Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«МЕТОДИ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ ФАКТОРУ ЗНАННЯ»

Виконала: студентка 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____ Вікторія КЛИШ

Керівник: к. т. н., доцент каф. ЗІ
_____ Юрій БАРИШЕВ
«16» грудня 2025 р.

Рецензент: к. т. н., доцент каф. ПЗ
_____ Оксана РОМАНЮК
«16» грудня 2025 р.

Допущено до захисту
В. о. завідувача кафедри ЗІ
д. т. н., професор
_____ Володимир ЛУЖЕЦЬКИЙ
«16» грудня 2025 р.

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України
Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека та захист інформації
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ЗІ,
д. т. н., проф.
_____Володимир ЛУЖЕЦЬКИЙ
«24» 09 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Вікторії КЛИШ

1. Тема роботи: «Методи автентифікації користувачів на основі фактору знання» керівник роботи: Юрій БАРИШЕВ, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 24 вересня 2025 року № 313.
2. Строк подання студентом роботи 16 грудня 2025 р.
3. Вихідні дані до роботи:
   - платформа виконання: Node.js;
   - мова програмування: JavaScript;
   - мінімальна довжина вихідних паролів – 12 символів;
   - мінімальна довжина мнемонічних фраз – 5 слів;
   - вид автентифікації користувачів – на основі фактору знання.
4. Зміст текстової частини: Вступ. 1 Аналіз методів автентифікації користувачів. 2 Метод автентифікації користувачів на основі паролів. 3 Метод автентифікації користувачів на основі мнемонічних фраз. 4 Експериментальні дóслідження та тестува́ння. 5 Економічна час́тина. Висновки. Перéлік використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: аналіз методів автентифікації , аналіз засобів автентифікації , математичний опис процесу автентифікації , метод автентифікації на основі паролів , узагальнений алгоритм автентифікації на основі паролів , таблиця заміни символів , алгоритм формування паролів , метод автентифікації на основі мнемонічних фраз , модель мнемонічних фраз , узагальнений алгоритм автентифікації на основі мнемонічних фраз , алгоритм генерування мнемонічних фраз , теоретичні оцінки стійкості методів , результати модульного тестування , результати експериментальних досліджень .

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняла
1	Ю. БАРИШЕВ, к.т.н., доц., доц. каф.ЗІ	25.09.25	06.10.25
2	Ю. БАРИШЕВ, к.т.н., доц., доц. каф.ЗІ	25.09.25	03.11.25
3	Ю. БАРИШЕВ, к.т.н., доц., доц. каф.ЗІ	25.09.25	17.11.25
4	Ю. БАРИШЕВ, к.т.н., доц., доц. каф.ЗІ	25.09.25	07.12.25
5	О. ЛЕСЬКО, к.е.н., доц., зав. каф. ЕПВМ	25.09.25	18.12.25

7. Дата видачі завдання 24 вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.10.2025	
3	Науково-технічне обґрунтування	11.10.2025 – 22.10.2025	
4	Аналіз методів автентифікації користувачів	23.10.2025 – 26.10.2025	
5	Аналіз та формування вимог до методів автентифікації та структурних моделей	27.10.2025 – 02.11.2025	
6	Розробка методів автентифікації користувачів на основі зрозумілих паролів та мнемонічних фраз	03.11.2025 – 10.11.2025	
7	Програмна реалізація, тестування та експериментальні дослідження методів	10.11.2025 – 17.11.2025	
8	Розробка розділу економічного обґрунтування доцільності розробки	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка магістерської роботи на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студентка _____ Вікторія КЛИ

Керівник роботи _____ Юрій БАРИШ

АНОТАЦІЯ

УДК 004.056

Клиш В. Методи автентифікації користувачів на основі фактору знання. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. с. 97 Укр. мовою. Бібліогр.: 94 назв; рис. 18; табл. 14.

Магістерська кваліфікаційна робота присвячена розробці методів автентифікації користувачів на основі фактору знання із застосуванням зрозумілих паролів та мнемонічних фраз. Здійснено аналіз сучасних методів автентифікації та засобів реалізації на основі паролів, визначено їхні переваги та недоліки. Розроблено математичні описи завдань автентифікації користувачів на основі факторів знання. Запропоновано методи автентифікації на основі паролів та на основі мнемонічних фраз. Розроблено алгоритми, що реалізують запропоновані методи. Визначено теоретичні оцінки стійкості методів. Здійснено програмну реалізацію методів у вигляді програмної бібліотеки. Виконано тестування та експериментальне дослідження методів, що дозволило підтвердити коректність ухвалених технічних рішень. Визначено показники економічної ефективності розробки.

Ілюстративна частина містить 14 плакатів із демонстрацією результатів моделювання та експериментальних досліджень.

Ключові слова: кібербезпека, автентифікація, фактор знання, пароль, мнемонічна фраза, автентифікація користувачів, теоретичні оцінки стійкості, експериментальні дослідження.

ABSTRACT

Klysh V. Methods of user authentication based on the knowledge factor. Master's thesis in the field of 125 – Cybersecurity and Information Protection, educational programme – Security of Information and Communication Systems. Vinnytsia: VNTU, 2025. p. 93 In Ukrainian. Bibliography: 97 titles; fig. 18; table 14.

The master's thesis is devoted to the development of user authentication methods based on the knowledge factor using understandable passwords and mnemonic phrases. An analysis of modern authentication methods and password-based implementation tools has been carried out, and their advantages and disadvantages have been identified. Mathematical descriptions of user authentication tasks based on knowledge factors have been developed. Password-based and mnemonic phrase-based authentication methods have been proposed. Algorithms implementing the proposed methods have been developed. Theoretical estimates of the stability of the methods have been determined. The methods have been implemented in the form of a software library. Testing and experimental research of the methods have been carried out, confirming the correctness of the technical solutions adopted. The economic efficiency indicators of the development have been determined.

The illustrative part contains 14 posters demonstrating the results of modelling and experimental research.

Keywords: cybersecurity, authentication, knowledge factor, password, mnemonic phrase, user authentication, theoretical stability estimates, experimental studies.

ЗМІСТ

ВСТУП

Автентифікація за фактором знання базується на тому, що користувач має знати певну інформацію, наприклад пароль або PIN-код. Такий підхід простий у реалізації та використовується практично в усіх інформаційних системах, однак його стійкість значною мірою залежить від поведінки користувачів і складності обраних комбінацій.

Попри наявність широкого спектра методів автентифікації – переважна більшість систем і досі використовує автентифікацію на основі фактору знання, зокрема на основі паролів. Їх популярність пояснюється простотою впровадження, відсутністю потреби у додатковому обладнанні та зрозумілістю для користувачів. Водночас саме парольна автентифікація залишається одним із найвразливіших елементів безпеки через низьку складність паролів, повторне їх використання та людський фактор.

Проблема полягає в тому, що користувачам складно запам’ятовувати випадкові або занадто складні комбінації символів, через що вони схильні створювати короткі, передбачувані або повторювані паролі. Це значно полегшує зловмисникам реалізацію атак повного перебору (brute force), словникових атак, а також атак на основі соціальної інженерії. З іншого боку, надмірне ускладнення механізмів автентифікації призводить до втрати зручності користування системою.

Тому актуальним завданням є розроблення методів автентифікації, які забезпечують баланс між безпекою та зручністю. Одним із перспективних напрямів є розробка методів формування факторів знання, що поєднують асоціативне мислення користувача зі стійкістю. Такі підходи дають змогу знизити когнітивне навантаження, не жертвуючи рівнем захисту.

Метою роботи є покращення методів автентифікації користувачів на основі факторів знання, які забезпечують стійкість до атак при збереженні зручності запам’ятовування.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз сучасних методів автентифікації;

- дослідити особливості методів, що базуються на факторі знання, та визначити їхні недоліки;
- виконати математичні описи постановки завдання автентифікації на основі фактору знання;
- розробити метод автентифікації на основі паролів;
- розробити метод автентифікації на основі мнемонічних фраз;
- провести теоретичне оцінювання стійкості запропонованих методів до атак перебору;
- реалізувати засіб генерування паролів і фраз;
- виконати експериментальне дослідження ефективності методів;
- визначити показники економічної ефективності розробки.

Об’єктом дослідження є процес автентифікації користувачів в інформаційних системах.

Предметом дослідження є методи формування факторів автентифікації на основі зрозумілих паролів та мнемонічних фраз.

Наукова новизна роботи полягає у такому:

- отримав подальший розвиток метод автентифікації користувачів на основі паролів, який на відміну від відомих передбачає під час генерування паролю його доповнення до мінімально дозволеної довжини у спосіб, який визначається випадковим чином, що дозволяє збільшити стійкість паролю до зламу;
- удосконалено метод автентифікації користувачів на основі мнемонічних фраз, який на відміну від відомих передбачає застосування моделей речення для формування мнемонічної фрази, що дозволяє наблизити згенеровану мнемонічну фразу до природної мови та покращити її запам’ятовуваність.

Практична значущість полягає розробці програмної бібліотеки для автентифікації користувачів на основі фактору знання.

За результатом роботи отримано патент на корисну модель "Спосіб автентифікації користувачів" [1].

Результати роботи апробовані на конференціях з публікацією тез та матеріалів доповідей:

- LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ), 2024 [2];
- Theoretical and applied cybersecurity «TACS-2024» [3];
- The 13th International Scientific Conference «ITSec» (2025) [4];
- LIV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (ВНТКП ВНТУ), 2025 [5];
- Міжнародна конференція SMICS-2025 «Безпека сучасних інформаційно-комунікаційних систем», 2025 [6].

1 АНАЛІЗ МЕТОДІВ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

1.1 Аналіз підходів до автентифікації користувачів

Автентифікація – це процес перевірки, чи справді користувач є тією особою, за кого себе видає, і надання відповідного доступу до інформаційної системи.

Зазвичай виділяють основні підходи, які класифікуються за 3 традиційними факторами [7]:

– знання (what you know);

– володіння (what you have);

– біометрії (what you are).

До фактору знання належать класичні паролі, графічні ключі та PIN-коди, що застосовуються у банківських картах і мобільних пристроях, одноразові коди у застосунках Google Authenticator [8], Authy [8].

Фактор володіння реалізується за допомогою апаратних токенів (RSA SecureID [9], YubiKey [9]), програмних токенів (JWT [10]) та смарт-карток.

Фактор біометрії реалізовано у таких системах, як Touch ID та Face ID [11], а також у системах контролю доступу, де використовуються сканери відбитків чи розпізнавання облич. Біометричні методи є найзручнішими, однак через високу вартість їх впровадження такі системи використовуються рідше.

Кожен підхід має свої сильні і слабкі сторони. Наприклад, фактор знання (пароль, секретна мнемонічна фраза) простий у впровадженні і зрозумілий користувачеві, але часто вразливий до атак (словникові атаки, фішинг, повторне використання паролів) [12].

Сучасна тенденція полягає у поєднанні кількох факторів автентифікації (MFA) – наприклад, пароль (фактор знання) та SMS-підтвердження (фактор володіння) [13]. Це суттєво підвищує рівень безпеки.

Додатково все частіше використовується геолокаційний фактор, який враховує місцезнаходження користувача під час входу. Система може визначати, чи відповідає спроба входу типовій поведінці користувача, за допомогою GPS або IP-адреси, і блокувати підозрілі дії.

Сучасна автентифікація розвивається у напрямку комбінування різних критеріїв і контекстної перевірки (місцезнаходження, поведінка), що забезпечує баланс між зручністю та безпекою.

Підходи до автентифікації можна оцінювати за такими важливими параметрами [14]:

– стійкість до атак (перебір, фішинг);

– зручність користування – наскільки метод просто запам’ятати/використовувати;

– вартість і впровадження;

– контекст застосування (мобільні, банківські та вебзастосунки)

Вибір підходу до автентифікації має враховувати не лише технічну стійкість, але й зручність для користувача та контекст застосування.

1.2 Аналіз методів автентифікації користувачів на основі фактору знання

Методи автентифікації, засновані на знанні користувача, передбачають, що користувач повинен надати якийсь секрет, який він знає, наприклад пароль. Такий підхід залишився одним із найпоширеніших у реалізації інформаційних систем через простоту та відносно низьку вартість впровадження [14].

Типи методів на основі знання:

– PIN-коди – короткі цифрові коди, зазвичай застосовуються у поєднанні з фізичним носієм, але само по собі також реалізують фактор знання. PIN-код складається з 4-6 цифр, що обмежує простір можливих комбінацій до 10^4 - 10^6 варіантів.

– Когнітивні паролі / секретні запитання – питання, відповіді на які знає лише користувач (наприклад: «Ім’я першого домашнього улюбленця?»), часто застосовуються як резервний метод.

– Паролі (Passwords) – найрозповсюдженіший метод, який полягає в тому, що користувач вводить рядок символів, який він знає.

– Одноразовий пароль – динамічні паролі, які дійсні лише протягом короткого проміжку часу; підвищують безпеку, але потребують синхронізації або додаткового носія.

– Мнемонічні фрази – паролі сформовані із асоційованих слів чи фраз; поєднують високу стійкість до атак та зручність запам’ятовування для користувача.

Для систематичного порівняння методів автентифікації на основі фактору знання визначено критерії оцінювання, що відображають ключові аспекти їх практичного застосування.

Практичність впровадження відображає складність технічної реалізації методу. Методи, що не потребують спеціалізованого обладнання (паролі, PIN-коди), мають високу практичність. Методи, що вимагають додаткових компонентів (OTP), мають середню практичність.

Стійкість до атак характеризує здатність методу протистояти злому. Згідно з NIST SP 800-63B-4 [15], довжина паролю є первинним фактором стійкості: пароль зі спеціальними символами довжиною 8 символів може бути зламаний менше ніж за годину, тоді як проста фраза довжиною 12 символів вимагає понад 200 років при використанні тих самих обчислювальних ресурсів [16].

Зручність запам’ятовування оцінює когнітивне навантаження на користувача. Люди мають обмежену здатність запам’ятовувати складні, довільні секрети, тому вони часто обирають паролі, які легко вгадати [11]. Запам’ятовуваність парольної фрази значно покращується при регулярному використанні – послідовність з шести або семи випадково обраних слів може здаватися складною спочатку, але з часом стає звичною [17].

Поведінка користувачів є критичним фактором реальної безпеки. Більше 60% людей повторно використовують паролі, і лише 12% використовують різні паролі для усіх облікових записів [18]. З розвитком систем на основі штучного інтелекту з'являються нові типи атак, які можуть впливати на захищеність автентифікаційних систем [5]. Приблизно 79% користувачів менеджерів паролів обирають безкоштовні рішення, однак багато користувачів все ще покладаються на власну пам’ять для запам’ятовування паролів [19].

На основі цих критеріїв було складено порівняльну таблицю 1.1, яка демонструє сильні та слабкі сторони кожного методу.

Таблиця 1.1 – Порівняння методів автентифікації на основі фактору знання

Метод	Практичність	Стійкість до атак	Зручність запам’ятовування	Коментар
Пароль	Висока	Залежить від довжини: 8 симв. < 1 год, 12 симв. > 200 років	Низька для випадкових, середня для запам’ятовуваних	Найпоширеніший метод
PIN-код	Висока	Низька без обмеження спроб, 96% зламуються < 1 сек	Висока	Зазвичай комбінується з додатковими механізмами захисту
Секретні питання / когнітивні паролі	Висока	Дуже низька	Висока	Застосовуються як резервний метод автентифікації
Одноразові паролі	Середня	Висока завдяки обмеженню часу, не стійкі до фішингу	Не потрібно запам’ятовувати	Потребує додаткового носія (телефон, токен)
Мнемонічні фрази	Висока	Висока при використанні великого словника	Висока, покращується з практикою	Сучасний підхід, який легко відтворюється, проте зберігає стійкість

Методи автентифікації на основі фактору знання мають низку переваг, що зумовили їх широку популярність у сучасних інформаційних системах. Насамперед вони відзначаються простотою впровадження, адже не потребують додаткового обладнання, спеціальних сенсорів або фізичних носіїв – користувач просто вводить відомий йому пароль чи відповідь на запитання. Такий підхід звичний більшості користувачів. Крім того, ці методи є економічно доцільними, адже для їх реалізації достатньо бази даних із секретами та механізму перевірки [2-4].

Попри зручність, фактор знання має і суттєві недоліки. Головною проблемою є безпека – секретна інформація, якою оперує користувач, часто може бути вгадана або знайдена з відкритих джерел. Наприклад, відповіді на типові питання безпеки (на кшталт «Ім’я першого домашнього улюбленця» чи «Місто народження») легко з’ясувати через соціальні мережі або попередні витоки даних.

Додатковою складністю є людський фактор: чим складнішим є секрет для підвищення рівня безпеки, тим більша ймовірність, що користувач його втратить [20].

Ще однією проблемою є технічна уразливість – методи знання схильні до словникових атак, підбору паролів і соціальної інженерії, особливо коли користувачі повторно їх використовують або обирають надто прості комбінації [21]. Внаслідок цього такі методи не є дуже безпечними для систем, що працюють із критичними даними. Зокрема, Національний інститут стандартів і технологій США (NIST) зазначає, що фактор знання не може розглядатися як самодостатній спосіб автентифікації, а має бути лише частиною багатофакторної схеми.

Хоча багатофакторна автентифікація (2FA) значно підвищує безпеку, вона не завжди зручна для користувача та не вирішує проблему складності запам’ятовування паролів. Користувачі змушені одночасно управляти кількома факторами – паролями, токенами або кодами з SMS/застосунків, що може призводити до помилок, забуття або навіть відмови від застосування додаткового захисту.

Таким чином, класичні методи автентифікації на основі знання залишаються актуальними, проте вони обмежуються когнітивними можливостями користувачів та ризиком компрометації через повторне використання або слабкі комбінації. У цьому контексті мнемонічні фрази становлять більш збалансований підхід, який поєднує стійкість та легкість запам’ятовування.

1.3 Аналіз засобів автентифікації на основі фактору знання

Методи автентифікації на основі фактору знання залишаються одними з найбільш поширених та практичних у сучасних інформаційних системах. Вони передбачають, що користувач демонструє знання секретної інформації, відомої лише йому, для підтвердження своєї особи. До таких методів належать класичні паролі, PIN-коди, секретні питання та когнітивні паролі, а також одноразові паролі (OTP) і більш сучасні підходи, зокрема мнемонічні фрази.

Паролі та PIN-коди залишаються базовим засобом автентифікації. Вони реалізуються практично у всіх веб-сервісах, корпоративних системах та мобільних застосунках. Для підвищення безпеки сучасні системи використовують механізми гешування паролів, політики складності та обмеження кількості спроб входу. PIN-коди часто комбінуються з фізичним носієм або додатковим фактором, наприклад, з карткою або токеном, щоб знизити ризик компрометації секрету. Цей підхід простий у впровадженні та зручний для користувача, проте він уразливий до словникових атак, перебору та повторного використання паролів.

Секретні питання або когнітивні паролі широко використовуються як резервний метод відновлення доступу, коли користувач забув основний пароль. Вони дозволяють користувачеві самостійно відновити доступ до облікового запису без залучення адміністратора системи. Водночас цей метод піддається ризику соціальної інженерії, оскільки відповіді на питання, наприклад про ім’я домашнього улюбленця чи улюблену школу, часто можна дізнатися з відкритих джерел або соціальних мереж. Дослідження [22] показують, що близько 20%-22% користувачів забувають відповіді на свої секретні питання протягом трьох місяців.

Одноразові паролі (OTP) реалізують динамічну секретну інформацію, яка дійсна лише протягом короткого проміжку часу або одного сеансу. Для цього використовують апаратні токени, мобільні застосунки (Google Authenticator [23], Authy [23]) або SMS-повідомлення [23]. Одноразові паролі значно підвищують безпеку, оскільки унеможливлюють повторне використання секрету, проте потребують додаткового носія або синхронізації з сервером. Впровадження OTP у поєднанні з класичним паролем або PIN-кодом дозволяє реалізувати багатофакторну автентифікацію, що відповідає сучасним вимогам безпеки [24].

Сучасні веб-сервіси та менеджери паролів (LastPass [25], Bitwarden [26], 1Password [27]) реалізують автоматичне генерування складних паролів. Вони дозволяють отримати комбінації символів без додаткового навантаження на користувача, зберігають їх у зашифрованому вигляді та підставляють у форми входу. Такі рішення зменшують ризик використання слабких або повторюваних

паролів, але не вирішують проблему їх запам’ятовування, особливо якщо користувач не користується менеджером постійно.

Мнемонічні фрази представляють собою сучасний підхід до реалізації фактору знання, що поєднує зручність запам’ятовування та високу криптографічну стійкість. Такі фрази формуються з набору асоційованих слів або символів, які користувач може легко відтворити.

Приклади застосування мнемонічних фраз можна знайти у криптографічних гаманцях (MetaMask [28], Trezor [29]), де seed-фрази використовуються для генерування криптографічних ключів; у менеджерах, які дозволяють створювати запам’ятовувані паролі з реальних слів; а також у системах відновлення доступу (Microsoft Account Recovery Code [30], Apple Recovery Key [31]).

Мнемонічні фрази допомагають підвищити зручність використання та зменшити ризик помилок, пов’язаних із забуванням чи складністю введення. Завдяки асоціативності вони краще запам’ятовуються, ніж випадкові послідовності символів, і водночас можуть забезпечувати високу стійкість до перебору.

Кожен із засобів автентифікації на основі знання має свої переваги та недоліки. Паролі та PIN-коди зручні та поширені, але вразливі до атак перебору та словникових атак. Секретні питання підвищують відновлюваність доступу, але легко піддаються соціальній інженерії. Одноразові паролі забезпечують високий рівень безпеки, проте потребують додаткових носіїв або синхронізації. Мнемонічні фрази дозволяють досягти балансу між зручністю та стійкістю до атак, що робить їх перспективним засобом для подальшого дослідження та впровадження.

Аналіз сучасних засобів автентифікації на основі фактору знання показує, що класичні методи залишаються актуальними через простоту впровадження, проте вони обмежуються когнітивними здібностями користувачів (складні паролі важко запам’ятати) та технічними вразливостями (атаки перебору, соціальна інженерія, фішинг, OSINT). Користувачі часто обирають прості або повторювані комбінації, що критично знижує рівень безпеки, або записують складні паролі у небезпечних місцях.

Сучасні технології двофакторної автентифікації та біометричні системи значно підвищують рівень захисту, проте їх впровадження вимагає додаткового обладнання, програмного забезпечення та технічної підтримки, що не завжди є доступним або економічно виправданим для всіх категорій користувачів та систем. Мнемонічні фрази представляють собою компромісне рішення, яке поєднує асоціативну запам’ятовуваність з високою варіативністю комбінацій. У наступних розділах розробляються два методи генерування паролів: перший базується на трансформації користувацьких даних через таблицю замін з додаванням випадкової компоненти, другий використовує мнемонічні принципи для формування запам’ятовуваних паролів. Обидва методи спрямовані на зменшення когнітивного навантаження при збереженні стійкості.

1.4 Постановка завдання дослідження

Аналіз сучасних методів автентифікації на основі фактору знання показав, що існує суттєва потреба у забезпеченні балансу між безпекою та зручністю використання. Класичні паролі та PIN-коди широко застосовуються, проте користувачі часто вибирають слабкі або повторно використовують паролі, що підвищує ризик компрометації. Секретні питання та когнітивні паролі допомагають відновлювати доступ, але легко піддаються атакам соціальної інженерії. Одноразові паролі та токени підвищують рівень безпеки, однак потребують додаткових носіїв та синхронізації, що зменшує їх практичність у щоденному використанні.

Дослідження сучасних генераторів паролів демонструють стійкість у створенні складних комбінацій символів, проте такі рішення не завжди враховують легкість запам’ятовування для користувача, що є критично важливим для зниження ймовірності втрати або компрометації паролю.

У зв’язку з цим актуальним завданням є розробка методів та програмної бібліотеки для генерування факторів автентифікації на основі паролів та мнемонічних фраз, які забезпечують одночасно високу стійкість до атак та зручність запам’ятовування. Такий підхід дозволяє зменшити когнітивне

навантаження на користувача, підвищити безпеку автентифікації та покращити практичність її застосування в різних інформаційних системах, включаючи системи середньої та високої критичності.

1.5 Висновки з розділу

У цьому розділі було проведено комплексний аналіз методів автентифікації користувачів на основі фактору знання, включаючи огляд підходів, типів методів та відомих засобів їх реалізації. Дослідження показало, що класичні методи, такі як паролі та PIN-коди, залишаються найпоширенішими через простоту впровадження та зручність для користувачів, проте вони мають обмежену стійкість до атак, особливо при повторному використанні або виборі простих комбінацій.

Секретні питання та когнітивні паролі використовуються переважно як резервні механізми відновлення доступу, проте їхній захисний потенціал знижується через уразливість до соціальної інженерії та забування користувачами відповідей. Одноразові паролі та токени підвищують безпеку, але потребують додаткових носіїв та синхронізації, що зменшує їхню практичність у повсякденному використанні.

Аналіз сучасних засобів генерування паролів та альтернативних підходів показав, що важливо забезпечити баланс між стійкістю та зручністю запам’ятовування. Особливо перспективним напрямом є застосування мнемонічних фраз, які дозволяють підвищити стійкість до атак і одночасно спрощують запам’ятовування факторів автентифікації користувачем.

Таким чином, проведений аналіз дозволив підтвердити актуальність розробки нових методів та програмного засобу генерування факторів автентифікації на основі фактору знання, що поєднує безпеку, практичність і зручність використання, та створює основу для подальших досліджень і математичного моделювання у наступному розділі магістерської роботи.

2 МЕТОД АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ ПАРОЛІВ

2.1 Узагальнений математичний опис генерування паролів

Для формалізації процесу генерування паролів необхідно визначити математичну модель, яка описує структуру паролю, простір можливих значень та метрики якості його формування. Такий підхід дозволяє обґрунтувати вибір параметрів генерування та створити основу для розробки конкретних алгоритмів.

Теоретико-множинний підхід є найбільш доречним для математичного опису процесу генерування паролів з низки причин. По-перше, він дозволяє формально описати дискретні об’єкти (символи, алфавіти, обмеження) та операції над ними. По-друге, підхід природним чином представляє послідовний характер генерування через поняття станів системи та функцій переходу. Теоретико-множинний опис забезпечує можливість строгого математичного аналізу властивостей процесу генерування, таких як повнота простору паролів, коректність виконання обмежень та оцінка складності алгоритмів. Крім того, цей підхід широко використовується в теорії формальних систем та криптографії, що забезпечує сумісність з відомими дослідженнями в галузі інформаційної безпеки.

Згідно з теоретико-множинним підходом, систему генерування паролів можна представити як динамічну систему виду:

$$S=\langle T, X, Y, Z, z(t), C\rangle \qquad (2.1)$$

де T – модельний час, що представляє дискретні етапи процесу генерування паролю; X – множина вхідних параметрів (алфавіт допустимих символів, показники якості); Y – множина вихідних значень (згенерований пароль); Z – простір станів системи (згенеровані претенденти паролів, їх метрики якості, результати генерування випадкових/псевдовипадкових чисел або random oracle); z(t) – функція переходу між станами; C – множина допустимих процесів генерування.

Для системи генерування паролів модельний час визначається як дискретна множина:

$$T=\{0, 1, 2, \ldots, k\} \quad (2.2)$$

де k – кількість етапів обробки в процесі генерування, а кожен момент часу t відповідає черговому кроку трансформації даних.

Множина вхідних параметрів системи має вигляд:

$$X=\{ P_{user}, n_{target}, A, R, M\} \quad (2.3)$$

де P_{user} – користувацькі дані (початкова фраза або слово); n_{target} – цільова довжина паролю; n – задана довжина паролю; A – алфавіт символів; R – множина обмежень на структуру паролю; M – множина правил трансформації (таблиця замін).

Алфавіт символів представляє собою об’єднання чотирьох категорій:

$$A=A_{upper} \cup A_{lower} \cup A_{digit} \cup A_{special} \quad (2.4)$$

де A_{upper} – множина великих літер латинського алфавіту, A_{lower} – множина малих літер латинського алфавіту, A_{digit} – множина цифр, $A_{special}$ – множина спеціальних символів.

Повний перелік наведено у додатку.

Множина обмежень R визначає мінімальні вимоги до структури паролю:

$$R=\{R_1, R_2, R_3, R_4\} \quad (2.5)$$

де R_1 – пароль повинен містити принаймні одну велику літеру, R_2 – пароль повинен містити принаймні одну малу літеру, R_3 – пароль повинен містити принаймні одну цифру, R_4 – пароль повинен містити принаймні один спеціальний.

Таблиця замін M визначає правила трансформації символів:

$$m(\cdot):A \rightarrow A \quad (2.6)$$

Кожен символ може мати декілька варіантів заміни, наприклад:

– m(a)∈{@, 4, A};

– m(e)∈{3, E};

– m(i)∈{!, 1, I};

– m(o)∈{0, O};

– m(s)∈{, 5, S}.

Детальна таблиця замін наведена у таблиці В.2 додатку В.

Вихідне значення системи представляє собою згенерований пароль:

$$Y=P=\{c_1,c_2,\ldots,c_{ntarget}\} \quad (2.7)$$

де $c_1 \in A$ – символи паролю.

Простір станів системи Z характеризує поточний стан процесу генерування:

$$z(t)=\{P_{user}, P_{transformed}, P_{padding}, P_{result}\} \quad (2.8)$$

де $P_{transformed}$ – трансформовані дані після застосування таблиці замін довжиною x′, $P_{padding}$ –додатково згенерована частина довжиною y = n_{target} −x′, P_{result} – згенерований пароль.

Функція переходу між станами визначається як:

$$z(t+1)=f(z(t), \theta_t) \quad (2.9)$$

де θ_t – множина параметрів трансформації на кроці t, що включає вибір правил заміни з M, генерування випадкових символів та метод комбінування.

