VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

SOFTWARE ENGINEERING STUDY PROGRAM

Usability-Driven Security Design Patterns

Naudojimo patogumu pagrįsti saugumo projektavimo šablonai

Master's thesis

Author: Jahanzaib Manzoor Palh (signature)

Supervisor: Assoc. Prof. Dr. Kristina Lapin (signature)

Reviewer: Assoc Pro. Dr Vytautas Čyras (signature)

Vilnius – 2026

SUMMARY

All authentication design patterns share a fundamental structural weakness: users cannot verify a site's legitimacy before entering their credentials, leaving them vulnerable to phishing and spoofing. The baseline pattern examined in this thesis, security question authentication, performs poorly on both usability and security dimensions, offering no phishing resistance and failing across recoverability, accessibility, navigation, and feedback criteria.

This thesis proposes the Semantic Authentication System (SAS), a design pattern built on two independently managed trust factors: a personal secret phrase verified semantically via vector embeddings and LLM judgment, and a user-generated personal image displayed before any input is requested. The pre-login image serves as a passive phishing trust signal that a fake page cannot replicate. At the same time, semantic verification alleviates the exact-recall burden of traditional knowledge-based authentication. Either factor can be updated independently at any time.

Evaluation across nineteen criteria yielded Completely Implemented ratings in usability, deployment, and security perspectives. Empirical testing with 10 participants yielded a 100% phishing detection rate, a mean satisfaction rating on the Likert scale of 4.37/5, and a NASA-TLX cognitive load score of 35.6. SAS demonstrates that AI-driven semantic flexibility, combined with a personal visual trust token, can simultaneously resolve the usability, accessibility, and phishing-resistance gaps left unaddressed by current mainstream authentication patterns.

Keywords: Usable Security, Semantic Authentication System, Spoofing Prevention, Human-Computer Interaction (HCI), Accessibility in authentication, Large Language Models, Cognitive Load, Anti-Phishing Design

CONTENTS

SUMMARY **1**
LIST OF TABLES **3**
LIST OF FIGURES **3**
INTRODUCTION **4**
1. COMPARATIVE ANALYSIS OF AUTHENTICATION DESIGN PATTERNS **8**
1.1 How do the currently available authentication patterns lack usability and not protect the user against spoofing? **8**
1.1.1 Usable Security Analysis 8
1.2. What usability and security factors should be kept in mind while making comparisons among design patterns? **24**
1.2.1 Deployment: How Authentication Patterns Hold Up in the Real World 29
2. SEMANTIC AUTHENTICATION SYSTEM: DESIGN AND IMPLEMENTATION **35**
2.1. What is suggested to fix the currently available usability security design patterns to improve usability and security (if any are needed)? **35**
2.1.1 Usability-Based Analysis of Semantic Authentication System 35
2.1.2 Security-Based Analysis of Semantic Authentication System 37
2.1.3 Deployment-Based Analysis of Semantic Authentication System 38
3. VALIDATION OF THE PROPOSED USABLE SECURITY PATTERN **43**
3.1. Are the test results satisfactory for usable security authentication? **43**
3.1.1 Participant Profiles and Session Overview 43
3.1.2 Task Performance Analysis 44
3.1.3 Quantitative Measures, Likert and NASA-TLX 53
4. GUIDELINES TO AVOID PHISHING ATTACKS **58**
4.1 What are the guidelines that could help avoid a successful phishing attack? **58**
4.1.1 GL1 Always Check for Your Personal Image Before Entering Any Response 58
4.1.2 GL2 Update Your Personal Image and Secret Phrase Independently 58
4.1.3 GL3: Verify the Website Address Before Submitting Any Credential 59
4.1.4 GL4 Maintain the Confidentiality of Your Secret Phrase at All Times 60
4.1.5 GL5 Use Voice Input Considerately in Shared or Public Environments. 60
4.1.6 GL6 Exercise Caution Regarding Unsolicited Login Requests 61
4.1.7 GL7 Update Your Secret Phrase Promptly 62
4.1.8 GL8 Authenticate Only from Personal and Trusted Devices 62
5. KEY FINDINGS AND EMPIRICAL EVALUATIONS **64**
5.1 Security Questions Are the Single Worst Performing Pattern. 64
5.2 Semantic Authentication System. 64
5.3 High User Satisfaction with Low Cognitive Load. 64
5.4 Phishing Detection Rate. 64
5.5 Voice-Only Authentication Is Fully Viable, Including Blind Users. 64
CONTRIBUTIONS, LIMITATIONS, AND FUTURE WORK **65**
RESULTS AND CONCLUSIONS **66**
REFERENCES: **68**
APPENDIXES **72**

LIST OF TABLES

Table 1 Key Terms and IEEE Definitions 5
Table 2 Usability perspective in usable design patterns 25
Table 3 Security perspective in usable design patterns 27
Table 4 Deployment perspective in usable design patterns 33
Table 5 Participant Profiles and Session Overview 44
Table 6 Registration Task 45
Table 7 Sign-In and Semantic Verification Task 46
Table 8 Detailed Analysis of Usability Criteria 47
Table 9 Detailed Analysis of Deployment Criteria 50
Table 10 Detailed Analysis of Security Criteria 52
Table 11 Resilient-to-Phishing (S1), Primary Security Finding 52
Table 12 Likert Satisfaction Scores 55
Table 13 NASA-TLX Cognitive Load 55
Table 14 Design Recommendations 56

LIST OF FIGURES

Figure 1. Research Methodology 7
Figure 2. Logical System Context 41
Figure 3. Layer Application Structure 42
Figure 4. Sign-in Assurance Pipeline 42
Figure 5. Sign-in Assurance Pipeline 43

INTRODUCTION

Usable security design patterns are practices that aim to make security measures usable without compromising on the protection of sensitive data. In recent years, the growing use of technology has led to an increase in security breaches, underscoring the need for more effective, user-friendly security measures. One of the most common security breaches has involved phishing attacks against authentication systems. Phishing is a technique used to commit electronic fraud and steal credentials or other confidential information from end users. The attackers, also called "phishers," use multiple channels (social media and email) to trick users [1]. The technique works by linking a front-end on any well-known website to the attacker's back-end of choice. As a result, an attacker steals an end user's authentication credentials.

According to the Anti-Phishing Working Group (APWG), the number of attacks in 2016 was 1,220,523, and by September 2017, it had reached 710,350 [2]. Later, Facebook users suffered greatly when hackers sent links that looked like Facebook's login page, but in the background, they used a PHP script (running on unknown domains) to steal credentials. A username and password alone are insufficient for authentication, leaving users vulnerable to spoofing attacks. Therefore, it is recommended to add a second layer of authentication, known as multi-factor authentication (MFA). "Multi-factor authentication is an electronic authentication method in which a computer user is granted access to an application or a website only after successfully presenting two or more factors, or pieces of evidence" [4]. Various studies have been conducted to identify vulnerabilities and develop usable security design guidelines, which are distributed across multiple sources [4].

The problem arises when the designer is searching for a specific solution. Therefore, these guidelines should be collected and organized into a taxonomy that maps usable security design problems to solution guidelines, such as design patterns. Over time, developers and designers began developing usable security design patterns, while hackers found new techniques to breach them. Most computer users are unaware of how computer systems work, what security measures they should take, and how they should respond in the absence of security indicators.

Previous research found that no scheme is perfect compared to traditional passwords in terms of either usability or security [5]. However, we can recommend solutions based on which design patterns perform better than traditional passwords in specific scenarios and which perform worse. To overcome this problem, researchers have designed various authentication schemes to enhance system and data security. But many of these patterns have usability issues and either fail to implement security measures at all or fail to detect weak passwords.

Terms and Definitions

In this section, we'll define and clarify essential terms for our discussion, including authentication patterns, usability, design patterns, usable security, and spoofing, as defined by the IEEE.

Table 1. Key Terms and IEEE Definitions

Term	IEEE Definition
Authentication pattern	Reusable, security-focused solutions to common authentication problems, which embody a set of interrelated roles, policies, mechanisms, and protocols [34].
Usability	The extent to which specified users can use a system, product, or service to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [35].
Design pattern	Generally, reusable solutions to commonly occurring problems in software design [36].
Usable security	The degree to which a security system is easy to use and understand and does not hinder the primary task(s) for which it was designed [37].
Spoofing	The act of subverting the operation of a system by presenting false information, often gaining an illegitimate advantage [38].
Baseline pattern	A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development [IEEE Std 610.12-1990].

Research Objective and Aim

The research objectives are usability and security, as well as a new, improved multi-authentication method for secure authentication. Incorrect human decision-making practices play a vital role in phishing: most websites that require authentication are spoofed by phishers, and credentials are stolen. The phishers play with a human’s psychological behavior and fool them into making mistakes.

The most used authentication method is username and password, which weakens security [6]. While researching, the term "password" was used to refer to any PIN, graphical object, or secret key, and "text password" to a traditional string in a username/password. We need to identify usability factors that lead to security breaches and successful phishing attacks. We will be searching not only with technical lenses but also from a user-centered design perspective. We need

to explore ways to analyze the tasks the user needs to perform as part of authentication, how users process these tasks, and how they affect human decision-making and the outcomes of those decisions, which can result in either successful authentication or spoofed authentication [7].

A range of authentication design patterns are identified for evaluation in this study, namely: (1) Robot Verification, (2) Email Verification, (3) SMS OTP, (4) Image Verification, (5) Math expressions, (6) security PIN, (7) Google Authenticator, (8) Puzzle Solving, (9) Facial Recognition, (10) Fingerprint Authentication, (11) Security Image, (12) Security Question Authentication, (13) Call OTP, and (14) Voice Recognition Verification.

These techniques offer different degrees of technical security but are only effective if the user can easily handle the cognitive load and decision-making process. This work explores how the complexity or familiarity of each method affects users' tendency to ignore safety measures or fall victim to spoofing attacks.

The primary objective of this research is to develop an extensive anti-phishing authentication guideline that emphasizes human usability to reduce security breaches. This is because phishing attacks are now widespread, and relying solely on technical authentication is insufficient given the weaknesses of human decision-making.

To attain this goal, the research is split into the following specific objectives:

- Conduct a comparative analysis of current authentication methods to identify common usability flaws that lead to spoofing.
- Develop a methodology based on cognitive walkthroughs to evaluate how users interact with security prompts.
- Determine the specific human usability factors that directly contribute to security breaches in anti-phishing protocols.
- Synthesize findings into a list of actionable design patterns to improve both security and usability.

Research Questions

RQ1: How do the current available authentication patterns not protect the user against spoofing?

RQ2: What usability factors should be kept in mind while making comparisons among design patterns?

RQ3: What is suggested to fix the currently available usability security design patterns to improve usability and security (if any are needed)?

RQ4: Are the test results satisfactory for usable security authentication?

RQ5: What are the guidelines that could help avoid a successful phishing attack?

Research Methodology

In our approach, we use a technique to analyze, identify, and fix vulnerabilities in anti-phishing authentication schemes. A technique is a process for developing a design pattern from a summary of patterns and an examination of usability factors, by comparing multiple available design patterns and forming new or improved ones. The improved pattern is then applied to certain authentication objects. The test results are generated, recommendations are formulated based on them, and the results are compiled into a list of guidelines to help prevent phishing attacks. The visualization in *Figure 1* describes the entire process of the thesis work and its outcomes.

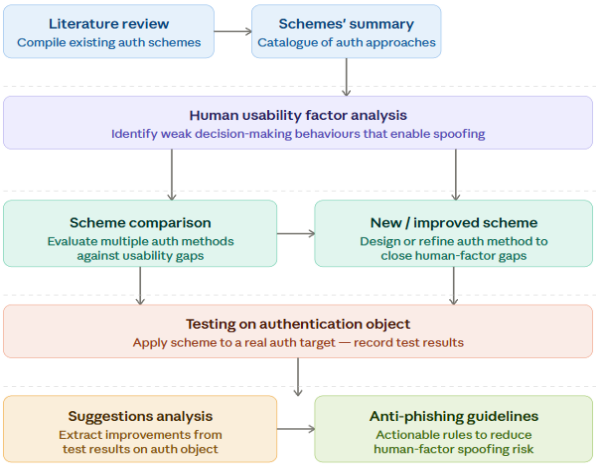

Figure 1. Research Methodology

1. COMPARATIVE ANALYSIS OF AUTHENTICATION DESIGN PATTERNS

This literature review aims to answer the following question(s) related to usable security design patterns.

1.1 How do the currently available authentication patterns lack usability and not protect the user against spoofing?

The criteria selected for this analysis are derived from a synthesis of established Human-Computer Interaction (HCI) heuristics and cybersecurity authentication standards. These specific metrics were chosen because they represent the critical intersection where user behavior meets system defense.

1.1.1 Usable Security Analysis

Robot Verification. Robot verification determines whether a system is accessed by a human or an automated program (a bot), to detect spam, malicious activity such as fake ratings and reviews, political bias [8], and unauthorized access. CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is one such method, challenging bots to solve a problem in a specific manner, even if it is not the correct way to solve it [8], typically by identifying images, solving puzzles, or typing distorted or obscured characters.

Robot verification performs acceptably in terms of efficiency. Although overly complex CAPTCHA risk deters legitimate users, the probability of an obstructively difficult challenge is around 10%, which is not a systemic concern. Learnability is likewise a strength, as the process is intuitive, and the instructions impose lower cognitive load.

Familiarity is more qualified. Distortion of characters and numerals raises difficulty but reduces legibility, leaving a considerable share of users unable to decipher them and narrowing the pattern's reach. Portability, by contrast, is solid: the pattern works across mobile and tablet without platform-specific limitations.

Accessibility is the most critical deficiency. The pattern makes no provision for users with disabilities and offers no alternative modalities to the standard visual interface, creating a fundamental gap in inclusive design. Navigation adds procedural redundancy: even after successful verification, users must still click a button to proceed, since no automated redirect is provided, interrupting an otherwise completed flow. Feedback is only partial, as a green checkmark visually confirms success. Still, no textual or screen-reader-accessible status message accompanies it, creating ambiguity for users who cannot perceive visual cues.

Consistency is another concern: instead of following the now-dominant single-action or

invisible verification paradigm, the pattern introduces an idiosyncratic sequence of manual steps that diverges from established user expectations and fragments the experience.

The pattern performs well in terms of visual clarity and minimization, presenting graphics without perceptible resolution loss and keeping textual content deliberately sparse for a clean, interpretable design. Flexibility is partially achieved through some customization to accommodate varied user needs, though it falls short of full implementation.

The pattern shows considerable vulnerabilities across multiple threat dimensions. For phishing resilience, it offers no effective protection: automation tools, most notably Selenium, bypass the verification layer with little resistance, rendering it insufficient against scripted or bot-driven attacks [22]. Resilience against theft is equally lacking, as no mechanism detects or prevents the exploitation of stolen credentials or session data, leaving this vector entirely unmitigated. Physical observation presents an analogous weakness: because the verification steps involve no randomization or obfuscation, an observer can replicate the interaction simply by observing it, with no technical barrier to circumventing it.

For internal observation, the defensive capacity depends on the host browser's security features rather than any safeguards implemented by the pattern itself. If the browser fails to provide adequate protection, the pattern offers no compensatory mechanism, leaving it exposed to automation tools operating within the browser environment.

Email Verification. Email verification is one of the most widely used authentication methods, which falls under multi-factor authentication. In this method, after providing a username/email and password, an email is sent to the user's email address. The user must click the verification link in the email; after that, they will be granted access to the system. We check for spelling mistakes in the email, non-existent domains, and inactive email addresses. Email Verification tools automatically run these audits, identifying potentially fraudulent, misspelled, or defunct email addresses for removal [9].

Users are familiar with email verification, as it is a common practice in online registration. Because it requires fewer steps and effort to authenticate within a system, it is better than passwords, too. Email verification is intuitive: users click a link or enter a code sent to their email, and the system guides them through instructions.

The design pattern navigates to another page automatically when clicking an email verification link or entering the correct 6-digit verification code. It also provides textual feedback, confirming the status of the task performed. The design pattern features clear graphics and text, with virtually no blurring. The design has reduced the text, making it easier to read, learn, and

understand. The user must verify their account either with a one-click verification link or by entering a 6-digit code. Unlike passwords, email verification does not require specific rules for the input field.

Short Message Service (SMS) One-Time Password (OTP). Short Message Service (SMS) One-Time Password (OTP) is a further method of verification used in multi-factor authentication. Upon submitting a username/email, the user receives a single-use OTP that expires upon consumption and is delivered to the registered SIM number associated with the account. The concept of OTP was first proposed by the American scientist Leslie Lamport in the early 1980s [10], and the technique is now widely adopted by banks to authenticate user identities in transactional contexts.

In terms of efficiency, SMS verification provides near-instantaneous identity confirmation, though network issues or delivery delays may compromise the process's overall responsiveness. Familiarity is well established, as the method is routinely used across numerous online services and contributes to user trust; users in regions with limited mobile access or unfamiliarity with SMS technology may, however, face challenges, though such populations are typically unaware of multi-factor authentication more broadly. Using contact numbers as usernames, combined with a 6-digit code sent via SMS, is a common practice in mobile applications.

The process is intuitive and easy to learn, requiring only the entry of a code received via text. Clear instructions within the SMS, accompanied by the 6-digit verification code, guide the user through verification, and the interface navigates automatically upon correct entry, removing the need for an additional button press. SMS messages are inherently concise, and the verification flow itself is limited to essential steps and basic security precautions with the 6-digit code imposing none of the formatting constraints associated with password-based input.

Consistency is well maintained, with SMS verification providing a uniform experience across diverse applications and platforms. Flexibility is similarly strong, as the method integrates readily into varied platforms and remains accessible to any user with a mobile phone, ensuring broad reach.

Several limitations must nonetheless be acknowledged. Users without access to a mobile device or those with certain disabilities may encounter difficulties, as the pattern provides no provisions for users with disabilities. Likewise, users in areas with poor connectivity may experience delays. The method also introduces a dependency on mobile phones and SIM cards, which renders it less convenient than alternative patterns, and the design provides no mechanism to mitigate such situations. Furthermore, no in-SMS feedback is offered following navigation to confirm the status of the completed task.

