

**PASSWORD HASHING SECURITY ANALYSIS: EVALUATING
PASSWORD STRENGTH, ENTROPY AND HASHING RESISTANCE
USING A WEB-BASED PASSWORD ANALYZER**

A Dissertation Submitted in Partial Fulfilment of the
Requirements for the Award of
MSc Cyber Security and Digital Forensics

Student Name: Jaraben Mangukiya

Student number:2434479

Chapter 1: Introduction

1.1 Background of the Study

Passwords remain one of the most widely used authentication mechanisms in modern digital environments. Despite the growing adoption of biometric authentication, passkeys and multi-factor authentication (MFA), password-based access control continues to support web applications, email systems, enterprise platforms, cloud services, online banking, e-commerce and administrative systems. This persistence is partly practical: passwords are inexpensive to deploy, familiar to users, compatible with existing systems and easy to integrate into web-based authentication workflows. However, their continued dominance also creates a structural cybersecurity problem because password security depends on two fragile conditions: users must create and manage strong secrets, and developers must store those secrets using secure password-hashing mechanisms.

The scale of password-related risk remains substantial. Microsoft's 2024 Digital Defense Report states that password-based attacks represented over 99% of 600 million daily identity attacks observed in Microsoft Entra data, with Microsoft blocking approximately 7,000 password attacks per second during the reporting period [1]. This evidence shows that password attacks are not a legacy issue; they remain active, automated and operationally significant in cloud and enterprise environments. At the same time, Verizon's 2026 Data Breach Investigations Report indicates that vulnerability exploitation has overtaken stolen passwords as the leading breach entry point, but the same report still identifies credential compromise as a major continuing security concern [2]. The implication is critical: even when attackers increasingly exploit software vulnerabilities, compromised credentials remain valuable because they allow attackers to access systems through legitimate login pathways.

The password problem is therefore socio-technical rather than purely technical. On the user side, individuals often create passwords that are memorable rather than random. These include short numeric strings, dictionary words, names, dates, keyboard patterns and predictable substitutions such as replacing "a" with "@" or adding "123" at the end of a word. Such passwords may satisfy basic complexity rules but remain vulnerable to dictionary, brute-force

and hybrid attacks. NIST has moved away from unnecessary composition rules and instead emphasises minimum length, screening against compromised passwords and secure verifier-side storage [3]. This is important because traditional password policies often encourage superficial complexity without necessarily increasing real resistance to guessing.

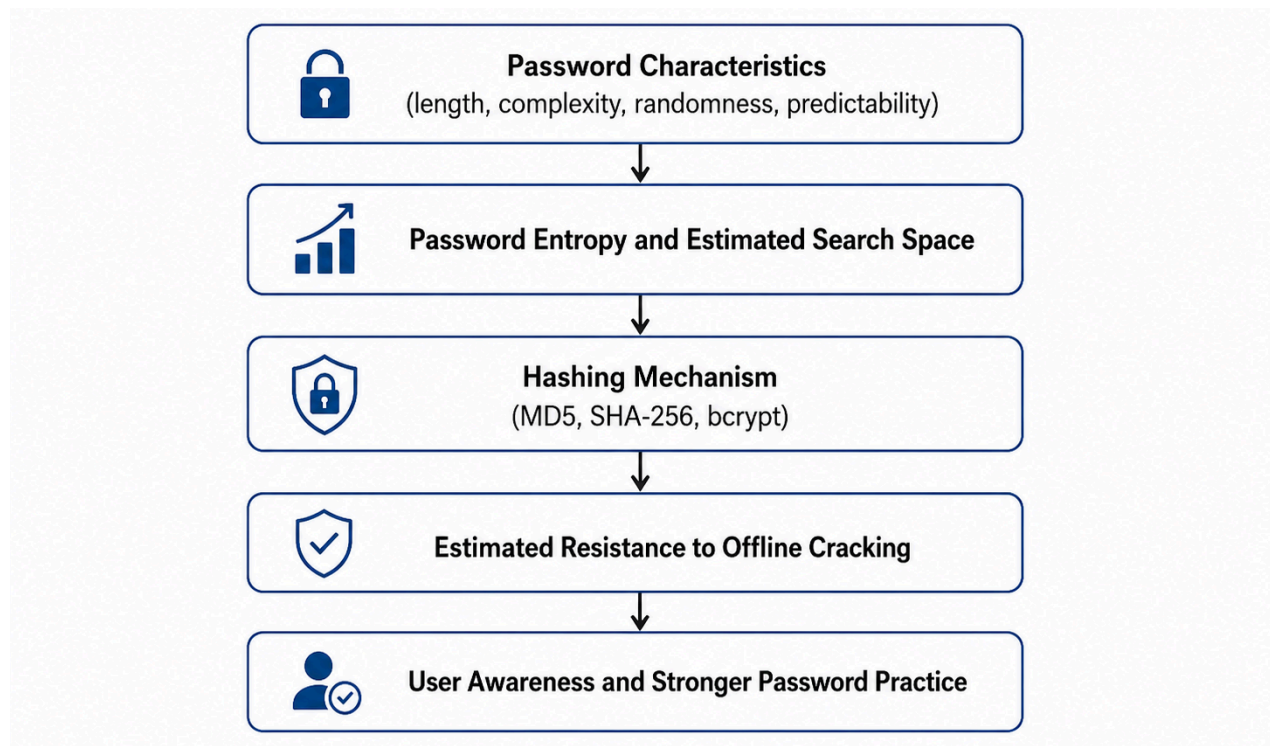
On the system side, password security depends heavily on how passwords are stored. A password should never be stored in plaintext. However, hashing a password is not automatically sufficient. A general-purpose hash function may protect data integrity in one context but still be unsuitable for password storage if it is too fast or unsalted. NIST states that verifiers must store passwords in a form resistant to offline attacks, using salted password-hashing schemes with a salt and cost factor so that each password guess becomes more expensive for an attacker who obtains a hashed password file [4]. OWASP similarly recommends modern password-hashing algorithms such as Argon2id, bcrypt, scrypt or PBKDF2, rather than fast general-purpose hashing functions [5].

This distinction is central to the dissertation. MD5 was originally specified as a message-digest algorithm that produces a 128-bit fingerprint from an input message [6]. SHA-256, within the SHA-2 family, is specified by NIST as part of the Secure Hash Standard for generating message digests to detect whether data has changed [7]. These algorithms have historical and technical importance, but they are not designed as adaptive password-hashing mechanisms. Their computational speed is useful for integrity checking, but it becomes a weakness in password storage because attackers can test large numbers of password guesses rapidly after obtaining a stolen hash database. By contrast, bcrypt was designed as a future-adaptable password scheme with an adjustable cost factor, allowing password hashing to become more computationally expensive as hardware improves [8].

This dissertation responds to this problem through the design, implementation and evaluation of a deployed web-based Password Security Analyzer. The application includes an Analyze Password section and a Strong Password Suggestions section, allowing users to enter a password and receive stronger alternatives [9]. The uploaded project notes also define the study around MD5, SHA-256 and bcrypt, with attention to brute-force resistance, password

complexity, cracking time, computational cost and secure storage recommendations . Therefore, the dissertation is not only a theoretical discussion of password hashing. It is an implementation-based cybersecurity study that evaluates how password strength, entropy, estimated cracking resistance and hashing awareness can be communicated through a practical web artefact.

Figure 1.1: Conceptual Framework for Password Security Analysis



Source: Author's conceptual framework based on NIST password guidance, OWASP password-storage guidance and the implemented Password Security Analyzer.

The conceptual framework shows that password security is not determined by one factor alone. A strong password can still be weakened by insecure storage, while secure hashing cannot fully compensate for extremely predictable user choices. For instance, bcrypt can slow each guessing attempt, but a password such as 123456 may still be tested early in a dictionary attack. Conversely, a long and random password is more resistant to guessing, but if it is stored using a fast unsalted hash, attackers can still perform high-speed offline guessing. This study therefore

treats password security as the combined outcome of password characteristics, hashing design and user awareness.

1.2 Research Problem and Rationale

The central research problem is that password security is often misunderstood at both user and developer levels. Users may believe that adding one number or symbol creates a secure password, while developers may believe that applying any cryptographic hash function is sufficient for password storage. Both assumptions are incomplete. Password complexity matters, but predictability can undermine complexity. Hashing matters, but algorithm choice and configuration determine whether the stored password is resistant to offline attacks.

This problem has practical relevance because password compromise frequently enables wider system compromise. A stolen credential can bypass many technical controls because the attacker appears to authenticate as a legitimate user. MFA reduces this risk by requiring additional verification factors, and CISA describes MFA as an important method for preventing unauthorised access beyond passwords alone [10]. However, MFA does not remove the need for strong password storage. If password databases are poorly protected, users remain exposed to credential reuse attacks, identity compromise and downstream account takeover.

The rationale for this dissertation is therefore based on the need for a practical and educational tool that links password strength with hashing awareness. Existing standards explain secure password storage, but users and junior developers may not fully understand how password characteristics and hashing mechanisms interact. The Password Security Analyzer addresses this gap by making password-security concepts visible through a deployed application. Its purpose is not to perform offensive password cracking, but to demonstrate risk, encourage stronger password selection and support secure password-storage understanding.

1.3 Problem Statement

Password security remains a persistent cybersecurity challenge because users often create weak or predictable passwords, while developers may misunderstand the difference between general-purpose cryptographic hashing and secure password hashing. Fast algorithms such as

MD5 and SHA-256 are unsuitable for password storage when used alone because their speed allows attackers to test large numbers of guesses rapidly. Although bcrypt and similar adaptive hashing functions offer stronger protection, users and junior developers may not fully understand why algorithm choice matters.

The problem is therefore twofold. First, many users lack meaningful feedback on why their passwords are weak. A password may contain uppercase letters, lowercase letters, numbers and symbols but still be predictable if it is based on a common word, personal name or date. Second, some systems may hash passwords without using appropriate password-hashing schemes, salts or cost factors. This creates a false sense of security because a hashed password database can still be vulnerable to offline guessing if the hash function is too fast or poorly configured.

This dissertation addresses the problem by designing and evaluating a web-based Password Security Analyzer that demonstrates password strength, estimated cracking resistance and secure hashing awareness. The application provides a practical mechanism for linking user-facing password feedback with technical concepts such as entropy, hashing, salting and adaptive computational cost.

1.4 Aim of the Study

The aim of this study is:

To design, implement and critically evaluate a deployed web-based Password Security Analyzer that demonstrates how password complexity, entropy and hashing mechanisms influence estimated resistance to password-cracking attacks, with the aim of improving password-security awareness and informing secure password-storage practice.

This aim reflects the applied nature of the dissertation. The study is not limited to reviewing password-security literature; it evaluates a working cybersecurity artefact that has been deployed online. The application provides a practical setting for analysing how users can receive feedback about password weakness and how secure hashing concepts can be communicated through a web interface.

1.5 Research Objectives

The objectives of this dissertation are:

1. To critically examine how password length, complexity, entropy and predictability influence password strength.
2. To evaluate the security implications of MD5, SHA-256 and bcrypt in relation to password storage and offline cracking resistance.
3. To design and implement a deployed web-based Password Security Analyzer using modern web technologies.
4. To test the analyser using weak, medium and strong password examples and assess how effectively it communicates password-security risk.
5. To develop evidence-based recommendations for secure password creation, password storage and future improvement of the implemented system.

These objectives are deliberately aligned with both the technical and evaluative parts of the dissertation. Objectives one and two focus on the cybersecurity theory underpinning password security. Objective three focuses on the implementation of the artefact. Objective four evaluates the artefact using controlled examples. Objective five translates the findings into practical recommendations.

1.6 Research Questions

This dissertation is guided by the following research questions:

1. How do password length, character diversity, entropy and predictability influence estimated resistance to password-cracking attacks?
2. Why are fast general-purpose hashing algorithms such as MD5 and SHA-256 unsuitable for password storage when used without salting and key stretching?
3. In what ways does bcrypt improve password-storage security through salting, adaptive cost and deliberate computational expense?

4. How effectively does the implemented Password Security Analyzer communicate password-strength risks and improvement guidance to users?
5. What secure password-creation and password-storage recommendations can be derived from the project's design, implementation and evaluation?

These questions ensure that the dissertation remains analytical rather than descriptive. They require the study to evaluate password strength, hashing design, user-facing communication and practical security recommendations.

1.7 Contribution of the Study

This study makes three main contributions. First, it provides a practical cybersecurity artefact in the form of a deployed Password Security Analyzer. This is significant because the dissertation does not only discuss password security in abstract terms; it implements a tool that demonstrates password analysis and stronger password suggestions.

Second, the study provides an implementation-based evaluation of password strength, entropy and hashing awareness. Many discussions of password security focus either on user behaviour or cryptographic design separately. This dissertation connects the two by showing how password characteristics and hashing mechanisms jointly influence estimated cracking resistance.

Third, the study contributes to secure password practice by translating technical concepts into user-facing feedback. Concepts such as entropy, bcrypt, salting and computational cost can be difficult for non-specialists to understand. The analyser helps communicate these ideas through a practical interface, thereby supporting cybersecurity education, user awareness and developer understanding.

Table 1.1: Summary of Study Contributions

Contribution Area	Description	Dissertation Value
-------------------	-------------	--------------------

Practical artefact	A deployed Password Security Analyzer	Demonstrates implementation, not only theory
Security evaluation	Tests weak, medium and strong password examples	Links password behaviour to cracking resistance
Hashing awareness	Compares MD5, SHA-256 and bcrypt implications	Shows why secure password hashing matters
User guidance	Provides strong password suggestions	Supports behavioural improvement
Academic contribution	Connects literature, implementation and evaluation	Strengthens applied cybersecurity relevance

1.8 Scope of the Study

The study focuses on password-strength analysis, password entropy, estimated cracking resistance, MD5/SHA-256/bcrypt comparison, bcrypt hashing demonstration, strong password suggestions and web-based implementation and deployment. This scope is appropriate because the dissertation investigates password-security awareness and password-storage principles rather than full enterprise identity management.

The live application provides the practical basis of the study. It includes an Analyze Password function and a Strong Password Suggestions function, which directly support the dissertation's focus on password evaluation and improvement [9]. The app is therefore used as a defensive educational artefact rather than an offensive cracking platform.

The study does not attack real accounts, store real user credentials, perform unauthorised penetration testing, replace professional password-auditing tools or implement a complete login and registration system. It also does not claim that estimated cracking time is equivalent to real-world cracking performance. Actual cracking success depends on attacker hardware, GPU acceleration, password dictionaries, breach context, salting, cost factors and attack strategy.

Therefore, crack-time estimates in the analyser are treated as comparative indicators rather than exact predictions.

1.9 Significance of the Study

The study is significant for cybersecurity students, software developers, general users, organisations and researchers. For cybersecurity students, the dissertation demonstrates how theoretical concepts such as entropy, hashing, salting and offline cracking can be operationalised through a working web application. This is important because cryptographic and authentication concepts can remain abstract unless connected to practical implementation.

For software developers, the study reinforces the importance of choosing password-hashing mechanisms that are suitable for authentication security. Developers may know that passwords should be hashed, but still fail to distinguish between fast message-digest algorithms and adaptive password-hashing functions. OWASP's password-storage guidance directly supports the position that modern systems should use password-hashing algorithms designed for password storage rather than relying on fast general-purpose hashing alone [5].

For general users, the study provides a practical way to understand password weakness. Users are often told to create "strong passwords" without being shown why some passwords are easier to crack than others. By giving password analysis and stronger password suggestions, the application supports more informed password behaviour.

For organisations, the study supports password-policy awareness and secure authentication practice. Password compromise remains relevant to identity security, and MFA should be treated as an additional protection layer rather than a substitute for secure password storage [10]. Organisations therefore benefit from both user education and secure backend implementation.

For researchers, the study provides an applied evaluation of password strength and hashing awareness. It demonstrates how a cybersecurity artefact can be designed and evaluated in response to a recognised security problem. This gives the dissertation practical and academic relevance.

Table 1.2: Significance of the Study by Stakeholder

Stakeholder	Significance
Cybersecurity students	Demonstrates practical password-security concepts such as entropy, hashing and crack-time estimation.
Software developers	Shows why secure password hashing matters and why fast hashes are unsuitable for password storage.
General users	Helps users understand weak-password risks and improve password choices.
Organisations	Supports password-policy awareness, secure authentication design and credential-risk reduction.
Researchers	Provides an implementation-based evaluation of password strength and hashing awareness.

1.10 Dissertation Structure

This dissertation is organised into six chapters. Chapter 1 introduces the study by presenting the background, research problem, aim, objectives, research questions, contribution, scope, significance and dissertation structure. Chapter 2 reviews the literature on password authentication, weak password behaviour, entropy, MD5, SHA-256, bcrypt, salting, key stretching and password attack techniques. Chapter 3 explains the methodology, including the design science research approach, development method, test data, evaluation metrics and ethical considerations. Chapter 4 presents the system design and implementation of the Password Security Analyzer, including system architecture, functional requirements, user interface, password-strength logic, hashing comparison and deployment. Chapter 5 evaluates the system through weak, medium and strong password tests, discusses the results and links the findings to the research questions. Chapter 6 concludes the dissertation by summarising the findings, identifying limitations, presenting recommendations and proposing future work.

Overall, this chapter has established the rationale for the dissertation. Passwords remain widely used because they are practical and familiar, but they remain risky because their security depends on both user behaviour and system-side storage practice. The study therefore investigates password security through a deployed web-based analyser that demonstrates password strength, estimated cracking resistance and secure hashing awareness.

Chapter 2: Literature Review

2.1 Introduction

This chapter critically reviews literature on password authentication, password attacks, password entropy, cryptographic hashing, secure password storage and password-strength

awareness. The purpose is not simply to describe password security concepts, but to evaluate how password weakness emerges from the interaction between human behaviour and technical implementation. This distinction is important because password compromise is rarely caused by one isolated factor. A user may choose a weak or reused password, but the risk becomes substantially worse when the system stores that password using a fast, unsalted or poorly configured hash.

