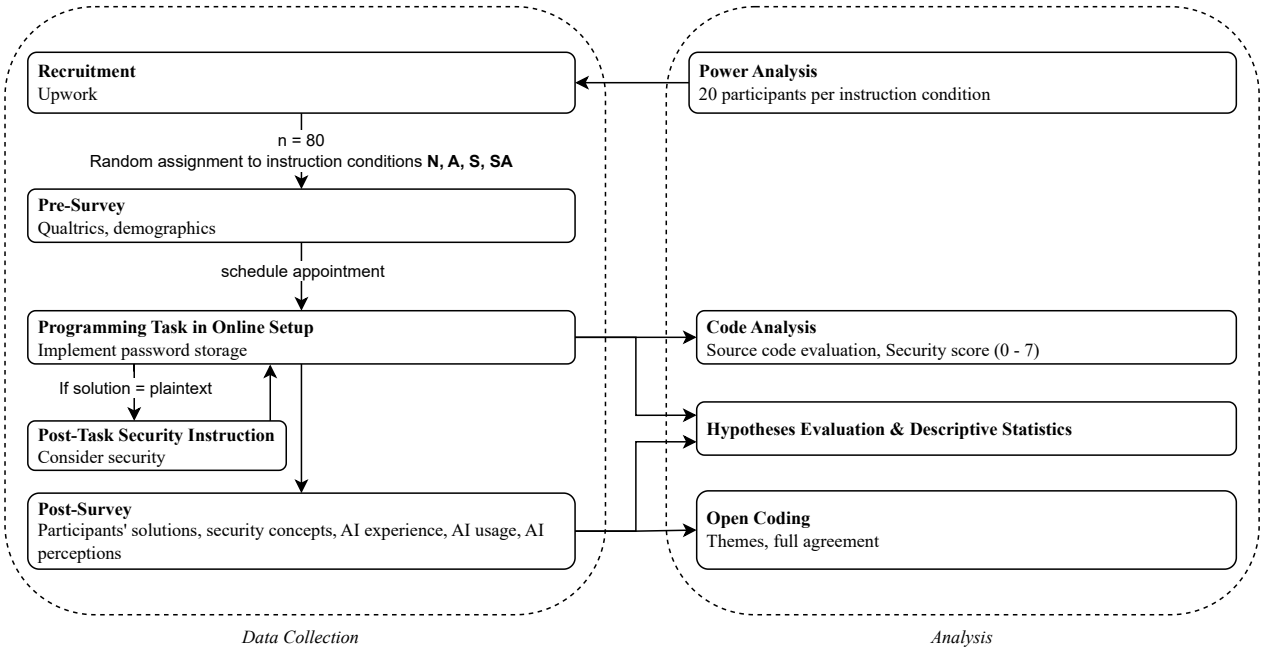


Figure 2: Methodology Overview of Study B.

used in cases where other study formats would not allow a research question to be investigated. Since the replication study by Danilova et al. [18] showed that, in the context of password storage studies, announcing the task as real company work versus a research study did not affect developers' security behavior, we followed their recommendation and informed freelancers in Study B about the research context of this study, i.e., while participants were aware that they were taking part in a study, they were still assigned to the different conditions as in Study A. Table 5 depicts an overview of our variables in comparison with the pre-AI study B [57]. For our study setup, we used the same source code, programming language, frameworks, and database as in Study A, such as Java [65], Eclipse IDE [21], PostgreSQL [71], and Hibernate [39]. However, participants were free to use any tools of their choice, including different IDEs. We set up a Windows Virtual Machine (VM) in Azure [52] deployed within the participants' closest geographical region to ensure minimal latency during the study. Additionally, an Azure Database for PostgreSQL [71] was provided, giving participants a database environment for their work. A GitHub repository for each participant was bookmarked to submit their solutions. Participants accessed the virtual machine through the Remote Desktop Protocol (RDP) [53]. After completing the pre-survey, participants scheduled their experiment with us. On the day of the experiment, participants received login credentials for a virtual machine, instructions for accessing the task description, and a reminder that the study could take up to eight hours. We emphasized the importance of completing all work within the provided virtual machine. All materials were provided in English.

Security and AI Instruction Interventions. We adopted the same methodology as in Study A, using the four conditions as described in Section 3.1.1. Further, upon receiving participants' solutions, we verified whether user passwords were stored in plaintext in the database. Similar to [57], participants received a *post-task security instruction* to revise their submission and ensure secure password storage if they submitted plaintext solutions.

Survey. Our post-survey was based on the pre-AI study B. We extended it to investigate developers' interactions with and perceptions of AI assistants and security practices. Beyond assessing AI assistance usage frequency, and trust in AI-generated suggestions with close-ended questions, the survey contained open-ended questions about participants' implementation strategies, their interactions with AI assistants, their trust in AI-generated suggestions, and their reasoning behind security decisions. Depending on the instruction scenario, we asked participants additional questions (e.g., about whether they would have used AI tools if not been instructed to do so).

We conducted a pilot study to test the functionality of our remote setup with four participants. We recruited via the platform Upwork, assigning them randomly to the groups N, S, A, and SA, each. We identified that Git [16] was not installed on the VM in the first round, which prevented the participant (N_{P1}) from submitting their solution. This issue was promptly addressed by installing Git, after which we completed the pilot study without further issues.

5.1.2 Power Analysis. We conducted a power analysis using G*Power[24] to determine the required sample size based on our main hypotheses. Three a priori analyses were performed: One

Table 5: Comparison of our study design with the pre-AI study B of Naiakshina et al. [57].

	Pre-AI Study B [57]	This Study
Independent Variables (IV)	IV1: Security instruction (yes/no) IV2: Payment (€100/€200)	IV1: Security instruction (yes/no) IV2: AI instruction (yes/no)
Survey	Task difficulty Security behavior	Task difficulty Security behavior AI tool experience, usage, and perception
Recruitment	Freelancer.com	Upwork
Evaluation	Code analysis Survey	Code analysis Survey

for an omnibus test comparing four instruction conditions (non-instructed, AI-instructed, security-instructed, security-and-AI-instructed) and two for post-hoc tests comparing AI-instructed (**A**) and security-and-AI-instructed (**SA**) individually to the non-instructed group (**N**). Calculations used $\alpha = 0.05$, power = 0.8 [23], and the two-tailed tests for Wilcoxon rank-sum tests [26]. Effect sizes were derived from prior work [57] and Study A, which indicated Cohen's d values of 1.137 for the group A compared to the group N and 2.256 for the group SA compared to the group N. Using group means ($\mu_N = 0$, $\mu_A = 2.4$, $\mu_{SA} = 3.7$) and a pooled standard deviation (1.98) from this study, we calculated Cohen's f = 0.77, yielding a sample size of 7 participants per group for a one-way ANOVA omnibus test. Since the security score is not strictly interval-scaled, we planned to use a non-parametric Wilcoxon rank-sum test, which G*Power does not support for power analysis. To adjust for the reduced power of non-parametric tests, we applied a correction factor of 1.16, increasing the sample size to 8 participants per group for the omnibus test. For post-hoc comparisons using Wilcoxon rank-sum tests, we estimated sample sizes of 6 participants per group for comparing SA to N and 20 participants per group for comparing A to N.

5.1.3 Participants. We recruited freelance software developers from Upwork [86] from February to June 2024, using the platform's user profiles to ensure that all participants had experience in software development. Additionally, we conducted a pre-survey via Qualtrics. To participate, individuals had to meet the following criteria: be at least 18 years old and have Java programming experience. To ensure data quality, we included two screening questions in the job post participants had to complete after being invited to the study by the Upwork team. We used a consecutive sampling approach [48], verifying participants' responses via their profiles as proposals arrived. The Upwork recruitment service initially invited 2369 developers, resulting in 1455 proposals. Of these, we contacted 118, and 80 participants responded, completed a pre-survey, scheduled an appointment, and completed the main study. Upwork invited a diverse pool of software developers without requiring Java experience, resulting in many proposals from participants lacking Java skills and a significant gap between proposals submitted and invitations sent. Consistent with prior findings that varying compensation did not affect developers' security behavior in the pre-AI Study B [57], we compensated participants via Upwork with a fixed amount independent of task completion time, converting the compensation of the pre-AI study B of €200 for the task and €20 for the survey into $215 and $25. The study was advertised as taking up to eight hours, with the actual average length per participant being 435 minutes (min: 87 minutes, max: 2927 minutes, median: 279 minutes, σ: 557).

Table 6 summarizes the demographics of our participants. The majority identified as men (96.3%), with a mean age of 31.5 years ($\sigma = 6.9$, range 18–50). Most participants were freelancers (65%) or industrial developers (25%), and they primarily developed web applications (72%) and enterprise applications (48%). On average, they had 6.74 years of Java experience ($\sigma = 4.4$) and 9.2 years of overall software development experience ($\sigma = 5.1$). Geographically, 37.5% were from countries such as Pakistan (10%), India (7.5%), Sri Lanka, Kenya, Bangladesh, and Egypt (5% each), with 62.5% from other locations. Additionally, 93.8% held a university degree. Familiarity with Java, PostgreSQL, Hibernate, and Eclipse IDE was generally high (mean Likert scores 5.3 - 6.3). The majority of participants (75%, SA and A) had prior experience using AI for programming.

5.1.4 Analysis. **Code Analysis.** We used the same assessment of *functionality* and *security* of participants' solutions as in Study A (see Section 3.1.3). This included scoring participants' solutions using a 0 - 7 security scale based on factors such as hashing algorithm choice, iteration count, and salt generation, similar to [57].

Survey. We analyzed survey responses using descriptive statistics and open coding [74]. Descriptive statistics summarized quantitative trends, patterns, and metrics. Open coding [74], applied to open-ended responses, identified recurring themes, concepts, and insights. Researchers R1 and R2 collaboratively developed an initial codebook by jointly coding four surveys. Each then independently coded 40 surveys, updating the codebook as needed and meeting regularly to discuss new codes. They then cross-coded each other's subsets so both had coded all surveys. Finally, both researchers discussed and resolved all coding conflicts to merge the documents and reach full agreement. Conflicts arose between segments assigned to the codes "AI-generated suggestions need to be assessed" (specific behaviors) and "AI needs to be handled responsibly" (broader interactions). Inter-coder agreement was not calculated to value individual researchers' views [50]. The full codebook is in Appendix B.7.

Hypotheses Evaluation. We conducted confirmatory hypothesis testing for H1, a structured approach that tests specific predictions informed by prior work. This involved conducting a power analysis to ensure the study was designed to reliably detect the expected effects [11]. Since our dependent variable is ordinal, we

Table 6: Demographics of the 80 participants.

Gender	Man: 77 (96.3%), Woman: 3 (3.8%)
Age	min = 18, max = 50, μ = 31.5, md = 31, σ = 6.8
Main occupation	Freelancer: 52 (65%), Industrial developer: 20 (25%), Student: 4 (5%), Other: 4 (5%)
Type of Software developed	Web Applications: 72, Enterprise Applications 48, Desktop Applications: 35, Mobile Applications: 31, Embedded applications 13 Other: 5
Java experience [in years]	min = 0, max = 21, μ = 6.7, md = 6, σ = 4.4
Software Development experience [in years]	min = 0, max = 21, μ = 9.22 , md = 8, σ = 5.2
Country	Pakistan: 8 (10%), India: 6 (7.5%), Sri Lanka: 4 (5%), Kenya: 4 (5%), Bangladesh: 4 (5%), Egypt: 4 (5%), Other: 50 (62.5%)
University Degree	Yes: 75 (93.8%), No: 7 (6.3%)
Java Familiarity [7-Likert]	min = 3, max = 7, μ = 6.3, md = 7, σ = 0.9
PostgreSQL Familiarity [7-Likert]	min = 2, max = 7, μ = 5.9, md = 6, σ = 1.2
Hibernate Familiarity [7-Likert]	min = 1, max = 7, μ = 5.4, md = 6, σ = 1.6
Eclipse IDE Familiarity [7-Likert]	min = 1, max = 7, μ = 5.3, md = 6, σ = 1.4
Used AI for Programming before [Only A & SA]	Yes: 30 (75%), No: 10 (25%)

used non-parametric tests, i.e. the Kruskal-Wallis test [26] to test H-1 and Wilcoxon ranksum tests [26] for post-hoc testing [26]. All hypotheses were tested using two-tailed tests [26] at the level of α=.05. To control for multiple comparisons, a Bonferroni-Holm [41] correction was applied to the p-values with a correction factor of six due to the six pairwise comparisons between our conditions.

5.1.5 Ethics. With no formal IRB requirement in Germany, our university had no active institutional review board (IRB) during its transition to a new ethics committee within the Faculty of Computer Science, recognizing that existing medical-focused structures were unsuited to computer science research. However, the consent form and study materials were derived from our IRB-approved Study A. We took steps to ensure ethical conduct by complying with the General Data Protection Regulation (GDPR) and clearing data handling with the data protection officer. Participants were informed of the study procedure, data policies, and their rights via a downloadable consent form, could ask questions at any time, and withdraw without consequences or loss of payment. They were also informed of their right to access, review, or delete their data.