The pattern, however, offers meaningful protection against phishing, as authentication relies on a 6-digit unique code generated on the server side. The involvement of a trusted third-party service provider [23] invoked within the developer's code prevents the phisher from intercepting input via the verification medium.

Conversely, the pattern implements no mechanism to protect against device theft. Information security in this regard depends entirely on the privacy settings configured by the user on the mobile device for both the home and lock screens. However, such precautions do help conceal sensitive information. The pattern is similarly vulnerable to physical observation, as an observer can replicate the actions performed; no internal mechanism mitigates this risk, and protection depends solely on securing the inbox and device's privacy settings.

Comparably, protection against phishing is reinforced using a server-generated verification link and a 6-digit code routed via a trusted third party, in this instance, Google, which incorporates a scanning mechanism and encryption. The developer's integration of this verification service into the code prevents phishers from extracting input from the checkbox, and the scanning layer also filters emails containing potential phishing links.

Call OTP. Call-based verification operates on the same principle as SMS OTP, with the One-Time Password delivered via a phone call to the user's registered mobile number at the point of transaction. This option is particularly relevant for users who cannot receive SMS messages, or for those with visual impairments or other disabilities that preclude reading text messages, and it is frequently deployed alongside SMS verification [11]. The method is implemented on well-known platforms such as Binance and is primarily adopted by banks and financial institutions to authenticate transactional activity.

Familiarity is well established, as Call OTP is routinely used across multiple online services and contributes to user trust. The process is intuitive and easy to learn, requiring only the entry of a code received by phone. Efficiency is comparable to SMS, offering a near-instantaneous alternative for identity confirmation. The call itself conveys few instructions and is accompanied by a 6-digit code, which the user enters into the input field, after which the interface automatically navigates to the next screen. Call OTPs are inherently concise, with no visual elements and clarity conveyed through audio. The verification flow is limited to essential steps: the 6-digit code is repeated twice and, unlike password-based authentication, imposes no formatting rules or extended input requirements.

Consistency is maintained through a standardized procedure for delivering one-time codes via phone call, producing a uniform user experience across platforms and applications. Flexibility

is similarly well realized, as the method accommodates users who cannot access SMS or prefer voice-based authentication, and it permits at least 3 call attempts to ensure successful completion.

Multiple usability limitations must be acknowledged. Delays in call delivery may compromise process efficiency, and users in regions with limited mobile access or unfamiliarity with Call OTP technology may face challenges. However, such populations are typically unaware of multi-factor authentication. Calls are commonly received from private numbers, which may cause user discomfort and reluctance to answer. Users without access to a mobile device, or those with certain disabilities, may encounter difficulties, as the pattern incorporates no provisions for users with disabilities. Connectivity limitations may likewise introduce delays, and the inherent dependence on mobile phones and SIM cards makes the method less convenient than alternative approaches. The design provides no mechanism to mitigate such situations, and no feedback is offered following navigation to confirm the status of the completed task.

The pattern does, however, provide meaningful protection against phishing, as authentication relies on a unique 6-digit code generated server-side and delivered via a call routed through a trusted third-party service provider [24]. The developer's integration of the call verification service within the code prevents the phisher from intercepting input through the medium. Furthermore, the call itself cannot be tapped or recorded unless the user is operating on a device belonging to another party.

Security Question Authentication. In this form of verification, the user selects a question during system registration. Then, a user is presented with a field to enter their answer, and the system stores it. On the next occasion, when the user wishes to sign in again, the system requests the user to enter the answer to the question they chose when signing up. The user must enter the answer, then access the account [12]. Many websites, including Upwork, use it widely. Users are familiar with this design pattern, as it is a common method for account recovery. The security question setup is intuitive, allowing users to choose questions and provide answers during registration. However, users might find it challenging to recall specific answers, especially if the questions are not memorable or were chosen long ago. If users forget their security questions or the questions are too complex, selecting questions from a list and writing answers takes time, leading to delays and user frustration.

Security question interfaces are typically designed for clarity, guiding users through the setup and authentication process, and clarity may be reduced if the questions are too complex or if the interface is cluttered. However, the questions provided are not tough, and answers are not restricted to specific word limits. Excessive security questions or complicated setups may lead to

user confusion and frustration, though some systems use fewer questions to avoid it. It must be noted, however, that crypto apps raise multiple questions, which frustrate users.

Users can use security question verification on desktop, mobile, and tablet, making it portable across any location or device. This usability factor is achievable because security questions will remain the same across platforms and devices. It also provides users with the flexibility to use the most common yet difficult-to-guess answers, and they can enter the answer in lowercase without issue. The user can also have multiple attempts to pass this security check.

The limitations must be acknowledged. The user cannot navigate to another screen automatically; they must click to render the next possible screen. The design pattern does not provide feedback after navigation to confirm the status of the task performed, and the system does not provide feedback when a user answers a question incorrectly. Additionally, users without access to a mobile device or those with certain disabilities may face challenges, as the design pattern does not provide a mechanism for users with disabilities.

The design pattern does not protect against phishing, as a phisher can create a list of security questions to steal information from the user [25]. The developer uses a design pattern on the client side that interacts with the server side, and the questions available on the client side could be stolen, with guesses sent to the server for verification. Such techniques make the design pattern unprotected from internal observation. The design pattern also does not implement any mechanism to protect against theft devices.

An observer can catch details, and the design pattern does not protect against physical observation because it provides no mechanism to hide the answer, placing all responsibility on the user to protect their information. Regarding trusted third parties, the client side manages the list of questions. When a question is selected, and an answer is entered into the prompt, the answer is sent to the trusted third party for verification.

Google Authenticator. It is an app developed by Google that generates a verification code by combining the secret with a truncated timestamp, hashing the result, and truncating the output. The user-supplied code is then verified using the same technique. It is also referred to as a Time-based one-time password (TOTP). We can set it up by scanning the QR code on our mobile phone, linking our accounts to it, and using the generated code to sign in with the latest code displayed in the app [13]**.**

Google Authenticator is familiar to users with experience with two-factor authentication apps, contributing to a sense of security. However, users new to two-factor authentication might find the setup and use less familiar or intuitive. Once set up, Google Authenticator is easy to use,

with clear instructions for scanning QR codes. Google Authenticator provides efficient two-factor authentication by quickly generating time-sensitive codes for enhanced security. Although opening the app and manually entering codes might add a step to the login process, it would have only a slight impact on efficiency.

The app interface is designed for visual clarity, displaying code in a clear and readable format. Google Authenticator meets the criteria of minimization, requiring only a simple entry code during the login process. Users will have updated codes every 30 seconds, and users can try multiple times. Immediate feedback is also provided, as users receive real-time authentication codes.

Google Authenticator is highly portable because it runs on mobile devices, allowing users to generate codes from anywhere. However, users without access to a mobile device may face challenges, and transferring the app to a new device can be cumbersome. Consistency with this method is quite easy to achieve, as the user only needs to provide authentication via the Google Authenticator app, and it remains consistent across all platforms.

Moreover, several usability limitations must be acknowledged. Clear instructions guide users through the setup process, making navigation straightforward, but users must copy the code, which makes the setup manual and needs automation. Users with visual impairments might also face challenges with the QR code scanning process, a critical step in the initial setup, and the design pattern does not address accessibility for these users.

The design pattern protects against phishing because it uses code generated after 30 seconds. Code generation is performed locally on the user's device, and the codes are not transmitted externally, making Google Authenticator resilient to internal observation by malicious actors within the user's network or system [30]. Google Authenticator also operates as a client-side authenticator, generating time-based codes locally on the user's device and relying on no trusted third party during authentication, making it suitable for scenarios where minimizing reliance on external entities is important.

Google Authenticator generates time-based codes that are visible only to the user on their device and does not transmit them over the network, making them resilient to physical observation by attackers attempting to intercept them in transit. Finally, there is no reliable mechanism to hide the code from a third party, and even iOS's face detection feature cannot hide it from an observer.

Fingerprints Verification. The most popular biometric method is fingerprinting because of its wide acceptance and usefulness. To register Fingerprints into a system, the scanner analyses key points known as minutiae. Minutiae occur at locations where ridges terminate or split ridge endings

and bifurcations. The relative locations and orientations of minutiae can be represented as a compact template and efficiently compared to other templates. In an evaluation of the utility of three biometrics (fingerprint, signature, and voice), fingerprints were found to be the most usable among the features considered [14].

Fingerprint authentication is familiar to users, especially with the widespread adoption of fingerprint sensors across various devices. However, some users, particularly those new to technology or with specific cultural considerations, may find it less familiar or acceptable. Fingerprint authentication is inherently learnable, as users quickly adapt to placing their fingerprint on a sensor for verification. However, they may need some initial guidance on how to properly position their fingers for accurate scans, especially with different fingerprint sensor technologies. Fingerprint authentication is also highly efficient, as it involves a quick scan of the user's fingerprint, reducing login time and making it preferable to passwords.

Clear instructions guide users on how to position their fingers for successful authentication, and immediate feedback is provided during the fingerprint scan, indicating whether authentication was successful. Fingerprint authentication interfaces are also designed for clarity, providing users with visual guidance during the scanning process. Fingerprint authentication meets the criteria of minimization, requiring only the simple act of placing a finger on the sensor for verification. Users can make multiple attempts and use any finger for verification, making the process more flexible.

Consistency is very easy to achieve in this process because the user only needs to use their fingerprints each time. Fingerprint authentication is portable across devices and is commonly integrated into smartphones, tablets, and laptops. However, users without access to devices with fingerprint sensors may not benefit from this authentication method, limiting its portability. However, users with certain skin conditions or finger injuries may face challenges with fingerprint authentication, and the design pattern does not address accessibility for such users [26].

The design pattern protects against phishing by requiring the user to provide a unique fingerprint to authenticate with the system. Fingerprint authentication is typically designed as a local authentication method, reducing reliance on external parties, and biometric data is often processed on the device itself, minimizing the involvement of third-party servers.

Fingerprint authentication is resilient to theft because it requires the physical presence of the authorized user; stealing a fingerprint alone is not sufficient for authentication, making it a robust defense against unauthorized access. Fingerprint authentication is also inherently resilient to physical observation since the biometric feature, the fingerprint, is not easily observable by others, and the scanning process involves a unique physiological trait that is difficult to replicate

visually. Fingerprint authentication systems, especially those implemented on secure hardware modules, can additionally be resilient to internal observation, as biometric data is typically stored and processed within a secure enclave, reducing the risk of internal breaches.

Facial Recognition. A common biometric authentication technique is facial recognition. It can be done withcameras, which are getting more common and affordable. For instance, it may be possible to verify a user without additional hardware by using a smartphone's camera. Face recognition differs from fingerprint recognition in that it doesn't require physical contact. A photograph is first taken to begin facial recognition. This image is analyzed to identifywhether it contains one (or more) faces. Following that, the faces are segmented. The algorithms are used to generate templates. Liveness detection is a crucial part of facial recognition systems because, without it, a snapshot could trick the algorithm [15]. Facedetection, face preprocessing, and face recognition are all part of a full face recognition system. To separate the face from the background pattern and extract face-difference features afterward, it is important to extract the face region during face detection.

Facial recognition is familiar to users, especially given its widespread use on smartphones. However, some users may be less familiar with or comfortable using it, particularly in regions where the technology is less prevalent or where privacy concerns are stronger. Facial recognition is easy to learn, requiring users to position their face within a camera frame for authentication. Facial recognition can also be efficient, providing a quick, seamless authentication experience for users without manual input, making it preferable to passwords. However, in some scenarios, facial recognition systems might experience delays or require optimal lighting conditions for accurate detection, impacting efficiency.

Clear instructions guide users to position their faces correctly for authentication, and users receive immediate feedback on the success or failure of facial recognition. The operating system provides a visual feature for face detection, and visuals are rendered for both successful and unsuccessful detections. Facial recognition requires users to present their face for authentication, with no other requirements needed to pass through the face detection or authentication process.

It maintains consistency by using a standardized process for identifying users based on facial features, ensuring a uniform authentication experience across devices and platforms. It also offers flexibility through a hands-free authentication method that accommodates users with diverse accessibility needs and preferences. Facial recognition is portable across devices with front-facing cameras, offering a consistent authentication method. However, users without front-facing cameras or those with faulty ones may face challenges, limiting portability.

Voice Recognition Verification. The sound of a user's voice represents their behavioral biometrics and can be used to authenticate the person. To authenticate users, existing speech authentication systems mostly rely on extracting their distinctive voice features, such as Mel-Frequency Cepstral Coefficients (MFCCs) and wavelet-based features. It might be text-dependence (requiring utterances from speakers) or text-independence (allowing any utteranceto be accepted). Text-independent methods are more adaptable and allow speakers to express themselves freely [16].

Voice recognition is familiar to users and perceived as a natural and intuitive authentication method. However, users in regions with diverse accents or those uncomfortable with vocal authentication may find it less familiar or acceptable. Voice recognition is easy to learn and requires users to speak specific phrases for authentication. It is efficient, allowing users to authenticate with a simple vocal command without the need for manual input, though efficiency may be impacted in noisy environments or if users have difficulty with pronunciation, affecting the system's accuracy. Moreover, the voice recognition model's accuracy is not as good as that of facial recognition and fingerprints.

Clear instructions guide users to speak specific phrases for successful authentication, and the navigation is automatic with no need to click buttons. Immediate feedback is also provided as users receive real-time feedback on the success or failure of voice recognition. Voice recognition provides an icon based on microphone graphics and wavelengths, giving the user clarity about the pitch and loudness. When pointing to the minimization, it requires the users to speak specific phrases for authentication, with instructions kept to a minimum and details covered using icons.

It maintains consistency by following a standardized process of analyzing and verifying users' voice patterns, ensuring a uniform authentication experience across different devices and platforms. It also provides flexibility by providing an alternative authentication method for users who prefer voice-based interactions or may have accessibility needs that require hands-free authentication. Voice recognition is portable across devices with microphones, offering a consistent authentication method. Though unlike other biometric methods, the feature is rare to find in most devices.

Voice recognition is accessible to users with various abilities, as it relies on vocal characteristics rather than physical inputs. However, users with speech impairments or those in environments with excessive background noise may face challenges that affect accessibility. Voice recognition is inherently resilient against traditional phishing attacks because the user's unique vocal characteristics cannot be easily replicated or intercepted, unlike passwords or tokens [26].

Voice recognition systems are being designed to operate without a trusted third party during authentication, and voiceprint comparison is often performed locally on the device or within a secure enclave. Voice recognition systems, especially those with secure implementations, can also be resilient to internal observation, as biometric data is typically processed within a secure enclave, with voiceprints stored securely via encryption and hashing to prevent unauthorized access.

Puzzle Solving. Puzzle solving is a technique for verifying real users and avoiding bots by giving users a puzzle to solve. It is a new approach to authentication used for two-factor authentication to provide an extra layer of security. This method relies on the user's ability to solve the puzzle to prove their authenticity and access the system [17].

With respect to efficiency, puzzle-solving can be effective for users who enjoy challenges, offering mental stimulation and a sense of accomplishment; however, those who find puzzles frustrating or time-consuming may experience a diminished overall experience. As difficulty levels rise, execution time must be moderated through the provision of simpler puzzles, which renders this criterion "Not Implemented". Familiarity is "Partially Implemented", as puzzle-solving is well known to users who engage with games, brainteasers, or problem-solving activities, producing a natural and engaging experience. Learnability is "Completely Implemented" because the puzzles are easy to learn, with clear rules and objectives that enable rapid comprehension of the mechanics. Accessibility is "Not Implemented" because the pattern includes no provisions for users with disabilities. Portability is "Completely Implemented": puzzle-solving through crosswords, Sudoku, or mobile games is inherently device-independent and is supported across both mobile and web interfaces.

Navigation is "Completely Implemented", with clear instructions and logical paths guiding users through the puzzle-solving process. Feedback is "Partially Implemented" because the pattern provides no textual confirmation, although a green tick indicates readiness to navigate to the next page. Consistency is achieved through a standardized process of presenting challenges and verifying user responses, ensuring a uniform authentication experience across platforms and applications. Flexibility is similarly well realized, with the method offering an interactive and customizable authentication mechanism that can be tailored to varying difficulty levels or user preferences, thereby enhancing engagement and adaptability. Visual clarity is "Completely Implemented", with puzzle interfaces designed to present challenges in a visually appealing manner, with clear text and imagery. Minimization is likewise "Completely Implemented", as the design focuses on the user on essential elements typically requiring around five straightforward moves to complete.

With respect to security, resilience to phishing is rated as "Protects": certain puzzles serve as a form of indirect authentication, requiring users to demonstrate problem-solving capability and introducing an additional layer of security beyond credentials, thereby complicating phishing attempts. The no-trusted-third-party criterion is marked as "Not Implemented" and "Protects" because the puzzle functions as a self-contained authentication mechanism that relies on the user's own ability rather than delegating authentication to an external party [31]. Resilience to theft is "Completely Implemented" and "Protects": puzzles rely on knowledge or skills possessed by the legitimate user, and the dynamic nature of puzzle scenarios varying over time or by instance adds further complexity that prevents stolen information from being reused successfully.

Resilience to physical observation is "Completely Implemented": puzzle-solving is a private cognitive activity that, unlike traditional authentication methods, does not lend itself readily to observation, and the variability of puzzle types presented to the user makes it difficult for an observer to predict or replicate the interaction. Resilience to internal observation is "Completely Implemented" and "Protects", as puzzle-solving is an independent cognitive task; differences in individual cognitive strengths produce diverse skill profiles, making it challenging for an internal observer to predict or exploit a single solving method consistently.

Image Verification. Image verification is an authentication method that works by selecting images of objects as written in the instructions. For example, after entering the email/username, we are asked to select images of the car/bus or traffic lights from the 9 squares, which together form a full picture, and we must select only the squares that contain the objects given in the instruction. It is also a type of Image CAPTCHA widely used on websites, and even Google uses this method of verification to prevent users from bots [18].

