МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проект

здобувача освіти денної форми навчання
КБ.02.24.000.ДП

ШУКАЛА
МИРОСЛАВА ЄВГЕНОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну

Проектний матеріал складається з пояснювальної записки на 72 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Шукало М.Є.)

Керівник _____ (Гаджиєв М.М.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «27» червня 2025 р. Протокол ЕК № 6

Оцінка ЕК 4 (добре) / 85 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Шукалу Мирославу Євгеновичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну.*

затверджена наказом по коледжу від "14" *листопада* 2024р. № *246*

2. Термін здачі закінченого проекту *16 червня 2025р.*

3. Вихідні данні до проекту

1. Використовувати кросплатформну мову програмування Python;

2. Впровадити користувальницький графічний інтерфейс засобами Tkinter;

2. Визначити стратегію (алгоритм) аутентифікації;

3. Реалізувати додаткові методи аутентифікації паролем та HASH-256;

4. Реалізувати безпеку даних користувачів засобами AES.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналіз предметної області; 2. Технології та засоби розробки; 3. Проєктування стратегії захисту та графічного інтерфейсу; 4. Розробка захищеного кросплатформного застосунку; 5. Тестування створеного захищеного кросплатформного застосунку; 6. Економічний розрахунок; 7. Аспекти охорони праці та техніки безпеки.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Титульна сторінка; Перелік обраних технологій застосунку; Архітектура підмодулів застосунку; UI/ UX мапа користування застосунком; Стратегія реєстрації користувача; Стратегія авторизації користувача; Формула перевірки патерну; Фрагмент коду логіну; Графічний інтерфейс; Хешування паролю SHA-256; Шифрування даних застосунку AES; Авторизація (шлях до програми).

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Гаджиєв М.М.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 12.05.2025

Керівник *Гаджиєв М.М.* _____
(підпис)

Завдання прийняв до виконання *Шукало М.Є.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Формування вступу*	15.05.25	виконано
2	*Аналіз предметної області*	16.05.25	виконано
3	*Підбір технічної літератури*	19.05.25	виконано
4	*Проєктування стратегії аутентифікації*	21.05.25	виконано
5	*Проєктування графічного інтерфейсу*	23.05.25	виконано
6	*Реалізація безпечного застосунку*	26.05.25	виконано
7	*Тестування розробленого застосунку*	28.05.25	виконано
8	*Оформлення пояснювальної записки*	30.05.25	виконано
9	*Оформлення графічної (презентаційної) частини*	06.06.25	виконано
10	*Економічний розрахунок*	09.06.25	виконано
11	*Опис охорони праці та техніки безпеки*	12.06.25	виконано
12	*Аналіз результатів розробки*	13.06.25	виконано
13	*Підготовка доповіді для захисту*	16.06.25	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 24 000. 00 ДП	Дипломний проект		
А4			КБ 02. 24 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 24 000. 00 ДП		
Розробив		Шукало М.Є.			Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну	Літ.	Аркуш
Перевірив		Гаджиєв М.М.				Н Д П	4 / 72
Н. контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Аутентифікація користувача є ключовим компонентом сучасних інформаційних систем, що забезпечує контроль доступу до ресурсів і захищає їх від несанкціонованого використання. Існуючі методи аутентифікації, зокрема паролі та біометричні дані, мають свої обмеження та вразливості, що створює потребу у пошуку нових, більш надійних підходів. Перспективним напрямом розвитку таких систем є використання поведінкових патернів користувача, що дозволяє реалізувати індивідуалізовані та складні механізми верифікації [1].

Метою цієї дипломної роботи є проектування та розробка програмного модулю аутентифікації, який базується на аналізі поведінкових особливостей користувача. Запропоноване рішення поєднуватиме традиційні методи перевірки, такі як аутентифікація паролем та застосування хеш-функції SHA-256, з алгоритмами поведінкового аналізу. Такий підхід дозволить підвищити надійність системи та знизити ризики компрометації облікових записів.

Для реалізації проекту планується використання мови програмування Python, що завдяки кросплатформеності та широкому набору бібліотек є оптимальним вибором для розробки адаптивних додатків. Користувальницький інтерфейс буде створений з використанням стандартного фреймворку Tkinter, що забезпечить простоту та зручність взаємодії користувача із системою аутентифікації.

Окрему увагу буде приділено забезпеченню конфіденційності та цілісності даних користувачів. Для цього передбачено впровадження механізму симетричного шифрування AES, що гарантує надійний захист інформації під час зберігання і передачі.

Очікується, що реалізація запропонованого модуля дозволить створити ефективний і безпечний інструмент аутентифікації, який можна буде інтегрувати у різноманітні програмні продукти. Використання поведінкових патернів у поєднанні з класичними методами захисту відкриває перспективи для подальшого розвитку інтелектуальних систем безпеки.

.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Дослідження предметної області

1.1.1 Поведінкова аутентифікація

Поведінкова аутентифікація являє собою метод ідентифікації та підтвердження особи користувача на основі аналізу його унікальних моделей поведінки при взаємодії з цифровими пристроями або сервісами. На відміну від традиційних методів, таких як паролі, PIN-коди або біометричні дані (відбитки пальців, розпізнавання обличчя), поведінкова аутентифікація фокусується на тому, як користувач діє, а не на тому, що він знає або ким він є фізично [1][2][3].

Основною ідеєю поведінкової аутентифікації є те, що кожна людина має характерні поведінкові шаблони, які важко відтворити або підробити. До таких шаблонів належать: динаміка натискання клавіш на клавіатурі (keystroke dynamics), стиль використання миші або сенсорного екрану, ритм ходи, манера прокручування сторінки, реакції на інтерфейс, навігаційні патерни тощо.

Системи поведінкової аутентифікації зазвичай працюють у двох режимах:

1. Початкова (разова) аутентифікація: здійснюється під час входу в систему. Наприклад, аналізується ритм набору пароля для підтвердження, що його вводить саме власник облікового запису.
2. Безперервна аутентифікація: відбувається у фоновому режимі під час взаємодії користувача з системою. Якщо поведінковий шаблон змінюється, система може вимагати повторну автентифікацію або обмежити доступ.

Переваги поведінкової аутентифікації включають:

1. Непомітність: користувач не повинен виконувати додаткові дії, аутентифікація відбувається пасивно.
2. Високий рівень захисту: комбінація численних поведінкових ознак ускладнює їхнє підроблення.
3. Зручність використання: зменшується потреба у запам’ятовуванні паролів чи використанні додаткових пристроїв.

Однак, разом з перевагами існують і певні виклики:

1. Зміна поведінки користувача (наприклад, внаслідок хвороби, стресу або використання іншого пристрою) може призвести до помилкової відмови у доступі.
2. Необхідність збору великих обсягів даних для навчання моделей машинного навчання, що підвищує ризики щодо конфіденційності.
3. Висока складність реалізації: потрібно враховувати численні фактори та мінімізувати хибнопозитивні/хибнонегативні результати.

Сучасні системи безпеки все частіше інтегрують поведінкову аутентифікацію як частину багатофакторної аутентифікації (MFA), де вона виступає додатковим рівнем захисту поряд із традиційними методами. Це дозволяє суттєво підвищити стійкість системи до атак типу "викрадення облікових даних", "фішинг" або "соціальна інженерія" [1][2][3].

У контексті розвитку кібербезпеки поведінкова аутентифікація розглядається як перспективний напрям, здатний забезпечити баланс між зручністю користувача та рівнем захисту даних. Її подальше впровадження та вдосконалення пов’язане із розвитком штучного інтелекту, обчислювальних потужностей і підвищенням обізнаності користувачів щодо питань безпеки.

1.1.2 Програмне забезпечення, що використовує поведінкову аутентифікацію

У сучасних інформаційних системах поведінкова аутентифікація знаходить дедалі ширше застосування, що зумовлено необхідністю підвищення рівня захищеності без впливу на зручність користування. На ринку представлено низку програмних рішень, які реалізують поведінкову аутентифікацію, використовуючи різні типи поведінкових біометричних ознак, такі як динаміка набору тексту, шаблони руху миші, навігаційна поведінка, жести тощо. У цьому розділі представлено аналіз та порівняння найбільш відомих програмних продуктів у цій сфері.

BehavioSec.

Шведська компанія BehavioSec є одним із лідерів у галузі поведінкової біометрії. Її платформа BehavioSense дозволяє в режимі реального часу аналізувати

поведінкові патерни користувача при взаємодії з пристроями. Рішення інтегрується з системами багатофакторної аутентифікації та забезпечує безперервний контроль сесії користувача без необхідності додаткових дій з його боку. Платформа орієнтована насамперед на фінансовий сектор, що пояснюється високими вимогами до безпеки [4].

На рисунку 1.1 зображено сервіс BehavioSec.

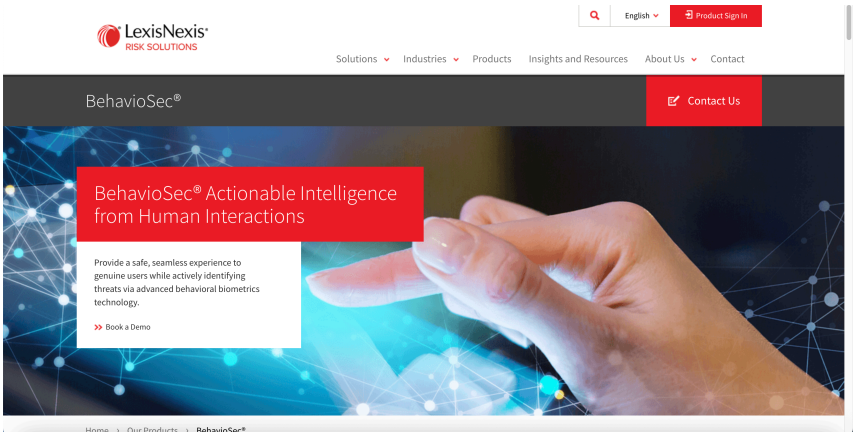


Рисунок 1.1. Сервіс BehavioSec

TypingDNA.

Програмне забезпечення TypingDNA фокусується на аналізі динаміки набору тексту як основного засобу ідентифікації. Особливістю рішення є легкість інтеграції через API та можливість використання як у веб-додатках, так і на мобільних платформах. TypingDNA активно використовується у сфері онлайн-освіти, банківських послуг і корпоративної безпеки, оскільки дозволяє підтверджувати особу користувача без додаткового обладнання [5].

На рисунку 1.2 зображено сервіс TypingDNA.

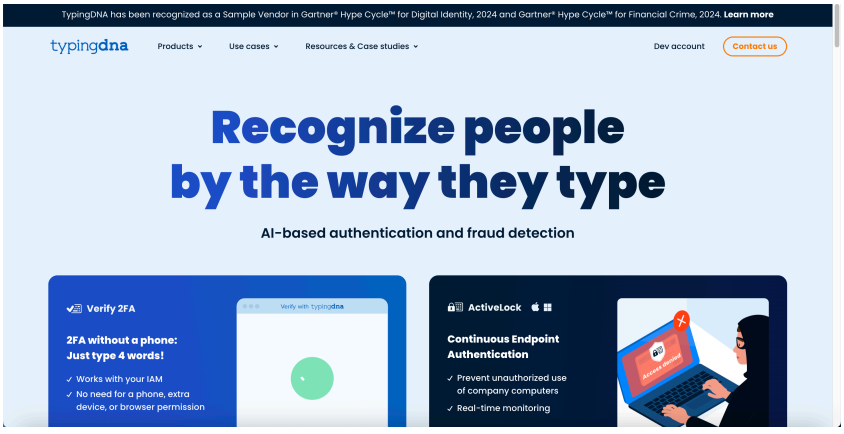


Рисунок 1.2. Сервіс TypingDNA

Zighra SensifyID.

Платформа Zighra використовує так званий "наслідуваний інтелект" (implicit AI), аналізуючи широкий спектр мікровзаємодій з пристроєм — від сили натискання до швидкості реакції. Система формує унікальний профіль користувача й виявляє навіть незначні відхилення в поведінці. Рішення придатне як для мобільних, так і для вбудованих систем, і може працювати автономно на пристрої користувача без потреби у хмарній обробці [6].

На рисунку 1.3 зображено сервіс Zighra SensifyID.

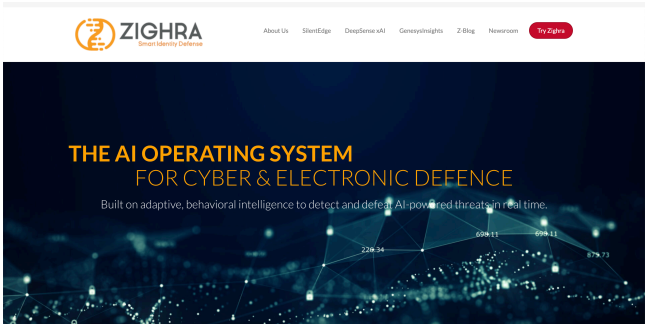


Рисунок 1.3. Сервіс TypingDNA

BioCatch.

Компанія BioCatch спеціалізується на виявленні шахрайства через аналіз когнітивних та поведінкових ознак під час користування веб-сайтами. Система фіксує понад 200 параметрів взаємодії користувача з браузером, зокрема характерні затримки, рухи курсора, ритм прокрутки тощо. BioCatch знаходить широке застосування у банківській сфері, особливо в контексті захисту онлайн-транзакцій [7].

На рисунку 1.4 зображено сервіс BioCatch.

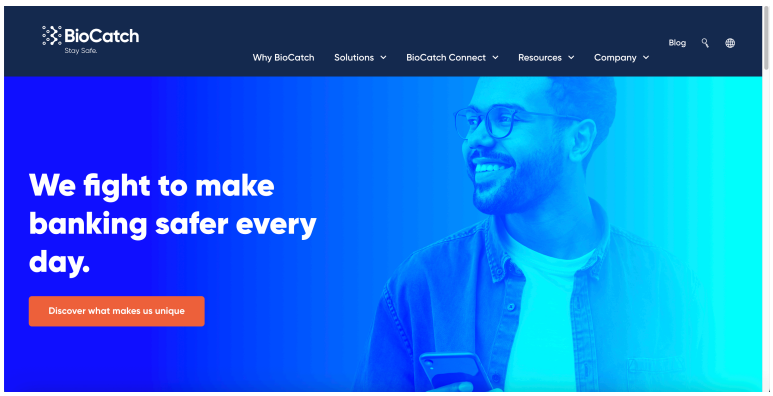


Рисунок 1.4. Сервіс BioCatch

Plurilock.

Рішення Plurilock забезпечує безперервну аутентифікацію в корпоративних середовищах. Програмне забезпечення аналізує поведінкові патерни в контексті використання клавіатури, миші та інших інтерфейсних пристроїв. Особливістю Plurilock є можливість інтеграції з системами управління доступом і розгортання в середовищах підвищеної безпеки, наприклад, у держустановах [8].

На рисунку 1.5 зображено сервіс Plurilock.

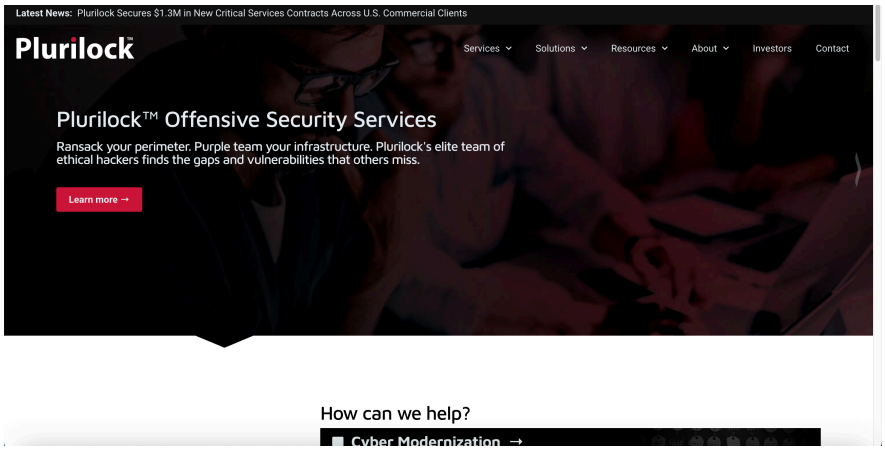


Рисунок 1.5. Сервіс Plurilock

Узагальнюючи, слід зазначити, що існуючі програмні рішення з поведінкової аутентифікації відрізняються як за глибиною аналізу, так і за сферами застосування. Вибір конкретного продукту залежить від потреб організації, вимог до конфіденційності даних, доступного середовища для розгортання, а також прийнятного рівня інтрузивності для користувача. Незважаючи на відмінності, всі згадані рішення демонструють високий потенціал поведінкової біометрії як засобу посилення кібербезпеки.

1.2 Проєктування захищенного застосунку

1.2.1 Запланована функціональність застосунку

У сучасних інформаційних системах питання надійної та зручної авторизації користувачів є одним із ключових аспектів забезпечення безпеки. Розглянутий застосунок реалізує унікальний підхід до ідентифікації користувача на основі аналізу індивідуального стилю набору тексту, що суттєво відрізняє його від традиційних методів аутентифікації, таких як введення пароля або використання

біометричних даних.

Основною функціональністю застосунку є:

1. Реєстрація користувача шляхом введення послідовності слів із фіксацією часових інтервалів набору (таймінгу) для формування унікального шаблону стилю введення.
2. Авторизація користувача на основі порівняння фактичного стилю введення з раніше збереженим профілем, що дозволяє підвищити рівень безпеки за рахунок поведінкової біометрії.
3. Наявність резервного пароля для випадків, коли стиль введення не співпадає із збереженим профілем, що забезпечує додаткову гнучкість у процесі доступу.
4. Можливість налаштування та запуску цільового додатку або команди після успішної авторизації.
5. Режим адміністратора, який дозволяє додавати нових користувачів, змінювати налаштування запуску програми та керувати списком зареєстрованих користувачів.
6. Захист конфігураційних даних за допомогою симетричного шифрування (бібліотека Fernet), що гарантує цілісність та конфіденційність інформації.

