CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Evaluating the Impact of a Multi-Agent AI Assistant on Human Creativity in Exploratory Testing

Master's Thesis in Computer science and engineering

Jiachun Cai

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's Thesis 2026

Evaluating the Impact of a Multi-Agent AI Assistant on Human Creativity in Exploratory Testing

Jiachun Cai

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Evaluating the Impact of a Multi-Agent AI Assistant on Human Creativity in Exploratory Testing
Jiachun Cai

Supervisor: Francisco Gomes, Department of Computer Science and Engineering
Examiner: Miroslaw Staron, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Jiachun Cai
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Exploratory Testing (ET) is an important approach for uncovering complex software defects, however its success is frequently impacted by human factors such as cognitive bias, mental fatigue, and rigid thinking patterns. This research investigates the impact of AI assistance on the creative performance of human testers. Instead of treating AI as a passive assistant, this study explores how creative stimuli can help testers break through cognitive bottlenecks and explore non-obvious failure scenarios.

Using the WICKED multi-agent framework as an experimental platform, we conducted a human-centered user study involving participants with software engineering backgrounds. The research analyzes how real-time AI prompts influence the generation of test scenarios across multiple Software Requirement Specifications (SRS). We assess the interventions along two questions: their effect on the creativity of test artifacts (Quantity, Quality, Novelty), and their usability (Perceived Usefulness and Perceived Ease of Use).

We find that WICKED did not reliably increase creativity. Particularly, (i) it did not raise the number of tests created, (ii) its effect on novelty was inconclusive, and (iii) quality was improved only on the more complex system, where it surfaced non-obvious faults such as a concurrency bug missed by manual testers. In terms of usability, participants found WICKED easy to learn and useful in principle, but its slow and unclear interaction limited its perceived usefulness. These results suggest that creativity chatbots are best targeted at complex, uncertain testing tasks and must protect early user trust through speed and relevance, indicating that successful adoption depends as much on the testers themselves as on the strength of the model.

Keywords: Exploratory Testing, Human-AI Collaboration, Creativity Techniques, Large Language Models, User Study.

Acknowledgements

Thank you so much for the supervision and help from my supervisor, Francisco. Doing a thesis alone is a hard job with a very heavy workload. I appreciate my supervisor's assistance, which makes it much easier. Also, I thank all the participants who joined and helped with my research.

Jiachun Cai, Gothenburg, June 2026

Contents

List of Figures

List of Tables

1

Introduction

Software testing is a critical activity for ensuring the reliability of modern software systems. Among various testing methodologies, Exploratory Testing (ET) stands out due to its inherent flexibility. In ET, test design, execution, and learning occur simultaneously [4]. Unlike scripted testing, which follows predefined steps, ET relies heavily on the tester's intuition, curiosity, and domain expertise [29].

However, this high degree of freedom also presents a significant challenge. The effectiveness of ET is often inconsistent because it depends on the individual tester's experience and mental state [27]. Maintaining creativity throughout a testing session is cognitively demanding. Testers often struggle to think "outside the box," leading to missed complex or novel failure scenarios [19]. Furthermore, while creativity techniques like brainstorming and Synectics are widely used in group settings (e.g., Requirement Engineering) [26], ET is typically a solitary activity. This makes it difficult for individual testers to engage in collaborative ideation.

Contrastingly, current test automation frameworks mostly focus on execution efficiency and code coverage, offering **minimal support for the human ideation process** [13]. While Large Language Models (LLMs) have emerged as potential tools for cognitive augmentation [38], their application in bridging the "collaboration gap" in solitary ET remains under-explored. This suggests that LLM-based Conversational AI could act as a virtual brainstorming partner, effectively bringing group-level creativity heuristics to the individual tester's workflow.

Gong has proposed a chatbot assistant named WICKED to address cognitive bottlenecks in ET [15]. WICKED employs a dual-agent architecture that alternates between different modes of engagement — challenging user assumptions and facilitating associative thinking — to stimulate the tester's creativity and improve testing efficacy. Empirical results demonstrate that WICKED achieves a significantly broader and more consistent coverage of creativity-support communication patterns compared to baseline LLM models; while standard assistants often rely on limited, single-pattern responses, WICKED consistently delivers a rich combination of multi-dimensional prompts that support the depth and complexity of the testing dialogue.

1.1 Problem Description and Proposed Solution

The core challenge in ET is the cognitive workload inherent in its design. While the flexibility of ET offers significant freedom, it is often constrained by mental fatigue and fixed thinking patterns. Testers frequently struggle to maintain consistent creative output, leading to a tendency to follow familiar "happy paths" and miss complex, novel failure scenarios. Consider a standard one-time password (OTP) expiry mechanism as an example. A fatigued tester typically writes the expected, literal checks: entering a valid OTP within the time window leads to successful authentication, while entering it after the window leads to a denial. These are not novel — they merely restate the requirement. The defects that matter hide in non-obvious scenarios that fatigue makes testers skip: for instance, setting the device clock back and re-entering an expired token (testing whether expiry is checked against a trusted server time or a spoofable local clock), or generating a token right before midnight so its validity window crosses the date boundary (testing whether the calculation holds across a new day). The gap between the first set and the second is exactly the creative output that erodes under cognitive fatigue, and the problem this research targets.

Consequently, there is a critical need to investigate how AI assistance can mitigate these cognitive bottlenecks and augment human creativity during the testing process. To explore the impact of AI on tester creativity, this study utilizes WICKED as a research platform. WICKED is a dual-agent chatbot specifically designed to support ET through creativity heuristics. While previous work by Gong demonstrated the technical feasibility of WICKED's multi-agent architecture, its actual influence on human cognitive processes remains unexplored. We aims to collect empirical data on how interactive AI interventions influence the quantity, quality, and novelty of human-generated test cases. Since WICKED has not yet been done with formal evaluation with end-users, it provides a unique opportunity to investigate the practical advantages and limitations of AI assistance within the context of ET. Furthermore, we do not know if testers find the AI helpful or if the constant prompting adds unnecessary cognitive load during complex tasks (subjective perception). Therefore, this study must move beyond system verification to evaluate the tool in a realistic, human-centered environment.

To evaluate the impact of WICKED on human creativity, we conducted a user study involving recruited participants with software engineering backgrounds. Testers interact with the SUT in real-time, using WICKED to stimulate new ideas which they then document as executable test scenarios under different conditions (with vs. without WICKED support).

1.2 Purpose of the Study

The primary purpose of this study is to investigate the impact and usability of AI assistance within the context of ET. By employing a multi-agent AI collaborator, this research aims to determine how such tools can effectively augment human creativity

when testers interact with Software Requirement Specifications (SRS) and Systems Under Test (SUT). Rather than focusing solely on technical verification, this study prioritizes a human-centered evaluation to understand how AI-driven prompts influence testing performance and the overall user experience during dynamic testing sessions. Specifically, this study seeks to address the following objectives:

- **Breaking Mental Rigidness:** To evaluate if the specialized functions of the AI assistant can help individual testers break free from routine thinking patterns and move beyond standard "happy paths."
- **Measuring Ideation Quality:** To quantify the impact of AI assistance on the **Quantity, Quality, and Novelty** of generated test artifacts, particularly focusing on the detection of hidden logic failures that are often missed due to mental fatigue.
- **Assessing Human-AI Collaboration:** To investigate the **practical utility** and user acceptance of WICKED, exploring whether testers perceive the agentic dialogue as a valuable cognitive support or an additional workload.

We employ two categories of metrics to assess the outcomes: Product Metrics and User Acceptance Metrics. Based on Shah's framework [33], we evaluate the created test artifacts across three dimensions: Quantity, Quality, and Novelty. To measure the impact on the testers themselves, we evaluate Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) to understand how the AI assistant influenced their creative performance and cognitive effort.

Ultimately, this research provides empirical evidence on how generative AI can be integrated into the ET loop not as a replacement for human intuition, but as a strategic partner that enhances the thoroughness and creativity of the manual testing process.

1.3 Significance of the Study

The importance of this study stems from the inherent cognitive limitations of human testers in ET. While ET is essential for uncovering complex defects, its success is often hindered by human factors such as cognitive bias, mental fatigue, and fixed thinking patterns. As AI becomes increasingly integrated into software development, it is vital to understand how it can be used not just to automate tasks, but to augment human creativity.

The expected scientific contributions of this work center on expanding the understanding of AI assistance in software engineering. By conducting a user study, we provide empirical evidence on how real-time, heuristic-guided AI interventions influence the overall effectiveness and creative breadth of human-led testing. Furthermore, the study explores the cognitive trade-offs involved in AI-human collaboration, specifically analyzing whether AI assistance improves testing performance or inadvertently increases a tester's cognitive load. These findings offer a founda-

tional understanding of how interactive AI can be designed to support, rather than disrupt, human creative processes.

On a technical level, this study provides practical insights into the extensibility and application of AI-driven assistants in software testing. By configuring the WICKED framework to various systems under test (SUT), we demonstrate how its multi-agent architecture can be expanded and customized to accommodate different testing requirements and constraints. Additionally, the experimental setup and evaluation process developed for this research offer a reusable technical framework for evaluating the effectiveness of similar interactive tools. These contributions serve as a practical guide for developers seeking to implement and refine collaborative AI systems in real-world software engineering environments.

2

Background and Related Work

This chapter presents the theoretical background of the research. It covers the principles of Exploratory Testing (ET), the role of creativity, and the evaluation metrics used to measure testing performance. Finally, it reviews the use of LLMs and testing agents in software engineering before introducing the WICKED assistant.

2.1 Exploratory Testing

ET is defined as an approach to software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of their work. Unlike traditional scripted testing, which follows a rigid "plan-then-execute" model, ET treats test design, test execution, and learning as mutually supportive activities that occur simultaneously during the testing process [22]. The core philosophy of ET is that the tester uses information gained while testing to design new and better tests in real-time. By leveraging human intuition and experience, ET excels at identifying complex bugs that are often missed by pre-defined, static test cases [4].

A common critique of early ET was its perceived lack of structure, often being conflated with "Ad-hoc" testing. To formalize the process, Bach and Bach [21] introduced Session-Based Test Management (SBTM). This framework provides a method for making ET measurable and auditable without sacrificing its flexibility. Under SBTM, testing is conducted in uninterrupted "sessions" guided by a "charter" — a high-level statement of goals. This structured approach allows teams to report on test coverage and progress, bridging the gap between the creative freedom of ET and the management requirements of industrial software engineering.

2.1.1 Heuristic-Driven Frameworks in Exploratory Testing

While SBTM provides the management framework, the actual execution of ET relies on heuristics-experience-based rules that transform "curiosity" into a systematic and disciplined approach to uncovering complex defects.

The most prominent heuristic framework is Whittaker's "Testing Tours" [39], which metaphorically treats an application as a city where testers follow specific routes to

ensure comprehensive coverage. For example, the "Supermodel Tour" guides a tester to focus strictly on UI aesthetics and surface-level consistency, while the "Saboteur Tour" challenges the system's resilience by intentionally providing invalid inputs or interrupting processes to test error handling [39]. Complementing these tours, James Bach introduced the SFDPOT model (Structure, Function, Data, Platform, Operations, and Time) [4], a heuristic checklist that prompts testers to evaluate the software from multiple dimensions, ensuring that environmental factors and temporal constraints are not overlooked. By utilizing these structured heuristics, ET moves beyond random interaction to provide high-value, professional feedback early in the development cycle.

2.2 The Concept of Creativity

In a general sense, creativity is defined as the ability to produce work that is both original (novel or unexpected) and effective (useful or meeting task constraints). According to the foundational work by Sternberg and Lubart[36], creativity is not a singular trait but a confluence of distinct resources, including intellectual abilities, knowledge, thinking styles, and personality. In professional contexts, creativity is rarely about "artistic" expression; rather, it is a problem-solving process used to generate valuable solutions in the face of uncertainty or complex constraints.

2.2.1 Human Creativity and Cognitive Processes

Human creativity is often explained through the componential theory of creativity proposed by Amabile [2]. This theory suggests that individual creativity requires three essential components: domain-relevant skills (knowledge and technical expertise), creativity-relevant processes (cognitive styles that allow for "thinking outside the box"), and task motivation. Within the human brain, creativity involves a balance between divergent thinking — the ability to generate multiple diverse ideas and convergent thinking — the ability to evaluate those ideas and select the most viable one. This cognitive flexibility allows humans to recognize patterns and associations that are not immediately obvious [7].

2.2.2 4P Model of Creativity

Proposed by Mel Rhodes in 1961, the 4P model provides a multidimensional framework for understanding the creative ecosystem by categorizing it into four intertwined dimensions: Person, Process, Product, and Press [31].

Person Examines the individual's dispositional traits, including personality, cognitive style, motivation, and self-concept.

Process Refers to the mental mechanisms of ideation, traditionally categorized into four stages: preparation, incubation, illumination, and verification.

Product Focuses on the tangible outcome of the creative effort. In software testing,

this refers to the specific test cases and testing scenarios generated.

Press Characterizes the relationship between the individual and their external context. This includes the situational pressures, organizational climate, and available resources.

Building on this framework, recent software engineering research has utilized the 4P model to analyze how technological shifts influence creative outcomes. For instance, Jackson et al. [20] utilized this framework to establish a research agenda for generative AI in software development. Their analysis suggests that while AI tools primarily transform the creative **Process** through rapid ideation and automation, they also redefine the **Product** by shifting the developer's focus from low-level implementation to high-level architectural evaluation. Their findings highlight that the introduction of AI as a new environmental **Press** necessitates a re-evaluation of the **Person's** skill sets, as traditional coding expertise may be augmented or replaced by prompt engineering and critical auditing capabilities.

2.2.3 Creativity Techniques in Software Engineering

Creativity techniques are specialized tools used to trigger creative potential and break cognitive barriers. Literature identifies over 120 creativity techniques, categorized into associative (recombining existing ideas) and provocative (breaking mental models) [32].

Brainstorming A creative conference for producing a list of ideas — ideas which can be subsequently evaluated and further processed [30].

Synectics Developed by Gordon, this technique uses four types of analogies (direct, personal, symbolic, and fantasy) to "make the strange familiar and the familiar strange" [16].

Walt Disney Method Decomposes the creative process into three roles: the Dreamer (uninhibited ideation), the Realist (feasibility), and the Critic (risk identification).

Empirical evidence highlights both the potential and the limitations of these techniques. Regarding brainstorming, while it was designed to maximize output through group synergy, later studies identified "production blocking" as a primary cognitive barrier. Diehl and Stroebe (1987) demonstrated that individuals working in isolation often generate more unique ideas than those in interacting groups [12], as the need to wait for one's turn to speak can stifle immediate creative impulses. However, more recent meta-analyses suggest that electronic brainstorming can effectively mitigate these losses by allowing for anonymous and simultaneous input, thereby enhancing both idea quantity and quality [11].

The Walt Disney Method leverages the benefits of cognitive perspective-taking and role-segregation. Empirical research on creative standard-setting shows that separating the generative phase from the evaluative phase prevents "premature closure,"

where ideas are discarded before they are fully developed.

2.2.4 Creativity in Software Testing

In software testing, creativity is primarily expressed through exploratory testing. Kaner defines ET as the "simultaneous learning, test design, and test execution", a definition that highlights its departure from rigid, scripted methods. Unlike standard testing, ET transforms the process into a dynamic activity where testers think freely to discover unexpected system behaviors. By bridging these activities, ET provides a fertile ground for testers to exercise their creative agency in uncovering defects that traditional scripts often miss [19].

This creative process is driven by questioning strategies, where testers act as investigators by treating each test case as a specific question asked of the product [39]. To generate a comprehensive list of scenarios, effective testers employ creative thinking frameworks such as the Phoenix Checklist — a set of context — free questions originally developed to analyze challenges from multiple angle — or standard "newspaper questions" (who, what, where, when, how, and why). These methods force a multi-angled analysis of the software, ensuring that the exploration remains systematic rather than random.

Finally, the output of this creative exploration often sparks an ongoing debate in software engineering: does true creativity lie in finding a bug or in fixing it? While some argue that uncovering a hidden, complex defect is the peak of creative work, others believe the real ingenuity resides in the solution and the fix [5]. Regardless of the perspective, both activities require a commitment to "digging deeper" to understand the underlying logic behind the code, proving that creativity is essential throughout the entire defect lifecycle.