5.1.6 Limitations. This study has limitations to consider when interpreting the results.

Recruitment Platform. First, our sample consisted of freelancers, whose practices may vary across platforms and differ from those of company developers, potentially affecting outcomes. Second, survey data may be influenced by social desirability bias: Participants on Upwork might have feared that poor performance, such as insecure password storage, limited proficiency, or failure to verify AI-generated suggestions could reduce future research opportunities.

Impact. As with Study A, the rapid evolution of LLMs raises questions about temporal validity. However, our analysis focuses on developers' behavior when engaging with AI rather than model-specific capabilities. As we outline in Section 3.1.5, such behavioral patterns tend to change more slowly [6, 40, 70] than underlying model updates, and prior work shows that security-relevant hallucinations and incomplete reasoning persist across versions. For Study B, these considerations similarly imply that the observed developer behaviors remain informative despite ongoing improvements in AI assistants.

6 Study B: Results of the Quantitative Online Study on Password Storage with Freelance Developers

We present solutions and statements of participants by labeling them with N, A, S, and SA from 1 to 20 according to the instruction scenario assigned to them. We report how many participants made statements via qualifiers where necessary to indicate a distribution. We used qualifiers for participants as shown in Figure 3. Table 7 and 8 give an overview of our participants' code submissions, including completion time, functionality, post-task security instructions, and security analysis. Expected task difficulty was comparable across all four groups before participants were assigned to conditions. Median values were identical (md = 4.0), and the mean differences were minimal (N: μ = 4.2, A: μ = 4.4, S: μ = 4.4, SA: μ = 4.1). The distributions show that most participants anticipated a moderate level of difficulty. No group anticipated markedly higher or lower difficulty, indicating that participants entered the study with broadly similar expectations and without systematic bias toward perceiving the task as easier or harder. However, on average the SA group experienced the task less difficult (SA: md = 6.0, σ = 1.1) compared to the other groups (N: md = 5.5, σ = 1.4, A: md = 5.0, σ = 1.1, S: md = 5.0, σ = 1.4).

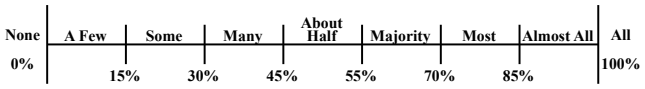


Figure 3: Qualifiers and percentage ranges used to report qualitative results (adopted from Amft et al. [3]).

6.1 H-1: Effect of Instructions on Participants' Security Score

Table 9 outlines security scores across all groups, including their initial and final submissions. The former refers to all participants' first submissions (prior to receiving a post-task security instruction for those who received one), while the latter encompasses the most recent submission from all participants. Considering all final submissions, a total of 69 (86.3%) participants used a hashing function, while eight (10%) opted for Base64 encoding. Across all groups,

Table 7: Evaluation of the functionality and security of participants' solutions. N=Non-Instructed, A=AI-Instructed

Bold: Participants who initially submitted a plaintext solution and received the additional post-task security instruction.

Participant ID	Time (hh:mm)	Functionality	Post-Task Security Instruction	Resubmit upon Post-Task Security Instruction	Used AI	Security					Total (7)
						Hashing				Salt	
						Hashfunction (at most +2)	Length (bits) (+1 if ≥ 160)	Iteration count (at most +1)	Used library (at most +2)	Length (bits) (+1 if ≥ 32)	
N1	06:08	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**DigestUtils**	**128**	**0/5**
N2	04:59	✓	✗	-	✗	BCrypt	192	65536	FavreBCrypt	128	6
N3	06:00	✓	✓	✓	✗	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
N4	01:42	✓	✓	✓	✗	**BCrypt**	**192**	**4096**	**FavreBCrypt**	**128**	**0/6**
N5	07:00	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**4096**	**FavreBCrypt**	**128**	**0/6**
N6	47:32	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**MessageDigest**	**-**	**0/2**
N7	09:11	✓	✗	-	✗	BCrypt	192	1024	MindrotBCrypt	128	6
N8	01:34	✓	✓	✓	✗	**PBKDF2WithHmacSHA1**	**128**	**65536**	**SecretKeyFactory**	**128**	**0/5**
N9	04:51	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
N10	02:32	✓	✓	✓	✗	**BCrypt**	**192**	**1024**	**Spring Security**	**128**	**0/6**
N11	04:10	✓	✗	-	✗	BCrypt	192	1024	MindrotBCrypt	128	6
N12	06:44	✗	✗	-	ChatGPT	-	-	-	-	-	-
N13	02:10	✓	✓	✓	✗	**Base64**	**-**	**-**	**Base64Util**	**-**	**0/0**
N14	02:10	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**MessageDigest**	**128**	**0/4**
N15	04:14	✓	✓	✓	✗	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
N16	05:03	✓	✓	✓	**Microsoft Copilot**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
N17	03:50	✓	✓	✓	✗	**BCrypt**	**192**	**1024**	**Spring Security**	**128**	**0/6**
N18	04:24	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
N19	02:26	✓	✗	-	ChatGPT	BCrypt	192	4096	FavreBCrypt	128	6
N20	04:49	✓	✓	✓	**ChatGPT**	**PBKDF2WithHmacSHA1**	**128**	**65536**	**PBEKeySpec**	**128**	**0/5**
A1	07:17	✓	✓	✓	**Codeium**	**SHA-256**	**256**	**1**	**MessageDigest**	-	**0/2**
A2	03:42	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
A3	07:02	✓	✓	✗	**ChatGPT**	**Base64**	-	-	**PgBase64**	-	**0/0**
A4	19:12	✓	✓	✓	**Tabnine**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
A5	07:45	✓	✓	✗	✗	**Base64**	-	-	**PgBase64**	-	**0/0**
A6	35:22	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
A7	02:52	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**DigestUtils**	-	**0/2**
A8	06:53	✓	✓	✓	**ChatGPT**	**PBKDF2WithHmacSHA256**	**128**	**10000**	**SecretKeyFactory**	**128**	**0/5**
A9	04:17	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**4096**	**FavreBCrypt**	**128**	**0/6**
A10	01:38	✓	✓	✓	**ChatGPT**	**SHA-256**	**256**	**1**	**DigestUtils**	-	**0/2**
A11	06:51	✗	✗	-	ChatGPT	-	-	-	-	-	-
A12	03:11	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**MessageDigest**	-	**0/2**
A13	01:54	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**Spring Security**	**128**	**0/6**
A14	01:53	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
A15	03:12	✓	✓	✓	**ChatGPT**	**MD5**	**128**	**1**	**MessageDigest**	-	**0/1**
A16	10:05	✓	✓	✓	✗	**SHA-256**	**256**	**1**	**MessageDigest**	-	**0/2**
A17	03:24	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
A18	48:47	✓	✓	✓	✗	**Base64**	-	-	**JavaBase64**	-	**0/0**
A19	05:16	✓	✗	-	Gemini	BCrypt	192	1024	Spring Security	-	4
A20	24:38	✓	✓	✓	**Gemini**	**BCrypt**	**192**	**1024**	**Spring Security**	**128**	**0/6**

BCrypt was the most commonly employed hashing function, used by 43 (53.8%) participants. In Group N, three participants utilized SHA-256, and two used PBKDF2. In Group A, SHA-256 was the second most popular choice (6), followed by MD5 and PBKDF2, each employed once. In Group S, four participants chose SHA-256, two used PBKDF2, and one used SHA-512 with the latter being the only instance of its use across the entire study. In Group SA, SHA-256 was implemented three times, MD5 twice, and PBKDF2 once.

Focusing on the effects of our four instruction scenarios, we compared security scores of initial submissions across the four groups using the Kruskal-Wallis Test due to non-normal score distribution (see Appendix B.1.1). Bonferroni-corrected results revealed significant differences between S and A (r_b = 0.75, p = 0.000092), SA and A (r_b = 0.63, p = 0.001511), S and N (r_b = 0.53, p = 0.005542), and SA and N (r_b = 0.44, p = 0.048280), showing that security instructions and the combination of security and AI instructions significantly improved security scores. Notably, the combination of security and AI instructions (SA) proved effective, suggesting that the AI instruction did not diminish the effectiveness of the security instruction.

Despite participants across all groups rating their understanding of security concepts (Q7, μ = 5.7, md = 6.0, σ = 1.2) and knowledge about secure password storage (Q12, μ = 5.3, md = 6.0, σ = 1.4) as rather positive (see Appendix B.1.2), about half of the participants demonstrated misconceptions about password storage security.

Some participants inaccurately referred to hashing algorithms as encryption methods or used the terms interchangeably (N = 6, A = 12, S = 9, SA = 13). A few participants believed that encryption provides secure password protection (N = 1, A = 2, S = 2, SA = 3), e.g., using AES. A few participants mistakenly believed that insecure algorithms or practices provide adequate security for passwords (N = 1, A = 3, S = 2, SA = 3). Similarly, a few considered Base64 encoding to be a secure method. A few participants demonstrated misconceptions about salting practices, using insecure methods, such as using the username as the salt.

From the security-instructed groups (S and SA), 31 participants (77.5%) reported they would have stored passwords securely even without explicit instructions (Q17). Of those instructed to use AI assistants (A and SA), 28 participants used AI assistants (see Appendix B.1.5). Of these, 19 (67.9%) indicated they would have independently chosen to use AI assistants, while nine (32.1%) would not.

Table 8: Evaluation of the functionality and security of participants' solutions. S=Security-Instructed, SA=Security-AI-Instructed

Bold: Participants who initially submitted a plaintext solution and received the additional post-task security instruction.

Participant ID	Time (hh:mm)	Functionality	Post-Task Security Instruction	Resubmit upon Post-Task Security Instruction	Used AI	Security					Total (7)
						Hashfunction (at most +2)	Hashing Length (bits) (+1 if ≥ 160)	Iteration count (at most +1)	Used library (at most +2)	Salt Length (bits) (+1 if ≥ 32)	
S1	05:01	✓	✓	✗	**ChatGPT, Perplexity**	**Base64**	-	-	**PgBase64**	-	**0/0**
S2	06:37	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
S3	07:05	✓	✗	-	✗	PBKDF2WithHmacSHA512	512	1000	PBEKeySpec	4096	5.5
S4	01:59	✓	✗	-	✗	BCrypt	192	1024	MindrotBCrypt	128	6
S5	03:37	✓	✗	-	✗	SHA-256	256	1	MessageDigest	128	3
S6	04:07	✓	✗	-	ChatGPT	BCrypt	192	4096	FavreBCrypt	128	6
S7	03:40	✓	✗	-	✗	PBKDF2WithHmacSHA1	128	65536	PBEKeySpec	128	5
S8	04:02	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
S9	04:21	✓	✗	-	ChatGPT	BCrypt	192	4096	FavreBCrypt	128	6
S10	04:53	✓	✗	-	Gemini	BCrypt	192	1024	Spring Security	128	6
S11	04:09	✓	✗	-	✗	BCrypt	192	1024	MindrotBCrypt	128	6
S12	04:33	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
S13	03:40	✓	✗	-	ChatGPT	BCrypt	192	4096	MindrotBCrypt	128	6
S14	01:36	✓	✗	-	GitHub Copilot	BCrypt	192	1024	MindrotBCrypt	128	6
S15	03:51	✓	✗	-	✗	-	-	-	-	-	0
S16	06:00	✓	✓	✓	✗	**BCrypt**	**192**	**4096**	**MindrotBCrypt**	**128**	**0/6**
S17	04:03	✓	✗	-	✗	SHA-512	512	1	MessageDigest	-	2
S18	06:58	✓	✓	✓	**ChatGPT**	**BCrypt**	**192**	**1024**	**Spring Security**	**128**	**0/6**
S19	01:58	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
S20	12:26	✓	✗	-	✗	SHA-256	256	100000	JasyptEncryptor	128	6
SA1	05:53	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
SA2	05:14	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
SA3	05:21	✓	✗	-	ChatGPT	SHA-256	256	1	MessageDigest	-	2
SA4	02:56	✓	✗	-	ChatGPT	MD5	128	-	MessageDigest	-	1
SA5	04:04	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA6	04:52	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA7	10:18	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA8	04:45	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA9	08:11	✓	✓	✓	✗	**PBKDF2WithHmacSHA256**	**256**	**65536**	**PBEKeySpec**	**80**	**0/5**
SA10	04:17	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA11	03:39	✓	✗	-	Microsoft Copilot	BCrypt	192	64	PgCrypto	128	5
SA12	02:45	✓	✗	-	Groq	BCrypt	192	1024	MindrotBCrypt	128	6
SA13	01:27	✓	✓	✗	✗	**Base64**	-	-	**PgBase64**	-	**0/0**
SA14	02:25	✓	✓	✗	✗	**Base64**	-	-	**PgBase64**	-	**0/0**
SA15	09:00	✓	✓	✓	✗	**BCrypt**	**192**	**1024**	**MindrotBCrypt**	**128**	**0/6**
SA16	05:27	✓	✗	-	ChatGPT	BCrypt	192	1024	MindrotBCrypt	128	6
SA17	02:00	✓	✗	-	GitHub Copilot	BCrypt	192	1024	MindrotBCrypt	128	6
SA18	05:35	✓	✗	-	✗	BCrypt	192	1024	Spring Security	128	6
SA19	44:03	✓	✓	✓	✗	**MD5**	**128**	-	**MessageDigest**	-	**0/1**
SA20	17:48	✓	✓	✗	✗	**Base64**	-	-	**JavaBase64**	-	**0/0**

Table 9: Group security scores of initial and final submissions. Up to 7 points. $_{\text{initial}}$=initial submission, $_{\text{final}}$=final submission.