Початковий стан системи:

$$z(0)=\{ P_{user,} \emptyset, \emptyset, \emptyset\{R_1 : false, R_2 :false, R_3: false, R_4: false\}\} \quad (2.10)$$

Кінцевий стан системи досягається при t = n за умови виконання всіх обмежень:

$$z(k) : | P_{result} | = n_{target} \wedge \forall R_i \in R : R_i(P_{result}) = true \quad (2.11)$$

Для комбінування компонентів визначено 3 методи:

$$\Lambda=\{\lambda_1, \lambda_2, \lambda_3\} \quad (2.12)$$

де λ_1 – розділення: додаткова частина ділиться навпіл і розміщується з обох боків, $P_{result} = P_1 + P_{transformed} + P_2$, де P_1 та P_2 – частини $P_{padding}$, λ_2 – префікс: додаткова частина на початку, $P_{result} = P_{padding} + P_{transformed}$, λ_3 – суфікс: додаткова частина в кінці, $P_{result} = P_{transformed} + P_{padding}$.

Вибір методу комбінування здійснюється випадково з рівномірним розподілом:

$$P(\lambda=\lambda_i)=\frac{1}{3} \quad (2.13)$$

Практичний простір паролів визначається як:

$$N_{prac} = \prod_{i=1}^{x} |m(u_i)| \times |A|^y \times 3 \quad (2.14)$$

де $|m(u_i)|$ – кількість варіантів заміни для символу u_i; y – довжина додаткової частини; 3 – кількість методів комбінування.

Для додаткової частини кожен символ обирається рівномірно випадково:

$$P(g_i = a_j) = \frac{1}{|A|} \quad (2.15)$$

Додаткова частина повинна компенсувати відсутні категорії символів у трансформованих даних для виконання всіх обмежень R.

Таким чином, математичний опис генерування паролів забезпечує формальне обґрунтування процесу створення паролів шляхом комбінування трансформованих користувацьких даних з випадково згенерованими символами. Наведений теоретико-множинний підхід дозволяє описати систему як динамічний процес з дотриманням заданих обмежень, використанням таблиці замін та варіативних методів комбінування.

2.2 Узагальнене представлення методу

Метод генерування паролів на основі користувацьких даних полягає у трансформації вхідної інформації, наданої користувачем, з подальшим доповненням випадково згенерованими символами до досягнення заданої довжини та виконання всіх обмежень безпеки. Метод складається з послідовності кроків, які забезпечують створення стійкого паролю зі збереженням елементів, що легше запам’ятовуються користувачем [1, 5].

Крок 1. Ініціалізація.

На початковому етапі визначаються такі вхідні параметри системи:

- цільова довжина паролю (за замовчуванням 12 символів);
- алфавіт символів з чотирьох категорій: великих і малих літер латинського алфавіту (по 26 символів у кожній), цифр (10 символів) та спеціальних символів (32 символи). Таким чином, загальна потужність алфавіту становить 94 символи;
- множина обмежень на структуру паролю;
- таблиця замін для трансформації символів.

Крок 2. Введення користувацьких даних.

Користувач вводить початкові дані – слово, фразу або комбінацію символів, яку він може запам’ятати. Вхідні дані позначено як послідовність символів довжиною x.

Крок 3. Визначення довжини та спотворення вхідних даних.

На цьому кроці визначається довжина вхідних даних і застосовується таблиця замін для їх трансформації. Для кожного символу вхідних даних застосовується функція заміни з таблиці, яка перетворює його у інший символ або комбінацію символів.

Для кожного символу вхідних даних випадково обирається один з можливих варіантів заміни згідно з таблицею. Детальна таблиця замін наведена в додатку В.

Визначається довжина трансформованих даних *x’*. Ця довжина може відрізнятися від початкової, якщо деякі символи замінюються на послідовності символів.

Крок 4. Генерування додаткової частини паролю.

Обчислюється необхідна кількість додаткових символів як різниця між цільовою довжиною паролю та довжиною трансформованих даних: $y = 12 - x'$.

Генерується додаткова частина паролю довжиною y символів. Кожен символ обирається випадково з алфавіту за допомогою криптографічно стійкого генератора псевдовипадкових чисел (CSPRNG).

При генеруванні додаткової частини враховується стан виконання обмежень. Якщо трансформовані дані не містять символів певних категорій (великі літери, малі літери, цифри, спеціальні символи), то додаткова частина повинна обов'язково включати відсутні категорії для забезпечення виконання всіх обмежень.

Крок 5. Комбінування компонентів паролю.

На цьому кроці здійснюється об'єднання трансформованих користувацьких даних з додатково згенерованою частиною. Випадково обирається один з трьох способів комбінування з однаковою ймовірністю: розділення, префікс, суфікс. Розділення полягає в тому, що додаткова частина ділиться на два фрагменти приблизно однакової довжини. Перша частина розміщується на початку, трансформовані дані – посередині, друга частина додаткових символів – в кінці. При способі комбінування як префіксу додаткова частина повністю розміщується на початку перед трансформованими даними. Якщо ж було обрано суфіксний спосіб комбінування, то додаткова частина повністю розміщується в кінці після трансформованих даних.

Крок 6. Валідація результату.

Перевіряється виконання всіх обмежень для згенерованого паролю:

- довжина паролю дорівнює цільовій (12 символів);
- пароль містить принаймні одну велику літеру;
- пароль містить принаймні одну малу літеру;
- пароль містить принаймні одну цифру;
- пароль містить принаймні один спеціальний символ.

Якщо всі обмеження виконані, процес завершується успішно і пароль повертається користувачеві.

Якщо хоча б одне обмеження не виконане, здійснюється повернення до кроку 4 з повторним генеруванням додаткової частини з урахуванням відсутніх категорій символів.

Крок 7. Формування результату.

Згенерований пароль повертається користувачеві як результат роботи методу.

Згенерований пароль використовується як частина механізму автентифікації користувача в системі. Після створення пароль зберігається у системі в гешованому вигляді із застосуванням криптографічно стійких функцій гешування, що забезпечує захист від несанкціонованого доступу навіть у разі компрометації бази даних.

Під час процедури входу користувач вводить запам’ятований пароль, сформований за допомогою описаного методу. Система обчислює геш введеного паролю та порівнює його з еталонним значенням у базі даних. Якщо значення збігаються, користувач успішно автентифікується; у протилежному випадку доступ відхиляється.

Представлений метод забезпечує баланс між безпекою та зручністю використання. Трансформація користувацьких даних через таблицю замін зберігає асоціативний зв’язок з початковою інформацією, що полегшує запам’ятовування, водночас доповнення випадковими символами та варіативність методів комбінування значно розширюють простір можливих паролів та підвищують стійкість до атак підбору.

2.3 Алгоритм формування паролю

Алгоритм формування паролю реалізує послідовність операцій, спрямованих на перетворення користувацьких даних у стійкий пароль заданої довжини з дотриманням усіх обмежень безпеки. Алгоритм базується на комбінуванні детермінованих трансформацій вхідних даних та випадкового

генерування додаткових символів. Узагальнений алгоритм представлений на рисунку 2.1, який демонструє послідовність основних етапів від ініціалізації параметрів до виведення результату.

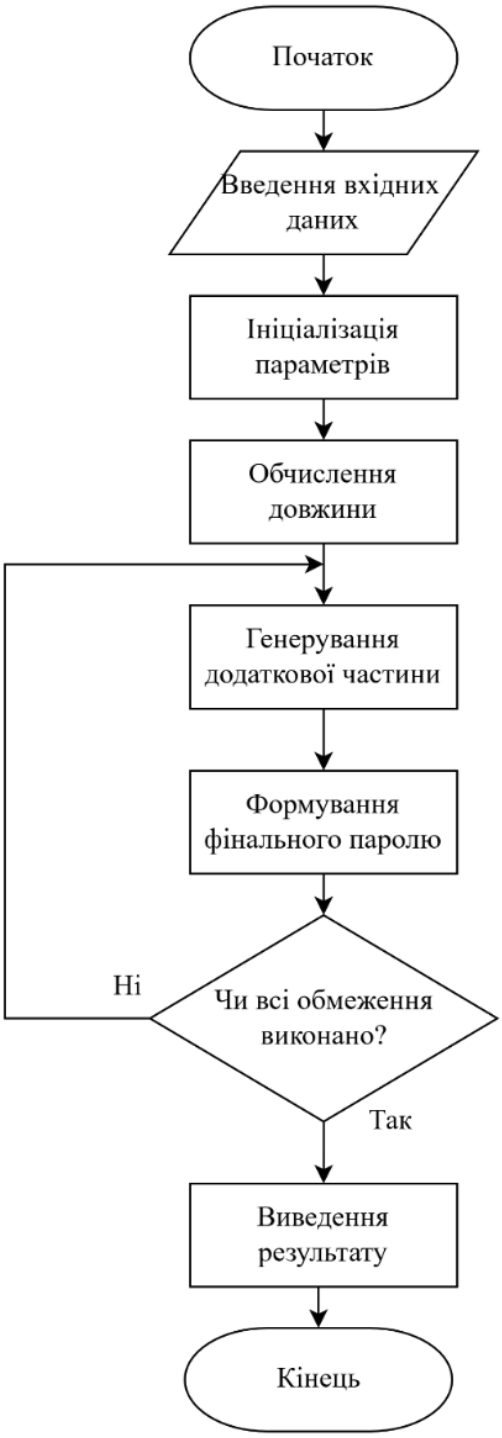


Рисунок 2.1 – Узагальнений алгоритм формування паролю

Процес починається з отримання вхідних даних від користувача. Ці дані можуть бути представлені словом, фразою або будь-якою послідовністю символів,

яку користувач здатен запам’ятати. На відміну від класичних методів генерування паролів, де користувач отримує повністю випадковий набір символів, даний алгоритм зберігає зв’язок з початковими даними, що полегшує запам’ятовування.

Після отримання вхідних даних система визначає їх довжину та переходить до етапу трансформації. Цей процес детально представлено на рисунку 2.2, де показано механізм застосування таблиці замін для кожного символу вхідної послідовності.

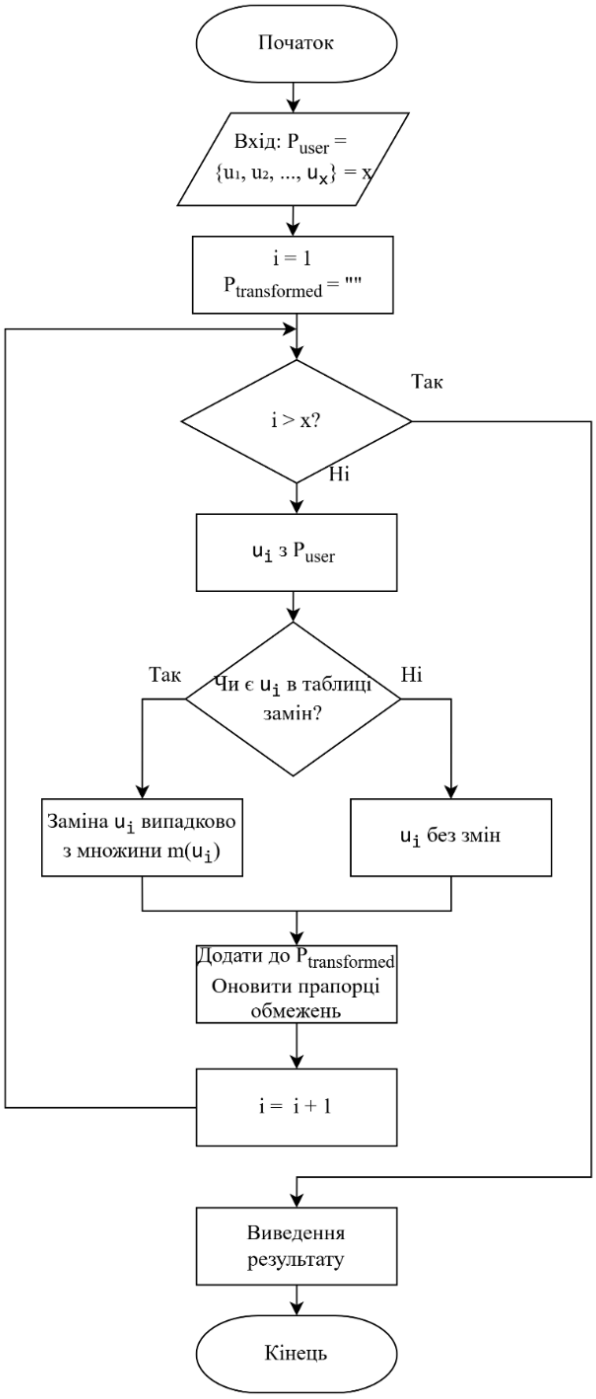


Рисунок 2.2 – Алгоритм модифікування вхідних даних

Таблиця замін містить відповідності між звичайними символами та їх можливими замінниками з різних категорій алфавіту. Для кожного символу вхідних даних існує декілька варіантів заміни, один з яких обирається випадково за допомогою криптографічно стійкого генератора псевдовипадкових чисел. Така трансформація дозволяє значно ускладнити пароль, зберігаючи при цьому його запам’ятовуваність через асоціативні зв’язки. Наприклад, літера ‘а’ може бути замінена на символ ‘@’ або цифру ‘4’, літера ‘е’ – на цифру ‘3’, літера ‘о’ – на цифру ‘0’ відповідно до таблиці В.2. Користувач може легко відтворити логіку цих замін завдяки їх інтуїтивності та схожості за формою або звучанням. Якщо символ відсутній у таблиці замін, він залишається без змін у трансформованих даних, що забезпечує коректну обробку будь-яких вхідних послідовностей.

Після завершення трансформації система аналізує отримані дані та визначає, скільки додаткових символів необхідно згенерувати для досягнення цільової довжини паролю. За замовчуванням цільова довжина становить 12 символів, що відповідає сучасним рекомендаціям з інформаційної безпеки. Різниця між цільовою довжиною та довжиною трансформованих даних визначає кількість символів, які необхідно додатково згенерувати.

Генерування додаткової частини паролю здійснюється за допомогою криптографічно стійкого генератора псевдовипадкових чисел. Кожен символ додаткової частини обирається випадково з повного алфавіту, що включає великі та малі літери, цифри та спеціальні символи. При цьому алгоритм враховує стан виконання обмежень безпеки. Якщо трансформовані користувацькі дані не містять певних категорій символів, наприклад великих літер або спеціальних символів, то при генеруванні додаткової частини обов’язково включаються відсутні категорії. Це гарантує, що фінальний пароль відповідатиме всім встановленим вимогам.

Наступним етапом є комбінування трансформованих користувацьких даних з додатково згенерованою частиною. Детальна схема цього процесу представлена на рисунку 2.4, який ілюструє 3 можливі стратегії об’єднання компонентів паролю. Алгоритм передбачає випадковий вибір однієї з цих стратегій з рівномірним розподілом ймовірностей.

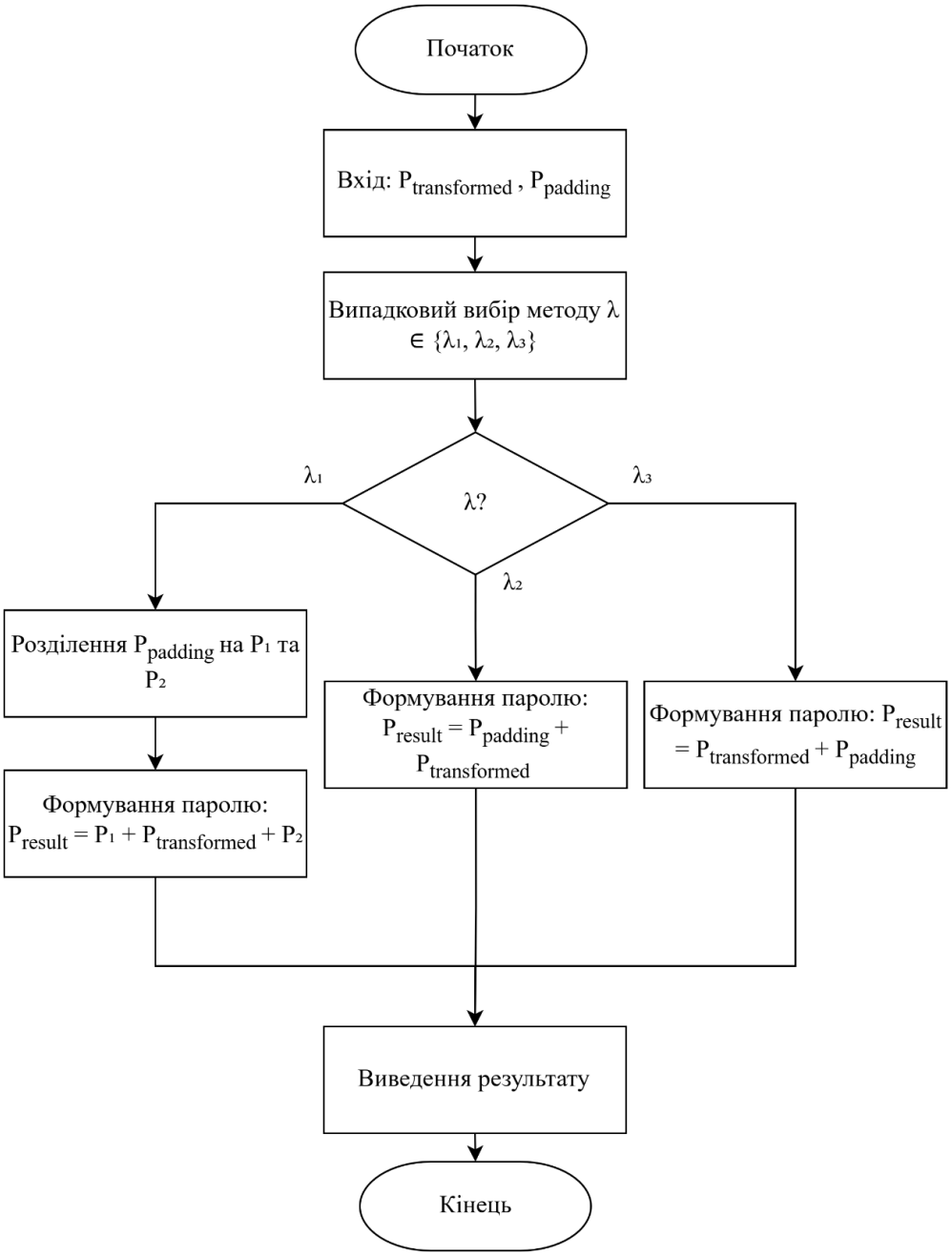


Рисунок 2.3 – Алгоритм комбінування вхідних даних з додатковою частиною

Перша стратегія, позначена як λ_1, полягає у розділенні додаткової частини на два фрагменти приблизно рівної довжини та розміщенні трансформованих даних між ними. Друга стратегія λ_2 розміщує всю додаткову частину на початку перед трансформованими даними, формуючи структуру з випадковим префіксом. Третя стратегія λ_3 розміщує додаткову частину в кінці після трансформованих даних, створюючи структуру з випадковим суфіксом. Така варіативність значно збільшує простір можливих паролів навіть при однакових вхідних даних

користувача, оскільки кожне виконання алгоритму може обрати різну стратегію комбінування.

Після формування вихідного паролю система здійснює валідацію для підтвердження виконання всіх обмежень. Перевіряється відповідність довжини паролю цільовому значенню та наявність символів усіх чотирьох категорій, а саме великих літер, малих літер, цифр та спеціальних символів з множини. Якщо хоча б одна умова не виконана, алгоритм повертається до етапу генерування додаткової частини та повторює процес з урахуванням виявлених недоліків. При повторному генеруванні система аналізує, які саме категорії символів відсутні в поточному паролі, та забезпечує їх обов'язкове включення в нову додаткову частину. Якщо всі умови виконані, згенерований пароль повертається користувачеві як результат роботи алгоритму [1, 4, 5].

Згенерований пароль використовується в процесі автентифікації користувача. Після створення пароль зберігається у системі в гешованому вигляді із застосуванням криптографічно стійких функцій гешування. Під час входу користувач вводить пароль, система обчислює його геш та порівнює зі значенням у системі.

Представлений алгоритм характеризується детермінованістю трансформації вхідних даних та випадковим генеруванням додаткової частини, що забезпечує баланс між передбачуваністю для користувача та непередбачуваністю для зловмисника. Використання таблиці замін дозволяє зберегти мнемонічний зв'язок з початковими даними, водночас значно ускладнюючи структуру паролю. Варіативність методів комбінування та випадковість вибору конкретних замін з таблиці забезпечують високу різноманітність результатів навіть при однакових вхідних даних різних користувачів.

2.4 Теоретичне оцінювання стійкості методу

Стійкість методу генерування паролів до атак методом перебору визначається кількістю операцій, необхідних для зламу паролю. Для коректного оцінювання стійкості запропонованого методу необхідно розглянути два сценарії

атаки: оптимістичний варіант, коли зловмисник не володіє інформацією про метод генерування, та песимістичний варіант, коли зловмисник знає алгоритм формування паролю та навіть вхідне слово користувача.

У випадку, коли зловмисник не володіє інформацією про метод генерування паролю та не знає алгоритм формування, для нього пароль є довільною послідовністю символів з певного алфавіту. Стійкість паролю в цьому сценарії визначається виключно довжиною паролю та потужністю використовуваного алфавіту. Кількість можливих комбінацій для повного перебору визначається формулою [13]:

$$N_{brute} = |A|^n \tag{2.16}$$

де |A| – потужність алфавіту символів, n – довжина паролю.

Ця оцінка є універсальною для будь-якого методу генерування паролів та не залежить від конкретної реалізації алгоритму. Вона представляє оптимістичний сценарій для методу генерування, оскільки відображає максимальну можливу стійкість паролю заданої довжини. Зловмисник змушений перевіряти всі можливі комбінації символів без можливості використання додаткової інформації про структуру паролю.

Для запропонованого методу з мінімальною довжиною паролю n_{min} та алфавітом потужністю |A|, мінімальна стійкість до brute force атаки становить:

$$N_{bf} = |A|^{nmin} \tag{2.17}$$

Цей показник визначає базовий рівень захисту, який гарантовано забезпечується методом незалежно від параметрів генерування. Навіть у найгіршому випадку зловмисник не може обійти цей бар’єр без знання додаткової інформації про структуру паролю.

Для конкретної реалізації запропонованого методу визначено наступні параметри: цільова довжина паролю $n_{target} = 12$ символів, алфавіт $|A| = 94$ символи. При цих параметрах мінімальна стійкість до brute force атаки становить:

$$N_{bf} = 94^{12} \approx 4.75 \times 10^{23} \text{ операцій}$$

Це значення представляє кількість операцій перевірки, необхідних для гарантованого знаходження паролю методом повного перебору. За сучасними стандартами інформаційної безпеки, простір порядку 10^{24} операцій вважається достатнім для захисту від brute force атак з використанням наявних обчислювальних потужностей. Навіть при швидкості перевірки один мільярд паролів за секунду, що є характерним для спеціалізованого обладнання на основі графічних процесорів, час повного перебору становить [32]:

$$T_{bf} = 4.75 \times 10^{23} / 10^{14} \approx 4.75 \times 10^{9} \text{ секунд} \approx 15 \text{ мільйонів років}$$

Такий рівень стійкості є достатнім для протидії brute force атакам навіть з урахуванням зростання обчислювальних потужностей у найближчі десятиліття. Це представляє оптимістичний сценарій захисту, коли зловмисник не має додаткової інформації про метод генерування.

Для оцінювання стійкості методу в найгіршому випадку необхідно припустити, що зловмисник володіє повною інформацією про алгоритм формування паролю та знає вхідне слово, яке використовував користувач. У такому сценарії простір можливих паролів значно звужується, оскільки зловмисник може обмежити перебір виключно варіантами, що відповідають структурі методу.

У цьому випадку кількість операцій перебору визначається трьома факторами: варіативністю замін символів вхідного слова згідно з таблицею замін M, кількістю випадково згенерованих символів та кількістю методів комбінування. Для формалізації цієї оцінки введено такі позначення: L_{word} – довжина вхідного слова користувача, n_{min} – мінімальна довжина паролю, k_{avg} – середня кількість варіантів заміни для одного символу згідно з таблицею замін M, $|A|$ – потужність алфавіту, λ – кількість методів комбінування.

Кількість випадково згенерованих символів визначається як різниця між мінімальною довжиною паролю та довжиною вхідного слова:

$$L_{random} = n_{min} - L_{word} \qquad (2.18)$$

За умови, що $L_{word} \leq n_{min}$, генерування додаткової частини не відбувається. Якщо $L_{word} > n_{min}$, то після трансформації вхідного слова через таблицю замін довжина паролю перевищує мінімальне значення і $L_{random} = 0$, тобто додаткове генерування випадкових символів не виконується.

Для визначення середньої кількості варіантів заміни k_{avg} необхідно проаналізувати таблицю замін M. Згідно з визначеною таблицею, для малих латинських літер виділено три групи: символи з 4 варіантами заміни (включаючи оригінальний символ), символи з 3 варіантами заміни та символи з 2 варіантами заміни (зміна регістру). Кількість літер у кожній групі відповідно становить 6, 9 та 11 символів).

Для обчислення середнього значення k_{avg} варто розглянути повний розподіл для малих літер:

– 6 символів × 4 варіанти = 24;

– 9 символів × 3 варіанти = 27;

– 11 символів × 2 варіанти = 22.

Сума варіантів: 24 + 27 + 22 = 73 варіанти на 26 символів.

$$k_{avg} = 73 / 26 \approx 2.81 \text{ варіанти на символ}$$

Для спрощення подальших розрахунків та забезпечення консервативної оцінки використовуватиметься округлене значення $k_{avg} \approx 3.0$ варіанти на символ, що дещо завищує можливості зловмисника і забезпечує більш обережну оцінку стійкості методу. Це значення є реалістичною оцінкою для збалансованої таблиці замін, яка забезпечує достатню варіативність без надмірного ускладнення.

Загальна кількість можливих паролів при відомому вхідному слові визначається формулою:

$$N = (k_{avg})^{L_{word}} \times |A|^{L_{random}} \times \lambda \qquad (2.19)$$

де перший множник $(k_{avg})^{L_{word}}$ представляє кількість варіантів трансформації вхідного слова через таблицю замін, другий множник $|A|^{L_{random}}$ – кількість способів згенерувати випадкову частину паролю, третій множник λ – кількість методів комбінування компонентів ($\lambda = 3$ для запропонованого методу).

Ця формула представляє песимістичний сценарій, оскільки зловмисник використовує максимум доступної інформації для звуження варіантів. Проте навіть у цьому випадку стійкість методу залежить від довжини випадкової компоненти, яка визначається різницею між цільовою довжиною паролю та довжиною вхідного слова.

Нижче наведено таблицю 2.1, яка демонструє залежність стійкості методу від довжини вхідного слова в обох розглянутих сценаріях.

Таблиця 2.1 – Оцінка стійкості методу для різних сценаріїв атаки

Довжина введеного слова	Lrandom	Brute force (оптим.)	Known word (песим.)	Час перебору (песим.)
0 символів	12	4.75×10^{23}	4.75×10^{23}	~15 млн років
1 символ	11	4.75×10^{23}	1.42×10^{23}	~4.5 млн років
2 символи	10	4.75×10^{23}	2.37×10^{22}	~751 тис. років
3 символи	9	4.75×10^{23}	7.17×10^{20}	~22.7 тис. років
4 символи	8	4.75×10^{23}	5.84×10^{19}	~1851 рік
5 символи	7	4.75×10^{23}	8.26×10^{17}	~26.2 років
6 символів	6	4.75×10^{23}	3.33×10^{15}	~38.6 днів
7 символів	5	4.75×10^{23}	6.78×10^{13}	~21.5 годин
8 символів	4	4.75×10^{23}	2.03×10^{12}	~33.8 хвилин
10 символів	2	4.75×10^{23}	6.09×10^{9}	~101 секунда
12 символів	0	4.75×10^{23}	5.31×10^{6}	~0.0053 секунди

Оптимістичний сценарій однаковий для всіх випадків і при швидкості перебору 10^9 спроб/с становить приблизно 1.5×10^7 років (≈15 мільйонів років). Песимістичний сценарій (Known word) залежить від довжини вхідного слова та кількості можливих замін.

Усі оцінки в таблиці проведено з припущенням, що зловмисник має обчислювальну потужність до 10^9 спроб за секунду, що відповідає можливостям сучасних GPU-ферм у реалістичних умовах. Дане припущення є консервативним і дозволяє уніфікувати обидва сценарії.

На основі проведеного аналізу можна сформулювати конкретні рекомендації щодо обмеження довжини вхідних даних користувача для забезпечення достатнього рівня стійкості методу. Для досягнення мінімального

часу перебору порядку декількох десятків років навіть у песимістичному сценарії, довжина вхідного слова не повинна перевищувати 5 символів при цільовій довжині паролю 12 символів. При довжині вхідного слова 6 символів час перебору становить близько 39 днів, що є недостатнім для систем.

Важливо відзначити, що навіть у песимістичному сценарії, коли зловмисник знає алгоритм та вхідне слово, запропонований метод забезпечує суттєво вищу стійкість порівняно з простою трансформацією слова без додавання випадкової компоненти. Випадкова частина паролю є ключовим фактором забезпечення стійкості методу і повинна мати достатню довжину для протидії цільовим атакам.

Щоб пароль мав достатню випадкову частину і залишався стійким до підбору, потрібно обмежити довжину вхідних даних користувача. Рекомендується, щоб початкове слово містило не більше 5 символів для звичайних випадків і до 4 символів для систем із підвищеними вимогами безпеки при загальній довжині паролю 12 символів.

2.5 Висновки з розділу

У цьому розділі розроблено та обґрунтовано метод автентифікації користувачів на основі зрозумілих паролів, який забезпечує поєднання високого рівня безпеки з підвищеною зручністю використання. На відміну від традиційних підходів, що ґрунтуються на повністю випадковому генеруванні, запропонований метод враховує асоціативні особливості сприйняття користувача.

Запропоновано теоретико-множинний математичний опис процесу генерування зрозумілих паролів, що формалізує структуру вхідних і вихідних даних, простір станів системи та функцію переходів між ними. Такий підхід дозволив описати процес формування паролю як динамічну систему, у якій кожен стан характеризується параметрами користувацьких даних, результатами трансформації, випадкового генерування та комбінування.