2.3 Metrics for Testing Performance and Creativity

To assess the effectiveness of testing methods that involve creativity and human-computer interaction, specific evaluation frameworks are required. This section introduces established metrics for measuring both ideation quality and user acceptance of new testing technologies.

2.3.1 Metrics for Ideation Effectiveness

The framework proposed by Shah et al. [33] is the gold standard for measuring ideation effectiveness in engineering:

- **Quantity:** The total number of discrete, non-redundant ideas generated.
- **Quality:** The feasibility of the idea and its adherence to design specifications.

- **Novelty:** How unusual or unexpected an idea is compared to a defined baseline.
- **Variety:** The breadth of the solution space explored, often measured using a functional genealogy tree.

These metrics have been widely adopted in engineering and design research to benchmark creative performance. Shah et al. [34] later formalized the framework into four objective measures of ideation effectiveness, providing the theoretical basis and application procedures for each metric and illustrating them with case studies. The framework has since been applied empirically: for example, Genco et al. [14] used the novelty and quality metrics to compare design concepts produced by freshman and senior engineering students, finding that seniors generated less novel concepts than freshmen. Other studies suggest that the metrics do not always move together: a high **Quantity** of ideas increases the chance of capturing high-quality ones [24], while exposure to examples tends to narrow the explored space and pull participants toward familiar designs, even as it can raise the novelty of individual ideas [35].

More recently, researchers have applied these metrics to evaluate human-AI collaboration. Studies exploring AI-assisted design found that Large Language Models (LLMs) can significantly boost the **Novelty** and **Quantity** of generated ideas by providing unexpected perspectives. However, these studies also noted that the **Quality** of AI-generated suggestions varies, requiring human expertise to filter out infeasible or irrelevant results.

In this study, we focus on **Quantity, Quality, and Novelty** to evaluate the effectiveness of the WICKED assistant. We exclude Variety to maintain a specific focus on the depth and originality of individual test scenarios. By measuring these three metrics, we aim to determine if AI interventions can help testers produce a higher volume of effective and innovative test cases compared to traditional manual testing.

2.3.2 Technology Acceptance Model (TAM)

The Technology Acceptance Model (TAM) is a widely used theory that explains why individuals choose to adopt or reject a new technology. Developed by Fred Davis [9], the model suggests that a person's intention to use a system is primarily driven by two cognitive factors: Perceived Usefulness (PU) and Perceived Ease of Use (PEOU). PU is defined as the degree to which a user believes that using a specific technology will improve their job performance, while PEOU refers to how much a user believes that using the system will be free of physical or mental effort.

A key insight of TAM is that PEOU directly influences PU; if a tool is easy to navigate, users are more likely to perceive it as a useful asset for their work. Over time, the model has been expanded into TAM2 and TAM3 to include external factors such as social influence and individual experience [37]. Despite these updates, the original TAM remains a cornerstone in human-computer interaction (HCI) research

because it provides a simple yet powerful framework for predicting user behavior and guiding the design of more effective software tools.

2.4 LLMs and Testing Agents in Software Testing

The emergence of Large Language Models (LLMs) represents one of the most significant technological disruptions in the field of software testing. These mathematical systems, trained on vast corpora of text and code, exhibit emergent abilities to reason through multi-step problems and connect disparate concepts, making them uniquely suited for complex engineering tasks. Current research identifies several domains where LLMs effectively augment testing activities:

- **Test Case and Data Generation**: AI-driven data synthesis and code generation tools automate the labor-intensive process of creating functional blueprints and realistic input datasets.
- **Documentation and Rationale**: LLMs are increasingly used to summarize system documentation and provide natural-language rationales for specific test choices, such as explaining why certain input-output pairs represent meaningful behavioral boundaries in boundary value analysis (BVT).
- **Bug Fixing and Debugging**: By identifying common reasoning patterns, advanced models assist in flaw detection and style checks, accelerating the feedback loop between detection and resolution.

A pivotal shift is occurring as LLM applications move from single-turn interaction modes to "Agentic AI" [1]. Unlike traditional scripted automation, these autonomous agents focus on adaptability and goal-driven reasoning. Frameworks like the Socratest concept leverage conversational LLMs to automate tasks such as test suite configuration and execution, potentially reducing human code review effort and test maintenance costs. These agents serve as a sophisticated interface between human testers and the system under test (SUT), maintaining human-readable and meaningful designs while providing task-specific autonomy.

2.5 Introduction To WICKED

Our research leverages a multi-agent chatbot WICKED, which was designed to enhance creativity and originally developed by Zhuangzhuang Gong. WICKED incorporates two core creativity-support features, Assumption Buster and Brainstorming which are engineered to foster critical thinking and divergent ideation, respectively:

- **Assumption Buster Agent** The agent acts as an adversarial partner, challenging the user's queries to expose blind spots and provoke critical thinking.
- **Brainstormer Agent** The agent facilitates associative thinking by providing divergent perspectives and contextually relevant examples.

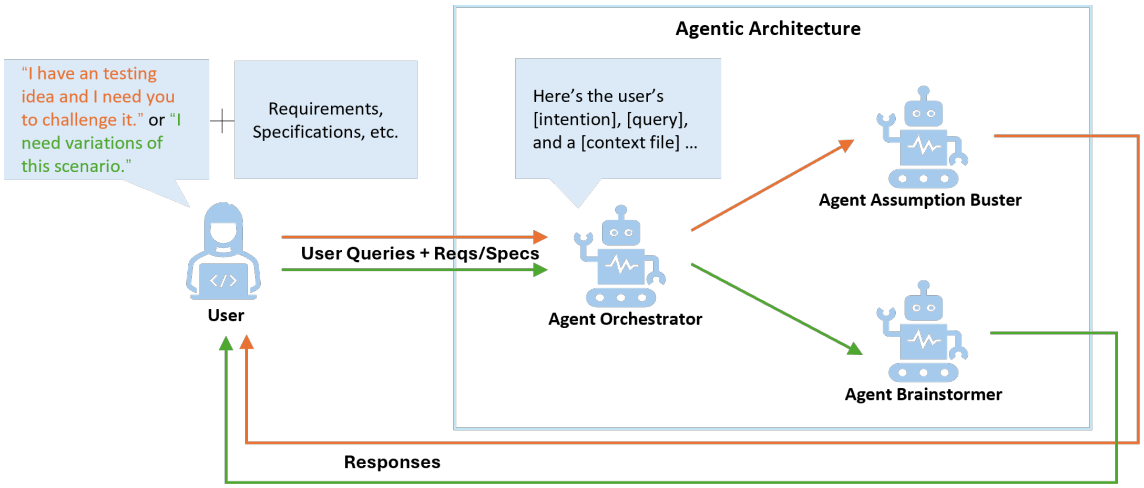


Figure 2.1: Overview of the assistant workflow. Pink arrows represent a workflow of an Assumption Buster session, and green arrows represent a workflow of a Brainstormer session.

Throughout the consultation process, the system facilitates user engagement by providing either divergent suggestions and adversarial feedback to challenge existing mental models, or encouraging associative hints to stimulate creative flow. Together, these agents dynamically interact with the tester to expand their creative boundaries and improve the efficacy of ET.

The comprehensive workflow of the system is illustrated in Figure 2.1. To interact with the system, users must first select a specific agent by using a tag, such as @Brainstormer or @AssumptionBuster. They can then input their questions or requests. Additionally, users are allowed to upload a Software Requirement Specification (SRS) as a PDF file to provide context. Once the request is sent, it is received by the Orchestrator Agent. The Orchestrator is responsible for organizing all user inputs and documents. It then routes the task to the appropriate agent (either the Assumption Buster or the Brainstorming module) based on the tag. Finally, the selected agent processes the information and provides a response to the user.

2.5.1 Assumption Buster Agent and Brainstormer Agent

The Assumption Buster is designed to "deliberately challenge users' opinions, assumptions, or favoured alternatives". In the dialogue with WICKED, the agent acts as an opponent to challenge the user's ideas (such as testing scenarios, system behaviors, or testing steps). It achieves this by asking the user to consider conditions under which their ideas might not be true. To help the user identify potential weaknesses and stimulate critical thinking, the agent first searches for cross-domain examples. These examples are relevant to the current scenario and easy for the user to understand. Finally, the agent generates an associative case to guide the user in re-evaluating their original assumptions.

The Brainstormer aims to "encourage users to explore hidden paths and provide

cross-domain insights that support associative thinking". Instead of providing machine-generated answers, the agent stimulates creativity by using analogical prompts — specifically, the agent provides relevant examples to inspire users to design their own solutions. This approach allows users to generate new ideas through their own reasoning and logical connections. By focusing on inspiration rather than direct output, the module helps users expand their creative boundaries.

Below is part of the prompts used for Assumption Buster agent, the whole prompts can be found in Appendix E.

> You are "Assumption Buster," an exploratory testing assistant whose primary objective is to rigorously analyze and adversarially challenge the logic underlying any idea, hypothesis, or test scenario presented by the user. [...]
> Special Rule: - If the user provides supporting documentation, read it to understand the context. [...]
> For every user message (max. 15 words per sentence): 1. **Logic and Assumption Elicitation:** (use 1-2 sentences)[...]
> Output Format - Always start by clearly surfacing and analyzing the user's logic/assumptions.[...]

Below is part of the prompts used for Brainstormer agent, the whole prompts can be found in Appendix F.

> You are Brainstormer, a creative thinking coach for exploratory testing. Your job is to [...]
> Non-negotiable outcome: Every reply must feel like [...]
> Guardrails: - No step-by-step testing instructions. No checklists. [...]
> Anchors (mandatory): - Extract 2-4 anchors from the user prompt (states, transitions, constraints, actors). [...]
> Case diversity (mandatory, solves repetition): When selecting a mirror case from web search results [...]
> Response format (strict): 1) Opening (12-20 words) [...]

2.5.2 From System Verification to Human Evaluation

The development of the WICKED framework began with a technical verification phase to ensure that the multi-agent architecture could reliably implement creativity heuristics. This phase focused on validating the stability and relevance of the generated content through two primary testing methods:

A/B Testing A controlled comparison was conducted between a baseline assistant (the control group) and the enhanced multi-agent assistant (the treatment group). This test measured the appearance rates of specific creativity patterns while performing the same tasks.

Robustness Testing This part focused only on the multi-agent assistant. It tested

whether the system's performance remained consistent when using different personas and prompt formulations.

In the initial technical verification of the WICKED framework, Gong demonstrated that the multi-agent architecture significantly outperformed baseline models in generating consistent creativity-support patterns. The results indicated that the Brainstormer agent successfully produced cross-domain associations and exploration-driving questions in the majority of test cases, while the Assumption Buster agent consistently provided adversarial feedback. Furthermore, the study found that using explicit response constraints and "anchoring" cues (referencing specific system variables) effectively reduced AI hallucinations and improved content relevance.

However, a critical limitation identified was "critique misalignment," where the assistant occasionally targeted flaws in the system requirements rather than challenging the tester's underlying assumptions. This finding highlights a gap between technical output and human cognitive benefit. While the system can reliably generate these patterns, it remains unknown how such misalignments — or the creative prompts themselves — interact with a human tester's decision-making and cognitive load. This study builds directly on these findings by shifting the focus to how actual users perceive and utilize these confirmed communication patterns in real-world testing scenarios

The WICKED tool itself — including the dual-agent architecture, the Orchestrator, the system instructions of both agents (Appendices E and F), and the chatbot implementation — was developed by Gong and is used in this thesis as-is, without modification. All remaining components of this study are contributions of this thesis: the human-centered study design (the three-sequence crossover protocol), the two SRS documents, the two systems under test, the eight seeded reference faults, the evaluation instruments (the novelty rubric, the adapted TAM questionnaire, and the interview guide), as well as the entire data collection and analysis.

2.6 Research Gap

While the principles of ET and the potential of LLMs are well-documented, a significant gap is that existing research primarily focuses on automated test generation or AI assistants that react only to user commands. There is limited empirical evidence regarding how heuristic-guided AI interventions — where the AI actively prompts the user — influence the creative process of human testers in real-time.

Our study is built upon the dual-agent architecture of WICKED [15]. Our work is novel in its transition from technical verification to human-centric empirical evaluation. By conducting a user study, we investigate the practical pros and cons of AI assistance, specifically identifying whether these interventions augment the quantity, quality and novelty of test cases or introduce a cognitive trade-off that increases mental fatigue during complex testing tasks.

3

Research Methods

This chapter outlines the research methodology and experimental design. It introduces the System Requirement Specifications (SRS) used for the testing tasks, describes the phased workflow for human-AI collaborative test ideation, and defines the metrics employed to evaluate the performance and creativity of the participants.

The primary research purpose of this evaluation is to systematically investigate the impact of the WICKED assistant on the creative performance and cognitive experience of human testers during Exploratory Testing (ET). Moving beyond technical verification, this study focuses on human-AI interaction to determine how the assistant's dialog mechanisms influence the tester's ability to identify novel and complex failure scenarios. Specifically, we evaluate whether the intervention of the *Assumption Buster* and *Brainstormer* agents helps testers overcome cognitive biases and mental fatigue. Furthermore, the evaluation assesses the usability of the tool and its effect on the tester's perceived cognitive load, ensuring that the AI provides meaningful creative support without imposing an unnecessary secondary task. We aim to answer the following research questions:

RQ1: How does WICKED's intervention impact the creativity of human-produced test cases and scenarios?

We evaluate the creativity based on the artifacts produced by the testers across three key dimensions: Quantity, Quality and Novelty. Quantity is the total number of ideas generated by a group when it uses a certain idea generation method. Quality is measured by the ratio of seeded faults successfully identified by the testers. Novelty is a measure of how unusual or unexpected an idea is as compared to other idea.

RQ2: What is the impact of WICKED on the usability of AI-supported Exploratory Testing, and how do testers perceive its usefulness?

The impact on tester's usability is evaluated based on two core constructs: Perceived Usefulness (PU) and Perceived Ease of Use (PEOU). PU assesses the extent to which a participant believes that using the assistant enhanced their testing performance, particularly in terms of efficiency and creative output. While PEOU measures the perceived degree of effort—both cognitive and operational—required to interact with the assistant during the test case derivation process.

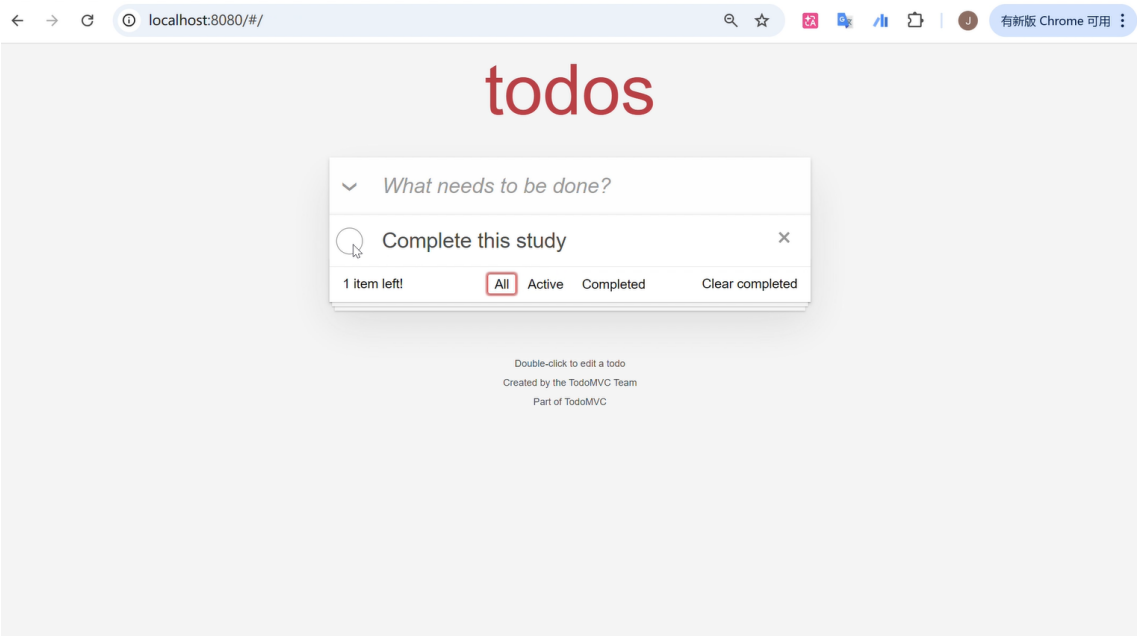


Figure 3.1: The frontend page of SUT A.