$\mathbf{N}_{\text{initial}}$	σ = 2.5	md = 0.0, μ = 1.3
$\mathbf{N}_{\text{final}}$	σ = 1.6	md = 6.0, μ = 5.2
$\mathbf{A}_{\text{initial}}$	σ = 1.0	md = 0.0, μ = 0.3
$\mathbf{A}_{\text{final}}$	σ = 2.4	md = 2.0, μ = 3.4
$\mathbf{S}_{\text{initial}}$	σ = 2.5	md = 5.8, μ = 4.0
$\mathbf{S}_{\text{final}}$	σ = 2.2	md = 6.0, μ = 4.6
$\mathbf{SA}_{\text{initial}}$	σ = 2.8	md = 3.5, μ = 3.3
$\mathbf{SA}_{\text{final}}$	σ = 2.5	md = 5.5, μ = 3.9

H-1 – Summary

Security instructions and the combination of security and AI instructions led to significantly more secure solutions. Notably, the latter combination of instructions demonstrated that the AI instruction did not compromise the effect of the security instruction. This confirms Study A's observation that participants in the combined instruction condition submitted the most secure solutions and extends pre-AI results [57, 59] by demonstrating that explicit security reminders remain effective for password storage tasks even in AI-assisted workflows.

6.2 H-2: Effect of Reported Usage of AI Assistants During the Study on Participants' Security Score

The majority of participants (46, 57.5%) reported to have used AI assistants to solve the task. Table 15 in our Appendix outlines the usage of the specific AI assistants as reported by participants in our survey. The most relevant AI assistant used during the task was ChatGPT (34) which was mainly used for general support and code generation. Still, many participants (34, 42.5%) did not use AI assistants to solve the task. From these, some participants (18, 22.5%) indicated they would have used AI assistants if it had not been a study, while some participants (16, 20%) reported they would not have. About half of our participants shared diverse reasons for using and not using AI assistants during our study. These reasons encompassed workflow-related factors (e.g., increased efficiency when using AI assistants), environmental considerations (e.g., task was too simple to warrant the use of AI assistants), and personal habits (e.g., regular reliance on AI assistants).

Overall, submissions involving AI achieved higher security scores than those without for both initial and final submissions (see Appendix B.1.3), with the exception of Group N for initial scores. Improvements from initial to final submissions were larger when AI was used, except in Group S, suggesting post-task instructions were more effective with AI-assisted submissions. In Group SA, participants who used AI received no post-task instruction. In Group A, all participants who did not use AI submitted plaintext solutions (see Table 8). Among the seven non-AI users in Group SA, all but one (SA18) also submitted plaintext. While this might suggest that participants ignored all instructions, half of them (SA13, SA19, SA20) indicated they believed their initial plaintext submissions were secure and chose not to resubmit.

Across groups, 41 (51.3%) participants received a post-task security instruction after submitting plaintext solutions (N: 15, A: 17, S: 3, SA: 7). Of these, 35 (85.4%) resubmitted their solutions, while six (14.6%) chose not to resubmit (A3, A5, S1, SA13, SA14, SA20), resulting in a security score of 0. A3, A5, and SA20 were confident the password was "encrypted." In fact, it was encoded with Base64, as seen in Table 7 and 8. Further, S1 believed the database had "encrypted" the password already, SA13 responded that the password was "encoded," while SA14 believed Base64 to be secure. Post-task instructions increased mean security scores notably for Groups N (3.9) and A (3.1;), indicating that post-task instructions improve security once functionality is in place (see Table 9). Final scores for N (5.2) and A (3.4) were similar to the initial scores for S (4.6) and SA (3.9), suggesting pre- and post-task instructions can be comparably effective in enhancing security scores. Compared to the pre-AI study B [57], Groups N and S in our study achieved higher mean security scores at both initial (0.4 and 1.8) and final (3.0 and 2.1) submission stages, indicating that AI assistance contributed to better security score.

6.2.1 Security Score. **Pre-Task Security Instruction.** We employed the Wilcoxon ranksum test due to non-normal score distribution. We observed a significant difference between participants who reported to have used AI assistants during the task and those who did not in the group SA (r_b = 0.80, p = 0.0142), revealing that the usage of AI assistants in addition to security and AI prompts significantly improved security scores.

Post-Task Security Instruction. For the smaller subset of participants who received only a post-task security instruction (groups N and A), we limited our analysis to those participants only, as groups S and SA had already received a pre-task security instruction. We used the Wilcoxon ranksum test due to non-normal score distribution and observed significant difference in group A (r_b = 0.51, p = 0.03501), indicating that when the security instruction is provided *after* the task, AI users were more likely to produce a secure revision than non-AI users.

6.2.2 Developers' Perception of AI Assistance. About half of the participants viewed AI assistants as incapable of generating secure suggestions, with many citing security limitations. Conversely, a majority considered them capable, about half noting potential to provide valuable insights, suggest *"the industry standard"* — [SA17], or deliver trusted security solutions. Still, many emphasized the need for thorough review. A majority stressed verifying AI-generated security suggestions through *"human verification"* — [S14] and cross-checking with *"trusted security resources"* — [A6], forums, and experts. This need for scrutiny, shared by some, including a few highlighting its importance in security contexts, was linked to careful implementation (N4), caution (S14), supplementary use (A6), and accountability (SA5). While participants generally trusted AI suggestions (AI31, μ = 3.7, md = 4.0, σ = 1.4), about half expressed distrust, with many specifically doubting AI-generated security outputs (e.g., *"[s]ecurity [...] cannot be entrusted to only AI"* — [A5]). Some expressed conditional trust, with a few depending on context and a few on user knowledge. Finally, a few emphasized self-reliance (e.g., *"I trust myself more for these aspects"* — [N5]).

H-2 – Summary

Our participants who used AI assistants achieved higher mean security scores than participants in the pre-AI study B, demonstrating improved password security practices with the support of AI assistants. Using AI assistants for secure password storage when paired with (pre- and post-task) security and AI instructions, resulted in a significant improvement in security scores compared to participants who did not use AI assistants, highlighting enhanced password security practices through the usage of AI assistants when guided appropriately. Similar to Study A, participants voiced concerns about accuracy and the need for verification highlighted a cautious approach to their use.

7 Discussion

Across both studies, we observed a trend that extends pre-AI work on secure password storage. Before the widespread adoption of AI assistants, developer studies showed that participants rarely implemented secure password storage without explicit security instructions [57, 59]. Our results replicated this effect in an AI-mediated landscape: In both the student lab (Study A) and the freelancer study (Study B), participants largely neglected secure password storage unless security was explicitly requested. At the same time, our data reveal a behavioral shift compared to pre-AI studies: Developers now routinely integrate AI assistants into their workflow when implementing password storage, even without being instructed to do so. Together, these patterns indicate that the introduction of AI assistants has not altered the requirement-driven nature of developers' security behavior. Instead, AI reshapes how developers pursue a solution by shifting their workflow toward copy/paste–oriented patterns while leaving underlying decision logics unchanged. This suggests a behavioral trend underlying developers' interactions with AI: When security is not framed as a task requirement, developers direct AI queries toward functionality rather than security, resulting in insecure implementations despite access to security-relevant suggestions.

7.1 Comparison with Past Research

Our findings extend pre-AI instruction research on secure password storage [57, 59] by showing that instructing remains a necessary condition for secure implementations even in AI-mediated workflows. In contrast to previous work, where instructions primarily

redirected developers' own coding actions, our results suggest a broader mechanism: Instructions now also shape the *interaction with AI assistants*. When security is explicitly required, developers channel their queries toward security-oriented solutions and copy/paste secure examples; when it is not, they engage AI only for functionality, reproducing the pre-AI pattern of neglecting security. This indicates that instructions do not lose relevance in the age of AI, rather, their influence widens to govern both developer decision-making and developer–AI communication.

Compared to previous work that highlighted the risks of insecure software development with AI assistants [63, 67, 69], our findings for H-2, RQ1, and RQ2 showed that AI assistants contribute positively to secure password storage practices when paired with explicit security and AI usage instructions. Participants who used AI assistants in conjunction with these instructions achieved higher security scores compared to non-AI users. These results align with the observations of Asare et al. [5], who identified potential security benefits of AI tools in addressing complex tasks. Extending their insights, we explored the challenging task of secure password storage requiring cryptographic knowledge, illustrating how AI tools can support developers in navigating security-critical implementations. However, while our participants achieved higher security scores with AI assistance, these tools can still produce insecure code without proper guidance [60, 67, 79]. Our participants in Study A and B underscored the critical role of prompts in guiding AI outputs toward secure practices, reinforcing the notion that AI assistants must be used with caution (see Sections 4.2.2 and 6.2.2). Our qualitative data further showed that participants' avoidance of secure third-party libraries stemmed from usability hurdles rather than uncertainty about permissions. Screen recordings indicate that participants attempted to import secure libraries but, after facing configuration or dependency errors, shifted to implementations requiring no additional setup (see Section 4.1.4). This convenience-driven behavior aligns with pre-AI observations that developers prioritize reducing friction in their workflow [45], and it illustrates how usability challenges can redirect AI-assisted coding toward less secure solutions.

One participant in study A achieved the highest score of 7 points using AI assistance, a result not seen in the pre-AI studies [57, 59]. However, this participant stood out by leveraging multiple resources and applying prior knowledge (see Section 4.1.2), emphasizing the importance of expertise and validation when using AI-generated solutions. On average, in study B the SA group perceived the task less difficult compared to the other groups N, A, and S (see Section 6), suggesting that a clear definition of requirements and the use of AI assistants might reduce cognitive load [25, 76, 89] and provide participants with more guidance when implementing secure password storage. The efficiency provided by AI assistance could potentially free up cognitive and time resources, enabling developers to dedicate greater attention to implementing and verifying secure practices. While AI tools can enhance secure practices when appropriately guided, they do not replace the need for explicit security instructions. Instead, the combination of AI assistance and security instructions provides a complementary strategy for improving security outcomes.

7.2 Implications and Future Research

Our findings show that in AI-assisted development, instructions no longer affect only developers' coding actions, as documented in pre-AI work, but also the *interaction between developers and AI assistants*. Instructions determine what developers ask AI tools for, what examples they copy/paste, and which solutions they pursue. As a result, instructions shape not just decisions, but the entire AI-mediated workflow. Additionally, participants varied in how they performed prompting: Some inserted the instructions verbatim, others reformulated them, and some provided only partial task context. These prompting choices influenced the AI's output and, in turn, the security of the resulting code. Thus prompting becomes a socio-technical practice shaped by organizational requirements, developers' interpretations, and AI assistance behavior, reflecting how developers translate task instructions into prompts within their workflow.

7.2.1 Implications for Secure Development Workflows. Organizations should continue to provide explicit security requirements even in AI-enhanced workflows, because instructing remains the primary mechanism that directs developers' goals and, in turn, their prompts to AI assistants. Our results show that without explicit security instructions, developers consistently queried AI assistants for functional guidance, reproducing the same insecure outcomes as in pre-AI studies. Security instructions therefore remain an organizational responsibility: They determine whether security enters the developer–AI interaction at all. At the same time, prompt engineering guidance becomes increasingly important in environments where explicit prompts cannot be consistently guaranteed. As prior work has shown, carefully structured prompts can mitigate insecure AI suggestions [68, 79], and our findings reinforce that such structure is necessary for secure outcomes.

7.2.2 Supporting Developers in Assessing AI-Generated Code. Organizations should provide explicit guidelines for assessing AI outputs. Even though participants in both studies acknowledged the need to verify AI-generated code, very few actually evaluated the correctness or security of snippets before integrating them (see Sections 4.2.2 and 6.2.2). This gap between awareness and practice highlights the need for actionable validation guidance—such as lightweight checklists, warnings embedded into task descriptions, or secure code review protocols—particularly for security-critical tasks such as password storage. Guidance can help developers balance the efficiency benefits of AI assistance with the need for independent validation.

7.2.3 Reducing Cognitive Load Without Causing Over-Reliance. AI assistance demonstrably reduced perceived task difficulty in Study B under security requirements, aligning with prior work on cognitive load reduction. By handling repetitive or complex tasks, AI assistants free up cognitive [25, 76, 89] and time resources [8, 87]. However, this cognitive relief did not automatically lead to secure outcomes in groups without security instructions. This suggests that AI-driven efficiency can support secure behavior only when security is already framed as part of the task. To optimize this balance, it is essential to design workflows that leverage AI efficiency without fostering over-reliance [88]. Future research should examine how to balance cognitive relief with the risk of over-reliance, and

develop workflows that preserve developers' independent problem-solving skills. Additionally, adaptive AI systems tailored to varying expertise levels could be explored to provide optimal support for both novice and experienced developers. For novice developers, AI systems could provide simplified explanations and context-specific examples to build foundational understanding, while offering advanced, nuanced suggestions for experienced developers to maintain efficiency.