Наведено узагальнене представлення методу, який включає 7 основних етапів – від ініціалізації параметрів і трансформації вхідних даних до комбінування та валідації результату. Метод забезпечує виконання встановлених обмежень

безпеки, передбачає використання криптографічно стійкого генератора псевдовипадкових чисел, а також зберігає мнемонічний зв’язок із вихідними даними користувача. Особливістю методу є використання таблиці замін, яка забезпечує варіативність відображень символів і підвищує стійкість паролю.

Розроблено алгоритм реалізації методу, який відображає основні етапи: обробку вхідних даних, застосування таблиці замін, генерування додаткової частини та комбінування компонентів паролю. Така структура дає змогу зручно реалізувати метод у програмному вигляді.

Проведено теоретичну оцінку стійкості паролів, створених за цим методом. Показано, що при довжині паролю 12 символів і використанні алфавіту з 94 символів стійкість до повного перебору в оптимістичному сценарії перевищує 10^{23} можливих комбінацій. Встановлено, що оптимальним є використання коротких вхідних слів (до 5 символів), щоб у паролі зберігалася достатня кількість випадкових символів для забезпечення стійкості понад 26 років навіть у песимістичному сценарії, коли зловмисник знає алгоритм та вхідне слово.

Отже, розроблений метод дозволяє створювати паролі, які є зрозумілими для користувача, але складними для зловмисників. Він забезпечує баланс між безпекою та зручністю використання і може бути застосований у сучасних системах автентифікації [1].

3 МЕТОД АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ МНЕМОНІЧНИХ ФРАЗ

3.1 Узагальнений математичний опис генерування мнемонічних фраз

Для формалізації процесу генерування мнемонічних фраз необхідно визначити математичну модель, яка описує структуру фрази, простір можливих значень, синтаксичні правила побудови та метрики якості формування. Такий підхід дозволяє обґрунтувати вибір параметрів генерування, забезпечити семантичну коректність фрази та створити основу для розробки конкретних алгоритмів.

Згідно з теоретико-множинним підходом, аналогічно математичному опису в підрозділі 2.1, процес генерування мнемонічних фраз представляється такий, що виконується динамічною системою виду:

$$S = \langle T, X, Y, Z, z(t), C \rangle \tag{3.1}$$

де T – модельний час, що представляє дискретні етапи процесу генерування фрази; X – множина вхідних параметрів (словники, структурні шаблони, параметри безпеки); Y – множина вихідних значень (згенерована мнемонічна фраза); Z – простір станів системи (частково сформовані фрази, результати вибору слів, метрики якості); z(t) – функція переходу між станами; C – множина допустимих процесів генерування.

Для системи генерування мнемонічних фраз модельний час визначається як дискретна множина:

$$T = \{0, 1, 2, ..., n\} \tag{3.2}$$

де кожен момент часу t відповідає черговому кроку вибору слова для мнемонічної фрази.

$$X = \{D, T_{pattern}, n_{words}\} \tag{3.3}$$

де D – множина словників; $T_{pattern}$ – структурний шаблон фрази; n_{words} – обрана користувачем кількість слів у фразі (5, 6 або 7)..

Множина словників представляє собою об’єднання словників за частинами мови:

$$D = D_{noun} \cup D_{adj} \cup D_{verb} \quad (3.4)$$

де D_{noun} – словник іменників, D_{adj} – словник прикметників, D_{verb} – словник дієслів.

Словники наведено у додатку Г.

Далі використовуються шаблони фраз. Вибір шаблону здійснюється користувачем перед початком генерування або визначається випадково з відповідної категорії за допомогою криптографічно стійкого генератора псевдовипадкових чисел.

Вихідне значення системи представляє собою згенеровану мнемонічну фразу:

$$Y = P_{mnemonic} = \{w_1, w_2, \ldots, w_n\} \quad (3.5)$$

де w_i – слова фрази, а їх послідовність відповідає структурному шаблону.

Простір станів системи Z характеризує поточний стан процесу генерування:

$$z(t) = \{W_{partial}, W_{used}\} \quad (3.6)$$

де $W_{partial}$ – частково сформована фраза на кроці t (містить t слів), W_{used} – множина вже використаних слів (для уникнення повторів).

Функція переходу між станами визначається як:

$$z(t+1) = f(z(t), w_t) \quad (3.7)$$

де w_t – слово, обране на кроці t з відповідного словника.

Початковий стан системи:

$$z(0) = \{\emptyset, \emptyset\} \quad (3.8)$$

де обидві множини порожні – фраза ще не почала формуватися.

Кінцевий стан системи досягається при t = n:

$$z(n): | W_{partial} | = n \wedge W_{partial} = \{ w_1, w_2, \ldots, w_n \} \quad (3.9)$$

Для кожної позиції *i* в шаблоні вибір слова здійснюється за рівномірним законом розподілу з відповідного словника за умови, що слово не входить у множину W_{used}:

$$P(w_i = w \mid w \in D_{pos}(i) \wedge w \notin W_{used}) = 1 / (|D_{pos}(i)| - | W_{used} \cap D_{pos}(i)|) \quad (3.10)$$

де $D_{pos}(i)$ – словник, відповідний і-й позиції шаблону.

Для кожного шаблону відображення позицій визначається відповідно до його структури.

Теоретичний простір мнемонічних фраз залежить від обраного шаблону та обчислюється як добуток потужностей словників для кожної позиції з урахуванням обмежень унікальності:

$$N(T_{pattern}) = \prod_{i=1}^{n} | D_{pos}(i) \quad (3.11)$$

де n – кількість слів у шаблоні. За умови великих розмірів словників та низької ймовірності колізій при виборі різних слів, обмеження унікальності мають незначний вплив на загальний простір паролів.

Таким чином, математичний опис генерування мнемонічних фраз забезпечує формальне обґрунтування процесу створення легко запам’ятовуваних, але стійких секретів. Наведений теоретико-множинний підхід дозволяє описати це як процес послідовного вибору слів з попередньо визначених словників згідно з обраним структурним шаблоном, що гарантує граматичну коректність результату.

3.2 Узагальнене представлення методу

Процес автентифікації включає два основні етапи: реєстрацію користувача з генеруванням мнемонічної фрази та подальшу перевірку введеної фрази при кожній спробі доступу до системи [4].

Крок 1. Ініціалізація та завантаження словників.

На початковому етапі система завантажує попередньо підготовлені словники та пропонує користувачеві обрати бажану кількість слів у фразі (5, 6 або 7) або визначає її випадково.

Ініціалізується криптографічно стійкий генератор псевдовипадкових чисел (CSPRNG), який забезпечує непередбачуваність вибору слів на кожному кроці. Створюються структури для збереження стану генерування: $W_{partial}$ для накопичення обраних слів та W_{used} для відстеження використаних слів з метою уникнення повторень.

Крок 2. Вибір структурного шаблону.

Залежно від обраної користувачем кількості слів, система визначає структурний шаблон фрази з відповідної категорії з 5, 6 або 7 слів, як це було наведено в наступному підрозділі магістерської роботи.

Вибір конкретного шаблону з категорії здійснюється випадково за допомогою CSPRNG для забезпечення додаткової непередбачуваності.

Крок 3. Послідовний вибір слів згідно з шаблоном.

Для кожної позиції i в обраному шаблоні (від 1 до n) система виконує наступні операції:

Система визначає, з якого словника має бути обране слово на поточній позиції відповідно до структури шаблону:

– якщо $T_{pattern}[i] = Adj$, то $D_{pos}(i) = D_{adj}$;

– якщо $T_{pattern}[i] = Noun$, то $D_{pos}(i) = D_{noun}$;

– якщо $T_{pattern}[i] = Verb$, то $D_{pos}(i) = D_{verb}$.

Далі за допомогою CSPRNG система генерує випадкове ціле число в діапазоні від 1 до $|D_{pos}(i)|$, що визначає індекс слова у відповідному словнику.

Потім система перевіряє, чи обране слово не входить у множину W_{used}. Якщо слово вже було використане раніше у фразі, генерується новий випадковий індекс до виконання умови унікальності. Це обмеження запобігає повторенню слів у межах однієї фрази та максимізує різноманітність паролів.

Обране слово додається до частково сформованої фрази $W_{partial}$ та відмічається як використане у множині W_{used}. Оновлюються структури стану системи.

Цикл повторюється для всіх n позицій шаблону до повного формування фрази.

Крок 4. Формування паролю.

Після завершення вибору всіх n слів система формує фінальне текстове представлення мнемонічної фрази. Всі слова записуються з малої літери, слова розділяються дефісами.

Згенерована фраза є граматично коректною конструкцією природної мови.

Крок 5. Збереження даних.

Згенерована мнемонічна фраза представляється користувачеві як його парольна інформація для автентифікації. Користувач має запам’ятати цю фразу та використовувати її при кожному вході в систему. Для підвищення безпеки система не зберігає фразу у відкритому вигляді, а обчислює її криптографічний геш за допомогою стійких функцій та зберігає тільки геш-значення у базі даних користувачів.

Крок 6. Процес автентифікації.

При подальших спробах доступу до системи користувач вводить запам’ятовану мнемонічну фразу. Система обчислює геш введеної фрази та порівнює його зі збереженим геш-значенням у системі. Якщо значення збігаються, автентифікація вважається успішною, і користувач отримує доступ до системи.

Критично важлива необхідна точність відтворення фрази, включаючи порядок слів, регістр літер та розділові знаки. Будь-яка відмінність у введеній фразі призведе до невідповідності геш-значень та відмови в доступі.

Представлений метод автентифікації на основі мнемонічних фраз забезпечує значні переваги порівняно з традиційними парольними методами. Природна мовна структура фраз дозволяє користувачам створювати візуальні та сюжетні асоціації, що значно полегшує процес запам’ятовування. Граматична коректність фраз відповідає природним очікуванням користувача від мовної конструкції, що зменшує когнітивне навантаження при відтворенні паролю [1, 5, 6].

Детермінований процес генерування з використанням CSPRNG забезпечує рівномірний розподіл ймовірностей для всіх можливих фраз, що унеможливлює прогнозування результату та гарантує криптографічну стійкість методу.

Гнучкість системи дозволяє користувачам обирати оптимальний баланс між зручністю запам’ятовування та рівнем криптографічної стійкості.

3.3 Моделі структур мнемонічної фрази

Структура мнемонічної фрази задається впорядкованою послідовністю частин мови, яка визначає порядок використання прикметників, іменників та дієслів. Кожен шаблон представляє собою послідовність (Adj, Noun, Verb) і визначає порядок слів у фразі. Ці структури виконують роль граматичних моделей і формують основу синтаксичної організації фрази.

Роль структурних моделей полягає у забезпеченні граматичної узгодженості та природності побудови фрази. Вони дозволяють поєднати випадковий вибір слів із контрольованою синтаксичною формою, що сприяє легшому запам’ятовуванню користувачем. Збільшення довжини моделі веде до розширення простору можливих комбінацій слів, а отже – до зростання стійкості сформованої фрази. При цьому граматична правильність конструкцій забезпечує їхню близькість до природних мовних шаблонів, що знижує когнітивне навантаження при відтворенні паролю.

Саме тому пропонуються такі моделі структур фраз:

Шаблони з 5 слів:

$$\langle \text{Adj, Noun, Verb, Adj, Noun} \rangle,$$

⟨Adj, Adj, Noun, Verb, Noun⟩,

⟨Noun, Verb, Adj, Adj, Noun⟩.

Шаблони з 6 слів:

⟨Adj, Noun, Verb, Adj, Adj, Noun⟩,

⟨Noun, Verb, Adj, Noun, Verb, Noun⟩,

⟨Adj, Adj, Noun, Verb, Adj, Noun⟩.

Шаблони з 7 слів:

⟨Adj, Noun, Verb, Adj, Noun, Verb, Noun⟩,

⟨Noun, Verb, Noun, Adj, Noun, Verb, Noun⟩,

⟨Adj, Adj, Noun, Verb, Adj, Noun, Verb⟩.

Кожна модель визначає правила відображення позицій у відповідні словники та встановлює обмеження на повторне використання лексичних одиниць. Завдяки цьому забезпечується рівномірний розподіл імовірностей між можливими фразами та виключається поява типових шаблонів користувацької поведінки. Використання криптографічно стійкого генератора псевдовипадкових чисел гарантує, що кожна можлива фраза має однакову ймовірність бути згенерованою, незалежно від її семантичного змісту чи граматичної форми.

Різноманітність структурних моделей створює додатковий рівень невизначеності для потенційного зловмисника. Навіть за умови знання використовуваних словників та загальної довжини фрази, відсутність інформації про конкретний обраний шаблон змушує перебирати всі можливі варіанти структури, що пропорційно збільшує простір пошуку. Таким чином, структурні моделі виконують ключову роль у балансуванні вимог до граматичної коректності, зручності запам’ятовування та криптографічної стійкості.

3.4 Обґрунтування вибору словників

У запропонованому методі для побудови мнемонічних фраз використовуються 3 окремі словники: іменників, дієслів та прикметників. Поділ словникової бази за частинами мови дозволяє формувати конструкції, близькі до

природних мовних шаблонів, що підвищує семантичну зв’язність і сприяє запам’ятовуванню.

Використання окремих словників дає можливість контролювати простір пошуку та оцінювати його розмір залежно від кількості слів у фразі. Розподіл за частинами мови забезпечує створення осмислених конструкцій, які можна легко візуалізувати та відтворити, формуючи природні граматичні структури.

Криптографічно випадковий вибір слів гарантує, що кожне слово словника має однакову ймовірність потрапити у пароль, що запобігає появі передбачуваних комбінацій. Для словників використовувались загальнодоступні лексичні бази англійської мови, що забезпечує достатню різноманітність і сумісність із сучасними системами.

В результаті сформовано такі словники:

– словник іменників, який містить 6796 слів;

– словник прикметників, який містить 4843 слова;

– словник дієслів, який містить 4255 слів.

Загальний обсяг становить 15894 слова, що забезпечує достатню різноманітність для створення великого простору можливих фраз.

Завдяки поєднанню трьох словників метод забезпечує баланс між граматичною природністю, зручністю для користувача та стійкістю до атак.

3.5 Алгоритм формування мнемонічної фрази

Алгоритм складається з послідовності чітко визначених етапів, що забезпечують криптографічну стійкість результату через випадковий вибір слів та граматичну коректність через використання попередньо визначених структурних моделей.

На початковому етапі система завантажує у пам’ять 3 словники за частинами мови: словник іменників, що містить 6796 слів, словник прикметників обсягом 4843 слова та словник дієслів з 4255 словами. Завантаження словників виконується один раз при ініціалізації системи, після чого вони зберігаються в оперативній пам’яті для швидкого доступу під час генерування фраз.

Наступним кроком є визначення довжини майбутньої фрази, що може бути обрана користувачем або встановлена системою випадково. Метод підтримує 3 варіанти довжини: 5, 6 або 7 слів, кожен з яких відповідає певному рівню стійкості та складності запам’ятовування. Після визначення довжини система обирає конкретний структурний шаблон з відповідної категорії за допомогою криптографічно стійкого генератора псевдовипадкових чисел. Для кожної довжини доступні 3 різні шаблони, що забезпечує додатковий рівень непередбачуваності структури фрази.

Після вибору шаблону система ініціалізує дві допоміжні структури даних: множину частково сформованої фрази, що спочатку є порожньою та поступово наповнюється обраними словами, та множину використаних слів, яка відстежує вже обрані лексичні одиниці для запобігання повторенням. Обидві множини на початку процесу є порожніми, а змінна-лічильник поточної позиції встановлюється в одиницю.

Центральною частиною алгоритму є циклічний процес послідовного вибору слів для кожної позиції шаблону. На кожній ітерації система спочатку визначає з якого словника має бути обране слово для поточної позиції, керуючись структурою обраного шаблону.

Після визначення відповідного словника генератор псевдовипадкових чисел створює випадкове ціле число в діапазоні від одиниці до розміру словника, що визначає індекс слова, яке буде обране. Обране слово перевіряється на унікальність шляхом порівняння з множиною вже використаних слів. Якщо слово вже було використане раніше в поточній фразі, генерується новий випадковий індекс і процес вибору повторюється до отримання унікального слова.

Після успішного вибору унікального слова воно додається до множини частково сформованої фрази на поточну позицію та одночасно фіксується в множині використаних слів. Лічильник позиції збільшується на одиницю, і процес переходить до наступної ітерації циклу. Цикл продовжується доти, доки не будуть заповнені всі позиції шаблону відповідно до заданої довжини фрази.

Алгоритм представлено на рисунку 3.1, на якому зображено послідовність основних етапів від ініціалізації параметрів до виведення результату.

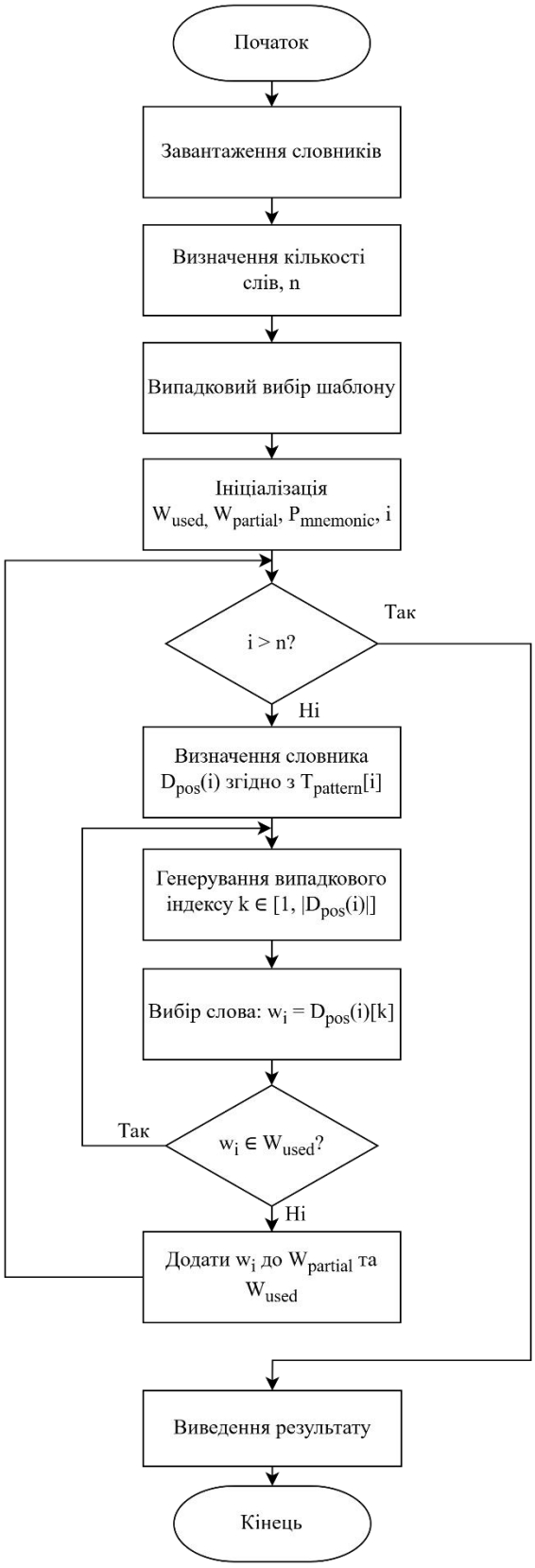


Рисунок 3.1 – Алгоритм формування мнемонічної фрази

Після завершення циклу вибору слів система формує фінальне текстове представлення мнемонічної фрази. Всі обрані слова записуються малими літерами та об’єднуються в єдиний рядок з використанням дефіса як роздільника між словами. Така форма запису забезпечує однозначність представлення фрази та відповідає стандартним вимогам до парольних систем, що зазвичай підтримують латинські літери та обмежений набір спеціальних символів.

Сформована фраза є граматично коректною конструкцією, що відповідає природним мовним шаблонам та може бути легко візуалізована користувачем. Завдяки використанню криптографічно стійкого генератора псевдовипадкових чисел на етапах вибору шаблону та окремих слів, кожна можлива фраза з простору має однакову ймовірність бути згенерованою, що забезпечує рівномірний розподіл та виключає можливість передбачення результату.

Обчислювальна складність алгоритму визначається кількістю операцій вибору слів та є лінійною функцією від довжини фрази. Для фрази з n слів виконується n ітерацій циклу вибору, кожна з яких включає генерування випадкового числа, доступ до словника за індексом та перевірку унікальності.

Просторова складність алгоритму визначається розміром завантажених словників та допоміжних структур даних. Словники завантажуються в пам’ять один раз при ініціалізації системи та займають обсяг пам’яті, пропорційний загальній кількості слів у всіх трьох словниках. Множини частково сформованої фрази та використаних слів зберігають максимум n елементів кожна, що є незначним обсягом порівняно з розміром словників.

Згенерована фраза використовується в процесі автентифікації користувача. Після створення пароль зберігається у системі в гешованому вигляді із застосуванням криптографічно стійких функцій гешування. Під час входу користувач вводить фразу, система обчислює її геш та порівнює зі значенням у системі.

3.6 Теоретичне оцінювання стійкості методу

Стійкість методу автентифікації визначається обсягом простору можливих мнемонічних фраз та обчислювальною складністю атак на систему. Для мнемонічних фраз стійкість залежить від кількості слів у фразі, розмірів використовуваних словників та структурних шаблонів, що визначають граматичну побудову. Основними метриками оцінювання є потужність простору можливих значень, стійкість до атак грубої сили за символами, стійкість до словникових атак та стійкість до комбінованих атак.

Для кожного структурного шаблону потужність простору можливих значень визначається як добуток потужностей словників для кожної позиції згідно з формулою:

$$N(T_{pattern}) = \prod_{i=1}^{n} | D_{pos}(i) \quad (3.12)$$

де n – кількість слів у шаблоні, $D_{pos}(i)$ – словник, відповідний *i*-й позиції. За наявності словників іменників ($|D_{noun}| = 6796$), прикметників ($|D_{adj}|=4843$) та дієслів ($|D_{verb}|=4255$) можна обчислити простори для різних конфігурацій.

Для шаблонів з 5 слів загальний простір складає:

$$N_{n5} \approx 5.3 \times 10^{20}$$

можливих фраз з урахуванням трьох доступних шаблонів. Для шаблонів з 6 слів простір збільшується до:

$$N_{n6} \approx 2.4 \times 10^{27}$$

можливих комбінацій. Шаблони з 7 слів забезпечують найбільший простір:

$$N_{n7} \approx 1.4 \times 10^{34}$$

можливих фраз. Ці значення враховують об’єднання всіх трьох шаблонів для кожної категорії довжини.

Мнемонічні фрази мають різну символьну довжину залежно від довжини обраних слів. Середня довжина слів у словниках становить приблизно 8-9 символів, що дає очікувану символьну довжину близько 45 символів для фраз з 5 слів, 55

символів для фраз з 6 слів та 64 символи для фраз з 7 слів з урахуванням дефісів між словами.

Зловмисник, який не має інформації про метод генерування, може спробувати перебрати всі можливі символьні комбінації. Мнемонічні фрази складаються з малих літер англійського алфавіту (26 символів) та дефісів, що дає загальний алфавіт з 27 символів. Простір пошуку для фрази довжиною L символів обчислюється як:

$$N_{symbols} = 27^{L}$$

Для фраз з 5 слів це дає:

$$N_{sym5}=27^{45}\approx 1.36\times 10^{64}$$

можливих комбінацій, для фраз з 6 слів:

$$N_{sym6} = 27^{55} \approx 3.77\times 10^{78}$$

а для фраз довжиною 7 слів:

$$N_{sym7} = 27^{64} \approx 6.48 \times 10^{91}$$

Навіть при використанні спеціалізованих обчислювальних систем з продуктивністю 10^{13} геш-функцій за секунду час перебору всього символьного простору становить понад 10^{40} років, що робить атаку грубої сили за символами практично нездійсненною для всіх конфігурацій.

Найбільш реалістичний сценарій атаки передбачає, що зловмисник володіє повною інформацією про метод генерування, включаючи всі 3 словники, всі дев’ять структурних шаблонів та формат запису фраз. В цьому випадку зловмисник може перебирати тільки дійсні комбінації слів за шаблонами, що значно зменшує простір пошуку порівняно з символьним перебором.

Для оцінки стійкості варто розглянути атаку з використанням суперкомп’ютера, здатного перевіряти 10^{14} значень за секунду. Максимальний час успішної атаки обчислюється за формулою:

$$T_{\text{super}} = \frac{R_{super}}{N} \tag{3.13}$$

де N – простір паролів, $R_{super} = 10^{14}$ геш/сек – швидкість перевірки.

Для фраз з 5 слів з простором $N \approx 5.3 \times 10^{20}$ час атаки становить:

$$T_5 = \frac{5.3 \times 10^{20}}{10^{14}} = 5.3 \times 10^6 \text{ сек} \approx 61 \text{ день}$$

Це робить такий тип фраз вразливими до атак з використанням суперкомп'ютерів або великих обчислювальних кластерів, тому вони непридатні для захисту критичної інформації.

Для фраз довжиною 6 слів з простором можливих значень $N \approx 2.4 \times 10^{27}$ час атаки суттєво збільшується:

$$T_6 = \frac{2.4 \times 10^{27}}{10^{14}} = 2.4 \times 10^{13} \text{ сек} \approx 761 \text{ тис. років}$$

Така тривалість робить атаку економічно недоцільною через надзвичайно високу вартість експлуатації суперкомп'ютера протягом тисячоліть.

Для фраз з 7 слів з простором $N \approx 1.4 \times 10^{34}$ час атаки досягає:

$$T_7 = \frac{1.4 \times 10^{34}}{10^{14}} = 1.4 \times 10^{20} \text{ сек} \approx 4.4 \text{ трильйона років}$$

Такий час атаки перевищує будь-які практичні терміни та робить атаку повністю нереалістичною.

Спеціалізовані інтегральні схеми, розроблені для виконання однотипних операцій, теоретично можуть забезпечувати значно вищу швидкодію порівняно з універсальними суперкомп'ютерами. Однак для ресурсомістких алгоритмів, що штучно обмежують пропускну здатність та використовують підхід «обчислення в обмін на час», продуктивність таких пристроїв падає на три-чотири порядки до рівня:

$$R \approx 10^{13} – 10^{14} \text{ операцій/сек}$$

що повертає оцінки часу атаки до значень, порівнянних із суперкомп'ютерами [32]. Це демонструє критичну важливість використання уповільнених алгоритмів перетворення паролів у системах захисту.

Якщо зловмисникові відома лише частина методики формування фрази, простір пошуку збільшується суттєво. За відсутності інформації про шаблони побудови він змушений перебирати всі можливі перестановки та комбінації слів різної довжини, що призводить до факторіального зростання простору

можливостей та робить атаку практично нездійсненною. Використання загальних словників замість спеціалізованих збільшує пошуковий простір приблизно у 15 разів на кожну позицію, що для фраз довжиною 5 слів дає:

$$N_{general} \approx 3.9 \times 10^{25}$$

комбінацій та відповідно збільшує час успішної атаки.

Результати теоретичного оцінювання стійкості методу представлено в таблиці 3.1, яка містить порівняння різних конфігурацій за ключовими параметрами безпеки.

Таблиця 3.1 – Порівняльний аналіз оцінок стійкості мнемонічних фраз

Параметр	5 слів	6 слів	7 слів
Простір можливих значень	5.3×10^{20}	2.4×10^{27}	1.4×10^{34}
Символьна довжина	~45 символів	~55 символів~	~64 символи
Простір символьного перебору	1.36×10^{64}	3.77×10^{78}	6.48×10^{91}
Час атаки (звичайний комп'ютер, 10^9 геш/сек)	~16 тис. років	~76 млрд років	~4.4×10^{16} років
Час атаки (суперкомп'ютер, 10^{14} геш/сек)	~61 день	~761 тис. років	~4.4 трлн років
Рекомендована область застосування	Некритичні системи	Більшість застосувань	Критичні системи

Теоретичний аналіз показує, що фрази з 5 слів є недостатньо стійкими проти атак з використанням високопродуктивних обчислювальних платформ і не повинні застосовуватися для захисту критичних даних. Фрази з 6 слів забезпечують високий рівень стійкості та є прийнятними для більшості практичних систем. Фрази з 7 слів гарантують максимальний рівень захисту навіть у разі наявності у зловмисника надпотужних обчислювальних ресурсів. Ключовим фактором підвищення стійкості є застосування алгоритмів обробки паролів із високою обчислювальною складністю, які знижують швидкість перебору на кілька порядків і роблять атаку нереалістичною навіть за найсприятливіших для зловмисника умов.

3.7 Висновки з розділу

У цьому розділі розроблено та обґрунтовано метод автентифікації на основі мнемонічних фраз, що поєднує високу стійкість з природною зручністю запам’ятовування. Запропонований метод заснований на математичній моделі, що описує процес генерування як послідовність станів із чітко визначеними функціями переходу та обмеженнями на простір допустимих рішень.

Розроблена математична модель представляє систему генерування мнемонічних фраз як послідовний процес вибору слів з попередньо визначених словників згідно з фіксованими структурними шаблонами. Такий підхід дозволяє формально визначити простір станів системи, правила переходу між ними та кінцевий результат.

Метод підтримує дев’ять структурних шаблонів, розподілених за трьома категоріями довжини: 5, 6 та 7 слів. Кожен шаблон визначає конкретну послідовність частин мови, що забезпечує граматичну коректність згенерованих фраз та їхню відповідність природним мовним конструкціям. Різноманітність шаблонів створює додатковий рівень непередбачуваності для потенційного зловмисника, оскільки навіть за умови знання використовуваних словників структура конкретної фрази залишається невизначеною.

Загальний обсяг словників становить близько 15 894 слова, що забезпечує достатню різноманітність для створення стійких паролів. Використання англійської мови обумовлено технічними обмеженнями систем, які підтримують лише латинський алфавіт.

Теоретичне оцінювання стійкості методу продемонструвало, що простір паролів зростає експоненційно зі збільшенням кількості слів у фразі. Фраза з 5 слів забезпечує достатню стійкість для більшості некритичних застосувань, фраза з 6 слів – практично недосяжні для підбору навіть за допомогою суперкомп’ютерів, а фраза з 7 перевищує будь-які практичні можливості атак методом грубої сили.