The review is structured around two linked arguments. First, passwords remain dominant because they are cheap, familiar and highly deployable, despite their known weaknesses. Second, password security depends on both user-side password quality and verifier-side storage design. NIST requires verifiers to store memorised secrets in a form resistant to offline attacks and states that password-hashing schemes should use a password, salt and cost factor to make guessing attacks expensive [3], [4]. OWASP similarly recommends password-specific hashing algorithms such as Argon2id, bcrypt, scrypt or PBKDF2, rather than fast general-purpose hashing functions [5]. These sources establish the technical foundation for evaluating MD5, SHA-256 and bcrypt in this dissertation.

This chapter is directly linked to the implemented Password Security Analyzer, a deployed web application that analyses password strength and provides stronger password suggestions [9]. The literature therefore provides the theoretical basis for evaluating password complexity, entropy, estimated cracking resistance and hashing awareness in later chapters.

2.2 Passwords in Authentication Systems

Passwords remain embedded in digital authentication because they are inexpensive, familiar and compatible with existing infrastructure. They are used across web applications, email accounts, enterprise platforms, cloud services, online banking, e-commerce systems and administrative dashboards. Compared with biometric systems, hardware tokens or fully passwordless authentication, passwords are easier to deploy and require no specialised hardware. Bonneau et al. explain that password alternatives must compete not only on security but also on usability and deployability, which helps explain why passwords have remained difficult to replace [11].

However, the persistence of passwords should not be confused with superiority. Passwords remain common largely because they are practical, not because they are ideal. Password authentication depends on the user's ability to create, remember and protect a secret. This creates a structural tension between usability and security. Highly random passwords are more secure but difficult to memorise, while meaningful passwords are easier to remember but easier to guess. Florêncio, Herley and van Oorschot describe this as a "finite-effort user" problem, where users managing many accounts cannot realistically maintain unique, strong passwords for every service without simplification or reuse [12].

The current threat landscape confirms that password-based risks remain operationally significant. Microsoft reports that password-based attacks accounted for more than 99% of 600 million daily identity attacks observed in Microsoft Entra data, with approximately 7,000 password attacks blocked per second during the reporting period [1]. Verizon's 2026 DBIR also shows that credential risks remain significant even though vulnerability exploitation has overtaken stolen credentials as the leading breach entry point, with exploitation of vulnerabilities reported as the starting point for nearly one-third of breaches [2]. This is analytically important because it shows that passwords are part of a wider identity-security problem rather than an isolated login issue.

Therefore, password authentication should be viewed as a trade-off. It is convenient and deployable, but it creates risk when users select predictable secrets or when developers store passwords using unsuitable hashing practices. This dual weakness justifies the dissertation's combined focus on password-strength analysis and password-hashing awareness.

2.3 Weak Passwords and Human Behaviour

Weak passwords are often produced by rational user behaviour under practical constraints. Users are required to manage many accounts, remember multiple credentials and respond to inconsistent password policies. As a result, they frequently choose passwords that are memorable rather than random. Common weaknesses include password reuse, short numeric strings, dictionary words, personal names, dates of birth, keyboard sequences and predictable substitutions. A password such as P@ssword123 appears complex because it contains

uppercase letters, a symbol and numbers, but it remains highly predictable because it follows a common word-plus-substitution pattern.

This creates a weakness in traditional password-composition policies. Many systems require uppercase letters, lowercase letters, numbers and symbols, but such rules often encourage predictable transformations rather than genuine unpredictability. NIST's guidance reflects this shift by reducing emphasis on arbitrary composition rules and focusing instead on length, compromised-password screening and secure verifier-side handling [3], [4]. In other words, a password should not be considered secure merely because it contains a number or symbol.

Password-cracking literature reinforces this point. Weir et al. show that probabilistic context-free grammars can generate password guesses in high-probability order by learning structures from disclosed passwords [13]. This means attackers do not need to search the entire theoretical password space randomly. They can prioritise human patterns such as word-plus-number, name-plus-year and symbol substitutions. Wheeler's zxcvbn password-strength estimator also criticises simplistic lower/upper/digit/symbol rules and argues for estimation based on pattern recognition and attacker-like guessing models [14]. Dropbox's zxcvbn documentation further explains that the estimator recognises common passwords, names, words, dates, repeats, keyboard patterns and leetspeak patterns [18].

The critical implication for this dissertation is that password strength cannot be evaluated only through visible complexity. A good password analyser must consider length, character diversity, randomness and predictable structure. The Password Security Analyzer is therefore valuable because it provides users with feedback and stronger password suggestions rather than simply enforcing arbitrary composition rules.

2.4 Password Entropy and Complexity

Password entropy is commonly used to estimate unpredictability. It measures the approximate size of the password search space and is often expressed in bits. A longer password drawn from a larger character set has higher theoretical entropy than a short password drawn only from digits or lowercase letters. For example, a password using lowercase letters, uppercase letters,

numbers and symbols has a larger theoretical character pool than a password containing only numbers.

However, entropy must be interpreted critically. Theoretical entropy assumes randomness, but many human-created passwords are patterned. For example, Summer2026! contains uppercase letters, lowercase letters, digits and a symbol, but it is based on a dictionary word, a likely year and a common suffix. A purely mathematical calculation may therefore overestimate its practical strength if it ignores human predictability. This is why modern password-strength estimation has moved toward pattern-aware models such as zxcvbn [14], [18].

Table 2.1: Password Complexity Factors

Factor	Security Benefit	Limitation
Length	Expands the possible search space and usually increases resistance to brute-force guessing.	Weak if based on predictable phrases, repeated patterns or common words.
Uppercase letters	Increases the character pool and can improve theoretical entropy.	Often used predictably at the beginning of a password.
Numbers	Adds variation and increases the possible character set.	Commonly added at the end, especially “123” or years.
Symbols	Improves character diversity and may increase entropy.	Not enough if the password remains short or dictionary-based.
Randomness	Reduces predictability and improves resistance to structured guessing.	Harder to memorise without a password manager.

Entropy is therefore useful, but it should not be treated as a complete measure of security. The Password Security Analyzer can use entropy and complexity indicators to educate users, but its estimated crack time should be treated as a comparative indicator rather than an exact

prediction. Actual cracking success depends on attacker hardware, GPU acceleration, leaked password lists, attack strategy, hash algorithm, salt use and cost factor.

2.5 Cryptographic Hashing

Cryptographic hashing converts an input into a fixed-length digest. A secure hash function is designed to be one-way, meaning that the original input should not be practically recoverable from the digest. Password systems use hashing so that a verifier stores a hash rather than a plaintext password. If a database is compromised, attackers should not immediately obtain users' real passwords.

However, password cracking does not require reversing the hash mathematically. Instead, attackers guess candidate passwords, hash each guess using the same algorithm and compare the computed digest with the stolen digest. If the values match, the password is recovered. This is why the cost of each guess matters. A fast hash function allows attackers to test large numbers of guesses quickly. A slow password-hashing function deliberately increases the attacker's cost.

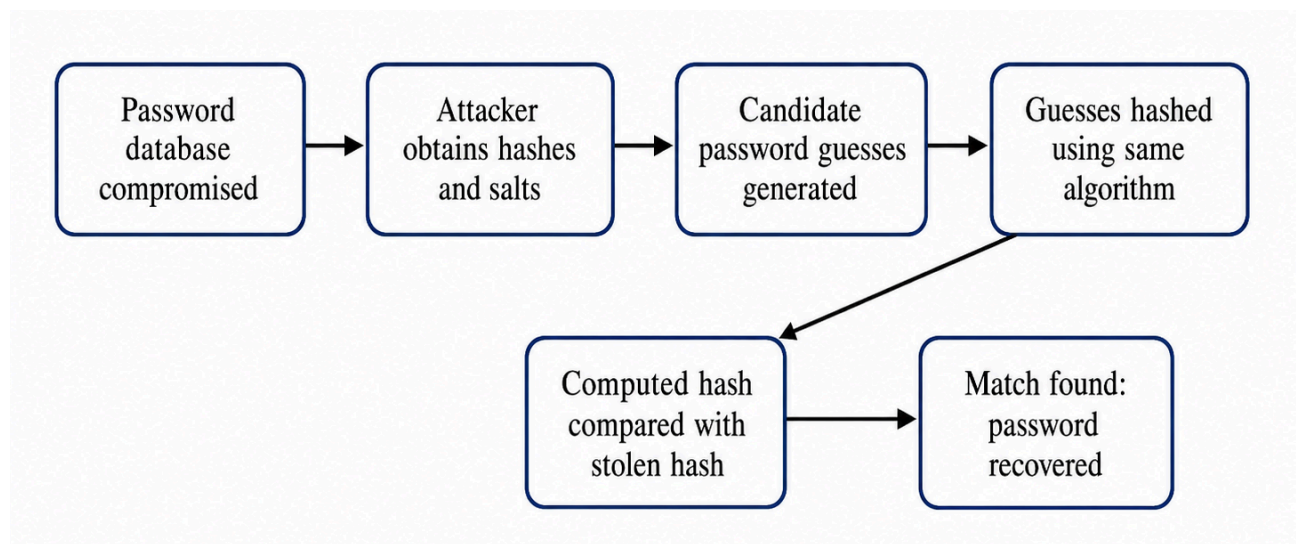


Figure 2.1: Offline Password-Cracking Process After Database Compromise

Source: Author's synthesis based on NIST SP 800-63B, OWASP Password Storage guidance and password-cracking literature.

Insert Figure 2.1 here: Offline Password-Cracking Process

For an additional online explanatory image, a password-hashing and verification flow diagram from LaunchCode can be used to show how a password is hashed during registration and compared again during login. However, this should be used only as a basic explanatory visual, not as core academic evidence, because it is an educational web resource rather than a peer-reviewed cybersecurity source. A more dissertation-appropriate approach is to use the author-created Figure 2.1 and cite NIST/OWASP in the caption [3]–[5].

NIST requires memorised secrets to be salted and hashed using a suitable password-hashing scheme that includes a salt and cost factor [3], [4]. OWASP also warns against using fast hashing algorithms for password storage and recommends slow password-hashing algorithms [5]. Therefore, a distinction must be made between general cryptographic hashing and password hashing. A function may be secure for file integrity but still unsuitable for password storage.

2.6 MD5

MD5 was introduced by Rivest as a message-digest algorithm that produces a 128-bit fingerprint or digest of an input message [6]. Historically, it was used for checksums and integrity verification. However, MD5 is now unsuitable for modern password storage. Its weaknesses include known collision vulnerabilities and very high computational speed. Wang and Yu demonstrated practical attacks against MD5 and related hash functions, supporting the conclusion that MD5 should not be used in security-sensitive contexts [16].

For password storage, MD5's weakness is not only collision resistance. Its speed is a major practical problem. Passwords are typically weaker than cryptographic keys because they are created by humans and often follow predictable patterns. If a stolen password database is protected with MD5, an attacker can compute large numbers of guesses rapidly and compare them against the stolen hashes. Therefore, MD5's efficiency becomes a security liability in the password-storage context.

This is a common developer misunderstanding. Developers may assume that storing an MD5 hash is secure because hashes are one-way. However, attackers do not need to invert the hash;

they only need to guess likely passwords and hash them. MD5 should therefore be treated as historically important but operationally inappropriate for password protection. In this dissertation, MD5 functions as a baseline example showing why fast general-purpose hashes should not be used for password storage.

2.7 SHA-256

SHA-256 is part of the SHA-2 family defined in NIST's Secure Hash Standard [7]. NIST explains that these hash algorithms generate message digests that can be used to detect whether messages have changed [7]. SHA-256 is stronger than MD5 for cryptographic integrity purposes and does not share MD5's practical collision weaknesses in the same way. However, this does not make SHA-256 automatically suitable for password storage when used alone.

The key issue is contextual suitability. SHA-256 is a fast general-purpose hash function. Its speed is useful for integrity verification, digital signatures and other cryptographic applications. In password storage, the same speed becomes problematic because attackers who obtain a hash database can compute guesses quickly. OWASP therefore advises against using fast hashing algorithms directly for password storage and recommends password-specific mechanisms such as Argon2id, bcrypt, scrypt or PBKDF2 [5].

A careful review should avoid overstating the weakness of SHA-256. SHA-256 is not "broken" as a hash function. The problem is that password storage requires a different security property: resistance to high-speed offline guessing. Therefore, SHA-256 may be secure for detecting data alteration but weak for password protection if used without salting and key stretching. This distinction is important for junior developers because it shows that stronger general cryptography does not automatically equal secure password storage.

2.8 bcrypt

bcrypt was introduced by Provos and Mazières as a future-adaptable password scheme [8]. Its key contribution is the use of an adjustable cost factor. This allows the hashing process to become more computationally expensive as hardware improves. bcrypt also includes salting,

which ensures that identical passwords do not produce identical stored hashes and reduces the usefulness of precomputed attacks.

bcrypt differs from MD5 and SHA-256 because it deliberately prioritises computational cost. In ordinary authentication, a small delay during login is acceptable. For an attacker attempting millions of guesses, that delay becomes a significant obstacle. OWASP includes bcrypt among acceptable password-hashing algorithms, while recommending Argon2id as the preferred modern option where available [5]. Argon2 is described in RFC 9106 as a memory-hard function for password hashing and proof-of-work applications, designed to make attacks more expensive by requiring memory as well as computation [15]. Percival's script work also supports the principle that memory-hard key derivation can reduce the advantage of specialised cracking hardware [17].

However, bcrypt should not be overstated. It does not make weak passwords safe. If users choose 123456, password or a predictable name-plus-year password, attackers are likely to test these guesses early. bcrypt increases the cost of each guess but cannot eliminate risk created by predictable password behaviour. Therefore, bcrypt is best understood as one part of a layered password-security strategy that includes strong passwords, compromised-password screening, password managers and MFA.

2.9 Salting, Peppering and Key Stretching

Secure password storage depends on protective mechanisms that increase resistance to offline cracking. Salting, peppering, key stretching and adaptive hashing are especially important.

A salt is a unique random value added to each password before hashing. Its purpose is not secrecy but uniqueness. If two users choose the same password, unique salts ensure that their stored hashes differ. Salts also reduce the effectiveness of rainbow tables because attackers cannot use one precomputed table against many users. NIST requires password-hashing schemes to include salts as part of verifier-side protection [3], [4].

A pepper is a secret value added to the hashing process but stored separately from the password database. It adds another layer of defence if the database is stolen but the application

secret remains protected. However, peppering introduces operational challenges because the pepper must be stored, protected and rotated securely. OWASP treats peppering as an additional defence, not a replacement for secure password hashing [5].

Key stretching increases the computational effort needed for each password guess. PBKDF2 uses iterations, bcrypt uses a cost factor, scrypt uses memory-hard computation and Argon2 uses memory, time and parallelism parameters [5], [15], [17]. This is important because modern attackers often use GPUs or specialised hardware for cracking. Memory-hard approaches make such attacks less efficient by increasing memory requirements as well as computation.

Table 2.2: Password Storage Protection Techniques

Technique	Meaning	Security Value	Limitation
Salting	Adds unique random data to each password before hashing.	Defends against rainbow tables and duplicate-hash exposure.	Does not stop dictionary attacks against weak passwords.
Peppering	Adds a secret value stored separately from the database.	Adds protection if the password database alone is stolen.	Requires careful key management and rotation.
Key stretching	Repeats or increases computational work.	Slows brute-force and dictionary guessing.	Can affect server performance if poorly configured.
Adaptive hashing	Allows cost to increase as hardware improves.	Maintains resistance over time.	Requires periodic review of cost parameters.
Memory-hard hashing	Uses significant memory as part of hashing.	Makes GPU/ASIC cracking less efficient.	May be harder to configure in constrained environments.

For an online visual supplement, ExpressVPN’s salting illustration can be used to explain why the same password produces different stored hashes when different salts are used. However, because this is a vendor blog image, it should be cited only as an explanatory illustration and not as primary academic evidence. The stronger approach is to base the analysis on NIST, OWASP, RFC 9106 and the author-created diagrams [3]–[5], [15].

2.10 Password Attack Techniques

Password attacks exploit weak user behaviour, poor storage mechanisms or both. The main attack techniques relevant to this dissertation are brute-force attacks, dictionary attacks, rainbow-table attacks, hybrid attacks and credential stuffing.

A brute-force attack attempts possible combinations until a match is found. Its success depends on password length, character set, hashing speed and attacker resources. A dictionary attack uses common words, leaked passwords and known credential patterns. A hybrid attack modifies dictionary words with numbers, symbols, capitalisation or dates, making passwords such as Password2026! weaker than they appear. A rainbow-table attack uses precomputed hashes, which is why salts are critical. Credential stuffing uses leaked username-password pairs from one service against other services, exploiting password reuse rather than directly breaking the hash.

Table 2.3: Password Attack Techniques

Attack Type	Description	Relevance to Dissertation
Brute-force attack	Tries possible combinations until a match is found.	Shows why password length and character-space size matter.
Dictionary attack	Uses common words, leaked passwords and known password lists.	Shows weakness of human-created passwords.
Rainbow-table attack	Uses precomputed hashes to speed recovery.	Shows why salting is important.

Hybrid attack	Modifies dictionary words with numbers, symbols and predictable patterns.	Shows why superficial complexity can fail.
Credential stuffing	Reuses leaked credentials across services.	Shows the danger of password reuse.

The most important critical point is that attackers do not usually guess passwords randomly from the full theoretical space. They prioritise likely guesses, leaked credentials and human patterns. This is why password evaluation must consider predictability and not merely character variety. The Password Security Analyzer is therefore positioned as an educational tool that helps users understand password weakness and move toward stronger alternatives.

2.11 Review of Existing Literature

The literature shows broad agreement that password security depends on both user behaviour and system-side controls. However, different sources emphasise different parts of the problem. Standards such as NIST and OWASP provide implementation guidance. Cryptographic specifications define hash functions and password-hashing mechanisms. Usable-security research explains why users choose weak or reused passwords. Password-cracking research explains how attackers exploit patterns in human-created passwords.