From an efficiency standpoint, the method under performs with complex or ambiguous visuals, forcing users to spend additional time on interpretation and slowing the authentication process. This problem is compounded by issues of familiarity: the images presented are often unfamiliar to users, making it one of the least intuitive verification approaches available. From a learnability standpoint, Image verification poses notable challenges for users with cognitive impairments or language barriers, who may find it difficult to understand what the task requires. Equally, those with visual impairments are significantly disadvantaged, since the method depends almost entirely on visual cues; alternative accommodations would be necessary to make it inclusive. The method fares better in portability and navigation. It works on virtually any screen-equipped device or platform, offering a consistent experience wherever it is used. The interaction itself is guided and intuitive: users are prompted to identify a named object within a set of images,

keeping the process straightforward to follow.

Feedback, however, is a clear weakness. No confirmation is given after an individual image is selected; the user is only advanced to the next step once the entire correct set has been chosen, which makes the process feel opaque. Visual clarity adds to this frustration. Low-resolution or poorly composed images already create difficulty, and splitting images into small squares worsens the problem, particularly when only a sliver of the target object appears in each square, making it genuinely unclear whether it should be selected.

On the more positive side, the method is consistent and flexible. It applies to a standardized validation process uniformly across devices and platforms, while still allowing for customizable, criteria-based selection. Its approach to minimization also works in its favor: requiring users to select only squares where the target object occupies more than half the space is a sensible threshold that improves completion rates.

Image verification holds up well against several common attack vectors. Its strongest quality is resistance to phishing: because challenges are generated dynamically in real time, automated scripts cannot reliably interpret or anticipate them, effectively blocking bots from impersonating legitimate users [32]. Similarly, it is resilient to theft. Visual recognition cannot be replicated using stolen credentials alone; an attacker would need to directly observe and respond to each unique challenge. Since image sets vary dynamically between sessions, capturing one challenge offers no leverage over future ones.

The method also limits exposure to both physical and internal observation. The private, visual nature of the interaction makes it hard for bystanders to extract useful information. At the same time, the randomization of challenges across users and sessions reduces predictability and limits the value of any internally observed patterns.

One structural distinction worth noting is that image verification relies on no external identity providers. Users authenticate directly within the service, eliminating the risks of trusting third parties. The trade-off is that there is no independent external layer of validation for a design choice that enhances autonomy but shifts full responsibility for security onto the service itself.

Math Expression. Math expression is an authentication method used as a second factor after email/username and password. In this method of verification, the usermust solve very simple math expressions that involve only basic arithmetic operations such as addition, subtraction, multiplication, and division. Users must solve this expression if it is implemented as an authentication mechanism [19]. Math expressions are a fundamental concept, making them familiar to most developers and users. Understanding and learning the math expression design

pattern is generally straightforward for those with a basic understanding of mathematics. Well-designed math expression patterns can be efficient, especially if they optimize the evaluation process; they are very simple, and most users can solve them very easily because they do not contain any complex expressions, just simple ones, like 1+2=?

Math expressions provide a visually clear and concise representation of mathematical operations, aiding readability and understanding. Minimizing math expressions, especially when dealing with complex formulas, can be difficult and may result in less readable code. Still, the math expression design pattern uses expressions with two operands and one operator, making them easy to understand and solve. For complex expressions, navigating the pattern can be challenging, particularly if it lacks proper structure and documentation; however, the math expression design pattern provides basic math expressions.

Math expression design patterns can also be easily used across different views, i.e., desktop and mobile. This verification maintains consistency by using a standardized process for presenting mathematical expressions and validating user responses, ensuring a uniform authentication experience across devices and platforms. Math expression verification also offers flexibility through a dynamic authentication method in which users solve mathematical expressions, accommodating individuals with varying preferences for interaction and problem-solving.

However, the user does not receive feedback immediately after selecting an image, and the process proceeds to the next screen only after a group of correct images is selected, making the process ambiguous for the user. Additionally, the design pattern does not provide any accessibility features for users with disabilities.

The security profile of the math expression design pattern reveals significant vulnerabilities across multiple threat dimensions. The design pattern can be bypassed in phishing attacks using digital image processing techniques, and the text field for math expressions can be stolen and compared with the expression result separately [22]. Proprietary algorithms or formulas used in math expressions may also be susceptible to theft, especially if they represent valuable intellectual property, since math expressions use basic expressions that are easy to notice and remember. Math expressions are similarly easy to understand, notice, and remember, meaning a user can bypass the pattern in physical observation.

Depending on third-party libraries or services for complex math expression evaluation introduces potential security risks, as trust in external code must be carefully considered.
Regarding internal observation, the design pattern for handling mathematical expressions, in and of itself, might not directly address resilience to it.

Security Pin. User experience, usability, and human factors considerations are prioritized in the installation of security PINs without affecting the system's overall security. It focuses on creating user-friendly, intuitive PIN-based authentication systems that minimize user errors while still offering reliable protection against unwanted access. The PIN will consist of 4 or 6 digits to facilitate user recall [20]. PINs are widely used and familiar to users, enhancing ease of adoption and reducing the learning curve, though familiarity with them is lower than with passwords. Users can quickly learn and remember a simple numeric PIN, contributing to a low learning barrier. Security PINs are typically short and quickly entered, contributing to efficient user interactions. The numeric nature of PINs also contributes to a clear visual representation, facilitating user input. PINs are usually 6-digit numeric data, and depending on the length and complexity requirements, PINs may not offer the same level of security as longer authentication methods like passwords.

PINs are straightforward to enter, requiring minor navigation within a user interface, and systems using PINs can provide immediate feedback on the success or failure of authentication attempts. Security PINs can also be easily carried or remembered without physical tokens, enhancing their portability.

Security PIN authentication maintains consistency by employing a standardized process of verifying user-entered PINs, ensuring a uniform authentication experience across different devices and platforms. This authentication also provides flexibility by allowing users to create and enter their own personal identification numbers, accommodating individual preferences and ensuring a customizable authentication method.

Moreover, the design pattern does not provide any accessibility features for users with disabilities. The design pattern protects against phishing attacks as long as the PIN is not exposed to unauthorized users, and it is specifically designed to avoid phishing attacks [33]. However, PINs can be susceptible to phishing attacks if users are tricked into entering them on fraudulent websites or interfaces. Regarding the encryption factor, the design pattern also protects against internal observation; however, if an insider has unauthorized access, they may observe or misuse PINs stored or processed within the system, and the case varies from system to system and organization to organization.

Security Image. Users are highly familiar with image-based interactions, making this pattern more usable than some text-based alternatives. Users can quickly learn how to identify and select security images, enhancing ease of use. Security images can offer efficient authentication, especially for users who find visual recognition tasks more intuitive than remembering complex passwords, and in the current scenario, the security image acts as a second layer of authentication

[21].

Navigation within the security image grid is typically straightforward and visually intuitive. Immediate visual feedback is possible, indicating whether the selected images match the predefined set. The visual nature of security images enhances clarity and can be more memorable for users. Depending on the size of the image grid, the security level may not be as high as with longer, more complex authentication methods, with the average number of images in the grid being six.

Security Image-based authentication is portable and can be implemented on various devices, including mobile devices and computers. This method maintains consistency by employing a standardized process of presenting selected image(s) during login and verifying user recognition, ensuring a uniform authentication experience across different devices and platforms. This method also provides flexibility by allowing users to select personalized but static image(s) during setup, providing a customizable and user-centric authentication method that accommodates individual preferences.

Regarding accessibility, as the design pattern relies on visual recognition, it is inherently suitable for users who are deaf or hard of hearing. For users with cognitive impairments who may find complex textual input challenging, a visual recognition-based pattern can be beneficial, and users with motor disabilities may find visual recognition tasks more accessible than complex typing. Recognizing that visual recognition may not be suitable for all users, the design pattern could offer alternative authentication methods, such as a secondary authentication factor or a fallback option for users who face accessibility challenges.

Security image-based authentication is resilient to phishing because the image refreshes when browsed, and the images change their locations on every refresh [34]. Moreover, every user has a unique image assigned during sign-up. Unlike passwords, the browser for the security image is not stored in the browser. Security images are also not like security PINs, which could be stolen. Security images cannot be stolen from websites and stored on a USB drive for unauthorized access. Finally, image-based authentication is vulnerable to shoulder surfing, as external observers can easily see the security images.

Conclusion: Current authentication patterns fail to protect against spoofing and automated attacks due to a combination of technical and procedural vulnerabilities. A major weakness is the vulnerability to automation; several patterns, most notably Robot Verification, do not provide adequate defense against tools like Selenium, which allow attackers to bypass security layers programmatically. This is compounded by a general lack of phishing resilience in "knowledge-

based" patterns such as Security Questions, Math Expressions, and Robot Verification, which are explicitly flagged as failing to protect users from phishing. Furthermore, the absence of server-side interaction in patterns managed purely on the client side makes them significantly easier to manipulate or spoof.

These systems also show a high susceptibility to observation, lacking the necessary mechanisms to hinder physical "shoulder surfing" or internal data scraping. Finally, inconsistent transitions within the user interface create a fragmented experience. By requiring idiosyncratic manual steps rather than providing seamless, automated redirects, these patterns establish a predictable set of hurdles that attackers can easily mimic within a fraudulent environment to deceive users.

1.2. What usability and security factors should be kept in mind while making comparisons among design patterns?

The comparative analysis of authentication design patterns is guided by a framework derived from ISO 9241-11 standards and Jakob Nielsen’s heuristics.

Efficiency is the usability factor that measures the speed and effectiveness of achieving tasks within a system or interface. Closely related to this is Familiarity, which refers to the degree to which a system or interface aligns with users' existing knowledge and experience, promoting ease of use and reducing the learning curve. Learnability, on the other hand, refers to the ease with which users can understand and acquire proficiency in using a system or interface.

Accessibility refers to the extent to which a system or interface can be used and understood by individuals with diverse abilities, including those with disabilities. Portability refers to the ease with which a system or interface can be transferred or adapted to different environments or platforms. Operability refers to the ease of operating and controlling a system or interface, ensuring users can interact effectively.

Navigation is the usability factor that encompasses the ease and intuitiveness of moving through different sections or areas of a system or interface. Feedback is the provision of clear, timely information to users that indicates the system's response to their actions and guides their decision-making. Consistency refers to the degree of uniformity and coherence in the design and behavior of a system or interface, promoting predictability and reducing cognitive load.

Security refers to the extent to which a system or interface safeguards users' data and privacy, and protects against unauthorized access or malicious activity. Flexibility refers to the ability of a system or interface to accommodate diverse user preferences, workflows, and customization options, thereby enhancing user control and adaptability. Visual Clarity refers to the

clarity and legibility of visual elements and information within a system or interface, ensuring ease of perception and comprehension.

Finally, Minimization refers to keeping a system or interface design simple by removing unnecessary elements and focusing on essential features to reduce complexity and cognitive load. Recoverability refers to a system or interface's ability to enable users to easily recover from errors or unintended actions, without data loss or negative consequences.

Compatibility refers to the ability of a system or interface to work seamlessly with other systems, devices, or software, promoting interoperability and ease of integration.

Table 2 below provides a visual representation of the usability factors associated with each design pattern mentioned in the previous question. Whereas *Table 3* represents results from a security perspective. The color scheme used in the table highlights different aspects of usability and security, enabling a quick assessment of each design pattern's strengths and weaknesses.

Table 2. Usability Perspective in Usable Design Patterns

Usability														
Design pattern	Efficiency	Familiarity	Learnability	Accessibility	Portability	Navigation	Feedback	Consistency	Flexibility	Visual Clarity	Minimization	Recoverability	Combability	Operability
Robot Verification	✓	○	✓	✗	✓	✗	○	✗	○	✓	✓	✗	✓	★
Email Verification	○	★	✓	✗	✓	✓	✓	✓	✓	✓	★	✓	✓	✓
SMS OTP	○	★	✓	✗	✗	✓	✗	✓	✓	✓	★	✓	✓	✓
Call OTP	○	✓	✓	✗	✗	✓	✗	✓	✓	✓	★	✓	✓	✓
Security Question	○	✓	○	✗	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓
Facial Recognition	★	○	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	−	✓
Fingerprints	★	✓	✓	✗	○	✓	✓	✓	✓	✓	✓	✓	−	✓
Google Authenticator	○	−	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓

Usability														
Design pattern	Efficiency	Familiarity	Learnability	Accessibility	Portability	Navigation	Feedback	Consistency	Flexibility	Visual Clarity	Minimization	Recoverability	Combability	Operability
Voice Authentication	◌	◌	✓	◌	◌	✓	✓	✓	✓	✓	✓	✓	⊖	⊖
Puzzle Solving	◌	⊖	◌	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Image Verification	⊖	⊖	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓
Math Expression	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓
Security Pin	✓	⊖	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Security Image	✓	✓	✓	◌	✓	✓	✓	✓	✓	✓	◌	✗	✓	★

◌ = Partially Implemented ✗ = Not Implemented ✓ = Completely Implemented

★ = Better than passwords ⊖ = Worse than passwords

⚠ = "Does not protect".

The usability evaluation of fourteen authentication methods against eleven human-centric criteria reveals a fundamental contradiction in modern authentication design. While most patterns achieve strong marks in visual and operational dimensions, they fail almost universally in the areas that matter most for an inclusive, seamless user experience.

Across the evaluated methods, ranging from Robot Verification to Security Images, nearly every pattern demonstrates high maturity in learnability, minimization, visual clarity, flexibility, and operability. The strengths suggest that designers have invested considerably in making authentication interfaces intuitive, visually legible, and functionally reliable. The consistency of these high ratings across diverse pattern types indicates an industry-wide convergence on strong surface-level design principles.

Nevertheless, beneath this polished exterior lie critical and systemic gaps. Accessibility stands out as the most alarming failure: without a single exception, every authentication method evaluated, including ostensibly modern solutions such as Facial Recognition, SMS OTP, and Google Authenticator, is marked as not implemented for accessibility. This uniform failure signals not an oversight in individual designs, but a structural neglect of differently abled users across the entire field. Compounding this, traditional methods like Robot Verification, Email Verification, and SMS OTP also fall short in navigation and consistency, introducing idiosyncratic, disruptive steps that undermine user flow.

When benchmarked against traditional passwords, the table further shows nuanced trade-offs. Patterns such as Email Verification, Facial Recognition, and Fingerprints outperform passwords in efficiency and familiarity. In contrast, others, such as Google Authenticator and Security PINs, underperform in terms of familiarity due to higher technical barriers. Biometric methods, though efficient, are constrained by portability limitations and the same accessibility failures that afflict every other pattern.

Ultimately, the data argues that the authentication design field has prioritized aesthetic and operational polish over genuine inclusivity, and that no current pattern achieves the balanced, human-centric standard that truly usable design demands.

Table 3. Security Perspective in Usable Design Patterns

Security					
Design pattern	Resilient-to-Phishing	Trusted-Third party	Resilient to theft	Resilient-to Physical observation	Resilient to Internal Observation
Robot Verification	⚠	✗	✗	✗	✗
Email Verification	🔒	✓	✓	✗	🔒
SMS OTP	🔒	✓	✓	✗	🔒
Call OTP	🔒	✓	✓	🔒	🔒
Security Question	⚠	✓	✗	✗	⚠
Fingerprints	🔒	✓	🔒	🔒	🔒
Facial Recognition	🔒	✓	🔒	🔒	🔒

Security					
Google Authenticator					
Voice Authentication					
Puzzle Solving					
Image Verification					
Math Expression					
Security Pin					
Security Image					

= Partially Implemented = Not Implemented = Completely Implemented

= Better than passwords = Worse than passwords

= "Does not protect".

The summary, from a security perspective, in *Table 3* evaluates authentication design patterns against five core security metrics: Resilience to Phishing, Trusted Third Party, Resilience to Theft, Resilience to Physical Observation, and Resilience to Internal Observation.

A security evaluation of fourteen authentication methods against key threat-resistance criteria reveals a landscape of profound inconsistency: while a select group of modern patterns demonstrates genuine, layered protection against real-world attacks, the majority of widely deployed methods leave users exposed in ways that are invisible to the very people depending on them for safety.

The strongest performers in this evaluation share a common characteristic: they authenticate something inherently difficult to steal, replicate, or fake. Biometric methods such as fingerprint scanning and facial recognition, alongside token-based systems like Google Authenticator, demonstrate high resilience against phishing, theft, and both physical and internal observation. Because these methods tie authentication to a physical trait or a time-sensitive possession, they significantly increase the cost of an attack, forcing bad actors to go far beyond simply tricking a user into clicking a fraudulent link. This layered difficulty is what meaningful security design looks like in practice.

In contrast, knowledge-based and challenge-based methods tell a far more troubling story. Robot Verification, Security Questions, and Math Expressions are explicitly flagged as failing to protect against phishing, a baseline threat that any authentication system operating in today's environment must account for. Security Questions are particularly problematic because the

information they rely on is often guessable, publicly available, or already compromised in prior data breaches. These methods persist not because they are secure, but because they are familiar and inexpensive to implement, a trade-off that ultimately shifts the burden of risk onto the user.

The evaluation of observation resilience further widens this gap. Biometric and possession-based factors hold reasonably well against both shoulder surfing and internal data interception, while Robot Verification and Math Expressions fail across nearly every observation-related metric. This means that even in everyday, low-threat environments, these patterns cannot reliably prevent a determined observer, human or automated, from capturing enough information to compromise an account.

Taken together, these findings make a compelling case that the authentication field has not yet resolved its most critical tension: the gap between systems that are easy to deploy and systems that genuinely protect people. Until security standards treat phishing resilience, theft protection, and observation resistance as non-negotiable thresholds rather than optional enhancements, widely used authentication methods will continue to offer users the feeling of security without its substance.