7.2.4 Designing AI Systems to Address Misconceptions and Insecure Defaults. AI assistants should be improved to address persistent misconceptions. Developers' misconceptions about security practices, such as conflating encryption and hashing terminology, persisted even after completing the task (see Section 6.1). AI assistants could address these knowledge gaps in user prompts and provide corrective explanations. This would transform AI tools into active educators, empowering developers with a deeper understanding of security concepts rather than just offering passive guidance. While it is important to acknowledge that AI tools can generate inaccurate information [60, 67, 92], they have become an integral part of developers' workflows [82, 83]. This makes it critical to embed mechanisms that highlight insecure suggestions, prioritize secure libraries, and warn when a developer requests insecure implementations. These capabilities directly address the tendencies we observed when participants rejected library-based implementations and received less secure alternatives in return.

7.2.5 Improving Access to External Security Resources. AI environments should improve access to reliable external resources. Participants who used Microsoft Copilot benefited from default-provided source links, which enabled them to validate recommendations and ultimately arrive at more secure solutions. By contrast, ChatGPT's lack of default linking required developers to manually request references. Ensuring seamless access to documentation and authoritative guidance can help developers move beyond copy/paste adoption and support a deeper understanding of secure practices.

While this work focused on the security-critical task of password storage in a database, we anticipate that similar trends are likely to appear in comparable security use cases. However, further research is required to replicate the findings in other security use cases in order to generalize them. Overall, our results indicated that secure AI-assisted development requires a combination of organizational practices (explicit instructions, validation guidance) and technical improvements in AI assistants (robust secure defaults, detection of misconceptions, transparent linking). Future research should explore (1) adaptive prompting systems that guide developers toward secure interactions with AI, (2) mechanisms for detecting and correcting conceptual misunderstandings in real time, (3) strategies for leveraging cognitive relief without promoting over-reliance, and (4) AI-generated warnings triggered by insecure developer inputs or fallback requests. As AI continues to shape everyday development workflows, aligning human behavior, organizational processes, and AI design will be essential for reliably achieving secure outcomes.

8 Conclusion

Our work examined whether established instruction interventions remain effective in the era of AI-assisted software development. Building on pre-AI studies of password storage [57, 59], we conducted a qualitative lab study with 21 students (Study A) and a quantitative follow-up with 80 freelance developers (Study B). Study A revealed how participants approached AI assistance in a password storage task: Convenience often outweighed security, copy/paste strategies dominated, and skepticism about AI-generated security advice persisted. These behaviors underscored the need for explicit interventions. Study B then confirmed this statistically: Security instructions significantly improved outcomes, and AI use strengthened instruction effects. Together, these findings extend pre-AI work on security instructions by showing that such interventions remain essential even as development practices shift toward AI assistance. While prior research highlighted the risks of insecure AI outputs [60, 67, 79], our results extend this perspective: AI assistance does not ensure security, but when paired with targeted instructions, it can reinforce secure practices rather than undermine them.

Acknowledgments

We thank our anonymous reviewers for helping us improve our paper. We are additionally grateful to Marco Gutfleisch for facilitating recruitment through Upwork for this study. 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, Simson Garfinkel, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2017. Comparing the Usability of Cryptographic APIs. In *2017 IEEE Symposium on Security and Privacy (SP)*. 154–171. doi:10.1109/SP.2017.52

[2] 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. doi:10.1109/SP.2016.25 ISSN: 2375-1207.

[3] Sabrina Amft, Sandra Höltervennhoff, Rebecca Panskus, Karola Marky, and Sascha Fahl. 2024. Everyone for Themselves? A Qualitative Study about Individual Security Setups of Open Source Software Contributors. In *In 45th IEEE Symposium on Security and Privacy, IEEE S&P 2024, May 20-23, 2024.* IEEE Computer Society. https://www.ieee-security.org/TC/SP2024/accepted-papers.html

[4] Argon 2025. GitHub - Argon2 Binding for the JVM. https://github.com/phxql/argon2-jvm. Accessed: September 2025.

[5] Owura Asare, Meiyappan Nagappan, and N. Asokan. 2024. A User-centered Security Evaluation of Copilot. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 158, 11 pages. doi:10.1145/3597503.3639154

[6] Hala Assal, Srivathsan G Morkonda, Muhammad Zaid Arif, and Sonia Chiasson. 2025. Software security in practice: knowledge and motivation. *Journal of Cybersecurity* 11, 1 (03 2025), tyaf005. arXiv:https://academic.oup.com/cybersecurity/article-pdf/11/1/tyaf005/62386596/tyaf005.pdf doi:10.1093/cybsec/tyaf005

[7] Baeldung 2025. Baeldung - Hashing a Password in Java. https://www.baeldung.com/java-password-hashing. Accessed: September 2025.

[8] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 78 (apr 2023), 27 pages. doi:10.1145/3586030

[9] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. 2015. Argon2: the memory-hard function for password hashing and other applications. *Tech. rep., Password Hashing Competition (PHC)*. (2015).

[10] Richard E Boyatzis. 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development.* Sage Publications, Inc.

[11] JEPS Bulletin. 2019. Exploratory and Confirmatory Hypothesis Testing. https://blog.efpsa.org/2019/11/20/exploratory-and-confirmatory-hypothesis-testing/ Accessed: September 2025.

[12] Business Insider 2025. Microsoft pushes staff to use internal AI tools more, and may consider this in reviews. 'Using AI is no longer optional.'. https://www.businessinsider.com/microsoft-internal-memo-using-ai-no-longer-optional-github-copilot-2025-6. Accessed: September 2025.

[13] Calendly 2025. Calendly. https://calendly.com/. Accessed: September 2025.

[14] ChatGPT 2020. OpenAI. OpenAI ChatGPT. https://chat.openai.com/chat. Accessed: September 2025.

[15] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46. arXiv:https://doi.org/10.1177/001316446002000104 doi:10.1177/001316446002000104

[16] The Git Development Community. [n. d.]. Git - Distributed Version Control System. https://git-scm.com/. Version 2.44.0.windows.1, Accessed: September 2025).

[17] Craigslist 2025. Craigslist. https://craigslist.org. Accessed: September 2025.

[18] Anastasia Danilova, Alena Naiakshina, Johanna Deuter, and Matthew Smith. 2020. Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 165–183. https://www.usenix.org/conference/soups2020/presentation/danilova

[19] Dinei Florencio, Cormac Herley, Paul C. van Oorschot. 2014. An Administrator's Guide to Internet Password Research. *Proceedings of the USENIX conference on Large Installation System Administration* (2014). https://www.usenix.org/system/files/conference/lisa14/lisa14-paper-florencio.pdf

[20] Discord 2025. Discord. https://discord.com/. Accessed: September 2025.

[21] Eclipse Foundation. 2025. Eclipse - The Eclipse Foundation Open Source Community Website. https://www.eclipse.org/ Accessed: September 2025.

[22] Eclipse Marketplace 2025. Copilot4Eclipse. https://marketplace.eclipse.org/content/copilot4eclipse. Accessed: September 2025.

[23] Paul D. Ellis. 2010. *The Essential Guide to Effect Sizes. Statistical Power, Meta-Analysis, and the Interpretation of Research Results.* Cambridge University Press.

[24] Franz Faul, Edgar Erdfelder, Albert-Georg Lang, and Axel Buchner. 2007. G*Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences. *Behavior Research Methods* 39, 2 (May 2007), 175–191. doi:10.3758/BF03193146

[25] Lijuan Feng. 2024. Investigating the Effects of Artificial Intelligence-Assisted Language Learning Strategies on Cognitive Load and Learning Outcomes: A Comparative Study. *Journal of Educational Computing Research* (2024). doi:10.1177/07356331241268349

[26] Andy Field. 2013. *Discovering Statistics Using IBM SPSS Statistics* (4th ed.). Sage Publications Ltd.

[27] Felix Fischer, Konstantin Bottinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security. 121–136. doi:10.1109/SP.2017.31

[28] Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. 2013. *Statistical Methods for Rates and Proportions.* John Wiley & Sons.

[29] GitHub 2025. eclipse-github-copilot-integration. https://github.com/masecla22/eclipse-github-copilot-integration. Accessed: September 2025.

[30] GitHub 2025. Unofficial GitHub Copilot integration with Eclipse. https://github.com/vgcpge/eclipse.copilot. Accessed: September 2025.

[31] GitHub Copilot 2025. Getting free access to Copilot Pro as a student, teacher, or maintainer. https://docs.github.com/en/copilot/managing-copilot/managing-copilot-as-an-individual-subscriber/managing-your-github-copilot-pro-subscription/getting-free-access-to-copilot-pro-as-a-student-teacher-or-maintainer. Accessed: September 2025.

[32] GitHub Copilot 2025. GitHub. GitHub Copilot. https://github.com/features/copilot. Accessed: September 2025.

[33] GitHub Copilot Changelog 2025. Changelog. https://github.blog/changelog/label/copilot/. Accessed: November 2025.

[34] GPT-5 2025. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/. Accessed: November 2025.

[35] Paul A Grassi, James L Fenton, and M Elaine. 2017. Newton, Ray A Perlner, Andrew R Regenscheid, William E Burr, Justin P Richer, Naomi B Lefkovitz, Jamie M Danker, Yee-Yin Choong, Kristen K Greene, and Mary F Theofanos. 2017. Digital identity guidelines: authentication and lifecycle management. *Digital identity guidelines: Authentication and lifecycle management. Special Publication (NIST SP)-800-63B* (2017).

[36] Paul A Grassi, Michael E Garcia, and James L Fenton. 2017. Draft nist special publication 800-63-3 digital identity guidelines. *National Institute of Standards and Technology, Los Altos, CA* (2017).

[37] Matthew Green and Matthew Smith. 2016. Developers are Not the Enemy!: The Need for Usable Security APIs. *IEEE Security & Privacy* 14, 5 (2016), 40–46. doi:10.1109/MSP.2016.111

[38] George Hatzivasilis, Ioannis Papaefstathiou, and Charalampos Manifavas. 2015. Password Hashing Competition - Survey and Benchmark. *IACR Cryptol. ePrint Arch.* 2015 (2015), 265. https://api.semanticscholar.org/CorpusID:15654811

[39] Hibernate Community. 2025. Hibernate - Relational Persistence for Idiomatic Java. https://hibernate.org/ Accessed: September 2025.

[40] Jonas Hielscher, Annette Kluge, Uta Menges, and Angela Sasse. 2021. "Taking out the Trash": Why Security Behavior Change requires Intentional Forgetting. doi:10.1145/3498891.3498902

[41] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.

[42] Kasper Hornbæk, Søren S. Sander, Javier Andrés Bargas-Avila, and Jakob Grue Simonsen. 2014. Is once enough? on the extent and content of replications in human-computer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3523–3532. doi:10.1145/2556288.2557004

[43] Saki Imai. 2022. Is GitHub copilot a substitute for human pair-programming? An Empirical Study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ACM Digital Library)*, Matthew B. Dwyer (Ed.). Association for Computing Machinery, New York,NY,United States, 319–321. doi:10.1145/3510454.3522684

[44] Joseph Bonneau and Sören Preibusch. 2010. The password thicket: technical and market failures in human authentication on the web. (2010). https://web.archive.org/web/20120110093533/http://weis2010.econinfosec.org/papers/session3/weis2010_bonneau.pdf

[45] Iacovos Kirlappos, Simon Parkin, and M. Angela Sasse. 2015. "Shadow security" as a tool for the learning organization. *SIGCAS Comput. Soc.* 45, 1 (Feb. 2015), 29–37. doi:10.1145/2738210.2738216

[46] Kleinanzeigen 2025. Kleinanzeigen. https://www.kleinanzeigen.de/. Accessed: September 2025.

[47] S. Kruger, J. Spath, K. Ali, E. Bodden, and M. Mezini. 2021. CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. *IEEE Transactions on Software Engineering* 47, 11 (nov 2021), 2382–2400. doi:10.1109/TSE.2019.2948910

[48] T. Lunsford and B. Lunsford. 1995. The Research Sample, Part I: Sampling. *JPO Journal of Prosthetics and Orthotics* 7 (1995), 17A. doi:10.1097/00008526-199500730-00008

[49] Manic Time 2025. Time Tracker Managment Tracking Software. https://www.manictime.com/ Accessed: September 2025.

[50] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 72 (nov 2019), 23 pages. doi:10.1145/3359174

[51] Microsoft 2024. AI-powered success -— with more than 1,000 stories of customer transformation and innovation. https://www.microsoft.com/en-us/microsoft-cloud/blog/2025/07/24/ai-powered-success-with-1000-stories-of-customer-transformation-and-innovation/. Accessed: August 2025.

[52] Microsoft Corporation. 2025. Azure Virtual Machines - Cloud Computing Services. https://azure.microsoft.com/en-us/products/virtual-machines Accessed: September 2025.

[53] Microsoft Corporation. 2025. Understanding Remote Desktop Protocol. https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol Accessed: September 2025.

[54] MicrosoftCopilot 2025. Microsoft 365 Copilot release notes. https://learn.microsoft.com/en-us/copilot/microsoft-365/release-notes. Accessed: November 2025.