Порівняно з існуючими аналогами, які, як правило, базуються на стандартних методах аутентифікації (логін/пароль, двофакторна аутентифікація, біометрія за відбитками або обличчям), запропонований застосунок вирізняється наступними перевагами:

1. Впровадження поведінкової біометрії, що складніше піддається підробці або компрометації, ніж традиційні паролі.
2. Відсутність необхідності у спеціальному обладнанні (наприклад, сканерах відбитків або камері для розпізнавання обличчя).
3. Простота інтеграції з будь-якими програмами шляхом налаштування команди запуску.
4. Гнучкість у керуванні користувачами через адміністративний інтерфейс.
5. Недоліками можуть бути:

6. Залежність від якості введення та потенційна чутливість до змін у поведінці користувача (наприклад, через втому або зміну пристрою введення).
7. Обмежена кількість слів для формування унікального шаблону, що може знижувати точність розпізнавання при великій кількості користувачів.

Отже, запропонований застосунок має збалансовану функціональність, яка поєднує інноваційний метод аутентифікації з практичністю використання та високим рівнем захисту даних. Такий підхід дозволяє ефективно використовувати його у різних сферах, де необхідна надійна і водночас зручна авторизація користувачів.

1.2.2 Стратегія захисту застосунку

Захист інформаційних систем та застосунків є одним із найважливіших аспектів сучасної розробки програмного забезпечення. У контексті розгляду захисту даного застосунку доцільно проаналізувати основні підходи та механізми автентифікації і авторизації користувачів, а також порівняти їх із існуючими аналогами.

У представленому застосунку реалізована багаторівнева система захисту, яка базується на поєднанні біометричних характеристик поведінки користувача (стиль введення тексту) з класичними методами перевірки (пароль). Такий підхід дозволяє підвищити надійність автентифікації, знижуючи ризик несанкціонованого доступу.

Основні складові стратегії захисту в застосунку:

1. Поведенчий біометричний контроль (повільність введення трьох слів): Застосунок фіксує час набору кожного з трьох слів та порівнює цю інформацію з профілем користувача. Допустима похибка встановлена на рівні 0.3 секунди, що забезпечує баланс між точністю та зручністю.
2. Захист резервним паролем: У разі невідповідності стилю введення або за бажанням користувача передбачена можливість входу за допомогою заздалегідь встановленого резервного пароля, захищеного хешуванням SHA-256.
3. Шифрування конфігураційних даних: Вся інформація про користувачів та

налаштування зберігається у зашифрованому вигляді за допомогою симетричного шифрування (модуль cryptography.fernet). Це унеможливлює несанкціоноване прочитання даних конфігурації навіть при доступі до файлів.

4. Адміністративний режим: Для зміни налаштувань застосунку (реєстрація нових користувачів, налаштування шляху запуску програми, керування користувачами) необхідно активувати режим адміністратора, що додатково захищено паролем, який генерується динамічно (хеш від поточної дати).

У таблиці 1.1. зведено порівняння методів захисту.

Таблиця 1.1. Порівння методів захисту

Метод захисту	**Переваги**	**Недоліки**
Парольна автентифікація	Простота реалізації, широке застосування	Вразливість до підбору паролів, фішингу, повторного використання
Біометрична автентифікація (відбитки, розпізнавання обличчя)	Висока точність, складно підробити	Висока вартість впровадження, питання приватності
Поведенчий біометричний захист	Додатковий рівень безпеки, без необхідності додаткового обладнання	Може залежати від фізичного стану користувача, складність у налаштуванні та адаптації
Двофакторна автентифікація	Значно підвищує рівень безпеки	Необхідність додаткових пристроїв або додатків
Шифрування конфігураційних файлів	Захищає дані від несанкціонованого доступу при компрометації	Потребує безпечного зберігання ключів, додаткові обчислювальні ресурси

Застосування поведенчого біометричного захисту у поєднанні з резервним паролем і симетричним шифруванням конфігурації є оптимальним варіантом для середовища, де важлива баланс між зручністю використання і рівнем безпеки. Даний підхід має перевагу над традиційною парольною системою, оскільки додає унікальний елемент поведінкової автентифікації без необхідності залучення додаткового обладнання, що відрізняє його від деяких біометричних аналогів.

Однак слід врахувати, що подібна система може мати обмеження, пов’язані

з варіативністю поведінки користувача (наприклад, зміни швидкості набору під час стресу або втоми), а також можливість відновлення доступу через резервний пароль, який повинен зберігатися та передаватися користувачем із високою відповідальністю.

У підсумку, стратегія захисту застосунку, реалізована у представленому програмному продукті, забезпечує комплексний підхід до безпеки за рахунок інтеграції поведінкової автентифікації, резервних механізмів і шифрування даних, що відповідає сучасним вимогам до інформаційної безпеки в сегменті настільних застосунків.

На рисунку 1.6 зображено стратегію реєстрації користувача.

Рисунок 1.6. Стратегія реєстрації

На рисунку 1.7 зображено стратегію авторизації користувача.

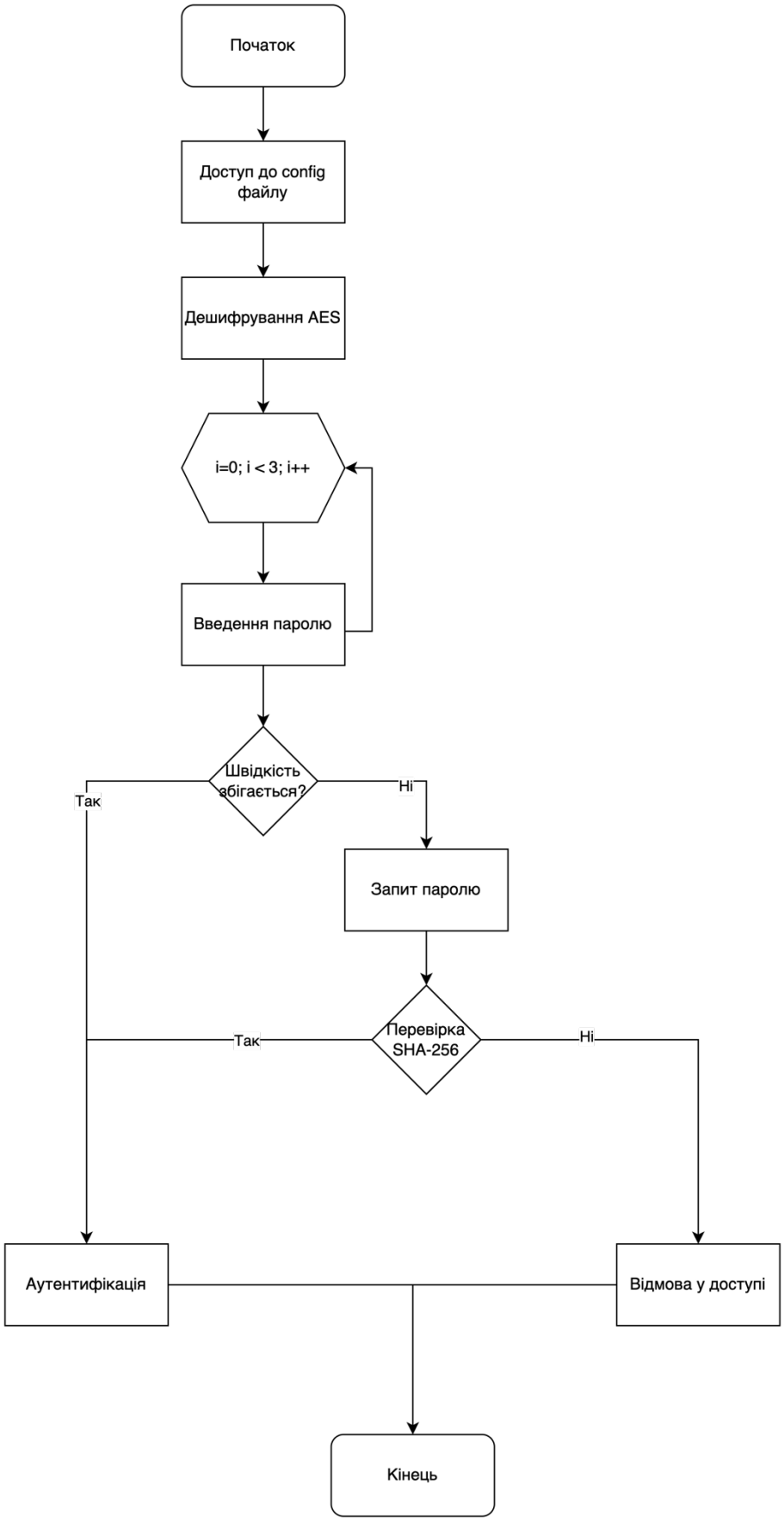


Рисунок 1.7. Стратегія авторизації

1.2.3 Обгрунтування обраних технологій для застосунку

Розробка застосунку для авторизації за стилем введення передбачала вибір технологій, які забезпечують надійність, простоту інтеграції та безпеку користувацьких даних. Основним інструментом реалізації обрано мову програмування Python з графічною бібліотекою Tkinter, що зумовлено кількома ключовими факторами.

По-перше, Python є високорівневою мовою з широкими можливостями для роботи з обробкою даних, криптографією та файловою системою, що дозволяє ефективно реалізувати алгоритми аналізу часу набору тексту (таймінги) та хешування паролів. Бібліотека Tkinter надає простий у використанні засіб для створення кросплатформеного графічного інтерфейсу, що є важливим для забезпечення зручності користувача [9].

По-друге, для забезпечення безпеки збереження конфігураційних даних застосовано шифрування з використанням бібліотеки cryptography (модуль Fernet), що гарантує конфіденційність та цілісність інформації про користувачів та параметри запуску програми. Використання ключа шифрування, який генерується один раз і зберігається у файлі, дозволяє уникнути несанкціонованого доступу до критичних даних.

По-третє, вибір механізму авторизації на основі аналізу патернів набору тексту реалізований через збір та порівняння часу введення трьох слів користувачем. Такий підхід дозволяє додатково захистити систему від несанкціонованого доступу, поєднуючи біометричні характеристики стилю введення з резервною аутентифікацією за паролем.

Також обрана технологія дозволяє легко адаптувати команду запуску цільової програми, що підвищує гнучкість застосунку. Застосунок підтримує кросплатформений запуск програм як у Windows, так і в інших операційних системах, що досягається через динамічне визначення команди запуску.

Таким чином, інтеграція Python, Tkinter, криптографічних методів шифрування та механізмів аналізу поведінкових характеристик користувача створює надійний, безпечний та зручний у використанні застосунок, що відповідає

сучасним вимогам інформаційної безпеки.

На рисунку 1.8 зображено обрані технології.

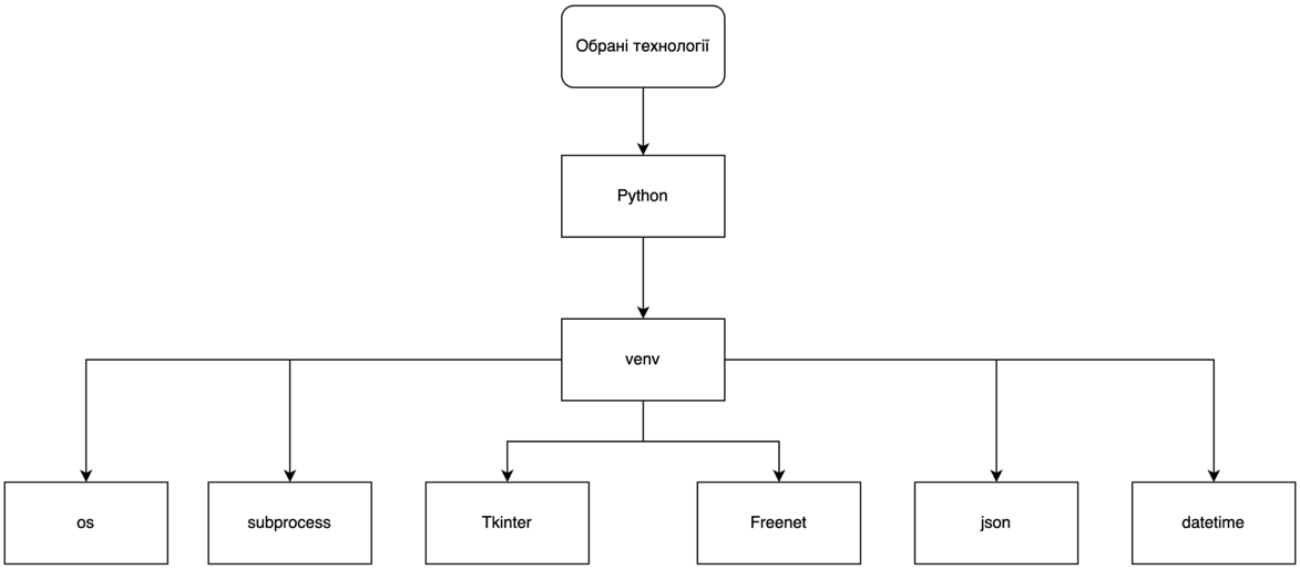


Рисунок 1.8. Обрані технології

1.2.4 Проєктування програмної архітектури застосунку

Проєктування програмної архітектури застосунку є ключовим етапом розробки, що визначає структуру системи, взаємодію її компонентів та забезпечує виконання основних функціональних і нефункціональних вимог. Архітектура розглянутого застосунку побудована на принципах модульності, безпеки, простоти розширення та підтримки.

Застосунок реалізує механізм авторизації користувачів на основі аналізу стилю введення, що підвищує рівень безпеки доступу до захищеної програми. Для цього використано бібліотеку tkinter для побудови графічного інтерфейсу, модуль cryptography для захисту конфігураційних даних, а також вбудовані бібліотеки hashlib та json для обробки даних користувачів.

Архітектура застосунку включає наступні основні компоненти:

1. Інтерфейс користувача (UI) — реалізований за допомогою tkinter, забезпечує взаємодію з користувачем через діалогові вікна для введення даних, повідомлення про стан операцій та керування адміністративними функціями.
2. Модуль безпеки — забезпечує шифрування і розшифрування конфігураційного файлу з даними користувачів, реалізує функції хешування

паролів та контроль доступу через механізм адміністративного режиму.

3. Модуль авторизації — збирає та аналізує тимчасові характеристики набору слів користувачем, порівнює їх із збереженим шаблоном для підтвердження автентичності.
4. Модуль управління конфігурацією — відповідає за збереження та завантаження даних у зашифрованому вигляді, зберігання команди запуску захищеної програми.
5. Адміністративний модуль — дозволяє реєструвати нових користувачів, керувати їх даними та задавати параметри запуску цільової програми.

Взаємодія між компонентами організована таким чином, щоб забезпечити чітку ізоляцію логіки інтерфейсу від логіки безпеки і обробки даних. Це підвищує надійність системи та спрощує внесення змін і додавання нових функцій.

Застосунок підтримує двоступеневу авторизацію: спочатку здійснюється перевірка унікального патерну набору слів користувача, а у випадку невідповідності стилю — пропонується резервний пароль. Такий підхід забезпечує баланс між зручністю користування і безпекою.

Для захисту конфігураційних файлів використовується симетричне шифрування з ключем, який зберігається у окремому файлі. Це гарантує, що навіть у разі доступу до файлу з даними, без ключа відновити інформацію буде неможливо.

Використання бібліотеки subprocess дозволяє безпосередньо запускати захищену програму після успішної авторизації, що інтегрує механізм захисту безпосередньо у робочий процес користувача.

Таким чином, архітектура застосунку побудована на основі сучасних практик безпеки, з урахуванням зручності користування та можливості масштабування. Вона забезпечує ефективний контроль доступу, надійність збереження даних і гнучкість у керуванні користувачами та параметрами системи.

Окрім цього, структура застосунку дозволяє легко інтегрувати додаткові механізми автентифікації або розширити функціонал без порушення вже існуючої логіки взаємодії між модулями.

На рисунку 1.9 зображено заплановані програмні модулі.

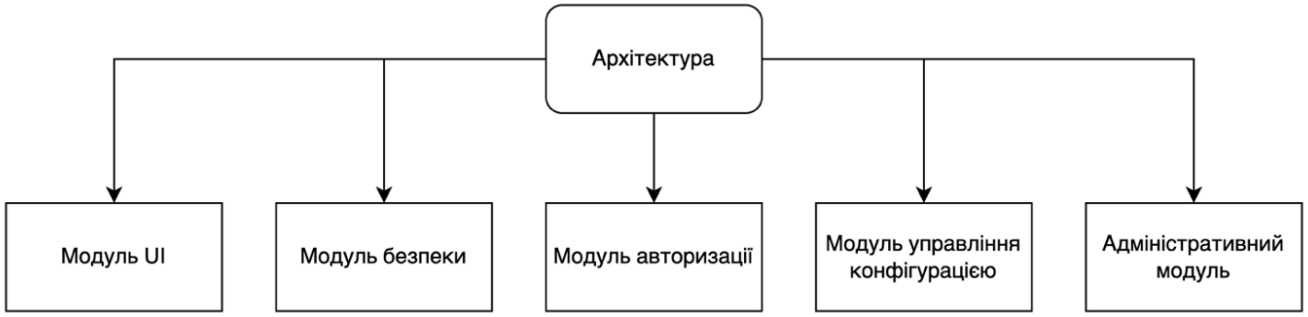


Рисунок 1.9. Програмні модулі

1.2.5 Проєктування дизайну графічного інтерфейсу застосунку

Проєктування графічного інтерфейсу користувача (GUI) є одним із ключових етапів розробки програмного забезпечення, оскільки від нього залежить зручність, інтуїтивність та ефективність взаємодії користувача із застосунком. У розглянутому проєкті дизайн інтерфейсу був створений із урахуванням основних принципів ергономіки, мінімалізму та безпеки користувацьких даних.

Для реалізації інтерфейсу застосовано бібліотеку Tkinter, що є стандартним інструментом для створення віконних додатків на мові Python. Вибір Tkinter зумовлений його простотою, кросплатформністю та можливістю швидкого прототипування. Графічний інтерфейс складається з головного вікна, яке містить кнопки для авторизації користувача, переходу в режим адміністратора, а також функції керування користувачами та налаштування шляху запуску цільової програми.

Структура інтерфейсу передбачає логічне групування елементів управління, що забезпечує інтуїтивну навігацію. Головне вікно містить основні функції: увійти до системи, увімкнути режим адміністратора, зареєструвати нового користувача, задати шлях до запускаємого застосунку, керувати списком користувачів і вийти з програми. Важливою особливістю є блокування адміністративних кнопок за замовчуванням, які активуються лише після успішної автентифікації адміністратора, що підвищує безпеку системи.