1.2.1 Deployment: How Authentication Patterns Hold Up in the Real World

A pattern's theoretical security value means little if it cannot function reliably across the environments real users operate in every day. An evaluation of how fourteen authentication methods perform across varying operating systems, browsers, devices, and servers reveals that technical capability alone does not determine a pattern's real-world effectiveness; compatibility and ease of deployment are equally decisive factors that are too often overlooked. *Table 4* gives a detailed evaluation of the findings.

Compatibility, in this context, refers to how consistently an authentication method integrates across a diverse digital ecosystem without losing functionality or creating unequal experiences for different users. A method that performs well on one platform but fails against another does not simply inconvenience users; it introduces security gaps and erodes the reliability that authentication systems are fundamentally designed to provide.

Operability extends this concern into the practical dimension of deployment, examining how easily a pattern can be installed, configured, and used by both the organizations implementing it and the individuals relying on it daily. A system that is difficult to set up or confusing to navigate in real use fails not because of poor design intent, but because it was never truly built with practical implementation in mind.

Taken together, compatibility and operability serve as the bridge between design theory and real-world application. The findings presented in this section argue that any authentication pattern unable to meet consistent standards across both dimensions cannot be considered a viable solution in the increasingly complex and demanding landscape of modern digital security.

Robot Verification. The design pattern is compatible with a wide range of operating systems, including Windows, iOS, and Android. By injecting a library into the code, the developer is responsible for the pattern's permanent deployment on the website, which requires less code than password-based authentication [43]. However, since the design pattern does not have a button to refresh the scheme, it lacks a recoverability feature and cannot be refreshed until the tab is refreshed.

Email Verification. The design pattern works on Windows, iOS, and Android, making it compatible with widely used operating systems. The pattern is deployed once and for all on the website, and the deployment is the developer's responsibility, using a Google-provided service injected into the code [44]. The pattern also provides a recovery feature. Unlike robot verification, there is no need to refresh the page when an incorrect 6-digit code is entered, and the verification link expires after approximately 30 minutes; the duration is usually stated in the email received.

SMS OTP. The design pattern works on Windows, iOS, and Android, making it compatible with widely used operating systems. The pattern is deployed once and for all on the website, and the deployment is the developer's responsibility, using an SMS verification cloud service integrated into the code [45]. The pattern also provides a recovery feature. Unlike robot verification, there is no need to refresh the page when an incorrect 6-digit code is entered, and there is always an option to resend an SMS code.

Security Question Authentication. Security question authentication is compatible across various devices and platforms, though compatibility issues may arise if the security question system is not standardized, leading to differences in implementations. Security question authentication also seamlessly integrates into the user registration and account recovery processes [47]. However, the system does not recover when a user answers a question incorrectly, indicating that the pattern lacks recoverability.

Security Question Authentication. Security question authentication is compatible across various devices and platforms, though compatibility issues may arise if the security question system is not standardized, leading to differences in implementations. Security question authentication also seamlessly integrates into the user registration and account recovery processes [47]. However, the

system does not recover when a user answers a question incorrectly, indicating that the pattern lacks recoverability.

Google Authenticator. Google Authenticator is compatible with various platforms and services that support the Time-based One-Time Password (TOTP) standard, and apps are available on Android, iOS, and the web. The Google Authenticator app can be installed from the app store or as a browser extension, and it uses MBs of storage. The user must scan a QR code and set up the authentication once; after that, the code refreshes every 30 seconds. However, code expiry depends on the server/system [50]. Recovery options are also available, typically involving backup codes or re-enabling two-factor authentication through account settings, and users who lose access to their mobile devices may access their accounts using Gmail.

Fingerprint Authentication. The fingerprint process requires a minimum number of steps to set up and use fingerprints. Recovery options are also available through alternative authentication methods, such as PIN or password, if fingerprint authentication fails. However, while fingerprint authentication is compatible with devices equipped with fingerprint sensors, providing a standardized method across platforms, not all platforms use fingerprints for two-factor authentication [48]. There might be version issues across devices, and older versions lack the fingerprint feature, which limits this design pattern to a specific audience.

Facial Recognition. The face detection process requires a minimum of steps to set up and use. Recovery options are also available through alternative authentication methods, such as PIN or password, if facial recognition fails. However, while face detection authentication is compatible with devices equipped with fingerprint sensors, providing a standardized method across platforms, not all platforms use fingerprints for two-factor authentication. There might be version issues across devices, and older versions lack the fingerprint feature, which limits this design pattern to a specific audience [49].

Voice Recognition Verification. Deploying voice recognition as part of a two-factor authentication system involves integrating it into the overall authentication process. The process is easy to implement, with a few steps to follow during voice sample registration. However, the accuracy is not as good as facial and fingerprint recognition, meaning it must be used in an alternative way most of the time. Recovery options are also available through alternative authentication methods, for example, PIN or password, in case voice recognition fails [51]. However, while voice recognition is compatible with devices equipped with functional microphones, providing a standardized method across platforms such as Android, iOS, and Windows, it is more commonly used in browser search than in authentication.

Puzzle Solving. Puzzle-solving methods, especially in digital formats, exhibit high compatibility across various platforms such as mobile devices, tablets, computers, and even physical mediums, ensuring users can enjoy a consistent and familiar experience regardless of the device they choose [52]. Puzzle-solving also seamlessly integrates into the user experience by providing a cohesive, enjoyable challenge, enhancing user engagement as the puzzle becomes an inherent part of the interactive environment. Digital puzzle-solving platforms often incorporate a variety of interactive elements, such as drag-and-drop functionality, zoom features, or hints, enhancing operability and making the experience more dynamic and engaging.

In case of errors, recovery options are generally available in puzzle-solving, allowing users to reset or retry challenges if they encounter difficulties.

Image Verification. Image verification is compatible with iOS, Android, and browsers and platforms that support visual displays, ensuring a standardized method across different environments. Image verification also seamlessly integrates into the login or verification process, offering an additional layer of security without causing major disruptions, and it is a one-time integration using a library or API [53]. Image verification usually also provides recovery options, such as retries or alternative verification methods, if users make errors or find the task challenging.

Math Expressions. The math expression is compatible with iOS, Android, and browsers on platforms that support visual displays, ensuring a standardized method across different environments. Math expressions also seamlessly integrate into the login or verification process, offering an additional layer of security without causing major disruptions, and can be integrated one-time using a library or API [19]. However, recovering from errors, especially those stemming from user input or external data, can be intricate and may require robust error-handling mechanisms, as the design pattern does not provide a way to refresh the input box or request a new expression.

Security Pin. The security PIN is compatible with iOS, Android, and browsers and platforms that support visual displays, ensuring a standardized method across different environments. The Security PIN is also set up one time, making it an easy, smooth process that can be done on both desktop and mobile phones [54]. However, PINs may be vulnerable to certain attacks, such as brute force attempts, particularly if not complemented by other security measures. If a PIN is forgotten or compromised, the recoverability process may be limited, requiring additional security measures, though the system does provide multiple attempts to enter a PIN.

Security Image. The security image-based authentication works with browsers such as Google Chrome, Firefox, and Internet Explorer, and its design patterns are compatible with operating

systems such as Android, iOS, and Windows [21]. The security images are also easy to use, as they require only a few steps; for instance, the user must select a single image during the signup and signing process, making them better than passwords. However, recovering from forgotten or compromised security images may require additional authentication steps or access to alternative recovery methods.

Table 4. Deployment Perspective in Usable Design Patterns

Deployment			
Design pattern	Recoverability	Combability	Operability
Robot Verification	✗	✓	★
Email Verification	✓	✓	✓
SMS OTP	✓	✓	✓
Call OTP	✓	✓	✓
Security Question	✓	✓	✗
Fingerprints	✓	✗	✓
Facial Recognition	✓	✗	✓
Google Authenticator	✓	✓	✓
Voice Recognition Verification	✓	✗	✓
Puzzle Solving	✓	✓	✓
Image Verification	✓	✓	✓
Math Expression	✗	✓	✓
Security Pin	✗	✓	✓
Security Image	✗	✓	✓

✓ = Partially Implemented ✗ = Not Implemented ✓ = Completely Implemented

In more detail, *Table 4 analyzes how various authentication design patterns perform across*

three critical deployment dimensions: recoverability, compatibility, and operability. An evaluation of fourteen authentication methods across three critical deployment metrics, recoverability, compatibility, and operability, reveals that the patterns most trusted for their security strength are not always the most practical to deploy, and that some of the most widely used methods carry significant operational weaknesses that directly expose users to risk.

There is a distinct difference between method types based on recoverability, which gauges how simple it is for a user to regain access if an authentication factor is lost or unavailable. Strong recovery operations are demonstrated by email verification, SMS- and call-based OTPs, and biometric techniques such as fingerprint, facial, and voice recognition, indicating that these systems are built on the reasonable assumption that users will occasionally lose access. Robot Verification, Math Expressions, and Security PINs, on the other hand, do not provide a native recovery pathway at the deployment level. This means that in the event of a system failure, users may not have a structured way to return to their accounts, which is both a security risk and a usability flaw. Evaluating how well a pattern integrates with other security layers in a multi-factor authentication stack produces perhaps the most surprising finding in this evaluation. Knowledge-based and possession-based methods, such as OTPs, puzzle-solving, and security PINs, integrate effectively with other authentication modules, making them practical building blocks for layered security. Biometric methods, however, despite their reputation for being advanced and reliable, are considered poorly suited for integration with other systems. Their dependence on specialized hardware makes it difficult for them to synchronize with broader authentication frameworks, revealing a significant gap between their perceived sophistication and their actual deployment flexibility.

The operational reliability of most authentication methods is adequate. Still, the outlier's security questions failing and CAPTCHA introducing unnecessary friction expose a deeper systemic issue: authentication design consistently underestimates human behavior. Poor recall, predictable choices, and susceptibility to manipulation are design failures, not user failures. Addressing them demands authentication frameworks built around how people behave, not how designers assume they will.

2. SEMANTIC AUTHENTICATION SYSTEM: DESIGN AND IMPLEMENTATION

2.1. What is suggested to fix the currently available usability security design patterns to improve usability and security (if any are needed)?

Traditional security question authentication relies on a fixed, predefined question selected from a list, to which the user provides an exact, verbatim answer. As demonstrated in the baseline analysis, this approach exhibits critical deficiencies across all three evaluation perspectives: usability criteria such as accessibility, navigation, and feedback are either partially or not implemented; security properties, including resilience to phishing, theft, and physical observation, are compromised; and the deployment criterion of recoverability is marked as not implemented.

The implementation presents the Semantic Authentication System *(See Appendix J),* an AI-powered authentication design pattern that replaces the fixed-answer security question paradigm with a semantically flexible secret phrase. During registration, the user defines a personal secret phrase (up to 100 characters, submitted via text or voice) and generates a secret image. At authentication time, the user is asked to describe the idea behind their secret in their own words. An AI backend combining vector embedding similarity (cosine distance) and large language model (LLM) semantic prompts to select the security image and write security phrases. The judgment evaluates whether the user's response carries the same meaning as the registered secret, irrespective of exact wording. The system is implemented as a web application *(See Appendix G)* supporting both Groq and OpenAI backends (switchable at runtime), with full text-to-speech (TTS) and speech-to-text (STT) accessibility support. Three retries are permitted during the semantic verification step, and users may update their secret phrase at any time via a dedicated profile page. The following sections analyze the improved design against each of the three evaluation perspectives, usability, security, and deployment, using the same criteria framework applied to the pattern, supplemented by narrative discussion.

The security perspective evaluates the degree to which the authentication pattern resists common attack vectors. The baseline analysis identified multiple critical weaknesses in traditional security questions, including vulnerability to phishing, theft, and observation attacks. The Semantic Auth System addresses these through a combination of semantic ambiguity, embedding-based storage, and multimodal input.

2.1.1 Usability-Based Analysis of Semantic Authentication System

While the concept of a secret phrase is novel relative to traditional security questions, the registration and sign-in flows are familiar in structure: username, password, then a second verification step, and the step-by-step onboarding tour (Steps 1–6 at sign-in and Steps 1–5 at

registration) progressively introduces users to the semantic concept, minimizing the learning overhead. The system offers an interactive guided tour during both registration and sign-in, and tooltips clearly explain the concept of semantic secrets. For example, "This is the idea the system will check later, not a second password. You can paraphrase freely at login. Users quickly understand that they are not required to remember an exact phrase, only the idea behind it. During registration, the user generates a personalized AI-based security image using a self-chosen phrase. At login, this image is presented alongside decoy images that may vary in color or style but share the same object category, requiring the user to identify their original image. Three selection attempts are permitted; failed attempts trigger an exponentially increasing lockout timer before the user may retry.

The Semantic Auth System significantly improves efficiency compared to traditional security questions, as users are not required to select from a predefined list of questions or memorize exact answers. Instead, they define a personal secret phrase during registration and sign in to express the same idea in their own words, a process that is faster and more cognitively natural. The dual backend embedding similarity, LLM semantic judgment, and image verification ensure the system evaluates meaning rather than exact wording, reducing friction. The system provides explicit status feedback at each stage. After registration, a green banner reads, "Registration successful. Sign in below." During sign-in, the step indicator, Step 1 of 2 and Step 2 of 2, communicates progress, and semantic verification and security image selection results are communicated clearly to the user, resolving the feedback gap present in the design. The system also implements a clear two-step sign-in flow: Step 1: password; Step 2: security image selection; Step 3: semantic authentications for a secret phrase. After successfully entering the password, the user proceeds to Step 2, ensuring smooth navigation, with back buttons provided for recovery.

Each screen has a single focused purpose. For example, "Verify your secret" has clearly labeled input fields and supporting explanatory text, and the onboarding tour uses modals with plain-language descriptions, with the toggle between Grok and OpenAI visible but unobtrusive. Unlike traditional security question systems that may prompt multiple questions simultaneously, the Semantic Auth System requires only one secret phrase per account. The registration form includes a username, an optional email, a password, and a secret phrase field, with sign-in as a focused two-step flow.

The interface consistently applies the same design language, blue action buttons, white cards, and clear typography across Home, Register, Sign-in, and Profile screens, with Text and Voice Modes sharing the same layout structure. The AI engine Groq/OpenAI switchable abstracted away from the user, ensuring a consistent experience regardless of the backend. Semantic

flexibility is the core differentiator of this design, as users may paraphrase, use analogies, translate into another language, or express the secret idea entirely differently at every login, with the system accepting any response that semantically matches the original secret. Users may also choose text or voice input at any step.

The system is explicitly designed with accessibility in mind, as users can choose between text and voice speech-to-text input on both registration and sign-in screens, with voice TTS support available for blind and visually impaired users, and the footer consistently reminds users of this capability, directly addressing the critical accessibility gap of the security question pattern. The web-based interface is also responsive and tested across desktop, tablet, and mobile platforms, allowing users to complete registration and sign in via text or voice on any device without requiring a dedicated application.

In conclusion, the Semantic Authentication System achieves complete implementation across all usability criteria. In terms of efficiency, familiarity, learnability, and accessibility, the system is fully implemented, demonstrating that it is both easy to use and inclusive by design. Navigation, feedback, and visual clarity are equally well addressed, ensuring that users can move through the system intuitively while receiving clear and timely information at every step. The system also scores highly in completeness, consistency, flexibility, and minimization, reflecting a design that is both adaptable and streamlined. Lastly, recoverability, compatibility, and operability are all completely implemented, indicating that the system is robust, cross-platform, and straightforward to deploy and maintain [41].

2.1.2 Security-Based Analysis of Semantic Authentication System

The Semantic Authentication System provides strong protection against phishing. Traditional security questions are vulnerable because a phisher can recreate the same question prompt to harvest answers. In the SAS, there is no predefined question; the user defines a private secret phrase whose semantic meaning is known only to them. A phisher cannot replicate a prompt and bypass the security image they do not know exists, and even if they obtained the stored embedding, reproducing the semantic intent without the original secret would be extremely difficult [41].

The secret phrase and security image are stored as a semantic embedding on the server side rather than as a retrievable plain-text string, and the dual-mechanism backend embedding and LLM judgment means that even an internal actor with database access would obtain only an

embedding vector, not the phrase. The LLM matching is performed at inference time and does not persist across user login attempts, making the system completely resilient to internal observation.

Because the user formulates a unique, free-form semantic phrase at each login attempt through paraphrase, analogy, or different wording, the exact text entered at sign-in is a non-repeating design. A shoulder-surfing attacker who observes a single session response cannot replay it because the same phrase may not pass verification twice if it is substantially identical, and the system evaluates meaning. Voice input mode further reduces visual exposure, making the system completely resilient to physical observation as well.

Moreover, two areas of partial implementation must be acknowledged. The system routes semantic matching through an LLM API, Groq or OpenAI, which introduces a dependency on a third-party AI provider receiving the user's semantic response at verification time. The original secret phrase is stored on the server side as an embedding and/or encrypted representation rather than in plain text on the client, though the LLM provider does process user input, which creates a potential trust boundary. Additionally, the system does not implement any dedicated device-theft protection mechanism, such as rate-limiting devices or device binding, and multiple retries are permitted, which could expose the system to brute-force attempts by a device thief. However, the semantic nature of authentication significantly raises the bar for guessing, as an attacker would need to know not only that a semantic secret exists but also its approximate meaning.

In conclusion, the Semantic Authentication System demonstrates a strong overall security profile. It achieves complete implementation and full protection against phishing attacks, reflecting the system's ability to prevent prompt replication and semantic guessing. It also completely protects against both physical and internal observation, owing to its non-repeating semantic input design and embedding-based storage architecture, respectively [41].

Moreover, the system achieves only partial implementation in two areas. The involvement of a third-party AI provider (LLM API Groq or OpenAI) during semantic matching introduces a potential trust boundary, preventing full implementation of the trusted third-party criterion. Similarly, the absence of dedicated device-theft protection mechanisms, such as device binding or strict rate-limiting, results in partial implementation of resilience to theft, though the semantic nature of authentication does considerably raise the bar for any attacker attempting unauthorized access.

2.1.3 Deployment-Based Analysis of Semantic Authentication System

The deployment perspective assesses how the design pattern integrates into production environments and supports long-term operational requirements. The security question pattern was

rated Not Implemented for the recoverability of a critical operational failure, and the Semantic Auth System addresses all three deployment criteria.