[55] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. 2016. Jumping through hoops: why do Java developers struggle with cryptography APIs?. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 935–946. doi:10.1145/2884781.2884790

[56] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. 2020. On Conducting Security Developer Studies with CS Students: Examining a Password-Storage Study with CS Students, Freelancers, and Company Developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Honolulu</city>, <state>HI</state>, <country>USA</country>, </conf-loc>) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376791

[57] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. 2019. "If you want, I can store the encrypted password": A Password-Storage Field Study with Freelance 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, 1–12. doi:10.1145/3290605.3300370

[58] Alena Naiakshina, Anastasia Danilova, and Christian Tiefenau. 2018. Deception Task Design in Developer Password Studies: Exploring a Student Sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018) (SOUPS '18)*. USENIX Association, Baltimore, MD, 297–313.

[59] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. 2017. Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association

for Computing Machinery, New York, NY, USA, 311–328. doi:10.1145/3133956.3134082
[60] Nhan Nguyen and Sarah Nadi. 2022. An Empirical Evaluation of GitHub Copilot's Code Suggestions. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 1–5. doi:10.1145/3524842.3528470
[61] OBS 2025. OBS: Open Broadcaster Software. https://obsproject.com/ Accessed: September 2025.
[62] Philippe Oechslin. 2003. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2729)*. Springer, 617–630. doi:10.1007/978-3-540-45146-4_36
[63] Sanghak Oh, Kiho Lee, Seonhye Park, Doowon Kim, and Hyoungshick Kim. 2023. Poisoned ChatGPT Finds Work for Idle Hands: Exploring Developers' Coding Practices with Insecure Suggestions from Poisoned AI Models. http://arxiv.org/abs/2312.06227 arXiv:2312.06227 [cs].
[64] S. Oh, K. Lee, S. Park, D. Kim, and H. Kim. 2024. Poisoned ChatGPT Finds Work for Idle Hands: Exploring Developers' Coding Practices with Insecure Suggestions from Poisoned AI Models. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 182–182. doi:10.1109/SP54263.2024.00046
[65] Oracle Corporation. 2025. Java Developer Portal - Resources for Java Development. https://dev.java/ Accessed: September 2025.
[66] OWASP 2025. OWASP - Password Storage Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. Accessed: September 2025.
[67] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 754–768. doi:10.1109/SP46214.2022.9833571
[68] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2023. Examining Zero-Shot Vulnerability Repair with Large Language Models. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2339–2356. doi:10.1109/SP46215.2023.10179324
[69] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants? *(CCS '23)*. Association for Computing Machinery, New York, NY, USA, 2785–2799. doi:10.1145/3576915.3623157
[70] Shari Pfleeger, Angela Sasse, and Adrian Furnham. 2014. From Weakest Link to Security Hero: Transforming Staff Security Behavior. *Journal of Homeland Security and Emergency Management* 11 (12 2014). doi:10.1515/jhsem-2014-0035
[71] PostgreSQL Global Development Group. 2025. PostgreSQL - The World's Most Advanced Open Source Relational Database. https://www.postgresql.org/ Accessed: September 2025.
[72] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That It Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* 31, 1, Article 4 (nov 2023), 31 pages. doi:10.1145/3617367
[73] Qualtrics 2025. Qaltrics. https://www.qualtrics.com/. Accessed: September 2025.
[74] Johnny Saldana. 2016. *The Coding Manual for Qualitative Researchers* (3rd ed.). Sage Publications, London, UK.
[75] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2023. Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 2205–2222. https://www.usenix.org/conference/usenixsecurity23/presentation/sandoval
[76] Johanna Schmidhuber, Stephan Schlögl, and Christian Ploder. 2021. Cognitive Load and Productivity Implications in Human-Chatbot Interaction. *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)* (2021), 1–6. doi:10.1109/ICHMS53169.2021.9582445
[77] Raphael Serafini, Asli Yardim, and Alena Naiakshina. 2025. Exploring the Impact of Intervention Methods on Developers' Security Behavior in a Manipulated ChatGPT Study. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 520, 26 pages. doi:10.1145/3706598.3713989
[78] Martin Shepperd, Nemitari Ajienka, and Steve Counsell. 2018. The role and value of replication in empirical software engineering results. *Information and Software Technology* 99 (2018), 120–132. doi:10.1016/j.infsof.2018.01.006
[79] Yong Shi, Nazmus Sakib, Hossain Shahriar, Dan Lo, Hongmei Chi, and Kai Qian. 2023. AI-Assisted Security: A Step towards Reimagining Software Development for a Safer Future. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 991–992. doi:10.1109/COMPSAC57700.2023.00142
[80] Katta Spiel, Oliver L. Haimson, and Danielle Lottridge. 2019. How to do better with gender on surveys: a guide for HCI researchers. *Interactions* 26, 4 (June 2019), 62–65. doi:10.1145/3338283
[81] Stack Overflow 2011. How to hash some String with SHA-256 in Java? https://stackoverflow.com/questions/5531455/how-to-hash-some-string-with-sha-256-in-java. Accessed: September 2025.
[82] Stack Overflow 2023. Stack Overflow - Developer Survey 2023. https://survey.stackoverflow.co/2023/#ai. Accessed: September 2025.
[83] Stack Overflow 2024. Stack Overflow - Developer Survey 2024. https://survey.stackoverflow.co/2024/ai. Accessed: September 2025.
[84] Stackoverflow Thread 2025. Stack Overflow - Java Argon2 Hashing. https://stackoverflow.com/questions/66594009/java-argon2-hashing. Accessed: September 2025.
[85] Catherine Tony, Mohana Balasubramanian, Nicolás E. Díaz Ferreyra, and Riccardo Scandariato. 2022. Conversational DevBots for Secure Programming: An Empirical Study on SKF Chatbot. In *The International Conference on Evaluation and Assessment in Software Engineering 2022 (ACM Digital Library)*, Miroslaw Staron (Ed.). Association for Computing Machinery, New York,NY,United States, 276–281. doi:10.1145/3530019.3535307
[86] Upwork 2015. Upwork Homepage. https://www.upwork.com/. Accessed: September 2025.
[87] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, Simone Barbosa, Cliff Lampe, Caroline Appert, and David A. Shamma (Eds.). ACM, New York, NY, USA, 1–7. doi:10.1145/3491101.3519665
[88] Helena Vasconcelos, Matthew Jörke, Madeleine Grunde-McLaughlin, Tobias Gerstenberg, Michael Bernstein, and Ranjay Krishna. 2022. Explanations Can Reduce Overreliance on AI Systems During Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 7 (2022), 1 – 38. doi:10.1145/3579605
[89] Feng Hsu Wang. 2024. On the Effect of Explainable AI on Programming Learning: A Case Study of using Gradient Integration Technology. *Education & Information Technology* (2024). doi:10.5121/csit.2024.141202
[90] Chamila Wijayarathna and Nalin A. G. Arachchilage. 2018. Why Johnny Can't Store Passwords Securely? A Usability Evaluation of Bouncycastle Password Hashing. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (Christchurch, New Zealand) *(EASE '18)*. Association for Computing Machinery, New York, NY, USA, 205–210. doi:10.1145/3210459.3210483
[91] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle L. Mazurek, and Sascha Fahl. 2017. Security Developer Studies with GitHub Users: Exploring a Convenience Sample. In *SOUPS 2017*. Santa Clara, California, 81–95. https://www.usenix.org/conference/soups2017/technical-sessions/presentation/acar
[92] Burak Yetistiren, Isik Ozsoy, and Eray Tuzun. 2022. Assessing the Quality of GitHub Copilot's Code Generation. In *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering* (Singapore, Singapore) *(PROMISE 2022)*. Association for Computing Machinery, New York, NY, USA, 62–71. doi:10.1145/3558489.3559072

Appendix

A Study A

A.1 AI Usage of our Participants

Table 15 summarizes participants' usage of AI assistants before and during the study. The latter is separated between what participants remembered in the interviews (self-report) and what we observed participants used in the videos during the task (actually). Participants in groups C and S were not specifically primed for AI assistants; consequently, data on the AI assistants they used before the study was not collected during the survey to avoid priming them.

A.2 Study Invitation

Dear Computer Science Students,

We are conducting several scientific studies, in which you can participate and earn 115 euros!

Details: We are looking for motivated Computer Science Students (over 18), who want to take part in a scientific study with the topic of Web Development in Java and to earn some extra money. The goal of the study is to test the usability of different Java web development APIs. In the study you will be asked to implement parts of a web application. Basic knowledge of Java and the IDE Eclipse is required. The study will last 8 hours. Afterwards we will conduct a hort

interview and ask some questions about the tasks. The interview will be audio recorded to facilitate the evaluation of results. A video of your screen will be recorded to evaluate the usability of the APIs. The aim of the study is not to test your knowledge but the usability of the APIs. All data gathered during the study will be anonymized. Anonymized data and quotes may be published as part of a scientific publication. You will be paid 115 euros via PayPal for taking part in the study.

Knowledge required: IDE Eclipse, JAVA

Interested? Please fill in the following questionnaire to register your interest: (Link to Screening Survey on Qualtrics)

We are looking for a good mix of skills, so please fill out the questionnaire as accurately as possible. We are looking for up to 20 participants. Registering your interest does not guarantee participation.

Location of Study: ...

Contact Information: ...

A.3 Screening Survey

- Gender?
  - woman
  - man
  - non-binary
  - prefer not to disclose
  - prefer to self-describe: [text]
- How old are you? [text]
- What is your current main occupation?
  - Student
  - Other: [text]
- Which university are you at? [text]
- In which program are you currently enrolled?
  - Bachelor - Applied Computer Science
  - Master - Applied Computer Science
  - Other: [text]
- Your semester: [text]
- Currently, do you have a part-time job in the field of Computer Science?

[If yes] What part-time job in the field of Computer Science do you have?

- Did you have a part-time job in the field of Computer Science before?

[If yes] What part-time job in the field of Computer Science did you have?

- How familiar are you with Java?
  - 1: Not familiar at all - 7: Very familiar
- How familiar are you with PostgreSQL?
  - 1: Not familiar at all - 7: Very familiar
- How familiar are you with Hibernate?
  - 1: Not familiar at all - 7: Very familiar
- How familiar are you with Eclipse IDE?
  - 1: Not familiar at all - 7: Very familiar

A.4 Interview Guide

A.4.1 Code Security.

- Do you think that you have stored the end-user passwords securely?
- No
  - Why?
  - (C and A Group:) Were you aware that the task needed a secure solution?
- Yes (If yes, further questions below)

Further Questions (If participants believe they have stored the end-user password securely.)

- (C and A Group:) How did you become aware of the necessity of security in this task? At which point did you decide to store the end-user password securely?
- What did you do to store the user password securely? Please name all steps taken in order to store the end-user password securely
- What were the general problems you encountered when implementing secure end-user password storage?
- (S and SA Group:) Do you think you would have stored the end-user passwords securely if you had not been told about it? Please explain your answer.

A.4.2 Security Concepts.

- What is the purpose of hashing?
- What is the purpose of salting?
- Did you hash the end-user password?
- Did you salt the end-user password?
- Did the Web framework JSF support you in storing the end-user password securely? Please explain your answer.
- Did the Web framework JSF prevent you from storing the end-user password securely? Please explain your answer.
- Imagine you were working in a company and you had exactly the same task as you got today with exactly the same task description and time constraint. Would you have solved the task in exactly the same way as you solved it today?
- Did you solve the task functionally?
- Did you solve the task securely?

A.4.3 AI Tools.

- Have you ever used AI tools for programming before the study?
  - Please specify which AI tool(s) you used.
  - What kind of tasks did you use the tool(s) for?
  - How often do you use AI tools?
  - How did AI tools change your programming workflow?
- Did you use any AI tool(s) for solving the task?
  - Yes
    - Please specify which AI tool(s) you used.
    - Are there premium versions of the tool(s) you mentioned?
    - How did you use the tool(s)?
    - How did the tool(s) assist you during the task?
    - Did you receive any functional suggestions from the AI tool(s) during the task?
      - What do you think of these suggestions?
    - Did you use the AI tool(s) to get information about functionality during the task?
      - Why? / Why not?
    - Did you receive any security suggestions from the AI tool(s) during the task?

Table 10: Participants' Usage of AI Assistants for Programming In- and Outside of Study.