Візуальний стиль інтерфейсу орієнтований на мінімалізм та лаконічність. Кожна кнопка має чітку текстову та символічну індикацію свого призначення

(наприклад, іконки замка, людини, шестерні), що покращує сприйняття інформації. Вікна для введення даних (діалогові вікна) реалізовані з використанням простих та зрозумілих форм, що зменшує ймовірність помилок користувача.

Інтерфейс також забезпечує зворотний зв'язок з користувачем через інформаційні, попереджувальні та помилкові повідомлення, реалізовані за допомогою стандартних діалогових вікон Tkinter. Такий підхід сприяє прозорості роботи застосунку та полегшує взаємодію.

Таким чином, проєктування графічного інтерфейсу застосунку базується на принципах простоти, безпеки та зручності, що дозволяє ефективно реалізувати механізми авторизації за стилем введення та адміністрування користувачів у рамках обраної платформи.

На рисунку 1.10 зображено макет графічного інтерфейсу головного екрану.

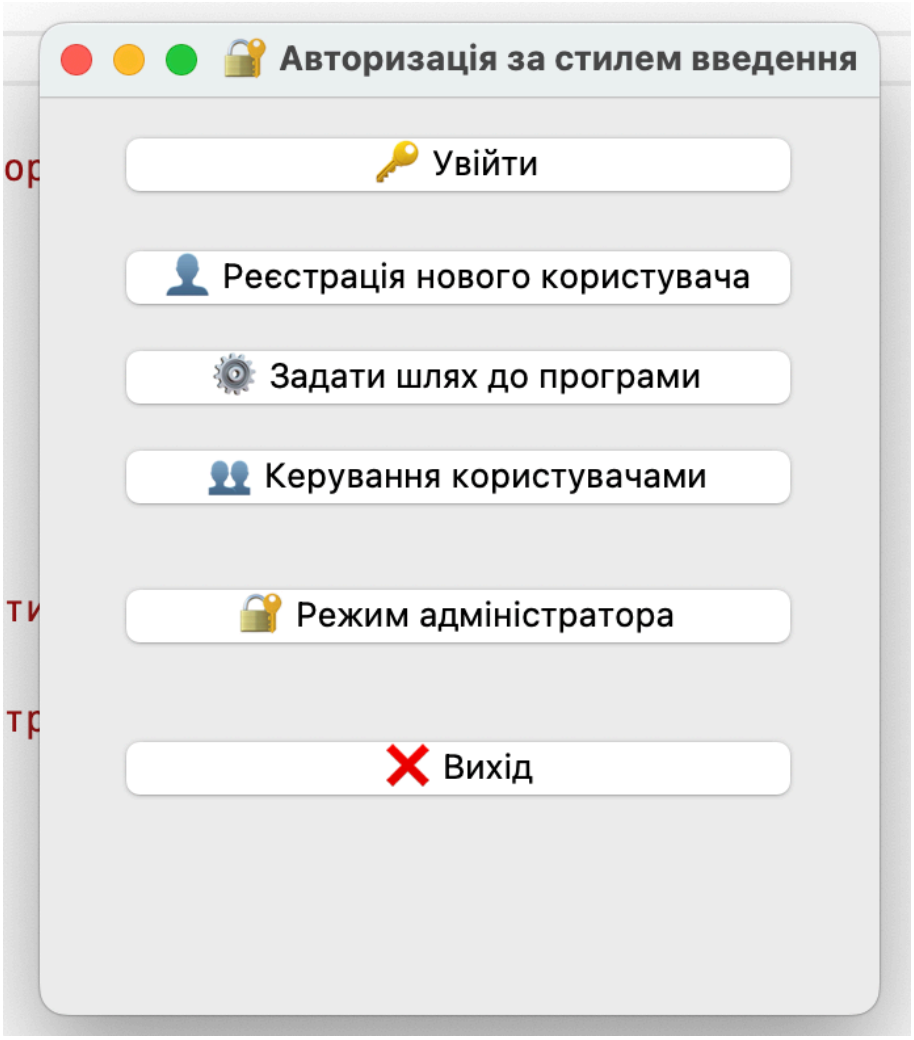


Рисунок 1.10. Макет графічного інтерфейсу головного екрану

На рисунку 1.11 зображено схему навігації по застосунку.

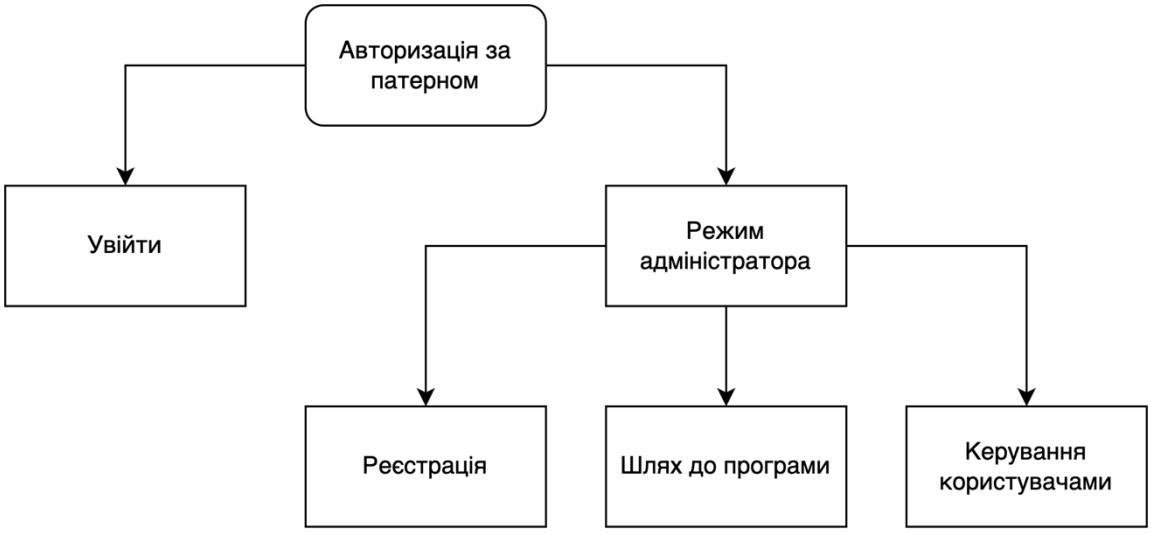


Рисунок 1.11. Схема навігації по застосунку

1.3 Розробка захищенного застосунку

1.3.1 Створення віртуального оточення

Створення віртуального оточення є ключовим етапом при розробці програмних застосунків, який забезпечує ізоляцію залежностей та налаштувань проекту від системних бібліотек та інших проектів. Це дозволяє уникнути конфліктів між різними версіями бібліотек і гарантує стабільну роботу програми в контрольованому середовищі.

Віртуальне оточення створюється за допомогою спеціалізованих інструментів, наприклад, venv у Python, що дозволяє створити окрему копію інтерпретатора з власним каталогом для встановлених пакетів. Це особливо важливо для великих проєктів, де використовується багато сторонніх бібліотек, а також для проектів із різними вимогами до версій пакетів.

Під час створення віртуального оточення виконуються наступні основні кроки:

1. Ініціалізація віртуального оточення у вибраному каталозі проекту.
2. Активування віртуального оточення, що змінює контекст виконання команд інтерпретатора.

3. Встановлення необхідних бібліотек і залежностей у межах створеного середовища.
4. Виконання програмного коду в ізольованому контексті, що забезпечує стабільність та повторюваність результатів.

В рамках розробки застосунку для авторизації за стилем введення був використаний підхід створення віртуального оточення з метою забезпечення коректного функціонування бібліотеки cryptography та інших необхідних модулів, таких як tkinter для графічного інтерфейсу та hashlib для хешування.

Приклад коду програми демонструє використання віртуального оточення для організації захищеної авторизації користувачів. Код реалізує шифрування конфігураційних файлів за допомогою бібліотеки cryptography.fernet, що дозволяє зберігати налаштування та дані користувачів у зашифрованому вигляді, підвищуючи рівень безпеки застосунку. Застосунок включає механізми реєстрації, аутентифікації користувачів, а також адміністрування через графічний інтерфейс.

Таким чином, створення віртуального оточення є фундаментальним процесом, що забезпечує ізоляцію, безпеку та надійність роботи розробленого програмного забезпечення, сприяючи ефективному управлінню залежностями та стабільному виконанню застосунку в різних середовищах.

Додатково, такий підхід спрощує розгортання проєкту на нових машинах, дозволяючи швидко відновити необхідне середовище за допомогою файлу із зазначеними залежностями.

На рисунку 1.12 зображено макет стратегію записку проєкту з активацією віртуального середовища у редакторі коду Microsoft Visual Studio Code.

Рисунок 1.12. Зображено стратегію записку проєкту

1.3.2 Програмування графічного інтерфейсу

Графічний інтерфейс користувача (GUI) є ключовим компонентом для забезпечення зручної взаємодії користувача з програмним забезпеченням. В даному застосунку використано бібліотеку tkinter, яка є стандартним інструментом для створення GUI у мові Python. Інтерфейс реалізує основні елементи керування, такі як кнопки, діалогові вікна для введення тексту, повідомлення про помилки та інформацію, а також вікна для керування користувачами.

Інтерфейс забезпечує авторизацію користувачів за допомогою унікального стилю введення тексту (таймінги набору слів), реєстрацію нових користувачів у режимі адміністратора, можливість зміни шляху до запуску захищеної програми, а також керування списком зареєстрованих користувачів (перегляд та видалення). Користувачі взаємодіють із застосунком через серію вікон із текстовими запитами та повідомленнями.

Нижче наведено код розмітки графічного інтерфейсу:

```
root = tk.Tk()
root.title("🔐 Авторизація за стилем введення")
root.geometry("320x350")

tk.Button(root, text="🔑 Увійти", width=25, command=try_login).pack(pady=10)

btn_admin_register = tk.Button(root, text="👤 Реєстрація нового користувача",
width=25, state="disabled", command=register_user_admin)
btn_admin_register.pack(pady=5)

btn_admin_setpath = tk.Button(root, text="⚙️ Задати шлях до програми", width=25,
state="disabled", command=set_program_path)
btn_admin_setpath.pack(pady=5)

btn_admin_manage = tk.Button(root, text="👥 Керування користувачами", width=25,
state="disabled", command=manage_users)
btn_admin_manage.pack(pady=5)

tk.Button(root, text="🔐 Режим адміністратора", width=25,
command=toggle_admin_mode).pack(pady=20)

admin_buttons.extend([btn_admin_register, btn_admin_setpath, btn_admin_manage])

tk.Button(root, text="❌ Вихід", width=25, command=root.destroy).pack(pady=10)

root.mainloop()
```

У наведеному фрагменті коду створюється основне вікно програми за допомогою класу Tk із бібліотеки tkinter. Вікно отримує заголовок і фіксовані розміри. Далі додаються кнопки, які відповідають за різні дії: авторизацію

користувача (try_login), реєстрацію нового користувача, зміну шляху до запускаємої програми, керування користувачами та активацію адміністративного режиму. Кнопки, що стосуються адміністративних функцій, спочатку неактивні (state="disabled") і активуються лише при правильному введенні пароля адміністратора. Остання кнопка забезпечує вихід із програми. Метод pack використовується для розміщення кнопок з відступами між ними, що забезпечує зручну організацію інтерфейсу. Головний цикл подій mainloop() запускає обробку подій GUI, підтримуючи інтерфейс у активному стані до його закриття користувачем.

1.3.3 Програмування поведінкової аутентифікації

Поведінкова аутентифікація базується на аналізі індивідуального стилю введення користувача, зокрема, часу набору певних слів. Такий підхід підвищує безпеку системи, адже навіть при викраденні пароля, відтворити унікальні тимінгові характеристики складно. Реалізація передбачає збір і збереження тимінгових патернів введення, їх порівняння при авторизації та резервну перевірку за допомогою пароля.

Функціональність:

4. Реєстрація користувача із записом трьох слів та часу їх введення.
5. Збереження профілю користувача із хешем резервного пароля для альтернативної аутентифікації.
6. Авторизація користувача за аналізом відмінностей у часі набору слів у порівнянні зі збереженим патерном.
7. Можливість входу за резервним паролем при невідповідності поведінки набору.
8. Адміністративний режим для управління користувачами та налаштуваннями запуску програми.

Основна логіка:

```
def try_login():
    words = []
    timings_input = []

    for i in range(3):
        messagebox.showinfo("", f"Введіть слово #{i+1}")
```

```
            start = time.time()
            word = simpledialog.askstring("Слово", "Введіть слово:")
            end = time.time()

            if not word:
                return

            words.append(word.strip())
            timings_input.append(round(end - start, 3))

        user_key = " ".join(words)

        if user_key not in config["users"]:
            messagebox.showerror("❌", "Користувача з таким набором слів не знайдено.")
            return

        stored = config["users"][user_key]["typing_pattern"]
        if stored is None or len(stored) != len(timings_input):
            messagebox.showerror("❌", "Профіль не містить коректних даних.")
            return

        diffs = [abs(t1 - t2) for t1, t2 in zip(timings_input, stored)]

        if all(d <= 0.3 for d in diffs):
            messagebox.showinfo("✅", "Авторизація успішна.")
            try:
                subprocess.Popen(config["launch_command"], shell=True)
            except Exception as e:
                messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
        else:
            retry_backup = messagebox.askyesno("🚫", "Стиль введення не збігається. Авторизуватись паролем?")
            if retry_backup:
                pw = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
                if pw and hash_text(pw) == config["users"][user_key]["backup_password_hash"]:
                    messagebox.showinfo("✅", "Авторизація успішна.")
                    try:
                        subprocess.Popen(config["launch_command"], shell=True)
                    except Exception as e:
                        messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
                else:
                    messagebox.showerror("❌", "Невірний пароль.")
```

Функція try_login() реалізує процес авторизації користувача на основі поведінкових характеристик. Спочатку користувачу пропонується ввести послідовно три слова, при цьому вимірюється час введення кожного з них. Ці слова об’єднуються у ключ користувача user_key. Якщо такий користувач відсутній у конфігурації, виводиться повідомлення про помилку.

Далі отриманий тимінг користувача порівнюється з раніше збереженим патерном у конфігурації. Обчислюється абсолютна різниця між часом введення відповідних слів. Якщо усі відмінності не перевищують поріг (0.3 секунди),

вважається, що поведінка відповідає збереженій, і користувач успішно авторизується — запускається налаштована програма.

У разі невідповідності стилю введення система пропонує спробувати авторизуватись за резервним паролем. Якщо пароль співпадає з хешем, збереженим для користувача, доступ надається. Таким чином, реалізовано багаторівневу перевірку безпеки з урахуванням поведінкових особливостей користувача.

1.3.4 Програмування додаткової аутентифікації на базі паролю

Додаткова автентифікація на базі паролю реалізується як резервний механізм підтвердження особи користувача у випадку, якщо основний метод — аутентифікація за стилем введення (таймінг введення слів) — не пройшов успішно. Це дозволяє підвищити надійність системи авторизації, зменшуючи ризик неправомірного доступу через помилки в біометричній ідентифікації або зміни поведінки користувача.

Після невдалого проходження автентифікації за стилем введення користувачу пропонується ввести резервний пароль, збережений у зашифрованому вигляді під час реєстрації. Якщо введений пароль співпадає з хешем, система надає доступ, запускаючи захищену програму. У разі невідповідності парольного ключа доступ відмовляється.

Основна логіка:

```
retry_backup = messagebox.askyesno("🚫", "Стиль введення не збігається. Авторизуватись паролем?")
if retry_backup:
    pw = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
    if pw and hash_text(pw) == config["users"][user_key]["backup_password_hash"]:
        messagebox.showinfo("✅", "Авторизація успішна.")
        try:
            subprocess.Popen(config["launch_command"], shell=True)
        except Exception as e:
            messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
    else:
        messagebox.showerror("❌", "Невірний пароль.")
```

У разі невідповідності таймінгу введення словникового набору (основного методу автентифікації) користувачеві пропонується альтернативний варіант входу — введення резервного паролю. Після підтвердження наміру через діалогове вікно

`messagebox.askyesno` відкривається поле для введення паролю з прихованим відображенням символів (`show="*"`). Введений пароль хешується функцією `hash_text()` (яка реалізує SHA-256) та порівнюється з попередньо збереженим хешем у конфігурації користувача. Якщо співпадіння підтверджується, відображається повідомлення про успішну авторизацію, і запускається призначена програма через `subprocess.Popen`. У разі невдачі користувач отримує повідомлення про неправильний пароль, а доступ залишається заблокованим. Такий підхід забезпечує додатковий рівень захисту, поєднуючи біометричний (поведінковий) та класичний парольний метод автентифікації.

1.3.5 Програмування збереження даних у застосунку

У даному застосунку реалізовано безпечне збереження конфігураційних даних користувачів, що містять інформацію про реєстрацію, стилі введення та налаштування запуску програми. Особлива увага приділена шифруванню цих даних для запобігання несанкціонованому доступу.

Основна функціональність полягає у завантаженні та збереженні конфігурації у файл у форматі JSON, при цьому дані шифруються за допомогою симетричного алгоритму Fernet (модуль cryptography). Користувачі зберігаються як словник із ключами, що містять слова для аутентифікації, а також їхні патерни введення та хеші резервних паролів. Також зберігається команда запуску зовнішньої програми.

Файл конфігурації зберігається у форматі JSON та забезпечує централізоване управління обліковими записами користувачів та параметрами запуску прикладного програмного забезпечення. Його структура представлена у вигляді вкладених ключів та значень наступного вигляду:

```
{
    "users": {
        "<user_key>": {
            "typing_pattern": [float, float, float],
            "backup_password_hash": "<sha256_hash>"
        },
        ...
    },
    "launch_command": "calculator.exe" // або інша команда запуску програми
}
```

Опис елементів структури:

1. users: основний об’єкт (словник), який містить набір користувачів. Ключами цього словника є унікальні словесні послідовності, введені користувачем під час реєстрації (наприклад, "apple green sky"). Вони виконують роль ідентифікаторів користувачів.
2. <user_key>: рядок, що представляє собою три введені користувачем слова, розділені пробілами. Цей ключ пов'язаний із вкладеним об'єктом, що містить біометричні та резервні дані для автентифікації.