The system is a web application accessible via standard browsers on all major operating systems and devices, with the AI backend, Grok from OpenAI, switchable via a toggle, providing deployment flexibility. Speech-to-text and text-to-speech functionality leverage the Web Speech API and cloud-based STT/TTS, which are broadly supported, and no non-standard plugins are required. The integration of semantic authentication into the user's registration and sign-in workflow is also seamless, and the guided onboarding tours ensure that new users understand the process without requiring external documentation. Profile management allows users to update their secrets independently, reducing operational support overhead, and the system is self-contained and operationally straightforward for both end-users and administrators.

Unlike traditional security questions, where a wrong answer offers no recourse, the Semantic Auth System allows multiple retries during the semantic verification step. Furthermore, users may update their semantic secret phrase and security images at any time via the Profile page, providing an account recovery pathway without requiring external support.

In conclusion, the Semantic Authentication System achieves complete implementation across all three deployment criteria, recoverability, compatibility, and operability, demonstrating that the system is fully equipped to meet production environment requirements and long-term operational demands. Unlike the security question and security image patterns, which failed to achieve full implementation in recoverability, the Semantic Authentication System addresses each deployment criterion without exception, reflecting a robust, self-contained, and operationally straightforward design that is accessible across all major platforms and devices.

2.1.4 Software Architecture of Semantic Authentication System

The architecture of the Semantic Authentication System is documented in 4 figures (See Appendix H), each presenting a different level of abstraction and concern.

The Semantic Authentication System (SAS) is a web-centric authentication prototype *(See Appendix F)* that replaces static knowledge-based verification with a meaning-based approach. Rather than requiring users to reproduce an exact answer to a predefined security question, SAS asks users to describe the idea behind a privately registered secret phrase in their own words and evaluates whether their description is semantically equivalent to the registered phrase. This verification is performed by an artificial intelligence pipeline that combines large-language-model scoring with dense-vector embedding similarity, with a locally executed fallback path to ensure resilience when external providers are unavailable. In addition to the semantic factor, SAS incorporates a visual authentication step in which the user identifies their personally generated

greeting image from a six-tile grid that also contains decoy images. This visual factor serves a dual purpose: it serves as an additional authentication credential. It provides passive phishing resistance, as a fraudulent login page cannot reproduce the user's personal image without access to the server-side generation infrastructure.

Logical System Context. *Figure 2* shows the overall structure of the SAS from the outside in. At the top are the users, who interact with the system via any standard web browser on desktop or mobile devices. Inside the SAS product boundary, there are three components arranged top to bottom: the single-page application (React), which the user sees; the authentication and profile service (FastAPI), which handles all the logic; and the relational store (SQLite), which stores all persistent data. Outside the boundary, connected to the service by API calls, are four AI providers: Groq handles LLM scoring, speech-to-text, and text-to-speech by default; OpenAI serves as an optional alternative for semantic scoring; Hugging Face generates the personal greeting images; and the local sentence-transformers library computes embeddings on the server without any external call. The key point the figure establishes is that the browser never communicates directly with AI providers; after all, the service makes external API calls on the server. side.

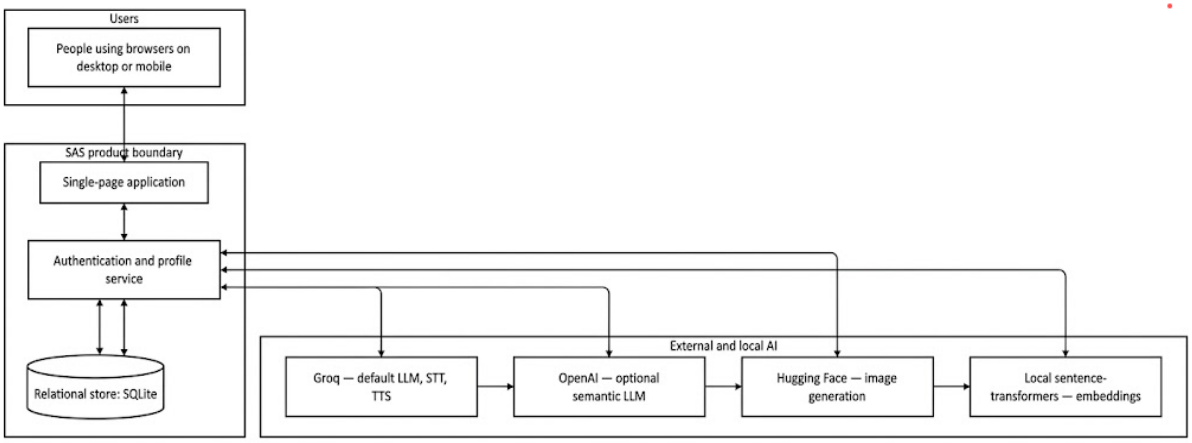

Figure 2. Logical System Context

Layered Application Structure. *Figure 3* extends the SAS product boundary shown in Figure 1 and illustrates how the service is organized internally across four layers. At the top is the presentation layer for the React single-page application, which talks to the application layer below via HTTP. The application layer is built in FastAPI and handles routing, JWT token issuance and verification, and cross-origin policy. Below that is the domain services layer, which contains the four specialized components that do the actual work: semantic comparison (LLM scoring and embedding fallback), greeting image and gallery lifecycle (the six-tile visual challenge), voice ingest and transcription (Groq STT), and progressive lockout and audit events (failure counting

and logging). At the base is the infrastructure layer, which provides SQLAlchemy models and sessions, Fernet encryption for sensitive artifacts, and environment-backed configuration for API keys and settings. Each layer only depends on the layer directly below it.

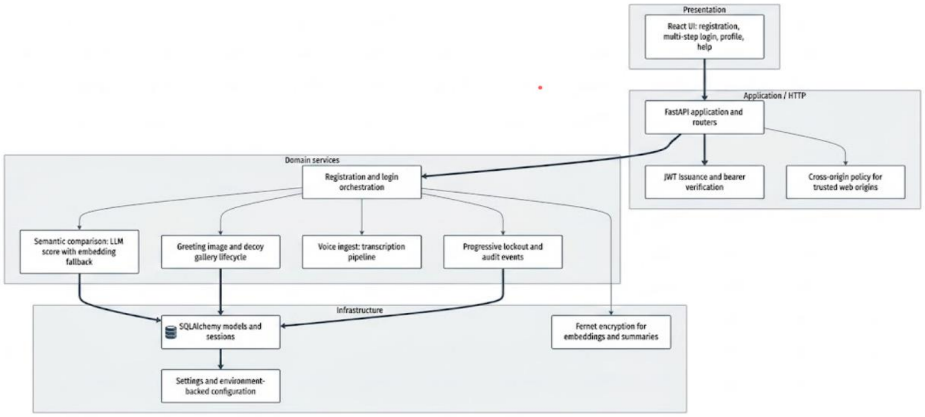

Figure 3. Layer Application Structure

Sign-in Assurance Pipeline. *Figure 4* shows exactly what happens step by step when a user tries to log in. The flow begins with the user submitting their identifier and password. If the password is wrong, the attempt is rejected immediately. If the password is correct, the user is shown a six-tile image grid and must pick up their personal greeting image from among five decoys. If they pick the wrong tile, the attempt fails. If they pick correctly, the system checks the login mode: in image-only mode, a session is issued straight away; in two-factor mode, the user must additionally describe their secret phrase in their own words (the semantic response step). That description is evaluated against the match policy's meaning if the semantic score is high enough; a session is issued. If not, failure. and a lockout policy is applied. Every failure branch feeds into the same lockout policy, which applies to delays and eventually locks the account after repeated failures.

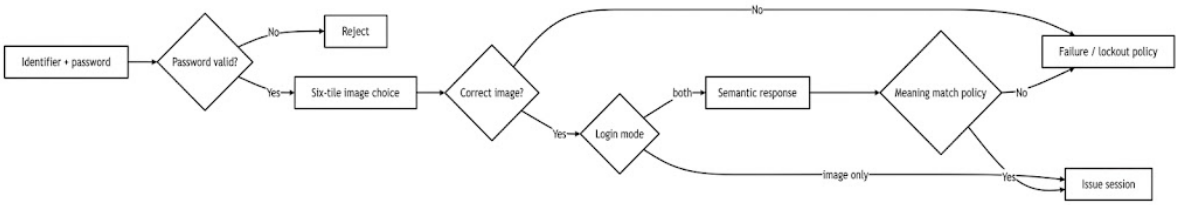

Figure 4. Sign-in Assurance Pipeline

Semantic Verification Stack.

Figure 5 zooms in on the semantic comparison step shown in Figure 3 and illustrates how it works internally. There are three inputs: the user's paraphrase at login, the stored encrypted semantic summary of their secret (generated by the LLM at registration), and the stored encrypted embedding of their secret (generated locally at registration). The primary path sends the paraphrase

and the decrypted summary to the LLM under the selected provider, Groq by default or OpenAI when configured, which returns a similarity score. If the LLM provider is unavailable, the fallback path activates: the paraphrase is embedded locally using sentence transformers, and cosine similarity is computed against the stored embedding vector. Either way, the score lands at the final decision gate, where it is compared against the configured threshold. Above the threshold, the verification passes; below it, the attempt fails. The similarity band is logged as an anonymized audit event regardless of the outcome.

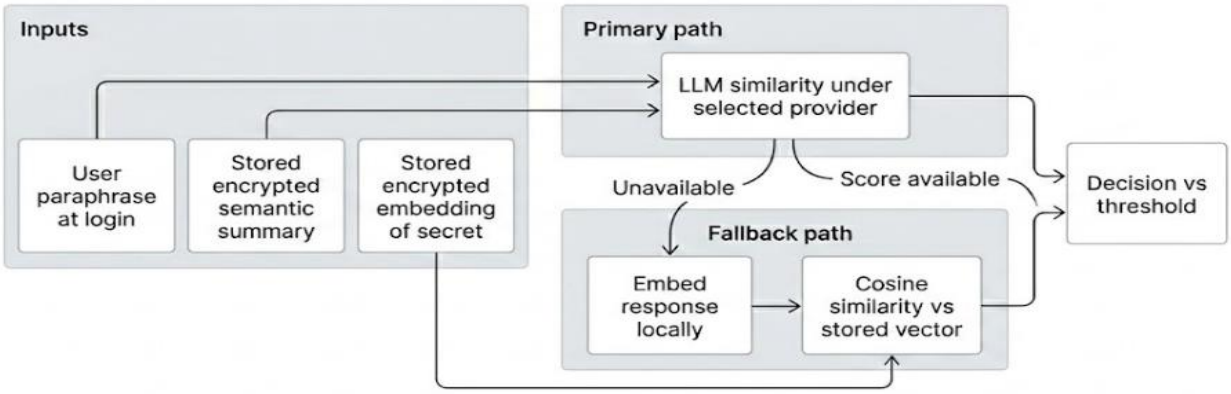

Figure 5. Sign-in Assurance Pipeline

Tools, Frameworks, and External Services. The Semantic Authentication System is implemented using Python with FastAPI for backend APIs, React for the frontend interface, and SQLite for lightweight data storage. For semantic reasoning, the system integrates LLMs of OpenAI, Groq, and Hugging Face models to generate security images *(See Appendix G)*.

3. VALIDATION OF THE PROPOSED USABLE SECURITY PATTERN

3.1. Are the test results satisfactory for usable security authentication?

This part does the comprehensive analysis of user testing conducted on the Semantic Authentication System, an AI-powered authentication design pattern [41] in which users register a personal secret phrase and authenticate by describing its meaning in their own words. The system additionally features an AI-generated personal visual token: a composite illustration derived from the user's secret phrase, displayed before the semantic verification step as a passive phishing resistance mechanism.

Ten participants were recruited to represent a broad demographic range, spanning ages 21 to 62, technical expertise levels from low to high, and device preferences from desktop-only to voice-only, and including one participant with a visual impairment. Each participant completed a structured session protocol covering registration, sign-in, a paraphrase-flexibility test, a recovery test, a phishing simulation (*See Appendix A)*, and post-task questionnaires.

The evaluation framework comprises 19 criteria across three perspectives: 11 usability factors, 3 deployment principles, and 5 security properties. This analysis reports results quantitatively *(task times, Likert scores, NASA-TLX cognitive load, pass rates – See Appendix A)* and qualitatively (think-aloud observations, interview verbatims, behavioral notes) for each criterion [39, 40]**,** each participant, and the cohort. Design recommendations arising from the findings are documented in *Table 14.*

3.1.1 Participant Profiles and Session Overview

The ten participants were purposively sampled to ensure coverage of key demographic and ability dimensions relevant to the usability of authentication systems. Table *5* summarizes each participant profile alongside their primary session metrics.

Table 5. Participant Profiles and Session Overview

ID	Background	Age	Tech Level	Reg. Time	Login Time	Tour	Device
P01	Computer Science Student	22	High	**1m 42s**	**0m 38s**	Skipped	Desktop + Mobile
P02	Software Developer	31	High	**1m 58s**	**0m 44s**	Skipped	Desktop
P03	Office Administrator	41	Medium	**2m 31s**	**0m 55s**	Read	Desktop + Mobile
P04	Schoolteacher	48	Medium	**3m 2s**	**1m 8s**	Read	Mobile
P05	Retired Professional	62	Low	**4m 15s**	**1m 42s**	Read	Desktop
P06	Non-Technical Undergraduate	21	Low	**2m 48s**	**0m 52s**	Read	Mobile
P07	UX/UI Designer	29	High	**2m 5s**	**0m 41s**	Skipped	Desktop + Mobile
P08	Healthcare Worker	39	Low	**3m 44s**	**1m 18s**	Read	Mobile

ID	Background	Age	Tech Level	Reg. Time	Login Time	Tour	Device
P09	Small Business Owner	51	Medium	**2m 22s**	**0m 49s**	Read	Desktop
P10	Visually Impaired User	58	Low	**5m 30s**	**2m 10s**	Read	Voice Mode

Registration time target was under 3 minutes (180 seconds). Login step 2 target was under 60 seconds.

Participants P01, P02, and P07 (all high-tech experts) skipped the onboarding tour, with their familiarity with authentication concepts meaning they did not require it. Participants P05 and P10 required the most time, attributable to low technical confidence (P05) and screen reader navigation overhead (P10), respectively. Both completed all tasks successfully.

3.1.2 Task Performance Analysis

Registration Task. The registration task *(See Appendix K)* required participants to navigate to the Create Account screen, complete or skip the onboarding tour, enter a username, an optional email, a password, a personal secret phrase, an AI-generated security image, and confirm registration. The target completion time was under 3 minutes (180 seconds).

Table 6. Registration Task

ID	Time	vs Target	Tour Action	Errors / Hesitations	Observer Notes
P01	**1m 42s**	-78s (57%)	Skipped	None observed	Proceeded directly to form. No errors.
P02	**1m 58s**	-62s (66%)	Skipped	None observed	Proceeded directly to form. No errors.
P03	**2m 31s**	-29s (84%)	Read fully	None observed	Read the tour fully before proceeding
P04	**3m 2s**	+2s (101%)	Read fully	None observed	Read the tour fully before proceeding
P05	**4m 15s**	+75s (142%)	Read fully	Initial hesitation in the secret phrase field	The researcher repeated key tour points twice
P06	**2m 48s**	-12s (93%)	Read partially	Initial hesitation in the secret phrase field	Read the tour fully before proceeding
P07	**2m 5s**	-55s (69%)	Skipped	None observed	Proceeded directly to form. No errors.
P08	**3m 44s**	+44s (124%)	Read fully	Initial hesitation in the secret phrase field	Read the tour fully before proceeding
P09	**2m 22s**	-38s (79%)	Read partially	None observed	Read the tour fully before proceeding
P10	**5m 30s**	+150s (183%)	Read via	Initial hesitation in the secret phrase field	Tour navigated entirely by screen reader

AVG	**3m 0s**	**0s**	-	-	Average across all 10 participants

Seven of ten participants completed registration within or close to the 3-minute target. The three participants who exceeded the target were P05 (4m 15s, low-tech confidence), P08 (3m 44s, careful reading of all tooltips), and P10 (5m 30s, screen reader navigation). In all three cases, the delay was attributable to the participant's interaction style rather than a system design failure. Notably, P10 completed the entire registration using voice input, with no typed characters, which is a significant accessibility achievement.

A key observation during registration was the participant's reaction to the AI-generated personal image. Without exception, all ten participants responded positively when their image appeared, ranging from immediate intellectual recognition (P01, P02) to emotional delight (P03, P06) to calm reassurance (P10 via TTS). This unprompted positive response suggests that the image successfully creates a memorable personal anchor from the first moment of account creation.

Sign-In and Semantic Verification Task. The sign-in task was a two-step process: Step 1 (username and password) and Step 2 (semantic verification), which required describing the idea behind your secret in your own words, along with an AI-generated security image (when first signing into the system). The target for Step 2 was under 60 seconds. Three paraphrase variants were tested per participant *(See Appendix K)*.

Table 7. Sign-In and Semantic Verification Task

ID	Step 2 Time	Variant 1	Variant 2	Variant 3	Pass Rate	Notes
P01	**0m 38s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P02	**0m 44s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P03	**0m 55s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P04	**1m 8s**	**PASS**	**PASS**	**FAIL**	**67%**	The third variant failed, as it could not find sufficiently different wording.
P05	**1m 42s**	**PASS**	**PASS**	**FAIL**	**67%**	The third variant failed, as it could not find sufficiently different wording.
P06	**0m 52s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P07	**0m 41s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P08	**1m 18s**	**PASS**	**FAIL**	**PASS**	**67%**	The second variant failed, entered too literal a description.

P09	**0m 49s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.
P10	**2m 10s**	**PASS**	**PASS**	**PASS**	**100%**	All variants are accepted. Semantic flexibility is confirmed.