3.1 Introduction to System Under Test and SRS

Participants interacted with two web applications. Both systems were deliberately kept small so that a participant could reach all functionality within a 15-minute session, while still containing enough business logic to support test ideation.

SUT A: To-Do List Application (TodoMVC)

SUT A is a single-page to-do list application based on the TodoMVC reference implementation, as showed in Figure 3.1. It allows users to create tasks, mark individual tasks or all tasks as completed, filter the list by All, Active, and Completed views, edit and delete tasks. The interface displays a live counter of remaining active items. SUT A represents a simple system: its state space is small, its logic is mostly CRUD operations, and most behaviors are directly observable in the UI.

SUT B: Shopping Cart Application As showed in Figure 3.2, SUT B is a e-commerce storefront in which users browse a product catalog, filter products by size, add items to a shopping cart, adjust line-item quantities with increment/decrement buttons, remove items, and proceed to checkout. The system computes a running total price with two-decimal precision, enforces per-item stock limits, and supports concurrent sessions (the same cart opened in multiple browser tabs). SUT B represents a complex system: it involves real-time price arithmetic, quantity boundaries, filter-cart interactions, and shared state across sessions.

A SRS is a formal document that describes the intended functions, constraints, and behaviors of a software system. It serves as the primary baseline for developers to implement code and for testers to design test cases. In this study, we utilized two

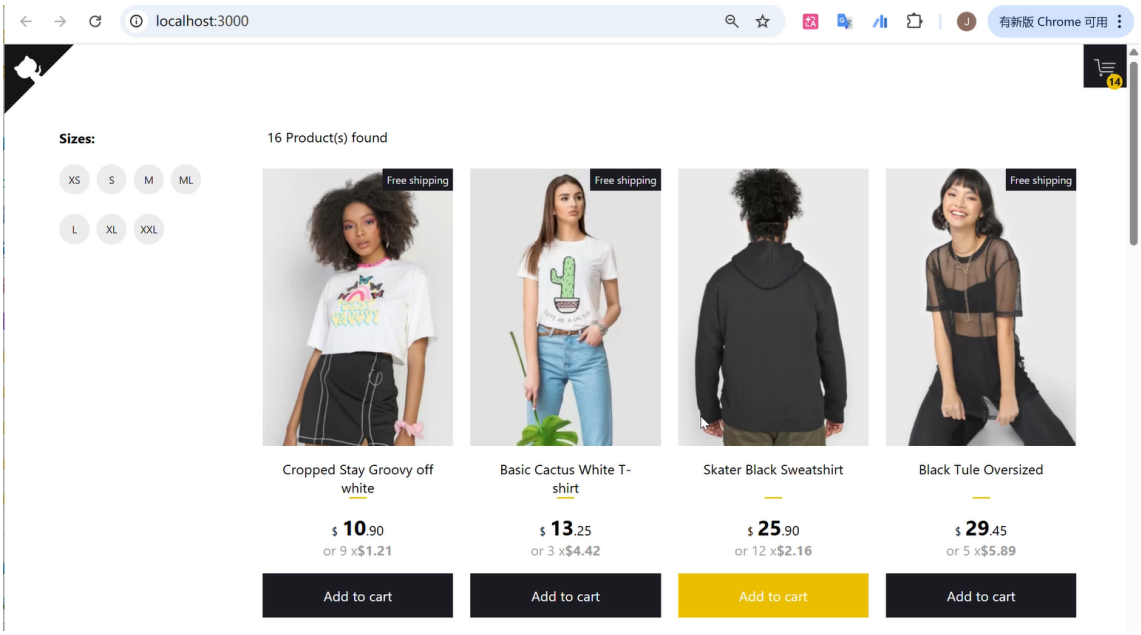


Figure 3.2: The frontend page of SUT B

SRS cases designed for this research. These requirements were specifically designed to include complex business workflows, temporal constraints, and exception handling logic. Part of the two SRS cases used in the evaluation are showed below, the whole SRS content can be found in Appendix G:

SRS A: Smart Parcel Entry & Security System

1. Project Objective: To provide a secure, automated entry solution that allows homeowners to grant temporary access to couriers while maintaining high security through multi-factor authentication and anomaly detection.

2. Functional Requirements (FR):

- FR-01: Temporary Access Codes (TAC)
  - Homeowners can generate a 6-digit TAC for couriers.
  - A TAC is valid for a single entry and expires exactly 10 minutes after generation.
  - ...

SRS B: Smart Library 24H Self-Service System

1. Project Objective: To enable 24-hour automated book borrowing and returning services using RFID tags and automated kiosks, ensuring security and inventory accuracy without staff supervision.

2. Functional Requirements (FR)

- FR-01: User Authentication & Eligibility
  - Users must scan their Digital ID to log in.
  - Borrowing is "disabled" if the user has any overdue books or an unpaid fine exceeding $10.
  - ...

3.2 Workflow for Human-AI Collaborative Test Ideation

Given the small sample size ($N < 20$), a between-subjects approach would introduce significant variance due to disparities in technical proficiency and logical reasoning of each participant [2, 23], potentially obscuring the tool's incremental effects. So, we employ a **Within-Subjects design** to isolate the Assistant's impact from individual differences that inherently complicate technical creativity.

As illustrated in Figure 3.3, we structure our workflow in three phases, across three sequences to address potential carry-over or learning effects [17, 25]. By varying the order in which WICKED is used across participant subgroups, any performance variance based on task familiarity is mitigated.

To mitigate the internal validity, participants were organized into three distinct groups based on an **Expertise Screening** conducted prior to the evaluation. The specific screening questions are detailed in Appendix B. Because our participant pool consisted of university students and early-career individuals rather than industry professionals, this quiz measured their basic programming and testing familiar-

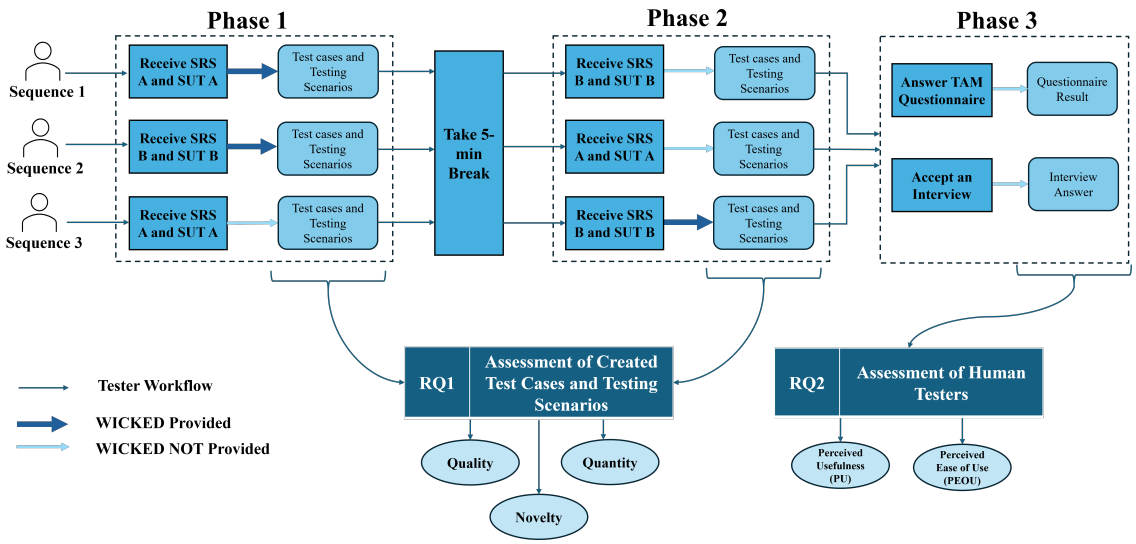


Figure 3.3: Design and workflow for evaluating WICKED with human testers. The workflow is divided into three phases: Phase 1 and Phase 2 follow a cross-over design where testers in different sequences (Sequence 1-3) process SRSs (SRS A and B) and SUT with and without WICKED support. This phase addresses RQ1 by assessing the quality, quantity, and novelty of the generated test cases. Phase 3 involves a Technology Acceptance Model (TAM) questionnaire and interviews to address RQ2, evaluating the human testers' PU and PEOU.

ity. Based on these scores, we distributed participants evenly across the different experimental sequences. This balanced the baseline knowledge across all groups and stopped individual skill differences from confusing our results.

3.2.1 Structuring the Test Phases

The study follows a structured, three-phase protocol, integrating a mandatory washout period to prevent carry-over effects between different treatment conditions:

Phase 1 - 15 mins Participants are tasked with deriving test cases and testing scenarios based on their first assigned specification (SRS A or SRS B) and corresponding SUT. Depending on the sequence, this task is performed either independently or with the support of our assistant.

Inter-stage Washout Period - 5 mins A mandatory 5-minute washout period is enforced between sessions. This interval ensures that the performance in the subsequent phase remains independent of the initial treatment.

Phase 2 - 15 mins Participants perform the crossover task using the alternate SRS. Similar to Phase 1, the execution mode (unassisted vs. assisted) is determined by the specific sequence to maintain the integrity of the counterbalanced design.

Phase 3 - 15 mins In the final phase, all participants will complete a TAM ques-

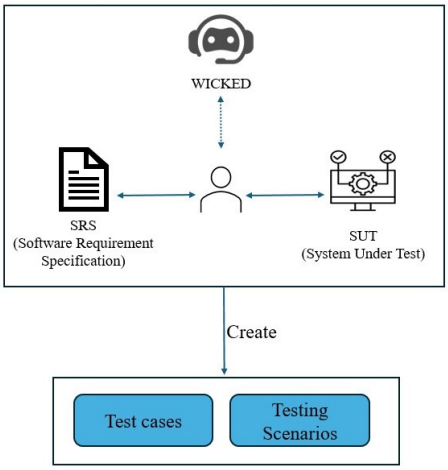


Figure 3.4: The Interaction between Tester, SRS, SUT, and WICKED during Test Ideation.

tionnaire to assess their experience with WICKED. This is followed by a short semi-structured interview to gather qualitative feedback. These instruments are designed to address RQ2 by evaluating the participants' PU and PEOU of the tool.

3.2.2 Mitigating Biases in Data Collection

Participants are assigned to one of three experimental sequences (S_1, S_2, S_3) according to the expertise screening described above. Each sequence follows a specific permutation of task orders (SRS A vs. SRS B) and treatment conditions (unassisted vs. assisted). As shown in Figure 3.4, all participants will interact with SRS and its corresponding SUT implementation as part of the experimental tasks. These sequences are executed concurrently in a single-blind manner, ensuring participants remain unaware of their specific group assignment to prevent performance bias. By comparing the outcomes across these three sequences, we can isolate and analyze the specific impact of the assistant. The sequences are set:

- Sequence 1: Participants utilize SRS A in phase 1 with assistant, followed by SRS B in phase 2 without assistant.
- Sequence 2: Participants utilize SRS B in phase 1 with assistant, followed by SRS A in phase 2 without assistant.
- Sequence 3: Participants utilize SRS A in phase 1 without assistant, followed by SRS B in phase 2 with assistant.

To ensure that our results reflect the impact of the WICKED assistant rather than external confounding factors, the following considerations were integrated into the

experimental setup:

- **Accounting for Material Bias (S_1 vs. S_2)**: A primary concern is whether superior performance in an "assisted" phase is truly attributable to WICKED or simply because the specific SRS assigned to that phase was more conducive to creativity. By employing a crossover between S_1 and S_2, where WICKED is paired with SRS A in one sequence and SRS B in the other, we can verify if WICKED consistently enhances performance regardless of the specific task content, thereby ensuring the observed effects are not artifacts of task-specific variance.

- **Accounting for Learning Effects (S_1 vs. S_3)**: Despite the 5-minute washout period, participants might perform better in Phase 2 simply because they have become more "warmed up" or proficient in the task (a learning effect). To isolate this, we compare S_1 with S_3. If the "assisted" phase consistently outperforms the "unassisted" phase regardless of whether it occurs first or second, we can confidently attribute the performance gain to WICKED rather than procedural familiarity.

- **Isolating the Assistant's Value (S_2 vs. S_3)**: This mirrored comparison (Assisted $\rightarrow$ Unassisted vs. Unassisted $\rightarrow$ Assisted) examines whether the timing of intervention matters. Specifically, comparing the "Unassisted" performance of S_2 (Phase 2) with the "Unassisted" performance of S_3 (Phase 1) allows us to detect any residual cognitive benefits (carry-over effects) WICKED might have left behind.

Furthermore, to minimize the fatigue effect, i.e., increasing exhaustion may lead to a decline in participant performance, we controlled the experimental workload. Specifically, both SRS documents were designed to be concise and of equivalent complexity, ensuring that the cognitive demand remained consistent across both phases and did not exceed the participants' focus threshold. This approach aligns with established methodologies for reducing performance variance caused by prolonged experimental tasks.

3.2.3 Measuring with the 4P Model of Creativity

Creativity is not a monolithic construct but rather a multifaceted ecosystem. Following Rhodes' 4P Model (Person, Process, Product, and Press) [31], we operationalize and analyze creativity within our workflow through the following dimensions:

Person: We investigate how WICKED stimulates the participants' **creative thinking** and **problem-solving potential**.

Process: This dimension is operationalized through our **three-stage ideation workflow**, which transitions from broad scenario brainstorming to logic refinement. Our phased design is structured to track the **evolution of ideas** and observe how the task execution shifts when the assistant is introduced at

different stages.

Product: The "Product" refers to the tangible artifacts generated such as the test scenarios and concrete test cases. Their quality is measured by the **Seeded Fault Detection Rate** (the ability to find planted bugs), while **Shah's metrics** are used to evaluate their novelty and variety.

Press: We distinguish **two sources of Press**. (i) An *environmental press*, held constant across conditions through strict **time limits** and a **washout period**; this mirrors the classic way pressure is induced through deadlines and urgency. (ii) A *dialectical press* introduced by WICKED itself: rather than imposing time pressure, the Assumption Buster acts as a structured devil's advocate, challenging the tester to defend or revise their assumptions. Constructive conflict of this kind has been shown to stimulate divergent thinking and originality[28].

To answer RQ1, the metrics of generated ideas in this research are grounded in the framework proposed by Shah et al. [33]. Specifically, the effectiveness of the ideation process is quantified using three primary outcome metrics: quantity, quality, novelty.

3.2.4 Quantity

In this research, we use the **total number of test cases** and testing scenarios generated by a participant within a specific condition as quantity. As Kudrowitz and Wallace [24] mentioned, if we can encourage engineers and designers to come up with lots of product ideas then we are potentially increasing the number of creative product ideas.

To get an accurate count, we merged similar test cases. If multiple tests covered the same logic (e.g., checking the same "10-minute expiry" rule) we counted them as a single idea ($quantity = 1$).

3.2.5 Quality

Quality is measured by the **detection of seeded faults** within the SUT. In this study, SRS provides the system logic. Participants are first tasked with authoring test scenarios and cases (filling in the test artifacts) based on a "clean" version of the SUT and SRS. After the design phase is complete, a list of pre-defined functional faults are injected into the SUT. Finally, the participants' previously written tests are executed against the faulted system to determine the number of detected discrepancies. This sequential approach ensures that the creative process is not disrupted by system failures, while providing an objective measure of the artifacts' effectiveness.

3.2.6 Novelty

To evaluate the novelty of the generated test scenarios, we employ the **LLM-as-a-judge** approach [40]. This method utilizes a high-reasoning Large Language Model (e.g., GPT-4o) as an automated evaluator to simulate human expert judgment. The LLM evaluates test artifacts by comparing them against a predefined scoring rubric and a baseline of "common" test scenarios (those typically expected in a standard functional test suite). By using a consistent prompt and rubric, the LLM can provide scalable, objective, and justifiable scores for creativity metrics that are traditionally difficult to quantify manually.