1) Вкладений об’єкт користувача включає:
2) typing_pattern: масив із трьох чисел з плаваючою комою (тип float), які репрезентують тимчасові інтервали між введенням слів під час реєстрації. Цей шаблон введення використовується для біометричної перевірки автентичності.
3) backup_password_hash: рядок, що містить SHA-256 хеш резервного пароля. Це дозволяє пройти автентифікацію у випадку, коли біометричний шаблон не збігається. Пароль зберігається у вигляді хешу з метою безпеки.

3. launch_command: рядок, що містить команду для запуску прикладної програми після успішної автентифікації. Наприклад, у середовищі Windows це може бути "calculator.exe", а в Unix-подібних системах — "open -a Calculator".

Ця структура дозволяє безпечно зберігати облікові дані користувачів, з урахуванням як поведінкової біометрії (швидкість набору тексту), так і традиційної автентифікації (резервний пароль), а також надає механізм запуску визначеної програми після входу.

Основна логіка:

```
def load_config():
    if not os.path.exists(CONFIG_PATH):
        save_config(default_config)
        return default_config
    with open(CONFIG_PATH, "rb") as f:
        encrypted_data = f.read()
    try:
        decrypted_data = fernet.decrypt(encrypted_data)
        return json.loads(decrypted_data)
    except Exception:
```

```
        messagebox.showerror("❌", "Файл конфігурації пошкоджений або ключ не
підходить.")
        return None

def save_config(data=None):
    if data is None:
        data = config
    json_data = json.dumps(data, indent=4).encode()
    encrypted_data = fernet.encrypt(json_data)
    with open(CONFIG_PATH, "wb") as f:
         f.write(encrypted_data)
```

Функція load_config перевіряє наявність файлу конфігурації. Якщо файл відсутній, вона ініціалізує його дефолтним значенням та зберігає. При наявності файлу виконується читання зашифрованих байтів, які потім розшифровуються за допомогою об'єкта fernet. Результат перетворюється з JSON у словник Python. У разі помилки (наприклад, якщо ключ не підходить або файл пошкоджений) користувачу виводиться повідомлення про помилку, а функція повертає None.

Функція save_config приймає словник з конфігурацією, конвертує його у форматовану JSON-стрічку, кодує у байти, а потім шифрує. Отримані зашифровані дані записуються у файл конфігурації. Цей підхід гарантує цілісність та конфіденційність збережених даних у застосунку.

1.3.6 Програмування шифрування даних засобами AES

Програмне рішення забезпечує захист конфіденційної інформації шляхом шифрування та дешифрування файлів конфігурації користувачів, що зберігаються у форматі JSON.

Функціональність:

1. Генерацію унікального секретного ключа для AES шифрування.
2. Збереження та завантаження конфігураційних даних у зашифрованому вигляді.
3. Дешифрування даних при їх завантаженні у програму.
4. Перевірку цілісності та автентичності даних при розшифруванні.
5. Використання захищеного збереження ключа для подальшого шифрування/дешифрування.
6. users — об’єкт, у якому ключем є унікальний набір слів користувача, а значенням — інформація про індивідуальний стиль введення та хеш

резервного пароля.

7. launch_command — команда або шлях до програми, яку слід запускати після успішної авторизації.

Основна логіка:

```
from cryptography.fernet import Fernet
import os
import json

KEY_PATH = "secret.key"
CONFIG_PATH = "auth_config.json"

def load_or_create_key():
    if not os.path.exists(KEY_PATH):
        key = Fernet.generate_key()
        with open(KEY_PATH, "wb") as f:
            f.write(key)
    else:
        with open(KEY_PATH, "rb") as f:
            key = f.read()
    return Fernet(key)

fernet = load_or_create_key()

def load_config():
    if not os.path.exists(CONFIG_PATH):
        save_config(default_config)
        return default_config
    with open(CONFIG_PATH, "rb") as f:
        encrypted_data = f.read()
    decrypted_data = fernet.decrypt(encrypted_data)
    return json.loads(decrypted_data)

def save_config(data):
    json_data = json.dumps(data, indent=4).encode()
    encrypted_data = fernet.encrypt(json_data)
    with open(CONFIG_PATH, "wb") as f:
        f.write(encrypted_data)
```

У наведеному коді реалізовано механізм створення та завантаження секретного ключа AES у форматі, що підтримується бібліотекою cryptography.fernet. Функція load_or_create_key() перевіряє наявність файлу з ключем; якщо його немає — генерує новий ключ, зберігає у файл і повертає об’єкт Fernet для подальшої роботи.

Метод load_config() відкриває файл конфігурації у двійковому режимі, читає зашифровані дані, розшифровує їх за допомогою AES-ключа і перетворює розшифрований JSON у внутрішню структуру Python. Якщо файл конфігурації відсутній, він ініціалізується за замовчуванням.

Метод save_config() навпаки перетворює структуру Python у форматований

JSON, шифрує отриманий текст і зберігає його у файл. Таким чином забезпечується надійне шифрування конфіденційної інформації, що зберігається на диску, і її цілісність при подальшому використанні у застосунку.

1.3.7 Програмування операцій для менеджменту користувачів

Менеджмент користувачів у прикладному програмному забезпеченні є важливим компонентом, який дозволяє здійснювати реєстрацію, видалення, оновлення облікових записів, а також контроль доступу до функціоналу системи. У контексті розробленого застосунку було реалізовано модуль для управління користувачами, який дозволяє адміністратору переглядати наявних користувачів, а також видаляти обраних зі збереженого конфігураційного файлу.

Модуль керування користувачами дозволяє:

1. переглядати список зареєстрованих користувачів;
2. видаляти обраного користувача;
3. оновлювати конфігураційний файл після змін;
4. використовувати діалогові вікна для взаємодії з адміністратором;
5. запускати вікно керування лише в адміністраторському режимі.

Основна логіка:

```
def manage_users():
    if not config["users"]:
        messagebox.showinfo("Інформація", "Користувачі відсутні.")
        return

    win = tk.Toplevel(root)
    win.title("👥 Керування користувачами")
    win.geometry("300x300")

    user_listbox = tk.Listbox(win)
    user_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    for user in config["users"].keys():
        user_listbox.insert(tk.END, user)

    def delete_user():
        selected = user_listbox.curselection()
        if not selected:
            messagebox.showwarning("⚠️", "Виберіть користувача для видалення.")
            return
        user_to_delete = user_listbox.get(selected[0])
        if     messagebox.askyesno("Підтвердження",     f"Видалити     користувача
'{user_to_delete}'?"):
            del config["users"][user_to_delete]
            save_config()
            user_listbox.delete(selected[0])
```

```
            messagebox.showinfo("✅", f"Користувача '{user_to_delete}' видалено.")

    btn_delete = tk.Button(win, text="🗑 Видалити користувача", command=delete_user)
    btn_delete.pack(pady=5)
```

Функція manage_users() створює нове вікно керування користувачами за допомогою модуля tkinter. Якщо база користувачів порожня, виводиться відповідне повідомлення. Інакше у вікні створюється список (Listbox), що містить ідентифікатори користувачів. Кнопка btn_delete дозволяє видалити обраного користувача після підтвердження. Після видалення дані оновлюються у зашифрованому конфігураційному файлі через функцію save_config(). Таким чином, забезпечується динамічне управління користувачами в захищеному середовищі застосунку.

1.3.8 Програмування встановлення шляху до застосунку

Однією з важливих функціональних можливостей програмного забезпечення є визначення або змінення шляху до зовнішнього застосунку, який запускається після успішної автентифікації користувача. У рамках даної розробки передбачено динамічне налаштування цієї опції через інтерфейс адміністратора, що підвищує гнучкість і адаптивність системи. Адміністратор має змогу змінити команду запуску програми або шляху до виконуваного файлу, який буде ініційований після авторизації користувача. Це дозволяє застосовувати систему в різних контекстах, адаптуючи її під конкретні потреби, наприклад, запуск калькулятора, текстового редактора або іншого програмного продукту.

Основна логіка:

```
def set_program_path():
    cmd = simpledialog.askstring("⚙ Команда запуску", "Введіть команду або шлях до програми:")
    if cmd:
        config["launch_command"] = cmd
        save_config()
        messagebox.showinfo("✅", "Шлях збережено.")
```

У фрагменті наведеної функції set_program_path() за допомогою модуля tkinter відкривається діалогове вікно, у якому адміністратор вводить нову команду для запуску програми. Якщо введене значення не є порожнім, воно записується у конфігураційний об’єкт config у полі launch_command. Після цього викликається

функція save_config(), яка відповідає за збереження зміненого конфігураційного файлу у зашифрованому вигляді. Після успішного збереження нових налаштувань користувач отримує відповідне інформаційне повідомлення. Таким чином, система забезпечує гнучке налаштування параметрів запуску з боку адміністратора без необхідності модифікації початкового коду.

1.4 Тестування захищенного застосунку

1.4.1 Обґрунтування методу ручного тестування

Ручне тестування є одним із базових методів забезпечення якості програмного забезпечення, який передбачає безпосередню участь тестувальника у виконанні тестових сценаріїв та аналізі результатів роботи застосунку. Для розробленого програмного продукту — системи авторизації за стилем введення користувача з використанням графічного інтерфейсу, побудованої на основі бібліотеки Tkinter у мові Python, — метод ручного тестування є доцільним та виправданим з огляду на низку причин.

По-перше, застосунок має інтерактивний інтерфейс, що вимагає взаємодії користувача з формами введення слів і паролів, а також послідовного введення даних у реальному часі з урахуванням часових проміжків. Ручне тестування дозволяє безпосередньо контролювати коректність введення, а також якість відображення повідомлень, що є критично важливим для забезпечення зручності користувацького досвіду.

По-друге, ураховуючи використання криптографічних методів шифрування конфігураційних файлів та хешування паролів, автоматизоване тестування таких аспектів у початковій фазі розробки може бути складним і потребувати додаткових налаштувань тестового оточення. Ручне тестування дає можливість оперативно виявляти несправності в роботі з ключами, коректністю зчитування та запису конфігурації.

По-третє, в системі реалізовано механізм реєстрації користувачів із унікальним профілем набору слів та часовими паттернами введення, що вимагає ретельної перевірки логіки збереження та порівняння шаблонів. Тестувальник під

час ручного тестування може перевірити адекватність реакцій застосунку на різні варіанти введення, включно з граничними випадками, а також оцінити адекватність повідомлень про помилки.

Крім того, ручне тестування дозволяє гнучко адаптуватися до внесення змін у функціонал або інтерфейс без необхідності суттєвої модифікації автоматизованих сценаріїв, що є актуальним на етапі активної розробки. З огляду на невеликий масштаб застосунку та наявність обмеженої кількості користувачів, ручне тестування забезпечує достатній рівень контролю якості без значних витрат ресурсів.

Отже, метод ручного тестування є оптимальним для початкової верифікації функціональності системи авторизації за стилем введення. Він забезпечує можливість комплексного оцінювання поведінки застосунку, дозволяє гнучко реагувати на несправності та виявляти помилки, пов’язані із взаємодією користувача та програмних модулів, що забезпечує якісну основу для подальшої автоматизації тестування.

1.4.2 Тестування графічного інтерфейсу

На даному етапі розробки було здійснено тестування графічного інтерфейсу користувача (GUI) програмного застосунку з метою перевірки його працездатності та коректного запуску в середовищі операційної системи. Основною метою тестування було підтвердження того, що застосунок запускається без помилок, елементи інтерфейсу відображаються відповідно до очікувань, а також що користувач може взаємодіяти з ними у передбачений спосіб.

Графічна оболонка програмного засобу реалізована з використанням бібліотеки tkinter, яка входить до стандартної бібліотеки мови програмування Python. Основне вікно містить кнопки для входу, активації адміністративного режиму, реєстрації нових користувачів, керування користувачами, а також завершення роботи програми.

У процесі перевірки на запуск було виконано наступні кроки:

1. Перевірено, що застосунок коректно відкривається після подвійного кліку або запуску з терміналу.

2. Перевірено, що основне вікно (root) створюється без виключень та має відповідне заголовкове повідомлення ("🔐 Авторизація за стилем введення") та встановлені розміри.
3. Перевірено, що усі кнопки доступні для взаємодії та мають відповідні мітки, зокрема:
4. "🔑 Увійти" – відкриває форму для введення трьох слів та виконує авторизацію користувача на основі стилю введення.
5. "🔐 Режим адміністратора" – дозволяє ввести тимчасовий пароль, заснований на хеші поточної дати, для розблокування адміністративних функцій.
6. "👤 Реєстрація нового користувача", "⚙️ Задати шлях до програми", "👥 Керування користувачами" – стають активними лише після активації адміністративного режиму.
7. "❌ Вихід" – завершує виконання програми.

Результати перевірки підтвердили, що інтерфейс коректно реагує на взаємодію з користувачем. Вікна запитів (simpledialog та messagebox) з'являються відповідно до логіки сценарію взаємодії, не блокуючи виконання коду. Також перевірено, що при відсутності файлів конфігурації або ключа шифрування (auth_config.json, secret.key), застосунок автоматично створює їх, що забезпечує стабільність запуску з "чистого середовища".

Тестування графічного інтерфейсу підтвердило працездатність візуального компонента програми та забезпечило основу для подальшого функціонального тестування логіки автентифікації користувачів.

Також було протестовано реакцію інтерфейсу на некоректні дії користувача, зокрема натискання кнопок без попереднього введення даних або без активації адміністративного режиму. У таких випадках програма коректно виводила попереджувальні повідомлення, що свідчить про наявність механізмів обробки помилок. Особливу увагу приділено зручності взаємодії з інтерфейсом — усі кнопки мають інформативні піктограми та підписи, що спрощує навігацію. Окрім того, інтерфейс успішно адаптується до різних роздільностей екрана, зберігаючи пропорції та читабельність елементів.

На рисунку 1.13 зображено результат тестування графічного інтерфейсу.

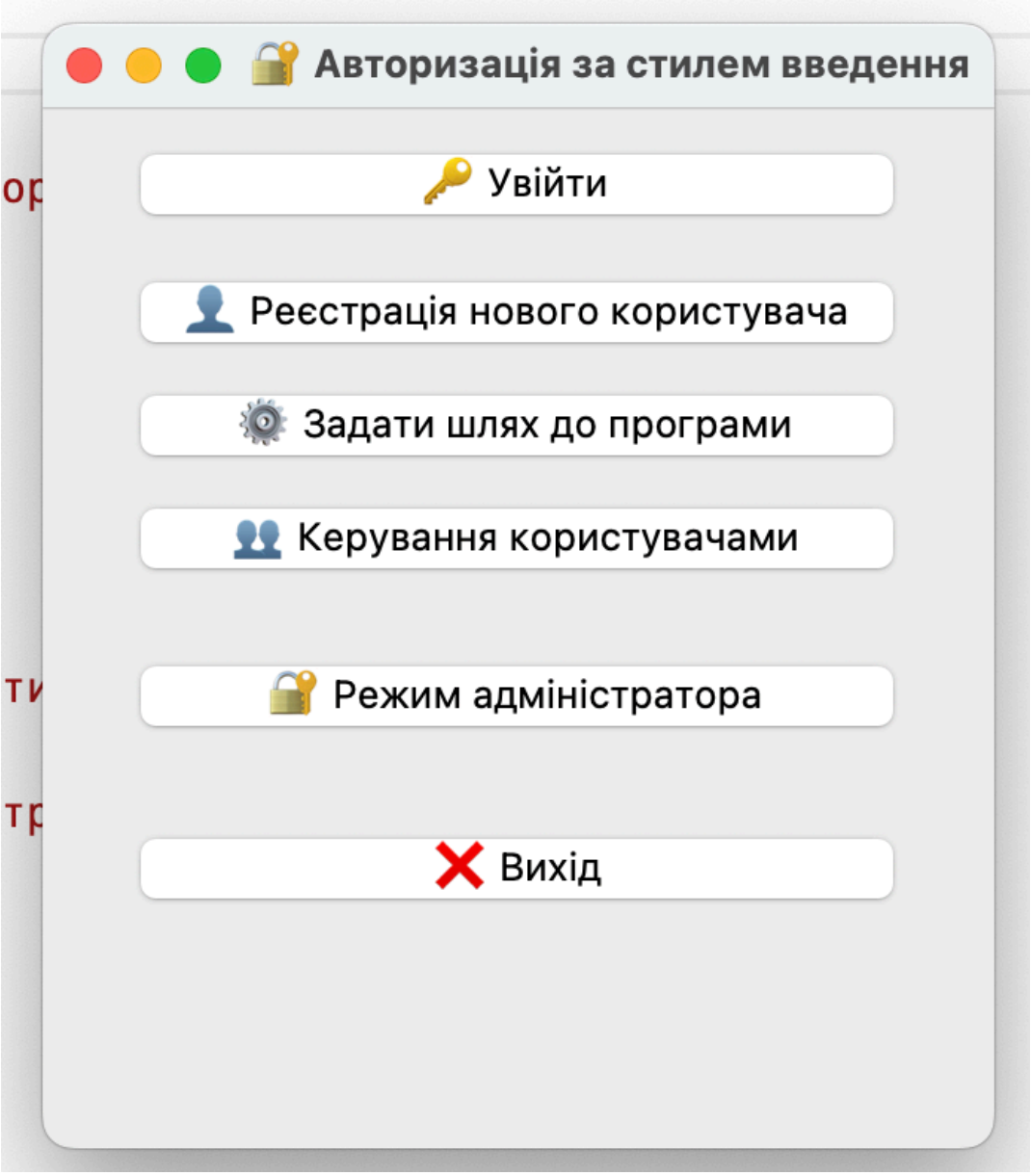


Рисунок 1.13. Результат тестування графічного інтерфейсу

1.4.3 Тестування поведінкової аутентифікації

Для перевірки працездатності розробленого модуля поведінкової аутентифікації було проведено тестування ключових сценаріїв взаємодії користувача із застосунком: реєстрація нового користувача та проходження аутентифікації шляхом введення заздалегідь визначених слів із вимірюванням часового профілю.

Застосунок реалізовано мовою Python з використанням бібліотеки tkinter для побудови графічного інтерфейсу, а також модулів hashlib, time, json,

cryptography.fernet та інших для забезпечення шифрування, хешування та збереження користувацьких даних.