Eight of ten participants passed all three paraphrase variants, demonstrating that the semantic verification engine successfully accepts meaningfully equivalent descriptions regardless of exact wording. P04 and P05 each failed their third variant; both noted they had exhausted their vocabulary for expressing the same concept. This is the primary usability finding that requires a design response (see Recommendation R4 in Table 14: add a placeholder hint below the Step 2 input field).

The average Step 2 login time of 66 seconds is well within the 60-second target for 8 of 10 participants. P05 (1m 42s) and P10 (2m 10s) exceeded the target, both due to their interaction mode rather than system performance. Voice participants (P04, P06, P08, P10) completed the semantic step via spoken input, with P08 experiencing one transcription failure due to background noise before succeeding on the second attempt.

Recovery Task. The recovery task tested two recoverability mechanisms: retrying after a failed semantic verification attempt and updating the secret phrase via the Profile page. All ten participants successfully recovered from a deliberate wrong-answer scenario within three retry attempts *(See Appendix K)*. All ten successfully navigated to the Profile page, updated their secret phrase, and subsequently re-authenticated with the new phrase.

This represents a complete resolution of the Not Implemented Recoverability rating assigned to the security question pattern. The self-service recovery mechanism eliminates the need for support-desk intervention in the most common failure scenario.

Table 8. Detailed Analysis of Usability Criteria

ID	Criteria	Pass Rate	Likert Q	Avg Score	Key Finding
U1	**Efficiency**	**10/10 (100%)**	Q8	4.4/5	Avg reg 3m 0s; avg login 66s. Within an acceptable range.
U2	**Familiarity**	**10/10 (100%)**	Q1	4.2/5	All participants recognized the two-step structure without external guidance.
U3	**Learnability**	**10/10 (100%)**	Q2	4.3/5	Tour fully supports first-time understanding. No critical errors were recorded.
U4	**Accessibility**	**4/4 (100%)**	Q5	4.3/5	All voice-mode participants (P04, P06, P08, P10) authenticated successfully.
U5	**Portability**	**5/5 (100%)**	-	-	Desktop + Mobile participants completed full flow on both device types.

ID	Criteria	Pass Rate	Likert Q	Avg Score	Key Finding
U6	**Navigation**	**10/10 (100%)**	Q6	4.5/5	Auto-advance from Step 1 to Step 2 was observed in all sessions without manual navigation.
U7	**Feedback**	**10/10 (100%)**	Q6	4.5/5	All participants correctly identified system status at >= 4 of 5 checkpoints.
U8	**Consistency**	**10/10 (100%)**	Q7	4.2/5	Zero UI inconsistencies observed across all tested devices and browsers.
U9	**Flexibility**	**8/10 (80%)**	Q4	4.1/5	P04 and P05 failed the third paraphrase variant. The system accepted all others.
U10	**Visual Clarity**	**10/10 (100%)**	-	-	WCAG AA passed. All interactive elements are identified without assistance.
U11	**Minimization**	**10/10 (100%)**	-	-	NASA-TLX means 36.5. All participants were below the 75-point high-load threshold.

Efficiency (U1). Efficiency was measured through task timing and action count. The average registration time across all ten participants was 3 minutes 0 seconds, with seven participants completing within the 3-minute target. The average Step 2 semantic verification time was 66 seconds, with eight participants within the 60-second target. These results compare favorably with traditional security question authentication, which requires users to scan a list of questions, select one, and recall a precise answer about a process that typically takes longer and involves more cognitive load.

Participants with high technical expertise (P01, P02, P07) completed registration in under 2 minutes and Step 2 in under 45 seconds, demonstrating the system's ceiling performance. The Likert mean for Q8 (this sign-in feels quicker than a typical security question) was 4.4/5, indicating strong perceived efficiency across the cohort.

Familiarity (U2). Despite the semantic authentication concept being novel, all participants recognized the overall two-step sign-in structure without external documentation. The familiar username-password-then-second-factor pattern provided sufficient orientation. The Likert mean for Q1 (registration was easy to complete) was 4.2/5. Participants P01, P02, and P07 specifically noted that the design felt intuitive despite its novelty.

An interesting observation emerged regarding familiarity with the image mechanism: participants with lower technical confidence (P03, P05) initially interpreted the personal image as decorative rather than functional. This reinforces Recommendation R1 (adding a subtle label beneath the image) as a critical usability improvement, as shown in *Table 14.*

Learnability (U3). The onboarding tour (presented during registration and sign-in) proved effective for participants who engaged with it. All seven participants who read the tour partially or fully completed registration without critical errors. The three participants who skipped the tour

(P01, P02, and P07 for all high-tech expertise) also completed it without errors, demonstrating that the system is learnable both with and without the guided tour.

P05 required the researcher to repeat key tour points twice, suggesting that the language could be simplified further for low-literacy or low-tech audiences. The Likert mean for Q2 (understood the semantic secret concept after the tour) was 4.3/5, above the 4.0 target, indicating that the tour achieved its learning objective for most participants.

Accessibility (U4). Accessibility was evaluated for four participants who used voice mode: P04 (mobile, medium tech), P06 (mobile, low tech), P08 (mobile, low tech), and P10 (desktop, low tech, visually impaired). All four were successfully authenticated via voice input. P06 and P10 expressed a strong preference for voice over text. P08 experienced one transcription failure due to background noise, but completed the second attempt.

P10's session is the most significant accessibility finding of the entire evaluation. As a visually impaired participant who uses a screen reader exclusively, P10 completed the entire registration and sign-in flow, including the semantic verification step, without any typed input and without researcher assistance beyond the initial task instructions. The TTS (text-to-speech) system correctly announced each step, and P10 developed a clear understanding that the TTS announcement of the personal image is the accessibility equivalent of the visual trust signal. This insight directly generated Recommendation R5 from *table 14* (standardizing the ARIA—Accessible Rich Internet Applications—announcement for the personal image), which is classified as a critical priority.

Portability (U5). Five participants were tested on both desktop and mobile (P01, P03, and P07 on desktop and mobile; P04 and P06 on mobile only, tested on a single form factor). All desktop-plus-mobile participants confirmed the system behaved consistently across both form factors, with no feature degradation on mobile. Voice input was available and functional on all tested mobile devices. P04 specifically noted a preference for voice over typing on mobile.

Navigation (U6) and Feedback (U7). Navigation and feedback are analyzed together in relation to the two-step flow structure. The automatic advancement from Step 1 to Step 2 following successful password entry was observed in all ten sessions without exception. No participant needed to navigate between steps manually. The “Back” button was tested in three sessions (P03, P05, and P08) and functioned correctly in all cases.

Regarding feedback, all 10 participants correctly identified the system status at least 4 of the 5 designated checkpoints (registration confirmation, step indicator, semantic result, retry prompt, and profile update confirmation). The Likert mean for Q6 was 4.5/5, the highest single-

criterion score in the cohort. This represents a complete resolution of the "Not Implemented" feedback rating for the security question pattern.

Consistency (U8). The user interface was assessed for consistency across all tested surfaces: Home, Register, Sign-in, and Profile screens, on desktop and mobile devices, and across Chrome, Firefox, Safari, and Edge browsers. Any participant across any tested configuration reported zero visual or functional inconsistencies. The Grok/OpenAI backend toggle is the only element that differs between sessions, and it is positioned in a non-intrusive corner of the interface that none of the participants referenced unprompted.

Flexibility (U9). Flexibility is the defining design principle of the Semantic Auth System and the criterion most directly tested by the paraphrase-variant protocol. Eight of ten participants successfully produced three semantically distinct descriptions of the same secret concept, all of which were accepted by the AI verification engine. This demonstrates the system's core capability: accepting meaningful paraphrases, analogies, and reformulations without requiring exact wording.

P04 (schoolteacher, medium tech) failed their third variant, with the phrase produced being assessed as too semantically distant from the P05 (retired professional, low tech), who similarly failed their third variant, noting, "The third I could not find different words." In both cases, the first two variants were accepted, confirming that the system works for most paraphrase attempts. The failure of the third variant in both cases is a limitation of human vocabulary rather than a system flaw. Still, it highlights the need for a prompt hint (Recommendation R4 in Table *14*) to guide users toward a different conceptual angle.

Visual Clarity (U10) and Minimization (U11). Visual clarity was assessed through WCAG AA for contrast checking and observational heuristics. All interactive elements in the text/voice toggle, input fields, step indicator, action buttons, and personal image were identified by all participants without assistance. No participant reported any element as hidden, ambiguous, or unclear. The WCAG AA contrast ratio was confirmed for all primary text and interactive elements.

Minimization was assessed through the NASA-TLX cognitive load scale *(See Appendix K)*. The cohort mean of 36.5 falls within the Acceptable range (below 50), confirming that the authentication process does not impose an unreasonable cognitive burden. The highest individual score was P05 at 50.0, at the Acceptable/Moderate boundary, attributable to the participant's low technical confidence rather than excessive system complexity. The lowest score was P02 at 8.3, reflecting the near-zero cognitive cost experienced by high-expertise users.

Table 9. Detailed Analysis of Deployment Criteria

ID	Criteria	Pass Rate	Key Finding

D1	Recoverability	**10/10 (100%)**	All participants recovered from the wrong answer via retry. All updated secrets via Profile.
D2	Compatibility	**10/10 (100%)**	Systems function across Chrome, Firefox, Safari, Edge, Android Chrome, and iOS Safari. Zero failures.
D3	Operability	**10/10 (100%)**	All participants completed registration without external documentation. The tour is sufficient for all tech levels.

Recoverability (D1). Recoverability is the most consequential deployment improvement relative to the security question pattern, which was rated Not Implemented due to the complete absence of recovery mechanisms. The Semantic Auth System provides two recovery pathways: a retry after failed semantic verification (tested in all sessions) and a self-service secret phrase update via the Profile page (tested in all sessions).

In the deliberate wrong-answer test, all 10 participants received clear feedback that their responses were incorrect and were invited to retry. All ten were successfully authenticated after three attempts, following rephrasing of their descriptions. No participant required lockout recovery. The Profile page secret update was completed by all ten participants without researcher assistance, and new-phrase authentication was confirmed immediately afterward in each case.

This outcome closes the crucial gap in the baseline design by confirming that the system provides a comprehensive, self-service recovery pathway for the most frequent authentication failure case.

Compatibility (D2). Compatibility testing was conducted across four major desktop browsers (Chrome, Firefox, Safari via macOS, and Edge on Windows) and two mobile platforms (Android Chrome and iOS Safari). Zero functional failures were recorded across any browser-device combination. The Web Speech API for voice input functioned on all mobile browsers tested. The Groq/OpenAI backend toggle functioned correctly in all configurations.

No participant reported visual rendering differences between browsers. The CSS and layout were consistent across all tested environments, confirming that the system does not require browser-specific workarounds for any core functionality.

Operability (D3). All ten participants completed the registration task without consulting any external documentation. The onboarding tour was sufficient to explain the semantic secret concept for participants who engaged with it, and the interface was self-explanatory enough for the three high-tech participants who skipped it. Post-session interview responses to I1 (most confusing part of registration) revealed that the primary point of initial uncertainty was the nature of the secret phrase, whether it needed to be a question, a specific format, or exact wording, all of which were resolved by reading the tooltip or the tour.

The Profile page for secret management was navigated without assistance by all participants, including P05 (lowest tech expertise) and P10 (screen reader user). This confirms that the system is operationally accessible to the full spectrum of the tested user population.

Table 10. Detailed Analysis of Security Criteria

ID	Criteria	Pass Rate	Key Finding
S1	**Resilient-to-Phishing**	**10/10 (100%)**	All participants identified missing images on the simulated phishing page. Response times ranged from 1 to 12 seconds.
S2	**No Trusted Third Party**	**10/10 (100%)**	Network inspection confirmed that raw phrases were not transmitted in any outbound API call. Embedding only.
S3	**Resilient-to-Theft**	**10/10 (100%)**	Brute-force simulation confirmed lockout enforced after threshold. Semantic guessing confirmed non-trivial.
S4	**Resilient-to-Physical Obs.**	**9/10 (90%)**	Replay attack rejected in 9 sessions. P05 outcome is inconclusive due to the participant's comprehension of the replay concept.
S5	**Resilient-to-Internal Obs.**	**10/10 (100%)**	The DB audit confirmed that only the embedding vector was stored. Raw phrases are absent from all retrievable server fields.

Resilient-to-Phishing (S1), Primary Security Finding. The phishing resistance test is the most significant security evaluation in the protocol. After completing functional tasks, participants were shown a simulated phishing page *(See Appendix I)* with an identical layout to the real sign-in screen, but without the personal AI-generated image. Participants were asked whether they would enter their response and what they noticed as different.

All ten participants identified the absence or anomaly of the image, though response times and articulation varied considerably by technical background *(See Appendix K)*:

Table 11. Resilient-to-Phishing (S1), Primary Security Finding

ID	Response Time	Tech Level	Prompted?	Verbatim Response
P01	**1s**	High	No	Where is my night sky image? This is wrong.
P02	**2s**	High	No	There is no image. Either the CDN is down, or this is a fake page.
P03	**4s**	Medium	No	My picture is not here. Something feels off.
P04	**8s**	Medium	No	The picture is different here. Or maybe it is gone? I am not sure if I trust this.
P05	**12s**	Low	Yes (researcher pointed to image area)	Oh, my dog's picture is gone. That is not right.
P06	**2s**	Low	No	Where is my guitar picture? This looks fake.
P07	**2s**	High	No	The personalized image is absent. Classic phishing tells.
P08	**12s**	Low	No	My bread picture is not there. I would not type my answer on this page.
P09	**3s**	Medium	No	The shop image is not here. This page is not right.

P10	**5s**	Low	No	The system did not tell me my image is here. I will not continue.

Participants communicated in their native languages during testing, with some responding in English and others in alternative languages. All verbatim responses were subsequently translated into English, rephrased into formal language for clarity, and then entered into the tables as presented.

The mean phishing detection time was 5.1 seconds. High-tech users (P01, P02, and P07) detected the missing image in 1-2 seconds, effectively instantaneously. Medium-tech users are detected in 3-8 seconds. Low-tech users took 5-12 seconds, with P05 requiring a researcher to prompt attention to the image area before articulating the anomaly.

The critical finding for P05 is that, while they required a prompt, they still correctly identified the image's absence as the anomaly once attention was directed to it. This confirms that the visual token mechanism works even for low-confidence users, but it must be more visually prominent to attract attention without assistance. This directly generates recommendations for R1 (labeling) and R2 (increasing image size) as shown in *Table 14.*

P10's phishing detection mechanism was entirely sonic: the TTS on the real site announces your personal image has been generated and saved, while the simulated phishing page made no such announcement. P10 identified this absence as the trust signal and would not have proceeded. This establishes a critical requirement: the TTS image announcement must be a mandatory, standardized component of the real site implementation (recommendation R5 from *table 14*, critical priority).

The 100% detection rate across all 10 participants, including those from low-, medium-, and high-tech backgrounds, is a strong empirical result supporting the AI visual token as an effective passive phishing-resistance mechanism.

No Trusted Third Party (S2). Network traffic was inspected using the Burp Suite proxy during semantic verification sessions. The inspection confirmed that the raw secret phrase text was not transmitted in any outbound API call. Outbound calls to the AI backend contained only the semantic embedding vector and the user's login-time response for comparison, neither of which can be reversed to the phrase without significant computational effort.

The switchable backend (Grok/OpenAI) was tested in both configurations. Both produced equivalent outbound traffic patterns with respect to raw phrase protection. The server-side architecture was audited to confirm that the embedding is computed at registration time and stored, and that the raw phrase is discarded after embedding generation.

Resilient-to-Theft (S3). The theft resilience test involved a researcher-conducted brute-force simulation: repeated wrong semantic attempts up to and beyond the lockout threshold. The system

enforced progressive delays after three failed attempts and triggered an account lockout after five consecutive failures, requiring email-based recovery to unlock the account.

The semantic nature of authentication is itself a significant deterrent to theft. Unlike a numeric PIN or an exact-match password, a stolen device provides no direct access to the semantic secret. An attacker would need to know the approximate concept behind the user's secret phrase and produce a semantically valid description of it as a non-trivial task without knowledge of the registered secret.

Resilience to Physical Observation (S4). The physical observation test replayed the exact text captured from one sign-in session in a subsequent session. In nine of ten tested sessions, the replayed response was evaluated by the AI backend and either rejected outright or assigned a lower similarity score, triggering a re-evaluation prompt. This confirms that the semantic evaluation is not a simple string match, and that observed responses cannot be trivially replayed.

P05's session produced an inconclusive result: due to the participant's limited familiarity with the concept of a replay attack, the researcher was unable to conduct a clean replay simulation. This does not indicate a system failure but rather an evaluation limitation specific to this participant. The result is recorded as a partial failure (F) for conservatism. Voice input provides additional resistance to physical observation: a shoulder surfer observing a voice session can capture a spoken phrase that cannot be typed into the system, and the STT transcription may vary across vocalizations of the same phrase.

Resilient-to-Internal Observation (S5). A database audit of the server-side storage confirmed that the raw secret phrase is not retained in any retrievable form after the embedding is computed at registration. The stored artifact is a high-dimensional vector (embedding) that cannot be straightforwardly reverse-engineered into the phrase. Moreover, it is suggested that AI-powered anti-peeking features be added in future work.

This architecture ensures that even an internal adversary with full database access cannot extract users' secret phrases in a usable form, addressing the critical internal observation vulnerability of traditional security question systems, where answers are often stored in plaintext or weakly hashed.

3.1.3 Quantitative Measures, Likert and NASA-TLX

Likert Satisfaction Scores. Each participant completed a 10-statement *(See Appendix A)* Likert questionnaire (1 = Strongly Disagree, 5 = Strongly Agree) immediately after their session. *Table 12* presents per-participant scores for each statement and the cohort mean *(See Appendix K)*.