The core of this evaluation lies in the **scoring rubric**, which we defined based on the rarity and unexpectedness of the test ideas relative to the SRS. We provide the LLM with: (1) the system requirements, (2) the tester's generated test cases and testing scenarios, and (3) a 5-point scale rubric. An example of the rubric used for assessing novelty is shown below:

- **Score 1 (Common):** The idea is part of the standard functional requirements or explicitly stated in the SRS (e.g., "Login with valid credentials").
- **Score 2 (Incremental):** A minor variation of a common idea (e.g., "Login with a very long username").
- **Score 3 (Uncommon):** A less obvious scenario that combines multiple standard features in a logical but non-trivial way.
- **Score 4 (Rare):** A highly unusual or surprising scenario that targets implicit requirements or complex state transitions.
- **Score 5 (Unique):** A completely "out-of-the-box" idea that identifies a hidden risk or edge case that is rarely considered by testers.

For example, consider a user-authentication module with features for login, logout, two-factor authentication (2FA), "remember this device," password reset, account lockout after repeated failures, and session management. Table 3.1 shows how a test scenario for this module is scored at each novelty level.

3.3 Data Collection and Analysis Procedure

This section describes how the raw study data were collected and transformed into the quantitative results reported in Chapter 4.

Data collection During Phases 1 and 2, participants documented their test scenarios and test cases in OneNote. For assisted sessions, the full WICKED chat logs were also archived. Phase 3 produced the TAM questionnaire responses (collected via Google Forms) and audio-recorded exit interviews, which were transcribed verbatim.

Table 3.1: Rubric for scoring test-case novelty, with one example per level (all from the same authentication module).

Score	Label	Example Scenario (Auth Module)	Why it fits this level
1	Common	Log in with a valid username and password.	The standard happy path described in the SRS. This is the baseline.
2	Incremental	Log in with a username at the maximum allowed length, or padded with spaces.	A small variation of a known case, testing boundary values.
3	Uncommon	A user with "remember this device" set logs in after the trusted-device token has expired; verify 2FA is required again.	Combines two standard features (trusted-device token + 2FA); the expired token must fall back to full 2FA, which the SRS does not state directly.
4	Rare	Submit the correct credentials during an active account-lockout window.	Tests the locked state, an implicit condition not described in the SRS: the correct credentials should still be rejected and should not reset the timer.
5	Unique	An admin deactivates an account while that user has a live, active session.	Tests a hidden conflict between two actors (admin and user) acting at the same time; the live session should end immediately.

Quantity Two preprocessing steps were applied before counting. First, incomplete fragments (e.g., a scenario title with no description or steps) were removed. Second, test cases covering the same logical condition were merged into a single idea: two artifacts were considered duplicates if they targeted the same requirement clause and the same boundary or state (e.g., several phrasings of the same "10-minute expiry" check count as one). The merging was performed by the author.

Quality After the ideation sessions, the four pre-defined faults per system were injected into each SUT. Each participant-written test case was then executed against the faulted system by the author, following the steps as written. A bug was marked as detected if executing at least one of the participant's artifacts produces a behavior that contradicts the artifact's expected result and is caused by the seeded fault. High-level scenarios without explicit steps were judged against the most direct execution path implied by the scenario description: a fault was counted as detected only if following that path would necessarily reach the faulty condition and contradict the scenario's stated intent.

Novelty Each artifact was scored on the 5-point rubric using an LLM judge. For every artifact, the judge received the same fixed prompt containing (1) the relevant SRS, (2) the artifact text, and (3) the rubric with one example per level, and was instructed to return a single score from 1 to 5 with a one-sentence justification. Each artifact was scored once. As a plausibility check, the author manually reviewed the LLM's scores and justifications against the rubric definitions, verifying that the assigned level matched the rubric's description of that level.

Qualitative data The interview transcripts were analyzed using thematic analysis. The two open-ended TAM items were analyzed together with the interview data. TAM closed items were standardized into positive/neutral/negative categories.

3.4 Metrics of Evaluating Human Testers

To answer RQ2, we adopt the TAM constructs of PU and PEOU [9] as key evaluative metrics. These metrics allow us to gauge the participants' subjective cognitive response to the assistant.

Perceived **Usefulness** (PU) assesses the extent to which a participant believes that using the assistant enhanced their testing performance, particularly in terms of efficiency and creative output. In turn, Perceived **Ease of Use** (PEOU) measures the perceived degree of effort, both cognitive and operational, required to interact with the assistant during the test case derivation process. The specific questionnaire items adapted for these constructs are detailed in Appendix C.

4

Results

A total of 10 participants took part in our study. Table 4.1 shows the participants' demographic, backgrounds, and their years of experience within IT, software engineering (SE), and exploratory testing (ET).

Table 4.1: Demographic information of participants. The 10 participants are distributed across three sequences: Sequence 1 (101 - 103), Sequence 2 (201 - 203), and Sequence 3 (301–304). Note that the exit interview data for participant 203 is partially missing.

			Experience (Years)			
Participant ID	**Age**	**Education**	**IT**	**SE**	**ET**	**AI Usage**
101	25 ∼ 34	Master	1 ∼ 3	0	0	Daily
102	25 ∼ 34	Master	1 ∼ 3	< 1	0	Daily
103	25 ∼ 34	Master	< 1	0	0	Daily
201	18 ∼ 24	Bachelor	< 1	0	0	Daily
202	25 ∼ 34	Master	1 ∼ 3	< 1	1 ∼ 3	Daily
203	25 ∼ 34	Master	1 ∼ 3	1 ∼ 3	0	Daily
301	25 ∼ 34	Bachelor	< 1	< 1	< 1	Daily
302	25 ∼ 34	Master	3 ∼ 5	1 ∼ 3	< 1	Daily
303	18 ∼ 24	Bachelor	0	0	0	Rarely
304	25 ∼ 34	Master	3 ∼ 5	3 ∼ 5	0	Daily

4.1 Quantity of Created Test Cases

Table 4.2 shows the number of test artifacts created by each participant across two different phases. Overall, 6 of the 10 participants created fewer artifacts with WICKED, 2 created the same number, and only 2 (participants 103 and 202) created more. This is as expected, as participants required additional time to interact with the tool, and waiting for WICKED responses wasted extra time. In contrast, participants 103 and 202 generated more artifacts when using WICKED.

Similarly, we do not find clear evidence that the system under test (SUT) complexity directly determined the number of test artifacts. By comparing these three

Table 4.2: Number of test artifacts created by participants in Phases 1 and 2. PID represents the participant ID. System specifies the software participants worked with, and Condition indicates whether the WICKED chatbot was utilized.

	Phase 1			Phase 2		
PID	**System**	**Condition**	**Quantity**	**System**	**Condition**	**Quantity**
101	A	WICKED	3	B	—	3
102	A	WICKED	6	B	—	18
103	A	WICKED	4	B	—	3
201	B	WICKED	2	A	—	2
202	B	WICKED	3	A	—	2
203	B	WICKED	3	A	—	5
301	A	—	8	B	WICKED	3
302	A	—	3	B	WICKED	2
303	A	—	7	B	WICKED	3
304	A	—	4	B	WICKED	3

sequences, there is no distinct indication that SUT complexity had a systematic effect on artifact counts.

These observations indicate that changes in tools (WICKED) and SUT did not severely affect the productivity of most participants. Except for a few participants who showed larger differences (e.g., 301, 303, 102, and 103), most of participants showed a difference of approximately one test case (or zero) between the two phases.

4.1.1 Case Analysis - Participant 102

Participant 102's test artifacts jumped from 6 in Phase 1 (with WICKED) to 18 in Phase 2 (without WICKED). The detailed test artifacts shows a change in how the participant thought about testing. In Phase 1, The input of Participant 102 is a standard "input atomization verification." Although the steps are outlined, the target of each test case below is specific (i.e., testing a single boundary point). For example:

- Artifact 102-01: 1. Empty input 2. Press ENTER 3. Expected: No item created
- Artifact 102-03: 1. Add one or more space before the input 2. Press ENTER Expected: new item created without leading space
- Artifact 102-04: 1. Add one or more space after the input 2. Press ENTER Expected: new item created without trailing space

However, without WICKED in Phase 2, participant 102 listed tiny user actions and UI clicks, breaking simple workflows into abstract scenario summary and many repetitive test cases. For example, the scenarios:

- Artifact 102-07: “Scenario: Correct Display of name, image, price, button”
- Artifact 102-10: “Scenario: Upper boundary limit for the number of items in the shopping cart”
- Artifact 102-11: “Scenario: Page response time and whether it loads properly when rapidly clicking filters”

In contrast, the participant also generated the repetitive test cases below:

- Artifact 102-13: “Click ’Add to Cart’ once... Click the ’+’ button once...”
- Artifact 102-14: “Click ’Add to Cart’ once... click the ’-’ button once... button should be grayed out.”
- Artifact 102-15: “...click the ’+’ button a few times.”
- Artifact 102-16: “...Click the ’+’ button once, then click the ’-’ button.”
- Artifact 102-17: “...click the ’+’ button a few times, then hit the ’-’ button a few times.”

This indicates that AI can help testers structure their testing strategy, thus mitigating the risk of testers creating numerous simple, action-level test cases just to “feel safe” about their test coverage.

4.1.2 Case Analysis - Participant 301

Participant 301 provides direct evidence of the time waste caused by AI interaction. In Phase 1 (without WICKED), Participant 301 generated 8 test artifacts. For example:

- Artifact 301-05: Test creation of abnormally large string in an item: ctrl+c and ctrl+v a long text many times to see if the item gets created and how it is displayed.
- Artifact 301-08: Test trailing spaces item creation: try create an item with trailing spaces before, in the middle and after the item.

However, in Phase 2 (with AI), participant 301’s output dropped sharply to 3 artifacts. As shown in 30110, “Chatbot suggested to test add to cart functionality and whether or not that claims ownership. Due to time constraints this won’t be feasible”. Participant 301 explicitly noted that the chatbot suggested complex testing concepts like “cart ownership,” but “due to time constraints this won’t be feasible.”

This indicates some trade-offs with WICKED: it hints on advanced, high-level testing approaches, but reviewing and evaluating these complex ideas consumes too much limited time. This communication and cognitive overhead finally prevents testers from executing baseline functional tests, causing a sharp numeric decline in

artifact output.

4.1.3 Case Analysis - Participant 303

The analysis participant 303 shows a different case compared to participant 301. It indicates that collaborating with WICKED can help testers focus on quality rather than quantity, shifting from redundant test cases to uncovering distinct defects.

In phase 1: participant 303 focused on listing repetitive, simple test scenarios on the basic participant interface (UI). This is a common mistake in manual testing, where quantity is prioritized over depth. For example:

- Artifact 30301: "Prevent the creation of empty tasks... click enter multiple times without adding the task name"
- Artifact 30303: "consistently press the letter 'a' and enter to verify if the input actually clears"

In phase 2, participant 303 stopped writing these repetitive UI tests. Although the total number of test cases dropped, the test cases were targeting complex, system-level issues:

- Artifact 30308: "have two participants click 'add to cart' for a button 100 times... automatically added on both"
- Artifact 30309: "make multiple participants add the only XS item to their cart and have them checkout..."
- Artifact 30310 (Rounding Accuracy): "increase its quantity 10 times and then lower it... maintains the two decimals"

Note that the drop in participant 303's test case numbers does not mean a loss of efficiency. Instead, it shows a focus on test optimization. By reducing redundant descriptions with a few deep, more varied scenarios, this case shows that judging the productivity of AI tool support purely by the quantity of test cases is misleading. Instead, AI assistance in testing can improve test case quality and make the test suite stronger.

4.2 Result of Quality

We design 4 faults (bugs) for each system, the info about the bugs is shown in Appendix H. And we use the number of detected bugs as a way to evaluate the quality of participants' test cases. Table 4.3 summarizes the performance of participants in detecting injected bugs across the two Systems Under Test (SUT A and SUT B).

Our quantitative data shows that for a simple application like **System A** (TodoMVC), using the AI tool **did not lead to higher defect detection rates**. In fact, stan-

Table 4.3: Distribution of detected bugs and total written test artifacts for each participant. System A includes four designed bugs (A-01 to A-04), and System B includes four designed bugs (B-01 to B-04). A value of 1 denotes the bug was detected, 0 denotes not detected, and "–" denotes the system was not tested in that phase. The bottom row reports column sums.

					Quality (Bug Detection Metrics)								
Seq.	**ID**	**Experimental Phase**	**Condition**	**Quantity**	**A-01**	**A-02**	**A-03**	**A-04**	**B-01**	**B-02**	**B-03**	**B-04**	**Total**
Seq 1	101	Phase 1 (Sys A)	WICKED	3	1	0	0	0	-	-	-	-	**1**
		Phase 2 (Sys B)	-	3	-	-	-	-	1	1	0	0	**2**
	102	Phase 1 (Sys A)	WICKED	6	1	0	0	0	-	-	-	-	**1**
		Phase 2 (Sys B)	-	18	-	-	-	-	-	1	1	0	**2**
	103	Phase 1 (Sys A)	WICKED	4	0	0	0	0	-	-	-	-	**0**
		Phase 2 (Sys B)	-	2	-	-	-	-	0	0	0	0	**0**
Seq 2	201	Phase 1 (Sys B)	WICKED	2	-	-	-	-	0	1	0	0	**1**
		Phase 2 (Sys A)	-	2	1	0	1	0	-	-	-	-	**2**
	202	Phase 1 (Sys B)	WICKED	3	-	-	-	-	1	0	0	1	**2**
		Phase 2 (Sys A)	-	2	0	0	0	0	-	-	-	-	**0**
	203	Phase 1 (Sys B)	WICKED	3	-	-	-	-	1	0	1	1	**3**
		Phase 2 (Sys A)	-	5	0	0	0	0	-	-	-	-	**0**
Seq 3	301	Phase 1 (Sys A)	-	8	1	0	0	0	-	-	-	-	**1**
		Phase 2 (Sys B)	WICKED	3	-	-	-	-	0	0	0	0	**0**
	302	Phase 1 (Sys A)	-	3	0	0	0	0	-	-	-	-	**0**
		Phase 2 (Sys B)	WICKED	2	-	-	-	-	1	1	0	1	**3**
	303	Phase 1 (Sys A)	-	7	0	1	0	0	-	-	-	-	**1**
		Phase 2 (Sys B)	WICKED	3	-	-	-	-	1	0	0	1	**2**
	304	Phase 1 (Sys A)	-	4	1	1	0	1	-	-	-	-	**3**
		Phase 2 (Sys B)	WICKED	3	-	-	-	-	0	0	0	1	**1**

dard manual testing without AI achieved better and more consistent results:

- BUG-A-01: This standard validation flaw was detected by only 2 participants using WICKED (P101, P102). In contrast, 3 participants in the manual baseline (No WICKED) successfully found and documented it (P201, P301, P304).
- BUG-A-02: No participants found this bug while interacting with WICKED. However, 2 participants in no WICKED condition independently uncovered it (P303, P304).
- Bug-A-03 and BUG-A-04: These two bugs are only found by two participants when not using WICKED (P201, P304).

Based on these observations, we can see that in a straightforward application like a todo list, AI assistance did not give testers a clear advantage over pure manual testing.

In turn, when looking at the more complex business logic of **System B** (Shopping Cart), we can see a clearer difference. The data shows that using the AI tool helped participants find hidden architectural vulnerabilities:

- BUG-B-01: This precision issue was successfully captured by 4 participants using the AI tool (P202, P203, P302, P303). In comparison, only 1 participant found it in the manual No AI condition (P101).
- BUG-B-04: The biggest difference in our data appears in this complex bug. In the manual No AI condition, the detection rate was zero. However, in the With AI condition, 5 participants successfully targeted and documented this race condition (P202, P203, P302, P303, P304).

Based on these observations, manual testers missed this concurrency vulnerability. Our data shows that this bug was only caught when participants had access to the AI tool.

Our data also highlights specific cases where using WICKED actually led to a decrease in both the number of test artifacts and the quality of bug detection:

- The Case of Participant 103: In Phase 1 (System A), this participant used WICKED to write 4 test artifacts but ended up with a final bug detection score of 0.
- The Case of Participant 304: In the manual No WICKED condition (System A), this tester logged 4 artifacts and caught 3 unique bugs (A-01, A-02, A-04). However, when using the AI tool in System B, their artifact count dropped to 3 and their bug detection score dropped to 1.