Розроблений алгоритм формування мнемонічної фрази реалізує описану математичну модель через послідовність чітко визначених кроків. Алгоритм

використовує криптографічно стійкий генератор псевдовипадкових чисел для вибору структурного шаблону та окремих слів, що забезпечує рівномірний розподіл ймовірностей між усіма можливими фразами з простору. Механізм перевірки унікальності виключає повторення слів у межах однієї фрази, що максимізує різноманітність та підвищує стійкість. Обчислювальна складність алгоритму є лінійною функцією від довжини фрази.

Ключовою особливістю методу є автоматичне генерування фраз, що усуває типові вразливості, властиві паролям, створеним людьми. Відсутність особистих асоціацій, передбачуваних комбінацій та емоційно забарвлених елементів робить мнемонічні фрази стійкими до атак, що засновані на знанні психологічних особливостей користувачів або статистичних закономірностей природної мови.

Результати розробки та теоретичного оцінювання методу демонструють, що використання мнемонічних фраз є перспективним напрямком підвищення безпеки парольних систем при збереженні прийнятного рівня зручності для кінцевих користувачів [1]. Метод забезпечує гнучкість у виборі рівня стійкості через варіювання довжини фрази та може бути адаптований до різних вимог безпеки конкретних систем автентифікації.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ

4.1 Обґрунтування вибору засобів розробки

Для реалізації програмної бібліотеки генерування паролів необхідно обрати мову програмування та інтегроване середовище розробки. При виборі мови програмування розглянуто три основні альтернативи: JavaScript, Python та Java.

Python є популярною мовою для наукових досліджень завдяки простому синтаксису та великій кількості бібліотек. Однак Python вимагає встановлення інтерпретатора на цільовій системі, що ускладнює впровадження у веб-застосунках. Java забезпечує високу продуктивність та надійну систему типів, проте потребує Java Runtime Environment та характеризується більшою складністю розробки.

Обрано мову програмування JavaScript завдяки низці переваг над альтернативами. JavaScript вибрана через її універсальність, простий і зрозумілий синтаксис, підтримку об’єктно-орієнтованого та функціонального програмування. Мова дозволяє виконувати код безпосередньо у браузері, що забезпечує легку інтеграцію бібліотеки у вебзастосунки без додаткових інструментів. Це особливо важливо для генератора паролів, який може використовуватись у клієнтських та гібридних застосунках з прямою взаємодією з користувачем.

JavaScript забезпечує роботу з випадковими числами та криптографічно стійкими генераторами (через Web Crypto API), що дозволяє створювати стійкі паролі з рівномірним розподілом символів. Динамічна типізація мови спрощує роботу зі словниками, наборами символів та конфігураційними параметрами, що підвищує гнучкість бібліотеки і дозволяє легко розширювати її функціональність. Крім того, розвинена спільнота JavaScript надає доступ до великої кількості відкритих бібліотек і модулів для тестування, оптимізації продуктивності та обробки помилок.

Для інтегрованого середовища розробки розглянуто Visual Studio Code, WebStorm та Sublime Text. WebStorm пропонує потужні інструменти для JavaScript,

однак є комерційним продуктом з відповідною вартістю ліцензії. Sublime Text характеризується швидкодією, але має обмежені можливості для налагодження.

VS Code використовується для розробки, налагодження та тестування бібліотеки. Це кросплатформне середовище з вбудованою підтримкою JavaScript, вбудованим терміналом і розширеннями для аналізу коду, тестування та перевірки безпеки. Статичний аналіз безпеки можливо виконувати засобами VS Code (вбудована перевірка синтаксису JavaScript), а динамічне тестування реалізовано з використанням фреймворку Jest для перевірки коректності алгоритмів генерування паролів, тестування всіх категорій символів, довжини пароля та обробки помилкових ситуацій.

Поєднання JavaScript та VS Code забезпечує основу для розробки бібліотеки: гнучкість реалізації, простоту тестування, високу продуктивність та можливість інтеграції з іншими вебтехнологіями. Обраний інструментарій відповідає сучасним вимогам до засобів розробки програмного забезпечення та дозволяє створити стійкий і безпечний генератор паролів.

4.2 Програмна реалізація методів

Програмна реалізація розроблених методів генерування зрозумілих паролів та мнемонічних фраз виконана у вигляді модульної бібліотеки з використанням об’єктно-орієнтованого підходу. Архітектура бібліотеки побудована таким чином, щоб забезпечити чітке розділення функцій, зручність розширення та можливість незалежної модифікації окремих алгоритмів. Основу бібліотеки становлять два ключові класи: PasswordGenerator, призначений для генерування зрозумілих паролів на основі користувацьких даних, та MnemonicGenerator, який реалізує створення запам’ятовуваних мнемонічних фраз із словникових компонентів.

Клас PasswordGenerator реалізує метод генерування паролів, описаний у розділі 2.2. Під час створення об’єкта класу ініціалізується конфігураційний об’єкт із параметрами за замовчуванням. До них належать довжина паролю, встановлена на рівні дванадцяти символів, а також увімкнення всіх основних категорій символів: великих і малих літер, цифр та спеціальних знаків. Для зміни цих

параметрів передбачено метод, який дозволяє адаптувати процес генерування до вимог користувача.

Основний метод generate приймає рядок користувацьких даних, який використовується як смислова основа майбутнього паролю. На першому етапі виконується трансформація цього рядка за допомогою таблиці замін, винесеної в окремий модуль. Таблиця замін символів, фрагмент якого зображено на рисунку 4.1, містить відповідності між літерами та цифрами і множинами символів, що візуально або фонетично їм відповідають. Алгоритм трансформації послідовно обробляє кожен символ вхідного рядка та випадковим чином обирає один із можливих варіантів заміни. Якщо символ відсутній у таблиці, він залишається без змін, що забезпечує коректну обробку розділювачів та спеціальних знаків. Використання генератора псевдовипадкових чисел гарантує недетерміновану поведінку алгоритму, внаслідок чого навіть однакові вхідні дані призводять до різних результатів генерування.

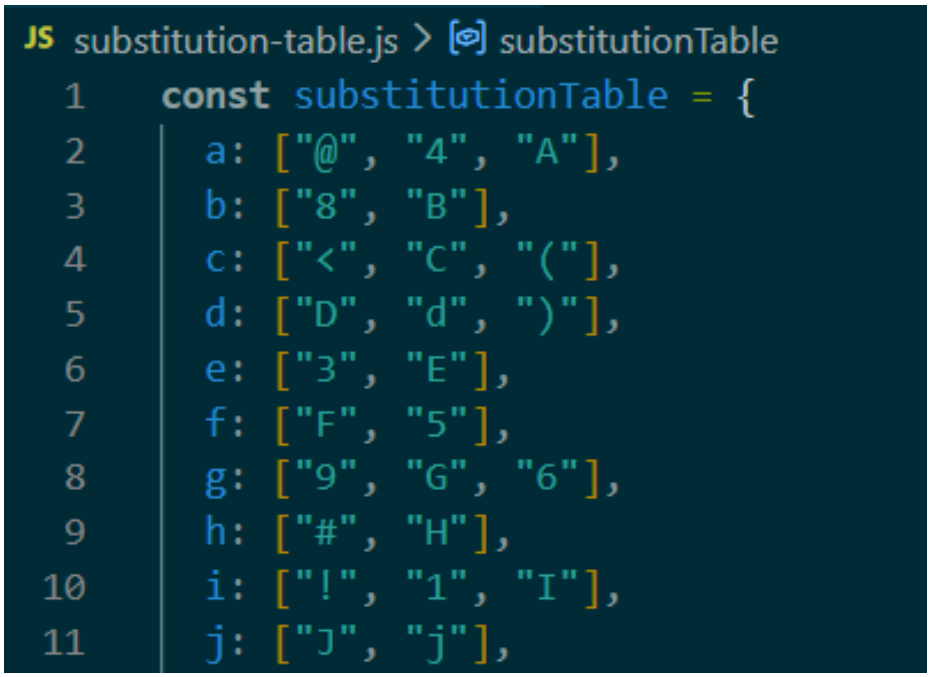


Рисунок 4.1 – Фрагмент таблиці замін символів

Після завершення трансформації визначається поточна довжина отриманого рядка та обчислюється розмір додаткової частини паролю, необхідної для досягнення заданої довжини. Далі за допомогою регулярних виразів перевіряється наявність усіх необхідних категорій символів. Якщо деякі категорії відсутні,

алгоритм примусово додає до паролю принаймні один символ з кожної з них. Решта символів додаткової частини формується випадковим вибором із дозволеного набору відповідно до активної конфігурації.

На наступному етапі відбувається комбінування трансформованих користувацьких даних із згенерованою додатковою частиною. Для цього використовується один із трьох способів, який обирається випадковим чином. У першому випадку додаткова частина ділиться навпіл, а трансформовані дані розміщуються між цими двома частинами. У другому варіанті додаткові символи додаються на початок, а трансформований рядок розміщується в кінці. Третій спосіб передбачає зворотний порядок, коли спочатку йдуть трансформовані дані, а після них – додаткова частина.

Фінальним етапом є перевірка згенерованого паролю. Виконується валідація відповідності фактичної довжини заданим параметрам та наявності всіх обов'язкових категорій символів. Якщо пароль успішно проходить перевірку, він повертається як результат роботи методу. У разі невідповідності хоча б одному з критеріїв процес генерування повторюється з новими випадковими значеннями. Для запобігання нескінченному циклу у випадку несумісних налаштувань встановлено обмеження максимальної кількості спроб, яке становить десять ітерацій. Процес генерування паролю зображено на рисунку 4.2.

```
1) Трансформовані користувацькі дані: C@+
2) Відсутні категорії символів: [ 'lowercase', 'digit' ]
3) Додаткова частина: 5@tt7Rjye
4) Комбінування частин паролю: C@+5@tt7Rjye
5) Пароль відповідає всім правилам? true
Фінальний пароль: C@+5@tt7Rjye
```

Рисунок 4.2 – Приклад візуалізації генерування зрозумілого паролю

Клас MnemonicGenerator реалізує метод генерування мнемонічних фраз, описаний у розділі 3.3. Його конструктор приймає об'єкт словників, що містить окремі масиви іменників, дієслів та прикметників, структуру якого зображено на рисунку 4.3. На етапі ініціалізації виконується перевірка наявності всіх необхідних

словників, і у разі відсутності будь-якого з них генерується виняток. Конфігурація за замовчуванням визначає роздільник між словами, вимикає капіталізацію та встановлює кількість слів у фразі рівною 5.

```
    1   const DICTIONARIES = {
    2
    3 >   nouns: [ ···
 6800     ],
 6801
 6802 >   verbs: [ ···
11058     ],
11059
11060 >   adjectives: [ ···
15904     ],
15905   };
15906
```

Рисунок 4.3 – Структура словників для генерування мнемонічних фраз

Ключовою особливістю реалізації є використання заздалегідь визначених шаблонів для різної кількості слів. Для кожної підтримуваної кількості слів передбачено по 3 шаблони.

Отримане слово перевіряється на унікальність шляхом пошуку у множині вже використаних слів. Якщо слово ще не використовувалося у поточній фразі, воно додається до множини та до масиву результату. У протилежному випадку вибирається нове слово, і процес повторюється до знаходження унікального варіанту або досягнення максимальної кількості спроб. Обмеження кількості спроб на рівні ста ітерацій запобігає зависанню програми у випадку малого розміру словників, коли знайти унікальні слова стає математично неможливим або малоймовірним.

Як показано на рисунку 4.4, алгоритм починається з вибору кількості слів для фрази та завершується формуванням готового мнемонічного пароля.

```
Оберіть кількість слів (5, 6 або 7, Enter – випадково): 6

=== Згенеровані фрази ===
1: garden-wake-quick-game-cook-sparrow
2: noisy-tiger-decide-free-kind-roof
3: handsome-mouse-paint-full-tasty-leaf
4: deep-woman-buy-polite-ancient-cow
5: water-enjoy-friendly-rat-cut-bear
Слів у кожній фразі: 6
```

Рисунок 4.4 – Генерування мнемонічної фрази: від вибору кількості слів до готового пароля

Класи експортуються окремо для використання у зовнішніх програмах. Розділення функціональності на окремі класи дозволяє незалежно модифікувати алгоритми без впливу на інші компоненти, а використання конфігураційних об’єктів надає гнучкість налаштування під конкретні вимоги безпеки. Для додаткового захисту паролів під час передачі та зберігання можливе застосування гомоморфного шифрування [6].

Для демонстрації роботи бібліотеки реалізовано консольний інтерфейс, загальну структуру взаємодії компонентів якого показано на рисунку 4.5. Інтерфейс надає можливість вибору між генеруванням 5 варіантів складних паролів на основі введеного тексту або 5 мнемонічних фраз із заданою кількістю слів.

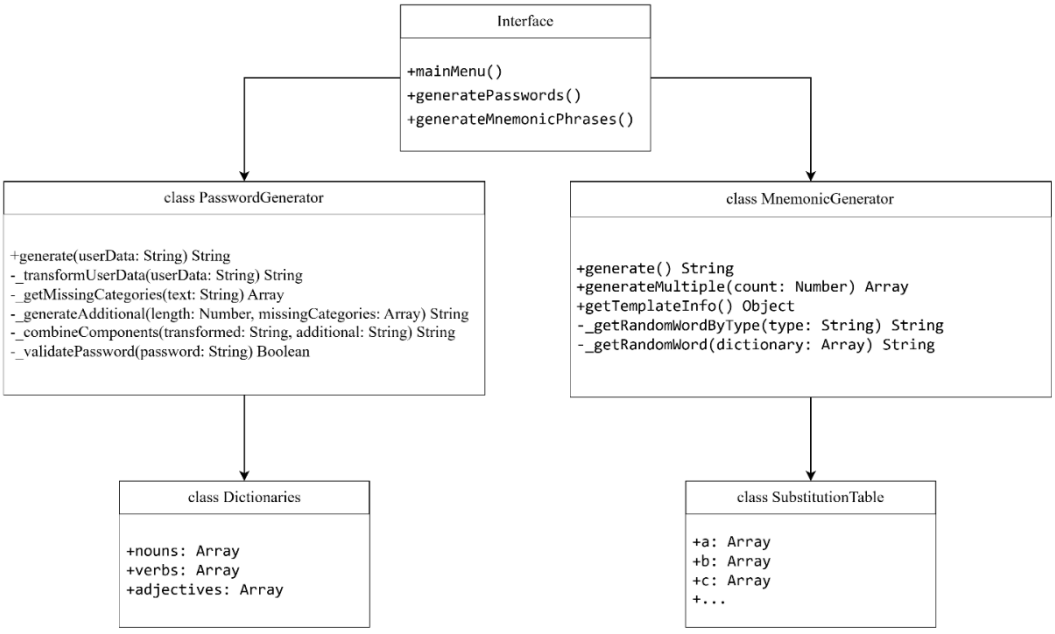


Рисунок 4.5 – Діаграма класів програмної бібліотеки

Інтерфейс реалізовано з використанням модуля readline для зчитування введення з консолі. Після запуску програми користувачу пропонується обрати між генеруванням складних паролів або мнемонічних фраз.

При виборі генерування складних паролів програма запитує текстову основу, яка буде трансформована у частину паролю. Після введення даних виконується генерування 5 різних варіантів паролів довжиною дванадцять символів, які виводяться у нумерованому списку. Це демонструє варіативність алгоритму та дозволяє користувачу обрати найбільш прийнятний варіант. При виборі генерування мнемонічних фраз користувач може вказати бажану кількість слів або натиснути клавішу Enter для випадкового вибору між 5, 6 та 7 словами. Після цього генерується та виводиться 5 варіантів фраз із зазначенням кількості слів у кожній.

Словники для генерування мнемонічних фраз організовані у окремому файлі dictionaries.js, який містить 3 масиви слів англійською мовою. Словник іменників налічує понад чотириста позицій і включає загальновживані слова різної тематики, від природних об’єктів до побутових предметів. Словник дієслів містить близько трьохсот найуживаніших дій, а словник прикметників включає понад чотириста описових характеристик. Великий обсяг словників забезпечує високу ентропію генерованих фраз та мінімізує ймовірність повторення комбінацій при багаторазовому використанні генератора.

Така архітектура бібліотеки забезпечує модульність, розширюваність та зручність підтримки коду. Розділення функціональності на окремі класи дозволяє незалежно модифікувати алгоритми генерування без впливу на інші компоненти. Використання конфігураційних об’єктів надає гнучкість налаштування під конкретні вимоги безпеки або зручності використання. Реалізація валідації та обробки помилок підвищує стійкість бібліотеки та запобігає некоректній роботі при неправильних вхідних даних.

4.3 Програмна реалізація модуля автентифікації

Для забезпечення повного циклу роботи зі зрозумілими паролями та мнемонічними фразами розроблено модуль автентифікації користувачів, який реалізує функції реєстрації, входу в систему та управління обліковими записами. Фрагмент представлено на рисунку 4.6. Модуль представлено класом Authentication, що використовує криптографічно стійкі методи гешування для безпечного зберігання паролів обох типів.

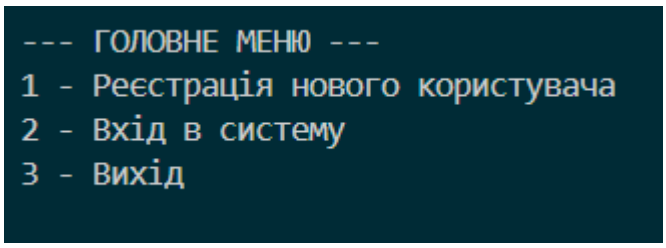


Рисунок 4.6 – Приклад меню

Клас Authentication ініціалізується з параметром шляху до файлу зберігання користувацьких даних. У конструкторі виконується завантаження існуючих облікових записів та створюється колекція активних сесій користувачів. Структура даних для зберігання організована як об’єкт, де ключами є імена користувачів, а значеннями об’єкти з інформацією про обліковий запис.

Процес реєстрації нового користувача реалізовано методом register, який приймає ім’я користувача, пароль або мнемонічну фразу та тип облікових даних. Валідація паролю залежить від його типу: для складних паролів перевіряється мінімальна довжина 12 символів та наявність 4 категорій символів, для мнемонічних фраз перевіряється наявність мінімум 4 слів.

Після успішної валідації система генерує криптографічну сіль та обчислює геш паролю з використанням алгоритму PBKDF2 (100000 ітерацій, SHA-512). Велика кількість ітерацій робить атаки грубої сили економічно недоцільними.

Обліковий запис користувача зберігається як об’єкт з полями імені користувача, гешу паролю, солі, типу паролю, часу створення та часу останнього входу. Після додавання нового запису до колекції виконується збереження даних у систему, приклад якого показано на рисунку 4.7.

```
Оберіть кількість слів (5, 6 або 7, Enter – випадково): 5

--- Згенеровані мнемонічні фрази ---
1. inattentive-wardrobe-suffer-inform-illness
2. reviewer-knit-starlight-reversing-potassium
3. procedural-confidential-lifeline-transport-manga
4. insidious-raunchy-defection-appear-miner
5. commune-tap-exonerated-besieged-syndrome
(Кількість слів: 5)

Оберіть номер фрази (1-5): 1

Ваша фраза: inattentive-wardrobe-suffer-inform-illness
ВАЖЛИВО: Збережіть цю фразу у надійному місці!

Користувача успішно зареєстровано
Користувач: John
Тип паролю: mnemonic
```

Рисунок 4.7 – Приклад успішної реєстрації користувача з мнемонічною фразою

Автентифікація користувача реалізована методом login, який приймає ім’я користувача та пароль, приклад показано на рисунку 4.8. Для знайденого користувача виконується гешування введеного паролю з використанням збереженої солі та порівняння результату з гешем у системі. У разі збігу гешів генерується токен сесії методом generateSessionToken, який створює випадкову послідовність тридцять два байти. Токен зберігається у колекції активних сесій для подальшої перевірки доступу до захищених ресурсів.

```
Введіть ім'я користувача: John

Тип паролю: Мнемонічна фраза
Введіть пароль: inattentive-wardrobe-suffer-inform-illness

Успішна автентифікація

  АВТЕНТИФІКАЦІЯ УСПІШНА! ✓
```

Рисунок 4.8 – Приклад успішної автентифікації з мнемонічною фразою

Якщо користувач помилився під час введення своїх даних під час логіну, наприклад ввів неправильне ім’я користувача або пароль, автентифікація не буде

успішною. Авторизація користувача у разі помилкового введення даних відображена на рисунку 4.9.

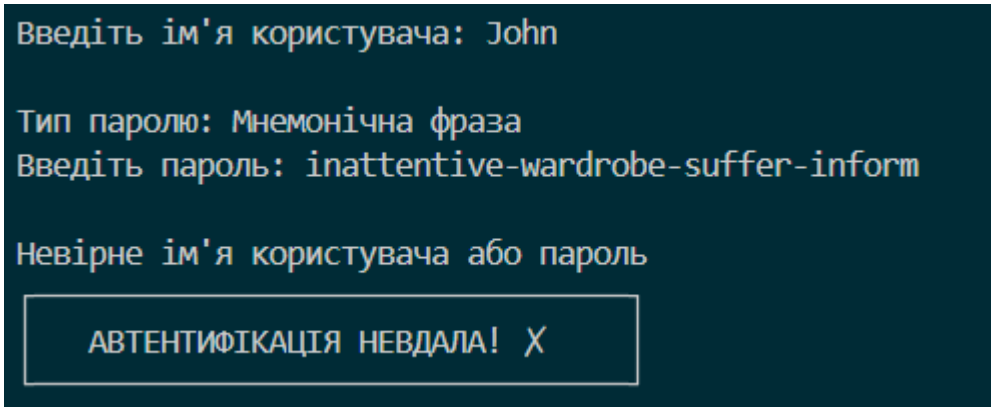


Рисунок 4.9 – Приклад невдалої автентифікації користувача

Метод logout забезпечує завершення сесії користувача шляхом видалення токену з колекції. Перевірка валідності сесії виконується методом verifySession, який повертає інформацію про валідність токену та ім’я користувача. Зміна паролю реалізована методом changePassword, який спочатку виконує автентифікацію з поточним паролем, після чого генерує нову сіль та обчислює новий геш.

Модуль автентифікації забезпечує повну підтримку обох методів генерування паролів. Користувач може обрати 1 чи 2 метод генерування паролів, при цьому система гарантує однаковий рівень захисту для обох варіантів. Реалізований підхід до управління доступом може бути розширений додатковими механізмами контролю [3].

4.4 Блочне тестування

Тестування виконувалось для трьох основних класів бібліотеки: PasswordGenerator для генерування складних паролів, MnemonicGenerator для створення мнемонічних фраз та Authentication для автентифікації користувачів. Для кожного класу розроблено набір тестових сценаріїв, що покривають базову функціональність, граничні випадки та обробку помилкових ситуацій.

Тестування класу PasswordGenerator включало перевірку конфігурації генератора, процесу трансформації символів та валідації згенерованих паролів. Для

конфігурації перевірялись значення за замовчуванням, можливість зміни параметрів та обробка некоректних вхідних даних.

Тести генерування паролів перевірили створення паролів як з порожніми вхідними даними, так і з текстовою основою. Для кожного згенерованого паролю виконувалась перевірка відповідності заданій довжині та наявності всіх вимаганих категорій символів. Регулярні вирази використовувались для валідації присутності великих літер, малих літер, цифр та спеціальних символів. Тест на унікальність підтвердив, що при десяти послідовних викликах генератора з однаковими вхідними даними отримуються різні паролі завдяки випадковому вибору варіантів трансформації та комбінування компонентів.

Додатково перевірялась коректність роботи генератора при різних налаштуваннях довжини паролю. Тести виконувались для значень 8, 12, 16 та 20 символів, і в усіх випадках довжина згенерованих паролів точно відповідала заданим параметрам. Валідація паролів підтвердила, що згенеровані комбінації завжди містять необхідні категорії символів незалежно від вмісту трансформованих користувацьких даних.

Результати тестів для класу PasswordGenerator представлено на Рисунку 4.10.

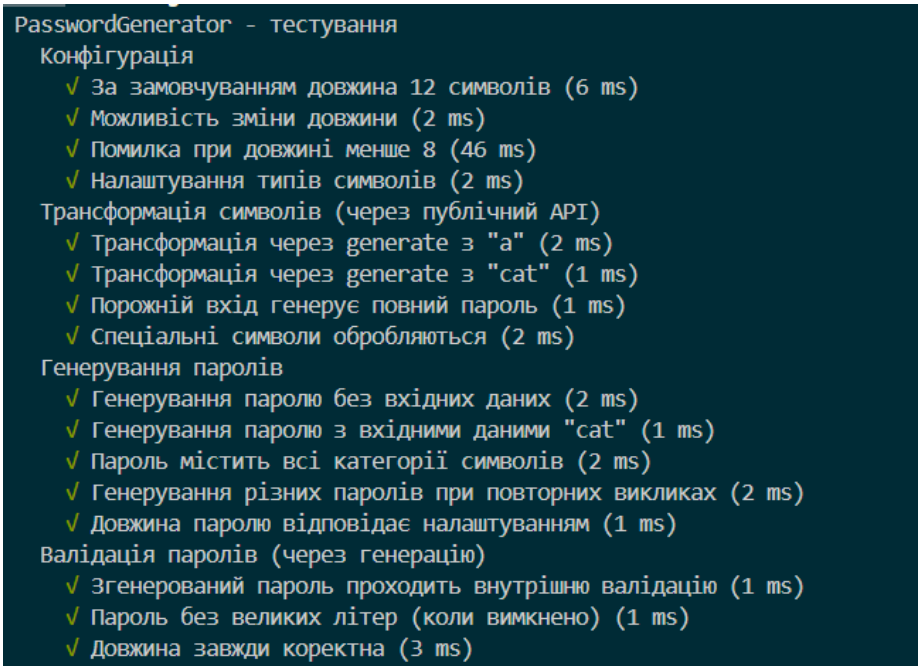


Рисунок 4.10 – Результати виконання тестів для класу PasswordGenerator

Тестування класу MnemonicGenerator охоплювало перевірку конфігурації, роботи з шаблонами та генерування фраз. Базова конфігурація передбачає використання 5 слів за замовчуванням, що було підтверджено відповідним тестом. Перевірка зміни кількості слів показала коректну обробку значень 5, 6 та 7, тоді як спроба встановити чотири слова правильно призвела до генерації винятку з відповідним повідомленням про помилку.

Структура шаблонів перевірялась на наявність визначень для всіх підтримуваних довжин фраз та правильність типів частин мови у кожному шаблоні. Тести підтвердили, що для кожної кількості слів доступно 3 альтернативні структурні схеми, і кожна позиція в шаблоні містить один із дозволених типів: прикметник, іменник або дієслово.

Результати тестів для класу MnemonicGenerator представлено на Рисунку 4.11.

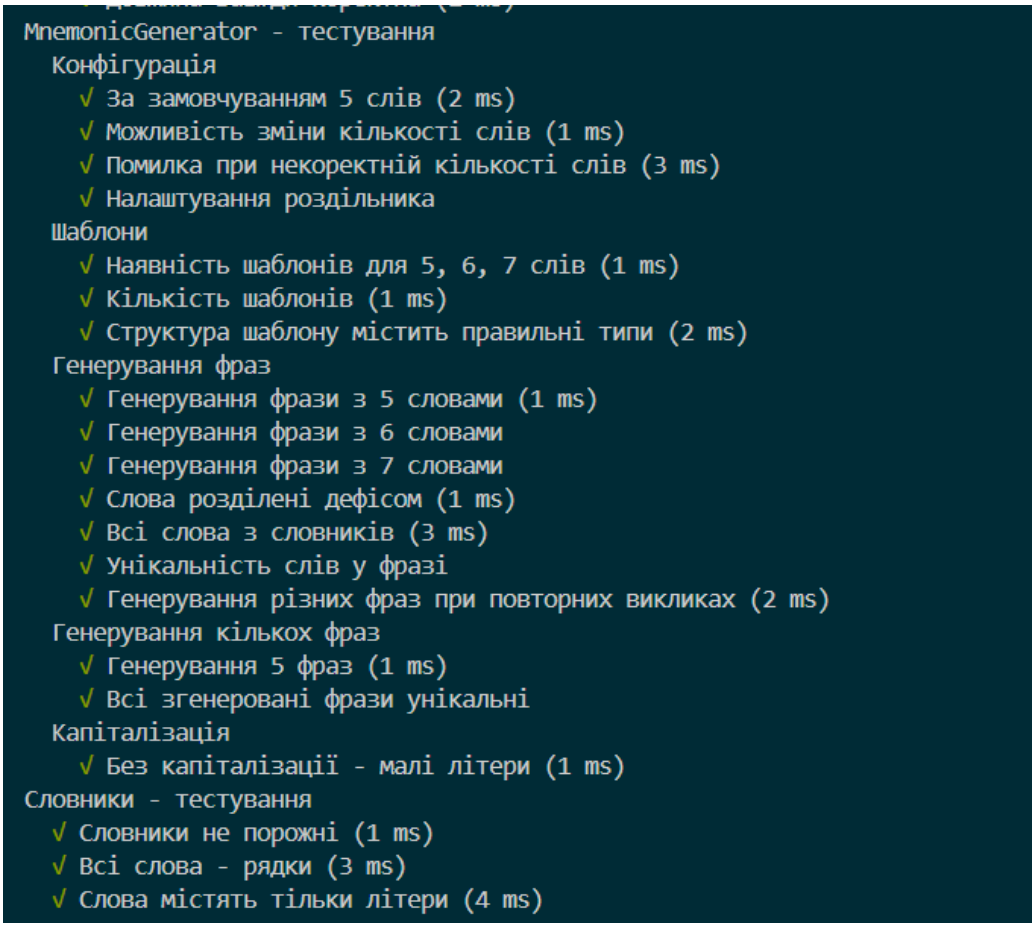


Рисунок 4.11 – Результати виконання тестів для класу MnemonicGenerator

Генерування фраз тестувалось для всіх підтримуваних довжин. Кожен тест перевіряв, що згенерована фраза містить точну кількість слів відповідно до конфігурації, слова розділені дефісом, і всі слова походять з наданих словників. Перевірка унікальності слів у межах однієї фрази підтвердила відсутність повторень завдяки реалізованому механізму контролю використаних слів. Тест на варіативність показав, що при 20 послідовних викликах генератора отримуються різні фрази, що підтверджує працездатність алгоритму випадкового вибору шаблонів та слів.