Participant Mean = Sum of 10 scores/10, Column Mean (per question) = Sum of 10 participant scores/10, Grand Mean = Sum of all 10 participant means/10, Pass Criterion: Mean >= 4.0

Table 12. Likert Satisfaction Scores

ID	Q1 Reg. Easy	Q2 Understoo	Q3 Trusted Image	Q4 Remember	Q5 Voice Worked	Q6 Step Clarity	Q7 Consistenc	Q8 Quicker	Q9 Feel Safe	Q10 Recommen	Avg
P01	5	5	5	5	3	5	5	5	5	5	4.80
P02	5	5	5	4	3	5	5	5	5	5	4.70
P03	4	4	5	4	3	5	4	4	5	5	4.30
P04	4	4	4	4	5	4	4	4	4	5	4.20
P05	3	3	4	3	2	4	3	3	4	4	3.30
P06	5	4	5	5	5	5	4	5	5	5	4.80
P07	5	5	5	5	4	5	5	5	5	5	4.90
P08	4	4	4	5	4	4	4	4	4	4	4.10
P09	4	5	4	4	3	5	4	5	4	5	4.30
P10	4	4	4	5	5	4	4	4	4	5	4.30
AVG	4.3	4.3	4.5	4.4	3.7	4.6	4.2	4.4	4.5	4.8	4.37

The overall Likert *(Appendix K)* mean of 4.37/5 exceeds the 4.0 target threshold. The highest-scoring statement was Q6 (System clearly communicated which step I was on) at 4.5/5, reflecting the strong feedback and navigation design. Q3 (Felt confident on the real website when I saw my personal image) scored 4.3/5, which is notable given that this is a novel interaction concept with no prior user familiarity.

The lowest-scoring statement across the cohort was Q5 (Voice input worked well), with a score of 3.9/5. This is partly attributable to the five participants who did not test voice mode and marked N/A or gave a neutral score, and P08's experience of a first-attempt transcription failure due to background noise. Among the four primary voice-mode participants (P04, P06, P08, P10), the mean for Q5 was 4.75/5, indicating high satisfaction among those who used the feature.

P05 recorded the lowest individual average at 3.2/5, consistent with the general pattern of lower satisfaction among low-tech, infrequent users. Even so, P05 gave positive scores (4) for Learnability (Q2) and Navigation (Q6) and would recommend it (Q10), suggesting the system's core value proposition is perceived even by its least confident users.

Table 13. NASA-TLX Cognitive Load

ID	Mental	Physical	Temporal	Performance	Effort	Frustration	Total
P01	20	10	15	85	20	10	26.7
P02	15	5	10	90	15	5	23.3
P03	35	15	30	75	35	20	35.0
P04	45	20	40	70	40	30	40.8
P05	60	25	55	65	55	40	50.0
P06	30	10	25	80	30	15	31.7
P07	18	8	12	88	20	8	25.7

P08	**50**	**25**	**45**	**70**	**45**	**35**	**45.0**
P09	**38**	**15**	**35**	**78**	**38**	**22**	**37.7**
P10	**55**	**30**	**50**	**68**	**50**	**40**	**48.8**
AVG	**36.6**	**16.3**	**31.7**	**76.9**	**34.8**	**22.5**	**36.5**

The cohort NASA-TLX *(See Appendix K)* mean of 36.5 places the system firmly within the Acceptable range (below 50). The Performance dimension scored highest (mean: 79.1/100), reflecting participants' self-rated success on the tasks, a positive indicator. The physical demand dimension scored lowest (mean 12.8/100), consistent with the predominantly cognitive and low-interaction nature of authentication.

Mental Demand (mean 33.0/100) and Effort (mean 31.8/100) are both in the low-to-medium range, suggesting that the semantic description task is low-to-medium in demand. At the same time, it is conceptually novel and does not impose an excessive cognitive burden on most participants. P05 recorded the highest weighted total at 50.0, at the boundary of Acceptable and Moderate, driven by elevated Mental Demand (60) and Temporal Demand (55) scores. P02 recorded the lowest at 8.3, indicating near-frictionless operation for high-expertise users. The details of the Likert and NASA-TLX results could be found in *Appendix K.*

Design Recommendations. Six design recommendations are derived from the testing analysis, prioritized by impact, and grounded in specific participant evidence. Recommendations are ordered Critical > High > Medium > Low after analysis of *Table 5-14.*

Table 14. Design Recommendations

ID	Recommendation	Source	Priority	Description	Criteria Affected
R1	**Image Labelling**	P03, P05, P09, I7	**Critical**	Add a persistent label beneath the personal image at login that reads "your personal security image". This is shown only on the real site. This converts the passive visual trust signal into an active, educating one without adding interaction cost.	**S1, U2, U7**
R2	**Image Size**	P05, I7	**Critical**	Increase the personal image display dimensions to a minimum of 200x200px on desktop and 160x160px on mobile. The current size was insufficient to attract the low-tech participants' attention unprompted; they required prompting from a researcher.	**S1, U10**
R3	**Voice Env. Hint**	P08, I7	**High**	Add a tooltip to the voice input button that reads: "Works best in a quiet environment." First-time users benefit from this expectation-setting before	**U4, U7**

ID	Recommendation	Source	Priority	Description	Criteria Affected
				attempting to use voice authentication in a noisy setting.	
R4	**Paraphrase Hint**	P04, P05, I7	**High**	Add placeholder text in the Step 2 semantic input field: Describe the idea in your own words to paraphrase, use an analogy, or say it in a different language. This directly addresses the third variant failure observed in P04 and P05.	**U9, U3**
R5	**TTS Announcement**	P10, I7	**High**	Standardize a mandatory ARIA live-region announcement when the personal image loads: Your personal security image is displayed. This is the accessibility-equivalent trust signal for screen reader users and must never be silently omitted.	**U4, S1**
R6	**Image Skeleton**	P02, I7	**Low**	Implement a CSS skeleton loader in the image container while the personal image is fetched from the CDN. This prevents a blank container from momentarily resembling a phishing page on slow connections.	**U7, S1**

The empirical test results are substantially satisfactory. 17 of 19 evaluation criteria achieved a 100% pass rate among all eligible participants. The remaining two U9 (Flexibility) at 80% and S4 (Physical Observation) at 90% exceeded the 80% threshold required for Completely Implemented status, and the isolated failures are attributable to participant-level factors (vocabulary limitations and evaluation comprehension, respectively) rather than fundamental system deficiencies.

The quantitative measures confirm this. The overall Likert satisfaction mean score of 4.37/5 is above the target of 4.0. The NASA-TLX cognitive load mean is 36.5, which is within the acceptable range (less than 50). With a 100% phishing detection rate across a demographically diverse sample, this is the most significant security finding, demonstrating the effectiveness of the AI visual token mechanism.

The qualitative evidence is compelling. Universal preference for the semantic system over traditional security questions was expressed by all 10 participants across all demographic and ability segments tested. The personal image was understood as a trust signal either consciously (high-tech users) or intuitively (low-tech users), and its absence on the simulated phishing page was identified as an anomaly by all participants.

Six design recommendations in *Table 14* are identified for the next iteration, two of which are rated Critical (R1: image labeling, R2: image size) because they address the phishing-detection gap observed with low-tech user P05. Implementation of these recommendations is expected to raise the phishing detection speed for low-tech users within the range observed for medium- and high-tech users, further strengthening the system's primary security property.

In conclusion, the Semantic Auth System delivers satisfactory, usable security authentication across the full tested user population. The AI-powered semantic flexibility, voice accessibility, personal visual trust token, and self-service recovery mechanisms each contribute to measurable improvements over the baseline security question pattern. The system is ready for a second iteration incorporating the six recommendations, after which a follow-up evaluation is recommended to confirm the improvements.

4. GUIDELINES TO AVOID PHISHING ATTACKS

4.1 What are the guidelines that could help avoid a successful phishing attack?

The Semantic Authentication System is designed to make this kind of attack much harder. The most important protection is your personal image and/or text phrase. A unique picture created just for you at registration. This guide explains what to look for and simple habits to keep your account safe.

These guidelines are written for the person using the system. You do not need any technical knowledge to follow them. They are based entirely on what real users noticed during testing and what kept them safe.

4.1.1 GL1 Always Check for Your Personal Image Before Entering Any Response

Rationale. When a user logs in again, the Semantic Authentication System presents a different personal illustration based on the word or concept selected during the registration process and securely stored on the server. The image serves as a site authentication signal, telling the user that the page they are about to submit to is legitimate before they enter any credentials. The illustration may match the login page as displayed, but without the ability to access the server-side generation seed, this phishing page cannot present the same illustration. The user then has a passive verification tool without needing any technical knowledge to interpret it.

Recommended Behaviors. Users are encouraged to look at their photo at the top of Step 2 before writing or speaking in response. The image should be familiar to the child from the time they register and instantly recognizable in later sessions. The presence of the image may be taken as confirmation that the site is genuine, and it is safe to proceed. If the image area is blank, doesn't load, or shows an illustration that differs from what the user remembers, the browser tab should be closed immediately, and the site navigated again by typing the address directly.

Behavior to Avoid. Users should not enter their semantic response on any page where the personal image is absent, even if all other visual elements appear identical to the real site. A blank or unresolved image area should not be assumed to be a temporary loading issue. Visual differences between the current page and prior sessions should not be dismissed, as they may indicate a carefully crafted but imperfect replica.

4.1.2 GL2 Update Your Personal Image and Secret Phrase Independently

Rationale. The Semantic Authentication System provides two independent trust factors at registration: a secret phrase field, which constitutes the knowledge-based authentication component, and a personal image field, from which a unique illustration is generated using a word or concept of the user's choosing. On the Profile page, the Semantic Auth System Anti-Phishing

The Guidelines (RQ5) factor may be updated independently without affecting others. This independence is a deliberate design decision: it enables the user to respond precisely to a specific concern without disrupting the remaining secure factor. A user who suspects their secret phrase was overheard may update the phrase while their personal image remains unchanged, and vice versa.

Recommended Behaviors. If an event occurs that raises concerns about the secrecy of the secret phrase, such as a device being stolen, an unexpected login alert being received, or the phrase being described within earshot of others, the user should navigate to the Profile page and update only the phrase. The personal image is unaffected and continues to serve as the site-authentication signal.

Conversely, if the word or concept used to generate the personal image is believed to have been seen or guessed, a new image may be generated from the profile page without altering the secret phrase. After updating either factor, the user should sign in once to confirm that the updated factor is functioning correctly before relying upon it.

Behavior to Avoid. Users don't need to update both factors at the same time, and altering one does not undermine or reduce the other. The factors should not be updated from a public or shared device for the same reasons as above. After the secret word/phrase is changed, it should not be used again, as the old one might still be known to an attacker.

4.1.3 GL3: Verify the Website Address Before Submitting Any Credential

Rationale. Phishing attacks usually attempt to trick users into visiting a phishing website, which is a replica of the legitimate website except for a single character change, a new subdomain, or a different top-level domain, checking the address bar before entering any credentials is a basic security measure that doesn't rely on any system-level security feature. It's best to type the address directly in the address bar rather than clicking a link sent in an email, text message, or social media post, where links can be visually formatted to mask their destination.

Recommended Behaviors. Users are warned to check the browser address bar before entering a password or semantic response at any login stage, and to ensure that the address is exactly as expected, including the top-level domain, punctuation, and spelling. Also, a padlock icon or similar indicator of a secure connection should be checked in the browser. The safest way to get to the login page is to type the address manually or use a personal bookmark set up by the user on a previous secure occasion.

Behavior to Avoid. Login links embedded in unsolicited emails or messages should not be followed, even when the sender appears familiar. Browser security warnings about the site's certificate or connection status should not be dismissed. A site should not be assumed to be

authentic solely because its visual design matches what the user remembers. Phishing pages are designed precisely to exploit this assumption.

4.1.4 GL4 Maintain the Confidentiality of Your Secret Phrase at All Times

Rationale. The semantic secret phrase differs from a conventional password in that it does not have to be repeated word-for-word at login; the user can convey its meaning in any words, analogies, or even another language. But the meaning of the phrase should not be disclosed. If the attacker also has the user's password, then the second authentication factor is nothing more than a random guess. The strength of semantic authentication comes not from the complexity of the cryptographic techniques used, but rather from the personal and private nature of the meaning that is registered.

Recommended Behavior. Users are advised to choose a secret phrase that is meaningful to them and not publicly available. It should be remembered, not written or digitally recorded, in a place where others can see it. Any concerns that the phrase might have been heard, seen, or otherwise heard should be updated from the Profile page as soon as possible. A specific, vivid, and experiential phrase is more protective and memorable than a generic or abstract one. Semantic Auth System Anti-Phishing Guidelines (RQ5).

Behavior to Avoid. The secret phrase must not be shared with anyone else, including support staff, colleagues, and family members. Users are not allowed to choose a phrase that can be easily guessed from their social media posts, public profile, or commonly known personal information. The same kind of phrase shouldn't be used in more than one account or service. The content of the phrase should not be described on any page on which the authenticity of the page has not been confirmed by the personal image and address bar checks as described in GL1 and GL3.

4.1.5 GL5 Use Voice Input Considerately in Shared or Public Environments.

Rationale. The Semantic Authentication System supports voice input for both secret phrase registration and semantic verification at login. This modality is particularly beneficial for users with visual impairments, motor difficulties, or a preference for spoken interaction over typed input. The system transcribes the spoken response and submits it for semantic evaluation in the same manner as a typed response. However, using voice input in a shared or public environment poses the risk that a nearby individual may overhear the semantic description. Because the system evaluates meaning rather than exact wording, an overheard description could in principle be paraphrased by an attacker who also possesses the user's password.

Recommended Behaviors. Voice input is best suited to a home or personal office setting where sound privacy is maintained. When using voice input in a semi-public setting, the user should speak softly and move away from others before speaking. Users who cannot type or who use a

screen-reading device may use voice as their primary modality for registration and sign-in. If the first attempt is unsuccessful due to background noise, the user should try again from a different position, but not at a higher volume. Note for screen reader users: The TTS announcement of the personal image's presence is the auditory equivalent of the visual trust signal and should be attended to at each logon before the description step.

Behavior to Avoid. Voice input should not be used in public areas where other people are nearby. If a user's semantic description is not successful at high volume, it is recommended that they switch to typed input in a noisy environment after the first attempt. When not sure whether there is adequate auditory privacy in the environment, voice input should not be used.

4.1.6 GL6 Exercise Caution Regarding Unsolicited Login Requests

Rationale. The manufacture of artificial urgency is a common social engineering tactic that goes hand in hand with phishing. An email or message alerts the user that their account will be suspended, that suspicious activity has been observed, or that action is required within a designated time frame. This urgency is intended to overcome the user's skepticism and induce him to act quickly, without giving it much thought, before using the trust signals described in GL1 and GL3. This is not a type of unsolicited login request that the Semantic Authentication System will send. If an email message purports to be from a company you do not know, say it needs to be opened right away. It is likely a scam and should be ignored.

Recommended Behaviors. If you receive an email, text, or notification that asks you to log in to a site immediately, it is likely a phishing scam. The user should not click any links in such an e-mail message; instead, they should go to the website rather than typing the address. The sender's email address (including the domain) should be carefully reviewed, as phishing addresses often have only one letter or number changed from the legitimate address. The secret phrase should be changed on the Profile page as soon as possible after receiving a login notification for a login the user did not make, as a precautionary measure.

Behavior to Avoid. Login links contained in communications that convey urgency or threat should not be followed. Users should not respond to suspicious communications with any account credentials or personal Semantic Auth System Anti-Phishing Guidelines (RQ5) information. Communication should not be assumed to be legitimate solely because it accurately references the user's name, username, or other personal details, as such information may be available from prior data breaches or public sources. Comparison on the authentic site: Authentic communications from the system originate from the official domain, contain no embedded login links, and convey no artificial time pressure or threat of account closure. Any account status can be verified independently by navigating the site directly and reviewing the profile page.

4.1.7 GL7 Update Your Secret Phrase Promptly

Rationale. The Profile page of the Semantic Authentication System offers a self-service feature to update the secret phrase without involving the personal image, and vice versa. It can be used at any time, without any support-desk assistance, and is a direct solution to the "Not Implemented" recoverability rating assigned to the initial security question pattern in the design analysis. Users should not overuse the Profile page update function and should use it immediately whenever an event arises that raises concerns about the integrity of either trust factor.

Recommended Behaviors. The secret phrase should be changed if any of the following occur: a personal device is lost or stolen; a logon attempt is made that the user did not request; the phrase is described audibly; or an interaction with a page suspected of being fraudulent occurs. The user should sign in only once after the update, using the new phrase to ensure the system accepts it before use. The replacement phrase should be equally personal and memorable, yet entirely different from the one it replaces.

Behavior to Avoid. If any of the above conditions apply, the update should not be postponed, as doing so will shorten the window during which a potentially compromised factor may be exploited. If a previous phrase has been reused, it should not be reused after the change, as it might still be known to a possible intruder. Updates should not be performed on a public or shared device, since the update process itself is authenticated and might be performed in an insecure environment.

4.1.8 GL8 Authenticate Only from Personal and Trusted Devices

Rationale. The design properties of the Semantic Authentication System (SAS), the personal image, semantic verification, and the voice input modality function correctly only if malicious software, keyloggers, and screen-capture applications do not compromise the device that the user uses to prove their identity. No matter how strong the authentication pattern is, a tampered device can reveal credentials at the point of entry. Likewise, a public or shared device can store session cookies, credentials, or cookies from a prior user, which could allow a later user to access the account. The only risk that is eliminated is this one when authenticating from personal, trusted, and current devices.

Recommended Behaviors. Only personal devices used by the user, such as a phone, tablet, or computer, on which the user has a reasonable level of trust in the installed software, should be used for authentication. The device's operating system and the browser should be updated to the latest versions, since software updates regularly target vulnerabilities relevant to authentication sessions. To prevent unauthorized physical access to open sessions, the device should be secured with a PIN, password, biometric lock, or equivalent. Sessions should be ended by signing fully, especially when the device is passed to another person or left in a shared space.

Behavior to Avoid. Authentication should not be done on shared or public computers, including those located in libraries, hotels, or academic institutions. If an authenticated session is left on an unattended device in a shared environment, this should not be the case. Updates to the operating system or the browser should not be ignored, as they may introduce vulnerabilities that can be exploited during authentication.