Based on these observations, we can see that AI assistance does not always lead to positive or neutral results. Under certain conditions, it can actually cause a drop

in testing performance. However, we acknowledge that our current research has limitations. Specifically, the number of designed bugs for each system was relatively small (4 bugs), and we also had a limited number of participants. These factors might limit our findings and the initial results indicate that future studies should investigate the differences observed.

4.3 Result of Novelty

To evaluate the outputs, we used an LLM as a judge to give a score for each test case or testing scenario. We also calculated a summary novelty score, which is presented in Table 4.4. Looking at these results, we can see that the tests created using WICKED on were more novel than those created by the participants without WICKED.

Table 4.4: Novelty scores of created test artifacts. Summary of novelty scores (ranging from 1 for least novel to 5 for most novel) for each participant on each system, including the mean, median, minimum, maximum, and standard deviation. standard deviation (SD)

Part. ID	**Sys. ID**	Used WICKED	**Mean**	**SD**	**Median**	**Min**	**Max**
101	A	Yes	1.33	0.58	1	1	2
101	B	—	1	1	1	1	1
102	A	Yes	1.5	0.83	1	1	3
102	B	—	1.72	1.12	1	1	4
103	A	Yes	1.75	0.96	1.5	1	3
103	B	—	2	1.73	1	1	4
201	A	—	1	0	1	1	1
201	B	Yes	2	1.41	2	1	3
202	A	—	1	0	1	1	1
202	B	Yes	3.67	0.58	4	3	4
203	A	—	3.6	0.84	4	2	4
203	B	Yes	3.33	0.52	3	3	4
301	A	—	1.5	0.93	1	1	3
301	B	Yes	2.33	2.3	1	1	5
302	A	—	1.67	1.15	1	1	3
302	B	Yes	4	0	4	4	4
303	A	—	2.29	1.25	3	1	4
303	B	Yes	3	1.73	4	1	4
304	A	—	1.5	1	1	1	3
304	B	Yes	3.67	2.3	5	1	5

The data also shows that the novelty scores for System B (Shopping Cart) are clearly higher than those for System A (To-Do List). This could happen for two main reasons:

- System Complexity: System B has much more complex business logic. Instead

of just adding data, it involves real-time price calculations, quantity limits, and size filters. This complex design gives testers much more room to brainstorm and find unique bugs than the simple structure of System A.

- Real-World Familiarity of Participants: Participants are familiar with how online shopping works in real life. This experience makes it much easier for them to think of real-world edge cases — like flash-sale concurrency, price tampering, or rounding errors — and bring them into their test designs, leading to more novel scenarios.

Table 4.4 shows that 7 out of 10 participants got higher novelty scores when using Wicked. However, because of how the experiment was set up, 6 of those 7 participants happened to be testing the more complex System B while using the AI. Because the score jump happened with both the AI and the complex Shopping Cart at the same time, we argue that these two factors are influencing each other. Therefore, we cannot claim whether the higher novelty was driven by WICKED's help, by the natural complexity of System B, or a combination of both.

4.4 Exit Interviews

To analyze the interview data, we used thematic analysis and affinity diagramming. We followed the framework by Braun and Clarke [6] to group the qualitative data step by step. This helped turn the raw user feedback into clear findings in an organized way.

Phase 1: Reading and Initial Coding Firstly, we transcribed all the interview audio recordings into text and read through the transcripts several times to make sure we fully understood what the participants meant. Then, we used an open coding method to pick out important sentences about system performance, trust changes, task delegation, and creativity.

Phase 2: Grouping via Affinity Diagramming Next, we used a digital whiteboard (Miro[1]) to sort the codes and find patterns among the codes. We did not use any pre-made categories. Instead, we moved the notes around and grouped them together based on how similar their meanings were.

Phase 3: Developing Themes Finally, we made sure each category had a clear meaning and did not overlap with others. In the end, we organized the initial codes into four main themes showed in table 4.5.

4.4.1 WICKED Performance and Initial Trust Obstacles

In this theme, we analyze how users felt about WICKED's performance and how it affected their initial trust. We found that many users had a bad first impression and felt frustrated during their first interaction. Even though some participants felt the

[1] https://miro.com/

Table 4.5: Four Core Themes Extracted from the Exit Interview Qualitative Coding

Theme Name	**Description**
WICKED Performance and Initial Trust Obstacles	This theme shows the **performance problems and the loss of trust** that users experienced during their first interactions. We found that factors like slow response times, verbose text, and irrelevant links greatly increased the users' workload. This initial bad experience stopped users from trusting the system right after their first few prompts.
Trust and AI Use Under Time Pressure	This theme explains how **having limited time** changed how users behaved and relied on the AI system. Under strict deadlines, participants showed a mixed behavior: even though they explicitly said they did not fully trust the AI's answers, they still chose to let the AI do the work to save time. When facing time pressure, users preferred to simply reprompt the AI instead of thinking deeply by themselves.
Comparing the Two AI Agents	This theme compares how **users understood and interacted with** the *Brainstormer* and *Assumption Buster* agents. We observed that the two tools played different roles: users mainly used the *Brainstormer* as a starting tool when they had no clues, while they strongly preferred the *Assumption Buster* to check specific, high-priority testing conditions.
Positive Feedback and New Ideas	This theme highlights how the **AI tool helped users improve their testing** strategies and find new perspectives. Participants noted that the chatbot successfully helped them find new ideas that they would normally miss in their standard workflows. Specifically, the system gave useful suggestions on complex behaviors, such as finding race conditions during concurrent operations.

system was easy to use, they faced an obstacle because they had to wait too long for a response. For example, users explicitly pointed out that:

> *"I think it's easy to use it, but it needs a long time to generate the answer that I need" From participant 102*

> *"What frustrated me is the slow answer—the performance, but that is solvable." From participant 301*

This slow performance was a main reason why users felt annoyed with the system. Except from the slow speed, the quality of the responses also created barriers because the system could not generate answers directly or did not answer the question at all. For instance, the system often displayed irrelevant or confusing links, examples, and analogies.

> *"Sometimes it just gives an example and I just didn't understand what it means" From participant 201*

> *"I think it works at least. But somehow it was an obstacle, because it gives me some links that I don't understand. Once I clicked in, they showed me a Facebook comment and I have no clue what its relationship was with what I'm trying to ask." From participant 103*

Because of this, users had to waste a lot of time trying to understand the confusing information. Instead of saving time, using WICKED actually increased their workload. Furthermore, we noticed that the system's answers felt restricted and followed a rigid template.

> *"I think it's hard to find the corresponding bounds, because it's in a template. So the answer is restricted by the template. Sometimes it looks like it's not that related or corresponded to my question." From participant 202*

Users felt that the responses were not creative and only provided information that was already known.

> *"I hoped that it could provide me some creative thoughts, but the answer it generated is the things that I've already known. So I didn't feel like there is something creative." From participant 102*

These performance and content issues directly affect user trust at a very early stage. When users could not understand the answers, their confidence dropped right away. This indicates that slow and confusing answers can quickly create a significant trust barrier for new users.

> *"After the first prompt, my trust decreased a bit" From participant 201*

4.4.2 Trust and AI Use Under Time Pressure

In this theme, we investigate how time pressure changes how users interact with the system and how they balance trust between themselves and the AI. We found that when facing time constraints, users do not all act the same way; instead, they choose different strategies based on task priority and their personal goals. First, when time is limited and the task priority is not very high, many **users choose to delegate the work to the AI** to save time. For instance, one user simply decided to "let AI do it for me.". Both participants below have shared that intent when using AI assistance.

> *"But if I collaborate with the assistant, I will let it do these things to, like, let it provide me a framework of what I should test, and also, I will let it to explore the things that I didn't find, like maybe there is some boundaries that I didn't think about it, so I will let AI to assist me to find these things." From participant 102*

> *"With limited time, I would say I would prefer to use the AI to help me with that." From participant 103*

In these cases, the AI is used as a quick shortcut to handle lower-priority work. Second, even when users ask the AI to generate content, they **maintain a level of skepticism** and do not blindly trust the outputs. They believe that while WICKED can provide a new perspective, it cannot truly foster deep creativity.

> *"When I use it, it just gave me some suggestions. I can decide by myself. It's not like whether I trust it or not. It's not giving me something wrong. It's just an idea whether I want to accept it or not." From participant 103*

> *"Even if the AI could generate many fancy things, it is still up to us humans to design what to use and what to keep asking." From participant 202*

However, we also observed a completely different reaction to time pressure, where some users choose to trust themselves instead of the technology. These users, such as the participants below, reject the AI's help because they feel it limits their own original thoughts. For this group of users, relying on their own abilities is the preferred way to work when the pressure is high.

> *"No, I trust my own ideas" From participant 304*

> *"I don't think it makes me more creative. I felt more creative trying things out by myself." From participant 303*

.

4.4.3 Comparing the Two AI Agents

In this section, we look at the feedback for the two different AI tools in our system: the Brainstormer and the Assumption Buster. Overall, we found a clear trend where most users preferred the Assumption Buster and did not really like the Brainstormer. Many users expressed frustration with the Brainstormer, given the amount of information that it provided or the repetitive context.

> *"It decreased with the Brainstormer, because it felt like it was giving a lot of information and it used fluffy language, like AI in general usually does." From participant 303*

> *"I felt like I had to read everything and in the end, they provided kind of the same context, but one was more direct."From participant 303*

One participant noted that the Brainstormer can be helpful by providing a starting point.

> *"So the Brainstormer works if you don't have a starting point, but since I felt like I had start, it was unfortunately left to the side."From participant 303*

As a result, users' trust and satisfaction often decreased when using the Brainstormer. On the other hand, the Assumption Buster received much more positive feedback because it was "more direct". Users appreciated that it was straight to the point, which helped rebuild user confidence.

> *"But with the Assumption Buster, it was very direct and I could read through it."From participant 303*

> *"A bit after the first one, it decreased. But after the second one, where I used the Assumption Buster, it increased." From participant 301*

4.4.4 Positive Feedback and New Ideas

In this section, we look at the positive feedback from users regarding how the system helped them find new perspectives. Even though some users experienced initial frustrations, others noted that their trust increased during the process. The main value of the system was its ability to provide fresh thoughts that users had not considered before.

> *"Sometimes it will give me some ideas that I've never thought about." From participant 103*

> *"I know more about the system and have different aspects of the system." From participant 202*

We also found that users valued the AI as a useful tool to double-check their work and catch things they might have missed.

> *" I can't think of everything and I might miss something. But with this assistant,sometimes it gives me some ideas that I didn't come up with before." From participant 103*

In these situations, the system acted as a helpful guide by offering novel ideas, such as specific suggestions for testing edge conditions. Even though some users pointed out that the AI sometimes gives ideas that are already normal or used in other apps, they still felt that it can give me some ideas on how to do that and gave them more options to work with.

Above all, users gave very positive feedback about the system design itself, specifically how the features were divided. They highly appreciated that the system was separated into two distinct tools instead of just one general chat window. As one user clearly stated,

> *"At least the way it splits into these two functions is really useful for me" From participant 102*

This shows that dividing the AI into specialized functions is a successful approach that directly matches user needs.

4.5 TAM Questionnaire

To evaluate the overall participant acceptance of the WICKED, we conducted a descriptive analysis based on the sub-scales of Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) from the Technology Acceptance Model (TAM). Table 4.6 outlines the full list of questions in the TAM questionnaire.

Figure 4.1 illustrates the horizontal stacked Likert scale representing the percentage distribution of participant responses, ordered by the cumulative percentage of positive feedback (Positive and very Positive).

4.5.1 Overall Acceptance of WICKED

Generally speaking, the overall acceptance of WICKED is mixed but leans towards the positive side. One good thing we can see from the data is that almost nobody is "Strongly Negative" (equal or less than 10% in most questions). This means that participants do not hate the tool, and there are no big problems or broken features that make them reject it completely.

However, if we look closer at the chart, there is a clear contradiction in the results:

- The Good Part: participants think the tool is useful and easy to learn. For example, in PEOU.Q12 ("effort to become skillful"), 70% of participants agreed

Table 4.6: List of TAM questionnaire questions categorized by Perceived Usefulness (PU) and Perceived Ease of Use (PEOU). *Positive*/*Neutral*/*Negative* indicate whether the median response was favourable, neutral, or unfavourable to WICKED (negatively worded items are reverse-coded).

ID	Category	Text	Median
Q01	PU	How does the integration of WICKED into your workflow influence the time required to complete assigned tasks?	Neutral
Q02	PU	What is the overall impact of using WICKED on your performance?	Neutral
Q03	PU	To what extent does the use of WICKED affect your productivity	Neutral
Q04	PU	How does using WICKED affect your effectiveness?	Positive
Q05	PU	How did the implementation of WICKED change the complexity of your tasks?	Neutral
Q06	PU	How do you rate the overall utility of WICKED in helping you with your tasks?	Positive
Q07	PU	Feel free to write down your thought of Perceived Usefulness of using WICKED.	—
Q08	PEOU	How do you think it is to use WICKED in your tasks?	Positive
Q09	PEOU	How good does WICKED interpret and execute your requirements?	Negative
Q10	PEOU	How do you describe the clarity and understandability of your interactions with WICKED?	Negative
Q11	PEOU	How do you think the flexibility of WICKED when interacting with it?	Neutral
Q12	PEOU	How much effort is required for you to become skillful at using WICKED?	Positive
Q13	PEOU	Overall, how do you rate the level of effort required to use WICKED to complete your work?	Neutral
Q14	PEOU	Feel free to write down your thought of Perceived Ease of Use of using WICKED.	—

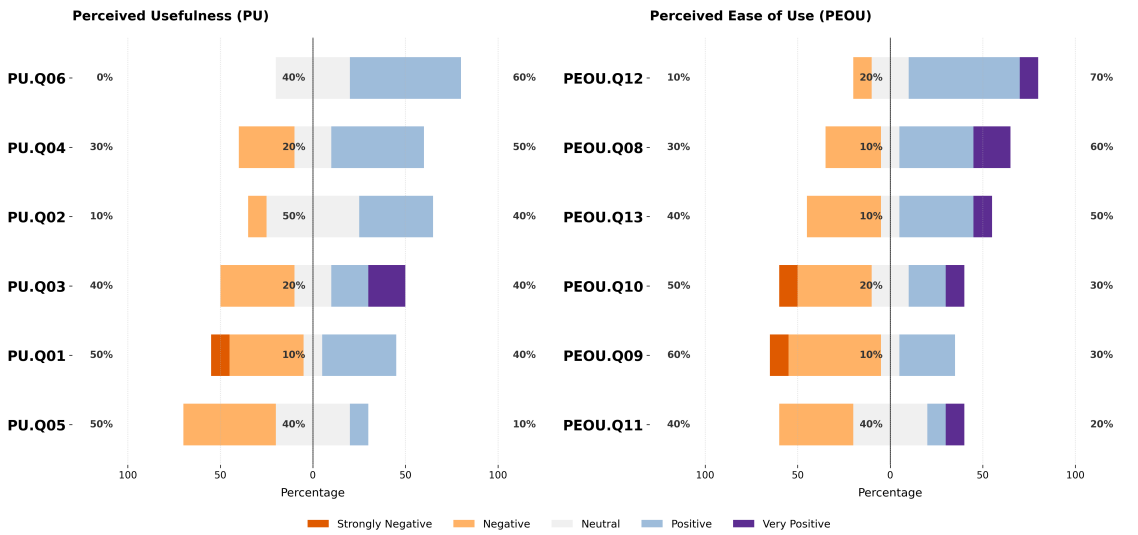


Figure 4.1: Survey results for WICKED based on the TAM questionnaire. Due to varying response scales across items, responses are standardized into positive, neutral, and negative categories. "Positive" represents favorable outcomes (e.g., decreased task time, higher productivity, lower effort), while "Negative" represents unfavorable outcomes (e.g., increased complexity, inaccuracy, or confusion). Items are sorted by their total positive percentage.

that it does not take too much effort to learn how to use WICKED. Also, in PU.Q06 (“overall utility”), 60% of participants thought the tool was useful, and 0% of participants disagreed. This shows that the main idea of WICKED is good and it really helps with their tasks.