	Used before (Pre-Survey)	**Still using** (Pre-Survey)	**Used before** (Interview)	**Used in study** (Interview)	**Actually used in study** (Screen Recording)
N1	N/A	N/A	ChatGPT, Microsoft Copilot	Microsoft Copilot	Microsoft Copilot
N2	N/A	N/A	ChatGPT	No	Microsoft Copilot
N3	N/A	N/A	ChatGPT	ChatGPT	ChatGPT
N4	N/A	N/A	ChatGPT	None	None
N5	N/A	N/A	ChatGPT	None	None
A1	None	None	ChatGPT	ChatGPT	ChatGPT
A2	None	None	ChatGPT	ChatGPT, Microsoft Copilot	ChatGPT, Microsoft Copilot
A3	ChatGPT	ChatGPT	ChatGPT, Microsoft Copilot	ChatGPT	ChatGPT
A4	ChatGPT	None	ChatGPT	Microsoft Copilot	Microsoft Copilot
A5	None	None	ChatGPT, Microsoft Copilot	ChatGPT, Microsoft Copilot	ChatGPT
S1	N/A	N/A	ChatGPT	ChatGPT	ChatGPT
S2	N/A	N/A	ChatGPT, Poe	ChatGPT	ChatGPT
S3	N/A	N/A	ChatGPT	ChatGPT	ChatGPT
S4	N/A	N/A	ChatGPT	None	None
S5	N/A	N/A	ChatGPT	ChatGPT	ChatGPT
S6	N/A	N/A	ChatGPT, Microsoft Copilot	None	None
SA1	None	None	None	ChatGPT	ChatGPT, Microsoft Copilot
SA2	ChatGPT	None	ChatGPT	ChatGPT, Microsoft Copilot	ChatGPT, Microsoft Copilot
SA3	ChatGPT, GitHub Copilot	GitHub Copilot	ChatGPT, GitHub Copilot	ChatGPT	ChatGPT
SA4	ChatGPT	ChatGPT	ChatGPT	ChatGPT	ChatGPT
SA5	ChatGPT, GitHub Copilot	ChatGPT, GitHub Copilot	ChatGPT, GitHub Copilot	ChatGPT	ChatGPT

        - What do you think of these suggestions?
    - Did you use the AI tool(s) to get suggestions about security during the task?
        - Why? / Why not?
- (A and SA Group:) Do you think you would have used AI Tools if you had not been told about it? Please explain your answer.
- No
    - Why not?
    - Would you normally use AI tools if it was not a study?
        - Why? / Why not?
- Do / Would you find yourself attempting to implement security more often when using AI tool(s)?
- Do you trust AI-generated suggestions for functionality?
    - Why? / Why not?
- Do you trust AI-generated suggestions for security?
    - Why? / Why not?

A.5 Surveys

A.5.1 Pre-Survey.

- Expectation asked before solving the task:
    - What is your expectation? Overall, this task is?
    - 1: Very difficult - 7: Very easy
- (A and SA Group:) Have you ever used AI tools for programming?
    - Which AI tools did you use for programming?
    - Do you still use AI tools for programming?
    - Which AI tools do you use for programming?

A.5.2 Post-Survey.

- Experience asked after solving the task:
    - Overall, this task was ...?
    - 1: Very difficult - 7: Very easy
- I have a good understanding of security concepts.
    - 1: Strongly disagree - 7: Strongly agree
- How often do you ask for help facing security problems?
    - 1: Never - 7: Every time
- How often are you asked for help when somebody is facing security problems?
    - 1: Never - 7: Every time
- How often do you need to add security to the software you develop in general (S and SA Group: apart from this study)?
    - 1: Never - 7: Every time
- How often have you stored passwords in the software you have developed (S and SA Group: apart from this study)?
    - 1: Never - 7: Every time
- How would you rate your background/knowledge with regard to secure password storage in a database?
    - 1: Not knowledgeable at all - 7: Very knowledgeable
- Do you think that you stored the end-user passwords securely?
- The Web framework JSF supported me in storing the enduser password securely.
    - 1: Strongly disagree - 7: Strongly agree

A.6 Evaluation of Screen Recording

A.6.1 Terms for Attempted Security. We considered an attempt at implementing security if participants searched for the following terms in a search engine or AI tool:

- Security
- Password Security
- Secure storage

- Encryption
- Hashing
- Password best practices
- Securely store user
- Password guidelines
- Security protocols
- Password safety
- Application security
- Protecting passwords
- Unauthorized access
- Password safety measures
- Password confidentiality
- User credential protection
- Authentication security
- Implementing strong password
- Data protection
- Cybersecurity measures
- Access control
- Password defense
- Cryptography
- Crypto libraries
- Information security
- Secure login
- Java Security

We considered an attempt at implementing security if participants try to implement security in the source code such as:

- Importing security libraries
- Comments in the source code contain security-related terms (see above terms)
- Writing security-related source code such as hashing, salting, encryption

A.6.2 Strategies Employed by Participants.

Actions used by Participants

- Copy/Paste: Parts of AI Code after assessment
- Copy/Paste: Parts of AI Code
- Copy/Paste: Complete AI Code after assessment
- Copy/Paste: Complete AI Code
- Copy/Paste: Blog Post / Tutorial
- Copy/Paste: Stack Overflow
- Copy/Paste: Documentation
- Copy/Paste: Youtube
- Follows Tutorial: Youtube
- Follows Tutorial: Stack Overflow
- Follows Tutorial: Blog Post
- Follows Tutorial: AI Response

Strategies followed by Participants

- General support
- Error debugging
- Learning about unfamiliar topics
- Source code generation
- Providing information to AI

A.7 Codebook

Study Task

- Password Storage
  - Functional password storage
    - functional implementation (1)
    - no functional implementation (0)
    - Challenges
      - API
      - IDE Eclipse
      - Database
- Secure password storage
  - Challenges
    - Outdated algorithm
    - Import of libraries
  - secure implementation (1)
  - no secure implementation (0)
  - How did you implement security?
    - Hypothetical secure implementation (Q)
    - Import of libraries
  - Awareness of necessity of secure solution (Q)
    - No awareness
- Usability of JSF
  - issues / challenges
    - Documentation
- Would you have used AI w/o instructions? (Q)
- Would you normally use AI tools if it was not a study? (Q)
- Would you have tried implementing security w/o instructions? (Q)
  - Deception
- If task was given in company (Q)
  - Tool
  - Ask for help
    - Colleague

Resources

- Website
  - YouTube
  - Stack Overflow
  - GitHub / GitLab
- Experience
  - lack of experience

Security Concepts

- Misconception
- Salting
  - reasoning / purpose
- Hashing
  - Hash functions
  - reasoning / purpose
    - Attacks
    - CPU-intensive
    - Confidentiality
    - Password Confidentiality

AI Tools

- Prompt Engineering
- reason for not using AI
  - during work
- Copy/Paste
- Challenge
- Types of tasks

  - General support
  - Error debugging
  - Source code generation
  - Explanations of topics
    - Documentation of libraries
- Tools
  - Other
    - Other (yes in private)
  - Poe
    - Poe (yes in private)
    - Premium Version
  - ChatGPT
    - Prompt Engineering
    - Premium Version
    - ChatGPT (yes during task)
      - General Support
      - Error debugging
      - Generate documentation for source code
      - Source code generation
      - Explanations of topics
    - ChatGPT (yes in private)
      - General Support
      - Error debugging
      - Source code generation
      - Explanations of topics
  - Microsoft Copilot
    - Prompt Engineering
    - Premium Version
    - Microsoft Copilot (yes during task)
      - General Support
      - Error debugging
      - Source code generation
      - Explanations of topics
    - Microsoft Copilot (yes in private)
  - GitHub Copilot
    - Prompt Engineering
    - Premium Version
    - GitHub Copilot (yes in private)
      - General support
      - Source code generation
      - Error debugging
    - GitHub Copilot (yes during work)
- change to programming workflow due to AI
  - no change
  - frequency of use
  - change to security behavior
- AI suggestions
  - Assessment of AI Response
  - Misconception
  - perception
    - positive
    - neutral
    - negative
  - AI suggestions for functionality
    - warnings
    - perception
      - trust
  - AI suggestions for security
    - perception
      - trust

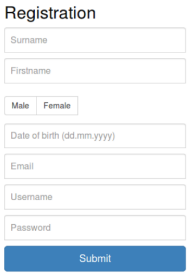
Uncertainty
Private Use of AI (no programming context)
Attempted Security

A.8 Task (Adapted from [59])

Welcome! We would like to invite you to participate in our study and thus make a valuable contribution to our research. Our research goal is to study the usability of the Web Framework ***JavaServer Faces (JSF).***

(Security Instruction: We are specifically interested in the security aspects of password storage with JSF. AI Instruction: We are specifically interested in the use of AI tools.)

Task:
Imagine, the Ruhr University Bochum wants to offer a **social network**, which enables students and members of the University to communicate and interact with each other (similar to Facebook). For this purpose, a platform has to be provided allow users to **register** to the social network. The University has employed a team of developers who have already implemented parts of the platform. For instance, a front-end designer has created this interface:



A database administrator provides a database access using **PostgreSQL** (Object Relational Database Management System (ORDBMS)) and **Hibernate** (Object Relational Mapping (ORM) Framework). Another team member started to implement the program logic, but left the team recently.
Please help the team and complete the partially available implementation. In order to do so, please implement the registration process of the application.
(Security Instruction: We are specifically interested in the security aspects of password storage with JSF. AI Instruction: We are specifically interested in the use of AI tools for solving the task.)

In order to solve the task, you are allowed to use ***any kind of source available on the Internet that may be helpful or has valuable information****. Consequently, you can search and read every website, which you believe is helpful. Further implementation hints can be found on the next page.*

Please fill in the following survey **before** working on the task: [Link to survey]

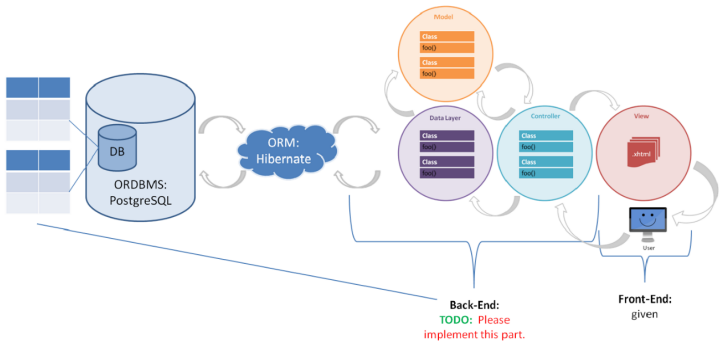


Figure 1: Back-End and Front-End

Figure 4: First Page of the Task PDF

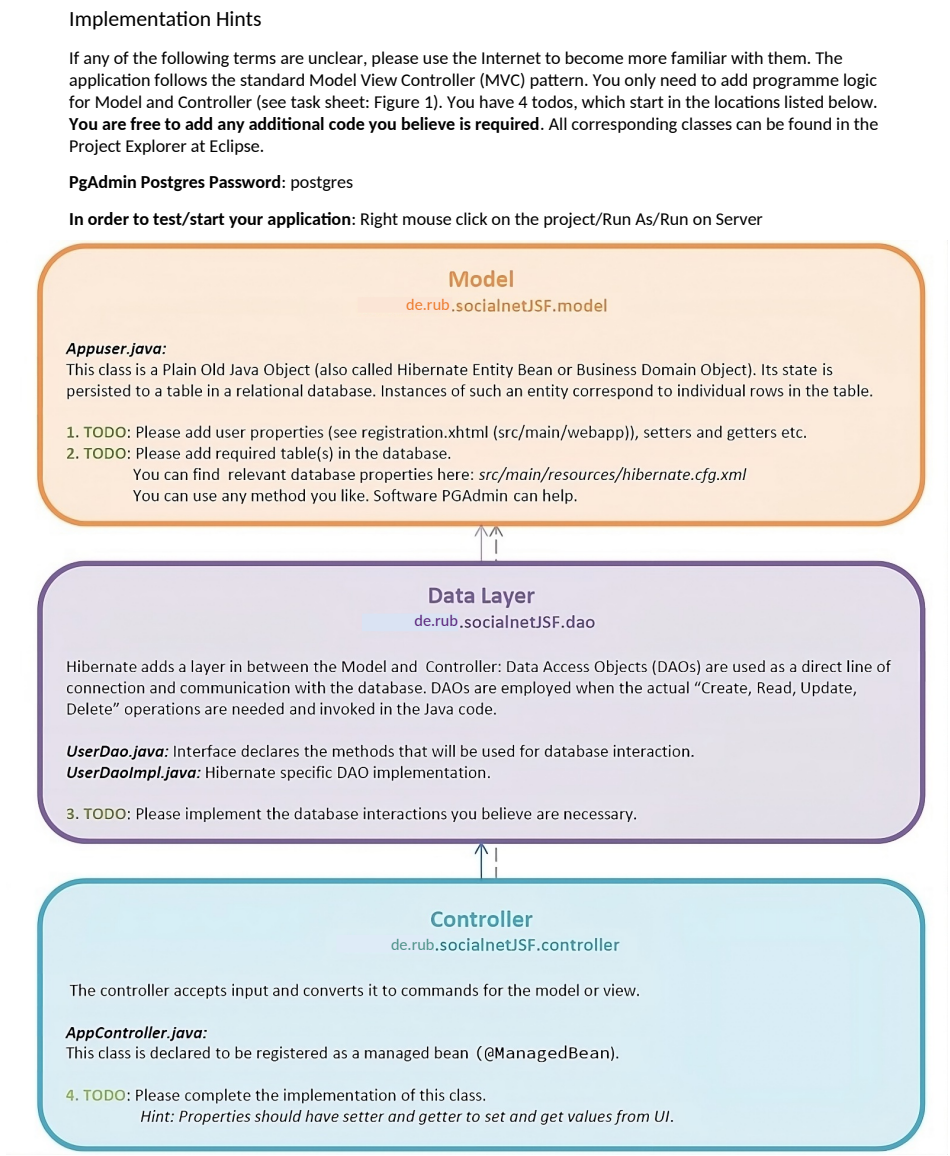


Figure 5: Second Page of the Task PDF

Table 11: Kruskal-Wallis test results with effect sizes, power, and corrected p-values (Bonferroni).