Table 2.4: Literature Review Matrix

Author/Source	Focus	Key Finding	Limitation	Relevance to Study
Microsoft [1]	Identity attack trends	Password-based attacks remain dominant in Microsoft Entra	Vendor data reflects Microsoft's ecosystem and may not represent all	Supports the contemporary relevance of password attacks.

		identity-attack data.	environments equally.	
Verizon [2]	Breach trends	Credential risks remain important even as vulnerability exploitation increases.	Broad industry report rather than password-specific research.	Supports the wider breach context.
NIST SP 800-63B [3], [4]	Authentication and password verifier requirements	Passwords should be stored using salted hashing schemes with cost factors.	Standards guidance does not evaluate this dissertation's artefact.	Supports secure password-storage criteria.
OWASP [5]	Password-storage best practice	Passwords should use strong, slow hashing mechanisms such as Argon2id, bcrypt or PBKDF2.	Practice-focused rather than empirical user-behaviour research.	Supports bcrypt and password-hashing recommendations.
Rivest [6]	MD5 specification	Introduced MD5 as a 128-bit	MD5 is outdated for security-sensitive	Provides historical background for MD5 discussion.

		message-digest algorithm.	password storage.	
NIST FIPS 180-4 [7]	Secure Hash Standard	Defines SHA-256 and related secure hash algorithms.	SHA-256 is not specifically a password-hashing scheme.	Supports the distinction between general hashing and password hashing.
Provos and Mazières [8]	bcrypt	Introduced an adaptive-cost password scheme.	bcrypt performance depends on cost configuration and password quality.	Supports bcrypt implementation and evaluation.
Bonneau et al. [11]	Password replacement	Password alternatives face usability, deployability and security trade-offs.	Does not directly evaluate the dissertation artefact.	Explains why passwords remain dominant.
Florêncio et al. [12]	User password portfolios	Users face finite effort when managing many passwords.	Focuses on user behaviour rather than hashing implementation.	Supports password reuse and weak-password discussion.

Weir et al. [13]	Password cracking	Attackers exploit password structures using probabilistic models.	Based on particular datasets and attack assumptions.	Supports critique of superficial complexity.
Wheeler [14]	Password-strength estimation	Realistic meters should consider patterns and attacker-like guessing.	Focuses on estimation rather than secure storage implementation.	Supports password-feedback logic.
RFC 9106 [15]	Argon2	Defines Argon2 as a memory-hard password-hashing function.	Argon2 is discussed as future work, not implemented in the current app.	Supports future enhancement and modern hashing comparison.
Wang and Yu [16]	MD5 cryptanalysis	Demonstrated weaknesses in MD5 collision resistance.	Collision attacks are not identical to password guessing.	Supports the argument that MD5 is obsolete.
Percival [17]	scrypt	Introduced memory-hard key derivation to resist	Not implemented in the current system.	Supports wider password-hashing discussion and future work.

		hardware attacks.		
Dropbox/zxcvbn [18]	Pattern-aware strength estimation	Recognises common passwords, names, dates, sequences, repeats and leetspeak.	Tool documentation is not a standards document.	Supports practical password-strength feedback.

This matrix shows that the dissertation is grounded in both authoritative guidance and academic research. It also shows that password security must be examined across algorithm design, storage configuration, user behaviour, attack strategy and practical communication.

2.12 Research Gap

The literature clearly shows that password security depends on both user-side password strength and system-side hashing design. NIST and OWASP provide clear guidance on salted, cost-based password storage [3]–[5]. Usable-security research explains why users often choose weak or reused passwords [11], [12]. Password-cracking research shows that attackers exploit predictable human patterns rather than relying only on random brute force [13], [14].

However, much of the existing discussion remains theoretical, standards-based or algorithm-focused. There is a practical gap in demonstrating these concepts through an interactive artefact that allows users to observe password weakness, estimated cracking resistance and stronger password suggestions. Users may be told that passwords should be “strong,” but they may not understand how length, entropy, character diversity, salting and adaptive hashing affect security. Similarly, junior developers may know that passwords should be “hashed” without understanding why MD5 and unsalted SHA-256 are unsuitable for password storage.

This dissertation addresses that gap by designing and evaluating a deployed web-based Password Security Analyzer [9]. The artefact is not intended to replace enterprise password-auditing tools or perform offensive cracking. Its contribution is educational and analytical: it connects password characteristics, hashing awareness and stronger password practice in a practical web-based environment.

2.13 Conceptual Framework

The conceptual framework for this dissertation is derived from the reviewed literature. It treats password security as the outcome of five connected elements: user password behaviour, password complexity and entropy, hashing algorithm choice, storage protection controls and security outcome.

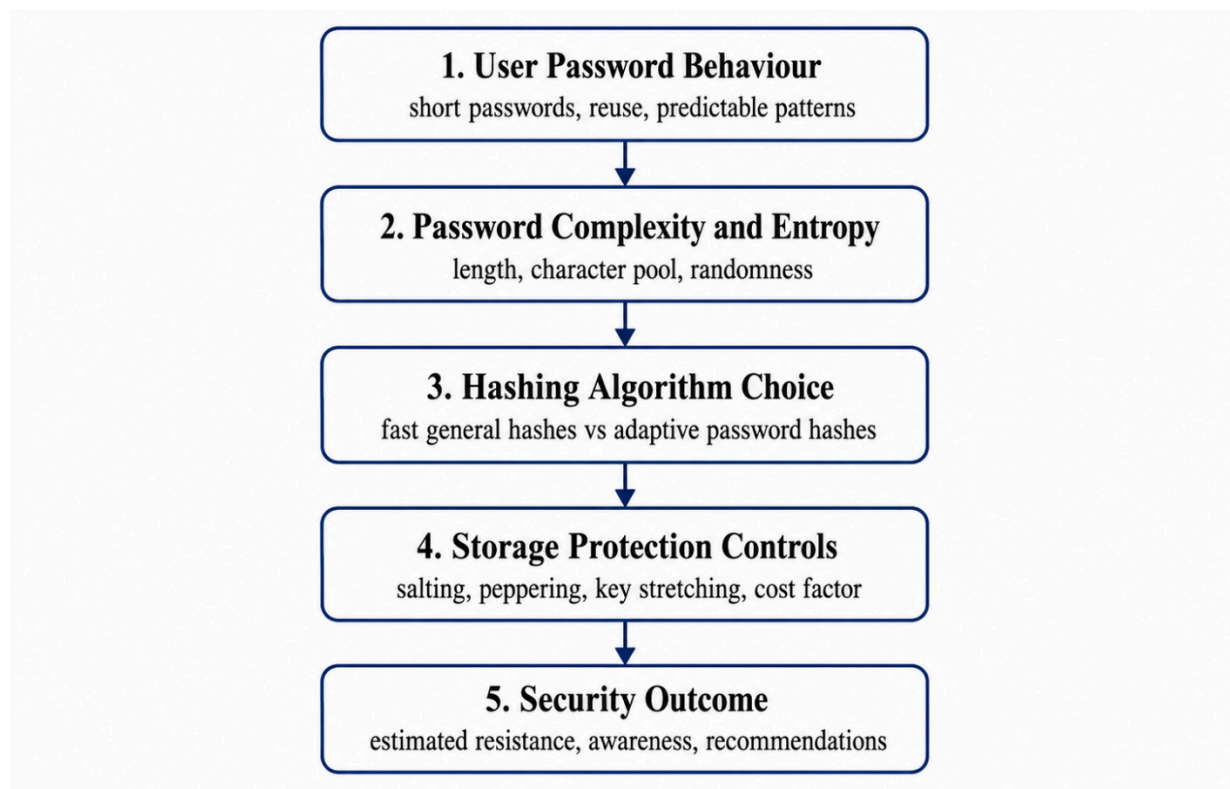


Figure 2.2: Literature-Derived Framework for Password Security

Source: Author's synthesis of password-authentication, password-hashing and usable-security literature.

The framework begins with user password behaviour because passwords are created by people operating under memory and convenience constraints. These behaviours influence password complexity and entropy. Entropy then affects the estimated search space available to an attacker. However, hashing algorithm choice also matters because it determines the computational cost of each guess after a database compromise. Storage protection controls such as salting, peppering, key stretching and adaptive cost further shape the difficulty of offline cracking. The final security outcome is expressed through estimated resistance, user awareness and practical recommendations.

This framework supports the dissertation methodology by linking literature to system evaluation. The Password Security Analyzer can be evaluated according to how effectively it communicates this chain of security reasoning: from password characteristics to estimated cracking resistance and stronger password practice.

2.14 Summary

This chapter critically reviewed the literature on password authentication, weak password behaviour, entropy, hashing, MD5, SHA-256, bcrypt, salting, peppering, key stretching and password attack techniques. The review shows that passwords remain dominant because they are familiar, inexpensive and deployable, but they remain insecure when users choose predictable passwords or when systems store passwords using fast unsalted hashes.

The chapter established that password strength cannot be judged only by visible complexity. Length, randomness, predictability, password reuse, algorithm choice and storage configuration all affect password security. MD5 is unsuitable for password storage because it is outdated and fast. SHA-256 is stronger than MD5 for integrity purposes but remains unsuitable as a standalone password-storage mechanism because it is a fast general-purpose hash. bcrypt is more appropriate because it is salted and adaptive, but it cannot fully compensate for very weak user-created passwords.

The research gap identified in this chapter is the need for a practical, educational artefact that connects password strength, entropy, estimated cracking resistance and hashing awareness.

This gap justifies the implemented Password Security Analyzer and leads directly into Chapter 3, which explains the methodology used to design, implement and evaluate the system.

Chapter 3: Research Methodology

3.1 Introduction

This chapter explains the methodology used to design, implement and evaluate the Password Security Analyzer. It presents the research design, development approach, tools and technologies, test password dataset, evaluation metrics, data collection method, ethical considerations and methodological limitations. The chapter is important because the dissertation is not a purely theoretical discussion of password hashing. It is an applied cybersecurity study based on the development and evaluation of a working web-based artefact.

The methodological challenge in this dissertation is that password security must be evaluated carefully. Password strength, entropy and estimated cracking resistance can be measured or estimated, but they cannot be treated as absolute indicators of real-world attack success. Actual cracking depends on attacker hardware, password dictionaries, leaked credential datasets, GPU acceleration, hash algorithm, salting, cost factors and attack strategy. NIST states that password-hashing schemes should use a password, salt and cost factor to make each guessing attempt more expensive for an attacker who has obtained a hashed password file [4]. OWASP also emphasises the use of password-specific hashing functions such as Argon2id, bcrypt and PBKDF2 rather than fast general-purpose hashes [5]. Therefore, this methodology treats cracking-time outputs as comparative indicators rather than exact predictions.

The chapter adopts a Design Science Research approach because the dissertation develops and evaluates a practical cybersecurity artefact. The artefact is the deployed Password Security Analyzer, which assesses password strength and communicates password-security risk to users. The methodology is therefore structured around three connected activities: building the artefact, testing it with controlled password examples and evaluating its usefulness against password-security criteria.

3.2 Research Design

This dissertation adopts Design Science Research because it creates and evaluates a practical artefact in response to an identified cybersecurity problem. Design Science Research is suitable for information-systems and computing projects where knowledge is produced through the design, development and evaluation of an artefact. Hevner et al. explain that design science seeks to extend human and organisational capabilities by creating innovative artefacts, and that knowledge is achieved through building and applying the designed artefact [23]. This aligns directly with the dissertation because the Password Security Analyzer is not merely described conceptually; it is implemented and deployed as a working system.

Peppers et al. provide a structured Design Science Research Methodology consisting of six stages: problem identification and motivation, objectives of a solution, design and development, demonstration, evaluation and communication [24]. These stages fit the dissertation well because the project begins with the problem of weak passwords and insecure storage practice, then develops a web-based analyser, demonstrates it through controlled testing and communicates the findings through dissertation chapters, tables, figures and recommendations.

Table 3.1: Design Science Research Alignment

Design Science Stage		Application in this Dissertation
Problem identification and motivation		Password insecurity arises from weak user-created passwords and poor password-storage practices.

Objectives of the solution	Develop a tool that analyses password strength, estimates cracking resistance and communicates hashing awareness.
Design and development	Build the Password Security Analyzer using Next.js, React, TypeScript and bcryptjs.
Demonstration	Deploy the application and run controlled password tests using weak, medium and strong examples.
Evaluation	Assess password-length analysis, entropy indication, crack-time estimation, hashing comparison and suggestion quality.
Communication	Present the artefact, findings, limitations and recommendations through the dissertation.

This research design is more appropriate than a purely experimental cracking study because the dissertation does not attempt to breach systems or recover real user passwords. It is also more appropriate than a purely literature-based study because the research contribution is tied to the development and evaluation of a cybersecurity artefact. The artefact-based approach strengthens the dissertation by linking theory, implementation and evaluation.

3.3 Development Approach

The Password Security Analyzer was developed using an iterative software-development approach. This was suitable because the project required repeated refinement of both technical logic and user-facing communication. Password analysis tools must not only calculate or estimate strength; they must also present the results in a way that users can understand. Therefore, the development process involved continuous improvement of the interface, password input handling, strength evaluation, entropy and crack-time estimation, bcrypt hashing demonstration, password suggestions and deployment.

The iterative development process followed seven main stages:

1. interface design;

2. password input implementation;
3. password-strength logic development;
4. entropy and crack-time estimation;
5. bcrypt hashing demonstration;
6. strong password suggestion generation;
7. deployment and controlled testing.

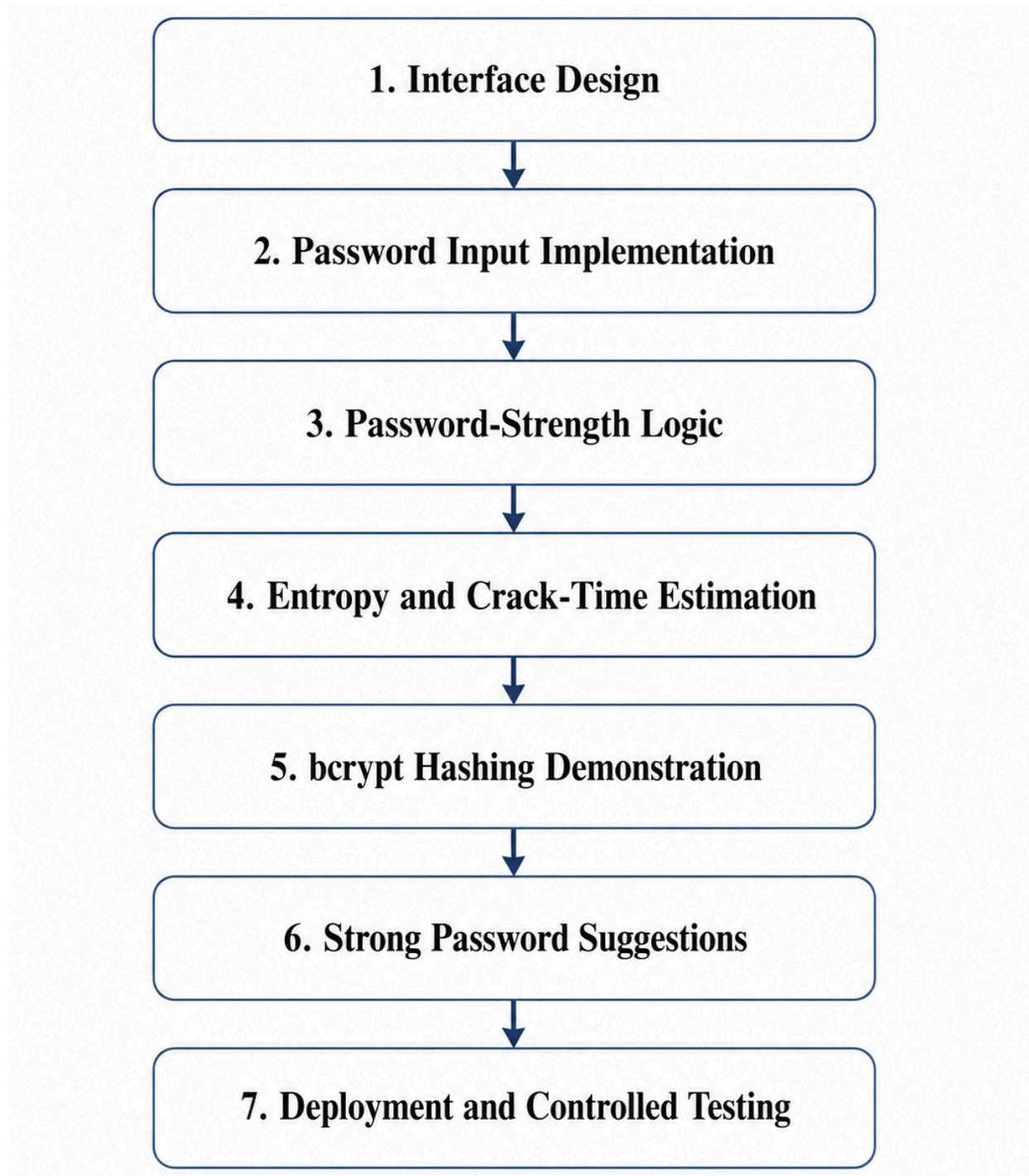


Figure 3.1: Iterative Development Workflow for the Password Security Analyzer

Source: Author's development workflow for the implemented Password Security Analyzer.

This approach was useful because the project required both functional and interpretive testing. For example, if a password was classified as weak, the output needed to be technically reasonable and also clear to the user. Similarly, if the system generated a stronger password suggestion, the suggestion needed to demonstrate improved password characteristics such as length, character diversity and reduced predictability. The iterative approach therefore supported both software functionality and cybersecurity education.

A critical limitation of the development approach is that it did not include a large formal user-experience study. The evaluation focuses on controlled testing and functional review rather than participant-based usability testing. However, this is appropriate for the dissertation scope because the primary objective is to design and evaluate the artefact's password-security logic, not to conduct a large behavioural experiment.