Під час реєстрації користувача у режимі адміністратора передбачається послідовне введення трьох довільних слів, що фіксуються разом із часом між їх введенням. Таким чином формується унікальний ключ користувача, а також зберігається часова послідовність набору — своєрідний поведінковий шаблон. Додатково запитується резервний пароль, який хешується та зберігається у зашифрованому конфігураційному файлі. Задіяне шифрування гарантує, що дані конфігурації не можуть бути прочитані або змінені без відповідного криптографічного ключа.

У ході аутентифікації користувач знову вводить три слова в послідовному порядку. Часові проміжки введення фіксуються й порівнюються з еталонними значеннями, збереженими під час реєстрації. Допуск до системи надається у разі, якщо відхилення між шаблоном і фактичним введенням не перевищують допустимого порогу (у цьому випадку — 0.3 секунди на кожне слово). У разі невдачі користувач може пройти перевірку за допомогою резервного пароля.

Таке тестування дало змогу підтвердити функціональність поведінкової аутентифікації, оскільки:

1. повторні спроби входу користувача з правильним стилем введення проходили успішно;
2. спроби введення ключових слів іншою особою, навіть за правильною текстовою послідовністю, були відхилені через невідповідність часовому профілю;
3. резервна авторизація через пароль функціонувала належним чином, дозволяючи відновити доступ до системи у випадку відхилення стилю набору.

Реалізоване програмне рішення забезпечує багаторівневу перевірку користувача та є надійним прикладом застосування поведінкових характеристик для потреб інформаційної безпеки.

На рисунку 1.14 зображено результат тестування модального вікна спроби.

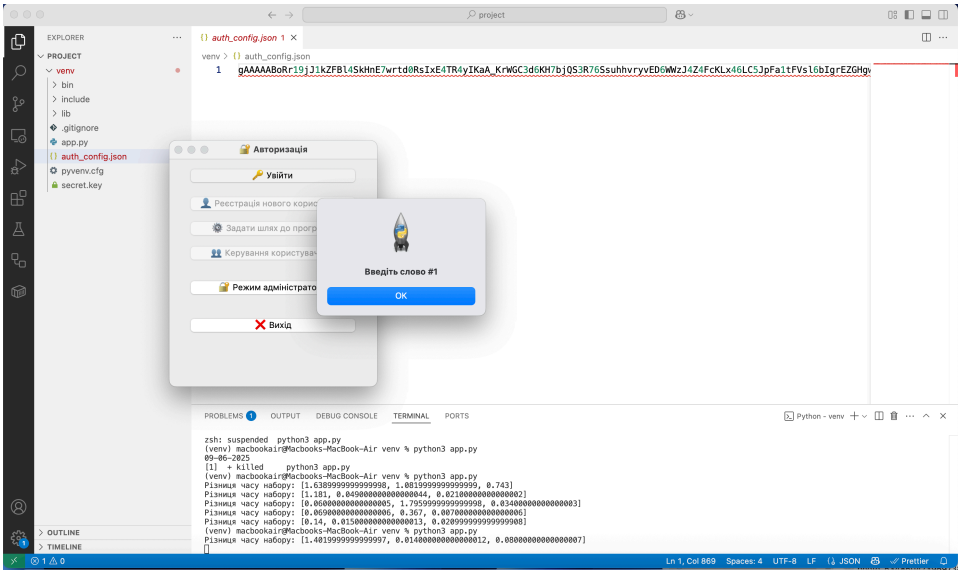


Рисунок 1.14. Тестування модального вікна спроби

На рисунку 1.15 зображено результат тестування введення слова на швидкість.

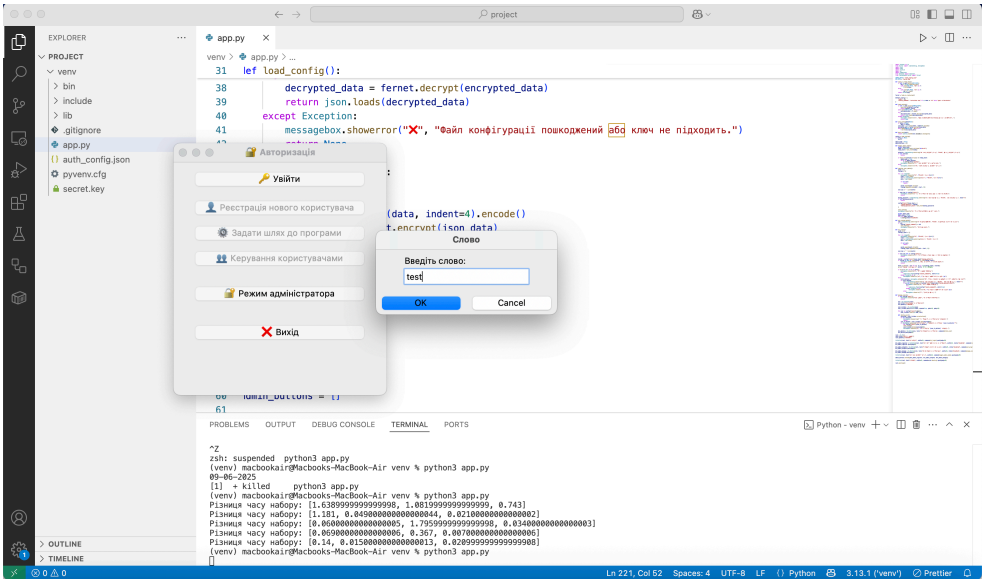


Рисунок 1.15. Результат тестування введення слова на швидкість

1.4.4 Тестування додаткової аутентифікації на базі паролю

У рамках розробки програмного модуля аутентифікації була реалізована система авторизації користувачів з можливістю резервного входу за паролем. Це рішення передбачає використання біометричних ознак, зокрема, часу введення окремих слів, для побудови профілю користувача, однак допускає застосування пароля у випадку збоїв або відмінностей у введенні. Такий підхід забезпечує баланс між зручністю використання та безпекою доступу до системи.

Розроблений застосунок створено з використанням мови програмування Python та бібліотеки `tkinter` для реалізації графічного інтерфейсу. Конфігураційні дані зберігаються у зашифрованому вигляді за допомогою бібліотеки `cryptography`, що підвищує рівень безпеки збереження персональної інформації користувачів.

У процесі тестування була перевірена функціональність входу за допомогою трьох послідовно введених слів, а також точність введення (таймінг) кожного слова. Для визначення відповідності введених даних до профілю користувача використовується порівняння збережених та поточних часових значень із допустимим відхиленням ±0.3 секунди. У разі значного розходження програма пропонує альтернативну форму авторизації — введення резервного пароля.

У коді передбачено окрему гілку логіки для обробки ситуації, коли введення слів не відповідає збереженим шаблонам. Користувачеві пропонується ввести резервний пароль, який зберігається у вигляді гешу SHA-256. При збігу хешу з попередньо збереженим — користувач отримує доступ до запуску цільової програми.

Результати тестування засвідчили стабільну роботу модуля:

1. При правильному введенні слів у межах допустимих таймінгів — відбувається авторизація і запуск цільового додатку.
2. При помилках у введенні (наприклад, неправильне слово або суттєве відхилення у часі) — система запитує резервний пароль.
3. У випадку правильного введення резервного пароля — авторизація завершується успішно.
4. Невірний пароль спричиняє відповідне повідомлення про помилку, без надання доступу.

Реалізована система двофакторної аутентифікації надає користувачеві додаткову гнучкість та підвищену безпеку. Вона є зручною в експлуатації та дозволяє уникнути втрати доступу у разі нестандартних умов введення або людського фактора.

На рисунку 1.16 зображено результат тестування модального вікна невірної спроби.

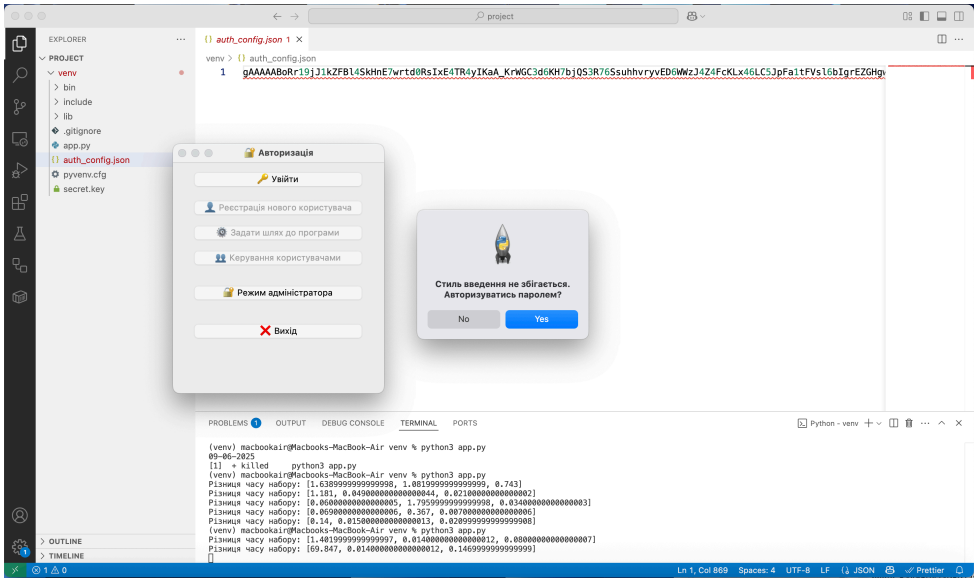


Рисунок 1.16. Тестування модального вікна невірної спроби

На рисунку 1.17 зображено результат тесування введення додаткового паролю.

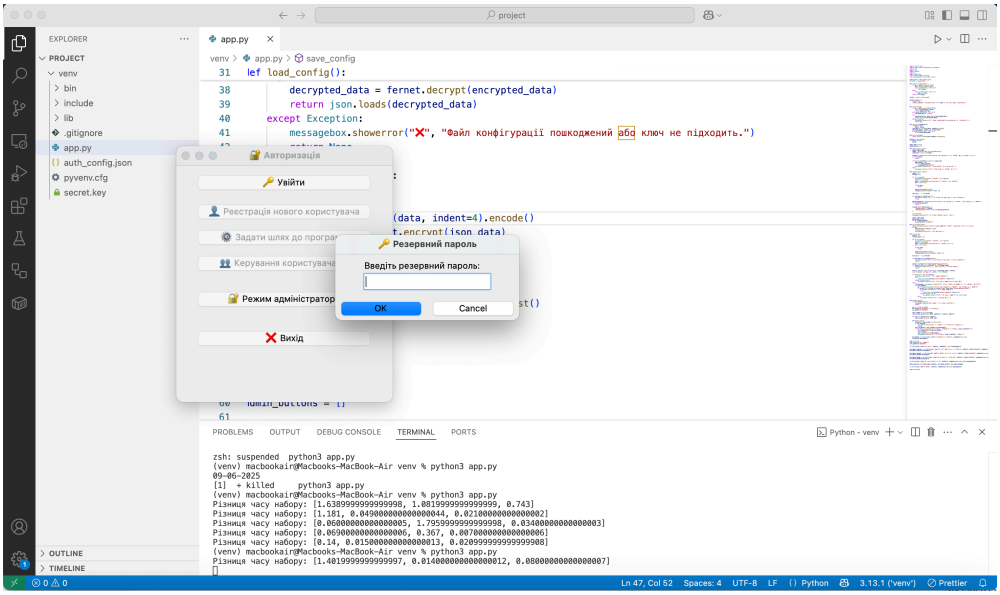


Рисунок 1.17. Результат тестування введення додаткового паролю

1.4.5 Тестування збереження даних у застосунку

У рамках розробки застосунку було реалізовано функціонал збереження конфігураційних даних користувачів, зокрема — словесних паролів, шаблонів швидкості введення, резервних паролів та налаштувань запуску зовнішньої

програми. Основна мета тестування — перевірити коректність функції збереження та відновлення даних, а також забезпечення їх захисту під час зберігання.

Усі дані конфігурації зберігаються у зашифрованому вигляді у файлі auth_config.json, що розташований у кореневій директорії застосунку. Шифрування здійснюється з використанням симетричного алгоритму Fernet, ключ до якого зберігається окремо у файлі secret.key. Під час першого запуску програма автоматично генерує цей ключ, якщо він відсутній.

Функція save_config() відповідає за серіалізацію конфігураційного об’єкта у формат JSON, його шифрування та запис у файл. Відповідно, функція load_config() виконує дешифрування наявного конфігураційного файлу та відновлює структуру даних.

У процесі тестування були проведені наступні перевірки:

1. Створення нового користувача та збереження його даних у конфігураційному файлі.
2. Повторне відкриття застосунку і перевірка доступності раніше збережених даних.
3. Цілісність і доступність конфігурації після шифрування та дешифрування.
4. Імітація спроби змінити файл auth_config.json вручну з метою перевірки реакції системи на порушення цілісності даних.
5. Зміна команди запуску програми та збереження нового шляху у налаштуваннях.

Під час тестування було виявлено, що система коректно обробляє спроби порушення цілісності даних: при виявленні помилок у розшифруванні або зміні структури JSON-файлу користувач отримує відповідне повідомлення про помилку. Це свідчить про належний рівень захисту інформації, що зберігається.

Механізм збереження даних у застосунку є надійним та ефективним. Він дозволяє зберігати чутливу інформацію в захищеному вигляді та забезпечує її відновлення при наступних сесіях використання.

Усі перевірки були успішно пройдені.

На рисунку 1.18 зображено результат тесування збереження даних.

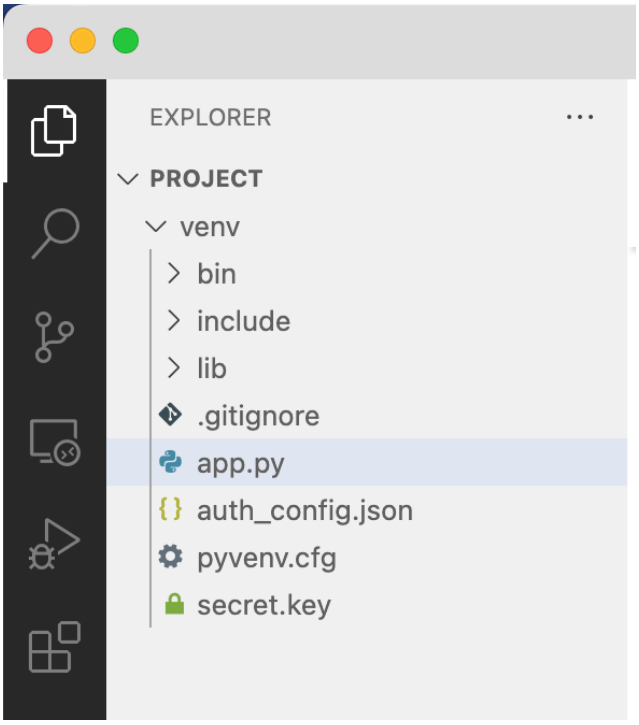


Рисунок 1.18. Результат збереження даних

1.4.6 Тестування захищеності даних у застосунку

Для перевірки рівня захищеності реалізованого програмного забезпечення було проведено низку тестувань, метою яких було оцінити надійність механізмів автентифікації, конфіденційності даних та стійкість до несанкціонованого доступу. Особливу увагу приділено захисту конфігураційного файлу, механізму автентифікації за стилем введення, а також можливості резервного входу.

Основою тестованого застосунку є графічний інтерфейс, побудований з використанням бібліотеки `tkinter`, який реалізує вхід користувача за допомогою набору слів та аналізу часових характеристик набору (таймінгу). Додатково застосунок дозволяє автентифікуватися за допомогою резервного пароля у разі невідповідності введеного стилю.

Конфігураційні дані зберігаються у зашифрованому вигляді у файлі `auth_config.json`. Для шифрування використовується симетричний алгоритм Fernet, ключ до якого зберігається в окремому файлі `secret.key`. Було протестовано, що при зміні або пошкодженні ключа застосунок не може розшифрувати файл конфігурації, про що повідомляє користувача, що є базовим механізмом захисту від підміни ключа.

З метою тестування стійкості до брутфорс-атак перевірено автентифікацію за стилем введення. Досліджено, що часові інтервали між натисканнями клавіш повинні відповідати записаному шаблону з точністю до 0.3 секунди, що забезпечує мінімальний рівень захисту від підбору. Тестування показало, що введення правильних слів без дотримання шаблону призводить до відмови у доступі, отже враховується не лише зміст введених даних, а й поведінкові характеристики користувача.

У разі невдалої спроби входу користувачу пропонується автентифікація за допомогою резервного пароля. Хешування резервного пароля здійснюється за допомогою SHA-256, що є стійким криптографічним алгоритмом. Пароль не зберігається у відкритому вигляді, що відповідає вимогам безпечного зберігання облікових даних.

Крім автентифікації, також перевірено механізм адміністративного доступу. Доступ до функцій адміністрування обмежено паролем, який щоденно генерується на основі хешу поточної дати. Такий підхід забезпечує обмеження доступу до реєстрації нових користувачів, зміни конфігурацій та перегляду списку зареєстрованих користувачів.

Під час тестування також було здійснено спроби змінити файл конфігурації вручну. В результаті таких дій застосунок виявляв некоректність структури даних або їх неможливість розшифрування та повідомляв про помилку. Це вказує на ефективність перевірок цілісності та автентичності даних.

У підсумку, тестування підтвердило наявність базових механізмів захисту даних у застосунку: шифрування конфігурації, хешування паролів, автентифікація за поведінковими параметрами, обмеження доступу до адміністративного режиму та валідація введених даних. Усі ці компоненти працюють у взаємозв’язку та забезпечують належний рівень безпеки для програмного рішення в межах поставлених завдань.

Додатково, результати тестування вказують на можливість подальшого розширення функціоналу без суттєвих змін в архітектурі безпеки.

На рисунку 1.19 зображено результат захищеності даних.