Comparison	p-value	Effect Size (r_b)	Power
A vs N	1.000000	-0.1274	0.9993
A vs S	0.000092	-0.7500	0.9993
A vs SA	0.001511	-0.6342	0.9993
N vs S	0.005542	-0.5263	0.9993
N vs SA	0.048280	-0.4368	0.9993
S vs SA	1.000000	0.1200	0.9993

B Study B

B.1 Supplementary Insights

B.1.1 Kruskal-Wallis Test Results for Instruction Scenarios. Table 11 provides the Kruskal-Wallis Test results of our four instruction scenarios.

B.1.2 Participants' Security and Password Storage Practices. Table 12 provides a summary of participants' security and password storage practices.

B.1.3 Group Security Scores by AI Assistant Usage. Table 13 shows an overview of group security scores of initial and final submissions of participants who used AI assistants during the task and those that did not use AI assistants.

B.1.4 Hashing and Salting Methods of our Participants. Table 14 provides an overview of the hashing and salting methods used by participants in their final submissions.

B.1.5 AI Usage of our Participants. Table 15 provides an overview of the AI assistants used during the study by our participants.

B.2 Upwork Job Post

Participants for Programming Study with Software Developers on Java API Usability

We are researchers from the University of Bochum working in the field of software usability. We are looking for software developers who are interested in taking part in one of our studies (programming and surveys). All data will be processed pseudonymously and stored anonymized after the study; there will be no identifying information published in any form.

We are conducting a study focused on the development of a social networking website to share pictures with family and friends. People need to register to this website in order to be able to share their pictures. Your task would be to program the registration functionality for this website in the backend. The project is in Java and uses Hibernate and JSF, our database uses postgreSQL. The front-end and some parts of the program logic are already developed.

You will receive login credentials for a virtual machine (VM), and you are required to complete the task within this VM.

The payment is divided into 2 milestones. 215 dollars for your code release and further additional 25 dollars for completing our survey about your programming experience and your experience with this task.

You are free to take breaks during the study but must finish participation within 8 hours of starting your participation.

Please complete the study in a focused setting, using a laptop or PC.

1. *What is your level of expertise with Java?*
2. *Which kind of projects did you work on that involved Java?*

B.3 Pre-Survey

The pre-survey is adapted from the study by Naiakshina et al. [57] on freelancers and supplemented it with questions addressing AI.

(1) Your study-ID
[free text]
(2) *Information text and link to the consent form*
(3) I confirm that I have been informed about the research project. I confirm that I have been given an information sheet about the project and that I have had sufficient opportunity to take note of the information. I confirm that I was able to download a copy of the consent form.
I consent/I do not consent
- If 'I do not consent': participant won't take part in the study.
- If 'I consent':
  - How familiar are you with Java?
    1 - Not familiar at all - 7 - Very familiar
  - How familiar are you with PostgreSQL?
    1 - Not familiar at all - 7 - Very familiar
  - How familiar are you with Hibernate?
    1 - Not familiar at all - 7 - Very familiar
  - How familiar are you with Eclipse IDE?
    1 - Not familiar at all - 7 - Very familiar
  - (A/SA only) Have you ever used AI assistants for programming?
    Yes/No
  - (A/SA only) Which AI assistants did you use for programming?
    [free text]
  - (A/SA only) Do you still you use AI assistants for programming?
    Yes/No
  - (A/SA only) Which AI assistants do you use for programming?
    [free text]
  - What is your expectation? Overall, this task is?
    1 - Very difficult - 7 - Very easy

B.4 Post-Survey

The post-survey is structured into three distinct subsections. Initially, it integrates the survey adapted from Naikashina et al.[57]. Subsequently, it incorporates AI-related questions. It concludes with the demographics, adopted from Naiakshina et al.[57]. In extending the survey from the original study [57], we made specific modifications to align it with our revised study design. We omitted questions associated with the deception design, which was not a component of our study. Further, we omitted questions regarding freelancer.com as they were not applicable.

ID1 Your study-ID
[free text]

Table 12: Summary of security and password storage practices.

Question	N	A	S	SA	All
I have a good understanding of security concepts. (Q7)	md = 6	md = 6	md = 6	md = 5	md = 6
7-Point Likert	μ = 5.65	μ = 5.80	μ = 5.60	μ = 5.80	μ = 5.71
Strongly disagree - Strongly agree	σ = 0.933	σ = 1.20	σ = 1.50	σ = 1.01	σ = 1.16
Knowledge of secure password storage (Q12)	md = 6	md = 5.5	md = 5.5	md = 5.5	md = 6
7-Point Likert	μ = 5.30	μ = 5.25	μ = 5.55	μ = 5.10	μ = 5.30
Not knowledgeable - Very knowledgeable	σ = 1.75	σ = 1.45	σ = 1.05	σ = 1.48	σ = 1.44

Table 13: Group security scores of initial and final submissions of participants who used AI assistants during the task and those that did not use AI assistants. Up to 7 points. $_{\text{initial}}$=initial submission, $_{\text{final}}$=final submission.

$\mathbf{N}_{\text{initial\_AI}}$	min = 0, max = 6, σ = 2.45	md = 0, μ = 1.00
$\mathbf{N}_{\text{initial\_No\_AI}}$	min = 0, max = 6, σ = 2.63	md = 0, μ = 1.38
$\mathbf{N}_{\text{final\_AI}}$	min = 4, max = 6, σ = 0.41	md = 6, μ = 5.83
$\mathbf{N}_{\text{final\_No\_AI}}$	min = 0, max = 6, σ = 1.89	md = 6, μ = 4.92
$\mathbf{A}_{\text{initial\_AI}}$	min = 0, max = 6, σ = 1.16	md = 0, μ = 0.43
$\mathbf{A}_{\text{initial\_No\_AI}}$	min = 0, max = 0, σ = 0	md = 0, μ = 0
$\mathbf{A}_{\text{final\_AI}}$	min = 0, max = 6, σ = 2.25	md = 5.5, μ = 4.14
$\mathbf{A}_{\text{final\_No\_AI}}$	min = 0, max = 6, σ = 1.10	md = 2, μ = 1.20
$\mathbf{S}_{\text{initial\_AI}}$	min = 0, max = 6, σ = 2.60	md = 6, μ = 4.18
$\mathbf{S}_{\text{initial\_No\_AI}}$	min = 0, max = 0, σ = 2.54	md = 5 μ = 3.72
$\mathbf{S}_{\text{final\_AI}}$	min = 0, max = 6, σ = 2.24	md = 6, μ = 4.73
$\mathbf{S}_{\text{final\_No\_AI}}$	min = 0, max = 6, σ = 2.20	md = 5.5, μ = 4.39
$\mathbf{SA}_{\text{initial\_AI}}$	min = 1, max = 6, σ = 2.02	md = 6, μ = 4.62
$\mathbf{SA}_{\text{initial\_No\_AI}}$	min = 0, max = 6, σ = 2.27	md = 0, μ = 0.86
$\mathbf{SA}_{\text{final\_AI}}$	min = 1, max = 6, σ = 2.02	md = 6, μ = 4.62
$\mathbf{SA}_{\text{final\_No\_AI}}$	min = 0, max = 6, σ = 2.94	md = 1, μ = 2.57
$\mathbf{All}_{\text{initial\_AI}}$	min = 0, max = 6, σ = 2.75	md = 2, μ = 2.68
$\mathbf{All}_{\text{initial\_No\_AI}}$	min = 0, max = 0, σ = 2.60	md = 0, μ = 1.69
$\mathbf{All}_{\text{final\_AI}}$	min = 0, max = 6, σ = 2.03	md = 6, μ = 4.66
$\mathbf{All}_{\text{final\_No\_AI}}$	min = 0, max = 6, σ = 2.47	md = 5, μ = 3.75

Table 14: Frequency of hashing function and salting usage of participants' final submissions.

Group	None	Base64	Salt	MD5	SHA-256	SHA-512	PBKDF2	BCrypt	Argon2
N	1	1	3	0	3	0	2	13	0
A	1	3	8	1	6	0	1	8	0
S	1	1	15	0	4	1	2	11	0
SA	0	3	11	2	3	0	1	11	0

Q1 Did you create the code on your own or in a team?
I worked on my own./I worked in a team.

Q2 Can you please fill out this questionnaire together with all people who did the coding?
Ok, my team is here./Sorry, I can't get my team right now, but I was involved in the coding./Sorry, I can't get my team right now and I was not involved in the coding process.

Q3 Which IDE did you use to solve the task?
[free text]

Table 15: The number of mentions of AI assistants used during the study and as the most relevant for solving the task.

AI Assistant	During Study					Most Relevant				
	N	A	S	SA	All	N	A	S	SA	All
ChatGPT	6	10	8	10	34	6	10	8	10	34
GPT Model	0	0	1	0	1	0	0	0	0	0
Groq	0	0	0	1	1	0	0	0	1	1
GitHub Copilot	0	0	1	1	2	0	0	1	1	2
Codeium	0	1	0	0	1	0	1	0	0	1
Tabnine	0	2	0	0	2	0	2	0	0	2
Gemini	0	2	1	0	3	0	2	1	0	3
Copilot Microsoft	1	0	0	2	3	1	0	0	1	2
Perplexity	0	0	0	0	0	0	0	1	0	1
All	7	15	11	14	47	7	15	11	13	46

Q4 Overall, this task was ...?
1 - Very difficult - 7 - Very easy

Q5 Do you think your solution is optimal?
Yes/No

Q6 Why do you think your solution is (not) optimal?
[free text]

Q7 I have a good understanding of security concepts.
1 - Strongly disagree - 7 Strongly agree

Q8 How often do you ask for help facing security problems?
1 - Never - 7 - Every time

Q9 How often are you asked for help when somebody is facing security problems?
1 - Never - 7 - Every time

Q10 How often do you need to add security to the software you develop in general (S/SA: apart from this study)?
1 - Never - 7 - Every time

Q11 How often have you stored passwords in the software you have developed (S/SA: apart from this study)?
1 - Never - 7 - Every time

Q12 How would you rate your background/knowledge with regard to secure password storage in a database?
1 - Not knowledgeable at all - 7 - Very knowledgeable

Q13 Do you think that you stored the end-user passwords securely?
Yes/No

- If *Yes*:

Q14 What did you do to store the passwords securely?
[free text]

Q15 Do you think your solution is optimal?
Yes/No
Q16 Why do you think your solution is (not) optimal?
[free text]
Q17 (S/SA only) Do you think you would have stored end-user passwords securely, if you had not been told about it? Please explain your decision.
[free text]
- If *No*:

Q18 Why do you think that you did not store the passwords securely?
[free text]
Q19 (C/A only) Were you aware that the task needed a secure solution?
Yes/No
Q20 What would you do, if you needed to store the end-user passwords securely?
[free text]
Q21 In the previous questions, you mentioned why your solution is not secure and what you would need to do to obtain secure password storage. Why did you not implement your suggestion?
[free text]
Q23 If you had the same job outside of a study, would you have solved it differently in terms of secure password storage?
Yes/No
Q24 Please explain your decision.
[free text]
Q25 Did you use libraries to store the end-user passwords securely?
Yes/No
- If *Yes*:

Q26 Which libraries did you use to store the end-user passwords securely (in this study)?
[free text]
Q27 Please name the most relevant library you have used to store the end-user passwords securely (in this study).
[free text]
Q28 You have identified {participant's answer} as the most relevant library to store end-user passwords securely. How would you rate its ease of use in terms of accomplishing your tasks functionally / securely?
1 - Very difficult - 7 - Very easy
Q29 Please explain your decision.
[free text]
Q30 Usability scale for {participant's answer}. See Appendix B.5.
Q31 The Web framework JSF supported me in storing the end-user password securely.
1 - Strongly disagree - 7 - Strongly agree
Q32 Please explain your decision.
[free text]
Q33 The Web framework JSF prevented me in storing the end-user password securely.
1 - Strongly disagree - 7 - Strongly agree
Q34 Please explain your decision.
[free text]
Q35 Usability scale for JSF; the term *library* was replaced by *framework*. See Appendix B.5.
Q36 Have you used Java APIs / libraries to store end-user passwords securely before?
Yes/No
- If *Yes*:

Q37 Which Java APIs / libraries to store end-user passwords securely have you used before?
[free text]
Q38 What is your most-used API / library for secure password storage?
[free text]
Q39 How would you rate its ease of use in terms of accomplishing your tasks functionally?
1 - Very difficult - 7 - Very easy
Q40 Please explain your decision.
[free text]
Q41 How would you rate its ease of use in terms of accomplishing your tasks securely?
1 - Very difficult - 7 - Very easy
Q42 Please explain your decision.
[free text]

B.4.1 AI-Related Extension. We extended the original post-survey[57] by 38 questions on AI assistants.