3.4 Tools and Technologies

The Password Security Analyzer was implemented using modern web-development technologies. These technologies were selected because the dissertation required a browser-accessible, interactive and deployable cybersecurity artefact. Next.js was used as the web application framework, React supported component-based interface development, TypeScript supported type-safe logic, Tailwind CSS supported styling and responsive layout, Node.js provided the runtime environment, bcryptjs supported bcrypt hashing, Vercel supported deployment, and Git/GitHub supported version control and project storage.

Next.js was appropriate because its App Router provides a file-system-based routing model and supports modern React features such as Server Components and Server Functions [26]. React was suitable because it allows user interfaces to be built from reusable components, supporting the modular structure needed for input fields, analysis panels, feedback displays and suggestion sections [27]. TypeScript was used because it extends JavaScript by adding types, which helps catch errors and improve maintainability during development [28]. bcryptjs was appropriate because it provides an optimised bcrypt implementation in JavaScript with TypeScript support and compatibility with Node.js and browser-based environments [29].

Table 3.2: Tools and Technologies

Technology	Purpose	Justification
Next.js	Web application framework	Supports structured web application development and deployment-ready architecture.
React	User interface development	Enables reusable components for password input, analysis display and suggestion sections.
TypeScript	Type-safe application logic	Improves maintainability and reduces coding errors through static typing.
Tailwind CSS	Styling and responsive layout	Supports clean interface design and responsive presentation.
Node.js	Runtime environment	Enables JavaScript/TypeScript execution for the application environment.
bcryptjs	bcrypt password hashing	Provides bcrypt hashing functionality in JavaScript/TypeScript.
Vercel	Deployment platform	Supports hosting of the deployed Password Security Analyzer.
Git/GitHub	Version control and project storage	Supports version tracking, project management and code preservation.

The technology choices reflect the applied nature of the dissertation. The project is not intended to be an enterprise authentication suite, but it must still demonstrate professional software-development practice. The chosen stack allowed the development of a web-accessible cybersecurity artefact that could be tested through a deployed interface.

3.5 Test Password Dataset

The study used a controlled dataset of lab-generated passwords. This was necessary because the dissertation involves password-security testing, and using real user passwords would create ethical and privacy risks. The selected passwords represent weak, medium and strong password

categories. This allows the analyser to be tested against different levels of complexity and predictability.

Table 3.3: Password Test Dataset

Password	Category	Reason for Selection
123456	Weak	Common numeric password with very low complexity and high predictability.
password	Weak	Dictionary-based password frequently associated with weak password behaviour.
Parthik123	Medium	Name/text plus numbers; useful for testing predictable mixed-format passwords.
P@rth!k#2026\$	Strong	Mixed-case, symbols and numbers; tests improved complexity and length.
Random generated password	Strong	Tests the strength and usefulness of system-generated password suggestions.

The dataset was intentionally small and controlled. This is methodologically acceptable because the dissertation evaluates a software artefact rather than conducting a large password-cracking experiment. A larger dataset would improve generalisability, but it would also require a broader empirical design, additional controls and potentially ethical approval if human participants or real passwords were involved.

The weak passwords test whether the system correctly identifies highly predictable inputs. The medium password tests a realistic user-created pattern that appears stronger than a dictionary word but remains partly predictable because it combines text and numbers. The strong password and generated password test whether the analyser recognises improved length, character diversity and reduced simplicity. This design allows the dissertation to evaluate whether the application communicates meaningful differences between weak, medium and strong password choices.

3.6 Evaluation Metrics

The Password Security Analyzer was evaluated using metrics derived from the password-security literature reviewed in Chapter 2. The metrics were selected to assess both technical password strength and user-facing communication. This is important because the artefact is not only a calculation tool; it is also intended to improve password-security awareness.

Table 3.4: Evaluation Metrics

Metric	Description	Purpose
Password length	Number of characters in the password.	Measures basic complexity and search-space expansion.
Character diversity	Use of uppercase letters, lowercase letters, numbers and symbols.	Measures variation in the character set.
Entropy estimate	Approximate unpredictability of the password.	Provides a comparative measure of password strength.
Crack-time estimate	Estimated resistance to guessing attacks.	Communicates practical risk to users.
Hashing comparison	Comparison of MD5, SHA-256 and bcrypt.	Evaluates password-storage security awareness.
Suggestion quality	Strength of generated password alternatives.	Assesses whether the tool gives useful improvement guidance.
Interface clarity	Ease with which users can understand the result.	Evaluates awareness and communication effectiveness.

These metrics were selected because password security cannot be reduced to one measurement. Password length and character diversity provide basic indicators of complexity. Entropy estimates provide a more analytical view of possible search space. Crack-time estimates

help communicate risk in practical terms. Hashing comparison links the system to secure password-storage practice. Suggestion quality evaluates whether the tool helps users improve their behaviour.

However, each metric has limitations. Password length does not guarantee security if the password is a common phrase. Character diversity can be misleading if users apply predictable substitutions. Entropy can overestimate security when passwords follow human patterns. Crack-time estimates depend on hardware assumptions and should not be treated as exact. For these reasons, the evaluation uses the metrics critically rather than treating them as definitive proof of security.

3.7 Data Collection Method

Data was collected through controlled testing of the deployed Password Security Analyzer and review of the project source code. The data collection process was designed to produce evidence that could support the implementation and evaluation chapters without collecting sensitive real-world credentials.

The main data sources were:

Data Source	Purpose
Project source code	To verify implemented logic, structure and hashing functionality.
Live deployed application	To confirm that the artefact works in a browser-accessible environment.
Screenshots of test results	To provide visual evidence for the implementation and evaluation chapters.
Manually recorded outputs	To compare password classifications, estimates and suggestions.
Algorithm comparison tables	To support analysis of MD5, SHA-256 and bcrypt.

Literature sources	To ground evaluation criteria in recognised research and guidance.
--------------------	--

Controlled testing involved entering each selected password into the application and recording the observed output. The outputs included strength indication, estimated cracking resistance, hashing-related comparison and password suggestions where applicable. Screenshots were captured to support later discussion in the results chapter. Source-code review was used to confirm that the application’s functionality was consistent with the stated dissertation scope.

Secondary data came from NIST, OWASP, password-hashing literature, Design Science Research literature and official technology documentation. NIST and OWASP provided the security criteria for password storage and hashing [4], [5]. Hevner et al. and Peffers et al. provided the methodological justification for artefact design and evaluation [23], [24]. Official documentation supported the use of Next.js, React, TypeScript and bcryptjs as implementation technologies [26]–[29].

This combination of primary and secondary evidence strengthens the methodology because it links the implemented artefact to recognised academic and professional standards.

3.8 Ethical Considerations

Ethical considerations are central to this dissertation because password security is a sensitive cybersecurity topic. The project was designed for defensive education and analysis, not offensive exploitation. Only lab-created passwords were used for testing. No real user credentials were collected, stored, transmitted or attacked. No live password database was breached, and no unauthorised penetration testing was performed.

The application should not store user-entered passwords. This is important because password analysers can become risky if they log or transmit user passwords. The ethical design assumption in this study is that password analysis should occur without retaining sensitive credentials. Where bcrypt hashing is demonstrated, the purpose is educational: to show how secure password-hashing mechanisms work and why fast hashes are unsuitable for password storage.

The study also avoids providing operational attack guidance. Brute-force, dictionary and rainbow-table attacks are discussed only to explain password-security risk and justify defensive design decisions. The dissertation uses controlled examples such as 123456, password, Parthik123 and P@rth!k#2026\$, ensuring that testing remains within academic and ethical boundaries.

Table 3.5: Ethical Safeguards

Ethical Issue		Safeguard Applied
Real credential exposure		Only lab-created passwords were used.
User privacy		No real user passwords were intentionally collected or stored.
Unauthorised access		No live systems, accounts or databases were attacked.
Offensive misuse		Attack techniques were discussed only for defensive understanding.
Data protection		Screenshots and outputs were based on controlled test examples.
Research transparency		Limitations and estimation assumptions were clearly acknowledged.

This ethical framing is important because cybersecurity research must demonstrate technical understanding without creating harm. The dissertation therefore positions the Password Security Analyzer as a defensive and educational artefact.

3.9 Limitations of Methodology

The methodology has several limitations. First, crack-time values are estimates rather than exact measurements. Real-world cracking time depends on attacker hardware, GPU acceleration, hash algorithm, salt use, cost factor, password dictionaries and attack strategy. This means the analyser’s crack-time outputs should be interpreted as comparative indicators rather than exact predictions.

Second, entropy does not fully capture human predictability. A password may appear mathematically strong because it contains uppercase letters, lowercase letters, numbers and symbols, but it may still follow a predictable structure. For example, a name-plus-year pattern may receive a better score than it deserves if the system does not fully account for dictionary or personal-information patterns.

Third, MD5 and SHA-256 are compared mainly as theoretical or estimated references, while bcrypt is the primary implemented hashing demonstration. This is appropriate because bcrypt is a password-hashing function, but it means the application should not be described as a full cryptographic benchmarking platform.

Fourth, bcrypt is implemented, but Argon2id and PBKDF2 are not yet included in the current version. This limits the breadth of password-hashing comparison. Since OWASP recommends Argon2id as a preferred modern option where available, future development should include Argon2id and PBKDF2 for broader comparison [5], [15].

Fifth, no large-scale user study was conducted. The evaluation focuses on controlled testing and functional review rather than participant-based usability testing. This means the dissertation can evaluate whether the tool communicates risk logically, but it cannot make strong claims about how real users would change their password behaviour after using it. Future research could include pre-test/post-test user-awareness measures, usability questionnaires and statistical analysis of behavioural change.

Finally, the application does not implement a full login and registration system. It is a password-security analyser and educational artefact, not an enterprise identity-management platform. This limitation is appropriate because the dissertation's aim is to evaluate password-strength awareness and hashing principles rather than build a complete authentication infrastructure.

3.10 Summary

This chapter presented the methodology used to design, implement and evaluate the Password Security Analyzer. The dissertation adopts a Design Science Research approach because it

develops and evaluates a practical cybersecurity artefact. The methodology follows recognised Design Science Research stages: problem identification, objectives definition, design and development, demonstration, evaluation and communication [23], [24].

The chapter explained that an iterative development approach was used, moving from interface design to password input, strength logic, entropy and crack-time estimation, bcrypt hashing, strong password suggestions, deployment and controlled testing. It also described the tools and technologies used, including Next.js, React, TypeScript, Tailwind CSS, Node.js, bcryptjs, Vercel and Git/GitHub.

The chapter defined a controlled password test dataset containing weak, medium and strong passwords. Evaluation metrics included password length, character diversity, entropy estimate, crack-time estimate, hashing comparison, suggestion quality and interface clarity. Data collection was based on project source code, the deployed application, screenshots, manually recorded outputs, algorithm comparison tables and literature sources.

Ethical safeguards were also established. The study used only lab-created passwords, avoided real credential collection, did not attack live systems and treated attack techniques as defensive concepts. The chapter also acknowledged limitations, including estimated crack-time values, entropy limitations, limited hashing comparison, absence of Argon2id and PBKDF2 implementation, and lack of large-scale user testing.

The next chapter presents the system design and implementation of the Password Security Analyzer, including its architecture, functional requirements, interface design, password-strength logic, hashing comparison and deployment.

Chapter 4: System Design and Implementation

4.1 Introduction

This chapter presents the system design and implementation of the Password Security Analyzer, the practical cybersecurity artefact developed for this dissertation. The application was designed as a deployed web-based tool that allows users to analyse password strength, view estimated cracking resistance, observe bcrypt hashing output and receive stronger password

suggestions. The purpose of this chapter is to explain how the system was designed, how its components interact and how the implementation supports the dissertation's investigation into password strength and secure password-storage awareness.

The system responds directly to the problem identified in the literature review. Chapter 2 established that password security is shaped by both user behaviour and system-side implementation. Users frequently create passwords that appear complex but remain predictable, while developers may incorrectly assume that any cryptographic hash function is sufficient for password storage. NIST recommends that memorised secrets should be protected using salted password-hashing schemes with appropriate cost factors, while OWASP recommends password-specific hashing mechanisms such as Argon2id, bcrypt, scrypt or PBKDF2 instead of fast general-purpose hashes such as MD5 or unsalted SHA-256 [4], [5]. The Password Security Analyzer was therefore implemented to make these concepts visible through an interactive web-based system.

This chapter explains the system overview, functional requirements, non-functional requirements, tools and technologies, architecture, user interface design, password-strength logic, entropy calculation, crack-time estimation, hashing comparison, bcrypt implementation, strong password suggestion feature, deployment, security considerations and implementation limitations. The chapter includes two screenshots from the deployed website: one showing the landing interface before analysis, and one showing the interface after a password has been entered and analysed.

4.2 System Overview

The Password Security Analyzer is a browser-based application developed using Next.js, React, TypeScript, Tailwind CSS, Node.js, bcryptjs, Git/GitHub and Vercel. The application provides a user-facing interface through which a user can enter a password and receive feedback on its security strength. It also generates stronger password suggestions and demonstrates bcrypt hashing. This combination of features supports the dissertation aim because it connects password behaviour, password-strength estimation and password-storage awareness in a single practical artefact.

The deployed application contains two main functional areas. The first is the Password Security Analyzer section, where the user enters a password and triggers analysis. The second is the Strong Password Suggestions section, where the system displays suggested password alternatives. These two areas are important because the system does not only classify a password as weak, medium or strong; it also attempts to help users understand and improve password quality [9].

The implementation is structured around two central source-code areas. The frontend and password-analysis logic are implemented in `app/page.tsx`. This file handles password input, password-strength calculation, entropy estimation, crack-time estimation and password suggestion generation. The backend hashing logic is implemented in `app/api/hash/route.ts`, where the system generates a bcrypt hash using the `bcryptjs` package. This design separates the visible password-analysis interface from the hashing route, which is useful because bcrypt hashing represents a security-specific function.

The system is not a full authentication platform. It does not register users, store passwords in a database, manage sessions or implement multi-factor authentication. This limitation is intentional. The artefact is designed as an educational and analytical password-security tool rather than an enterprise identity-management system. Its purpose is to demonstrate password-security principles in a controlled and accessible way.

4.3 Functional Requirements

Functional requirements define the main operations that the system must perform. The Password Security Analyzer was designed to accept password input, analyse the password, estimate strength-related indicators, generate bcrypt output and provide stronger password suggestions. These requirements are closely aligned with the dissertation objectives because they support the evaluation of password complexity, entropy, cracking resistance and hashing awareness.

Table 4.1: Functional Requirements

Requirement ID	Requirement	Description
FR1	Password input	The system allows the user to enter a password into a text field.
FR2	Password analysis	The system analyses the entered password after the user clicks the analysis button.
FR3	Character-class detection	The system detects lowercase letters, uppercase letters, numbers and special characters.
FR4	Entropy calculation	The system estimates password entropy using password length and detected character-set size.
FR5	Strength classification	The system classifies passwords as Weak, Medium or Strong.
FR6	Crack-time estimation	The system estimates cracking resistance for MD5, SHA-256 and bcrypt assumptions.
FR7	bcrypt hashing	The system generates a bcrypt hash through a backend API route.
FR8	Strong password suggestions	The system generates stronger password alternatives.
FR9	Visual result display	The system displays strength level, entropy, estimated crack times, progress bar and bcrypt output.
FR10	Web deployment	The system is accessible as a deployed web application.

These functional requirements show that the system is not limited to password scoring. It combines password analysis with hashing awareness and improvement guidance. This is

important because password-security education should not only identify weakness; it should also show users how to improve password choices and why secure hashing mechanisms matter.

4.4 Non-Functional Requirements

Non-functional requirements define the qualities that the system should demonstrate. For a password-security tool, these qualities are particularly important because the system handles password-like input and presents security advice. Even though the application is intended for controlled testing and educational use, it still needs to be designed with security, usability and transparency in mind.

Table 4.2: Non-Functional Requirements

Requirement	Explanation
Security	The system should not collect, store or expose real user credentials.
Usability	The interface should present password feedback in a clear and understandable way.
Responsiveness	The layout should display properly across common screen sizes.
Maintainability	TypeScript and modular source files should support clearer development and future updates.
Performance	Password analysis should occur quickly enough for interactive use.
Transparency	The system should clearly display strength, entropy, crack-time estimates and hashing output.
Deployability	The system should be deployable online through a reliable hosting platform.
Educational value	The system should help users understand why some passwords are weak and how they can improve them.

The most important non-functional requirement is security. A password analyser can become problematic if users enter real credentials and the application logs or stores them. For this reason, the system should be framed clearly as a demonstration tool for lab-generated passwords. A future version should include a visible warning instructing users not to enter real passwords. This would strengthen the ethical and security design of the application.

4.5 Tools and Technologies Used

The system was developed using a modern web-development stack. Next.js was selected as the web application framework because it supports structured application development and deployment-ready architecture. React was used for user-interface development because it enables reusable components and dynamic interaction. TypeScript was used to improve code maintainability by adding static typing to JavaScript. Tailwind CSS supported styling and responsive layout. Node.js provided the runtime environment, bcryptjs enabled bcrypt hashing, Git/GitHub supported version control, and Vercel was used for deployment.

Table 4.3: Tools and Technologies Used

Technology	Purpose	Justification
Next.js	Web application framework	Supports modern web application structure and deployment-ready development [26].
React	User interface development	Enables reusable interface components and dynamic user interaction [27].
TypeScript	Type-safe logic	Improves code maintainability by adding static typing to JavaScript [28].
Tailwind CSS	Styling and responsive design	Supports clean layout, visual consistency and responsive interface design.
Node.js	Runtime environment	Enables JavaScript/TypeScript execution for application logic.

bcryptjs	bcrypt password hashing	Provides bcrypt hashing functionality in JavaScript/TypeScript [29].
Vercel	Deployment platform	Supports deployment and hosting of Next.js applications [30].
Git/GitHub	Version control	Supports project storage, version tracking and code management.

The selected technologies are appropriate because the artefact needed to be interactive, browser-accessible and deployable. At the same time, the technology stack defines the system's scope. The application can demonstrate password-strength analysis and bcrypt hashing, but it does not include all features expected in a production identity-management platform, such as database-backed registration, account recovery, MFA, rate limiting and session handling.