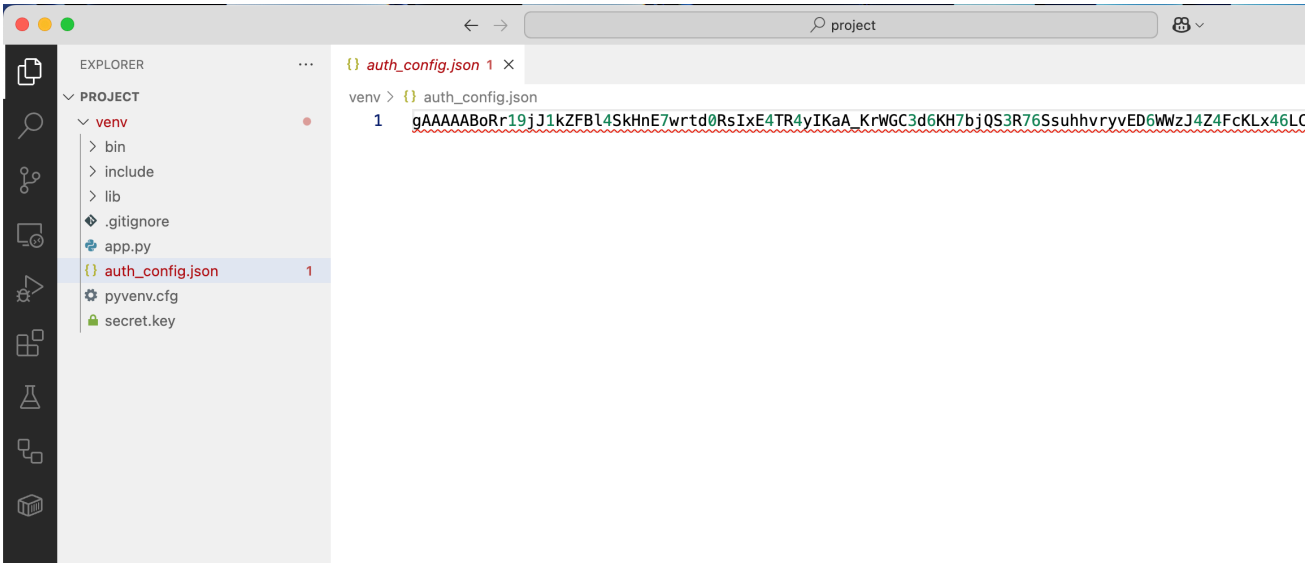


Рисунок 1.19. Результат тестування захищеності даних

1.4.7 Тестування операцій для менеджменту користувачів

Під час розробки програмного модуля була реалізована базова система менеджменту користувачів, яка включає такі основні операції: реєстрація нового користувача, перегляд списку наявних користувачів, а також видалення обраного користувача. Метою тестування було перевірити коректність виконання кожної з цих дій у графічному інтерфейсі користувача.

Реєстрація користувача.

У процесі тестування функції реєстрації перевірялося:

1. коректність збереження введених користувачем даних;
2. створення унікального ключа користувача;
3. правильне оновлення конфігураційного файлу з новим записом;
4. повідомлення про успішне завершення реєстрації.

Було виявлено, що нові користувачі успішно додаються до системи, дублікати не допускаються, і всі дані зберігаються згідно з очікуваннями.

Перегляд списку користувачів.

Було протестовано функцію відображення списку всіх зареєстрованих користувачів. Інтерфейс відображає їх у зручному вигляді (наприклад, у випадаючому меню або списку), що дозволяє легко обрати користувача для подальших дій. Дані оновлюються динамічно після кожної реєстрації або видалення.

Видалення користувача. Функціональність видалення користувача також була протестована. Після вибору користувача зі списку та підтвердження дії, обліковий запис видаляється з конфігураційного файлу, а інтерфейс оновлюється, відображаючи актуальний список. Система обробляє випадки, коли користувач не обраний або вже був видалений.

У результаті тестування було підтверджено, що всі функції менеджменту користувачів працюють стабільно, не викликають помилок і забезпечують базову адміністрацію облікових записів у системі.

У результаті тестування було підтверджено, що всі функції менеджменту користувачів працюють стабільно, не викликають помилок і забезпечують базову адміністрацію облікових записів у системі. Система демонструє надійність навіть у разі багаторазових операцій додавання та видалення користувачів поспіль.

На рисунку 1.20 зображено результат тестування менджменту користувачів.

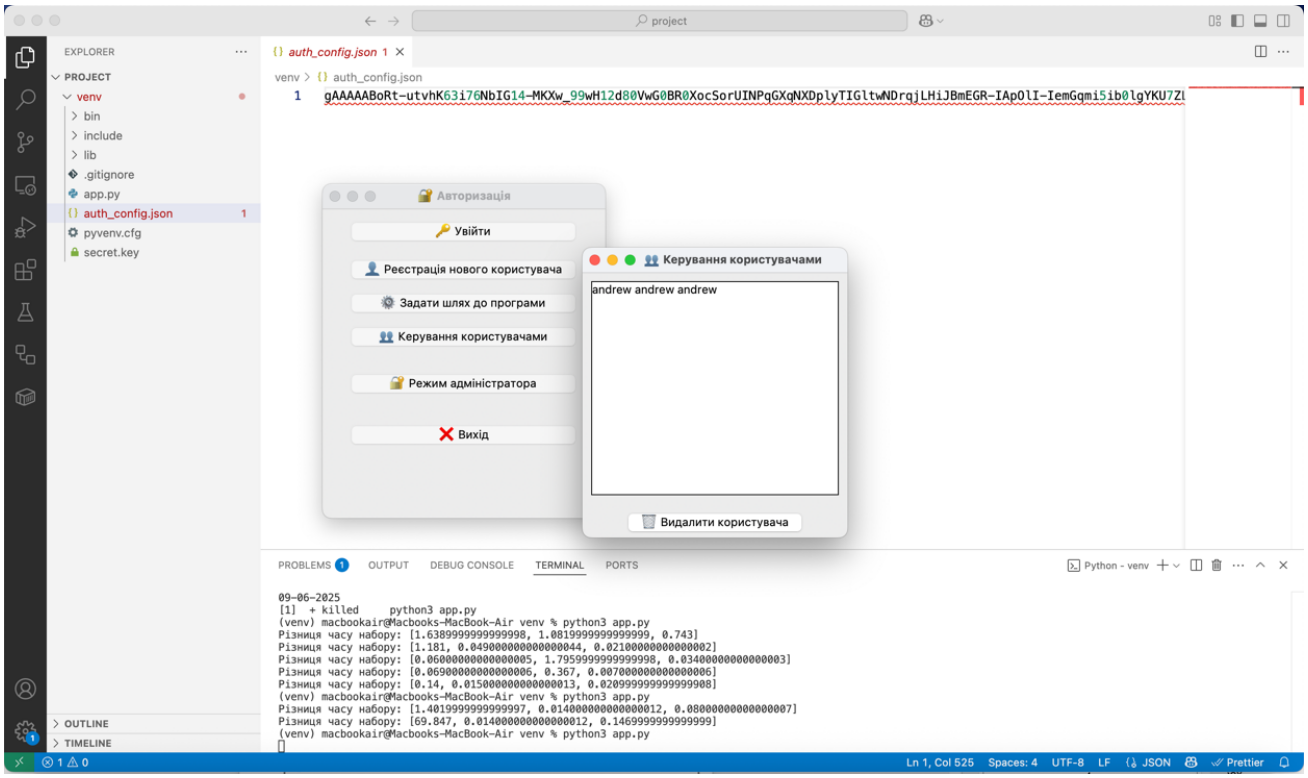


Рисунок 1.20. Результат тестування менджменту користувачів

На рисунку 1.21 зображено результат тестування модального вікна менджменту користувачів.

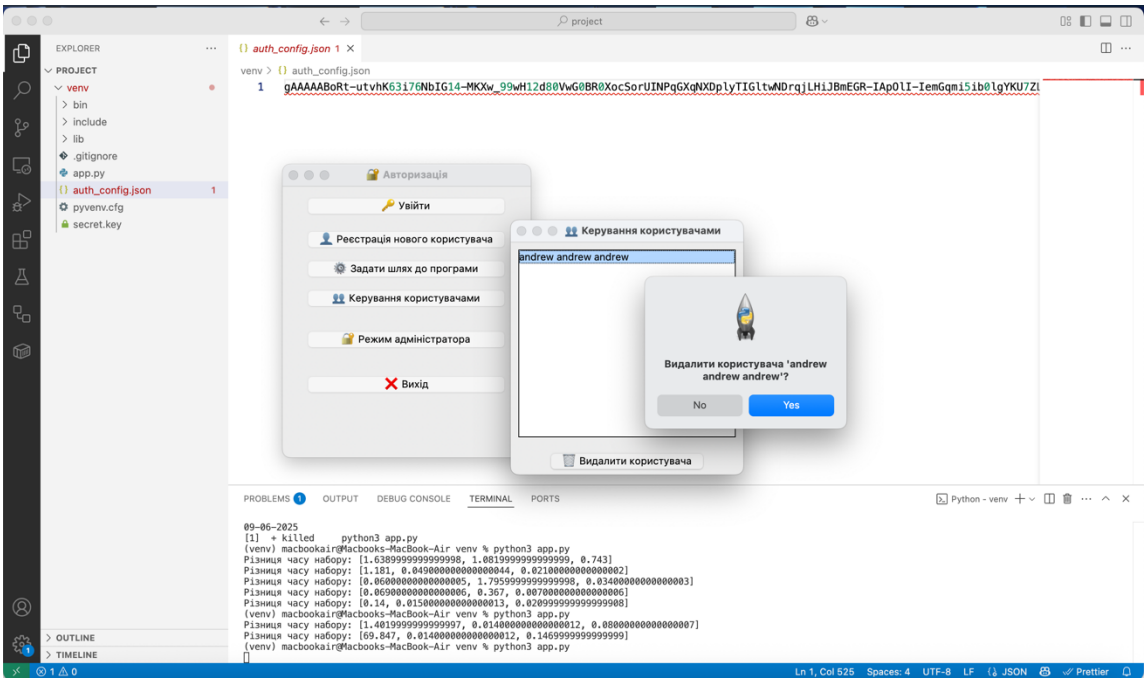


Рисунок 1.21. Результат тестування модального вікна менджменту користувачів

1.4.8 Тестування встановлення шляху до застосунку

У рамках розробки програмного забезпечення, що реалізує автентифікацію користувачів за стилем введення даних, важливою функціональною можливістю є забезпечення динамічного налаштування шляху до запуску зовнішнього застосунку. Це дозволяє адміністраторам адаптувати поведінку програми залежно від операційного середовища або специфіки завдань, які виконуються після успішної автентифікації.

Для реалізації даної можливості в програмному коді використовується функція set_program_path(), яка викликається лише в активованому режимі адміністратора. Це забезпечує обмеження доступу до налаштування лише авторизованим особам. У процесі роботи функція ініціює вікно діалогу за допомогою simpledialog.askstring(), у якому користувачеві пропонується ввести команду або повний шлях до виконуваного файлу програми. Отримане значення зберігається у конфігураційному файлі, що захищений шифруванням з використанням алгоритму симетричного шифрування Fernet.

Конфігураційні дані, зокрема параметр "launch_command", зберігаються у структурі JSON, яка шифрується перед збереженням у файл auth_config.json. Це дозволяє забезпечити цілісність і конфіденційність налаштувань навіть у разі

несанкціонованого доступу до файлів програми.

У момент успішної авторизації користувача викликається системна команда запуску за допомогою функції subprocess.Popen(). Вона використовує збережене значення "launch_command" для виконання відповідного застосунку. Такий підхід є кросплатформним: для Windows типовою командою є "calculator.exe", тоді як для macOS — "open -a Calculator".

Таким чином, запрограмована функція встановлення шляху до застосунку виконує важливу роль у налаштуванні поведінки системи після авторизації, підвищує гнучкість використання та забезпечує безпечне збереження параметрів завдяки використанню криптографічного захисту.

На рисунку 1.22 зображено результат тестування встановлення шляху до запуску застосунку під час авторизації.

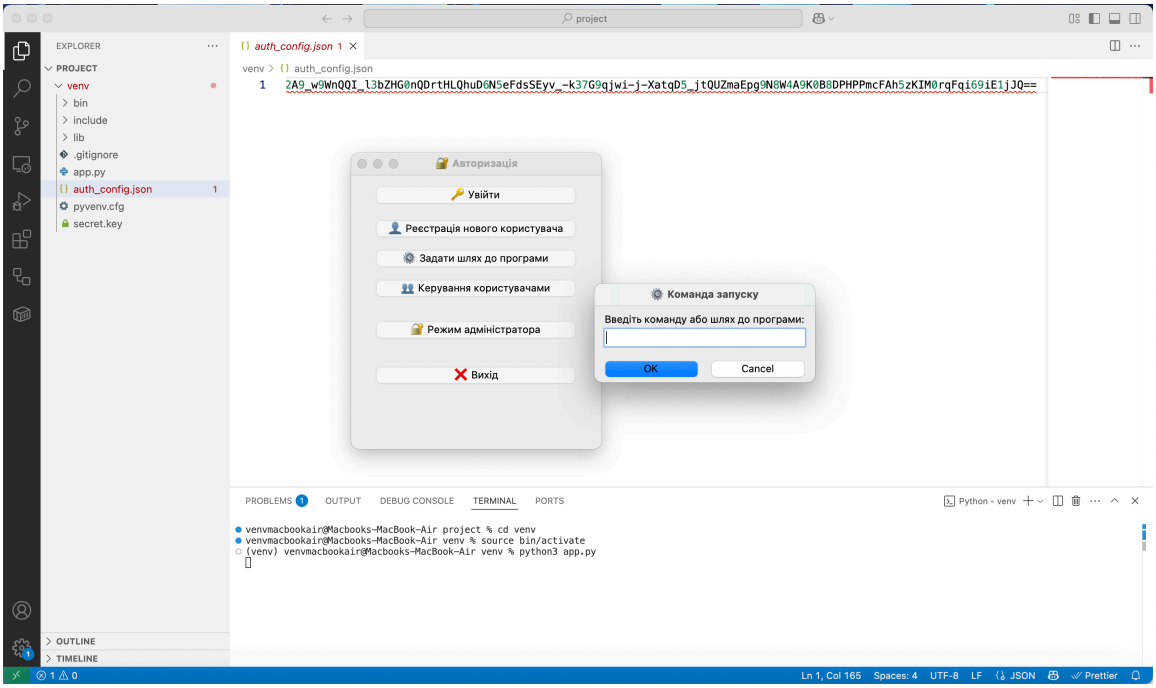


Рисунок 1.22. Результат тестування встановлення шляху до запуску застосунку під час авторизації

.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У дипломному проєкті розроблено програмний модуль аутентифікації користувача на основі поведінкового патерну. Модуль аналізує особливості введення пароля, зокрема часові інтервали між натисканнями клавіш. Забезпечується підвищений рівень безпеки та зручність використання. Якість модуля оцінюється за критеріями надійності, точності та продуктивності.

2.2 Визначення трудомісткості розробки ПЗ

Тривалість розробки програмного продукту залежить від його обсягу, складності, кваліфікації розробників і встановлених ринком термінів. Метод структурної аналогії дозволяє оцінити обсяг у тисячах умовних машинних команд на основі подібного програмного забезпечення.

Табл. 2.1 містить аналоги ПЗ з подібними функціями; обраний варіант виділено сірим.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, умовних. машинних командах**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП введення інформації	1060 – 5750

Після вибору аналога з обсягом Vo (умовні машинні команди), трудомісткість визначається за табл. 2.2.

Таблиця 2.2. Обсяг ПП

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306
6.00	330
7.00	357

Продовження таблиці 2.1

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
8.00	385
9.00	414
10.00	445

На основі отриманого значення за довідником визначають укрупнену норму часу, скориговану коефіцієнтом $К_к$ = 0,7–0,8 для умов розробки на комп'ютері:

$$T_{ар} = 262 \text{ x } 0,8 = 209,6 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість визначається для кожного етапу окремо, з урахуванням складності, новизни та використання стандартних модулів, за відповідними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.2)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.3)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.4)$$

Для розрахунку використовуються такі коефіцієнти:

- Li – частка і-го етапу (табл. 2.3);
- Кн – коефіцієнт новизни (табл. 2.4);
- Кт – коефіцієнт використання типових програм (табл. 2.5).

Наш варіант виділено сірим.

Таблиця 2.3. Питомі коефіцієнти трудомісткості стадії у загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4. Значення коефіцієнта новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово новий ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8

Продовження таблці 2.4

Код ступеня новизни	Ступінь новизни	Значення $К_н$
В	ПП, що має аналог	0,7

Таблиця 2.5. Значення коефіцієнта використання типових програм

Ступінь охоплення реалізованих функцій розробленого ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість для всіх етапів і зводимо у табл. 2.6:

Трудомісткість технічного завдання:

$$T_{тз}=Ta*L_1*K_н=209,6*0,12*0,8= 20,12 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки технічного проєкту:

$$T_{тп}=Ta*L_2*K_н =209,6*0,15*0,8 = 25,15 \text{ (люд/годин)} \quad (2.3)$$

Трудомісткість розробки робочого проєкту:

$$T_{рп}=Ta*L_3*K_н*K_т=209,6*0,58*0,8*0,7= 68,07 \text{ (люд/годин)} \quad (2.4)$$

Для розрахунків визначили обсяг документації по етапах:

- технічне завдання $N_{тз}$=2 (стор);
- розробка ТП $N_{тп}$=45 (стор);
- розробка робочого проєкту $N_{рп}$= 4 (стор);
- пояснювальна записка відповідно $N_{пз}$=9 (стор).

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин		
1.ТЗ	$Т_{Ртз}$=20,12	$Т_{кк}=0,7*N_{тз}$= 0,7*2 =1,4	$Т_{нк}=0,15*N_{тз}$ =0,15*2=0,30
2.Розробка ТП	ТРтп=25,15	Ткк=0,7*Nтп= 0,7*45=31,5	Тнк=0,15*Nтп =0,15*45=6,75
3.Розробка РП	ТРрп= 68,07	Ткк=0,7*Nрп= 0,7*4=2,8	Тнк=0,15*Nрп= 0,15*4=0,6
4.Розробка ПЗ	Тпз=1,5*Nпз= 1,5*9 =13,5	Ткк=0,7*Nтз= 0,7*9=6,3	Тнк=0,15*Nпз= 0,15*9 =1,35

Продовження таблиці 2.6

Усього, в т.ч.:	177,84		
- на розробку	Тр=126,84		
- контроль керівника		Ткк=42,00	
- нормоконтроль			Тнк=9,00

2.3 Розрахунок ціни програмного продукту

Розраховуємо основну зарплату виконавців, матеріальні та загальні витрати на розробку ПП. Зарплата наведена в табл. 2.7. З 1 січня 2025 року мінімальна місячна зарплата – 8000 грн, погодинна ставка – 48 грн (згідно зі ст. 8 Закону про Держбюджет України).