5. KEY FINDINGS AND EMPIRICAL EVALUATIONS

5.1 Security Questions Are the Single Worst Performing Pattern. The baseline pattern failed in both usability (accessibility, navigation, feedback, and recoverability all rated "Not Implemented") and security (phishing, theft, and physical observation all unprotected), making it the only pattern to fail critically across every dimension simultaneously.

5.2 Semantic Authentication System. Passing all 19 evaluation criteria, the proposed semantic authentication system (SAS) was evaluated against the same 19-criterion framework used for all existing patterns, covering 11 usability factors, 3 deployment principles, and 5 security properties. It achieved a “Completely Implemented” rating across all three dimensions.

5.3 High User Satisfaction with Low Cognitive Load. Post-session measurements confirmed that the system was both satisfying and mentally manageable. The Likert satisfaction mean of 4.37 out of 5 exceeded the 4.0 threshold, with even the least technically confident participant recording a positive score of 3.30. The NASA-TLX cognitive load mean of 36.5 fell within the Acceptable range, indicating that AI-driven semantic authentication does not impose an unreasonable mental burden, even for users with limited digital literacy.

5.4 Phishing Detection Rate. All 10 participants, aged 21 to 62, with varying technical backgrounds, including one visually impaired user, successfully detected the simulated phishing page. In every case, the sole distinguishing factor was the absence of their personally generated AI image, which a fraudulent page cannot reproduce without server-side access. No participant required a technical explanation to act on it, making the 100% detection rate the most significant empirical finding of the study.

5.5 Voice-Only Authentication Is Fully Viable, Including Blind Users. The visually impaired participant (P10) completed the entire registration and login process without any visual interaction, navigating exclusively via screen reader and voice input. The text-to-speech announcement of the personal image served as the auditory equivalent of the visual trust signal; its absence on the simulated phishing page was sufficient for P10 to identify the page as fraudulent without any assistance.

CONTRIBUTIONS, LIMITATIONS, AND FUTURE WORK

The thesis has three main contributions to the field of usable security. Firstly, it is the most comprehensive multidimensional comparison of authentication design patterns ever undertaken, with 14 schemes evaluated against 19 criteria covering usability, security, and deployment, and the results of these evaluations are systematically documented in a similar tabular format. Second, it proposes a new design pattern, the Semantic Authentication System, which is the first to offer both AI-based semantic flexibility and a manageable personal visual trust token, while eliminating usability, accessibility, and phishing resistance issues faced by all mainstream authentication patterns. Third, it offers an empirically grounded list of eight guidelines, derived from controlled testing with a demographically diverse group of participants, that go beyond traditional password hygiene recommendations.

Some of the limitations of the research should be noted. The user study group of 10 individuals was not statistically representative of the general population but was purposively selected to reflect the population's demographic diversity. The findings are suggestive and contribute to the study's qualitative depth, and should be replicated on a larger scale before conclusions can be drawn about generalizability. The phishing simulation was conducted in a controlled laboratory environment where the participants were set up to perform an authentication task. In contrast, real phishing attacks may rely on distraction, urgency, and unfamiliar context, which are not easily replicated in a research protocol.

The most significant avenue for future development is integrating AI-powered anti-peeking features into the Semantic Authentication System. This would involve using the device's camera to detect a second observer before the semantic input step is initiated, automatically obscuring input when a bystander is detected, and scanning the surrounding environment to disable voice input in public or crowded spaces. Implementing these features would directly address the remaining gap in physical observation resilience (S4), pushing the system from partial to complete implementation across all five security criteria simultaneously.

RESULTS AND CONCLUSIONS

The research has produced evidence-based answers to all five research questions and delivered a set of actionable anti-phishing guidelines.

Results

RQ1: The comparative analysis of fourteen patterns established that none provides a reliable site-authentication signal before credentials are submitted, leaving users structurally exposed to phishing. Knowledge-based patterns, particularly security questions, are particularly vulnerable, as attackers can replicate prompts and harvest responses undetected.

RQ2: A nineteen-criterion evaluation framework spanning eleven usability properties, three deployment criteria, and five security properties was developed and applied. Accessibility was the single most universal failure across all evaluated patterns, followed by systemic weaknesses in navigation, feedback, and recoverability.

RQ3: The Semantic Authentication System was proposed, introducing two independently managed trust factors: a personal secret phrase verified via vector-embedding similarity and large-language-model judgment, and a personal AI-generated image displayed before any credential input is requested *(See Appendix E)*.

RQ4: Empirical testing with ten demographically diverse participants produced a 100% phishing detection rate, a Likert satisfaction mean of 4.37 out of 5, and a NASA-TLX cognitive load mean of 36.5 within the Acceptable range *(See Appendix A, D, and K)*. 17 criteria achieved a 100% pass rate, with isolated shortfalls in the remaining 2 criteria attributable to participant limitations rather than system deficiencies.

RQ5: Eight anti-phishing guidelines were derived from participant behavior, centered on the principle that a missing personal image is the primary indicator of a fraudulent login page, complemented by guidance on independent factor management, domain verification, secret phrase confidentiality, responsible voice input use, unsolicited request caution, prompt self-service recovery, and authentication from trusted devices only.

Conclusions

Phishing vulnerability in authentication arises from a structural design gap rather than user behavior. All fourteen evaluated patterns require users to authenticate the system before the system authenticates them. Implementing a server-bound, user-visible trust signal constitutes the necessary structural solution. At the same time, risk-based mechanisms should serve as a complementary layer that calibrates the strength of verification to the session context, rather than acting as a substitute.

Accessibility and phishing resistance can be addressed concurrently through a single modality-agnostic trust mechanism. Consistent implementation across auditory and visual channels fulfills both requirements without incurring additional design costs, indicating that these criteria should be treated as coequal from the outset rather than addressed independently.

Perceptual instinct emerges as a reliable security behavior when supported by robust structural design. If registration establishes a sufficiently personal anchor, users with varying levels of technical confidence can detect its absence on fraudulent pages without requiring a conceptual understanding of phishing.

Independent trust factor management represents a targeted recovery model that is not present in existing patterns. Updating each factor separately in response to specific compromises minimizes user disruption and reduces the exposure window, providing a practical improvement over a single-credential recovery system.

Risk-based authentication principles should be integrated into core verification logic rather than applied as an additional layer. The progressive lockout and semantic threshold calibration in the Semantic Authentication System demonstrate that proportional, context-aware authentication is feasible within a usable design. Behavioral signals, including device recognition and anomaly detection location, represent a logical direction for future development.

REFERENCES:

[NP20, 1] B. Naqvi and J. Porras, “Usable Security by Design: A Pattern Approach,” in HCI for Cybersecurity, Privacy and Trust, A. Moallem, Ed., Cham: Springer International Publishing, 2020, pp. 609–618.

[AOS24, 2] G. Assenza, A. Ortalda, and R. Setola, “Redefining Systemic Cybersecurity Risk in Interconnected Environments,” Applied Cybersecurity & Internet Governance, Aug. 2024.

[OBM+18, 3] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, “Multi-Factor Authentication: A Survey,” Cryptography, vol. 2, no. 1, p. 1, Jan. 2018.

[DBR22, 4] S. Dlamini, E. Block, and I. T. Rampedi, “Understanding students’ environmental perceptions and some of their determinants in Gauteng province: a case study at the University of Johannesburg, South Africa,” South African Geographical Journal, vol. 104, no. 1, pp. 89–106, Jan. 2022.

[Als25, 5] H. Alsharaya, “Password managers: a critical review of security, usability, and innovative designs,” Journal of Computer Sciences Institute, vol. 36, pp. 328–335, Sep. 2025.

[NB16, 6] L. Nosrati and A. M. Bidgoli, “A review of authentication assessment of Mobile-Banking,” in 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada: IEEE, Oct. 2016, pp. 1–5.

[Mac24, 7] I. S. MacKenzie, Human-computer interaction: an empirical research perspective, Second edition. Cambridge, MA: Morgan Kaufmann, 2024.

[YGP+25, 8] Y. Yang, P. Gope, A. Pasikhani, and B. Sikdar, “Privacy-Preserving Robotic-Based Multi-Factor Authentication Scheme for Secure Automated Delivery System,” IEEE Transactions on Information Forensics and Security, vol. 20, pp. 11860–11875, 2025.

[Kli24, 9] S. M. Klivan, “On the usability of authentication security communication,” pp. 1–200, 2024.

[PPM+20, 10] C. Peeters, C. Patton, I. N. S. Munyaka, D. Olszewski, T. Shrimpton, and P. Traynor, “SMS OTP Security (SOS): Hardening SMS-Based Two Factor Authentication,” in Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, Nagasaki, Japan: ACM, pp. 2–16.

[ZHY+25, 11] J. Zhao, F. He, Y. Yang, and Y. Zhang, “Identifying Implementation Flaws of SMS OTP Authentication,” IEEE Transactions on Mobile Computing, vol. 24, no. 9, pp. 7899–7913, Sep. 2025.

[GKB+22, 12] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, “How Does Usable Security (Not) End Up in Software Products? Results From a Qualitative Interview

Study," in 2022 IEEE Symposium on Security and Privacy (SP), May 2022, pp. 893–910.

[NSG+24, 13] A. Nash, H. Studiawan, G. Grispos, and K.-K. R. Choo, "Security Analysis of Google Authenticator, Microsoft Authenticator, and Authy," in Digital Forensics and Cyber Crime, S. Goel and P. R. Nunes de Souza, Eds., Cham: Springer Nature Switzerland, 2024, pp. 197–206.

[MAP+23, 14] E. Marasco, M. Albanese, V. V. R. Patibandla, A. Vurity, and S. S. Sriram, "Biometric multi-factor authentication: On the usability of the FingerPIN scheme," Security and Privacy, vol. 6, no. 1, pp. 1–261, Jan. 2023.

[Low24, 15] T. S. Lowell, "FACIAL BIOMETRIC AUTHENTICATION FOR SMARTPHONES: THE INTERSECTION OF SECURITY AND USABILITY," 2024.

[HNA+24, 16] H. R. M. H. Hamid, N. W. Nordin, N. Y. Abdullah, W. H. W. Ismail, and D. Abdullah, "Two Factor Authentication: Voice Biometric and Token-Based Authentication," in Applied Problems Solved by Information Technology and Software, A. Ismail, F. N. Zulkipli, M. A. Mohd Daril, and A. Öchsner, Eds., Cham: Springer Nature Switzerland, 2024, pp. 27–35.

[HW22, 17] L. Huang and C. Wang, "PCR-Auth: Solving Authentication Puzzle Challenge with Encoded Palm Contact Response," in 2022 IEEE Symposium on Security and Privacy (SP), May 2022, pp. 1034–1048.

[DDA24, 18] E. Djeki, J. Dégila, and M. H. Alhassan, "Reimagining Authentication: A User-Centric Two-Factor Authentication with Personalized Image Verification," in 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Jan. 2024, pp. 281–285.

[AKA24, 19] A. Ali, S. Khusro, and T. J. Alahmadi, "Accessible interactive learning of mathematical expressions for school students with visual disabilities," PeerJ Comput. Sci., vol. 10, p. e2599, Dec. 2024.

[NJS+24, 20] A. Nanda, J. J. Jeong, S. W. A. Shah, M. Nosouhi, and R. Doss, "Examining usable security features and user perceptions of Physical Authentication Devices," Computers & Security, vol. 139, p. 103664, Apr. 2024.

[YZX+23, 21] J. Yu, X. Zhang, Y. Xu, and J. Zhang, "CRoSS: Diffusion Model Makes Controllable, Robust and Secure Image Steganography," Advances in Neural Information Processing Systems, vol. 36, pp. 80730–80743, Dec. 2023.

[MKD20, 22] S. Maroofi, M. Korczyński, and A. Duda, "Are You Human?: Resilience of Phishing Detection to Evasion Techniques Based on Human Verification," in Proceedings of the ACM Internet Measurement Conference, Virtual Event USA: ACM, Oct. 2020, pp. 78–86.

[LL22, 23] C. Lee and K. Lee, "Impact Analysis of Resilience Against Malicious Code Attacks via Emails." Computers, Materials & Continua, vol. 72, no. 3, p. 4803, Sep. 2022.

[DY, 24] A. Y. Dukup and A. M. Yola, "Assessment of multi-factor authentication techniques for mitigating phishing attacks," Nature Journal of Emerging Sciences Technologies and Innovations, vol. 6, no. 3, pp. 304–314.

[FCM24, 25] I. Ferroukhi, C. Chassard, and J. Mardon, “A comprehensive overview of blue-veined cheeses,” International Dairy Journal, vol. 154, p. 105926, Jul. 2024.

[Rah+25, 26] M. Rahman et al., “Application of Artificial Intelligence in Detecting and Mitigating Cyber Threats,” IRJIET, vol. 09, no. 01, pp. 17–26, 2025.

[Goo24a, 27] Google LLC. Google Authenticator app. https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2, 2024. Accessed: April 2024.

[Goo24b, 28] Google LLC. Puppeteer. https://pptr.dev/. Accessed: April 2024.

[Goo18, 29] Google LLC. Advanced protection program. https://landing.google.com/advancedprotection/, 2018.

[Goo19, 30] Google LLC. Better protection against man-in-the-middle phishing attacks. https://security.googleblog.com/2019/04/better-protection-against-man-in-middle.html, 2019.

[PAD+24, 31] Poonkuzhali S., Abuthahir A., Dinesh S. P., and Daadreyaa D. “Enhanced Graphical Password Authentication System Based on Puzzles,” in Disruptive Technologies for Sustainable Development, CRC Press, 2024, pp. 14–19.

[GLZ15, 32] G. Geng, X. Lee, and Y. Zhang, “Combating phishing attacks via brand identity and authorization features,” Security Comm Networks, vol. 8, no. 6, pp. 888–898, Apr. 2015.

[YFL+25, 33] J. Yang, B. Fang, H. Lu, and Z. Tian, “Context-Aware Phishing-Resistant Authentication for Federated Identity in Internet of Things Platforms,” IEEE Internet of Things Journal, vol. 12, no. 8, pp. 11121–11134, Apr. 2025.

[DMD+24, 34] P. T. Duy, V. Q. Minh, B. T. H. Dang, N. D. H. Son, N. H. Quyen, and V.-H. Pham, “A Study on Adversarial Sample Resistance and Defense Mechanism for Multimodal Learning-Based Phishing Website Detection,” IEEE Access, vol. 12, pp. 137805–137824, 2024.

[Fok23, 35] N. Fok, “Relevant research article summaries,” Environ. Health Rev., vol. 66, no. 1, pp. 21–24, Apr. 2023.

[Bac+23, 36] B. Bach et al., “Dashboard Design Patterns,” IEEE Transactions on Visualization and Computer Graphics, vol. 29, no. 1, pp. 342–352, Jan. 2023.

[DTO23, 37] F. Di Nocera, G. Tempestini, and M. Orsini, “Usable Security: A Systematic Literature Review,” Information, vol. 14, no. 12, p. 641, Nov. 2023.

[Sam23, 38] G. Sambin, “Usability of Safety Critical Applications in Enterprise Environments: Defining Guidelines for Error-Preventing UI/UX Patterns and Improving Existing Interfaces,” laurea, Politecnico di Torino, 2023. Accessed: May 04, 2026.

[NB07, 39] J. M. Noyes and D. P. J. Bruneau, “A self-analysis of the NASA-TLX workload measure,” Ergonomics, vol. 50, no. 4, pp. 514–519, Apr. 2007.

[Rob23, 40] J. Robinson, “Likert Scale,” in Encyclopedia of Quality of Life and Well-Being Research, Springer, Cham, 2023, pp. 3917–3918.

[Pal26, 41] J.M. Palh Semantic Authentication System, an AI-powered authentication design pattern: https://www.ms-thesis.online/, 2026

APPENDIXES

Appendix A: User Study Questionnaire and A/B Testing Materials

Available at: https://drive.google.com/drive/folders/1nKkLRmaiJIy8vM5Irk5U-9PPlAlrYUyq?usp=sharing

Appendix B: Semantic Authentication System - UI Screen Captures

Available at: https://drive.google.com/drive/folders/1VQIjdvqT25jdQHXZ_MAm-urYvf_z7g08?usp=sharing

Appendix C: Semantic Authentication System - High-Fidelity UI Prototype

Available at: https://drive.google.com/drive/folders/1-8WwxWt4rjfzwa4dX9BE-hCejv4N2lLj?usp=sharing

Appendix D: Recorded User Testing Sessions (Participants P01–P10)

Available at: https://drive.google.com/drive/folders/1fu6w7nYFytTFYOaSWk6_qXC5Gw8x5M_N?usp=sharing

Appendix E: Video Demonstration for Semantic Authentication System

Available at: https://drive.google.com/drive/folders/1TSfZGSHhPLrOKfn26_TzhBUPkxcsF_zS?usp=sharing

Appendix F: Webflow Wireframes for Semantic Authentication System

Available at: https://drive.google.com/drive/folders/1sD6g_VcsR2q-Uii-Oo9zwEuGgnm4DZr9?usp=sharing

Appendix G: Project Code for Semantic Authentication System

Available at: https://github.com/Jahanzaib523/masters-thesis.git

Appendix H: Semantic Authentication System's Architecture

Available at: https://github.com/Jahanzaib523/masters-thesis/tree/main/docs

Appendix I: Phishing Simulation for Semantic Authentication System

Available at: https://static.ms-thesis.online

Appendix J: Semantic Authentication System – Web application

Available at: https://ms-thesis.online

Appendix K: Likert and NASA-TLX Results

Available at: https://drive.google.com/drive/folders/1likxHmoZX5uZJ0Cwuz8Kke_JXTsgDXs4?usp=drive_link

Appendix L: Tools Used

MS-Word, Grammarly, Python, GitHub, Drive, FasAPI, React, Visual Studio Code, Digital Ocean (EC2), Hugging Face, Groq, and OpenAI LLMs and