4.6 System Architecture

The system architecture follows a lightweight web-application model. The user interacts with the application through a browser. The frontend, built with Next.js and React, handles password input, result presentation and most of the analysis logic. The password-strength logic checks password characteristics, estimates entropy and compares cracking resistance under different algorithmic assumptions. The backend API route `/api/hash` performs bcrypt hashing using bcryptjs.

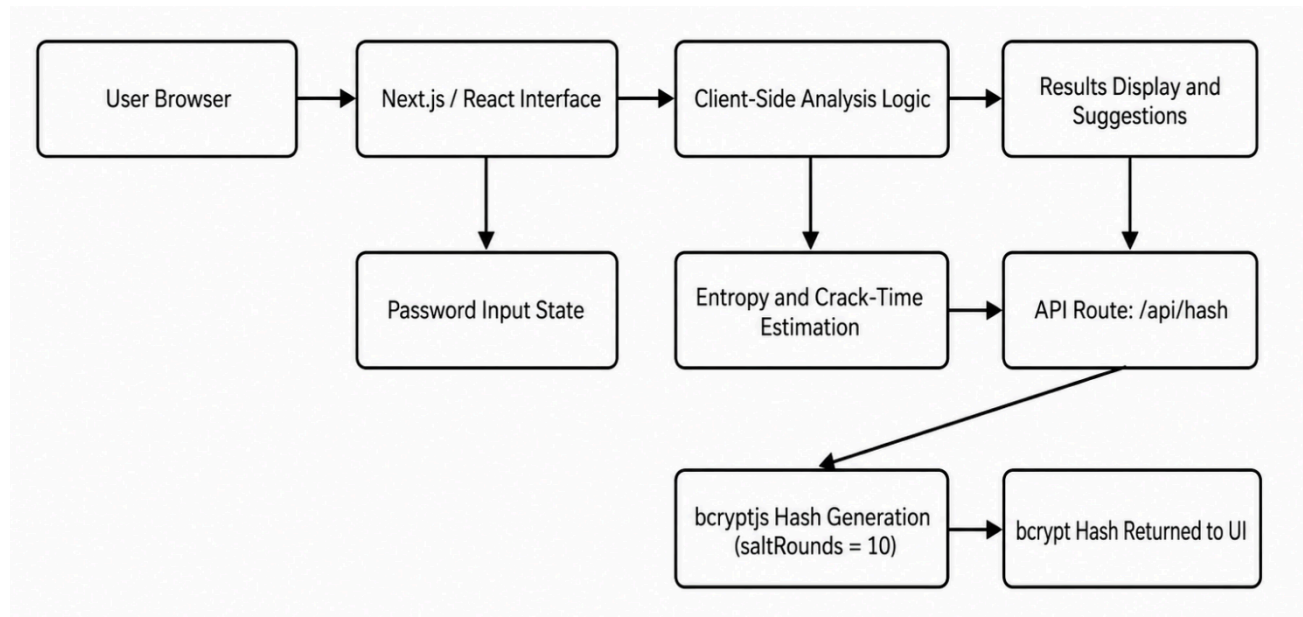


Figure 4.1: System Architecture of the Password Security Analyzer

Source: Author's synthesis based on the implemented project source code and deployed Password Security Analyzer.

The architecture has three main logical areas. The first is the user-interface layer, which captures password input and displays feedback. The second is the client-side analysis layer, which estimates password strength, entropy and crack-time resistance. The third is the backend hashing route, which generates the bcrypt hash. This separation is useful because it distinguishes visible user feedback from the hashing demonstration.

The architecture is intentionally lightweight. It does not include database storage, login sessions, registration or MFA. This is suitable for the dissertation because the system is designed as a password-security analyser, not as a complete authentication service. However, if the application were developed into a production system, it would require additional safeguards such as secure database storage, server-side validation, rate limiting, logging controls and formal security testing.

4.7 User Interface Design

The user interface was designed to make password-security feedback understandable. The deployed application presents password analysis and strong password suggestions through a clear two-panel web interface. The password input field allows the user to enter a test password, while the Analyze Password button triggers the analysis process. The output displays password strength, entropy, estimated crack times and bcrypt hash output. The interface therefore connects password characteristics with visible security feedback.

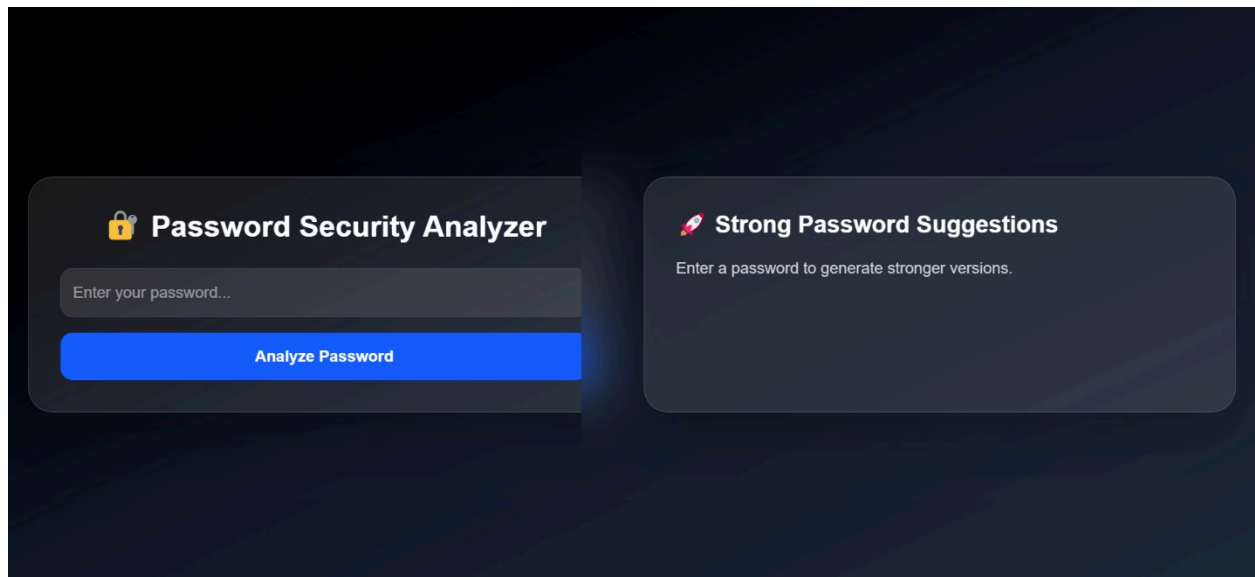


Figure 4.3: Deployed Password Security Analyzer Landing Interface

Source: Author's deployed Password Security Analyzer application.

Figure 4.3 shows the initial landing interface of the deployed Password Security Analyzer. The interface is divided into two main sections: the password analysis panel on the left and the strong password suggestion panel on the right. At this stage, no password has been entered. The system therefore displays the password input field, the Analyze Password button and an instruction prompting the user to enter a password to generate stronger versions. This design supports usability because it makes the two main functions of the system visible before interaction begins.

The two-panel layout is useful because it separates assessment from improvement. The left panel is concerned with evaluating the entered password, while the right panel is concerned with generating stronger alternatives. This reflects the dissertation's argument that

password-security tools should not only identify weakness; they should also provide users with practical guidance for improvement.

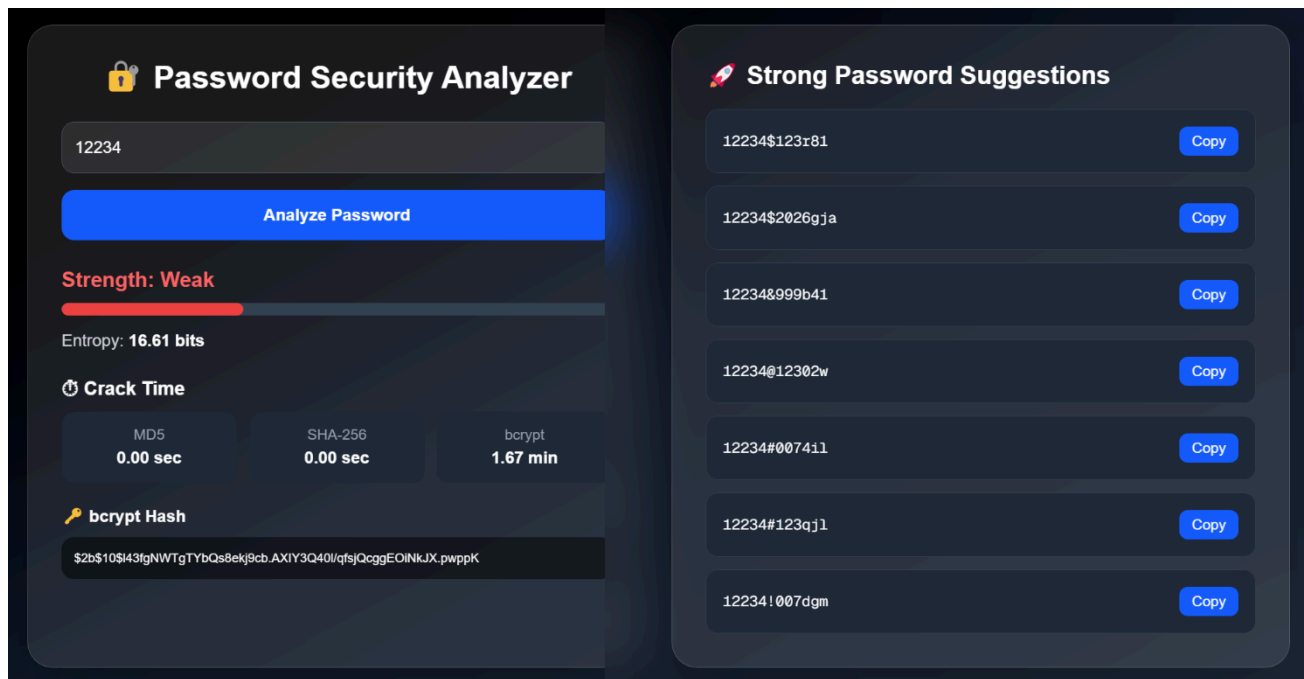


Figure 4.4: Password Analysis Result After Entering a Test Password

Figure 4.4 shows the application after the controlled test password 12234 has been entered and analysed. The system classifies the password as Weak, reports an entropy value of 16.61 bits, and displays estimated crack times for MD5, SHA-256 and bcrypt. The result demonstrates that short numeric passwords provide a small search space and therefore have low resistance to guessing attacks. The screenshot also shows the generated bcrypt hash output and the strong password suggestions displayed on the right-hand panel.

The screenshot is valuable because it demonstrates the implemented system's full interaction cycle. It shows password input, analysis, entropy output, crack-time estimation, bcrypt hash generation and password suggestion output in one interface. This supports the claim that the artefact is not merely a static interface but a working password-security analyser.

However, the results should be interpreted critically. The crack-time values are educational estimates, not exact predictions of real-world cracking time. Actual cracking performance

depends on attacker hardware, GPU acceleration, attack software, salting, hash configuration and password dictionaries. Nevertheless, the screenshot provides strong design evidence because it shows how the implemented system communicates password weakness and encourages stronger password choices.

4.8 Password Analysis Process Flow

The password-analysis process begins when a user enters a password and clicks the analysis button. The system then checks the password for character classes, calculates estimated entropy, classifies strength, estimates crack time under different hashing assumptions, generates a bcrypt hash through the API route, and displays stronger password suggestions. This process links user input to both technical analysis and educational feedback.

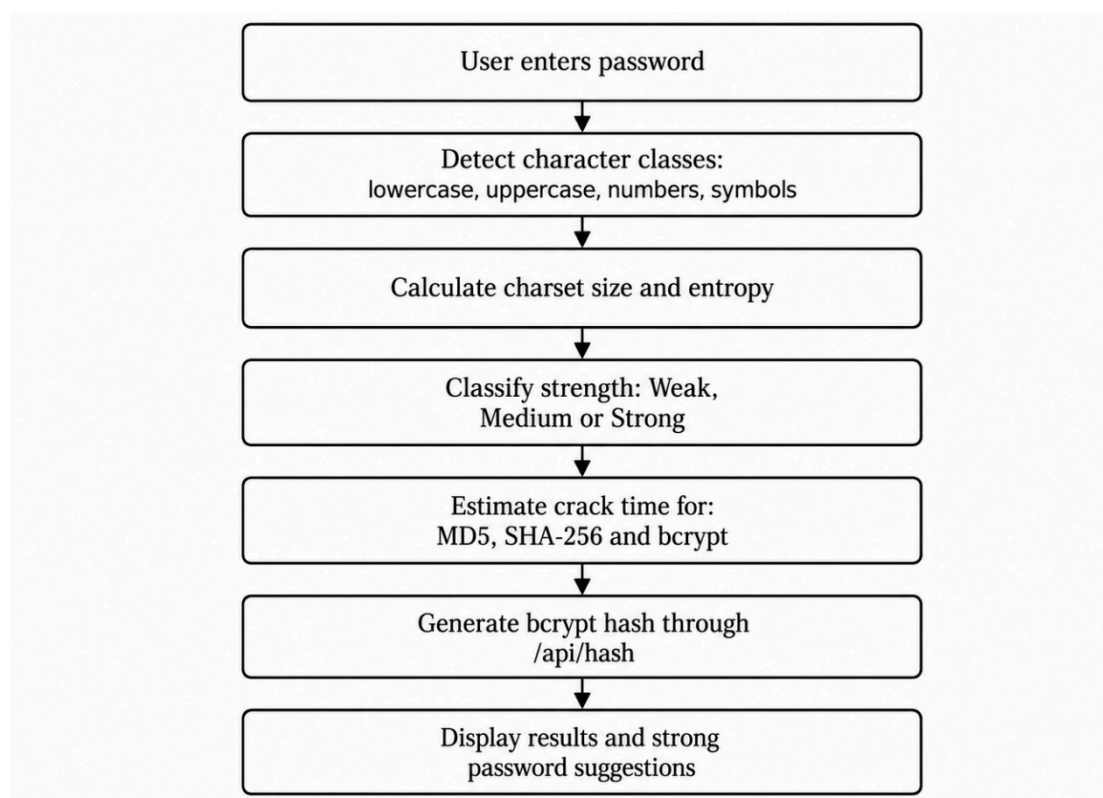


Figure 4.2: Password Analysis and Hashing Process Flow

Source: Author's synthesis based on the implemented project source code.

Figure 4.2 demonstrates that the system performs several connected operations rather than simply displaying a password score. It also shows that the bcrypt hash generation is handled separately through the API route. This distinction is important because the dissertation evaluates both password-strength awareness and secure password-hashing demonstration.

4.9 Password-Strength Logic

The password-strength logic checks whether the entered password contains lowercase letters, uppercase letters, numbers and special characters. The system then calculates a character-set size based on the detected character categories. Lowercase letters add 26 possible characters, uppercase letters add 26, numbers add 10 and special characters add approximately 32. The estimated entropy is then calculated using password length and character-set size.

$$\text{entropy} = \text{password length} \times \log_2(\text{character set size})$$

The password is classified according to entropy thresholds. Passwords with low entropy are classified as Weak, passwords with moderate entropy are classified as Medium, and passwords with higher entropy are classified as Strong.

Table 4.4: Password-Strength Classification Logic

Entropy Range	Strength Classification	Interface Colour	Score Used
≤ 40 bits	Weak	Red	33
> 40 and ≤ 60 bits	Medium	Yellow	66
> 60 bits	Strong	Green	100

This logic is useful because it demonstrates how password length and character diversity affect theoretical strength. However, it must be interpreted critically. Entropy based only on character-set size can overestimate password strength when the password follows a predictable human pattern. For example, a password such as Password2026! may contain uppercase letters, lowercase letters, numbers and a symbol, but it remains vulnerable to hybrid dictionary attacks because it uses a common word and a predictable year suffix. Therefore, the algorithm is

suitable for educational demonstration but should not be treated as equivalent to an advanced password-cracking model.

4.10 Entropy Calculation Logic

Entropy estimation is included to quantify password unpredictability. In the implemented system, entropy is estimated using password length and the size of the detected character pool. A password that uses more character classes and greater length creates a larger theoretical search space. This supports the educational function of the tool because users can see why longer and more diverse passwords are generally stronger.

Table 4.5: Character Set Detection Logic

Detected Character Type	Example	Character Pool Added
Lowercase letters	a–z	26
Uppercase letters	A–Z	26
Numbers	0–9	10
Special characters	@, #, \$, !	32

The limitation is that entropy assumes randomness. Many passwords are not random; they are built from names, dates, words and familiar substitutions. This means an entropy estimate can be mathematically high while practical resistance remains lower. Future versions of the application could improve this by integrating dictionary checks, keyboard-pattern detection, repeated-character checks, leaked-password screening or zxcvbn-style pattern analysis [14], [18].

4.11 Crack-Time Estimation Logic

The application estimates cracking time under three hashing assumptions: MD5, SHA-256 and bcrypt. The purpose of this feature is to demonstrate how algorithm speed affects offline cracking resistance. MD5 and SHA-256 are included as examples of fast general-purpose hashing, while bcrypt represents a deliberately slower password-hashing approach.

Table 4.6: Crack-Time Estimation Assumptions

Algorithm	Assumed Guessing Speed	Interpretation
MD5	100,000,000,000 guesses/second	Represents very fast general-purpose hashing.
SHA-256	10,000,000,000 guesses/second	Represents fast general-purpose hashing.
bcrypt	1,000 guesses/second	Represents deliberately slower adaptive password hashing.

The comparison supports the dissertation’s main security argument. Fast hashes benefit attackers during offline cracking because they allow many guesses to be tested per second. Slow, cost-based password-hashing functions increase the cost of each guess. This aligns with OWASP’s recommendation to use password-specific hashing functions rather than fast general-purpose hashes for password storage [5].

However, these crack-time values should be treated as estimates. Actual cracking performance depends on hardware, GPU acceleration, attack software, salting, implementation details and bcrypt cost factor. The system therefore uses crack-time estimation as a communication tool rather than a precise forensic measurement.