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	126,84	60,00	7 610,4
2.Контроль керівника	42,00	120,00	5 040,00
3.Нормоконтроль	9,00	120,00	1080,00
Усього	-	-	Зо= 13 730,4

Розрахуємо матеріальні витрати на розробку ПП та наведемо їх у табл. 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	70	5.0	350,0
Разом	-	-	-	$В_{мі}$ =350,0
Транспортно – заготівельні Витрати (10%)				40,00
Усього				$В_м=В_{мі}+В_{тр\_з=}$ 390,00

За отриманими даними складена калькуляція планової собівартості ПП, наведена в табл. 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	390,00	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	13 730,4	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	13 73,04	$З_д = 0.1 * З_о = 13\ 730,4*0,1$
4.Відрахування до єдиного фонду соціального внеску	3 322,75	$В_{є.с.в.} = 0.22 * (З_о + З_д) = 0,22*$ (13 730,4+13 73,04)
5. Накладні витрати	5 492,16	$В_{нак.} = 0.4 * З_о = 0.4 * 13\ 730,4$
6. Повна собівартість	27280,57	$С_{пов} = В_м + З_о + З_д + В_{є.с.в.} + В_{нак.} =$ 390,00+13 730,4+13 73,04+3 322,75+ 5 492,16

Розмір прибутку розраховується за формулою:

$$П=(С_п*Р)/100=(24\ 308,71*10)/100=24\ 30,87 \text{ грн.} \quad (2.5)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна розраховується за формулою:

$$Ц_о=С_п+П=24\ 308,71+24\ 30,87=26\ 739,58 \text{грн.} \quad (2.6)$$

За отриманими даними, ціна реалізації ПП за формулою становить:

$$Ц_р=Ц_о+ПДВ=26\ 739,58+26\ 739,58*0.2=32\ 087,49 \text{ грн.} \quad (2.7)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Основні положення

Безпека праці на підприємстві може бути забезпечена на належному рівні лише за умови всебічного дотримання вимог чинного трудового законодавства, державних стандартів України, а також нормативів і правил, спрямованих на збереження здоров'я працівників. Особливе значення при цьому має виконання організаційних заходів з охорони праці, а також дотримання трудової та виробничої дисципліни з боку працівників.

У цьому розділі дипломного проєкту для аналізу умов праці обрано робоче місце спеціаліста з кібербезпеки.

3.2 Аналіз дії небезпечних та шкідливих чинників на працівника під час роботи

Забезпечення безпечних та здорових умов праці значною мірою визначається коректною оцінкою шкідливих і небезпечних виробничих чинників. Однакові за складністю зміни в організмі людини можуть бути спричинені різноманітними факторами. Серед них — умови виробничого середовища, зокрема високий рівень шуму, підвищена температура повітря, недостатнє або відсутнє освітлення робочої зони, вплив електричного струму, наявність статичної електрики тощо. Також негативний вплив можуть мати надмірні фізичні чи розумові навантаження, психоемоційне перенапруження, а іноді — поєднання цих чинників..

3.3 Вимоги до виробничого середовища

На робочому місці програміста мають бути забезпечені умови, що сприяють безпечній та ефективній роботі. Створення безпечних умов праці передбачає впровадження заходів з безпеки виробничих процесів, які були обґрунтовані та затверджені у технологічному розділі дипломного проєкту.

3.3.1 Мікроклімат

Висока температура повітря негативно впливає на функціональний стан

людини. Хоча значне тепловиділення дисплеями досягає критичних значень переважно в найспекотніший період року, створення комфортних теплових умов повинно забезпечуватися постійно.

Параметри мікроклімату в приміщеннях мають враховувати особливості технологічного процесу, пов'язаного з роботою на комп'ютерах, і відповідати нормативним вимогам, встановленим в Україні. Згідно з положеннями ДСанПіН 3.3.2-007-98, у холодну пору року температура повітря повинна становити 22–24 °С, швидкість його руху — 0,1 м/с, а відносна вологість — 40–60 %. При цьому температура повітря може змінюватися в межах 21–25 °С за умови стабільності інших мікрокліматичних показників.

У теплий період року нормативи дещо відрізняються: температура повітря повинна бути в межах 23–25 °С, швидкість повітряного потоку — 0,1–0,2 м/с, а відносна вологість — також у межах 40–60 %.

Оптимальний рівень аероіонізації повітря в зоні дихання користувача визначається концентрацією легких іонів обох знаків у межах від 150 до 5000 на 1 см³ повітря.

На якість повітря в робочій зоні істотно впливають такі засоби, як примусова вентиляція, захисні екрани із заземленням та використання іонізаторів повітря..

3.3.2 Шум

Джерелами ряду звукових коливань, що охоплюють як чутний, так і ультразвуковий діапазони, можуть виступати деякі відеодисплейні термінали (ВДТ). Такий шум чинить несприятливий вплив на користувача, особливо при тривалому контакті. У працівників, чия діяльність пов'язана з обробкою інформації, це призводить до зниження розумової працездатності, збільшення кількості помилок, розвитку зорової втоми, порушення кольоросприйняття, появи головного болю та ослаблення концентрації уваги.

Нормативне обмеження рівня шуму на робочому місці становить 50 дБ. Основними методами боротьби з шумовим впливом є усунення або зменшення джерел шуму на етапі проєктування, впровадження звукопоглинальних матеріалів, а також раціональна організація простору виробничих приміщень.

3.3.3 Освітлення

Освітлення в приміщеннях, де працюють із персональними комп'ютерами, має бути комбінованим — включати як природне, так і штучне світло. Природне освітлення повинне надходити збоку та відповідати вимогам ДБН В.2.5-28-2006 «Природне і штучне освітлення».

При використанні природного освітлення слід передбачити сонцезахисні засоби для зменшення різниці яскравостей між світлом із вікна та яскравістю екрана монітора. До таких засобів належать металізовані захисні плівки або жалюзі з вертикальними ламелями, які можна регулювати.

Штучне освітлення реалізується у вигляді комбінованої системи з використанням світильників загального призначення, оснащених люмінесцентними лампами. На робочих місцях освітлення повинно бути рівномірним, бажано з розсіяним або відбитим характером світлорозподілу. Не допускається виникнення світлових відблисків від екрана, клавіатури чи інших елементів комп'ютера, які спрямовані в очі користувача.

Основними джерелами штучного світла є люмінесцентні лампи типу ЛБ. У порівнянні з лампами розжарювання вони мають низку переваг: світловий спектр наближений до природного, світловіддача в 2–5 разів вища, а строк експлуатації може сягати 10 тисяч годин. Лампи розжарювання дозволяється застосовувати лише в місцевому освітленні. Освітленість на робочих місцях повинна становити в межах 300–500 лк.

3.3.4 Електробезпека

Сила струму, що проходить через тіло людини, визначається прикладеною напругою та опором ділянки тіла, через яку протікає струм. У нашому випадку джерелом напруги є мережа змінного струму з напругою 229 В, яка регламентується стандартом ГОСТ 25861-83.

Щоб уникнути ураження електричним струмом, необхідно суворо та в повному обсязі дотримуватись правил безпечного виконання робіт та вимог технічної експлуатації. Потрібно повністю виключити можливість доступу оператора до частин обладнання, що перебувають під небезпечною напругою, до

відкритих провідних частин, які працюють на низькій напрузі, але не мають захисного заземлення. Крім того, електроживлення персональної електронно-обчислювальної машини (ПЕОМ) слід підключати через розетку за допомогою спеціальної вилки із заземлювальним контактом.

3.3.5 Організація робочого місця програміста

Всі конструктивні елементи робочого місця з персональним комп'ютером, а також їх розміщення повинні відповідати ергономічним вимогам, враховуючи специфіку та характер трудової діяльності працівника, згідно з положеннями ДСанПіН 3.3.2.-007-98. Обладнання і організація робочого простору мають бути такими, щоб забезпечувати комфортні та безпечні умови праці.

Робоче місце повинно мати конструкцію, що відповідає антропометричним, фізіологічним і психологічним нормам, а також особливостям виконуваної роботи. Меблі мають надавати можливість для індивідуального налаштування відповідно до параметрів користувача, що дозволяє підтримувати оптимальну робочу позу. Поверхня столу повинна мати матове покриття. Дисплей має бути розташований таким чином, щоб його верхній край знаходився на рівні очей, а відстань до екрана становила близько 70 см (допустимі межі – від 60 до 90 см). Частота оновлення зображення на моніторі має становити 100 Гц, що перевищує мінімально допустимий показник у 70 Гц.

З метою зниження нервово-емоційного навантаження, зменшення втоми, покращення кровообігу головного мозку, профілактики гіподинамії та попередження виснаження, рекомендується впроваджувати виконання спеціального комплексу вправ. Ці вправи передбачені Державними санітарними правилами і нормами роботи з візуальними терміналами електронно-обчислювальних машин (ДСанПіН 3.3.2.007-98).

Працівники, які використовують ПК у своїй діяльності, підлягають обов'язковим медичним оглядам: первинним – при прийомі на роботу, а також періодичним – протягом усього періоду працевлаштування, згідно з наказом МОЗ України № 45. Основними показниками для визначення придатності до роботи з комп'ютерною технікою є функціональний стан зорового апарату: гострота зору,

показники рефракції, функції бінокулярного зору тощо. Додатково при оцінюванні враховується загальний стан здоров’я працівника.

3.4 Пожежна безпека

Пожежі завдають значної шкоди виробничим будівлям: руйнують конструкції, знищують матеріали та готову продукцію, виводять з ладу обладнання, а також надовго припиняють виробничі процеси в цехах. Для забезпечення протипожежного захисту приміщень використовуються автоматичні установки пожежної сигналізації, наявні засоби пожежогасіння, будівельні конструкції з визначеними межами вогнестійкості, а також налагоджена система своєчасної евакуації працівників.

До основних засобів боротьби з вогнем належать внутрішні пожежні водопроводи (пожежні крани – ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок та інші пожежогасильні матеріали.

Кожен громадянин України має право на захист свого життя і здоров’я від пожеж, аварій, катастроф, стихійних лих та інших небезпечних подій. Це право гарантується державою, яка проводить відповід.

3.5 Підведення підсумків

Впровадження ефективних заходів з охорони праці сприяє зменшенню травматизму та покращенню безпеки на робочих місцях. Дотримання нормативних вимог і використання сучасних засобів захисту є запорукою збереження здоров’я працівників. Систематична підготовка та навчання персоналу підвищують рівень відповідальності та готовність до реагування у разі надзвичайних ситуацій.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено програмний модуль аутентифікації користувача на базі поведінкового патерну, що успішно інтегрує традиційні методи ідентифікації з аналізом поведінкових особливостей. Запропонована стратегія аутентифікації, яка поєднує парольний захист, хешування SHA-256 та поведінковий аналіз, дозволила суттєво підвищити рівень безпеки системи. Впроваджені алгоритми поведінкового патерну показали високу точність у розпізнаванні користувача, що зменшує ймовірність несанкціонованого доступу. Розроблений модуль адаптивний до різних сценаріїв використання та може бути інтегрований у широкий спектр інформаційних систем.

Розроблений користувальницький графічний інтерфейс на основі Tkinter забезпечує зручність і простоту взаємодії користувача з модулем, що позитивно впливає на загальний досвід використання системи. Застосування мови Python гарантувало кросплатформеність та гнучкість у подальшому розширенні функціоналу. Інтерфейс відрізняється інтуїтивністю і відповідає сучасним вимогам ергономіки, що сприяє швидкому освоєнню користувачами.

Впровадження симетричного шифрування AES забезпечило високий рівень захисту конфіденційних даних користувачів як при збереженні, так і при передачі інформації. Проведене тестування підтвердило ефективність та надійність реалізованого рішення у різних умовах експлуатації. Результати експериментальних досліджень демонструють стабільну роботу системи та відповідність вимогам безпеки сучасних інформаційних технологій.

Створений модуль аутентифікації є сучасним і безпечним інструментом, який може бути використаний для захисту інформаційних систем від несанкціонованого доступу. Запропонований підхід відкриває перспективи подальшого розвитку методів аутентифікації на основі поведінкових патернів та інтеграції їх у різноманітні програмні продукти.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ємельянов С. Л. Основи інформаційної безпеки [Текст] : навч. посіб. / С. Л. Ємельянов. – Київ : Центр навчальної літератури, 2019. – 256 с. – ISBN 978-617-7609-35-2 (irbis-nbuv.gov.ua)
2. Ковальчук Л. В. Математичні методи криптографії [Електрон. навч. посіб.] / Л. В. Ковальчук. – Київ : Національний технічний ун-т України «КПІ», 2020. – 200 с. – ISBN 978-966-02-4975-8 (irbis-nbuv.gov.ua)
3. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак [та ін.]. – Львів : Марченко Т. В., 2024. – 370 с. – ISBN 978-617-7937-91-2 (profbook.com.ua)
4. BehavioSec [Електронний ресурс] – Режим доступу: https://www.behaviosec.com/behaviosense (дата звернення: 21.05.2025)
5. TypingDNA [Електронний ресурс] – Режим доступу: https://www.typingdna.com/ (дата звернення: 21.05.2025)
6. Zighra SensifyID [Електронний ресурс] – Режим доступу: https://zighra.com/sensifyid/ (дата звернення: 21.05.2025)
7. BioCatch [Електронний ресурс] – Режим доступу: https://www.biocatch.com/ (дата звернення: 22.05.2025)
8. Plurilock [Електронний ресурс] – Режим доступу: https://www.plurilock.com/ (дата звернення: 22.05.2025)
9. Python Software Foundation. Python 3 Documentation [Електронний ресурс] – Режим доступу: https://docs.python.org/3/ (дата звернення: 22.05.2025)

ДОДАТОК А. Фрагмент програмного коду модулю аутентифікації та шифрування

```
import tkinter as tk
from tkinter import simpledialog, messagebox
import json
import time
import hashlib
import os
import subprocess
from datetime import datetime
from cryptography.fernet import Fernet

CONFIG_PATH = "auth_config.json"
KEY_PATH = "secret.key"

def load_or_create_key():
    if not os.path.exists(KEY_PATH):
        key = Fernet.generate_key()
        with open(KEY_PATH, "wb") as f:
            f.write(key)
    else:
        with open(KEY_PATH, "rb") as f:
            key = f.read()
    return Fernet(key)

fernet = load_or_create_key()

default_config = {
    "users": {},
    "launch_command": "calculator.exe" if os.name == "nt" else "open -a Calculator"
}

def load_config():
    if not os.path.exists(CONFIG_PATH):
        save_config(default_config)
        return default_config
    with open(CONFIG_PATH, "rb") as f:
        encrypted_data = f.read()
    try:
        decrypted_data = fernet.decrypt(encrypted_data)
        return json.loads(decrypted_data)
    except Exception:
        messagebox.showerror("❌", "Файл конфігурації пошкоджений або ключ не підходить.")
        return None

def save_config(data=None):
    if data is None:
        data = config
    json_data = json.dumps(data, indent=4).encode()
    encrypted_data = fernet.encrypt(json_data)
    with open(CONFIG_PATH, "wb") as f:
        f.write(encrypted_data)

def hash_text(text):
    return hashlib.sha256(text.encode()).hexdigest()

config = load_config()
if config is None:
    exit()
```

```
admin_mode = False
admin_buttons = []

def toggle_admin_mode():
    global admin_mode
    today = datetime.now().strftime("%d-%m-%Y")
    today_hash = hash_text(today)

    entered = simpledialog.askstring("🔐 Режим адміністратора", "Введіть пароль адміністратора:")
    if not entered:
        return

    if hash_text(entered.strip()) == today_hash:
        admin_mode = True
        for b in admin_buttons:
            b.config(state="normal")
        messagebox.showinfo("✅", "Режим адміністратора активовано.")
    else:
        messagebox.showerror("🚫", "Невірний пароль адміністратора.")

def register_user_admin():
    words = []
    timings = []

    for i in range(3):
        messagebox.showinfo("⌨️", f"Введіть слово #{i+1}")
        start = time.time()
        word = simpledialog.askstring("Слово", f"Введіть слово #{i+1}")
        end = time.time()

        if not word:
            return

        words.append(word.strip())
        timings.append(round(end - start, 3))

    user_key = " ".join(words)

    if user_key in config["users"]:
        messagebox.showerror("⚠️", "Користувач із таким набором слів вже існує.")
        return

    backup_password = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
    if not backup_password:
        return

    config["users"][user_key] = {
        "typing_pattern": timings,
        "backup_password_hash": hash_text(backup_password)
    }

    save_config()
    messagebox.showinfo("✅", "Користувача успішно зареєстровано.")

    global admin_mode
    admin_mode = False
    for b in admin_buttons:
        b.config(state="disabled")

def set_program_path():
    cmd = simpledialog.askstring("⚙️ Команда запуску", "Введіть команду або шлях до програми:")
```

```
    if cmd:
        config["launch_command"] = cmd
        save_config()
        messagebox.showinfo("✅", "Шлях збережено.")

def try_login():
    words = []
    timings_input = []

    for i in range(3):
        messagebox.showinfo("⌨️", f"Введіть слово #{i+1}")
        start = time.time()
        word = simpledialog.askstring("Слово", "Введіть слово:")
        end = time.time()

        if not word:
            return

        words.append(word.strip())
        timings_input.append(round(end - start, 3))

    user_key = " ".join(words)

    if user_key not in config["users"]:
        messagebox.showerror("❌", "Користувача з таким набором слів не знайдено.")
        return

    stored = config["users"][user_key]["typing_pattern"]
    if stored is None or len(stored) != len(timings_input):
        messagebox.showerror("❌", "Профіль не містить коректних даних.")
        return

    diffs = [abs(t1 - t2) for t1, t2 in zip(timings_input, stored)]
    print("Різниця часу набору:", diffs)  # Для відладки

    if all(d <= 0.3 for d in diffs):
        messagebox.showinfo("✅", "Авторизація успішна.")
        try:
            subprocess.Popen(config["launch_command"], shell=True)
        except Exception as e:
            messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
    else:
        retry_backup = messagebox.askyesno("🚫", "Стиль введення не збігається. Авторизуватись паролем?")
        if retry_backup:
            pw = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
            if pw and hash_text(pw) == config["users"][user_key]["backup_password_hash"]:
                messagebox.showinfo("✅", "Авторизація успішна.")
                try:
                    subprocess.Popen(config["launch_command"], shell=True)
                except Exception as e:
                    messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
            else:
                messagebox.showerror("❌", "Невірний пароль.")

def manage_users():
    if not config["users"]:
        messagebox.showinfo("Інформація", "Користувачі відсутні.")
        return

    win = tk.Toplevel(root)
    win.title("👥 Керування користувачами")
```

```
    win.geometry("300x300")

    user_listbox = tk.Listbox(win)
    user_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    for user in config["users"].keys():
        user_listbox.insert(tk.END, user)

    def delete_user():
        selected = user_listbox.curselection()
        if not selected:
            messagebox.showwarning("⚠️", "Виберіть користувача для видалення.")
            return
        user_to_delete = user_listbox.get(selected[0])
        if       messagebox.askyesno("Підтвердження",       f"Видалити       користувача
'{user_to_delete}'?"):
            del config["users"][user_to_delete]
            save_config()
            user_listbox.delete(selected[0])
            messagebox.showinfo("✅", f"Користувача '{user_to_delete}' видалено.")

    btn_delete = tk.Button(win, text="🗑️ Видалити користувача", command=delete_user)
    btn_delete.pack(pady=5)

root = tk.Tk()
root.title("🔐 Авторизація")
root.geometry("320x350")

tk.Button(root, text="🔑 Увійти", width=25, command=try_login).pack(pady=10)

btn_admin_register = tk.Button(root, text="👤 Реєстрація нового користувача", width=25,
state="disabled", command=register_user_admin)
btn_admin_register.pack(pady=5)

btn_admin_setpath = tk.Button(root, text="⚙️ Задати шлях до програми", width=25,
state="disabled", command=set_program_path)
btn_admin_setpath.pack(pady=5)

btn_admin_manage = tk.Button(root, text="👥 Керування користувачами", width=25,
state="disabled", command=manage_users)
btn_admin_manage.pack(pady=5)

tk.Button(root,         text="🔐         Режим         адміністратора",         width=25,
command=toggle_admin_mode).pack(pady=20)

admin_buttons.extend([btn_admin_register, btn_admin_setpath, btn_admin_manage])

tk.Button(root, text="❌ Вихід", width=25, command=root.destroy).pack(pady=10)

root.mainloop()
```