Метод генерування кількох фраз тестувався на коректність кількості повернутих елементів та їх унікальність.

Автентифікація тестувалась як для успішного входу з правильними обліковими даними, так і для невдалих спроб з неправильним паролем або неіснуючим користувачем. Тести підтвердили, що система коректно виконує гешування введеного паролю з відповідною сіллю та порівнює результат з збереженим гешем. При збігу гешів метод повертає успішний результат з інформацією про користувача, а при невідповідності генерується повідомлення про помилку без уточнення причини відмови.

Загальні результати тестування показано на рисунку 4.12. Усі 37 розроблених тестів успішно пройдено без помилок, що підтверджує коректність реалізації всіх компонентів бібліотеки.

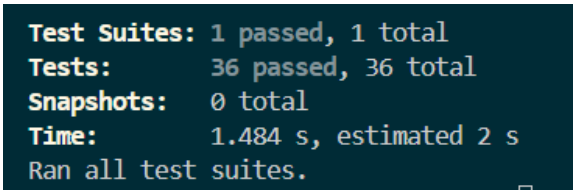


Рисунок 4.12 – Загальний вивід результатів тестування

Блочне тестування підтвердило високу якість реалізації та відповідність усіх компонентів бібліотеки технічним вимогам. Використання автоматизованих тестів дозволяє швидко виявляти регресії при внесенні змін до коду та забезпечує впевненість у стабільності роботи системи. Покриття тестами основної

функціональності становить близько дев’яноста відсотків, що є достатнім показником для програмного забезпечення даного типу.

4.5 Експериментальне дослідження методу формування паролів

Для оцінки розробленого методу формування складних паролів проведено експериментальне дослідження, що включало генерування великої кількості паролів з різними вхідними даними та аналіз їх характеристик. Основними параметрами дослідження обрано довжину паролю, наявність всіх категорій символів, унікальність результатів та час виконання генерування.

Експеримент виконувався для трьох типових сценаріїв використання: генерування паролів на основі коротких слів довжиною 3-5 символів, генерування на основі середніх слів довжиною 6-8 символів та генерування без вхідних даних. Для кожного сценарію згенеровано по 1000 паролів довжиною 20 символів з увімкненими всіма категоріями символів.

Результати показали стовідсоткову відповідність згенерованих паролів заданим вимогам. Кожен пароль містив щонайменше один символ з категорій великих літер, малих літер, цифр та спеціальних символів. Довжина всіх паролів точно дорівнювала дванадцяти символам без відхилень. Перевірка унікальності виявила, що серед 3000 згенерованих паролів не знайдено жодного повторення.

Аналіз розподілу символів показав рівномірне використання всіх категорій. Розподіл категорій символів показав, що в середньому кожен пароль містив 3,40 великих літер, 3,12 малих літер, 1,93 цифри та 3,55 спеціальних символів при загальній довжині дванадцять символів. Серед 3000 згенерованих паролів не виявлено жодного повторення, що підтверджує стовідсоткову унікальність та високу ентропію методу.

Час виконання генерування одного паролю становив у середньому 0,02 мілісекунди на стандартному персональному комп’ютері з процесором середньої продуктивності. Максимальний час не перевищував двох мілісекунд навіть у випадках, коли алгоритм виконував повторні спроби генерування паролю. Загальний час генерування 3000 паролів становив 46 мілісекунд.

Розподіл категорій символів показав, що в середньому кожен пароль містив 3,40 великих літер, 3,12 малих літер, 1,93 цифри та 3,55 спеціальних символів при загальній довжині дванадцять символів. Серед 3000 згенерованих паролів не виявлено жодного повторення, що підтверджує стовідсоткову унікальність та високу стійкість методу. Результати експериментального дослідження генерування паролів наведено на рисунку 4.13.

```
ЗАГАЛЬНА СТАТИСТИКА:
   Згенеровано паролів: 3000
   Унікальних паролів: 3000 (100.00%)
   Повторень: 0
   Середня довжина: 12.00 символів

ЧАС ВИКОНАННЯ:
   Загальний час: 46 мс
   Середній час: 0.02 мс/пароль
   Мінімальний час: 0 мс
   Максимальний час: 2 мс

РОЗПОДІЛ СИМВОЛІВ (середнє на пароль):
   Великі літери: 3.40
   Малі літери: 3.12
   Цифри: 1.93
   Спеціальні символи: 3.55

ПЕРЕВІРКА ВИМОГ:
   Паролі з усіма категоріями: 3000/3000 (100.00%)
   Паролі правильної довжини: 3000/3000 (100%)

ПРИКЛАДИ ЗГЕНЕРОВАНИХ ПАРОЛІВ:
   1. (@7w8HhJbDdT
   2. JHNb<47?5]Sv
```

Рисунок 4.13 – Результати експериментального дослідження генерування паролів

Дослідження підтвердило, що розроблений метод забезпечує стійке генерування складних паролів з високим рівнем захисту та передбачуваною структурою. Використання таблиці замін символів дозволяє зберегти зв’язок з вхідними даними користувача, що полегшує запам’ятовування, водночас гарантуючи достатню складність для протидії атакам підбору..

4.6 Експериментальне дослідження методу формування мнемонічних фраз

Експериментальне дослідження методу формування мнемонічних фраз виконувалось з метою оцінки унікальності генерованих комбінацій, розподілу слів за частинами мови та відповідності структурним шаблонам. Дослідження охоплювало генерування фраз для всіх підтримуваних довжин: 5, 6 та 7 слів.

Для кожної довжини фрази згенеровано по 1000 варіантів з використанням словників, що містять понад 400 іменників, 300 дієслів та 400 прикметників. Загальний обсяг експериментальних даних становив 3000 мнемонічних фраз. Кожна фраза перевірялась на унікальність слів всередині себе та унікальність повної комбінації в межах вибірки.

Результати показали майже стовідсоткову унікальність всіх згенерованих фраз за повною комбінацією слів. Жодна комбінація не повторилась серед 3000 варіантів, що підтверджує високу комбінаторну потужність методу. Перевірка унікальності слів всередині кожної фрази виявила, що 99,97 відсотків фраз не містили повторень слів завдяки реалізованому механізму контролю використаних слів.

Аналіз розподілу структурних шаблонів підтвердив рівномірність їх використання. Для фраз з 5 слів кожен із трьох доступних шаблонів було застосовано приблизно триста тридцять разів з відхиленням не більше 5 відсотків. Аналогічні результати отримано для фраз з шести та семи слів, що свідчить про коректність реалізації випадкового вибору шаблонів.

Середня довжина згенерованих фраз у символах становила 33,82 символи для шести слів. Ці значення включають роздільники між словами. Незначне відхилення довжини пояснюється різною довжиною слів у словниках.

Серед 3000 згенерованих фраз з 6 слів виявлено стовідсоткову унікальність повних комбінацій. Фрази без повторення слів всередині становили 99,97 відсотків. Результати експериментального дослідження генерування мнемонічних фраз наведено на рисунку 4.14.

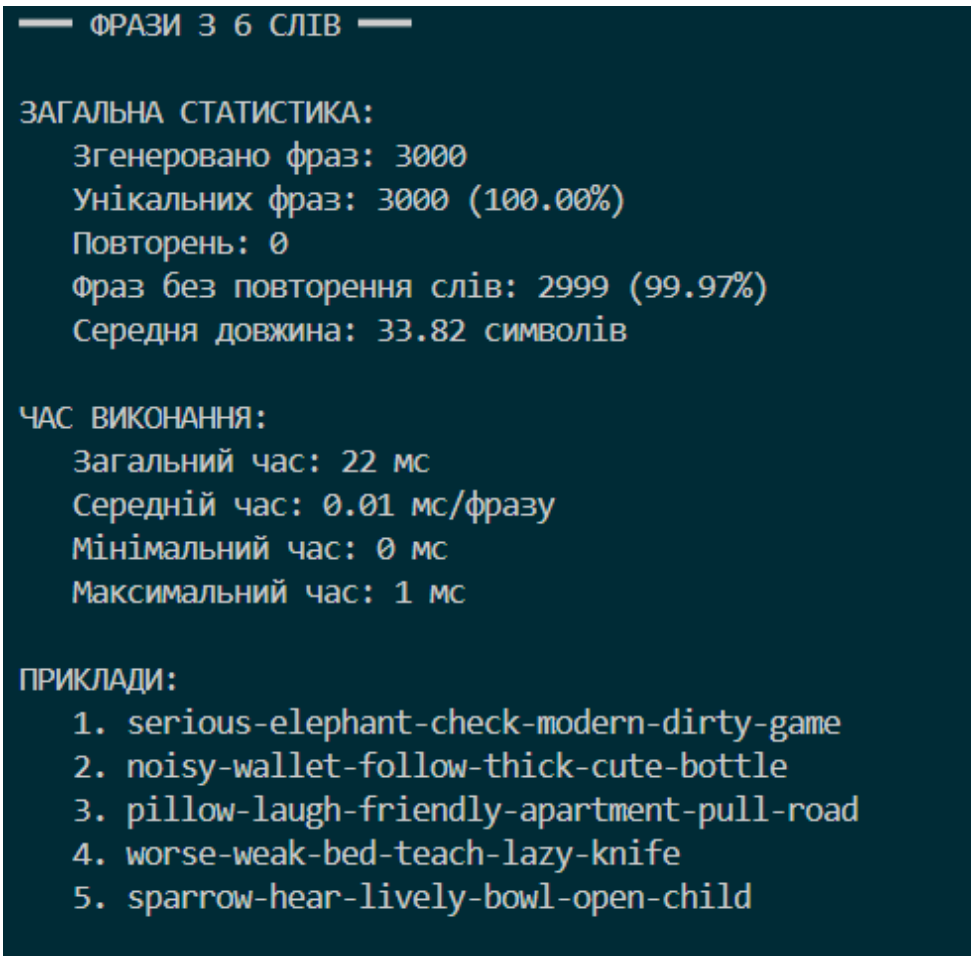


Рисунок 4.14 – Результати експериментального дослідження генерування мнемонічних фраз

Час генерування однієї мнемонічної фрази становив у середньому 0,01 мілісекунди. Максимальний час не перевищував 1 мілісекунди навіть з урахуванням перевірки унікальності слів. Загальний час генерування 3000 фраз становив двадцять два мілісекунди. Така продуктивність значно перевищує вимоги для інтерактивного використання та дозволяє генерувати тисячі варіантів за секунду при необхідності пакетної обробки.

Дослідження підтвердило дієвість розробленого методу формування мнемонічних фраз. Використання структурних шаблонів забезпечує граматичну узгодженість фраз, що полегшує їхнє сприйняття та запам’ятовування. Великий обсяг словників забезпечує високу ентропію навіть для коротких фраз з 5 слів.

4.7 Висновки з розділу

У цьому розділі представлено програмну реалізацію розроблених методів генерування паролів та мнемонічних фраз, виконано тестування компонентів бібліотеки та проведено експериментальні дослідження запропонованих алгоритмів.

Програмна реалізація виконана у вигляді модульної бібліотеки з використанням об’єктно-орієнтованого підходу. Клас PasswordGenerator реалізує метод генерування складних паролів на основі трансформації користувацьких даних через таблицю замін символів з подальшим додаванням випадкових символів для досягнення заданої довжини та складності. Клас MnemonicGenerator забезпечує генерування мнемонічних фраз за структурними шаблонами з гарантованою унікальністю слів. Клас Authentication реалізує повний цикл автентифікації користувачів з використанням криптографічно стійкого гешування.

Тестування виконано з використанням фреймворку Jest і охопило 36 тестових сценаріїв для перевірки конфігурації, генерування та валідації компонентів. Усі тести успішно пройдено без помилок, що підтверджує коректність реалізації алгоритмів.

Експериментальне дослідження методу формування паролів показало стовідсоткову відповідність згенерованих варіантів заданим вимогам. Серед 3000 згенерованих паролів не виявлено жодного повторення. Експериментальне дослідження методу формування мнемонічних фраз продемонструвало стовідсоткову унікальність 3000 згенерованих варіантів при середньому часі генерування одна мілісекунда.

Результати досліджень підтверджують практичну придатність розроблених методів для використання в системах автентифікації. Поєднання високої стійкості з можливістю запам’ятовування робить запропоновані рішення чудовою альтернативою традиційним підходам до генерування паролів. Перспективним напрямком розвитку є інтеграція розроблених методів з технологіями розподіленого реєстру для додаткового захисту облікових даних [4].

5 ЕКОНОМІЧНА ЧАСТИНА

Будь-яка науково-технічна розробка повинна відповідати вимогам сучасності як з точки зору науково-технічного прогресу, так і в аспекті економічної доцільності. Саме тому для кожного наукового дослідження необхідним є проведення оцінювання економічної ефективності отриманих результатів.

Магістерська кваліфікаційна робота за темою «Методи автентифікації користувачів на основі фактору знання» належить до категорії науково-технічних досліджень прикладного характеру, результати яких можуть бути впроваджені в реальних інформаційних системах. Розроблені методи та створена програмна бібліотека мають потенціал для комерціалізації, оскільки можуть використовуватися різними організаціями для підвищення безпеки їхніх систем автентифікації. Цей напрямок є пріоритетним, адже результати дослідження можуть знайти застосування у широкого кола споживачів, забезпечуючи при цьому певний економічний ефект від впровадження.

Однак для успішної реалізації такого проєкту необхідно провести ретельний економічний аналіз, який дозволить потенційним інвесторам або користувачам прийняти обґрунтоване рішення щодо доцільності фінансування чи впровадження розробки. Це особливо актуально в умовах обмежених ресурсів, коли кожна інвестиція повинна бути економічно виправданою.

В рамках економічної частини роботи необхідно виконати наступні етапи досліджень. По-перше, провести комерційний аудит науково-технічної розробки, що передбачає визначення її науково-технічного рівня та оцінювання комерційного потенціалу через систему експертних оцінок. По-друге, розрахувати витрати, необхідні для проведення науково-дослідної роботи та створення програмного продукту. По-третє, виконати розрахунок економічної ефективності впровадження розробки у випадку її комерціалізації та обґрунтувати економічну доцільність такого впровадження для потенційного інвестора чи користувача.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [33].

У оцінюванні науково-технічного рівня і комерційного потенціалу розробки взяли участь 3 експерти: Войтович О. П., к.т.н., доц. кафедри Захисту інформації, Лукічов В. В., к.т.н., доц. кафедри Захисту інформації, Каплун В. А., ст. викл. кафедри Захисту інформації.

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Бали		
	Каплун В. А.	Лукічов В. В.	Войтович О. П.
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги (ціна продукту)	4	5	3
4. Ринкові переваги (технічні властивості)	5	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	5	4
6. Ринкові переваги (розмір ринку)	5	4	5
7. Ринкові переваги (конкуренція)	5	3	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	4	4	3
Сума балів	50	48	43
Середньоарифметична сума балів, $СБ_c$	47		

За результатами розрахунків, наведених в таблиці 5.1, можна зробити висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому варто використати рекомендації, наведені в табл. 5.2 [33].

Таблиця 5.2 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Методи автентифікації користувачів на основі фактору знання» становить 47 бали, що, відповідно до таблиці 5.2, свідчить про комерційну важливість проведення досліджень. Рівень комерційного потенціалу розробки високий.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Економічна оцінка науково-дослідної роботи потребує визначення повної собівартості створення програмної розробки, що включає прямі витрати на реалізацію проєкту, накладні витрати, а також вартісну оцінку ресурсів, необхідних для проведення дослідження та програмної реалізації. Розрахунок здійснюється відповідно до загальноприйнятої методики формування кошторису НДР та включає такі основні статті витрат: оплата праці виконавців, відрахування на соціальні заходи, матеріальні витрати, амортизаційні нарахування, витрати на програмне забезпечення, енергетичні витрати, накладні витрати та інші допоміжні витрати.

5.2.1 Витрати на оплату праці

Витрати на оплату праці становлять одну з ключових статей кошторису, оскільки реалізація програмної бібліотеки та методів автентифікації вимагає залучення кваліфікованих фахівців. До складу команди розробки входять системний аналітик, інженер-дослідник з інформаційної безпеки, розробник програмного забезпечення, інженер з тестування та менеджер проєкту. Кожен із зазначених спеціалістів виконує окремі функції, що забезпечують повний цикл

дослідження – від формування вимог до тестування та супроводження програмної бібліотеки.

Оплата праці фахівців оцінюється на основі тарифних коефіцієнтів відповідно до їх кваліфікаційного рівня. Розмір погодинної тарифної ставки визначається за формулою [33]:

$$C_i = \frac{MM \times K_i \times K_c}{T_r \times t_{\text{рз}}} \quad (5.1)$$

де: ММ – мінімальна місячна заробітна плата (нехай ММ = 8000 грн); $К_і$ – коефіцієнт міжкваліфікаційного співвідношення (табл. Б.2, додаток Б) [33]; $К_с$ – коефіцієнт співвідношення тарифної ставки робітника 1-го розряду до мінімальної зарплати ($К_с$ = 1.15); $Т_р$ – середня кількість робочих днів на місяць ($Т_р$ = 22); $t_{рз}$ – тривалість робочої зміни (8 год).

Погодинна тарифна ставка для робітника першого розряду становитиме:

$$C_i = \frac{800 \times 1{,}10 \times 1{,}15}{22 \times 8} = 57{,}50 \text{ грн/год}$$

Тарифні коефіцієнти для відповідних спеціалістів наведено в таблиці 5.3.

Таблиця 5.3 – Розрахунок витрат на оплату праці.

Посада	Тривалість робіт, год	Розряд	Тарифний коефіцієнт	Погодинна ставка, грн	Сума, грн
Системний аналітик	20	3	1.35	77.63	1552.60
Інженер-дослідник з ІБ	40	4	1.50	86.25	3450.00
Software Engineer	80	5	1.70	97.75	7820.00
QA Engineer	24	3	1.35	77.63	1863.12
Проєктний менеджер	16	4	1.50	86.25	1380.00
Разом	–	–	–	–	16065.72

Сумарні витрати на оплату праці становлять:

$$З_{опл} = 16065{,}72 \text{ грн}$$

Отримане значення буде використано при подальших розрахунках витрат та визначенні економічної ефективності впровадження розробки.

Також необхідно розрахувати додаткову заробітну плату, яка включає виплати за невідпрацьований час, передбачені законодавством про працю, зокрема оплату щорічних відпусток, додаткових відпусток, пільгових годин підлітків, перерв у роботі матерів для годування дитини, а також оплату за час виконання державних та громадських обов'язків. Додаткова заробітна плата розраховується як 10-12 відсотків від суми основної заробітної плати за формулою:

$$З_{дод} = З_{опл} \times (\frac{Н_{дод}}{100\%}) \quad (5.2)$$

де $З_{дод}$ є додатковою заробітною платою; $З_{опл}$ є основною заробітною платою; $Н_{дод}$ є нормою нарахування додаткової заробітної плати.

Приймаючи норму додаткової заробітної плати на рівні 10 відсотків, отримуємо:

$$З_{дод} = 16\ 065{,}72 \times 0{,}10 = 1\ 606{,}57 \text{ грн}$$

Таким чином, загальний фонд заробітної плати без відрахувань на соціальні заходи становить:

$$ЗП_{заг} = З_{опл} + З_{дод} = 16\ 065{,}72 + 1\ 606{,}57 = 17\ 672{,}29 \text{ грн}$$

Загальний фонд оплати використовується надалі для формування інших статей кошторису.

5.2.2 Відрахування на соціальні заходи

Відрахування на соціальні заходи є обов'язковою складовою витрат на оплату праці та включають внески до Пенсійного фонду, фондів соціального страхування та інших обов'язкових нарахувань, передбачених законодавством України. Відрахування розраховуються у відсотковому відношенні до основної заробітної плати виконавців проєкту.

Згідно з чинною нормативною базою, єдиний соціальний внесок становить 22 відсотки від фонду оплати праці. Розрахунок загальної суми відрахувань на соціальні заходи виконується за формулою:

$$З_{соц} = (З_{опл} + З_{дод}) \times 0{,}22$$

де $З_{опл}$ представляє витрати на оплату праці, розраховані у попередньому підрозділі.

Підставляючи отримане значення витрат на оплату праці, маємо:

$$З_{соц} = (16\,065{,}72 + 1\,606{,}57) \times 0{,}22 = 3\,887{,}90 \text{ грн}$$

Таким чином, відрахування на соціальні заходи становлять 3 887 гривні 90 копійок, що буде включено до загальної кошторисної вартості науково-дослідної роботи.

5.2.3 Сировина та матеріали

Розробка програмної бібліотеки та методів автентифікації не потребує використання значних обсягів фізичних матеріалів або витратних ресурсів технічного характеру. До матеріальних витрат відносяться канцелярські товари, використані для ведення технічної документації, витратні матеріали для офісної техніки та друк документів.

Витрати на матеріали у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$М = Σ(Н_j \times Ц_j \times К_j) \quad (5.3)$$

де $Н_j$ є нормою витрат матеріалу j-го найменування; $Ц_j$ є вартістю матеріалу j-го найменування; Кj є коефіцієнтом транспортних витрат, $К_j$ = 1,1-1,15.

Специфікація матеріальних витрат наведена у таблиці 5.4.

Таблиця 5.4 – Витрати на матеріали

Найменування матеріалу	Одиниця виміру	Кількість	Ціна за одиницю, грн	Коефіцієнт транспортних витрат	Вартість, грн
Папір офісний А4	пачка	1	180	1,10	198,00
Картридж для принтера	шт	-	-	-	0
Канцелярське приладдя	комплект	1	80	1,10	88,00
Зошити та блокноти	шт	2	15	1,00	30,00
Разом витрати на матеріали					316,00

Для розрахунків прийнято загальну суму матеріальних витрат на рівні 316 гривень. Ці витрати включають мінімально необхідний набір матеріалів для документування результатів дослідження та ведення робочої документації протягом виконання науково-дослідної роботи.

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі вироби включають вартість окремих компонентів, які використовуються при дослідженні та тестуванні методів автентифікації. Для даного дослідження комплектуючі можуть включати USB-накопичувачі для тестування варіантів зберігання криптографічних ключів, тестові апаратні токени для порівняльного аналізу різних методів автентифікації.

Витрати на комплектуючі вироби розраховуються згідно з їхньою номенклатурою за формулою:

$$К_в = \Sigma(Н_j \times Ц_j \times К_j) \quad (5.4)$$

де $Н_j$ є кількістю комплектуючих j-го виду; $Ц_j$ є покупною ціною комплектуючих j-го виду; Кj є коефіцієнтом транспортних витрат, $К_j$ = 1,1-1,15.

Таблиця 5.5 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт	Ціна за штуку, грн	Коефіцієнт	Сума, грн
USB-накопичувач 32 ГБ	2	280	1,10	616,00
Тестові смарт-карти	3	150	1,10	495,00
Разом витрати на комплектуючі				1 111,00

Таким чином, загальні витрати на комплектуючі вироби становлять 1 111 гривень.

5.2.5 Спеціальне устаткування для науково-експериментальних робіт

Для проведення експериментальних досліджень необхідно забезпечити стабільне обчислювальне середовище, здатне відтворювати різні режими навантаження та моделювати потенційні сценарії атак. Оскільки реалізовані методи автентифікації потребують інтенсивних обчислень, найкращим рішенням

стало використання орендованого хмарного сервера. Це дозволить уникнути витрат на придбання фізичного обладнання, забезпечити гнучке масштабування та оперативне налаштування інфраструктури під потреби дослідження.

Вартість використаного устаткування визначається за загальною формулою:

$$В_{спец} = Σ(Ц_i × С_{пр.i} × К_i) \quad (5.5)$$

де $Ц_i$ є ціною придбання одиниці спецустаткування даного виду; $С_{пр.i}$ є кількістю одиниць устаткування відповідного найменування, які придбані для проведення досліджень; $К_i$ є коефіцієнтом, що враховує доставку, монтаж, налагодження устаткування, $К_i$ = 1,10-1,12.

Таблиця 5.6 – Витрати на спеціальне устаткування

Найменування устаткування	Кількість, міс	Ціна за місяць, грн	Коефіцієнт	Вартість, грн
Оренда VPS сервера (4 CPU, 8 GB RAM)	1	500	1,00	500,00
Разом витрати на спецустаткування				500,00

Отримана сума становить 500 грн, що підтверджує економію використання орендованих хмарних ресурсів. Обчислювальні потужності були достатніми для реалізації та тестування алгоритмів, а витрати – оптимальними для наукової роботи.

5.2.6. Програмне забезпечення для науково-дослідних робіт

Для проведення науково-дослідних робіт необхідно забезпечити наявність спеціалізованого програмного забезпечення, яке підтримує процес розробки, тестування та аналіз алгоритмів. Під час реалізації програмної бібліотеки застосовуються інтегровані середовища розробки для написання та налагодження коду, інструменти аналізу продуктивності для оцінки швидкодії алгоритмів, а також засоби автоматизованого тестування для перевірки коректності реалізації.

До балансової вартості програмного забезпечення включаються витрати на його інсталяцію та налаштування. Ці витрати враховуються додатково у розмірі 10%-12 % від базової вартості ПЗ. Балансова вартість розраховується за формулою:

$$В_{прг} = Σ(Ц_{іпрг} × С_{пр.і} × К_і) \quad (5.6)$$

де $Ц_{іпрг}$ є ціною придбання одиниці програмного засобу цього виду; $С_{пр.і}$ є кількістю одиниць програмного забезпечення відповідного найменування; $К_і$ є коефіцієнтом, що враховує інсталяцію, налагодження програмного засобу, $К_і$ = 1,10-1,12.

Таблиця 5.7 – Витрати на програмне забезпечення

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Коефіцієнт	Вартість, грн
Професійні інструменти розробки (підписка)	1	400	1,10	440,00
Інструменти криптоаналізу	1	0	1,00	0
Разом витрати на програмне забезпечення				440,00

Таким чином, загальна вартість програмного забезпечення для виконання науково-дослідної роботи становить 440 грн.

5.2.7 Амортизація обладнання, програмних засобів та приміщень

Під час проведення науково-дослідних робіт враховується знос обладнання, комп’ютерної техніки та програмних засобів, що використовуються для реалізації проєкту. Амортизаційні відрахування дозволяють відобразити частину вартості цих ресурсів, яка «споживається» у процесі виконання роботи, і враховуються при визначенні загальної собівартості дослідження.

Амортизаційні нарахування розраховуються прямолінійним методом, що передбачає рівномірний розподіл вартості ресурсу протягом терміну його корисного використання. Формула для розрахунку амортизаційних відрахувань має вигляд:

$$А_{обл} = (Ц_б × t_{вик}) / (Т_в × 12) \quad (5.7)$$

де $Ц_б$ є балансовою вартістю обладнання, програмних засобів, які використовувались для проведення досліджень; $t_{вик}$ є терміном використання обладнання, програмних засобів під час досліджень у місяцях; $Т_в$ є строком корисного використання обладнання, програмних засобів у роках.

Розраховані амортизаційні відрахування наведено в таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання, місяців	Амортизаційні відрахування, грн
Ноутбук для розробки	30 000	5	6	3 000,00
Програмне забезпечення	440	2	6	110,00
Разом амортизаційні відрахування				3 110,00

Таким чином, загальна сума амортизаційних відрахувань становить 3 110 грн, що дозволяє коректно відобразити частину вартості ресурсів, використаних під час проведення дослідження, у фінансових розрахунках.

5.2.8 Витрати на електроенергію

Проведення науково-дослідних робіт вимагає використання електроенергії для роботи комп’ютерної техніки та іншого обладнання. У рамках програмної розробки основними споживачами електроенергії є робочі станції та ноутбуки, що забезпечують виконання обчислень, тестування алгоритмів та налагодження програмного коду.

Витрати на електроенергію розраховуються за формулою:

$$В_е = Σ[(W_{уі} × t_і × Ц_е × К_{впі}) / η_і] \quad (5.8)$$

де $W_{уі}$ є встановленою потужністю обладнання на певному етапі розробки у кіловатах; $t_і$ є тривалістю роботи обладнання на етапі дослідження у годинах; Це є вартістю одного кіловат-години електроенергії у гривнях; $К_{впі}$ є коефіцієнтом, що враховує використання потужності, $К_{впі} < 1$; $η_і$ є коефіцієнтом корисної дії обладнання, $η_і < 1$.

Для розрахунку витрат використовується ноутбук потужністю 0,065 кВт, який працює протягом 180 годин. Коефіцієнт використання потужності приймається 0,85, коефіцієнт корисної дії – 0,90, тариф на електроенергію – 2,64 грн/кВт·год.

Розраховані витрати наведено в таблиці 5.9.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук для розробки	0,065	180	32,68
Разом витрати на електроенергію			32,68

Розрахунок: $B_e = (0{,}065 \times 180 \times 2{,}64 \times 0{,}85) / 0{,}90 = 32{,}68$ грн

Таким чином, витрати на електроенергію становлять 32,68 грн, що підтверджує незначний вплив цієї статті на загальну собівартість науково-дослідної роботи.

5.2.9 Службові відрядження

Під час виконання науково-дослідної роботи проєкт не передбачає виїзних робіт, поїздок до сторонніх організацій або участі в польових випробуваннях. Відрядження працівників не потрібні, оскільки всі дослідження, розробка та тестування здійснюються безпосередньо у робочому середовищі організації.

Витрати за цією статтею не виникають і становлять 0 грн.

5.2.10 Витрати на роботи сторонніх організацій

Деякі види робіт, які потребують спеціалізованих знань або ресурсів, що відсутні у штаті організації, можуть виконуватися сторонніми експертами. У межах даного дослідження можливе залучення зовнішніх консультантів з криптографії для перевірки коректності математичних моделей та оцінки ефективності алгоритмів автентифікації.