4.12 Hashing Algorithm Comparison

The system compares MD5, SHA-256 and bcrypt to demonstrate the difference between fast general-purpose hashing and adaptive password hashing. MD5 and SHA-256 are used as contrast cases, while bcrypt is implemented as the secure hashing demonstration.

Table 4.7: Hashing Algorithm Comparison in the System

Algorithm	Role in the Application	Security Interpretation
-----------	-------------------------	-------------------------

MD5	Used in crack-time comparison.	Demonstrates the weakness of very fast hashing for password storage.
SHA-256	Used in crack-time comparison.	Demonstrates that stronger general-purpose hashes can still be unsuitable when used alone for passwords.
bcrypt	Implemented through API hash route.	Demonstrates salted, adaptive password hashing with computational cost.

This section must be phrased carefully. SHA-256 is not universally insecure. It remains appropriate for many integrity-related purposes [7]. The problem is that password storage requires resistance to offline guessing, which is why NIST and OWASP recommend salted and cost-based password-hashing schemes [4], [5]. bcrypt is therefore more appropriate for password storage because it is adaptive and deliberately slower [8].

4.13 bcrypt Implementation

The bcrypt hashing function is implemented through the API route `app/api/hash/route.ts`. The route receives a password through a POST request, checks whether the password is provided, and then generates a bcrypt hash using `bcrypt.hash(password, saltRounds)`. The implemented cost setting is `saltRounds = 10`. The route then returns a JSON response containing the generated hash, algorithm name and cost factor.

Table 4.8: bcrypt API Route Logic

Step	Implementation Detail
Receive request	API route receives a POST request containing the password.
Validate input	If no password is provided, the route returns an error.
Set cost factor	The route sets <code>saltRounds = 10</code> .
Generate hash	<code>bcryptjs</code> generates the bcrypt hash.

Return response	The route returns the hash, algorithm and cost value as JSON.
-----------------	---

bcrypt is appropriate because it was designed as an adaptive password-hashing scheme [8]. The bcryptjs package supports bcrypt implementation in JavaScript/TypeScript environments [29]. However, displaying the generated bcrypt hash is appropriate only for educational demonstration. In a production authentication system, the hash would normally be stored securely in a database and not displayed to the user.

Future work should consider adding Argon2id because OWASP recommends Argon2id as a preferred modern password-hashing option where available [5], [15].

4.14 Strong Password Suggestion Feature

The application includes a Strong Password Suggestions feature. This function is significant because a password-security tool should not only identify weak passwords; it should also support improvement. After password analysis, the system generates stronger alternatives using the entered password as a base and combining it with capitalisation, symbols, numeric sequences and random suffixes.

Table 4.9: Password Suggestion Logic

Component	Example	Purpose
Original password base	12234	Uses the user input as a starting point.
Special character	\$, @, #, !	Adds symbol diversity.
Number sequence	123, 2026, 999, 007	Adds numeric variation.
Random suffix	r81, gja, b41	Adds a limited level of unpredictability.

The suggestion feature improves the usability value of the application, but it also has limitations. Since the generated suggestions may still be partly based on the original password, they can inherit predictable elements. For example, suggestions based on 12234 still retain the original numeric sequence. A more advanced version should generate fully random passwords

or passphrases and evaluate each suggestion before displaying it. Pattern-aware strength estimation could also be integrated in future development [14], [18].

4.15 Deployment

The Password Security Analyzer was deployed using Vercel. Vercel is suitable for this project because it supports deployment and hosting of Next.js applications [30]. Deployment is significant because it demonstrates that the system is not only a local prototype. The application can be accessed through a browser and used for controlled testing [9].

Deployment also creates ethical responsibilities. Since the system accepts password input, users should be warned not to enter real passwords. The application should be used only with lab-generated test passwords. In its current form, the app is suitable for dissertation demonstration, but a production version would need clearer user warnings, stronger privacy controls and formal security review.

4.16 Security Considerations

The application was developed as a defensive educational tool. Its main security consideration is that it should not collect, store or expose real user passwords. Although bcrypt hashing is implemented, the system should still be used only with test passwords. Password analysers can create privacy risks if users mistakenly enter real credentials.

The inclusion of bcrypt is a positive design decision because it demonstrates the difference between fast general-purpose hashing and password-specific hashing. The use of saltRounds = 10 demonstrates cost-based hashing, although production systems should periodically review the cost factor as hardware performance changes. NIST and OWASP both emphasise that password storage should make offline guessing expensive [4], [5].

A future version of the system should include a visible warning, a password-type input field with a show/hide option, confirmation that entered passwords are not stored, rate limiting for public API routes, Argon2id comparison and privacy-preserving compromised-password screening. These improvements would make the application stronger if it were developed beyond the dissertation prototype.

4.17 Implementation Limitations

The implementation has several limitations. First, the crack-time estimation uses fixed assumed guessing speeds for MD5, SHA-256 and bcrypt. Real cracking speeds vary depending on hardware, GPU acceleration, algorithm implementation, salt usage and cost factor. Therefore, the values should be interpreted as educational estimates rather than exact predictions.

Second, the entropy calculation is based mainly on password length and detected character-set size. This can overestimate the strength of passwords that follow predictable human patterns. A password made from a name, year and symbol may score higher than it deserves because the model does not fully account for dictionary words or personal information.

Third, bcrypt is implemented, but Argon2id, PBKDF2 and scrypt are not included in the current version. This limits the breadth of password-hashing comparison. Since OWASP recommends Argon2id where available, adding Argon2id would improve the system's technical completeness [5], [15].

Fourth, the password suggestion function may still produce suggestions based on the original weak password. This means that suggestions can retain predictable elements. Fully random password generation or passphrase generation would be more secure.

Finally, the system does not implement full registration, login, password verification, database storage, MFA or session management. This is appropriate for the dissertation scope because the project is a password-security analyser, not a complete authentication platform.

4.18 Summary

This chapter presented the system design and implementation of the Password Security Analyzer. The application was developed using Next.js, React, TypeScript, Tailwind CSS, Node.js, bcryptjs, Git/GitHub and Vercel. The frontend handles password input, strength analysis, entropy calculation, crack-time estimation and strong password suggestions. The backend API route demonstrates bcrypt hashing using bcryptjs.

The chapter explained the system's functional requirements, non-functional requirements, tools and technologies, architecture, interface design, password-strength logic, entropy calculation, crack-time estimation, hashing comparison, bcrypt implementation, password suggestion feature, deployment, security considerations and implementation limitations. Only two screenshots were included in the chapter: the landing interface before password entry and the interface after analysing the controlled password 12234.

The implementation supports the dissertation's aim because it demonstrates how password characteristics and hashing mechanisms influence password-security awareness. However, the system remains an educational analyser rather than a production authentication system. The next chapter evaluates the system using controlled weak, medium and strong password examples and discusses the findings in relation to the research questions.

Chapter 5: Testing, Results and Evaluation

5.1 Introduction

This chapter evaluates the implemented Password Security Analyzer using controlled password tests. The purpose of the chapter is to determine whether the application meets its intended function as a password-security awareness tool. The evaluation focuses on how the system responds to weak, medium and strong passwords, how it communicates entropy and estimated cracking resistance, how it demonstrates bcrypt hashing, and how useful the generated password suggestions are for guiding users toward stronger password practices.

The chapter builds directly on Chapter 4, which explained the system design and implementation. The evaluation does not attempt to perform real-world password cracking or attack any live system. Instead, it uses lab-generated passwords to test whether the application produces meaningful outputs and whether those outputs align with password-security principles established in the literature. This is important because password-cracking resistance is affected by many external factors, including attacker hardware, GPU acceleration, password dictionaries, salting, hashing algorithm and cost factor. Therefore, the crack-time results in this chapter are treated as comparative indicators rather than exact real-world measurements.

The evaluation uses the controlled password categories introduced in Chapter 3: weak passwords, a medium password, a strong password and system-generated suggestions. This allows the chapter to examine whether the system distinguishes between low-complexity passwords and stronger alternatives. The findings are then discussed against the dissertation's research questions and the wider literature on password behaviour, entropy, hashing and secure storage.

5.2 Testing Strategy

The testing strategy used controlled functional testing. This means that selected passwords were entered into the deployed application and the resulting outputs were recorded and interpreted. The purpose was not to test a large population of users, but to evaluate whether

the artefact performs its intended functions correctly and communicates password-security risk clearly.

The test process involved entering each password into the deployed Password Security Analyzer, clicking the Analyze Password button, observing the strength classification, recording the entropy value, reviewing the estimated crack times for MD5, SHA-256 and bcrypt, checking whether a bcrypt hash was generated, and reviewing the generated stronger password suggestions. Screenshots should be inserted in this chapter as evidence of controlled testing. Unlike Chapter 4, where screenshots were used to show the implemented interface, Chapter 5 screenshots should demonstrate test results.

Table 5.1: Testing Strategy

Test Area	Purpose	Evidence Used
Functional testing	To check that core functions work as intended.	Interface response, output display and hash generation.
Weak password testing	To assess whether the system detects low-strength passwords.	Results for 123456 and password.
Medium password testing	To test predictable mixed-character passwords.	Result for Parthik123.
Strong password testing	To assess response to higher-complexity passwords.	Result for P@rth!k#2026\$.
Hashing demonstration	To verify bcrypt hash generation.	bcrypt output shown after analysis.
Suggestion testing	To assess whether stronger alternatives are generated.	Suggested password list after analysis.
Usability review	To assess clarity of output and user guidance.	Layout, colour coding and result interpretation.

The testing strategy is appropriate for this dissertation because the research aim is to evaluate a password-security analyser rather than conduct a full penetration test or password-cracking benchmark. However, the strategy has limitations. The sample size is small, and the evaluation does not include a large user study. It is therefore best understood as controlled artefact evaluation rather than generalisable behavioural research.

5.3 Functional Testing

Functional testing assessed whether the core features of the Password Security Analyzer operated as expected. The tested functions included password input, analysis trigger, strength classification, entropy display, crack-time estimation, bcrypt hash generation and strong password suggestion output. The system successfully responded to entered passwords and displayed password-security feedback after the analysis button was clicked.

Table 5.2: Functional Testing Results

Function	Expected Behaviour	Observed Result	Status
Password input	Users can enter a password into the input field.	Input field accepted test passwords.	Pass
Analyze Password button	The system analyses the entered password after clicking.	Results were generated after clicking the button.	Pass
Strength classification	The system classifies passwords as Weak, Medium or Strong.	The strength result was displayed.	Pass
Entropy display	The system displays entropy in bits.	Entropy value was shown after analysis.	Pass
Crack-time estimate	The system displays MD5, SHA-256 and bcrypt estimates.	Crack-time values were displayed.	Pass

bcrypt hash generation	The system generates a bcrypt hash.	bcrypt hash was displayed after analysis.	Pass
Strong password suggestions	The system generates suggested alternatives.	Suggestions appeared in the right-hand panel.	Pass

The functional testing confirms that the application operates as a working artefact. However, passing functional tests does not automatically mean that the system is fully secure or production-ready. Functional testing only confirms that the implemented features behave as expected in the controlled test environment. Security claims must therefore remain carefully limited. The system demonstrates password-strength analysis and hashing awareness, but it does not provide complete enterprise authentication, password breach checking, MFA or professional password auditing.

5.4 Weak Password Testing

Weak password testing examined whether the analyser correctly identifies passwords with low complexity and high predictability. The weak test examples were 123456 and password. These examples were selected because they represent two common password weaknesses: numeric simplicity and dictionary-word predictability.

The password 123456 is weak because it is short, numeric and widely known. It has very limited character diversity and a small search space. A password composed only of digits is easier to brute-force because the possible character set is much smaller than a password using lowercase letters, uppercase letters, numbers and symbols. If the application classifies 123456 as weak, this is consistent with password-security theory and NIST's emphasis on avoiding compromised or easily guessed passwords [3], [4].

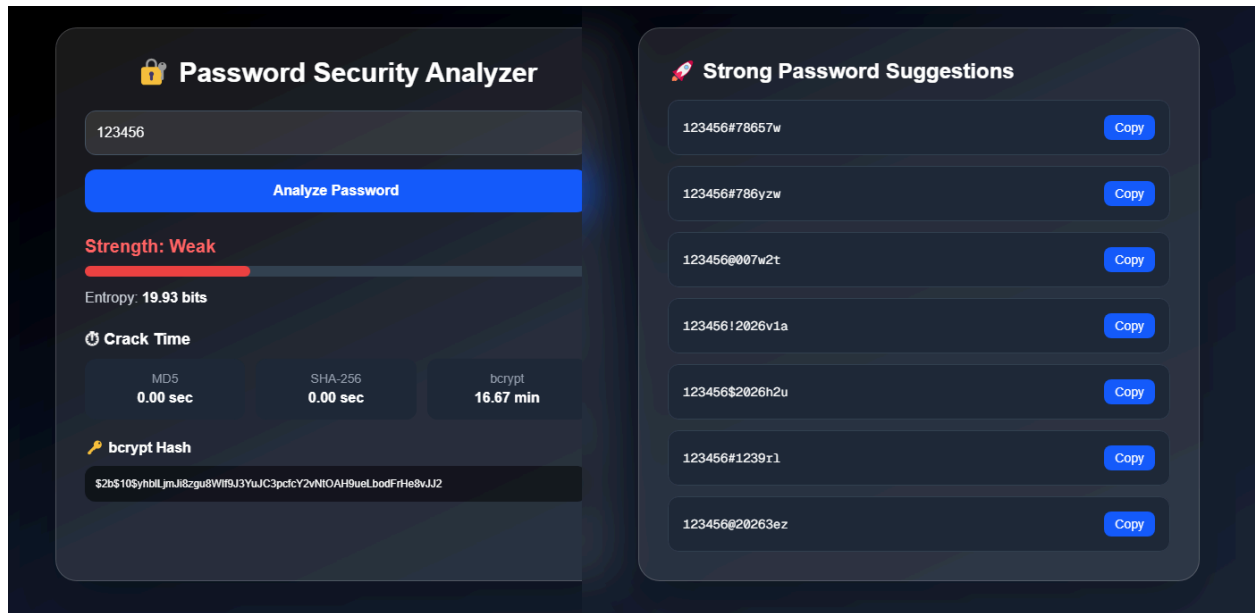


Figure 5.1: Weak Password Test Result for 123456

Source: Author's controlled testing of the deployed Password Security Analyzer.

The password is also weak, but for a different reason. It is longer than 123456, but it is a dictionary word. This demonstrates why length alone is not sufficient. A dictionary password is likely to appear early in password-cracking lists. Attackers rarely rely only on random brute-force attempts; they often begin with common passwords, dictionary words and known leaked credentials [13], [14]. Therefore, a password such as password is weak even if it contains more characters than a short numeric password.

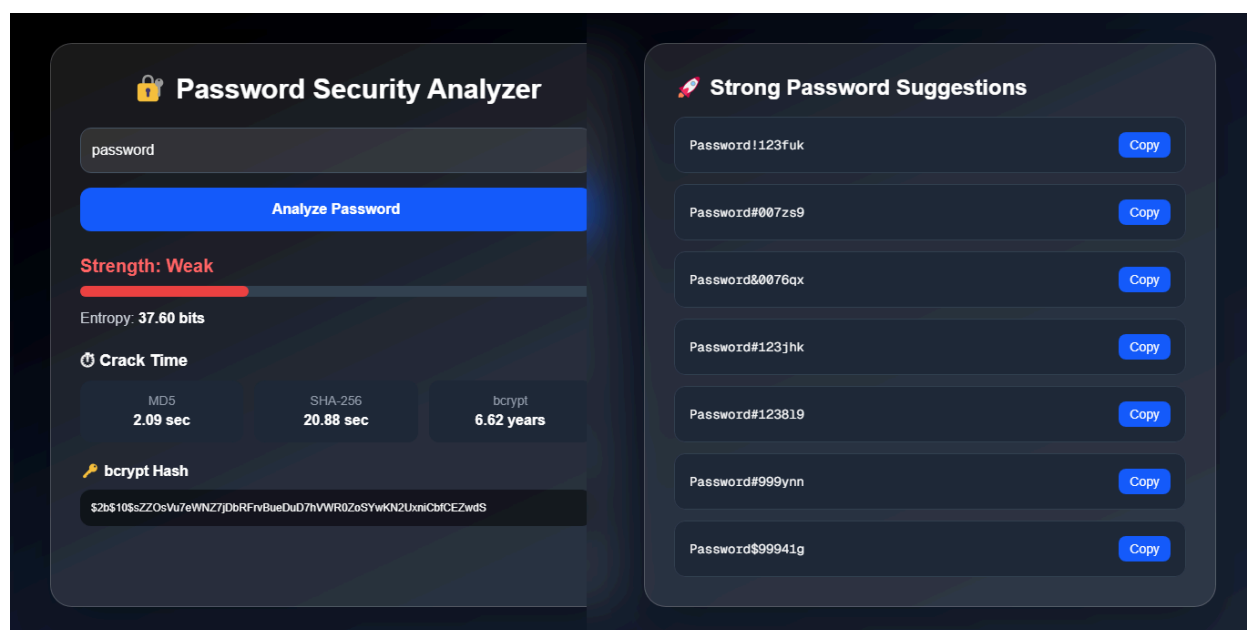


Figure 5.2: Weak Password Test Result for password

Source: Author's controlled testing of the deployed Password Security Analyzer.

The weak password results support the argument that password security depends on both search-space size and predictability. A password analyser should not encourage users to believe that a password is safe merely because it has a minimum length. It must also communicate that common words and predictable structures remain vulnerable. The implemented application demonstrates this by classifying weak inputs as low strength and displaying low entropy or short estimated cracking resistance.

5.5 Medium Password Testing

The medium password test used Parthik123. This password is more complex than 123456 and password because it contains letters and numbers. It may also contain uppercase and lowercase characters depending on its exact form. However, it remains partly predictable because it follows a common name-plus-number structure.