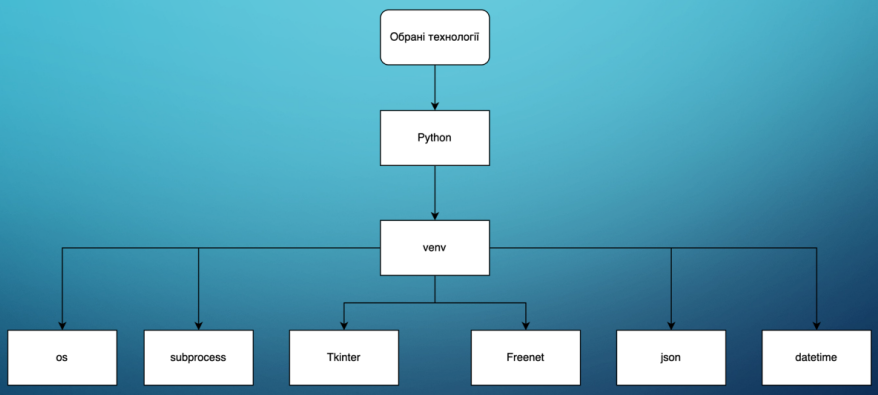
ДОДАТОК Б. Слайди мультимедійної презентації

РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ АУТЕНТИФІКАЦІЇ КОРИСТУВАЧА НА БАЗІ ПОВЕДІНКОВОГО ПАТЕРНУ

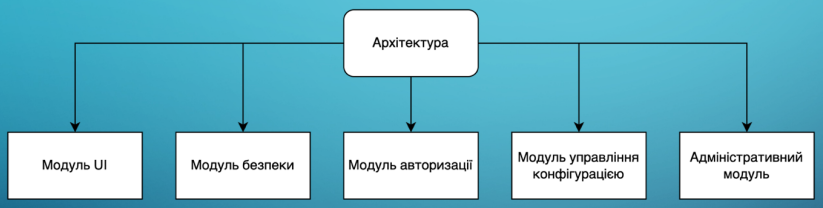
РОЗРОБИВ: СТУДЕНТ ГРУПИ 4КБ-08 ШУКАЛО М.Є.

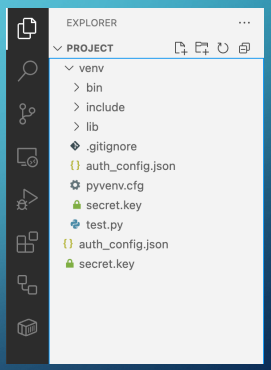
КЕРІВНИК: ГАДЖИЄВ М.М.

ПЕРЕЛІК ОБРАНИХ ТЕХНОЛОГІЙ ЗАСТОСУНКУ

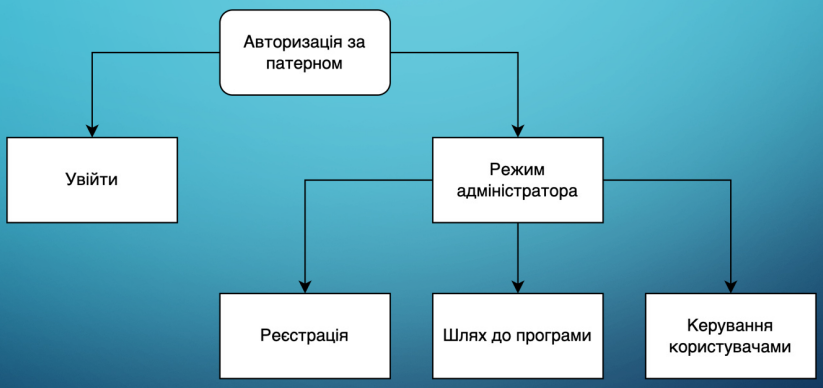


АРХІТЕКТУРА ПІДМОДУЛІВ ЗАСТОСУНКУ



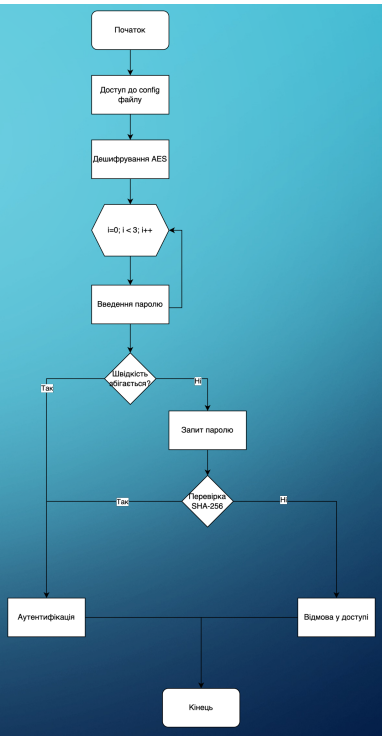


UI/ UX МАПА КОРИСТУВАННЯ ЗАСТОСУНКОМ



СТРАТЕГІЯ РЕЄСТРАЦІЇ КОРИСТУВАЧА

СТРАТЕГІЯ АУТЕНТИФІКАЦІЇ КОРИСТУВАЧА



ФОРМУЛА ПЕРЕВІРКИ ПАТЕРНУ

diffs = [abs(t1 - t2) for t1, t2 in zip(timings_input, stored)], де

- **timings_input** — список часу (затримки) введення слів користувачем під час поточної сесії (скільки часу зайняло введення кожного слова).
- **stored** — список збережених часу введення, які раніше були записані для цього користувача (еталонний шаблон).
- **zip(timings_input, stored)** — поєднує два списки в пари (t1, t2), де t1 — час введення з поточної сесії, t2 — відповідний час із збереженого шаблону.
- **t1 - t2** — різниця між поточним часом введення і збереженим часом для одного слова.
- **abs(...)** — абсолютне значення різниці, щоб врахувати тільки величину відхилення, незалежно від напрямку.
- **diffs** — новий список, який містить абсолютні відхилення часу введення кожного слова між поточним користувачем і шаблоном.

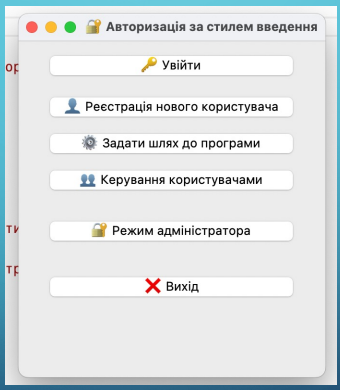
ФРАГМЕНТ КОДУ ЛОГІНУ

```
if user_key not in config["users"]:
    messagebox.showerror("❌", "Користувача з таким набором слів не знайдено.")
    return

stored = config["users"][user_key]["typing_pattern"]
if stored is None:
    messagebox.showerror("❌", "Профіль не містить даних.")
    return

diffs = [abs(t1 - t2) for t1, t2 in zip(timings_input, stored)]
if all(d <= 0.4 for d in diffs):
    messagebox.showinfo("✅", "Авторизація успішна.")
    try:
        subprocess.Popen(config["launch_command"], shell=True)
    except Exception as e:
        messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
else:
    retry_backup = messagebox.askyesno("🚫", "Стиль введення не збігається. Авторизуватись паролем?")
    if retry_backup:
        pw = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
```

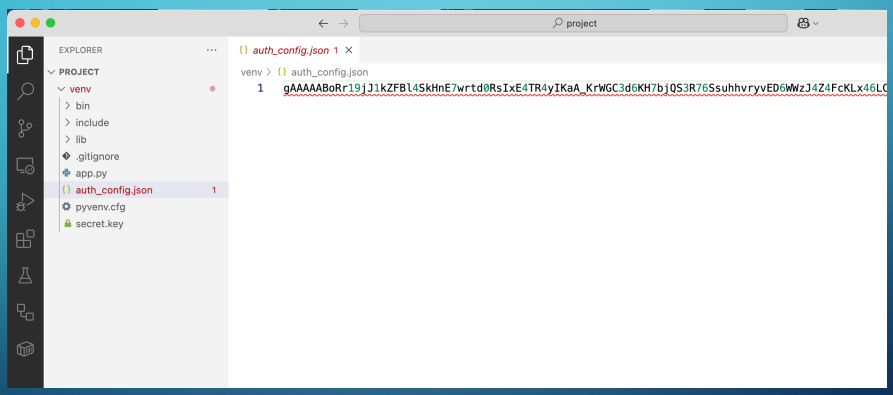
ГРАФІЧНИЙ ІНТЕРФЕЙС



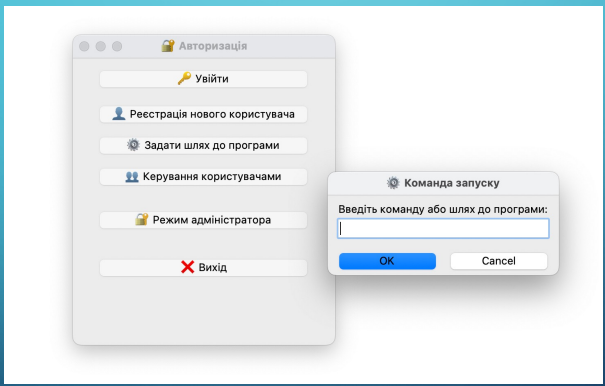
ХЕШУВАННЯ ПАРОЛЮ SHA-256

```
    retry_backup = messagebox.askyesno("🚫", "Стиль введення не збігається. Авторизуватись паролем?")
    if retry_backup:
        pw = simpledialog.askstring("🔑 Резервний пароль", "Введіть резервний пароль:", show="*")
        if pw and hash_text(pw) == config["users"][user_key]["backup_password_hash"]:
            messagebox.showinfo("✅", "Авторизація успішна.")
            try:
                subprocess.Popen(config["launch_command"], shell=True)
            except Exception as e:
                messagebox.showerror("🚫", f"Не вдалося запустити програму: {e}")
        else:
            messagebox.showerror("❌", "Невірний пароль.")
```

ШИФРУВАННЯ ДАНИХ ЗАСТОСУНКУ AES



АВТОРИЗАЦІЯ (ЗАДАТИ ШЛЯХ ДО ПРОГРАМИ)



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Шукала Мирослава Євгеновича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Безпека комп'ютерних систем та мереж»*

Керівник дипломного проекту (роботи) *Гаджиєв Матин Магсудович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну*

Обсяг розрахунково-пояснювальної записки *74* сторінок

Обсяг графічної (презентаційної) частини *12* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі аутентифікації та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 12 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Поєднання аналізу стилю введення з резервним паролем та захистом даних за допомогою AES шифрування. Використання відкритих технологій: Python, Tkinter, бібліотеки cryptography, hashlib.*

Можливість запуску захищеної програми після авторизації підвищує прикладну цінність розробки.

д) основні недоліки дипломного проекту

Поріг допустимого відхилення в таймінгу фіксований та не адаптується до змін у поведінці користувача або якості введення. Зберігання даних у локальному JSON + AES-файлі ускладнює масштабування та багатокористувацьку роботу. Використання Tkinter обмежує сучасні можливості інтерфейсу. Деякі недоліки оформлення пояснювальної записки

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

«___» _____ 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Шукала Мирослава Євгеновича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну.*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 74 сторінки. У пояснювальній записці описано етапи розробки програмного модулю аутентифікації користувача на базі поведінкового патерну засобами python, tkinter, SHA-256 та AES. Графічна частина складається з окремих 12 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою. Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.*

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі пчрограмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS Code, python, tkinter, SHA-256, AES. .*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Гаджиєв Матин Магсудович:*

Місце роботи і посада керівника дипломного проекту *д.т.н., професор кафедри «Кібербезпеки та технічного захисту інформації» Державного університету інтелектуальних технологій і зв'язку.*

Підпис

«20» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Шукало Мирослав Євгенович,
здобувач освіти гр. 4КБ-02, та

Гаджиєв Матин Магсудович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну» (автор роботи – Шукало М.Є., керівник роботи – Гаджиєв М.М.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Шукало М.Є. /

Керівник _____ / Гаджиєв М.М. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Шумілоа Максима Сергійовича

на тему *Розробка застосунку для обміну повідомленнями з можливістю шифрування*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 18.06.2025 р. значення коефіцієнту подібності в роботі становить 12,89%, коефіцієнт цитування – 3,62%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Шукала М.Є.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/18/2025
Дата редагування 6/18/2025

Документ прийнятий

Звіт подібності

метадані

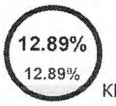
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмного модулю аутентифікації користувача на базі поведінкового патерну

Автор Науковий керівник / Експерт
Шукало Мирослав ЄвгеновичГаджиєв Матин Магсудович

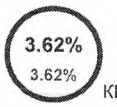
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25	12883	111720
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	16
Інтервали	A⟷	0
Мікропробіли	·	1
Білі знаки	ß	1065
Парафрази (SmartMarks)	a	65

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/8999d5af-6274-44f4-ae78-d23e08048d38/download	48 0.37 %
2	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	47 0.36 %
3	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	40 0.31 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	36 0.28 %
5	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	34 0.26 %

6	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	32 0.25 %
7	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	29 0.23 %
8	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	27 0.21 %
9	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	27 0.21 %
10	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	24 0.19 %

з домашньої бази даних (0.43 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] 6/12/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	46 (3) 0.36 %
2	Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	10 (1) 0.08 %

з програми обміну базами даних (0.26 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка та імплементація системи аутентифікації та авторизації з використанням JWT та Passport.js [redacted] 5/30/2024 Uzhhorod National University (Department)	10 (1) 0.08 %
2	КПІ_2025_Служенікін_3-017 [redacted] 5/26/2025 Ukrainian national aviation university (Криворізький Фаховий коледж)	10 (1) 0.08 %
3	SUMDU/out2015/Tezu_Mangelec T.pdf.txt [redacted] 7/20/2019 Sumy State University (SUMDU)	7 (1) 0.05 %
4	2022_60730000_Andrusiak_Kateryna_Andriivna_96941 [redacted] 10/26/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	6 (1) 0.05 %

з Інтернету (12.19 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	504 (42) 3.91 %
2	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**277 (19) 2.15 %**
3	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	108 (9) 0.84 %
4	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**98 (6) 0.76 %**
5	**https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download**	**90 (3) 0.70 %**

6	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**74 (3) 0.57 %**
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	69 (8) 0.54 %
8	**https://card-file.ontu.edu.ua/bitstreams/8999d5af-6274-44f4-ae78-d23e08048d38/download**	**57 (2) 0.44 %**
9	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	56 (6) 0.43 %
10	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	40 (4) 0.31 %
11	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**36 (1) 0.28 %**
12	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**32 (2) 0.25 %**
13	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	31 (3) 0.24 %
14	https://card-file.ontu.edu.ua/bitstreams/7b1e10b9-0ac2-4b07-afc4-8cdfff7db780/download	29 (2) 0.23 %
15	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	21 (1) 0.16 %
16	https://smekni.com/a/11909/blagoustry-ta-ozelenennya-teritor-motelyu-merkury-u-m-zhidachev/	19 (1) 0.15 %
17	https://ela.kpi.ua/bitstreams/7da6fd0e-4c92-49e1-95fa-e3ee8b559445/download	15 (1) 0.12 %
18	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	10 (1) 0.08 %
19	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж» Група: 4КБ- 02

Дипломний проект здобувача освіти денної форми навчання КБ. 02.24.000.ДП

ШУКАЛА
МИРОСЛАВА ЄВГЕНОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»
Група: 4 КБ-02

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проекту на тему: _____
_____ Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах) Дипломник _____ (Шукало М.Є.)
_____ Керівник _____ (Гаджиєв М.М.) Консультанти: з економічного розділу _____ (Канський М. Ю.)
з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.) з нормоконтролю _____ (Петрашова В.І.) старший консультант _____ (Кривченко Ю.В.)