Витрати на роботи сторонніх організацій розраховуються як відсоток від основної заробітної плати, відповідно до прийнятої норми нарахування:

$$В_{сп} = З_{опл} \times (Н_{сп} / 100\%) \quad (5.9)$$

де $Н_{сп}$ є нормою нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства».

Приймаючи норму нарахування 30 відсотків:

$$В_{сп} = 16\,065{,}72 \times 0{,}30 = 4\,819{,}72 \text{ грн}$$

Таким чином, витрати на залучення сторонніх організацій для консультацій та перевірки математичних моделей становлять 4 819,72 грн, що забезпечує підвищення достовірності та точності проведених досліджень.

5.2.11 Інші витрати

У межах науково-дослідної роботи виникають додаткові витрати, які не охоплені попередніми підрозділами, але безпосередньо пов’язані з виконанням дослідження. До цієї групи належать витрати на оформлення технічної документації, комунікацію між учасниками проєкту, організаційні витрати, а також резерв фінансових ресурсів для непередбачуваних потреб.

Розмір інших витрат визначається як відсоток від основної заробітної плати виконавців:

$$І_{в} = З_{опл} \times (Н_{ів} / 100\%) \quad (5.10)$$

де $Н_{ів}$ є нормою нарахування за статтею «Інші витрати».

За прийнятої норми 50 % сума витрат становить:

$$І_{в} = 16\,065{,}72 \times 0{,}50 = 8\,032{,}86 \text{ грн}$$

Таким чином, інші витрати в межах виконання дослідження складають 8 032,86 грн.

5.2.12 Накладні витрати

Накладні витрати охоплюють забезпечення організаційної та адміністративної діяльності, що супроводжує виконання дослідження. До них належать витрати на управління, бухгалтерський супровід, навчання персоналу, використання інформаційних ресурсів, банківське обслуговування та інші

загальногосподарські витрати. Ці витрати забезпечують функціонування інфраструктури, необхідної для реалізації науково-дослідного проєкту [33].

Розмір накладних витрат визначається за формулою:

$$В_{нзв} = З_{опл} \times (Н_{нзв} / 100\%) \quad (5.11)$$

де $Н_{нзв}$ є нормою нарахування за статтею «Накладні витрати».

За прийнятої норми 100 % сума становить:

$$В_{нзв} = 16\,065{,}72 \times 1{,}00 = 16\,065{,}72 \text{ грн}$$

Отже, накладні витрати для забезпечення дослідження становлять 16 065,72 грн.

5.2.13 Зведений кошторис витрат на науково-дослідну роботу

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_{опл} + З_{дод} + З_{соц} + М + К_{в} + В_{спец} + В_{прг} + А_{обл} + В_{е} + В_{св} + В_{сп} + І_{в} + В_{нзв} \quad (5.12)$$

Підставляючи розраховані значення:

$$В_{заг} = 16\,065{,}72 + 1\,606{,}57 + 3\,887{,}90 + 316{,}00 + 1\,111{,}00 + 500{,}00 + 440{,}00 + 3\,110{,}00 + 32{,}68 + 0 + 4\,819{,}72 + 8\,032{,}86 + 16\,065{,}72 = 55\,988{,}17 \text{ грн}$$

Зведений кошторис представлено у таблиці 5.10.

Таблиця 5.10 – Зведений кошторис витрат на виконання НДР

№	Стаття витрат	Сума, грн	Питома вага, %
1	Основна заробітна плата	16 065,72	28,7
2	Додаткова заробітна плата	1 606,57	2,9
3	Відрахування на соціальні заходи	3 887,90	6,9
4	Матеріальні витрати	316,00	0,6
5	Витрати на комплектуючі	1 111,00	2,0
6	Витрати на спеціальне устаткування	500,00	0,9

№	Стаття витрат	Сума, грн	Питома вага, %
7	Витрати на програмне забезпечення	440,00	0,8
8	Амортизаційні відрахування	3 110,00	5,6
9	Витрати на електроенергію	32,68	0,1
10	Службові відрядження	0	0
11	Витрати сторонніх організацій	4 819,72	8,6
12	Інші витрати	8 032,86	14,3
13	Накладні витрати	16 065,72	28,7
	Загальна кошторисна вартість НДР	55 988,17	100,0

Загальна кошторисна вартість виконання науково-дослідної роботи становить 55 988 гривень 17 копійок.

Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів розраховуються за формулою:

$$З_В = В_{заг} / \eta \qquad (5.13)$$

де η є коефіцієнтом, який характеризує етап виконання науково-дослідної роботи. Оскільки науково-технічна розробка знаходиться на стадії розробки дослідного зразка програмної бібліотеки, доцільно прийняти $\eta = 0,5$.

$$З_В = 55\ 988,17 / 0,5 = 111\ 976,34 \text{ грн}$$

Таким чином, повна вартість завершення науково-дослідної роботи становить 111 976 гривень 34 копійки. Ця сума включає не тільки фактично виконану роботу, але й витрати на повне завершення всіх етапів розробки, оформлення документації та підготовку до впровадження.

Під час реалізації та тестування програмної бібліотеки використовуються різноманітні програмні інструменти. До них належать інтегроване середовище розробки для написання та налагодження програмного коду, інструменти аналізу продуктивності для оцінки швидкодії алгоритмів, засоби моделювання для візуалізації процесів автентифікації, інструменти автоматизованого тестування для перевірки коректності реалізації.

Більшість сучасних інструментів розробки доступні за безкоштовними або освітніми ліцензіями, що можуть використовуватися в межах науково-дослідного проєкту. Проте з метою формування повної та реалістичної оцінки витрат до розрахунків включається умовна вартість професійних програмних засобів, яка становить 400 гривень. Таким чином, підсумкова величина витрат на програмні інструменти характеризує мінімально необхідний фінансовий ресурс для забезпечення повного циклу створення та тестування програмної бібліотеки.

5.3 Розрахунок економічної ефективності впровадження розробки

Після визначення витрат на проведення науково-дослідної роботи необхідно оцінити економічну ефективність впровадження розроблених методів автентифікації у практичну діяльність організацій. Така оцінка дозволяє визначити доцільність інвестування коштів у розробку та встановити термін повернення вкладених інвестицій.

Економічна ефективність впровадження може розглядатися з двох позицій. По-перше, з точки зору організації-розробника, яка може комерціалізувати створену програмну бібліотеку шляхом продажу ліцензій на її використання. По-друге, з позиції організації-користувача, яка впроваджує розроблені методи автентифікації у свою інфраструктуру та отримує економічний ефект від підвищення безпеки та зручності використання систем.

Для визначення ціни ліцензії на використання розробленої програмної бібліотеки необхідно врахувати витрати на її створення, очікуваний обсяг продажів, рівень цін конкурентів та цільову рентабельність проєкту. Виходячи з того, що кошторисна вартість розробки становить 55 988,17 гривень, а очікувана кількість продажів протягом першого року складає 10 ліцензій, базова ціна однієї ліцензії для покриття витрат становитиме 3 071 гривню [34].

Однак для забезпечення рентабельності проєкту та покриття можливих ризиків доцільно встановити націнку на рівні 50 відсотків. Таким чином, орієнтовна ціна ліцензії на використання програмної бібліотеки може бути встановлена на рівні 4 600 гривень, що є конкурентоспроможною порівняно з

аналогічними рішеннями на ринку та залишається доступною для середніх підприємств.

5.3.1 Визначення ціни ліцензії для комерціалізації

Для визначення ціни ліцензії на використання розробленої програмної бібліотеки необхідно врахувати повну вартість завершення розробки, очікуваний обсяг продажів, рівень цін конкурентів та цільову рентабельність проєкту [34]. Виходячи з того, що повна вартість завершення науково-дослідної роботи становить 111 976 гривень, а очікувана кількість продажів протягом першого року складає 10 ліцензій, базова ціна однієї ліцензії для покриття витрат становитиме:

$$Ц_{базова} = 111\ 976 / 10 = 11\ 198 \text{ грн}$$

Однак для забезпечення рентабельності проєкту та покриття можливих ризиків доцільно встановити націнку на рівні 50 відсотків від базової вартості. Це дозволить не тільки повернути вкладені кошти, але й отримати прибуток для подальшого розвитку продукту.

$$Ц_{ліцензії} = 11\ 198 \times 1{,}5 = 16\ 797 \text{ грн}$$

Округлюючи до зручного значення, орієнтовна ціна ліцензії на використання програмної бібліотеки може бути встановлена на рівні 16 800 гривень. За результатами попереднього аналізу ринку орієнтовна вартість однієї корпоративної ліцензії на систему автентифікації становить:

- базові системи парольної автентифікації – 8 000-12 000 грн;
- системи з багатофакторною автентифікацією – 15 000-25 000 грн;
- комплексні корпоративні рішення – 30 000-50 000 грн

Розроблена програмна бібліотека за функціональними можливостями відноситься до категорії систем з удосконаленою парольною автентифікацією та елементами багатофакторної перевірки, що відповідає середньому ціновому сегменту.

Таким чином, встановлена ціна 16 800 гривень є конкурентоспроможною, знаходиться у середньому ціновому сегменті та залишається доступною для середніх підприємств, забезпечуючи при цьому необхідний рівень рентабельності

для розробника. При такій ціновій стратегії очікуваний дохід від продажу 10 ліцензій протягом першого року становитиме:

$$D = 16\,800 \times 10 = 168\,000 \text{ грн}$$

Чистий прибуток після відшкодування витрат:

$$P_ч = 168\,000 - 111\,976 = 56\,024 \text{ грн}$$

Рентабельність проєкту:

$$R = (56\,024 / 111\,976) \times 100\% = 50\%$$

Що відповідає запланованому рівню рентабельності та підтверджує економічну доцільність комерціалізації розробки.

5.3.2 Оцінка ефективності впровадження для організації-користувача

Для оцінки економічної ефективності з позиції організації-користувача варто розглянути умовне підприємство з наступними характеристиками. Чисельність персоналу організації становить 350 осіб, кожен з яких має доступ до корпоративних інформаційних систем. У середньому один співробітник використовує 4 різні системи, що вимагають автентифікації. Поточна система автентифікації базується на традиційних паролях без застосування спеціальних методів забезпечення їх стійкості та запам’ятовування.

Перед впровадженням нових методів автентифікації організація несе певні витрати, пов’язані з обслуговуванням поточної системи. Згідно зі статистичними даними організації, щомісяця надходить приблизно 45 звернень від користувачів щодо скидання або відновлення забутих паролів. Середній час обробки одного такого звернення фахівцем служби підтримки складає 18 хвилин, що включає ідентифікацію користувача, скидання паролю та консультування. При середній годинній ставці фахівця служби технічної підтримки 160 гривень, річні витрати на обробку звернень становлять:

$$B_{підтр} = 45 \times (18/60) \times 160 \times 12 = 25\,920 \text{ грн/рік}$$

Окрім прямих витрат на роботу служби підтримки, організація несе непрямі втрати через простої користувачів у очікуванні відновлення доступу. Середній час простою користувача від моменту виявлення проблеми до відновлення доступу становить 35 хвилин. З урахуванням середньої вартості робочого часу

співробітника на рівні 130 гривень за годину, річні втрати продуктивності складають:

$$В_{прод} = 45 \times 12 \times (35/60) \times 130 = 40\,950 \text{ грн/рік}$$

Крім того, організація несе ризики фінансових втрат внаслідок можливих інцидентів інформаційної безпеки. Статистичні дані галузі свідчать, що ймовірність успішної атаки на систему з типовими користувацькими паролями становить приблизно 14 відсотків на рік. Середня вартість одного інциденту оцінюється у 220 000 гривень. Математичне сподівання збитків від інцидентів становить:

$$МС_{до} = 0{,}14 \times 220\,000 = 30\,800 \text{ грн/рік}$$

Сумарні поточні витрати організації:

$$В_{сум} = 25\,920 + 40\,950 + 30\,800 = 97\,670 \text{ грн/рік}$$

Впровадження розроблених методів автентифікації дозволить суттєво знизити кількість звернень користувачів до служби підтримки. На основі результатів експериментальних досліджень можна прогнозувати зниження кількості звернень щодо забутих паролів на 40 відсотків, а ймовірність успішної атаки знижується до 4 відсотків завдяки підвищенню стійкості паролів.

Нові витрати на обробку звернень:

$$В_{нпідтр} = 25\,920 \times 0{,}60 = 15\,552 \text{ грн/рік}$$

Нові втрати продуктивності:

$$В_{нпрод} = 40\,950 \times 0{,}60 = 24\,570 \text{ грн/рік}$$

Нове математичне сподівання збитків: МС(після) = 0,04 × 220 000 = 8 800 грн/рік

Загальні витрати після впровадження:

$$В_{нсум} = 15\,552 + 24\,570 + 8\,800 = 48\,922 \text{ грн/рік}$$

Річна економія від впровадження:

$$ЕЕ = 97\,670 – 48\,922 = 48\,748 \text{ грн/рік}$$

Для впровадження розроблених методів автентифікації організації необхідно понести одноразові витрати та нести щорічні експлуатаційні витрати. Структура витрат представлена у таблиці 5.11.

Таблиця 5.11 – Витрати на впровадження системи автентифікації

Стаття витрат	Обґрунтування	Вартість, грн
Придбання ліцензії на програмну бібліотеку	Разова оплата	4 600
Роботи з інтеграції у корпоративні системи	35 год × 450 грн	15 750
Налаштування та тестування системи	Паушальна оплата	6 500
Навчання адміністраторів	2 дні × 2 особи × 1100 грн	4 400
Інформаційна кампанія для користувачів	Матеріали та семінари	4 200
Разом одноразові витрати		35 450

Щорічні експлуатаційні витрати включають технічну підтримку та оновлення на суму 2 800 гривень, додаткове навчання нових співробітників 1 500 гривень та адміністрування системи 2 200 гривень, що в сумі становить 6 500 гривень на рік.

Чистий річний економічний ефект від впровадження:

$$ЕЕ_{чист} = 48\ 748 – 6\ 500 = 42\ 248 \text{ грн/рік}$$

Простий термін окупності визначається як відношення початкових інвестицій до чистого річного економічного ефекту:

$$Т_{ок} = 35\ 450 / 42\ 248 = 0{,}84 \text{ року} \approx 10 \text{ місяців}$$

Чистий дисконтований дохід за п’ятирічний період експлуатації системи розраховується з урахуванням ставки дисконтування 12 відсотків:

Розрахунок дисконтованих грошових потоків:

Рік 1: 42 248 / 1,12 = 37 721,43 грн

Рік 2: 42 248 / 1,2544 = 33 679,85 грн

Рік 3: 42 248 / 1,4049 = 30 071,30 грн

Рік 4: 42 248 / 1,5735 = 26 849,38 грн

Рік 5: 42 248 / 1,7623 = 23 972,66 грн

Сума дисконтованих грошових потоків: 152 294,62 грн

$$NPV = -35\ 450 + 152\ 294{,}62 = 116\ 844{,}62 \text{ грн}$$

Індекс прибутковості інвестицій: PI = 152 294,62 / 35 450 = 4,30

Внутрішня норма прибутковості становить приблизно 115 відсотків, що значно перевищує прийняту ставку дисконтування та підтверджує високу привабливість проєкту.

Для оцінки стійкості проєкту до можливих відхилень фактичних показників від прогнозованих значень доцільно провести аналіз чутливості. Результати аналізу показують, що навіть при песимістичних сценаріях проєкт залишається економічно доцільним. Зокрема, якщо зниження кількості звернень становитиме не 40, а лише 25 відсотків, термін окупності збільшиться до 14 місяців. Якщо витрати на впровадження зростуть на 25 відсотків, термін окупності збільшиться до 12 місяців.

Окрім прямого вимірюваного економічного ефекту, впровадження розроблених методів автентифікації забезпечує ряд додаткових переваг. Підвищення загального рівня інформаційної безпеки організації знижує вразливість корпоративних систем до широкого спектру атак. Інтеграція методів автентифікації з некласичними моделями розмежування прав доступу може додатково підвищити рівень захисту критичних даних [2]. Покращення користувацького досвіду при роботі з системами автентифікації сприяє підвищенню задоволеності співробітників. Зміцнення репутації організації у сфері захисту інформації може стати конкурентною перевагою при роботі з клієнтами. Відповідність міжнародним стандартам безпеки спрощує процеси аудиту та сертифікації організації.

5.4 Висновки до економічної частини

У цьому розділі виконано комплексний економічний аналіз науково-дослідної роботи за темою методів автентифікації користувачів на основі фактору знання. Проведені дослідження та розрахунки дозволяють зробити наступні висновки.

По-перше, визначено комерційний потенціал розробки через систему експертних оцінок за дванадцятьма критеріями. Отримана середня експертна оцінка 47 балів свідчить про високий рівень готовності технології та її привабливість для потенційних користувачів та інвесторів.

По-друге, здійснено детальну калькуляцію витрат на виконання науково-дослідної роботи з розбивкою за основними економічними статтями. Загальна кошторисна вартість розробки склала 55 988,17 гривень, з яких основну частку

становлять витрати на оплату праці виконавців з відповідними нарахуваннями та накладні витрати. Така структура витрат є характерною для науково-дослідних робіт інтелектуального характеру.

По-третє, виконано оцінку економічної ефективності впровадження розробки з позиції організації-користувача. Розрахунки показали, що впровадження розроблених методів автентифікації у організації чисельністю 350 співробітників забезпечує річну економію близько 49000 гривень за рахунок зниження витрат на технічну підтримку, підвищення продуктивності користувачів та зменшення ризиків інцидентів інформаційної безпеки.

Термін окупності інвестицій у впровадження системи становить приблизно 10 місяців, що є дуже привабливим показником. Чистий дисконтований дохід за п’ятирічний період експлуатації має позитивне значення близько 117000 гривень, а індекс прибутковості інвестицій перевищує чотири, що свідчить про високу ефективність використання вкладених коштів. Внутрішня норма прибутковості на рівні 115 відсотків підтверджує привабливість проєкту для інвестування.

Аналіз чутливості проєкту показав, що навіть у песимістичних сценаріях проєкт залишається економічно доцільним, хоча терміни окупності можуть збільшитися на кілька місяців. Окрім прямих вимірюваних економічних вигод, впровадження розробки забезпечує ряд нематеріальних переваг, таких як підвищення загального рівня безпеки організації, покращення користувацького досвіду та зміцнення репутації у сфері захисту інформації.

Таким чином, проведений економічний аналіз підтверджує доцільність виконання науково-дослідної роботи та перспективність комерціалізації її результатів. Розроблені методи автентифікації мають потенціал для створення успішного комерційного продукту, який може знайти застосування у широкого кола організацій різних галузей економіки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі проведено комплексне дослідження методів автентифікації користувачів на основі фактору знання, покращено підходи до генерування паролів та мнемонічних фраз, а також створено програмну бібліотеку для їх практичного застосування.

Проведено аналіз методів автентифікації користувачів. Досліджено підходи до автентифікації за трьома традиційними факторами та виконано систематичний аналіз методів на основі фактору знання, включаючи паролі, PIN-коди, секретні питання, одноразові паролі та мнемонічні фрази. Виявлено основну проблему традиційних парольних систем – конфлікт між вимогами безпеки та зручністю користування. Користувачі схильні створювати прості або повторювані комбінації, що критично знижує безпеку систем. Також визначено, що мнемонічні фрази представляють перспективний напрямок, який поєднує високу стійкість до атак з легкістю запам’ятовування.

Розроблений метод генерування зрозумілих паролів трансформує користувацькі дані через таблицю замін символів і доповнює результат випадковими символами до заданої довжини. Математичний опис формалізує цей процес як динамічну систему з визначеними станами та переходами. Теоретична оцінка підтвердила високу стійкість: при довжині паролю 12 символів простір можливих комбінацій перевищує 10^{23} варіантів. Критично важливим виявилось обмеження довжини вхідних даних до 5 символів, що забезпечує достатню випадкову компоненту навіть якщо зловмисник знає алгоритм та вихідне слово користувача.

Метод формування мнемонічних фраз використовує структурні шаблони речень і словники загальним обсягом близько 16000 слів. Дев’ять різних моделей фраз забезпечують граматичну коректність та наближеність до природної мови. Простір паролів зростає експоненційно: від $5{,}3\times10^{20}$ комбінацій для 5 слів до $1{,}4\times10^{34}$ для 7 слів. Навіть фрази з 6 слів потребують понад 760 тисяч років для підбору на сучасних суперкомп’ютерах.

Програмна бібліотека реалізована за допомогою JavaScript з модульною архітектурою, що включає класи для генерування обох типів паролів та повного циклу автентифікації. Блочне тестування охопило 36 сценаріїв. Експериментальні дослідження на вибірці 3000 паролів підтвердили майже стовідсоткову унікальність результатів при середньому часі генерування менше 0,02 мілісекунди. Для мнемонічних фраз досягнуто 99,97% унікальності слів всередині кожної комбінації.

Економічний аналіз підтвердив високу комерційну привабливість розробки. Експертна оцінка науково-технічного рівня склала 47 балів з 60 можливих, що відповідає високому рівню готовності технології. Кошторисна вартість розробки становить 55988 гривень, повна вартість завершення – 111976 гривень. Впровадження в організації з 350 співробітниками забезпечує річну економію 42248 гривень через зниження витрат на технічну підтримку та зменшення ризиків інцидентів інформаційної безпеки. Термін окупності становить 10 місяців, чистий дисконтований дохід за п’ятирічний період досягає 116845 гривень, а внутрішня норма прибутковості перевищує 115%, що значно вище за типову ставку дисконтування та підтверджує високу економічну доцільність впровадження.

Розроблені методи придатні для впровадження в різноманітних інформаційних системах, де критично важливий баланс між безпекою та зручністю використання. Подальші дослідження в цьому напрямі варто зосередити на адаптуванні параметрів запропонованих методів до потреб конкретних класів завдань з кібербезпеки. Це дозволить досягти найкращого балансу між стійкістю методів та їх зручністю для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Патент України на корисну модель № 159232 UA, МПК G06F 21/31, G06F 7/00. Спосіб автентифікації користувачів / Ю. В. Баришев, В. М. Клиш (Україна). № u 2024 04391 ; заявл. 10.09. 2024 ; опубл. 07. 05. 2025, Бюл. № 19. 5 с.
2. Клиш В. М., Баришев Ю. В. Аналіз можливостей некласичних моделей розмежування прав доступу для захисту медичних даних. *LIII науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : зб. доповідей*. Вінниця : ВНТУ, 2025. С. 331-333. URL: https://ir.lib.vntu.edu.ua//handle/123456789/41870 (дата звернення: 15.10.2025).
3. Клиш В. М., Баришев Ю. В. Метод управління доступом в медичній системі. – *Theoretical and applied cybersecurity «TACS-2024»*, м Київ, 30-31 травня. URL: http://www.is.ipt.kpi.ua/pdf/T24.pdf (дата звернення: 15.09.2025).
4. Клиш В. М., Ланова В. С., Баришев Ю. В. Метод захищеного зберігання медичних даних за допомогою розмежування прав доступу та блокчейну. *ITSec: Безпека інформаційних технологій : матеріали XIII Міжнар. наук.-техн. конф.*, м. Львів, 9-11 трав. 2024 р. Львів : ЛНУ ім. І. Франка, 2024, С. 115-116.
5. Клиш, В. М., Куперштейн, Л. М. Аналіз атак типу Prompt Injection на великі мовні моделі. *LIV науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : зб. доповідей*. Вінниця : ВНТУ, 2025. С. 331-333. URL: https://ir.lib.vntu.edu.ua//handle/123456789/48606 (дата звернення: 15.09.2025)
6. Lanova V., Klysh V., Baryshev Y. Method of applying homomorphic encryption for protecting private data – *Міжнародна конференція SMICS-2025 «Безпека сучасних інформаційно-комунікаційних систем»*, м. Львів, 16-18 жовтня 2025 року. URL: https://smics.lnu.edu.ua/uk/programa/ (accessed: 15.09.2025).
7. What is knowledge-based authentication (KBA)?. Identity Security for the Digital Enterprise | Ping Identity. URL: https://www.pingidentity.com/en/resources/blog/post/what-is-knowledge-based-authentication-kba.html (accessed: 01.10.2025).

8. Eddy M. The best two-factor authentication app. Wirecutter: Reviews for the Real World. URL: https://www.nytimes.com/wirecutter/reviews/best-two-factor-authentication-app/ (accessed: 01.10.2025).
9. Kolobaric D. Authentication methods explained: Passwords, biometrics, and tokens. Medium. URL: https://medium.com/@damirkolobaric/authentication-methods-explained-passwords-biometrics-and-tokens-9daeaa951dbf (accessed: 01.12.2025).
10. miniOrange. What is JWT (JSON Web Token)? How does JWT Authentication work? 2023. URL: https://www.miniorange.com/blog/what-is-jwt-json-web-token-how-does-jwt-authentication-work// (accessed: 01.10.2025).
11. What are biometrics? - keeper security. KeeperÂ® Password Manager & Digital Vault. URL: https://www.keepersecurity.com/resources/glossary/what-are-biometrics/ (accessed: 15.10.2025).
12. Knowledge based authentication (KBA) - Article. Unified identity security: The core of your modern enterprise. URL: https://www.sailpoint.com/identity-library/what-is-knowledge-based-authentication (accessed: 15.10.2025).
13. Kosinski M., Forrest A., Holdsworth J. What is MFA (multifactor authentication)? | IBM. IBM. URL: https://www.ibm.com/think/topics/multi-factor-authentication (accessed: 15.10.2025).
14. Papathanasaki M., Maglaras L., Ayres N. Modern authentication methods: a comprehensive survey. AI, computer science and robotics technology. 2022. Vol. 2022. P. 1–24. DOI: 10.5772/acrt.08 (accessed: 15.10.2025).
15. Temoshok D., et. al. Digital Identity Guidelines: Authentication and Authenticator Management. NIST Special Publication 800-63B-4. Gaithersburg (MD): National Institute of Standards and Technology; 2025. DOI:10.6028/NIST.SP.800-63B-4.
16. Password length vs complexity: which is more important?. Keeper Security Blog - Cybersecurity News & Product Updates. URL: https://www.keepersecurity.com/blog/2025/09/16/password-length-vs-complexity-which-is-more-important/ (accessed: 17.10.2025).
17. Frisenna D. F. Passphrases are easier to remember, but are they secure enough?. SySS Tech Blog. URL: https://blog.syss.com/posts/passphrases/ (accessed: 17.10.2025).

18. Psychology of Passwords - LastPass. #1 Password Manager & Vault App with Single-Sign On & MFA Solutions - LastPass. URL: https://www.lastpass.com/resources/reports/psychology-of-passwords (accessed: 17.10.2025).
19. 2025 password manager industry report and statistics. Security.org. URL: https://www.security.org/digital-safety/password-manager-annual-report/ (accessed: 17.10.2025).
20. Reeder R. W., Schechter S. . When the Password Doesn't Work: Secondary Authentication for Websites. *IEEE Security & Privacy*, March/April 2011, 43-49.
21. Alhakami H., ShouqAlhrbi. Knowledge based authentication techniques and challenges. International journal of advanced computer science and applications. 2020. Vol. 11, no. 2. DOI: 10.14569/ijacsa.2020.0110291 (accessed: 19.10.2025).
22. Albayram Y., Khan M. M. H. Evaluating smartphone-based dynamic security questions for fallback authentication: a field study. *Human-centric computing and information sciences*. 2016. Vol. 6, no. 1. DOI: 10.1186/s13673-016-0072-3 (accessed: 19.10.2025).
23. The one-time password (OTP) ultimate guide. Identity Security for the Digital Enterprise | Ping Identity. URL: https://www.pingidentity.com/en/resources/blog/post/one-time-password-ultimate-guide.html (accessed: 19.10.2025).
24. Mayorga O. E. A., Yoo S. G. One time password (OTP) solution for two factor authentication: a practical case study. Journal of computer science. 2025. Vol. 21, no. 5. P. 1099–1112. DOI: 10.3844/jcssp.2025.1099.1112 (accessed: 19.10.2025).
25. Password Generator - LastPass. #1 Password Manager & Vault App with Single-Sign On & MFA Solutions - LastPass. URL: https://www.lastpass.com/features/password-generator (accessed: 19.10.2025).
26. Username & password generator | bitwarden. Bitwarden. URL: https://bitwarden.com/help/generator/ (accessed: 20.10.2025).