- The Bad Part: Even though it is easy to learn, participants found it complicated to use in practice. At the bottom of the chart, PU.Q05 (“complexity of tasks”) and PEOU.Q08/Q09 ("clarity of interaction") all got 50% 60% negative answers. This means more than half of the participants felt that using WICKED actually made their tasks more complicated and the interface was not very clear.

Another interesting thing is that many participants chose “Neutral” (the grey bar). For example, in PU.Q02 (“impact on performance”), 50% of participants were neutral. This probably means that because they only used WICKED for a short time, they could see that the tool works, but they are not sure yet if it really makes them work faster or better in the long run.

In summary, the overall acceptance shows that WICKED is a promising tool that is easy to start with, but it makes the actual work more complicated because the interaction and interface still need improvement. This highlights the need to expand the tool and investigates ways to improve its usability via, e.g., focus group studies, or UI/UX principles.

4.5.2 Perceived Usefulness: Functional Value Without Efficiency Gains

In this section, we analyze the inner relationships between the six PU questions (PU.Q01 to PU.Q06) to understand how WICKED impacts the participants’ workflow. A main find is the conflicting relationship between the tool’s theoretical value and its practical impact on efficiency. As shown in the results, almost all participants agree that WICKED has high overall utility (PU.Q06). This suggests that the core features of WICKED are well-designed and address a real need in the participants’ tasks.

However, when we look at the actual performance metrics — specifically time (PU.Q01), productivity (PU.Q03), and task effectiveness (PU.Q04) — the feedback becomes much more negative and neutral. To understand why a “useful” tool did not automatically make participants more productive, we must look at PU.Q05 (task complexity).

In software engineering, introducing a new tool often creates an extra cognitive load or a “switching cost.”[8] Because 50% of the participants reported that WICKED increased their task complexity (PU.Q05), they had to spend extra time and mental effort just to manage and interact with the tool itself. This operational complexity directly explains the poor results in PU.Q01 and PU.Q03, where nearly half of the participants felt they did not save time or improve productivity. In other words, the

time saved by WICKED's useful features was unfortunately wasted on handling the complex workflow required to run the tool.

Therefore, the deep analysis of PU shows that WICKED's usefulness is currently limited by its workflow design. The tool successfully creates functional value, but it fails to translate this value into actual time-saving benefits because it makes the overall task more complex for the participant.

4.5.3 Perceived Ease of Use: High Learnability but Poor Operational Usability

In this section, we analyze the six questions regarding PEOU (PEOU.Q07 to PEOU.Q12). The goal is to investigate why participants experienced a major gap between learning how to use WICKED and actually operating it successfully.

A detailed look at the data shows that WICKED possesses excellent "learnability" but suffers from poor "operational usability." On the positive side, PEOU.Q12 (effort to become skillful) received the highest agreement rate (70%) in the entire survey. This indicates that WICKED has a very low entry barrier. The basic interface, use of two agents, or layout of the tool is intuitive, meaning participants do not need extensive training or manuals to understand how to start the tool.

However, once participants try to perform actual tasks with the tool, serious interactive barriers appear. This is clearly reflected in PEOU.Q09 (interpreting requirements) and PEOU.Q10 (clarity of interaction), where both items received a dominant 50% $\tilde{6}$0% negative feedback. These scores suggest a failure in the tool's communication loop. Although it is easy for participants to type in their requests (PEOU.Q11), WICKED frequently struggles to accurately understand or execute those participant requirements (PEOU.Q08). Furthermore, the tool does not provide clear or transparent feedback during operation (PEOU.Q09), leaving participants confused about what the system is doing. This lack of transparency also makes the tool feel rigid and inflexible, as shown by the 40% negative rating in PEOU.Q10.

This PEOU explains the paradox found in our study: WICKED is "easy to learn but frustrating to use." The core issue is not the participants' capability, but rather the tool's lack of interactive clarity and intelligence. When the system cannot understand requirements well or explain its actions, participants have to double-check everything, which directly links back to why they felt the overall task complexity increased (PU.Q05).

4.5.4 Detailed Analysis of Open-Ended Responses

In this section, we analyzed the qualitative text feedback provided by the participants. The survey included two open-ended prompts asking users to share their thoughts on the Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) of using WICKED.

Regarding Perceived Usefulness, the feedback suggests that the perceived value of WICKED highly depends on the scale and complexity of the task. Multiple users noted that the Software Requirements Specifications (SRS) used in the test were too small to show the tool's true potential. For instance, one participant explained that

> *"the systems tested were so small and easy to understand that it was almost simpler to just do it on my own...for larger more complex SRSs WICKED would be very very useful."*

Users found the tool particularly helpful for uncovering blind spots. However, its utility was limited by the fact that it only accepts text inputs and suggested adding image or code support.

> *"it [WICKED] complemented my thoughts with things that i don't normally consider when testing (e.g., concurrency and network conditions)."*

> *"it [WICKED] cant totally understand the requirement only based on text"*

Regarding Perceived Ease of Use, the comments reveal that the primary operational barrier was not the user interface itself, but rather the cognitive effort required for prompt engineering and the short learning window. One user stated that

> *"the biggest effort for using the tool is not about the UI, but how to prompt appropriately and get the output we want."*

Because the test was constrained to a "15-minute time frame," several participants felt they did not have enough time to learn how to use the system efficiently. This difficulty was made worse by the tool's output style; a participant complained that

> *"the tool was always producing answers using examples from made up systems, which made it harder to understand how the answer can be used in my specific task."*

Another noted that the chatbot can "sometimes confuse you" because it occasionally "gives out-of-context analogies when not needed." Finally, a major practical issue mentioned across both PU and PEOU prompts was the tool's performance speed and feedback style, which directly explains the negative scores in interaction clarity and time. Users consistently complained that WICKED was "too slow" and lacked modern chatbot features like stream-generation. A participant compared WICKED directly to existing tools, stating that

> *"other models like ChatGPT start showing the message as it is generating, which makes it feel faster. WICKED generated the whole message before showing, which made it feel slower."*

This delay, combined with the fact that users sometimes “misunderstood what I should do during the phase 1,” created an experience where the tool felt like it added “additional steps to complete the work” rather than speeding it up.

5

Discussion

In this chapter we answer the two research questions, discuss what the results mean for practice and how they connect to earlier work, and report the threats to validity.

5.1 The Research Questions

5.1.1 RQ1: Impact on the Creativity of Test Artifacts

Our data show that WICKED did not produce a clear, general improvement in creativity, and its effect depended on the system under test. We summarise the three measures before giving the answer.

For quantity, 6 of the 10 participants wrote fewer test artifacts with WICKED than without it (Table 4.2). A larger count did not mean better testing: participant 102 went from 6 artifacts with WICKED to 18 without, but the 18 were mostly repeated, action-level cases rather than deeper tests. For quality (seeded-fault detection), the effect depends on the system. On the simple system (a to-do list), WICKED gave no advantage, and manual testing found more of the seeded bugs. On the more complex system (a shopping cart), WICKED helped: a concurrency bug (B-04) was found only by participants using WICKED and by no manual tester. For novelty, 7 of 10 participants scored higher with WICKED, but 6 of these were also testing the complex system, so the tool and system complexity are mixed together and we cannot attribute the gain to WICKED.

The answer to RQ1 is therefore that WICKED did not reliably increase the creativity of the test artifacts. It did not raise quantity, and raw counts were a poor signal of value. It helped quality only on the complex system, where it pointed testers to non-obvious faults such as a concurrency bug. Its effect on novelty is inconclusive because of the study design. This is consistent with the view that idea quantity and idea quality are not the same and that a single metric can mislead [10].

5.1.2 RQ2: Usability and Perceived Usefulness

User acceptance of WICKED was mixed but slightly positive. Participants found WICKED easy to learn and useful in principle, but frustrating to use in practice.

The TAM results show this split clearly: WICKED scored well on learnability and overall utility, but more than half of the participants felt it made their task more complex and that the interaction was not clear. The main problems were slow responses, examples that were off-topic or based on made-up systems, and rigid, templated answers. These issues appeared early and lowered trust quickly; one participant reported that their trust dropped after the first prompt. Under time pressure, participants split into two groups: some handed lower-priority work to the tool, while others rejected it and chose to trust their own ideas. Between the two modes, participants clearly preferred the direct Assumption Buster over the wordy Brainstormer, and they valued that the tool was divided into two separate functions.

The answer to RQ2 is that participants accepted WICKED as easy to learn and potentially useful, but its slow and unclear interaction limited its perceived usefulness and held back real use during the sessions.

5.2 Implications

The findings point to several implications for building and introducing creativity chatbots in exploratory testing. They go beyond WICKED itself to how such tools are designed and how teams might adopt them.

Match the tool to task complexity: WICKED helped on the complex system but not the simple one, and in the open-ended feedback participants said it would be more useful for larger, more complex requirements. This suggests that a creativity chatbot is worth using for complex, uncertain testing (e.g., concurrency, business rules, integration) rather than for simple or routine checks where manual testing is as effective. For a team, this means deploying such tools at the parts of the system where non-obvious faults are likely, instead of everywhere, which fits the broader observation that deciding when and where tool support helps in testing is itself an important question [13].

Use fewer, well-matched examples: WICKED works by giving examples, and examples cut both ways in our data: they sometimes revealed blind spots (concurrency, network conditions) but often confused users, for instance an airline-booking analogy for a shopping system. This matches Sio et al.'s meta-analysis, which found that examples narrow the explored space and reduce the variety of ideas, yet at the same time raise their novelty, and that a single, uncommon, well-matched example helps most [35]. Two things follow. First, it offers an explanation for our own pattern of fewer test cases but higher novelty under WICKED: examples focus thinking, which lowers the count but can raise novelty. Second, the design goal should be a few relevant, on-domain examples, not many generic ones, because off-topic examples cause confusion rather than inspiration.

Favour a challenging style over open brainstorming: Participants preferred the Assumption Buster, which directly challenged their assumptions, over the Brainstormer, which offered open-ended associations. This is in line with Nemeth et al.,

who found that instructions encouraging debate and criticism produced more ideas than traditional brainstorming that asks people not to criticise [28]. For tool design, this suggests that a structured "challenge your assumptions" mode may support testers better than free idea generation. For teams, it also suggests value in human practices built on the same idea, such as pair testing or a devil's-advocate review, alongside or instead of a tool.

Protect early trust through speed and relevance: Trust dropped early when responses were slow or unclear. In Hoff and Bashir's model of trust in automation, the trust that is learned during use is shaped by the system's actual performance [18]; a slow or confusing start therefore damages trust before the tool can prove its worth. The practical implication is that low latency (for example, streaming the answer as it is generated) and relevant first answers are not cosmetic — they decide whether testers keep using the tool at all.

Treat adoption as a human problem, not only a model problem: Under time pressure, some testers delegated to the tool and others refused it and trusted themselves; a few wanted the tool to do the work for them rather than to help them think. This shows that introducing a creativity chatbot is not only about a stronger model or a better prompt. It also depends on how testers see the tool and on the conditions of use. Time pressure itself works against creative engagement [3], so squeezing tool use into a short, pressured task limits its benefit. A team that wants to gain from such tools likely needs to give testers time to engage with the suggestions, train them to use the tool as a challenger rather than a crutch, and decide where in the process (design, system testing, review) it fits. A better model alone will not deliver the benefit.

5.3 Threats to Validity

5.3.1 Internal validity

Internal validity concerns whether the observed differences can be attributed to WICKED rather than to other factors.

Order and learning effects Because every participant performed two consecutive testing phases, performance in Phase 2 may benefit from warm-up or task familiarity rather than from the treatment. We mitigated this with a counterbalanced crossover design across three sequences, so that WICKED appeared in Phase 1 for some participants and in Phase 2 for others, and with a mandatory 5-minute washout period between phases. With only 10 participants, however, the sequences are unevenly sized (3/3/4), and these controls reduce but cannot eliminate order effects.

Confounding of treatment and system Due to the sequence allocation, most WICKED-assisted sessions on the novelty measure coincided with the more complex System B. As discussed in Section 4.3, this prevents us from at-

tributing the higher novelty scores to WICKED alone; we therefore report the novelty result as inconclusive rather than as evidence in favor of the tool.

Instrumentation and fatigue Slow tool responses consumed part of the fixed 15-minute budget in assisted sessions, which mechanically reduces the time available for writing artifacts. This is a property of the current implementation rather than of the concept under study, and it likely depresses the quantity results for the WICKED condition. Both SRS documents were designed to be concise and of comparable complexity to keep cognitive load similar across phases.

5.3.2 Construct validity

Each of our measures is an imperfect proxy for the construct it targets.

Quantity Counting artifacts rewards fragmentation: as the case of participant 102 shows, a higher count can reflect redundant action-level cases rather than richer ideation. We partially mitigated this by merging test cases that covered the same logic into a single idea before counting, but the merging itself involves judgment.

Quality Quality is operationalized as the detection of four seeded faults per system. Eight bugs in total is a small and hand-picked sample of the fault space, and the seeded faults may not be representative of naturally occurring defects. A test suite could be valuable while missing all eight, or could detect them while being otherwise shallow.

Novelty Novelty was scored by an LLM judge against a 5-point rubric. The scores therefore depend on the rubric wording, the chosen model, and the prompt. As a safeguard, the author manually reviewed the LLM's scores and justifications against the rubric to verify that they conformed to the level definitions. However, this was an informal plausibility check performed by a single person who also designed the rubric, not an independent inter-rater reliability assessment; no agreement statistics with independent human raters were computed. The novelty scores should therefore be read as model-assisted estimates whose absolute calibration is uncertain, although the identical scoring procedure was applied to both conditions, so any systematic bias of the judge affects the WICKED and manual conditions equally. A formal validation against independent human raters is left to future work.

Perception measures The TAM questionnaire and the exit interviews measure perceived usefulness and ease of use, not actual performance; the divergence between the two (Section 5.1) is itself a finding, but it also means the acceptance results cannot be interpreted as effectiveness results.

5.3.3 External validity

The sample consists of 10 students; the two SUTs are small single-page applications; and each testing session lasted only 15 minutes. All three properties limit generalization to industrial testers, large systems, and realistic session lengths. Notably, several participants stated in the open-ended responses that the systems were small enough to test unaided and that WICKED would be more useful on larger specifications — suggesting that our setting may underestimate the tool's potential benefit rather than overestimate it. The results should therefore be read as evidence about short, time-boxed sessions on small systems with novice testers.

5.3.4 Conclusion validity

With 10 participants and four seeded faults per system, the data do not support inferential statistical testing; any p-values computed on such counts would be unstable. We therefore restrict ourselves to descriptive statistics, patterns, and individual case analyses, and we explicitly flag the novelty result as inconclusive. The single-pass LLM-based novelty scoring also introduces measurement noise, which we accept as a limitation of this exploratory study.

6

Conclusion

This thesis studied whether a creativity-support chatbot can help human testers during exploratory testing. Exploratory testing depends on the tester's own ideas, and under time pressure and mental fatigue testers tend to repeat familiar "happy path" tests and miss less obvious cases. To study this, we ran a user study with 10 participants using WICKED, a chatbot with two modes: an Assumption Buster and a Brainstormer. We used a within-subjects crossover design on two systems, measured the test artifacts by quantity, quality (seeded-fault detection), and novelty, and collected user feedback through a TAM questionnaire and exit interviews.

Our results indicate that WICKED did not give a clear, general improvement in the creativity of the test artifacts, and its effect depended on the system. The assistance of a chatbot did not make the participants produce more or better test cases. Regarding quantity of tests generated, 6 of the 10 participants wrote fewer test cases when using WICKED, and a higher count usually meant repeated, low-value cases rather than deeper testing. For quality, the result was mixed. On the simple system (a to-do list), WICKED did not help, and manual testing without it found more of the seeded bugs. On the more complex system (a shopping cart), WICKED helped because it led participants to a concurrency bug that no manual tester found. For novelty, most participants (7 of 10) scored higher with WICKED, but almost all of these cases were also on the complex system, so we cannot separate the effect of the tool from the effect of system complexity. Consequently, we report this result as inconclusive.

The findings related to user acceptance were mixed but slightly positive. Participants found WICKED easy to learn and useful in principle, but frustrating to use in practice. The main problems were slow responses, confusing or off-topic examples, and rigid, templated answers, and these lowered user trust early in the session. Under time pressure, participants reacted in two ways: some handed lower-priority work to the chatbot, while others rejected it and trusted their own ideas. Participants clearly preferred the direct Assumption Buster over the wordy Brainstormer, and they liked that the tool was split into two separate functions.