AI1 Have you ever used AI assistants during development apart from this study?
Yes/No
- If *Yes*:

AI2 Which AI assistants have you used during development apart from this study?
[free text]
AI3 Are you aware of any premium features of the AI assistants you named? Please name the premium features.
[free text]
AI4 How do you use AI assistants during development?
[free text]
AI5 How often do you use AI assistants during development?
1 - Never - 7 Every time
AI6 What is your most-used AI assistant during development?
[free text]
AI7 Please explain why this is your most-used AI assistant during development.
[free text]
AI8 How did AI assistants change your development workflow?
[free text]
AI9 Did you use any AI assistants to complete the task?
Yes/No
- If *No*:

AI10 Would you normally use AI assistants if it was not a study?
Yes/No
AI11 Please explain your decision.
[free text]

- If *Yes*:

AI12 (A/SA only) Do you think you would have used AI assistants if you had not been told about it?
Yes/No

AI13 Please explain your decision.
[free text]

AI14 Which AI assistants did you use to complete the task?
[free text]

AI15 Please name the most relevant AI assistant you have used to complete the task.
[free text]

AI16 How would you rate {participant's answer}'s ease of use in terms of completing the task functionally?
1 - Very difficult - 7 Very easy

AI17 Please explain your decision.
[free text]

AI18 How would you rate (participant's answer)'s ease of use in terms of completing the task securely?
1 - Very difficult - 7 Very easy

AI19 Please explain your decision.
[free text]

AI20 (participant's answer) supported me in storing the end-user password securely.
1 - Strongly disagree - 7 - Strongly agree

AI21 Please explain your decision.
[free text]

AI22 (participant's answer) prevented me in storing the end-user password securely.
1 - Strongly disagree - 7 - Strongly agree

AI23 Please explain your decision.
[free text]

AI24 Did you use (participant's answer) to get suggestions about functionality during the task?
Yes/No

AI25 Why did you (not) use (participant's answer) to get suggestions about functionality during the task?
[free text]

AI26 Did you receive any suggestions about security from (participant's answer) during the task?
Yes/No

AI27 Did you use (participant's answer) to get suggestions about security during the task?
Yes/No

AI28 Why did you (not) use (participant's answer) to get suggestions about security during the task?
[free text]

AI29 I trust AI-generated suggestions for functionality.
1 - Strongly disagree - 7 - Strongly agree

AI30 Please explain your decision.
[free text]

AI31 I trust AI-generated suggestions for security.
1 - Strongly disagree - 7 - Strongly agree

AI32 Please explain your decision.
[free text]

AI33 What do you think of AI-generated suggestions for functionality? Please explain.
[free text]

AI34 What do you think of AI-generated suggestions for security? Please explain.
[free text]

- If participant has used AI assistants in/apart from the study:

AI35 Do you find yourself attempting to implement security more often when using AI assistants?
Yes/No

AI36 Please explain your decision.
[free text]

- If participant has not ever used AI assistants:

AI37 Would you find yourself attempting to implement security more often when using AI assistants?
Yes/No

AI38 Please explain your decision.
[free text]

B.4.2 Demographics. We adopted the demographic questions from [57], updating D1 for better gender inclusivity as guided by [80].

D1 What is your gender?[80]
woman / man / non-binary / prefer not to disclose / prefer to self-describe: [free text]

D2 Age
[Number]

D3 What is your current main occupation?
Freelancer developer / Industrial developer / Industrial researcher / Academic researcher / Undergraduate student / Graduate student / Other: [free text]

D4 What type(s) of software do you develop?
Web applications / Mobile applications / Desktop applications / Embedded applications / Enterprise applications / Other (please specify): [free text]

D5 How many years of experience do you have with Java development?
[Number]

D6 How many years of experience do you have with software development in general?
[Number]

D7 What country do you live in?
[free text]

D8 What is your nationality?
[free text]

D9 Do you have a university degree?
Yes/No

- If *Yes*:

D10 At which university/universities were/are you enrolled?
[free text]

D11 What was/is your subject?
[free text]

D12 Were/Are you taught about IT-security at university?
Yes/No/I don't recall

D13 Were/Are you taught about IT-security extramural?
Yes/No

* If *Yes*:

D14 Where were/are you taught about IT-security extramural?
[free text]

D15 What was your main source of learning about IT-security?
[free text]

- If *No*:

D16 How did you gain your IT skills?
[free text]

D17 How did you gain your IT-security skills?
[free text]

D18 How do you rate the payment of the implementation task?
Way too little/Too little/Just right/Too much/Way too much

D19 How do you rate the payment of the survey?
Way too little/Too little/Just right/Too much/Way too much

D20 If you wish to be contacted by our research group to be informed about the study results, you can provide us your email address. The email address will be stored separately from the study data.
[free text]

D21 If you wish to be contacted by our research group for further studies in the future, you can provide us your email address. The email address will be stored separately from the study data.
[free text]

D22 Thank you for answering the questions! If you have any comments or suggestions, please leave them here:
[free text]

B.5 Usability Scale from [1]

Following the methodology of the original study [57], we used the modified [57, 58] usability scale from [1], omitting the "does not apply" option for questions Q10 and Q11. Thereby, we used the following scale:

Please rate your agreement to the following questions on a scale from 'strongly agree' to 'strongly disagree.' (Strongly agree; agree; neutral; disagree; strongly disagree) Calculate the 0-100 score as follows: $2.5 * (5 - Q_1 + \sum_{i=2..10}((Q_i - 1))$; for the score, Q11 is omitted.

(1) I had to understand how most of the assigned library works in order to complete the tasks.
(2) It would be easy and require only small changes to change parameters or configuration later without breaking my code.
(3) After doing these tasks, I think I have a good understanding of the assigned library overall.
(4) I only had to read a little of the documentation for the assigned library to understand the concepts that I needed for these task.
(5) The names of classes and methods in the assigned library corresponded well to the functions they provided.
(6) It was straightforward and easy to implement the given tasks using the assigned library.
(7) When I accessed the assigned library documentation, it was easy to find useful help.
(8) In the documentation, I found helpful explanations.
(9) In the documentation, I found helpful code examples.

Please rate your agreement to the following questions on a scale from 'strongly agree' to 'strongly disagree'. (Strongly agree; agree; neutral; disagree; strongly disagree).

(10) When I made a mistake, I got a meaningful error message/exception.
(11) Using the information from the error message/exception, it was easy to fix my mistake.

B.6 Task Description

Hello,
Thank you for helping us finish our project!
We are developing a social networking website to share pictures with family and friends. People need to register (email, name, gender, birth date, username and password) to this website in order to share their pictures. We need help with finishing the registration logic for our social networking website.
The front-end is already developed. Another developer started to implement the program logic (MVC), but left the team recently. We need you to complete the partially available implementation by programming the registration functionality in the back-end.
(Please use any AI assistants of your choice.)
(Please ensure that the user password is stored securely.)

According to our former developer, the following things have to be done:

Setting up the environment:
The following software will be needed:
1. Tomcat server
2. Optional: Eclipse IDE for Java EE Developers
3. Optional: pgAdmin

Prepare the project & the database:
If you use Eclipse, it could look the following way:

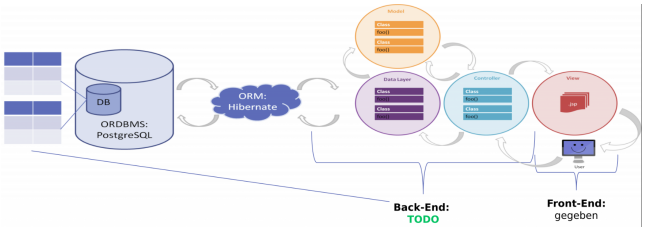
1. Import the existing project as Existing Maven Project.

2. Add Maven Dependencies to the project in Eclipse: Right mouse click on the project/Properties/Deployment Assembly/Add/Java Build Path Entries → choose Maven Dependencies → Finish→ Apply

3. Create a local database using PostgreSQL and connect to it. (For example in pgAdmin: File/Add Server…)
You will find example information (host, port, DB Name, username and password) in the hibernate.cfg.xml file (src/main/resources).

Implementing the registration process:
This was our developer's TODO list. It might not be complete, but it will help you understand what has already been developed and what is left to be done. Figure 1 shows how the different parts are communicating with each other.

In order to test/start your application: Right mouse click on the project/Run As/Run on Server



Util:
1. Done: Hibernate configuration
2. Done: Implement method: getSessionFactory()

View:
1. Done: Create registration.xhtml
2. Done: Create successRegistration.xhtml

Database:
1. TODO: Add table appuser in the database

Model – Appuser.java
1. TODO: Add user properties (see registration.xhtml)
2. TODO: Add getters & setters

DAO – UserDAO.java
1. Done: Implement method: validateUsername()
2. Done: Open session
3. TODO: Implement method: save()

Controller – AppController.java
1. TODO: Add variables for properties
2. TODO: implement method: saveUser()

(Please use any AI assistants of your choice.)
(Please ensure that the user password is stored securely.)

B.7 Codebook

B.7.1 Security.

Security Perception.

- Reasons for not implementing security
  - Error debugging
  - Security is not (only) developers' responsibility
  - Time
- Belief why password not stored securely
- Misconception
  - Encryption and hashing terminology
  - Encryption is secure
  - Insecure algorithm is secure
  - Encoding is secure
  - Insecure salting
  - Security is an afterthought in the development process
  - Improper use of hashing algorithm

Security Behavior.

- Reason for (not) attempting security more often with AI
  - No change to security behavior
    - Already enough experience
    - Relies on other resources instead of AI
    - Negative comment about AI
  - AI warns about security
  - AI is...
    - ...efficient
    - ...effective
    - ...easy to use
  - Security is always a requirement
  - Lack of security knowledge
  - Depends on requirements
  - If AI suggests security, it gets implemented (convenience)
  - Prefers to be prompted for security
- Does not rely on AI for security
- Consults person
- Implemented security suggestions by AI
- Relies on AI for security
- Requirements
  - Best/standard/common practice
  - Task didn't ask for security
    - Security was minimal part of task
  - Responsibility
- Ask AI to ignore security (to achieve functionality)
- Prefers to be prompted for security

B.7.2 AI.

Outside of study.

- Which AI tool used outside of study
  - Use (AI2)
    - ChatGPT
    - GitHub Copilot
    - Tabnine
    - Genie
    - Copilot (Microsoft)
    - Codeium
    - Gemini
    - Google Bard
    - OpenAI Codex
    - Groq
    - Perplexity
    - Google Assistant
  - Most-used (AI5)
    - ChatGPT
    - OpenAI Codex
    - GitHub Copilot
    - GPT model
    - Copilot (Microsoft)
    - Tabnine

    - Amazon Alexa
    - Jetbrains AI
    - Gemini
- Type of usage during development outside of study
  - Code review
  - Provide context
  - Code generation
  - General support
    - Search engine
  - Error debugging
  - Code completion

During study.

- Which AI tool used during study
  - Mentioned (AI14)
    - ChatGPT
    - GPT Model
    - Groq
    - GitHub Copilot
    - Codeium
    - Tabnine
    - Gemini
    - Copilot (Microsoft)
  - Most relevant (AI15)
    - ChatGPT
    - Perplexity
    - Groq
    - GitHub Copilot
    - Tabnine
    - Gemini
    - Codeium
    - Copilot (Microsoft)
- Type of usage during development outside of study
  - Provide context
  - General support
    - Search engine
  - Error debugging
  - Code completion
  - Code generation

AI Perception.

- AI is capable
  - Unspecified
  - Security
  - Functionality
- AI is incapable
  - Unspecified
  - Security
  - Functionality
- AI-generated suggestions need to be assessed
  - Unspecified
  - Security
  - Functionality
- AI needs to be handled responsibly
  - Unspecified
  - Data privacy violation (do not share sensitive information)
  - Security
  - Functionality
- (Dis)trust in AI
  - Distrust in AI
    - Distrust in general
    - Distrust in functionality
    - Distrust in security
  - Trust is dependent on
    - Case
      - Security
      - Functionality
    - Past experience
      - Functionality
    - Knowledge
      - Security
      - Functionality
  - Relies on themselves instead of AI
  - Trust since AI is up to date
    - Unspecified
    - Security
    - Functionality
  - Trust in platform
  - Trust in security due to lack of experience
  - Trust only if assessable
- Positive comment
- AI being compared to human

Perception of AI Usability.

- Unspecified
  - Negative perception of AI Usability
    - Need to edit AI-generated code
  - Positive perception of AI Usability
    - Efficient
    - Helpful
    - Easy to use
    - Effective
    - Well integrated
      - GitHub Copilot
      - Tabnine
    - Availability
    - Trained on a lot of data
  - Relies on themselves instead of AI
  - Depends on...
    - Provided context
    - Prompts
    - AI
  - Cost
  - Differences in AI assistants' quality
- Security
  - Negative perception of AI Usability
    - Not helpful
    - Needs to edit AI-generated code
    - Insecure AI suggestion
      - Trainings data might be malicious
  - Positive perception of AI Usability
    - Efficient
    - Helpful
    - Easy to use

    - Effective
    - Trained on a lot of data
  - Relies on themselves instead of AI
  - Depends on...
    - Case
    - Prompts
    - AI
    - Own knowledge
  - Lack of security knowledge makes it difficult to assess
  - No added value
- Functionality
  - Negative perception of AI Usability
    - Not helpful
    - Need to edit AI-generated code
    - Difficult to use
  - Positive perception of AI Usability
    - Efficient
    - Helpful
    - Easy to use
    - Effective
    - Well integrated
      - Unspecific
      - Tabnine
  - Relies on themselves instead of AI
  - Depends on...
    - Case
    - Prompts
    - AI
    - Own knowledge
    - AI response

Impact on Workflow.

- Efficiency
- Effectiveness
- Use AI instead of other resources
  - Resources
    - Documentation
    - StackOverflow
    - Search engine
- Habit
- Complicated workflow
- More enjoyable