27. A secure, strong password generator | 1password. Password Manager & Extended Access Management. URL: https://1password.com/password-generator (accessed: 15.10.2025).
28. What is a ‘secret recovery phrase’, and how to secure your wallet. *Metamask help center*. URL: https://support.metamask.io/start/what-is-a-secret-recovery-phrase-and-how-to-keep-your-crypto-wallet-secure/ (accessed: 20.10.2025).
29. How to use a wallet backup. *Trezor Hardware Wallet (Official)*. . URL: https://trezor.io/guides/backups-recovery/general-standards/how-to-use-a-wallet-backup (accessed: 26.10.2025).
30. Microsoft account recovery code. Microsoft Support. URL: https://support.microsoft.com/en-us/account-billing/microsoft-account-recovery-code-2acc2f88-e37b-4b44-99d4-b4419f610013 (accessed: 26.10.2025).
31. Set up a recovery key for your Apple account. *Apple support*.. URL: https://support.apple.com/en-us/109345 (accessed: 29.10.2025).
32. Top 10 most powerful supercomputers in the World. Interesting Engineering. URL: https://interestingengineering.com/science/10-most-powerful-supercomputers-in-2025 (accessed: 29.10.2025).
33. Козловський В. О., Лесько О. Й., Кавецький В. В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця : ВНТУ, 2021. 42 с.
34. Кавецький В. В., Козловський В. О., Причепа І. В. Економічне обґрунтування інноваційних рішень: практикум / . Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

Додаток А. Протокол перевірки наявності текстових запозичень

Назва роботи: Методи автентифікації користувачів на основі фактору знання
Автор роботи: Клиш Вікторія Миколаївна
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **0, 28 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к.т.н., доцент _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Юрій БАРИШЕВ
Здобувач _____ Вікторія КЛИШ

Додаток Б. Текст програми

password-generator.js

```
const substitutionTable = require("./substitution-table");

class PasswordGenerator {
  constructor() {
    this.config = {
      length: 12,
      includeUppercase: true,
      includeLowercase: true,
      includeNumbers: true,
      includeSymbols: true,
      excludeSimilar: false,
      excludeAmbiguous: false,
    };

    this.substitutionTable = substitutionTable;
  }

  /**
   * @param {Object} options
   * @returns {PasswordGenerator}
   */
  configure(options = {}) {
    this.config = { ...this.config, ...options };

    if (this.config.length < 8) {
      throw new Error("Мінімальна довжина пароля - 8 символів");
    }

    return this;
  }

  /**
   * @param {string} userData
   * @returns {string}
   */
  generate(userData = "") {
    const maxAttempts = 10;

    for (let attempt = 0; attempt < maxAttempts; attempt++) {
      const transformed = this._transformUserData(userData);

      const additionalLength = this.config.length - transformed.length;

      if (additionalLength < 0) {
        throw new Error(
          "Трансформовані дані перевищують цільову довжину пароля"
        );
      }

      const missingCategories = this._getMissingCategories(transformed);
      const additional = this._generateAdditional(
        additionalLength,
        missingCategories
      );

      const password = this._combineComponents(transformed, additional);

      if (this._validatePassword(password)) {
        return password;
      }
    }

    throw new Error(
      "Не вдалося згенерувати пароль, який задовольняє всі обмеження"
```

```
    );
  }

  /**
   * @private
   */
  _transformUserData(userData) {
    if (!userData || userData.length === 0) {
      return "";
    }

    let transformed = "";

    for (const char of userData) {
      if (this.substitutionTable[char]) {
        const options = this.substitutionTable[char];
        const randomIndex = Math.floor(Math.random() * options.length);
        transformed += options[randomIndex];
      } else {
        transformed += char;
      }
    }

    return transformed;
  }

  /**
   * @private
   */
  _getMissingCategories(text) {
    const categories = {
      uppercase: /[A-Z]/.test(text),
      lowercase: /[a-z]/.test(text),
      digit: /[0-9]/.test(text),
      special: /[^a-zA-Z0-9]/.test(text),
    };

    const missing = [];
    if (!categories.uppercase && this.config.includeUppercase)
      missing.push("uppercase");
    if (!categories.lowercase && this.config.includeLowercase)
      missing.push("lowercase");
    if (!categories.digit && this.config.includeNumbers) missing.push("digit");
    if (!categories.special && this.config.includeSymbols)
      missing.push("special");

    return missing;
  }

  /**
   * @private
   */
  _generateAdditional(length, missingCategories) {
    if (length <= 0) return "";

    const lowercase = "abcdefghijklmnopqrstuvwxyz";
    const uppercase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const numbers = "0123456789";
    const symbols = "!@#$%^&*()_+-=[]{}|;:,.<>?";

    const similar = "il1Lo0O";
    const ambiguous = "{}[]()/\\'\"`~";

    let charset = "";
    let result = "";

    if (this.config.includeLowercase) {
      let chars = lowercase;
      if (this.config.excludeSimilar) {
        chars = chars
          .split("")
```

```
      .filter((c) => !similar.includes(c))
      .join("");
  }
  charset += chars;
}

if (this.config.includeUppercase) {
  let chars = uppercase;
  if (this.config.excludeSimilar) {
    chars = chars
      .split("")
      .filter((c) => !similar.includes(c))
      .join("");
  }
  charset += chars;
}

if (this.config.includeNumbers) {
  let chars = numbers;
  if (this.config.excludeSimilar) {
    chars = chars
      .split("")
      .filter((c) => !similar.includes(c))
      .join("");
  }
  charset += chars;
}

if (this.config.includeSymbols) {
  let chars = symbols;
  if (this.config.excludeAmbiguous) {
    chars = chars
      .split("")
      .filter((c) => !ambiguous.includes(c))
      .join("");
  }
  charset += chars;
}

for (const category of missingCategories) {
  if (result.length >= length) break;

  if (category === "uppercase") {
    const chars = this.config.excludeSimilar
      ? uppercase
          .split("")
          .filter((c) => !similar.includes(c))
          .join("")
      : uppercase;
    result += chars[Math.floor(Math.random() * chars.length)];
  } else if (category === "lowercase") {
    const chars = this.config.excludeSimilar
      ? lowercase
          .split("")
          .filter((c) => !similar.includes(c))
          .join("")
      : lowercase;
    result += chars[Math.floor(Math.random() * chars.length)];
  } else if (category === "digit") {
    const chars = this.config.excludeSimilar
      ? numbers
          .split("")
          .filter((c) => !similar.includes(c))
          .join("")
      : numbers;
    result += chars[Math.floor(Math.random() * chars.length)];
  } else if (category === "special") {
    const chars = this.config.excludeAmbiguous
      ? symbols
          .split("")
          .filter((c) => !ambiguous.includes(c))
```

```
              .join("")
            : symbols;
        result += chars[Math.floor(Math.random() * chars.length)];
      }
    }

    for (let i = result.length; i < length; i++) {
      result += charset[Math.floor(Math.random() * charset.length)];
    }

    return result
      .split("")
      .sort(() => Math.random() - 0.5)
      .join("");
  }

  /**
   * @private
   */
  _combineComponents(transformed, additional) {
    if (additional.length === 0) return transformed;
    if (transformed.length === 0) return additional;

    const method = Math.floor(Math.random() * 3);

    if (method === 0) {
      const mid = Math.floor(additional.length / 2);
      const part1 = additional.substring(0, mid);
      const part2 = additional.substring(mid);
      return part1 + transformed + part2;
    } else if (method === 1) {
      return additional + transformed;
    } else {
      return transformed + additional;
    }
  }

  /**
   * @private
   */
  _validatePassword(password) {
    if (password.length !== this.config.length) return false;

    const checks = [];

    if (this.config.includeUppercase) {
      checks.push(/[A-Z]/.test(password));
    }

    if (this.config.includeLowercase) {
      checks.push(/[a-z]/.test(password));
    }

    if (this.config.includeNumbers) {
      checks.push(/[0-9]/.test(password));
    }

    if (this.config.includeSymbols) {
      checks.push(/[^a-zA-Z0-9]/.test(password));
    }

    return checks.every((check) => check);
  }
}

class MnemonicGenerator {
  constructor(dictionaries) {
    if (
      !dictionaries ||
      !dictionaries.nouns ||
      !dictionaries.verbs ||
```

```
    !dictionaries.adjectives
  ) {
    throw new Error(
      "Необхідно передати об’єкт словників з полями: nouns, verbs, adjectives"
    );
  }

  this.dictionaries = dictionaries;
  this.config = {
    separator: "-",
    capitalize: false,
    wordCount: 5,
  };

  this.templates = {
    5: [
      ["Adj", "Noun", "Verb", "Adj", "Noun"],
      ["Adj", "Adj", "Noun", "Verb", "Noun"],
      ["Noun", "Verb", "Adj", "Adj", "Noun"],
    ],
    6: [
      ["Adj", "Noun", "Verb", "Adj", "Adj", "Noun"],
      ["Noun", "Verb", "Adj", "Noun", "Verb", "Noun"],
      ["Adj", "Adj", "Noun", "Verb", "Adj", "Noun"],
    ],
    7: [
      ["Adj", "Noun", "Verb", "Adj", "Noun", "Verb", "Noun"],
      ["Noun", "Verb", "Noun", "Adj", "Noun", "Verb", "Noun"],
      ["Adj", "Adj", "Noun", "Verb", "Adj", "Noun", "Verb"],
    ],
  };
}

/**
 * @param {Object} options
 * @returns {MnemonicGenerator}
 */
configure(options = {}) {
  this.config = { ...this.config, ...options };

  if (this.config.wordCount && ![5, 6, 7].includes(this.config.wordCount)) {
    throw new Error("Кількість слів має бути 5, 6 або 7");
  }

  return this;
}

/**
 * @returns {string}
 */
generate() {
  const wordCount = this.config.wordCount;
  const availableTemplates = this.templates[wordCount];
  const template =
    availableTemplates[Math.floor(Math.random() * availableTemplates.length)];

  const usedWords = new Set();
  const words = [];

  for (let i = 0; i < template.length; i++) {
    const partOfSpeech = template[i];
    let word = null;
    let attempts = 0;
    const maxAttempts = 100;

    while (attempts < maxAttempts) {
      word = this._getRandomWordByType(partOfSpeech);

      if (!usedWords.has(word)) {
        usedWords.add(word);
        words.push(word);
```

```
        break;
      }

      attempts++;
    }

    if (attempts === maxAttempts) {
      words.push(word);
    }
  }

  if (this.config.capitalize) {
    return words
      .map((word) => word.charAt(0).toUpperCase() + word.slice(1))
      .join(this.config.separator);
  }

  return words.join(this.config.separator);
}

/**
 * @param {number} count
 * @returns {Array<string>}
 */
generateMultiple(count = 5) {
  const phrases = [];
  for (let i = 0; i < count; i++) {
    phrases.push(this.generate());
  }
  return phrases;
}

/**
 * @private
 */
_getRandomWordByType(type) {
  let dictionary;

  switch (type) {
    case "Noun":
      dictionary = this.dictionaries.nouns;
      break;
    case "Adj":
      dictionary = this.dictionaries.adjectives;
      break;
    case "Verb":
      dictionary = this.dictionaries.verbs;
      break;
    default:
      throw new Error("Невідомий тип слова: " + type);
  }

  return this._getRandomWord(dictionary);
}

/**
 * @private
 */
_getRandomWord(dictionary) {
  const index = Math.floor(Math.random() * dictionary.length);
  return dictionary[index];
}

/**
 * @returns {Object}
 */
getTemplateInfo() {
  return {
    availableWordCounts: [5, 6, 7],
    currentWordCount: this.config.wordCount,
    templatesPerCount: {
```

```
        5: this.templates[5].length,
        6: this.templates[6].length,
        7: this.templates[7].length,
      },
      currentTemplates: this.templates[this.config.wordCount],
    };
  }
}

if (typeof module !== "undefined" && module.exports) {
  module.exports = { PasswordGenerator, MnemonicGenerator };
}
```

Додаток В. Алфавіт символів та таблиця замін

Таблиця В.1 – Алфавіт

Категорія символів	Символи (повний перелік)	Кількість
Великі літери (A_{upper})	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z	26
Малі літери (A_{lower})	a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z	26
Цифри (A_{digit})	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	10
Спеціальні символи ($A_{special}$)	!, ", #, , %, &, `, (, ), *, +, ,, -, ., /, :, ;, <, =, >, ?, @, [, \, ], ^, _, , {, |, }, ~`	32

Таблиця В.2 – Таблиця замін

Малі літери (lowercase):	Великі літери (uppercase):	Цифри:
a → {@, 4, A}	A → {@, 4, a}	0 → {O, o}
b → {8, B}	B → {8, b}	1 → {I, i, l, |}
c → {(, C, <}	C → {(, c, <}	2 → {Z, z}
d → {D, d,)}	D → {d,)}	3 → {E, e}
e → {3, E}	E → {3, e}	4 → {@, A, a}
f → {F, 5}	F → {f, 5}	5 → {S, s, }
g → {9, G, 6}	G → {9, g}	6 → {b}
h → {#, H}	H → {#, h}	7 → {T, t}
i → {!, 1, I}	I → {!, 1, i}	8 → {B, &}
j → {J, j}	J → {j, ;}	9 → {G, g, Q, q}
k → {K, k}	K → {k, %}	
l → {1, |, L}	L → {1, |, l}	
m → {M, m}	M → {m, ^}	
n → {N, n}	N → {n, ~}	
o → {0, O, *}	O → {0, o}	
p → {P, p}	P → {p, ?}	
q → {9, Q}	Q → {9, q}	
r → {R, r}	R → {r, &}	
s → {$, 5, S}	S → {$, 5, s}	
t → {7, T, +}	T → {7, t}	
u → {U, u}	U → {u, _}	
v → {V, v}	V → {v, \}	
w → {W, w}	W → {w, =}	
x → {X, x}	X → {x, +}	
y → {Y, y}	Y → {y, /}	
z → {2, Z}	Z → {2, z}	

Додаток Г. Словники мнемонічних фраз

Іменники (6796 слів):
armour, barrymore, cabot, catholicism, chihuahua, christianity, easter, frenchman, lowry, mayer, orientalism, pharaoh, pueblo, pullman, rodeo, saturday, sister, snead, syrah, tuesday, woodward, abbey, absence, absorption, abstinence, absurdity, abundance, acceptance, accessibility, accommodation, accomplice, accountability, accounting, accreditation, accuracy, acquiescence, acreage, actress, actuality, adage, adaptation, adherence, adjustment, adoption, adultery, advancement, advert, advertisement, advertising, advice, aesthetics, affinity, aggression, agriculture, aircraft, airtime, allegation, allegiance, allegory, allergy, allies, alligator, allocation, allotment, altercation, ambulance, ammonia, anatomy, anemia, ankle, announcement, annoyance, annuity, anomaly, anthropology, anxiety, apartheid, apologise, apostle, apparatus, appeasement, appellation, appendix, applause, appointment, appraisal, archery, archipelago, architecture, ardor, arrears, arrow, artisan, artistry, ascent, assembly, assignment, association, asthma, atheism, attacker, attraction, attractiveness, auspices, authority, avarice, aversion, aviation, babbling, backlash, baker, ballet, balls, banjo, baron, barrier, barrister, bases, basin, basis, battery, battling, bedtime, beginner, begun, bending, bicycle, billing, bingo, biography, biology, birthplace, blackberry, blather, blossom, boardroom, boasting, bodyguard, boldness, bomber, bondage, bonding, bones, bonus, bookmark, boomer, booty, bounds, bowling, brainstorming, breadth, breaker, brewer, brightness, broccoli, broth, brotherhood, browsing, brunch, brunt, building, bullion, bureaucracy, burglary, buyout, by-election, cabal, cabbage, calamity, campaign, canonization, captaincy, carcass, carrier, cartridge, cassette, catfish, caught, celebrity, cemetery, certainty, certification, charade, chasm, check-in, cheerleader, cheesecake, chemotherapy, chili, china, chivalry, cholera, cilantro, circus, civilisation, civility, clearance, clearing, clerk, climber, closeness, clothing, clutches, coaster, coconut, coding, collaborator, colleague, college, collision, colors, combustion, comedian, comer, commander, commemoration, commenter, commissioner, commune, competition, completeness, complexity, computing, comrade, concur, condominium, conduit, confidant, configuration, confiscation, conflagration, conflict, consist, consistency, consolidation, conspiracy, constable, consul, consultancy, contentment, contents, contractor, conversation, cornerstone, corpus, correlation, councilman, counselor, countdown, countryman, coverage, covering, coyote, cracker, creator, criminality, crocodile, cropping, cross-examination, crossover, crossroads, culprit, cumin, curator, curfew, cursor, custard, cutter, cyclist, cyclone, cylinder, cynicism, daddy, damsel, darkness, dawning, daybreak, dealing, dedication, deduction, defection, deference, deficiency, definition, deflation, degeneration, delegation, delicacy, delirium, deliverance, demeanor, demon, demonstration, denomination, dentist, departure, depletion, depression, designation, despotism, detention, developer, devolution, dexterity, diagnosis, dialect, differentiation, digger, digress, dioxide, diploma, disability, disarmament, discord, discovery, dishonesty, dismissal, disobedience, dispatcher, disservice, distribution, distributor, diver, diversity, docking, dollar, dominance, domination, dominion, donkey, doorstep, doorway, dossier, downside, drafting, drank,

drilling, driver, drumming, drunkenness, duchess, ducking, dugout, dumps, dwelling, dynamics, eagerness, earnestness, earnings, eater, editor, effectiveness, electricity, elements, eloquence, emancipation, embodiment, embroidery, emperor, employment, encampment, enclosure, encouragement, endangerment, enlightenment, enthusiasm, environment, environs, envoy, epilepsy, equation, equator, error, espionage, estimation, evacuation, exaggeration, examination, exclamation, expediency, exploitation, extinction, eyewitness, falls, fascism, fastball, feces, feedback, ferocity, fertilization, fetish, finale, firing, fixing, flashing, flask, flora, fluke, folklore, follower, foothold, footing, forefinger, forefront, forgiveness, formality, formation, formula, foyer, fragmentation, framework, fraud, freestyle, frequency, friendliness, fries, frigate, fulfillment, function, functionality, fundraiser, fusion, futility, gallantry, gallery, genesis, genitals, girlfriend, glamour, glitter, glucose, google, grandeur, grappling, greens, gridlock, grocer, groundwork, grouping, gunman, gusto, habitation, hacker, hallway, hamburger, hammock, handling, hands, handshake, happiness, hardship, headcount, header, headquarters, heads, headset, hearth, hearts, heath, hegemony, height, hello, helper, helping, helplessness, hierarchy, hoarding, hockey, homeland, homer, honesty, horror, horseman, hostility, housing, humility, hurricane, iceberg, ignition, illness, illustration, illustrator, immunity, immunization, imperialism, imprisonment, inaccuracy, inaction, inactivity, inauguration, indecency, indicator, inevitability, infamy, infiltration, influx, iniquity, innocence, innovation, insanity, inspiration, instruction, instructor, insurer, interact, intercession, intercourse, intermission, interpretation, intersection, interval, intolerance, intruder, invasion, investment, involvement, irrigation, iteration, jenny, jogging, jones, joseph, juggernaut, juncture, jurisprudence, juror, kangaroo, kingdom, knocking, laborer, larceny, laurels, layout, leadership, leasing, legislation, leopard, liberation, licence, lifeblood, lifeline, ligament, lighting, likeness, line-up, lineage, liner, lineup, liquidation, listener, literature, litigation, litre, loathing, locality, lodging, logic, longevity, lookout, lordship, lustre, ma’am, machinery, madness, magnificence, mahogany, mailing, mainframe, maintenance, majority, manga, mango, manifesto, mantra, manufacturer, maple, martin, martyrdom, mathematician, matrix, matron, mayhem, mayor, means, meantime, measurement, mechanics, mediator, medics, melodrama, memory, mentality, metaphysics, method, metre, miner, mirth, misconception, misery, mishap, misunderstanding, mobility, molasses, momentum, monarchy, monument, morale, mortality, motto, mouthful, mouthpiece, mover, movie, mowing, murderer, musician, mutation, mythology, narration, narrator, nationality, negligence, neighborhood, neighbour, nervousness, networking, nexus, nightmare, nobility, nobody, noodle, normalcy, notification, nourishment, novella, nucleus, nuisance, nursery, nutrition, nylon, oasis, obscenity, obscurity, observer, offense, onslaught, operation, opportunity, opposition, oracle, orchestra, organisation, organizer, orientation, originality, ounce, outage, outcome, outdoors, outfield, outing, outpost, outset, overseer, owner, oxygen, pairing, panther, paradox, parliament, parsley, parson, passenger, pasta, patchwork, pathos, patriotism, pendulum, penguin, permission, persona, perusal, pessimism, peter, philosopher, phosphorus, phrasing, physique, piles, plateau, playing, plaza, plethora, plurality, pneumonia, pointer, poker, policeman, polling, poster, posterity, posting, postponement, potassium, pottery, poultry, pounding, pragmatism, precedence, precinct,

preoccupation, pretense, priesthood, prisoner, privacy, probation, proceeding, proceedings, processing, processor, progression, projection, prominence, propensity, prophecy, prorogation, prospectus, protein, prototype, providence, provider, provocation, proximity, puberty, publicist, publicity, publisher, pundit, putting, quantity, quart, quilting, quorum, racism, radiance, ralph, rancher, ranger, rapidity, rapport, ratification, rationality, reaction, reader, reassurance, rebirth, receptor, recipe, recognition, recourse, recreation, rector, recurrence, redemption, redistribution, redundancy, refinery, reformer, refrigerator, regularity, regulator, reinforcement, reins, reinstatement, relativism, relaxation, rendition, repayment, repentance, repertoire, repository, republic, reputation, resentment, residency, resignation, restaurant, resurgence, retailer, retention, retirement, reviewer, riches, righteousness, roadblock, robber, rocks, rubbing, runoff, saloon, salvation, sarcasm, saucer, savior, scarcity, scenario, scenery, schism, scholarship, schoolboy, schooner, scissors, scolding, scooter, scouring, scrimmage, scrum, seating, sediment, seduction, seeder, seizure, self-confidence, self-control, self-respect, semicolon, semiconductor, semifinal, senator, sending, serenity, seriousness, servitude, sesame, setup, sewing, sharpness, shaving, shoplifting, shopping, siding, simplicity, simulation, sinking, skate, sloth, slugger, snack, snail, snapshot, snark, soccer, solemnity, solicitation, solitude, somewhere, sophistication, sorcery, souvenir, spaghetti, specification, specimen, specs, spectacle, spectre, speculation, sperm, spoiler, squad, squid, staging, stagnation, staircase, stairway, stamina, standpoint, standstill, stanza, statement, stillness, stimulus, stocks, stole, stoppage, storey, storyteller, stylus, subcommittee, subscription, subsidy, suburb, success, sufferer, supposition, suspension, sweater, sweepstakes, swimmer, syndrome, synopsis, syntax, system, tablespoon, taker, tavern, technology, telephony, template, tempo, tendency, tendon, terrier, terror, terry, theater, theology, therapy, thicket, thoroughfare, threshold, thriller, thunderstorm, ticker, tiger, tights, to-day, tossing, touchdown, tourist, tourney, toxicity, tracing, tractor, translation, transmission, transmitter, trauma, traveler, treadmill, trilogy, trout, tuning, twenties, tycoon, tyrant, ultimatum, underdog, underwear, unhappiness, unification, university, uprising, vaccination, validity, vampire, vanguard, variation, vegetation, verification, viability, vicinity, victory, viewpoint, villa, vindication, violation, vista, vocalist, vogue, volcano, voltage, vomiting, vulnerability, waistcoat, waitress, wardrobe, warmth, watchdog, wealth, weariness, whereabouts, whisky, whiteness, widget, width, windfall, wiring, witchcraft, withholding, womanhood, words, workman, youngster

Дієслова:
accept, add, admire, admit, advise, afford, agree, adjust, alert, align, allow, amuse, analyse, annotate, announce, annoy, answer, apologise, appear, append, applaud, apply, appreciate, approve, argue, arrange, arrest, arrive, ask, assume, attach, attack, attempt, attend, attract, attribute, autocommit, automate, avoid, back, backmerge, bake, balance, ban, bang, bare, bat, bathe, battle, beam, beg, behave, belong, bleach, bless, blind, blink, block, blot, blush, boast, boil, bolt, bomb, book, bore, borrow, bounce, bow, box, brake, branch, break, breathe, bring, bruise, brush, bubble, build, bump, burn, bury, buzz, cache, calculate, call, camp, care, carry, carve, cause, challenge, change, charge, chase, cheat, check, cheer, chew, choke, chop, claim, clap, clean, clear, clip, close, coach, coil,

collect, colour, comb, command, comment, commit, communicate, compare, compete, complain, complete, concentrate, concern, confess, confuse, connect, consider, consist, contain, continue, convert, copy, correct, cough, count, cover, crack, crash, crawl, create, cross, crush, cry, cure, curl, curve, customise, cycle, dam, damage, dance, dare, decay, deceive, decide, decommission, decorate, define, delay, delete, delight, deliver, depend, dereference, describe, desert, deserve, destroy, detect, determine, develop, disable, disagree, disappear, disapprove, disarm, discard, discover, dislike, dismiss, display, divide, document, double, doubt, drag, drain, draw, dream, dress, drip, drop, drown, drum, dry, duplicate, dust, earn, educate, email, embarrass, embed, employ, empty, enable, encode, encourage, end, enforce, enhance, enjoy, ensure, enter, entertain, escape, examine, excite, exclude, excuse, exercise, exist, expand, expect, explain, explode, extend, extract, face, fade, fail, fancy, fasten, fax, fear, fence, fetch, file, fill, film, filter, find, fire, fit, fix, flap, flash, float, flood, flow, flower, fold, follow, fool, force, form, format, found, frame, frighten, fry, gather, gaze, generalise, generate, glow, glue, grab, grate, grease, greet, grin, grip, groan, guarantee, guard, guess, guide, hack, hammer, hand, handle, hang, happen, harass, harm, hash, hate, haunt, head, heal, heap, heat, help, hide, hook, hop, hope, hover, hug, hum, hunt, hurry, identify, ignore, imagine, implement, impress, import, improve, include, increase, influence, inform, inject, injure, inline, instruct, intend, interest, interfere, interrupt, introduce, invent, invert, invite, irritate, itch, jail, jam, jog, join, joke, judge, juggle, jump, kick, kill, kiss, kneel, knit, knock, knot, label, land, last, laugh, launch, learn, level, license, lick, lie, lighten, like, list, listen, live, load, lock, log, long, look, love, make, man, manage, march, mark, marry, match, mate, matter, measure, meddle, melt, memorise, mend, merge, mess up, milk, mine, miss, mix, moan, modify, moor, mourn, move, muddle, mug, multiply, murder, nail, name, need, nest, nod, note, notice, number, obey, object, observe, obtain, occur, offend, offer, open, optimise, order, overflow, owe, own, pack, paddle, paint, park, parse, part, pass, paste, pat, pause, peck, pedal, peel, peep, perform, permit, phone, pick, pinch, pine, place, plan, plant, play, please, plug, point, poke, polish, pop, possess, post, pour, practise, pray, preach, precalculate, precede, prefer, prepare, present, preserve, press, pretend, prevent, prick, print, produce, program, promise, protect, provide, publish, pull, pump, punch, puncture, punish, pull, push, put, query, question, queue, race, radiate, rain, raise, reach, read, realise, receive, recognise, record, rebuild, redirect, reduce, re-enable, refactor, reference, reflect, refuse, regret, reign, reject, rejoice, relax, release, rely, remain, remember, remind, remove, rename, re-order, repair, repeat, replace, reply, report, reproduce, request, require, rescue, resequence, resign, resolve, restore, restrict, resurrect, retain, retire, retrieve, return, reveal, reverse, revert, reword, rewrite, rhyme, rinse, risk, rob, rock, roll, rot, rub, ruin, rule, run, rush, sack, sail, satisfy, save, saw, scare, scatter, schedule, scold, scorch, scrape, scratch, scream, screw, scribble, scrub, seal, search, send, separate, serve, settle, shade, share, shave, shelter, shiver, shock, shop, show, shrug, sigh, sign, signal, simplify, sin, sip, ski, skip, slap, slip, slow, smash, smell, smile, smoke, snatch, sneeze, sniff, snore, snow, soak, soothe, sound, spare, spark, sparkle, specify, speed, spell, spill, spoil, spot, spray, sprout, squash, squeak, squeal, squeeze, stage, stain, stamp, standardise, stare, start, stay, steer, step, stir, stitch, stop, store, strap, strengthen, stretch, strip, stroke, stuff, subtract, succeed, suck, suffer, suggest, suit, supply, support,

suppose, surprise, surround, suspect, suspend, switch, sync, talk, tame, tap, target, taste, tease, telephone, tempt, terrify, test, thank, thaw, tick, tickle, tie, time, tip, tire, touch, tour, tow, trace, trade, train, transport, trap, travel, treat, tremble, trick, trip, trot, trouble, truncate, trust, try, tug, tumble, turn, twist, type, undress, unfasten, unite, unlock, unpack, untidy, update, upgrade, upload, use, validate, vanish, void, visit, wail, wait, walk, wander, want, warm, warn, wash, waste, watch, water, wave, weigh, welcome, whine, whip, whirl, whisper, whistle, wink, wipe, wish, wobble, wonder, work, worry, wrap, wreck, wrestle, wriggle, x-ray, yawn, yell, zip, zoom