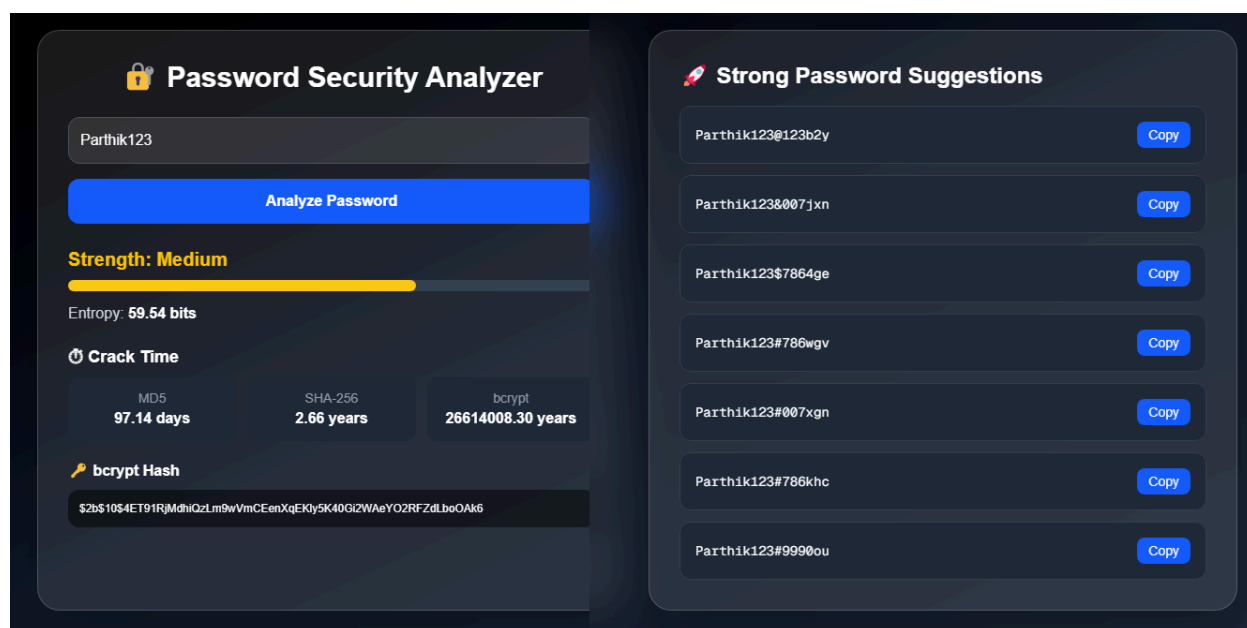


Figure 5.3: Medium Password Test Result for Parthik123

Source: Author's controlled testing of the deployed Password Security Analyzer.

The medium password test is important because many real users create passwords in this style. They often combine a name, word or familiar phrase with a short number sequence. Such passwords may satisfy basic composition rules, especially if they contain uppercase letters and numbers, but they remain vulnerable to hybrid attacks. Weir et al. show that attackers can exploit common password structures using pattern-based guessing methods [13]. Wheeler also argues that realistic password-strength estimation should consider common patterns rather than only checking whether a password contains numbers or symbols [14].

The result for Parthik123 should therefore be interpreted critically. If the analyser classifies it as medium, that classification is reasonable because it is stronger than purely numeric or dictionary-only passwords. However, it should not be treated as a highly secure password because the structure remains predictable. This demonstrates a key limitation of simple entropy-based models: they may reward character variety without fully recognising human naming patterns. Therefore, the medium password result supports the need for future enhancement through dictionary checks, pattern recognition and breached-password screening.

5.6 Strong Password Testing

The strong password test used P@rth!k#2026\$. This password includes mixed case, numbers and symbols. It is longer and more diverse than the weak and medium examples, so it should generate a higher strength classification and greater entropy estimate. It also provides a useful demonstration of how length and character diversity increase the theoretical search space.

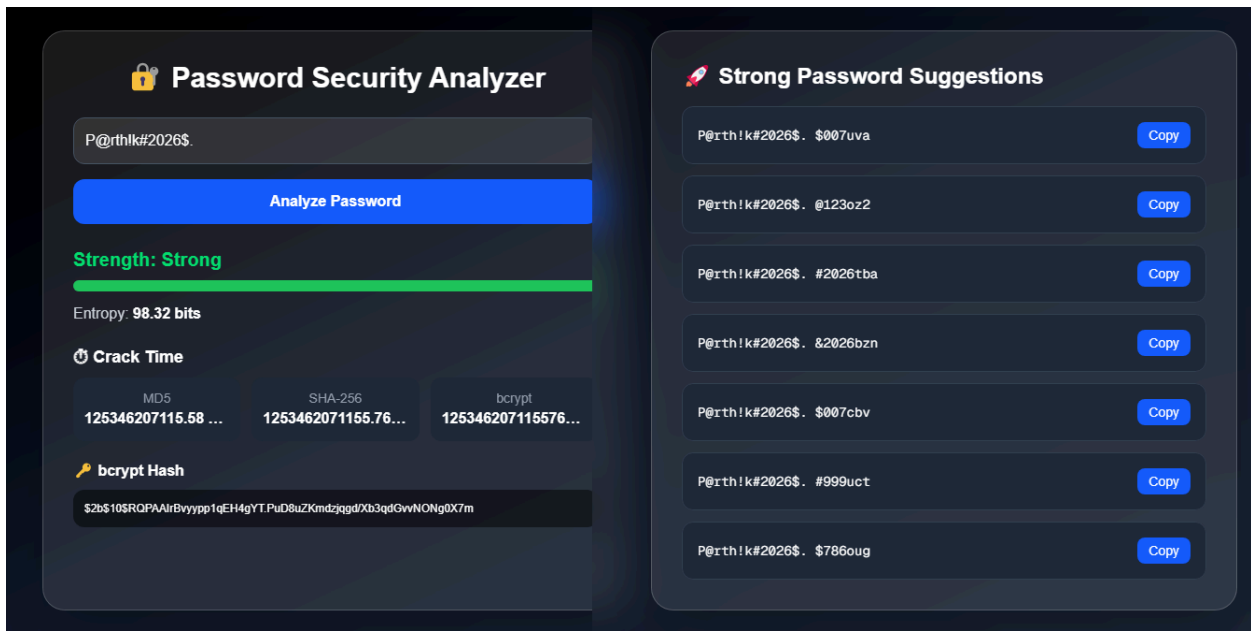


Figure 5.4: Strong Password Test Result for P@rth!k#2026\$

Source: Author’s controlled testing of the deployed Password Security Analyzer.

The strong password result demonstrates that the system can distinguish between low-complexity and higher-complexity passwords. The presence of uppercase letters, lowercase letters, numbers and symbols increases the character pool used in entropy calculation. Longer length also increases estimated search space. This supports the basic principle that password strength generally improves when length and randomness increase.

However, the result still requires critical interpretation. Although P@rth!k#2026\$ appears strong mathematically, it may still contain predictable elements if based on a name and year. In real-world attacks, personal information, names and years can reduce effective security. This means that even strong-looking passwords can be weaker than their entropy score suggests.

Therefore, the result should not be interpreted as proof that all symbol-rich passwords are safe. Instead, it shows that the system responds appropriately to increased complexity while also revealing the limitation of entropy-based strength estimation.

5.7 Password Suggestion Evaluation

The Password Security Analyzer includes a strong password suggestion feature. This feature is important because a useful password-security tool should not only diagnose weakness; it should guide improvement. In the implemented system, suggestions are generated after a user enters a password and runs the analysis. The application then displays several stronger alternatives in the right-hand panel.

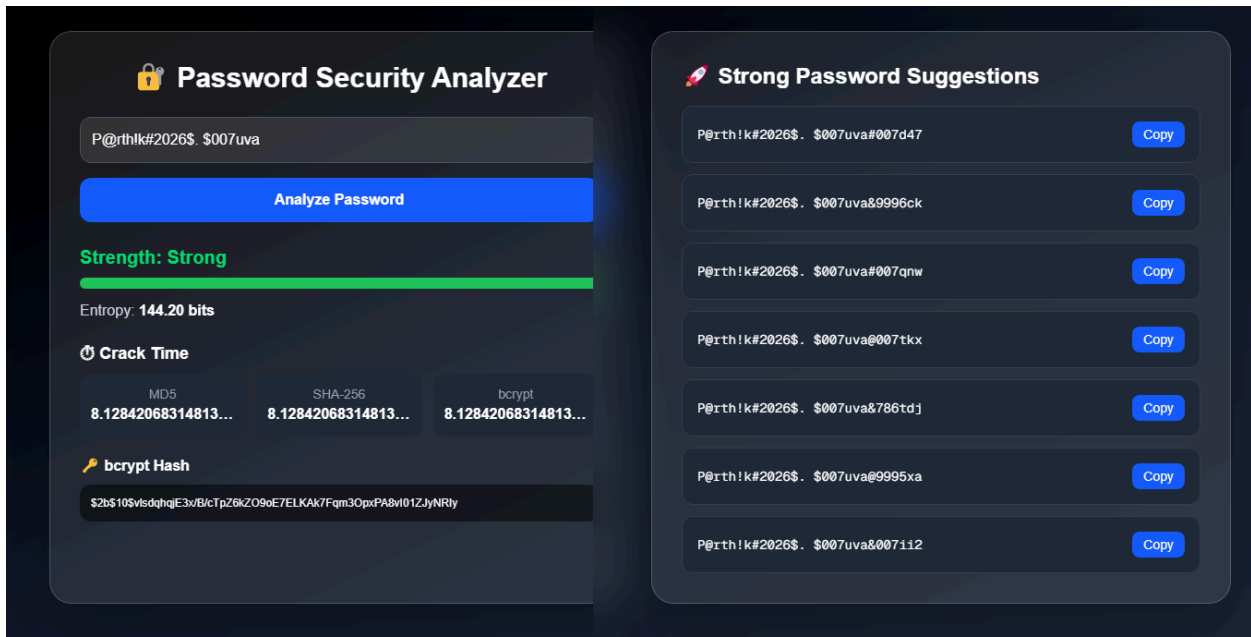


Figure 5.5: Strong Password Suggestion Test Result

Source: Author’s controlled testing of the deployed Password Security Analyzer.

The suggestion feature improves the educational value of the tool because it gives users a practical path from weak password selection to stronger password construction. For example, when a short numeric password such as 12234 is entered, the application generates alternatives that include symbols, number sequences and random suffixes. This demonstrates how added character diversity and length can increase password strength.

However, the suggestion feature has a limitation. Because suggestions may be partly based on the original password, they can preserve predictable elements. If the original password is a name, common word or short number sequence, the generated suggestions may still include that weak base. A more secure approach would generate fully random passwords or passphrases independent of the original password. Future development could also evaluate each generated suggestion before displaying it, using a pattern-aware estimator such as zxcvbn [14], [18].

Table 5.3: Evaluation of Password Suggestion Feature

Evaluation Area	Observation	Critical Interpretation
Length improvement	Suggestions are generally longer than the original input.	Increased length improves theoretical search space.
Character diversity	Suggestions include symbols, numbers and random suffixes.	Diversity improves complexity but does not guarantee unpredictability.
User guidance	Suggestions give users immediate alternatives.	Improves educational value and actionability.
Predictability risk	Suggestions may retain the original weak password base.	Future versions should generate fully random alternatives.
Usability	Copy buttons make suggestions easy to use.	Useful interface feature, but users should still avoid reusing predictable bases.

Overall, the suggestion feature supports the dissertation’s practical contribution. It makes the analyser more than a diagnostic tool by adding improvement guidance. However, the feature should be improved before being considered production-quality password generation.

5.8 Hashing Algorithm Evaluation

The application compares MD5, SHA-256 and bcrypt through estimated cracking resistance. The purpose is to demonstrate why fast general-purpose hashes are unsuitable for password storage

when used alone. MD5 and SHA-256 are included as contrast cases, while bcrypt is implemented as the password-hashing demonstration.

Table 5.4: Hashing Algorithm Evaluation

Algorithm	System Role	Security Evaluation
MD5	Used for crack-time comparison.	Very fast and unsuitable for password storage.
SHA-256	Used for crack-time comparison.	Strong for integrity purposes but unsuitable alone for password storage because it is fast.
bcrypt	Implemented through API route.	More appropriate for password storage because it is salted and cost-based.

The evaluation confirms a key finding from the literature review. MD5 should not be used for password storage because it is outdated and extremely fast [6], [16]. SHA-256 is stronger than MD5 for integrity purposes, but it is still a fast general-purpose hash and does not provide password-specific resistance to offline guessing when used alone [5], [7]. bcrypt is more appropriate because it is adaptive and deliberately slower [8].

The bcrypt output generated by the application demonstrates that the system implements secure password-hashing awareness in practice. However, it is important not to overstate this result. bcrypt improves resistance to offline guessing, but it cannot fully compensate for weak passwords. If a user chooses a highly common password, attackers may still test it early. Therefore, secure password practice requires both strong password selection and appropriate password hashing.

5.9 Evaluation Against Research Questions

The dissertation was guided by five research questions. The results from controlled testing provide evidence for answering each question.

Table 5.5: Evaluation Against Research Questions

Research Question	Evaluation Finding
How do password length, character diversity, entropy and predictability influence estimated resistance to password-cracking attacks?	The tests show that longer and more diverse passwords produce stronger outputs, but predictability can still reduce practical security.
Why are fast general-purpose hashing algorithms such as MD5 and SHA-256 unsuitable for password storage when used without salting and key stretching?	The crack-time comparison demonstrates that fast hashes allow rapid guessing and are therefore unsuitable for password storage when used alone.
In what ways does bcrypt improve password-storage security through salting, adaptive cost and computational expense?	bcrypt increases the cost of each password guess and is therefore more suitable for password storage than fast hashes.
How effectively does the implemented Password Security Analyzer communicate password-strength risks and improvement guidance to users?	The system communicates risk through strength classification, entropy, crack-time estimates, colour-coded output and suggestions.
What secure password-creation and password-storage recommendations can be derived from the project's design, implementation and evaluation?	Users should choose long, unique and less predictable passwords, while developers should use bcrypt, Argon2id or PBKDF2 with salts and cost factors.

The evaluation shows that the application answers the research questions at an artefact level. It demonstrates password-security principles effectively, but it does not provide population-level behavioural evidence because no large-scale user study was conducted.

5.10 Discussion of Findings

The results demonstrate that the Password Security Analyzer functions as an educational password-security artefact. It successfully shows that short and predictable passwords are

weak, that entropy increases with length and character diversity, and that bcrypt provides greater resistance than fast hashing assumptions. The system also provides stronger password suggestions, which supports practical user improvement.

The most important finding is that password security cannot be reduced to a single factor. Weak passwords remain risky even if hashed using a stronger algorithm. Similarly, strong passwords can be undermined if stored with fast unsalted hashes. This confirms the literature's emphasis on both user-side behaviour and system-side protection. NIST and OWASP guidance support the use of salted, cost-based password-hashing mechanisms, while usable-security literature shows that users often choose predictable passwords under practical constraints [3]–[5], [11], [12].

The evaluation also shows the value and limitation of entropy. Entropy helps quantify strength and communicate risk, but it does not fully capture human predictability. A password containing a name, date or common word may receive a relatively high entropy estimate because it contains multiple character classes, but real attackers may guess it through hybrid strategies. This finding supports the argument that future versions of the analyser should include pattern-aware strength estimation, dictionary checks and breached-password screening.

The hashing comparison provides another critical finding. MD5 and SHA-256 should not be treated equally with bcrypt in password-storage contexts. MD5 is obsolete for modern security-sensitive storage. SHA-256 remains useful for integrity purposes, but it is not sufficient for password storage when used alone because it is fast. bcrypt is more appropriate because it is salted and cost-based. However, bcrypt is not a complete solution. It must be combined with strong passwords, secure implementation, appropriate cost factors and ideally MFA.

The suggestion feature adds practical value because it helps users move from weak passwords toward stronger alternatives. However, the generated suggestions should be improved. Since some suggestions are based on the original input, they may retain predictable elements. A better approach would be to generate random passphrases or password-manager-style random strings. This would reduce reliance on user-provided weak bases.

Overall, the results show that the Password Security Analyzer is successful as a dissertation artefact. It demonstrates key password-security concepts, supports controlled testing and

provides visual evidence of implementation. However, it remains an educational tool rather than a production authentication system.

5.11 Limitations of the Evaluation

The evaluation has several limitations. First, the test dataset is small and controlled. This is appropriate for artefact evaluation but limits generalisability. A larger study using more password samples would provide stronger evidence about system accuracy across different password types.

Second, the crack-time estimates are based on simplified assumptions. Real-world cracking speeds vary significantly depending on hardware, GPU acceleration, attacker tools, password lists, salting and hashing configuration. The estimates are therefore useful for education but not exact measurements.

Third, the entropy model is limited because it relies primarily on length and character-set size. It does not fully account for dictionary words, personal names, dates, keyboard patterns or leaked-password frequency. This can overestimate the strength of some human-created passwords.

Fourth, the evaluation did not include a formal user study. Therefore, the dissertation can claim that the interface presents password-security feedback, but it cannot strongly claim that users will change their behaviour after using the tool. Future work should include participant testing, usability questionnaires and before-and-after password-awareness measures.

Fifth, the hashing comparison is not a full cryptographic benchmark. MD5 and SHA-256 are used as estimated comparison points, while bcrypt is implemented. The system does not currently implement Argon2id, PBKDF2 or scrypt. This limits the breadth of password-hashing evaluation.

5.12 Recommendations Based on Evaluation

The evaluation supports several recommendations. Users should create passwords that are long, unique and difficult to predict. They should avoid short numeric passwords, dictionary

words, names, dates and predictable substitutions. Password managers should be encouraged because they reduce reliance on memorised, reused or patterned passwords.

Developers should avoid MD5 and unsalted SHA-256 for password storage. Passwords should be stored using modern password-hashing mechanisms such as bcrypt, Argon2id or PBKDF2 with appropriate salts and cost factors [4], [5], [15]. Organisations should also use MFA to reduce the impact of password compromise, although MFA should not be treated as a replacement for secure password storage [10].