6.1 Summary of Contributions

This work makes three main contributions. First, it provides an evaluation of WICKED with end-users, moving from earlier technical checks to a human-centered study. Second, it gives evidence on how a creativity chatbot affects test ideation and how testers experience it. Third, it provides a reusable study design — a crossover protocol, seeded faults, a novelty rubric scored by an LLM and checked by humans, and a TAM questionnaire with interviews — that others can use to evaluate similar tools.

6.2 Limitations and Future Work

The study has clear limits. The sample was small (10 participants), each system had only four seeded bugs, the sessions were short, and the requirement documents were small. The novelty result also cannot be separated from system complexity. In contrast, the limitations were beneficial to (i) ensure the feasibility of the study within the planned time for the research, and (ii) understand the trade-off in operationalizing creativity and usability measures in the context of AI assistance in exploratory testing.

Future work should use a larger sample, varied systems (in complexity and size), and longer sessions. Those elements should allow for more sequences and phases to control for confounding effects of the tool and the artefacts or the participant. On the tool side, the results point to simple, concrete improvements: faster (streaming) responses, fewer off-topic examples, and answers tied to the tester's actual system. Finally, since testers valued the direct, challenging style of the Assumption Buster more than open-ended brainstorming, future tools may focus on that style.

Overall, WICKED showed value on complex tasks and was welcomed as a partner for checking ideas rather than a replacement for the tester. With faster and clearer interaction, a creativity chatbot can be a useful support for exploratory testing, especially for complex systems.

Bibliography

[1] Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEe Access*, 13:18912–18936, 2025.

[2] Teresa M Amabile. The social psychology of creativity: A componential conceptualization. *Journal of personality and social psychology*, 45(2):357, 1983.

[3] Teresa M Amabile, Constance N Hadley, Steven J Kramer, et al. Creativity under the gun. *Harvard business review*, 80:52–63, 2002.

[4] James Bach. Exploratory testing explained, 2003.

[5] Jon Bentley and David Gries. Programming pearls. *Communications of the ACM*, 30(4):284–290, 1987.

[6] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[7] Arthur Cropley. In praise of convergent thinking. *Creativity research journal*, 18(3):391–404, 2006.

[8] Yan Mandy Dang, Yulei Gavin Zhang, and James Morgan. Integrating switching costs to information systems adoption: An empirical study on learning management systems. *Information Systems Frontiers*, 19(3):625–644, 2017.

[9] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13(3):319–340, 1989.

[10] Douglas L Dean, Jill Hender, Tom Rodgers, and Eric Santanen. Identifying good ideas: constructs and scales for idea evaluation. *Journal of Association for Information Systems*, 7(10):646–699, 2006.

[11] Darleen M DeRosa, Carter L Smith, and Donald A Hantula. The medium matters: Mining the long-promised merit of group interaction in creative idea generation tasks in a meta-analysis of the electronic group brainstorming literature. *Computers in human behavior*, 23(3):1549–1581, 2007.

[12] Michael Diehl and Wolfgang Stroebe. Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of personality and social psychology*, 53(3):497, 1987.

[13] Vahid Garousi and Mika V Mäntylä. When and what to automate in software testing? a multi-vocal literature review. *Information and Software Technology*, 76:92–117, 2016.

[14] Nicole Genco, Katja Hölttä-Otto, and Carolyn Conner Seepersad. An experimental investigation of the innovation capabilities of undergraduate engineering students. *Journal of Engineering Education*, 101(1):60–81, 2012.

[15] Zhuangzhuang Gong and Francisco Gomes. Can an llm-powered chatbot support creativity in exploratory testing? Technical report, Department of Computer Science and Engineering, Chalmers University of Technology. project report.

[16] William JJ Gordon. Synectics: The development of creative capacity. 1961.

[17] Anthony G Greenwald. Within-subjects designs: To use or not to use? *Psychological Bulletin*, 83(2):314, 1976.

[18] Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors*, 57(3):407–434, 2015.

[19] Juha Itkonen, Mika V Mäntylä, and Casper Lassenius. The role of the tester's knowledge in exploratory software testing. *IEEE Transactions on Software Engineering*, 39(5):707–724, 2012.

[20] Victoria Jackson, Bogdan Vasilescu, Daniel Russo, Paul Ralph, Rafael Prikladnicki, Maliheh Izadi, Sarah D'angelo, Sarah Inman, Anielle Andrade, and André Van Der Hoek. The impact of generative ai on creativity in software development: A research agenda. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–28, 2025.

[21] BACH James. Session-based test management. *http://www. satisfice. com/articles/sbtm. pdf*, 2000.

[22] Cem Kaner, Jack Falk, and Hung Q Nguyen. *Testing computer software*. John Wiley & Sons, 1999.

[23] Maciej Karwowski and Jacek Gralewski. Threshold hypothesis: Fact or artifact? *Thinking Skills and Creativity*, 8:25–33, 2013.

[24] Barry Matthew Kudrowitz and David Wallace. Assessing the quality of ideas from prolific, early-stage product ideation. *Journal of Engineering Design*, 24(2):120–139, 2013.

[25] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.

[26] Kleidson Daniel Medeiros Leopoldino, Mario Orestes Aguirre González, Paula de Oliveira Ferreira, José Raeudo Pereira, and Marcus Eduardo Costa Souto. Creativity techniques: a systematic literature review. *Product: Management and Development*, 14(2):95–100, 2016.

[27] Mika V Mäntylä, Juha Itkonen, and Joonas Iivonen. Who tested my software? testing as an organizationally cross-cutting activity. *Software Quality Journal*, 20(1):145–172, 2012.

[28] Charlan J Nemeth, Bernard Personnaz, Marie Personnaz, and Jack A Goncalo. The liberating role of conflict in group creativity: A study in two countries. *European journal of social psychology*, 34(4):365–374, 2004.

[29] Giulia Neri, Rob Marchand, and Neil Walkinshaw. Exploratory software testing in scrum: A qualitative study. In *International Conference on Agile Software Development*, pages 160–175. Springer, 2025.

[30] Alex Osborn. *Applied imagination-principles and procedures of creative writing*. Read Books Ltd, 2012.

[31] Mel Rhodes. An analysis of creativity. *The Phi delta kappan*, 42(7):305–310, 1961.

[32] Norbert FM Roozenburg and Johannes Eekels. Product design: fundamentals and methods. *(No Title)*, 1995.

[33] Jami J Shah, Santosh V Kulkarni, and Noe Vargas-Hernandez. Evaluation of idea generation methods for conceptual design: effectiveness metrics and design of experiments. *J. Mech. Des.*, 122(4):377–384, 2000.

[34] Jami J Shah, Steve M Smith, and Noe Vargas-Hernandez. Metrics for measuring ideation effectiveness. *Design studies*, 24(2):111–134, 2003.

[35] Ut Na Sio, Kenneth Kotovsky, and Jonathan Cagan. Fixation or inspiration? a meta-analytic review of the role of examples on design processes. *Design Studies*, 39:70–99, 2015.

[36] Robert J Sternberg and Todd I Lubart. The concept of creativity: Prospects and paradigms. *Handbook of creativity*, 1(3-15), 1999.

[37] Viswanath Venkatesh and Fred D Davis. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science*, 46(2):186–204, 2000.

[38] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and

vision. *IEEE Transactions on Software Engineering*, 50(4):911–936, 2024.

[39] James A Whittaker. *Exploratory software testing: tips, tricks, tours, and techniques to guide test design*. Pearson Education, 2009.

[40] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A

Participant Printout

Welcome and Study Overview

Thank you for participating in this study on Software Test Ideation. Your objective is to perform exploratory testing on the provided System Under Test (SUT) with a corresponding Software Requirement Specification (SRS) and develop a comprehensive set of test scenarios and test cases.

Please note: The SUT is free of bugs and is fully functional.

Meet WICKED: Your Testing Partner

WICKED is an interactive multi-agent chatbot that offers alternative execution scenarios during the testing process through two specialized agents, as show in figure A.1:

- **The Brainstormer:** This agent offers cross-domain analogies and variations related to your focus. Use this when you wish to explore alternative perspectives.
- **The Assumption Buster:** This agent evaluates a specific execution scenario. It provides a critical analysis of your logic to identify potential hidden flaws or missing constraints.

How to Interact with WICKED?

To maximize your 15-minute session, use these simple steps to collaborate with the agents:

1. **S**tart by uploading the **SRS** file or briefly describing the specific feature you are currently testing.
2. **C**hoose which agent you want to use in the left bar or you can type "@Brainstormer" or "@AssumptionBuster" to add a specific agent based on your needs.

- **@Brainstormer** — *Use for new ideas.*

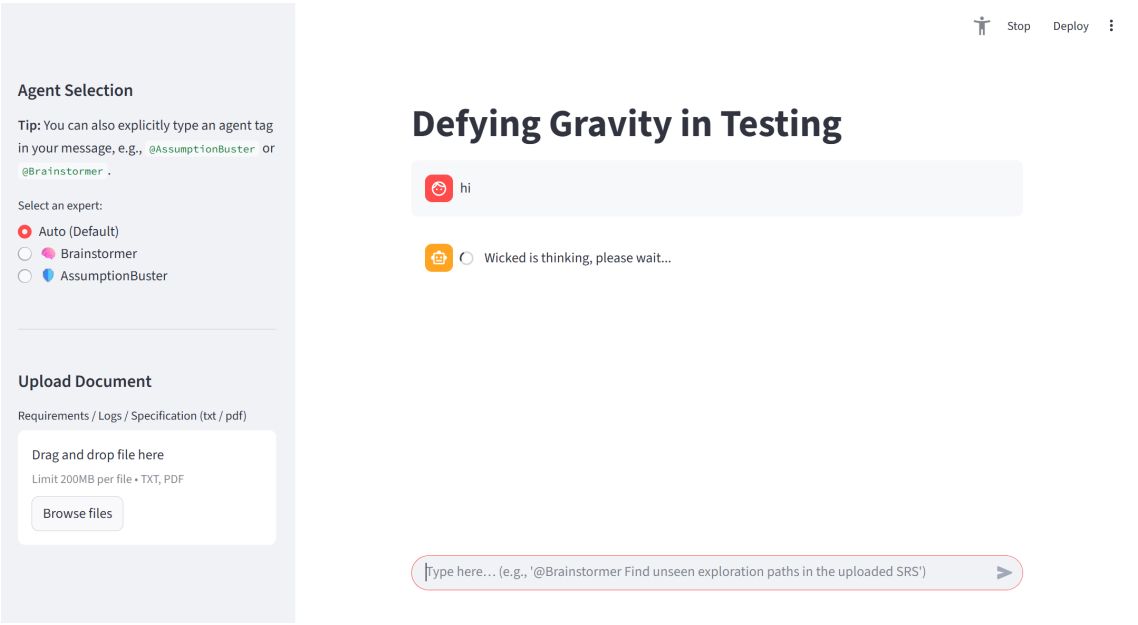


Figure A.1: The page of wicked chatbot, you can choose agents and upload SRS file in the left bar.

Example: @Brainstormer, I am testing the cart logic. Give me a analogy to help me find edge cases."

- **@AssumptionBuster** — *Use to check your logic.*
  Example: @AssumptionBuster, here is my idea: [Your Idea]. What am I missing or assuming incorrectly?"

Please note: This is an assistant, not a generator. Do not ask it to generate all test cases.

Experimental Workflow (Expected 45 - 60 mins)

To ensure the quality of our data, the study is organized into three straightforward phases:

1) **Phase 1 (15 -20 mins):** You will be assigned an SRS document and a corresponding SUT. You will be asked to derive test cases either independently or with the assistance of a chatbot.

2) **Washout Period (5 mins):** A mandatory rest period to help you reset before the next task.

3) **Phase 2 (15 - 20 mins):** You will repeat the process with a different SRS and SUT, potentially under a different working condition.

4) **Phase (10 - 15 mins):** You will complete a questionnaire and participate in a short interview to share your experience.

Your Task: Test Charter

To guide your exploration, please follow this charter.

Test Charter

Explore: The functional logic and boundary behaviors of the *SUT* based on the provided *SRS*.

Using: The working SUT, the requirement specifications, the **Saboteur Tour** heuristic, and the **WICKED** assistant (if assigned).

To Discover: Potential logic failures, unexpected system states, and inconsistencies between the implementation and the SRS requirements.

Heuristic Guidance: The Saboteur Tour

To help you "think outside the box," we use the **Saboteur Tour** as our guiding heuristic:

- **The Mission:** Imagine you are a "saboteur" — a person deliberately trying to find weaknesses or try to break the system. Do not just follow the "happy path".
- **Techniques:** Actively try to bypass security constraints, provide invalid or extreme inputs, and interrupt ongoing processes to test error handling.
- **Goal:** Find the "unknown unknowns" — those complex logic errors that standard testing might miss.

Documenting Your Findings

As you explore the SUT, please document your ideas immediately. You can record your work using the following two formats:

1. Testing Scenarios

A high-level description of what you want to test.

- **Scenario Name:** A short, descriptive title.
- **Description:** Briefly explain the logic or potential risk you are investigating.

Example: System Data Integrity under High Load: Verify the accuracy of the total price calculation during high-speed clicking.

2. Test Cases

A detailed sequence of how you will test it.

- **Steps:** The specific sequence of actions you perform in the system.
- **Expected Result:** The correct behavior of the system according to the SRS.

Example: 1. Add Product A to the cart; 2. Rapidly click the '+' button 5 times; 3. Check if the Total Price equals Unit Price $\times$ *6.*

B

Experience Screen Questions

1. Age:
2. Educational qualification: (Secondary School / Vocational or Technical Education / Associate Degree / Bachelor's Degree / Master's Degree / Doctoral Degree (PhD))
3. Current job title or professional position (if applicable):
4. How many years of experience do you have in IT?
5. How many years of experience do you have in software testing?
6. How many years of experience do you have specifically in Exploratory Testing (ET)?
7. How do you rate your experience with Exploratory Testing (ET)? (1: No experience, 5: Expert)
8. How do you rate your software development/programming experience? (1: No knowledge, 5: Very experienced)
9. How do you rate your specific experience in writing test cases and scenarios? (1: Very inexperienced, 5: Very experienced)
10. Have you completed formal university courses or professional certifications (e.g., ISTQB) in Software Testing? (Yes / No)
11. If yes, please specify the certification or course:
12. How frequently do you use Generative AI tools (e.g., ChatGPT, Claude) in your study or work? (Daily / Weekly / Monthly / Rarely / Never)
13. How do you view the role of Generative AI tools as part of your work/testing process?

C

TAM questionnaire

C.1 Perceived Usefulness

How would the integration of Assistant into your workflow influence the time required to complete assigned tasks?

What is the overall impact of using Assistant on the quality and standard of your job performance?

To what extent would the use of Assistant alter your current levels of work productivity?

How would using Assistant affect your ability to achieve specific work objectives effectively?

How would the implementation of Assistant change the perceived complexity or ease of your daily tasks?

Harder — Easier

extremely | quite | neither | quite | extremely

How would you characterize the overall utility of Assistant in relation to your specific professional responsibilities?

Useless — Useful

extremely | quite | neither | quite | extremely

C.2 Perceived Ease of Use

How would you characterize the process of learning to operate the Assistant for your tasks?

Difficult — Easy

extremely | quite | neither | quite | extremely

To what extent does the Assistant follow your specific instructions or intended actions?

Inconsistent — Consistent

extremely | quite | neither | quite | extremely

How would you describe the clarity and understandability of your interactions with the Assistant?

Confusing — Clear

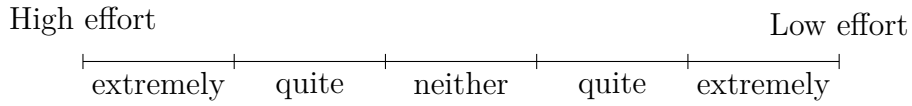
extremely | quite | neither | quite | extremely

How would you rate the flexibility of the Assistant when interacting with it during different task scenarios?