Прикметники:
aristotelian, arthurian, bohemian, brethren, mosaic, oceanic, proctor, terran, tudor, abroad, absorbing, abstract, academic, accelerated, accented, accountant, acquainted, acute, addicting, addictive, adjustable, admired, adult, adverse, advised, aerosol, afraid, aggravated, aggressive, agreeable, alienate, aligned, all-round, alleged, almond, alright, altruistic, ambient, ambivalent, amiable, amino, amorphous, amused, anatomical, ancestral, angelic, angrier, answerable, antiquarian, antiretroviral, appellate, applicable, apportioned, approachable, appropriated, archer, aroused, arrested, assertive, assigned, athletic, atrocious, attained, authoritarian, autobiographical, avaricious, avocado, awake, awesome, backstage, backwoods, balding, bandaged, banded, banned, barreled, battle, beaten, begotten, beguiled, bellied, belted, beneficent, besieged, betting, big-money, biggest, biochemical, bipolar, blackened, blame, blessed, blindfolded, bloat, blocked, blooded, blue-collar, blushing, boastful, bolder, bolstered, bonnie, bored, boundary, bounded, bounding, branched, brawling, brazen, breeding, bridged, brimming, brimstone, broadest, broiled, broker, bronze, bruising, buffy, bullied, bungling, burial, buttery, candied, canonical, cantankerous, cardinal, carefree, caretaker, casual, cathartic, causal, chapel, characterized, charcoal, cheeky, cherished, chipotle, chirping, chivalrous, circumstantial, civic, civil, civilised, clanking, clapping, claptrap, classless, cleansed, cleric, cloistered, codified, colloquial, colour, combat, combined, comely, commissioned, commonplace, commuter, commuting, comparable, complementary, compromising, conceding, concentrated, conceptual, conditioned, confederate, confident, confidential, confining, confuse, congressional, consequential, conservative, constituent, contaminated, contemporaneous, contraceptive, convertible, convex, cooked, coronary, corporatist, correlated, corroborated, cosmic, cover, crash, crypto, culminate, cushioned, dandy, dashing, dazzled, decreased, decrepit, dedicated, defaced, defective, defenseless, deluded, deodorant, departed, depress, designing, despairing, destitute, detective, determined, devastating, deviant, devilish, devoted, diagonal, dictated, didactic, differentiated, diffused, dirtier, disabling, disconnected, discovered, disdainful, diseased, disfigured, disheartened, disheveled, disillusioned, disparate, dissident, doable, doctrinal, doing, dotted, double-blind, downbeat, dozen, draining, draught, dread, dried, dropped, dulled, duplicate, eaten, echoing, economical, elaborated, elastic, elective, electoral, elven, embryo, emerald, emergency, emissary, emotional, employed, enamel, encased, encrusted, endangered, engraved, engrossing, enlarged, enlisted, enlivened, ensconced, entangled, enthralling, entire, envious, eradicated, eroded, esoteric, essential, evaporated, ever-present, evergreen, everlasting, exacting, exasperated, excess, exciting, executable, existent, exonerated, exorbitant,

exponential, export, extraordinary, exultant, exulting, facsimile, fading, fainter, faith-based, fallacious, faltering, famous, fancier, fast-growing, fated, favourable, fearless, feathered, fellow, fermented, ferocious, fiddling, filling, firmer, fitted, flammable, flawed, fledgling, fleshy, flexible, flickering, floral, flowering, flowing, foggy, folic, foolhardy, foolish, footy, forehand, forked, formative, formulaic, foul-mouthed, fractional, fragrant, fraudulent, freakish, freckled, freelance, freight, fresh, fretted, frugal, fulfilling, fuming, funded, funny, garbled, gathered, geologic, geometric, gibberish, gilded, ginger, glare, glaring, gleaming, glorified, glorious, goalless, gold-plated, goody, grammatical, grande, grateful, gratuitous, graven, greener, grinding, grizzly, groaning, grudging, guaranteed, gusty, half-breed, hand-held, handheld, hands-off, hard-pressed, harlot, healing, healthier, healthiest, heart, heart-shaped, heathen, hedonistic, heralded, herbal, high-density, high-performance, high-res, high-yield, hissy, hitless, holiness, homesick, honorable, hooded, hopeless, horrendous, horrible, hot-button, huddled, human, humbling, humid, humiliating, hypnotized, idealistic, idiosyncratic, ignited, illustrated, illustrative, imitated, immense, immersive, immigrant, immoral, impassive, impressionable, improbable, impulsive, in-between, in-flight, inattentive, inbound, inbounds, incalculable, incomprehensible, indefatigable, indigo, indiscriminate, indomitable, inert, inflate, inform, inheriting, injured, injurious, inking, inoffensive, insane, insensible, insidious, insincere, insistent, insolent, insufferable, intemperate, interdependent, interesting, interfering, intern, interpreted, intersecting, intolerable, intolerant, intuitive, irresolute, irritate, jealous, jerking, joining, joint, journalistic, joyful, keyed, knowing, lacklustre, laden, lagging, lamented, laughable, layered, leather, leathern, leery, left-footed, legible, leisure, lessening, liberating, life-size, lifted, lightest, limitless, listening, literary, liver, livid, lobster, locked, long-held, long-lasting, long-running, long-suffering, loudest, loveliest, low-budget, low-carb, lowering, lucid, luckless, lusty, luxurious, magazine, maniac, manmade, maroon, mastered, mated, material, materialistic, meaningful, measuring, mediaeval, medical, meditated, medley, melodic, memorable, memorial, metabolic, metallic, metallurgical, metering, midair, midterm, midway, mighty, migrating, mind-blowing, mind-boggling, minor, mirrored, misguided, misshapen, mitigated, mixed, modernized, molecular, monarch, monastic, morbid, motley, motorized, mounted, multi-million, multidisciplinary, muscled, muscular, muted, mysterious, mythic, nail-biting, natural, nauseous, negative, networked, neurological, neutered, newest, night, nitrous, no-fly, noncommercial, nonsense, north, nuanced, occurring, offensive, oldest, oncoming, one-eyed, one-year, onstage, onward, opaque, open-ended, operating, opportunist, opposing, opt-in, ordinate, outdone, outlaw, outsized, overboard, overheated, oversize, overworked, oyster, paced, panting, paralyzed, paramount, parental, parted, partisan, passive, pastel, patriot, peacekeeping, pedestrian, peevish, penal, penned, pensive, perceptual, perky, permissible, pernicious, perpetuate, perplexed, pervasive, petrochemical, philosophical, picturesque, pillaged, piped, piquant, pitching, plausible, pliable, plumb, politician, polygamous, poorest, portmanteau, posed, positive, possible, postpartum, prank, pre-emptive, precocious, predicted, premium, preparatory, prerequisite, prescient, preserved, presidential, pressed, pressurized, presumed, prewar, priced, pricier, primal, primer, primetime, printed, private, problem, procedural, process, prodigious, professional, programmed, progressive, prolific, promising,

promulgated, pronged, proportionate, protracted, pulled, pulsed, purgatory, quick, rapid-fire, raunchy, razed, reactive, readable, realizing, recognised, recovering, recurrent, recycled, redeemable, reflecting, regal, registering, reliable, reminiscent, remorseless, removable, renewable, repeating, repellent, reserve, resigned, respectful, rested, restrict, resultant, retaliatory, retiring, revelatory, reverend, reversing, revolving, ridiculous, right-hand, ringed, risque, robust, roomful, rotating, roused, rubber, run-down, running, runtime, rustling, safest, salient, sanctioned, saute, saved, scandalized, scarlet, scattering, sceptical, scheming, scoundrel, scratched, scratchy, scrolled, seated, second-best, segregated, self-taught, semiautomatic, senior, sensed, sentient, sexier, shadowy, shaken, shaker, shameless, shaped, shiny, shipped, shivering, shoestring, short, short-lived, signed, simplest, simplistic, sizable, skeleton, skinny, skirting, skyrocketed, slamming, slanting, slapstick, sleek, sleepless, sleepy, slender, slimmer, smacking, smokeless, smothered, smouldering, snuff, socialized, solid-state, sometime, sought, spanking, sparing, spattered, specialized, specific, speedy, spherical, spiky, spineless, sprung, squint, stainless, standing, starlight, startled, stately, statewide, stereoscopic, sticky, stimulant, stinky, stoked, stolen, storied, strained, strapping, strengthened, stubborn, stylized, suave, subjective, subjugated, subordinate, succeeding, suffering, summary, sunset, sunshine, supernatural, supervisory, supply-side, surrogate, suspended, suspenseful, swarthy, sweating, sweeping, swinging, swooning, sympathize, synchronized, synonymous, synthetic, tailed, tallest, tangible, tanked, tarry, technical, tectonic, telepathic, tenderest, territorial, testimonial, theistic, thicker, threatening, tight-lipped, timed, timely, timid, torrent, totalled, tougher, traditional, transformed, trapped, traveled, traverse, treated, trial, trunk, trusting, trying, twisted, two-lane, tyrannical, unaided, unassisted, unassuming, unattractive, uncapped, uncomfortable, uncontrolled, uncooked, uncooperative, underground, undersea, undisturbed, unearthly, uneasy, unequal, unfazed, unfinished, unforeseen, unforgivable, unidentified, unimaginative, uninspired, unintended, uninvited, universal, unmasked, unorthodox, unparalleled, unpleasant, unprincipled, unread, unreasonable, unregulated, unreliable, unremitting, unsafe, unsanitary, unsealed, unsuccessful, unsupervised, untimely, unwary, unwrapped, uppity, upstart, useless, utter, valiant, valid, valued, vanilla, vaulting, vaunted, veering, vegetative, vented, verbal, verifying, veritable, versed, vinyl, virgin, visceral, visual, voluptuous, walk-on, wanton, warlike, washed, waterproof, waved, weakest, well-bred, well-chosen, well-informed, wetting, wheeled, whirlwind, widen, widening, willful, willing, winnable, winningest, wireless, wistful, woeful, wooded, woodland, wordless, workable, worldly, worldwide, worst-case, worsted, worthless

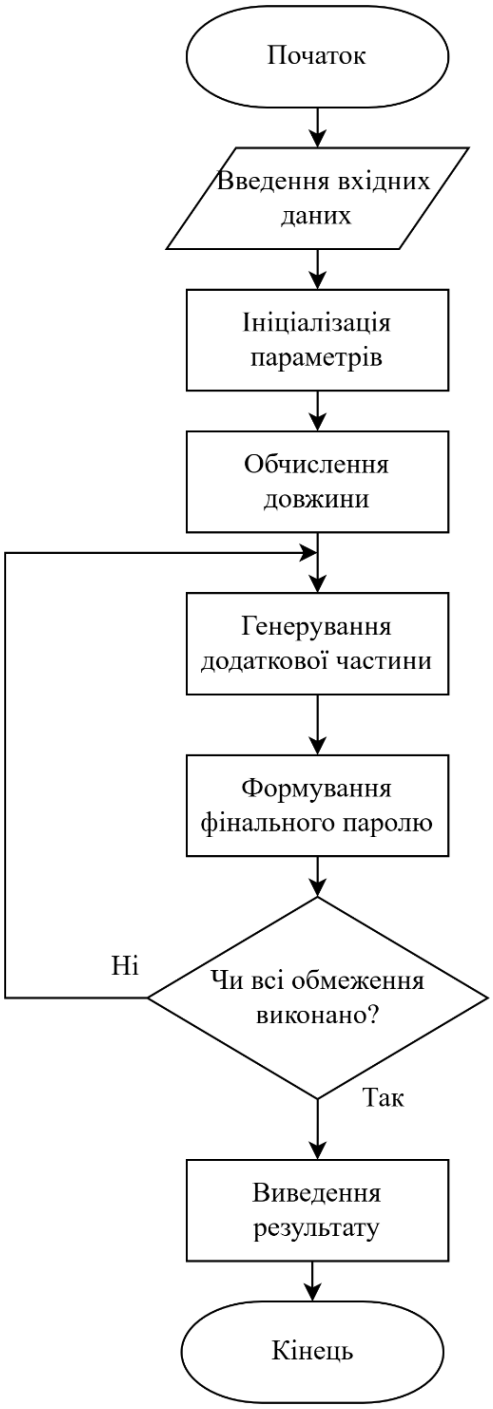
ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ ФАКТОРУ ЗНАННЯ

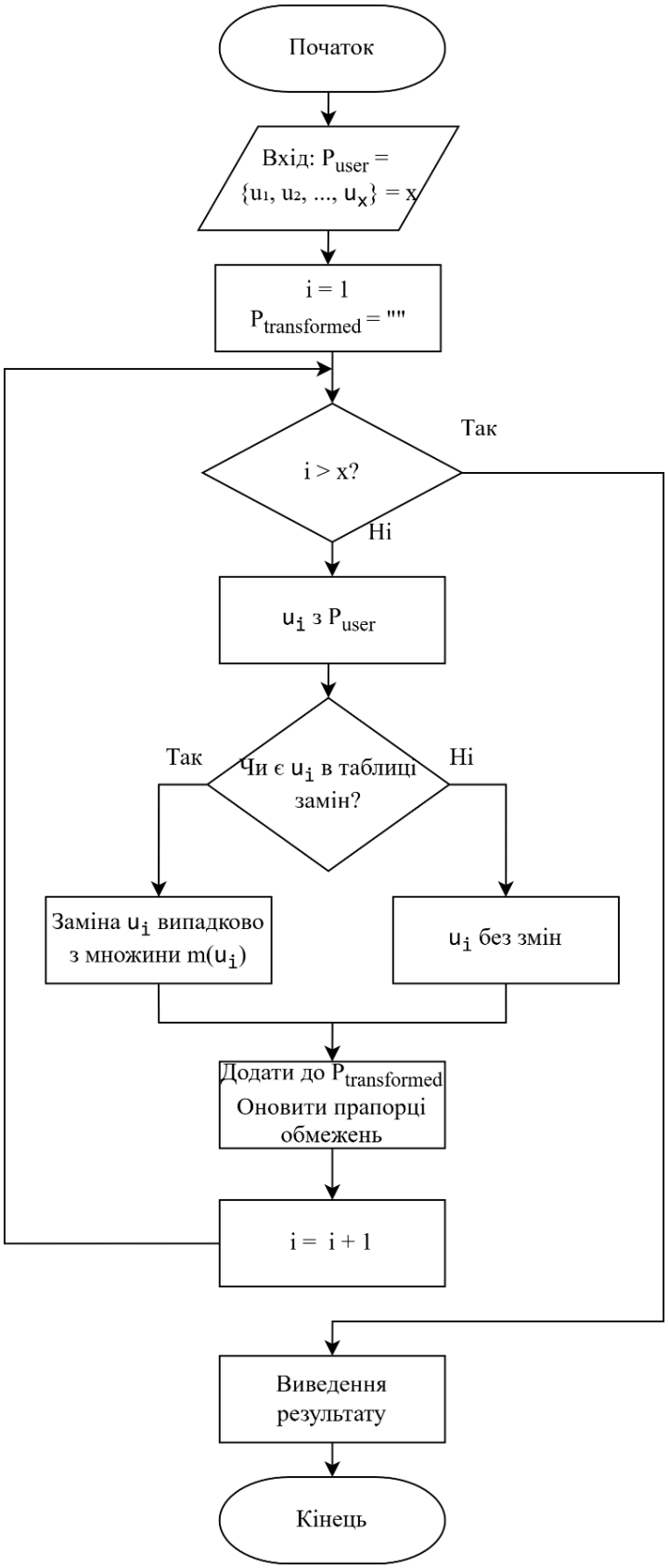
Виконала: студентка 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека та захист
інформації
_____ Вікторія КЛИШ
16.12 2025 р.

Керівник: к. т. н., доцент каф. ЗІ
_____ Юрій БАРИШЕВ
16.12 2025 р.

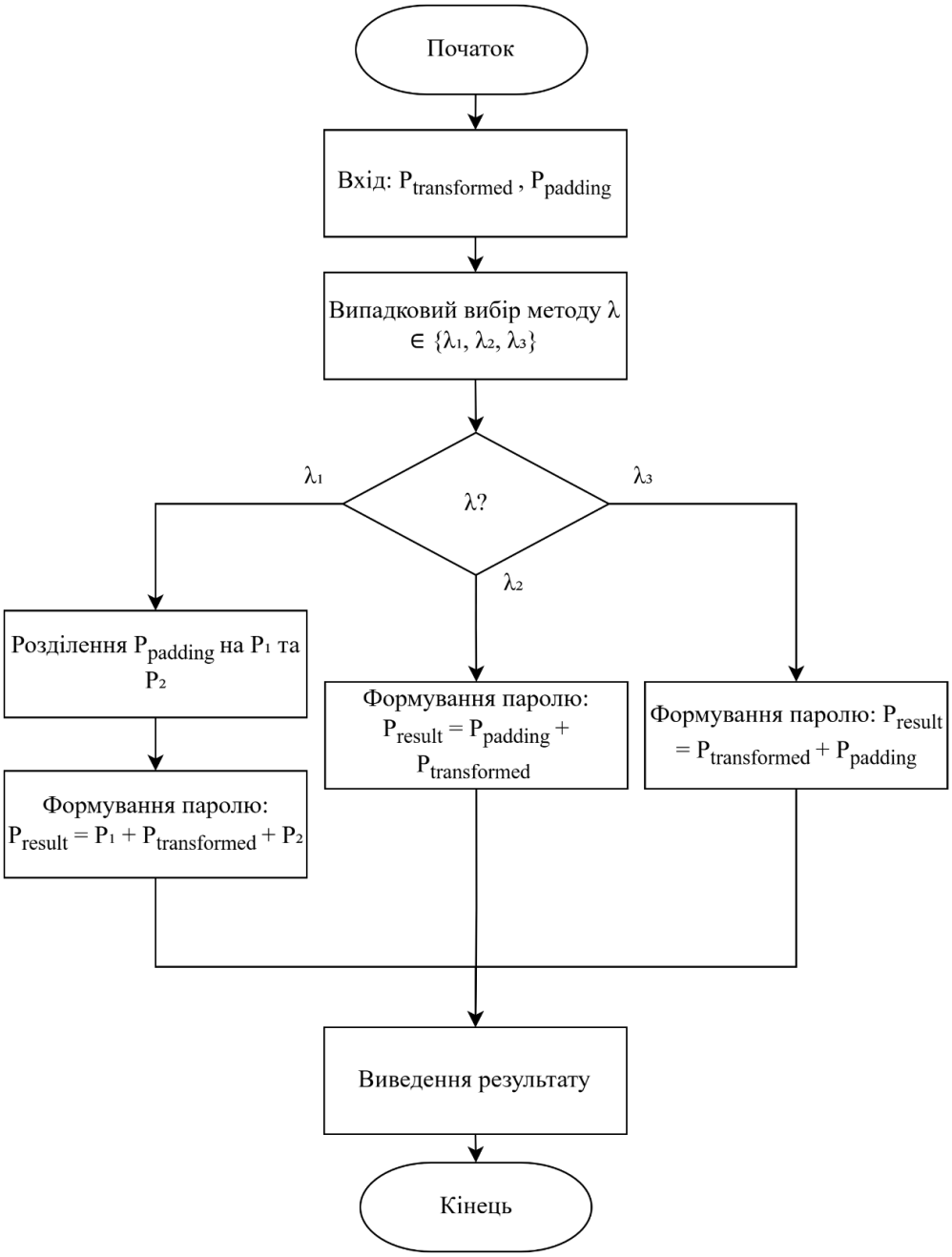
Узагальнений алгоритм формування паролю



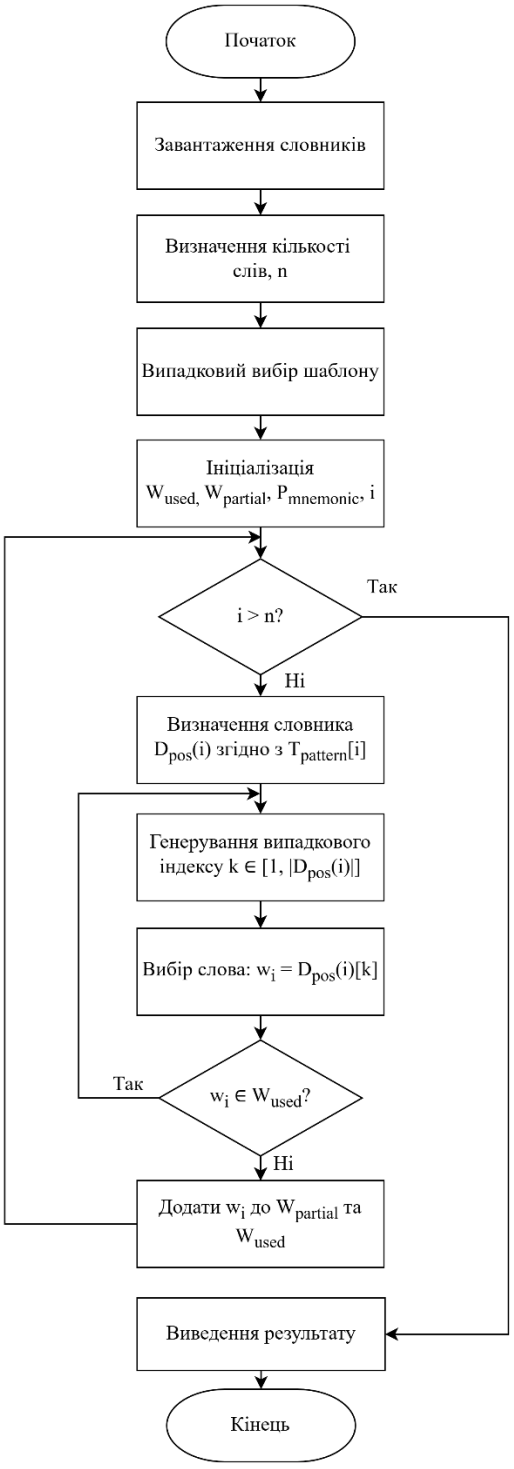
Алгоритм модифікування вхідних даних



Алгоритм комбінування вхідних даних з додатковою частиною



Алгоритм формування мнемонічної фрази



Фрагмент таблиці замін символів

substitution-table.js > substitutionTable

```
const substitutionTable = {
  a: ["@", "4", "A"],
  b: ["8", "B"],
  c: ["<", "C", "("],
  d: ["D", "d", ")"],
  e: ["3", "E"],
  f: ["F", "5"],
  g: ["9", "G", "6"],
  h: ["#", "H"],
  i: ["!", "1", "I"],
  j: ["J", "j"],
```

Приклад візуалізації генерування зрозумілого паролю

```
1) Трансформовані користувацькі дані: C@+
2) Відсутні категорії символів: [ 'lowercase', 'digit' ]
3) Додаткова частина: 5@tt7Rjye
4) Комбінування частин паролю: C@+5@tt7Rjye
5) Пароль відповідає всім правилам? true
Фінальний пароль: C@+5@tt7Rjye
```

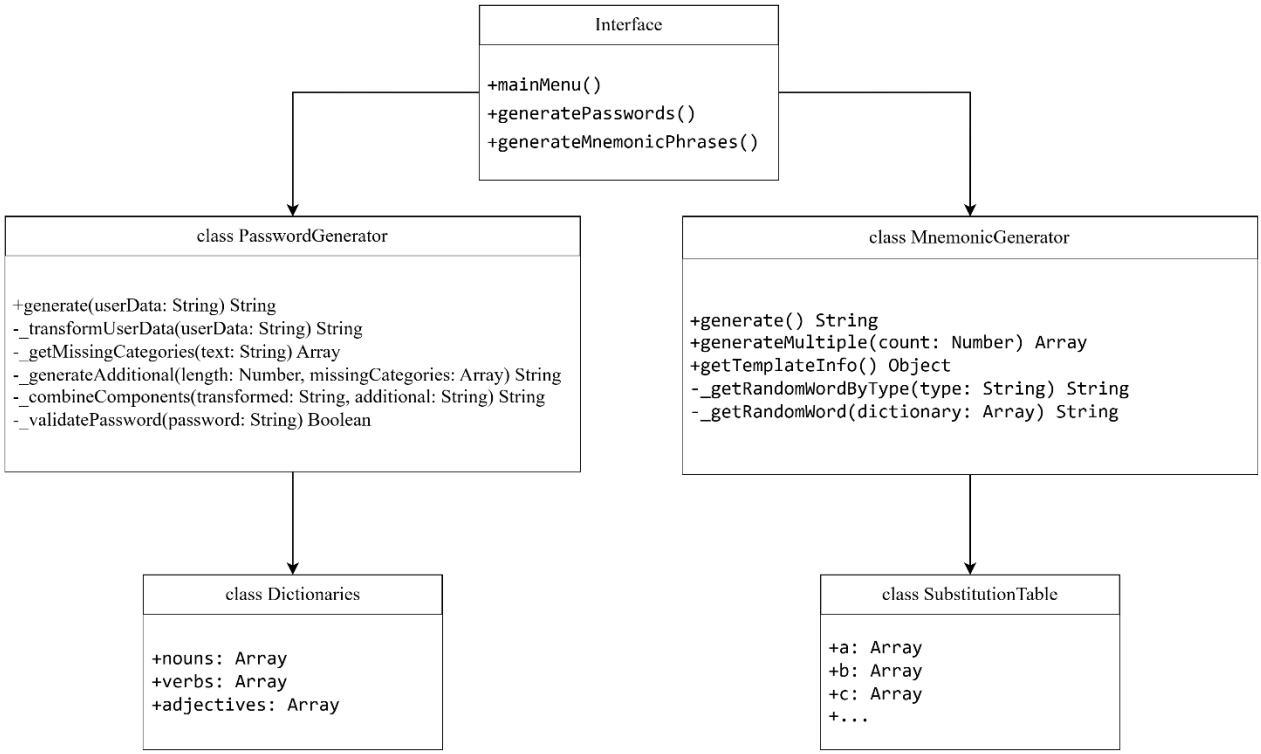
Структура словників для генерування мнемонічних фраз

```
const DICTIONARIES = {

  nouns: [ ···
  ],

  verbs: [ ···
  ],

  adjectives: [ ···
  ],
};
```

Генерування мнемонічної фрази: від вибору кількості слів до готового пароля

```
Оберіть кількість слів (5, 6 або 7, Enter – випадково): 6

=== Згенеровані фрази ===
1: garden-wake-quick-game-cook-sparrow
2: noisy-tiger-decide-free-kind-roof
3: handsome-mouse-paint-full-tasty-leaf
4: deep-woman-buy-polite-ancient-cow
5: water-enjoy-friendly-rat-cut-bear
Слів у кожній фразі: 6
```

Діаграма класів програмної бібліотеки



Приклад меню

```
--- ГОЛОВНЕ МЕНЮ ---
1 - Реєстрація нового користувача
2 - Вхід в систему
3 - Вихід
```

Приклад успішної реєстрації користувача з мнемонічною фразою

```
Оберіть кількість слів (5, 6 або 7, Enter – випадково): 5

--- Згенеровані мнемонічні фрази ---
1. inattentive-wardrobe-suffer-inform-illness
2. reviewer-knit-starlight-reversing-potassium
3. procedural-confidential-lifeline-transport-manga
4. insidious-raunchy-defection-appear-miner
5. commune-tap-exonerated-besieged-syndrome
(Кількість слів: 5)

Оберіть номер фрази (1-5): 1

Ваша фраза: inattentive-wardrobe-suffer-inform-illness
ВАЖЛИВО: Збережіть цю фразу у надійному місці!

Користувача успішно зареєстровано
Користувач: John
Тип паролю: mnemonic
```

Приклад успішної автентифікації з мнемонічною фразою

```
Введіть ім'я користувача: John

Тип паролю: Мнемонічна фраза
Введіть пароль: inattentive-wardrobe-suffer-inform-illness

Успішна автентифікація

   АВТЕНТИФІКАЦІЯ УСПІШНА! ✓
```

Приклад невдалої автентифікації користувача

```
Введіть ім'я користувача: John

Тип паролю: Мнемонічна фраза
Введіть пароль: inattentive-wardrobe-suffer-inform

Невірне ім'я користувача або пароль

   АВТЕНТИФІКАЦІЯ НЕВДАЛА! X
```

Результати виконання тестів для класу PasswordGenerator

```
PasswordGenerator - тестування
  Конфігурація
    √ За замовчуванням довжина 12 символів (6 ms)
    √ Можливість зміни довжини (2 ms)
    √ Помилка при довжині менше 8 (46 ms)
    √ Налаштування типів символів (2 ms)
  Трансформація символів (через публічний API)
    √ Трансформація через generate з "a" (2 ms)
    √ Трансформація через generate з "cat" (1 ms)
    √ Порожній вхід генерує повний пароль (1 ms)
    √ Спеціальні символи обробляються (2 ms)
  Генерування паролів
    √ Генерування паролю без вхідних даних (2 ms)
    √ Генерування паролю з вхідними даними "cat" (1 ms)
    √ Пароль містить всі категорії символів (2 ms)
    √ Генерування різних паролів при повторних викликах (2 ms)
    √ Довжина паролю відповідає налаштуванням (1 ms)
  Валідація паролів (через генерацію)
    √ Згенерований пароль проходить внутрішню валідацію (1 ms)
    √ Пароль без великих літер (коли вимкнено) (1 ms)
    √ Довжина завжди коректна (3 ms)
```

Результати виконання тестів для класу MnemonicGenerator

```
MnemonicGenerator - тестування
  Конфігурація
    √ За замовчуванням 5 слів (2 ms)
    √ Можливість зміни кількості слів (1 ms)
    √ Помилка при некоректній кількості слів (3 ms)
    √ Налаштування роздільника
  Шаблони
    √ Наявність шаблонів для 5, 6, 7 слів (1 ms)
    √ Кількість шаблонів (1 ms)
    √ Структура шаблону містить правильні типи (2 ms)
  Генерування фраз
    √ Генерування фрази з 5 словами (1 ms)
    √ Генерування фрази з 6 словами
    √ Генерування фрази з 7 словами
    √ Слова розділені дефісом (1 ms)
    √ Всі слова з словників (3 ms)
    √ Унікальність слів у фразі
    √ Генерування різних фраз при повторних викликах (2 ms)
  Генерування кількох фраз
    √ Генерування 5 фраз (1 ms)
    √ Всі згенеровані фрази унікальні
  Капіталізація
    √ Без капіталізації - малі літери (1 ms)
Словники - тестування
  √ Словники не порожні (1 ms)
  √ Всі слова - рядки (3 ms)
  √ Слова містять тільки літери (4 ms)
```

Загальний вивід результатів тестування

```
Test Suites: 1 passed, 1 total
Tests:       36 passed, 36 total
Snapshots:   0 total
Time:        1.484 s, estimated 2 s
Ran all test suites.
```

Результати експериментального дослідження генерування паролів

```
ЗАГАЛЬНА СТАТИСТИКА:
   Згенеровано паролів: 3000
   Унікальних паролів: 3000 (100.00%)
   Повторень: 0
   Середня довжина: 12.00 символів

ЧАС ВИКОНАННЯ:
   Загальний час: 46 мс
   Середній час: 0.02 мс/пароль
   Мінімальний час: 0 мс
   Максимальний час: 2 мс

РОЗПОДІЛ СИМВОЛІВ (середнє на пароль):
   Великі літери: 3.40
   Малі літери: 3.12
   Цифри: 1.93
   Спеціальні символи: 3.55

ПЕРЕВІРКА ВИМОГ:
   Паролі з усіма категоріями: 3000/3000 (100.00%)
   Паролі правильної довжини: 3000/3000 (100%)

ПРИКЛАДИ ЗГЕНЕРОВАНИХ ПАРОЛІВ:
   1. (@7w8HhJbDdT
   2. JHNb<47?5]Sv
```

Результати експериментального дослідження генерування мнемонічних фраз

```
═══ ФРАЗИ З 6 СЛІВ ═══

ЗАГАЛЬНА СТАТИСТИКА:
   Згенеровано фраз: 3000
   Унікальних фраз: 3000 (100.00%)
   Повторень: 0
   Фраз без повторення слів: 2999 (99.97%)
   Середня довжина: 33.82 символів

ЧАС ВИКОНАННЯ:
   Загальний час: 22 мс
   Середній час: 0.01 мс/фразу
   Мінімальний час: 0 мс
   Максимальний час: 1 мс

ПРИКЛАДИ:
   1. serious-elephant-check-modern-dirty-game
   2. noisy-wallet-follow-thick-cute-bottle
   3. pillow-laugh-friendly-apartment-pull-road
   4. worse-weak-bed-teach-lazy-knife
   5. sparrow-hear-lively-bowl-open-child
```