For future development of the Password Security Analyzer, the system should include pattern-aware strength estimation, breached-password checking, Argon2id support, PBKDF2 comparison, a clear warning not to enter real passwords, and stronger password generation independent of the original input. These improvements would strengthen both the security and educational value of the artefact.

5.13 Summary

This chapter evaluated the Password Security Analyzer through controlled testing. The evaluation confirmed that the system accepts password input, analyses strength, displays entropy, estimates crack time, generates bcrypt hashes and provides stronger password suggestions. Weak password tests showed that short numeric and dictionary-based passwords are correctly treated as low strength. The medium password test showed that name-plus-number structures are stronger than simple weak passwords but remain predictable. The strong password test demonstrated that length and character diversity increase estimated strength, although human predictability can still reduce practical security.

The hashing evaluation confirmed that MD5 and SHA-256 are unsuitable for password storage when used alone because they are fast, while bcrypt is more appropriate because it is adaptive and cost-based. The suggestion feature adds practical value, but future versions should generate fully random alternatives rather than modifying the original password base.

Overall, the results show that the Password Security Analyzer meets its purpose as an educational cybersecurity artefact. It demonstrates the relationship between password

characteristics, entropy, cracking resistance and secure hashing awareness. The next chapter concludes the dissertation by summarising the findings, identifying final limitations and presenting recommendations for secure password practice and future system development.

Chapter 6: Conclusion and Recommendations

6.1 Introduction

This chapter concludes the dissertation by summarising the study, reviewing the achievement of the research aim and objectives, presenting the main findings, discussing the contribution of the implemented artefact, identifying limitations and recommending future improvements. The dissertation investigated password-security weakness through the design, implementation and evaluation of a deployed web-based Password Security Analyzer. The project focused on password strength, entropy, estimated cracking resistance, hashing awareness and the comparison of MD5, SHA-256 and bcrypt in password-storage contexts.

The central argument developed throughout the dissertation is that password security cannot be reduced to one technical measure. A strong password can still be weakened if it is stored using an unsuitable hashing method, while secure password hashing cannot fully protect highly predictable passwords. Effective password security therefore requires both user-side and system-side controls. Users need to create long, unique and less predictable passwords, while developers must avoid fast unsalted hashes and use password-specific hashing mechanisms such as bcrypt, Argon2id or PBKDF2. This conclusion is consistent with NIST guidance on salted, cost-based password-hashing schemes and OWASP recommendations on secure password storage [4], [5].

6.2 Summary of the Study

The dissertation began by identifying password-based authentication as a continuing cybersecurity challenge. Although alternative authentication methods such as biometrics,

passkeys and multi-factor authentication are increasingly used, passwords remain embedded in websites, enterprise systems, email services, cloud platforms and financial applications. Their persistence is largely explained by familiarity, low cost and deployability, but these advantages also create risk because password security depends heavily on user behaviour and backend storage practice.

The literature review showed that users often create passwords that are short, reused or predictable. It also showed that visible complexity does not always equal real security. A password containing numbers and symbols may still be weak if it follows common human patterns such as dictionary words, names, dates or simple substitutions. The review further established that MD5 is unsuitable for password storage, SHA-256 is not ideal when used alone because it is a fast general-purpose hash, and bcrypt is more appropriate because it introduces salting and adaptive computational cost [5]–[8].

The methodology chapter adopted a Design Science Research approach because the dissertation developed and evaluated a practical cybersecurity artefact. This was appropriate because the study did not only analyse password-security theory; it produced a deployed web application that demonstrates password strength, entropy, estimated cracking resistance and bcrypt hashing. The implementation chapter then explained the system architecture, functional requirements, non-functional requirements, user interface, entropy logic, crack-time estimation, bcrypt implementation and password suggestion feature. Finally, the evaluation chapter tested the system using controlled password examples and assessed how effectively the application communicated password-security risk.

6.3 Achievement of Research Aim and Objectives

The aim of the study was to design, implement and critically evaluate a deployed web-based Password Security Analyzer that demonstrates how password complexity, entropy and hashing mechanisms influence estimated resistance to password-cracking attacks, with the purpose of improving password-security awareness and informing secure password-storage practice. This aim was achieved because the application was implemented, deployed and evaluated through controlled testing.

Table 6.1: Achievement of Research Objectives

Research Objective	Achievement
To critically examine how password length, complexity, entropy and predictability influence password strength.	Achieved through the literature review, entropy discussion and controlled password tests.
To evaluate the security implications of MD5, SHA-256 and bcrypt in relation to password storage crack-time estimation and offline cracking resistance.	Achieved through algorithm comparison, bcrypt implementation.
To design and implement a deployed web-based Password Security Analyzer using modern web technologies.	Achieved through the development of a Next.js, React, TypeScript, Tailwind CSS, bcryptjs and Vercel-based application.
To test the analyser using weak, medium and strong password examples and assess how effectively it communicates password-security risk.	Achieved through controlled testing of weak, medium and strong password examples in Chapter 5.
To develop evidence-based recommendations for secure password creation, password storage and future improvement of the implemented system.	Achieved through the recommendations presented in this chapter.

The objectives were therefore met within the dissertation's defined scope. However, the findings should be interpreted as an artefact-based evaluation rather than a large-scale empirical study of user behaviour.

6.4 Main Findings

The first major finding is that short and predictable passwords remain highly vulnerable. The weak password tests showed that numeric and dictionary-based passwords produce low strength classifications and low entropy. This confirms the argument that password weakness is not only a technical issue but also a behavioural one. Users often choose memorable passwords, but memorability can reduce resistance to guessing.

The second finding is that length and character diversity improve theoretical strength, but they do not fully guarantee practical security. A password with uppercase letters, lowercase letters, numbers and symbols may achieve a higher entropy estimate, but if it contains names, years or common words, it may still be vulnerable to hybrid dictionary attacks. This supports the critique developed in Chapter 2 that password-strength tools should not rely only on visible complexity or character-class rules [13], [14].

The third finding is that MD5 and SHA-256 should not be used as standalone password-storage mechanisms. MD5 is obsolete for modern security-sensitive storage because of known weaknesses and high speed [6], [16]. SHA-256 is stronger than MD5 for integrity purposes, but it remains a fast general-purpose hash and is therefore unsuitable for password storage when used without salting and key stretching [5], [7]. The system's crack-time comparison helped demonstrate why speed is a disadvantage in password-storage contexts.

The fourth finding is that bcrypt is more appropriate for password storage because it is adaptive, salted and deliberately slower. The application demonstrated bcrypt hashing through the /api/hash route, using bcryptjs and a cost factor. This supports the dissertation's system-side security argument. However, bcrypt does not make weak passwords safe. It increases the cost of guessing but cannot fully compensate for passwords such as 123456, password or predictable name-plus-number patterns.

The fifth finding is that the Password Security Analyzer is useful as an educational cybersecurity artefact. It communicates risk through strength classification, entropy, estimated cracking resistance, bcrypt output and password suggestions. This makes it more effective than a purely theoretical explanation because users can see how password choices affect security indicators. However, the tool remains educational and should not be treated as a complete enterprise password-auditing or authentication system.

6.5 Contribution of the Study

The study contributes in three main ways. First, it provides a practical cybersecurity artefact in the form of a deployed Password Security Analyzer. This strengthens the dissertation because

the research is not limited to theoretical discussion. The artefact demonstrates password analysis and hashing awareness through a working web application.

Second, the study links user-side password behaviour with system-side hashing practice. Many discussions of password security focus either on user password weakness or cryptographic storage mechanisms. This dissertation combines both perspectives by showing that password strength and hashing design must work together.

Third, the study translates technical password-security concepts into user-facing feedback. Concepts such as entropy, crack-time estimation, salting, bcrypt and adaptive hashing can be difficult for non-specialist users to understand. The application makes these ideas more visible by presenting them in an interactive interface. This gives the study practical value for cybersecurity education, developer awareness and password-security communication.

6.6 Recommendations

The findings support several recommendations for users, developers, organisations and future system development.

Table 6.2: Recommendations Based on the Dissertation Findings

The most important recommendation is that password security should be treated as layered. A strong password policy alone is insufficient if backend storage is poor. Similarly, strong hashing alone cannot fully protect predictable passwords. Secure practice should combine strong password creation, password managers, salted adaptive hashing, breached-password screening, MFA and clear user education.

Area	Recommendation	Justification
Password creation	Users should create long, unique and less predictable passwords.	Length and unpredictability increase resistance to guessing attacks.
Password reuse	Users should avoid reusing passwords across services.	Reuse increases risk from credential stuffing and breach exposure.
Password management	Password managers should be encouraged.	They help users generate and store stronger unique passwords.
Password storage	Developers should avoid MD5 and unsalted SHA-256 for password storage.	Fast hashes enable high-speed offline guessing.
Secure hashing	Developers should use bcrypt, Argon2id or PBKDF2 with salts and cost factors.	These mechanisms are designed to increase attacker cost [4], [5], [15].
Authentication security	Organisations should use MFA alongside secure password storage.	MFA reduces reliance on passwords alone but does not replace secure hashing [10].
User education	Password tools should explain why a password is weak, not only reject it.	Clear feedback improves security awareness and user decision-making.
Future system development	The analyser should include Argon2id, breached-password checks and stronger random password generation.	These improvements would increase technical completeness and practical value.

6.7 Limitations of the Study

The study has several limitations. First, the evaluation used a controlled and relatively small password dataset. This was appropriate for artefact evaluation, but it limits the generalisability of the results. A larger dataset containing more password types would provide stronger evidence about how consistently the system classifies different patterns.

Second, crack-time estimates were based on simplified assumptions. Real-world cracking performance depends on hardware, GPU acceleration, password lists, salting, algorithm implementation and cost factor. Therefore, the crack-time values should be interpreted as comparative educational indicators rather than exact predictions.

Third, entropy estimation does not fully capture human predictability. A password may appear strong because it contains different character classes, but it may still contain predictable names, words, dates or substitutions. This limits the accuracy of entropy-only strength estimation.

Fourth, the implemented system includes bcrypt but does not currently implement Argon2id, PBKDF2 or scrypt. This limits the breadth of password-hashing comparison. Since OWASP recommends Argon2id as a preferred modern password-hashing option where available, future development should include Argon2id support [5], [15].

Fifth, the study did not include a formal user study. Therefore, the dissertation can argue that the interface communicates password-security information, but it cannot conclusively prove that users would change their password behaviour after using the tool. A future study should involve participants, usability surveys and pre-test/post-test password-awareness measures.

Finally, the application does not implement a complete authentication system. It does not include user registration, login verification, database-backed password storage, MFA, account recovery or session management. This limitation is acceptable because the dissertation's focus is password analysis and hashing awareness, not full identity-platform development.

6.8 Future Work

Future development should improve both the technical depth and educational value of the Password Security Analyzer. The first improvement should be the inclusion of Argon2id, because OWASP recommends it as a preferred password-hashing option where available [5]. Adding

PBKDF2 and scrypt would also allow broader comparison between password-hashing mechanisms.

The second improvement should be a more advanced password-strength model. The current entropy-based method is useful for education, but it can overestimate passwords that follow predictable patterns. Integrating zxcvbn-style pattern recognition would allow the system to detect dictionary words, names, keyboard patterns, dates, repeated sequences and leetspeak substitutions [14], [18].

The third improvement should be privacy-preserving breached-password checking. A k-anonymity approach similar to the Have I Been Pwned password-checking model could allow users to check whether a password appears in known breaches without sending the full password to a server. This would strengthen the tool's practical security value.

The fourth improvement should be stronger password generation. The current suggestion feature modifies the original input, which can preserve weak patterns. A future version should generate fully random passwords or memorable passphrases independent of the original password. Each generated suggestion should also be tested before being displayed.

The fifth improvement should be formal usability testing. The dissertation currently evaluates the artefact through controlled testing, but future work should involve real participants. This would allow measurement of whether users understand the feedback, whether the interface changes password-selection behaviour, and whether the tool improves password-security awareness.

6.9 Final Conclusion

This dissertation investigated password security through the design, implementation and evaluation of a deployed web-based Password Security Analyzer. The study showed that password insecurity arises from the interaction between weak user behaviour and poor system-side storage practice. Users often choose passwords that are short, reused or predictable, while developers may mistakenly use fast general-purpose hashes for password storage.

The project demonstrated that password strength is influenced by length, character diversity, entropy and predictability. Weak passwords such as short numeric strings or dictionary words remain highly vulnerable. Medium passwords that combine names and numbers may appear stronger but can still be predictable. Stronger passwords benefit from greater length and diversity, but even these must be interpreted carefully if they contain personal or patterned elements.

The dissertation also demonstrated that password hashing must be context-specific. MD5 is unsuitable for password storage, and SHA-256 is not sufficient when used alone because it is fast. bcrypt is more appropriate because it is salted, adaptive and computationally expensive. However, bcrypt does not remove the need for strong passwords. The safest approach combines strong unique passwords, password managers, secure password hashing, appropriate cost factors, breached-password screening and MFA.

Overall, the Password Security Analyzer met its purpose as an educational cybersecurity artefact. It provided password-strength analysis, entropy feedback, crack-time estimation, bcrypt hashing demonstration and password suggestions. Although it is not a production authentication platform, it successfully demonstrates the relationship between password characteristics and secure password-storage awareness. The dissertation therefore contributes both a working artefact and a critical evaluation of password-security practice.

References

- [1] Microsoft, "Microsoft Digital Defense Report 2024," *Microsoft Security Insider*, 2024. [Online]. Available: <https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024>. Accessed: May 28, 2026.
- [2] Verizon, "2026 Data Breach Investigations Report," *Verizon Business*, 2026. [Online]. Available: <https://www.verizon.com/about/news/breach-industry-wide-dbir-finds>. Accessed: May 28, 2026.
- [3] National Institute of Standards and Technology, "Digital Identity Guidelines: Authentication and Lifecycle Management," *NIST Special Publication 800-63B*, Revision 3, 2017. [Online]. Available: <https://pages.nist.gov/800-63-3/sp800-63b.html>. Accessed: May 28, 2026.
- [4] National Institute of Standards and Technology, "Digital Identity Guidelines: Authentication and Lifecycle Management," *NIST Special Publication 800-63B*, Revision 4, 2025. [Online]. Available: <https://pages.nist.gov/800-63-4/sp800-63b.html>. Accessed: May 28, 2026.
- [5] OWASP Foundation, "Password Storage Cheat Sheet," *OWASP Cheat Sheet Series*, 2024. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. Accessed: May 28, 2026.
- [6] R. Rivest, "The MD5 Message-Digest Algorithm," *RFC 1321*, Internet Engineering Task Force, 1992. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc1321>. Accessed: May 28, 2026.
- [7] National Institute of Standards and Technology, "Secure Hash Standard," *FIPS PUB 180-4*, 2015. [Online]. Available: <https://csrc.nist.gov/pubs/fips/180-4/upd1/final>. Accessed: May 28, 2026.

- [8] N. Provos and D. Mazières, “A Future-Adaptable Password Scheme,” in *Proceedings of the USENIX Annual Technical Conference*, Monterey, CA, USA, 1999. [Online]. Available: <https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf>. Accessed: May 28, 2026.
- [9] P. James, “Password Security Analyzer,” deployed web application, 2026. [Online]. Available: <https://password-analyzer-red.vercel.app/>. Accessed: May 28, 2026.
- [10] Cybersecurity and Infrastructure Security Agency, “More Than a Password,” CISA, 2026. [Online]. Available: <https://www.cisa.gov/MFA>. Accessed: May 28, 2026.
- [11] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes,” in *Proceedings of the IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2012, pp. 553–567.
- [12] D. Florêncio, C. Herley, and P. C. van Oorschot, “Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts,” in *Proceedings of the 23rd USENIX Security Symposium*, San Diego, CA, USA, 2014, pp. 575–590.
- [13] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, “Password Cracking Using Probabilistic Context-Free Grammars,” in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2009, pp. 391–405.
- [14] D. L. Wheeler, “zxcvbn: Low-Budget Password Strength Estimation,” in *Proceedings of the 25th USENIX Security Symposium*, Austin, TX, USA, 2016, pp. 157–173.
- [15] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, “Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications,” *RFC 9106*, Internet Research Task Force, 2021. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc9106.html>. Accessed: May 28, 2026.
- [16] X. Wang and H. Yu, “How to Break MD5 and Other Hash Functions,” in *Advances in Cryptology — EUROCRYPT 2005*, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 19–35.
- [17] C. Percival, “Stronger Key Derivation via Sequential Memory-Hard Functions,” 2009. [Online]. Available: <https://www.tarsnap.com/scrypt/scrypt.pdf>. Accessed: May 28, 2026.

- [18] Dropbox, “zxcvbn: Realistic Password Strength Estimation,” *Dropbox Tech Blog*, 2012. [Online]. Available: <https://dropbox.tech/security/zxcvbn-realistic-password-strength-estimation>. Accessed: May 28, 2026.
- [19] P. C. van Oorschot, *Computer Security and the Internet: Tools and Jewels*. Cham, Switzerland: Springer, 2021.
- [20] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed. Harlow, England: Pearson, 2020.
- [21] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2020.
- [22] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, 2023. [Online]. Available: <https://toc.cryptobook.us/>. Accessed: May 28, 2026.
- [23] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [24] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [25] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. Thousand Oaks, CA, USA: SAGE, 2018.
- [26] Vercel, “Next.js Documentation,” *Next.js Docs*, 2026. [Online]. Available: <https://nextjs.org/docs/app>. Accessed: May 28, 2026.
- [27] Meta Open Source, “React Documentation,” *React*, 2026. [Online]. Available: <https://react.dev/learn>. Accessed: May 28, 2026.
- [28] Microsoft, “TypeScript Documentation,” *TypeScript*, 2026. [Online]. Available: <https://www.typescriptlang.org/docs/>. Accessed: May 28, 2026.

[29] npm, "bcryptjs," *npm Package Documentation*, 2025. [Online]. Available: <https://www.npmjs.com/package/bcryptjs>. Accessed: May 28, 2026.

[30] Vercel, "Vercel Documentation," *Vercel Docs*, 2026. [Online]. Available: <https://vercel.com/docs>. Accessed: May 28, 2026.