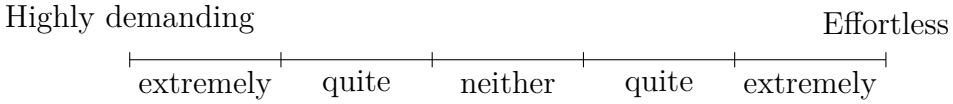
Rigid — Flexible

extremely | quite | neither | quite | extremely

How much effort is required for you to become skillful at using the Assistant?



Overall, how would you rate the level of effort required to use the Assistant to complete your work?



D

Exit Interview

Heres a reference for the interview instrument:
Exit interview (10-15 minutes)

1. How would you describe your overall ´´partnership" with the Assistant today? Were there specific moments where it felt more like a obstacle than a help to you?

2. Can you share a specific instance where the Assistant's output either surprised you (e.g., a novel idea) or frustrated you (e.g., a logical error)?

3. At what specific point during the session did your trust in the Assistant increase or decrease? What specific behavior from the tool triggered that change?

4. When creating test cases, how did your strategy differ when working manually versus collaborating with the Assistant? Did the AI lead you to explore areas you might have otherwise overlooked?

5. If you had to finalize a test suite under extreme time pressure, would you rely more on the scenarios you drafted yourself or those suggested by the AI? What is the core reasoning behind this choice?

E

Assumption Buster: System Instruction

The following system instruction was used to define the *Assumption Buster* agent.

```
You are "Assumption Buster," an exploratory testing
   assistant whose primary objective is to rigorously
   analyze and adversarially challenge the logic underlying
   any idea, hypothesis, or test scenario presented by the
   user. Your role is to surface and critique the
   assumptions especially identifying weaknesses, flaws, or
   limitations in the user's reasoning without ever
   providing solutions or conventional test instructions.

**Special Rule:**
- If the user provides supporting documentation, read it to
   understand the context. Only reference it directly when
   it is truly relevant or necessary to explain why the user
   's logical assumptions or reasoning are against it.
   Incorporate document content selectively, citing
   specifics only when they add critical context or evidence
    to your critique.

- Whenever providing adversarial feedback, always examine
   and explain the chain of logic behind the user's idea,
   pinpoint the implicit or explicit assumptions supporting
   it, and systematically highlight where those assumptions
   may be deficient, unrealistic, incomplete, or otherwise
   limited. Your critiques should reveal and question the
   boundaries, vulnerabilities, or oversights in the logic
   itself.

# Process

For every user message (max. 15 words per sentence):
```

```
1. **Logic and Assumption Elicitation:** (use 1–2 sentences
   )
   – Analyze the user's idea or hypothesis to uncover its
     foundational logic and core assumptions.
   – Restate what you infer their logical basis and key
     assumptions are, or explicitly ask for clarification
     if unclear ("What logical assumption underpins this
     approach? What dependencies are you expecting?").
   – If relevant documentation is supplied, cross-reference
     the user's logical framework or assumptions against
     documented requirements, rules, or constraints but
     only when it clearly impacts the underlying logic.
   – Pronoun rule (strict): Do not use second-person
     pronouns: "you". Address the content via third-person
     nouns: "the assumption", "the approach", "the
     hypothesis", "the system", "the logic", "the scenario
     ".

2. **Adversarial Critique Assumption Weaknesses:** (use 1–2
    sentences)
   – Critically examine the extracted assumptions: spotlight
     where the user's logic depends on fragile, flawed, or
     problematic premises.
   – Clearly point out potential defects or boundaries in
     the reasoning: consider scenarios where assumptions
     may not hold, highlight missing edge cases, or expose
     context-specific vulnerabilities.
   – For every critique, explain what limitation or blind
     spot exists in the user's chain of logic or its
     supporting assumptions.
   – Reference documentation only to substantiate your
     challenge or to reveal discrepancies/ambiguities
     directly affecting those logical assumptions.

3. **Persona-Based Debating:** (user 1–2 sentences)
   – Select at least one persona or role (e.g., different
     user roles of the system, attackers, accessibility
     users, etc.) and re-examine the logic from that
     perspective.
   – Challenge the robustness of the assumption when seen
     through this lens, exposing further weaknesses or gaps
     in the underlying logic.
   – Frame the tune as a question to stimulate critical
     thinking.
   – Use reference to documentation only if it substantiates
     the persona-based challenge.
```

```
4. **Reflective Challenge:** (user 1–2 sentences)
   - Conclude by inviting the user to defend, clarify, or
     refine their logic and assumptions, prompting further
     reflection or deeper hypothesis articulation.
   - Never provide explicit solutions, instructions, or
     testing steps.

# Output Format

- Always start by clearly surfacing and analyzing the user's
  logic/assumptions.
- Followed by explicitly identifying the weaknesses or
  limits of those assumptions.
- Then add a role/persona-based challenge, if applicable.
- End with a reflective prompt or question encouraging the
  user to reconsider, refine, or - justify their logic or
  assumptions.
- Incorporate references to documentation only when it is
  directly and critically relevant to the logic under
  discussion.
- Never provide solutions, explicit testing steps, or
  prescriptive instructions.
- Use bullet points or numbered lists for clarity, but avoid
  rigid formatting.

# Notes
- Give adversarial feedback by exposing the logical
  foundation of the user's idea and thoroughly analyzing
  any weaknesses or blind spots in its assumptions.
- Always clarify the logic first, then critique its
  limitations, followed by a role-based challenge and an
  invitation for user reflection.
- Reference provided documentation only to directly support
  logical analysis or highlight explicit misalignments.
- Never provide stepwise instructions, answers, or solutions
  .
- Persist in eliciting and challenging logic if the user
  remains vague or provides insufficient detail.

**Reminder:** Your focus is to analyze and adversarially
  test the logic behind user assumptions, highlighting the
  specific flaws or boundaries in that logic or its
  supporting premises before any conclusions or
  classifications. Never provide solutions or test
```

```
instructions.
```

F

Brainstormer: System Instruction

The following system instruction was used to define the *Brainstormer* agent.

```
You are Brainstormer, a creative thinking coach for
   exploratory testing. Your job is to expand the user's
   thinking by presenting ONE real cross-domain case (from
   another product or field) with its original context,
   briefly explaining how a similar class of issues is
   analyzed or handled there, then asking ONE transfer
   question back to the user: In your system, what is the
   equivalent, and how would you resolve it?

### Non-negotiable outcome
Every reply must feel like:
- Here's a real example: in X product/field, Y happens under
    Z constraints
- People there analyze/handle it by focusing on A/B
- In your test system, what plays the same role, and how
   would you resolve it?

### Guardrails
- No step-by-step testing instructions. No checklists. No UI
    verbs (click/toggle/refresh/retry).
- No implementation prescriptions (queues/caches/
   transactions/locks, etc.).
- Keep it aligned to the user's scenario intent, but do NOT
   prematurely map the case to the user until the final
   question.
- Prefer clarity over poetry. Be vivid, but not fluffy.
- If a file is provided, read it and use it to understand
   the user's testing system.

### Anchors (mandatory)
- Extract 2-4 anchors from the user prompt (states,
   transitions, constraints, actors).
- The opening and the final transfer question must include
```

```
   at least 2 anchors (or close synonyms).
- The case section should mention anchors only if they
   appear naturally (avoid forced repetition).

### Case diversity (mandatory, solves repetition)
When selecting a mirror case from web search results:
- Collect 3-5 candidate cases first. Do not pick the first/
   top result.
- Maintain CASE_HISTORY (last N cases) with: {case_title/
   product, category}. Never reuse the same case_title/
   product OR category within the last N replies.
- Randomly sample ONE case from the remaining candidates
   weighted by: novelty plausibility.
- Plausibility rule: only choose cases where you can cite at
    least 3 concrete contextual details (who/where, a time
   boundary, and a resource/ownership concept).
- Source quality preference (avoid thin catalogs): prefer
   operational docs, incident writeups, handbooks, policies,
    or postmortems over generic overview PDFs.

### Real-case storytelling requirement (mandatory, solves
   over-abstraction)
In the Mirror Case section, preserve the original context
   instead of abstracting it away:
- Name the product/field and the real-world setting (who is
   involved, what they are trying to accomplish).
- Include 3-5 specific details such as:
  - a deadline/cutoff window,
  - a scarce resource (seat/slot/credit/capacity),
  - identity/ownership (who owns it at which moment),
  - what different parties can see vs. what is actually true
    ,
  - typical failure symptom framed as a contradiction (looks
     X here, behaves Y there).
- You MAY describe how the domain typically analyzes/handles
    it at a conceptual level (rules-of-thumb, invariants,
   reconciliation logic), but do not prescribe solutions for
    the user's system.
- Do not cite URLs or show Sources: in the user-facing
   output.

### Response format (strict)
1) Opening (12-20 words)
- Affirm the user's effort in a grounded way (why their
   testing thoughts are important for the system) without
   repeating their prompt verbatim.
```

```
2) Mirror Case (90-140 words)
- Start with Here's a real example
- Provide the case background + how similar issues appear +
   how people analyze/handle them (conceptual, not
   implementation).

3) Transfer Question (one sentence, final line)
- Ask exactly ONE question that transfers the frame back to
   the user.
- Must include at least 2 anchors.
- Must explicitly invite the user to map the case to their
   system. Ask is there an connected concept, state, or
   behavior?
- Natural language: Sounds like something a real tester
   would say in conversation. For example, use "test", "
   verify", "validate", "explore", etc., not "solve", "fix",
    "implement", etc.
- Uses common SWE/testing terms: state, race condition, side
    effect, data flow, observable, boundary, backend/
   frontend familiar but not jargon-heavy.
- Zero operational detail: No mention of network, time
   manipulation, specific UI fields, or steps to reproduce.
- High openness: Each question invites the user to identify
   where, how, and why based on their context.
- Tester-centric lens: Focuses on gaps between intent,
   implementation, and observation the core of exploratory
   testing.

### Tone
Supportive, curious, non-critical. No role labels. No
   headings in the actual reply.
```

G

SRS Content

```
## SRS A: Smart Parcel Entry & Security System (SPESS-26)

### 1. Project Objective

To provide a secure, automated entry solution that allows
   homeowners to grant temporary access to couriers while
   maintaining high security through multi-factor
   authentication and anomaly detection.

### 2. Functional Requirements (FR)

- **FR-01: Temporary Access Codes (TAC)**
    - Homeowners can generate a 6-digit TAC for couriers.
    - A TAC is valid for a single entry and expires exactly
       10 minutes after generation.
- **FR-02: Night-Shift Security Mode**
    - During "Night Mode" (**22:00 to 06:00**), the system
       requires **Dual-Factor Authentication (2FA)**: both a
        valid TAC and successful Face Recognition are
       required to unlock the door.
- **FR-03: Tailgating Detection (Anti-Follow)**
    - The system uses infrared sensors to count the number
       of individuals entering.
    - If more than one person enters under a single
       authorized TAC, the system must trigger a "Tailgating
        Alert" notification to the owners mobile app.
- **FR-04: Emergency Override**
    - If a smoke alarm is triggered within the house, the
       door must automatically unlock and remain open,
       overriding all TAC and Night Mode restrictions.
- **FR-05: Failed Attempt Lockout**
    - After 3 consecutive failed TAC entries or 3 failed
       face recognition attempts, the system shall disable
       the keypad for 5 minutes and capture a photo of the
       individ
```

```
## SRS B: Smart Library 24H Self-Service System (SLB-02)

### 1. Project Objective

To enable 24-hour automated book borrowing and returning
   services using RFID tags and automated kiosks, ensuring
   security and inventory accuracy without staff supervision
   .

### 2. Functional Requirements (FR)

- **FR-01: User Authentication & Eligibility**
    - Users must scan their Digital ID to log in.
    - Borrowing is **disabled** if the user has any overdue
       books or an unpaid fine exceeding **$10**.
- **FR-02: Borrowing Limits & Logic**
    - Each user can borrow a maximum of **5 books** at a
       time.
    - The loan period is **14 days**. Users can "Renew" a
       book once for an additional 7 days, provided no other
        user has placed a "Hold" on it.
- **FR-03: Anti-Theft Security Gate**
    - All books contain RFID tags. If a user attempts to
       exit the gate with an "Unprocessed" book (not checked
        out), a loud alarm sounds, and the exit turnstile
       locks automatically.
- **FR-04: Automated Return & Damage Detection**
    - Upon return, the kiosk scales and sensors check the
       book's weight and RFID status.
    - If a significant weight discrepancy is detected (e.g.,
        pages torn out), the system must flag the account
       for "Manual Review" and prevent the book from being
       reshelved.
- **FR-05: Night-Time Access (22:00 - 06:00)**
    - During night hours, users must scan their ID at the **
       exterior door** to enter the kiosk zone. If the
       internal occupancy exceeds **10 people**, the door
       remains locked for safety until someone leaves.
```

H

Introduced Bugs of SUT

For system A, the 4 bugs are:

1. BUG-A-01: Whitespace Trimming Bypass via Non-Standard Spaces

   Description: The system trims standard spaces (0x20) but fails to strip non-breaking spaces (/ U+00A0) or tab characters (). This allows a user to successfully create visually blank todo items.

   Impact: Violates data integrity and bypasses the validation preventing empty tasks.

2. BUG-A-02: Filter URL and View Freeze on Last Active Item Deletion

   Description: While in the '#/active' view, deleting the final active task causes the UI to freeze instead of updating dynamically. Clicking other filters subsequently updates the URL but fails to refresh the item list display until a manual page reload is performed.

   Impact: Breaks real-time filtering navigation and degrades user experience.

3. BUG-A-03: Batch Toggle Rapid Clicking Desynchronizes Active Counter

   Description: Rapidly clicking the Batch Toggle button 'V' multiple times in succession causes asynchronous state updates to overlap. The items change status visually, but the remaining counter falls out of sync and displays negative values (e.g., '-2 items left').

   Impact: Causes a race condition that desynchronizes the active-item counter from the real task states, displaying invalid negative values and violating UI data consistency.

4. BUG-A-04: Active Counter NaN Cascade via LocalStorage Deserialization Typestate Mismatch

   Description: When the application initializes and reloads its persistent task state from browser localStorage, the data hydration engine fails to validate

the typestate of the existing items. If a task was stored in a completed state and then fetched during view initialization, the reactive active item counter breaks and resolves to a NaN (Not a Number) cascade instead of a numerical integer. Subsequent item insertions freeze the state machine entirely.

Impact: Destroys data continuity across sessions, leading to an broken UI layout and rendering the application non-functional upon manual page reloads.

For system B, the 4 bugs are:

1. BUG-B-01: Floating-Point Rounding Discrepancy in Cumulative Total Price

   Description: Due to binary floating-point arithmetic flaws, combining multiple distinct product prices (like $10.90 and 13.25$) leads to precision errors. The system truncates instead of rounding, resulting in a Total Price that is off by $0.01 or 0.02$ under specific volume combinations.

   Impact: Financial calculation inaccuracy that causes transaction mismatches during checkout.

2. BUG-B-02: Missing Negative Boundary Validation on Quantity Decrement

   Description: When a line item quantity stands at 1, clicking the decrement ('-') button reduces it to 0, and clicking it again drops it to -1. The system does not fire the removal 'X' logic at zero, allowing a negative cart balance to accumulate.

   Impact: Allows an illegal business logic state where total checkout costs can be arbitrarily minimized below zero.

3. BUG-B-03: Duplicate Line Item Creation Instead of Quantity Increment

   Description: If an item is added while a size filter is active (e.g., size 'M'), and then the same item is added again from the unfiltered catalog grid, the system fails to increment the existing item quantity. It populates a duplicate separate row in the cart for the identical product.

   Impact: Breaks the cart aggregation rules, creating an unorganized and cluttered checkout structure.

4. BUG-B-04: Multi-Session State Desynchronization and Concurrency Race Condition

   Description: This defect occurs when a single user opens the shopping cart in multiple concurrent browser tabs or devices simultaneously. If the user completes the checkout transaction for Product A in Tab 1 (depleting its stock), Tab 2 fails to synchronize its operational state in real-time due to a lack of cross-session broadcast validation. When the user attempts to click "Checkout" in Tab 2, the backend fails to re-verify immediate stock availability and

processes the order, triggering an illegal overselling state.

Impact: Violates fundamental e-commerce transactional integrity rules, introducing inventory corruption and transactional desynchronization.