МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Домбровський Максим Олександрович

Програмна система аналізу даних для ідентифікації користувачів / Software system of data analysis for users identification

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ-41
Домбровський Максим
Олександрович

Науковий Керівник
к.т.н. Савка Н.Я.

ТЕРНОПІЛЬ-2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Програмна система аналізу даних для ідентифікації користувачів» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «бакалавр» містить 79 сторінок пояснюючої записки, 27 рисунків, 1 таблиця, додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою кваліфікаційної роботи є розробка програмного засобу для аналізу даних та ідентифікації користувача з використанням додаткового методу захисту такого як шифрування даних стандартом AES256, додаткової перевірки сканером відбитка пальця, та Google 2FA.

Розглянуто задачу ідентифікації та аутентифікації користувачів із застосуванням шифрування, та перевірки методом сканування відбитка пальця та додавання методу Google 2FA.

Охарактеризовано методи ідентифікації користувача, доданий додатковий метод підтвердження користувача та його перевірки, за допомого сканера відбитків пальців, відрегульована точність сканування та роботи сканера. Введено метод шифрування даних за допомогою стандарту AES256. Налаштовано програмне забезпечення Google Authenticator на смартфоні з операційною системою Android.

Розроблений алгоритм ідентифікації, аутентифікації методом розпізнавання відбитків та генерації шестизначного коду Google 2FA.

Розроблено структуру та інтерфейс програмного забезпечення. Додана можливість входу за допомогою логіну і паролю, також при наявності зареєстрованих методів 2FA включення перевірки. Також розроблено архітектуру ідентифікації та аутентифікації користувача.

Ключові слова: ІДЕНТИФІКАЦІЇ ТА АУТЕНТФИІКАЦІЇ, СКАНУВАННЯ ВІДБИТКУ ПАЛЬЦІВ, GOOGLE 2FA, ЛОГІН, ПАРОЛЬ, ШИФРУВАННЯ ДАНИХ, ЗАХИСТ ДАНИХ КОРИСТУВАЧА.

RESUME

Qualification thesis “/ Software of Modeling Performance Indicators of the Enterprise Based on Artificial Neural Networks” in the specialty 123 "Computer Engineering" "Bachelor" education degree contains 79 pages of explanatory notes, 10 figures, 8 tables, 3 appendixes. The volume of graphic material is 2 sheets of A3 format.

The aim of the qualification project is the development of software for modeling performance indicators of the enterprise based on artificial neural networks with radial-basic functions.

The task of modeling and forecasting performance indicators of the enterprise with use of artificial neural networks with radial-basic functions is considered. The training of artificial neural networks of the radial type takes place in two stages: adjustment of the parameters of the hidden network layer and adjustment of synaptic weights.

The methods of training of artificial neural networks with radial-basic functions are described and it is shown that for the determination of centers of radial-basic functions we use a dynamic reconfiguration algorithm. Adjustment of weight coefficients of synaptic bonds is based on algorithms based on minimizing the mean square error.

An algorithm for the training of artificial neural networks with radial-basic functions for simulation performance indicators of the enterprise was developed.

The structure of the program module for training of artificial neural networks with radial-basic functions for modeling performance indicators of the enterprise was developed. The architecture of artificial neural network of radial type for modeling and forecasting performance indicators of the enterprise was developed.

Key words: ARTIFICIAL NEURAL NETWORKS WITH RADIAL BASIS FUNCTIONS, WEIGHTS, PERFORMANCE INDICATORS OF THE ENTERPRISE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CRM	–	Customer Relationship Management
SQL	–	Structured query language
ID	–	Identity document
ER	–	Entity-relationship
UML	–	Unified Modeling Language
ORM	–	Object-Relational Mapping
СУБД	–	Система управління базами даних
2FA	–	Two-Factor Authentication
AES	–	Advanced Encryption Standard
AWS	–	Amazon Web Services
RDS	–	Relational Database Service
CSV	–	Character-separated values
ACID	–	Atomicity, Consistency, Isolation, Durability
TOTP	–	Time-based One Time Password
MFA	–	Multi-Factor Authentication
API	–	Application Programming Interface
LGPL	–	Lesser General Public License
RTS	–	Request to Send
CTS	–	Clear to Send
USB	–	Universal Serial Bus
MVC	–	Model-View-Controller
PBKDF2	–	Password-based key derivation function

ВСТУП

Аналіз та ідентифікація користувачів є актуальною задачею, оскільки забезпечення безпеки даних, приватності та аутентифікація користувачів відіграють важливу роль в сучасному ІТ-середовищі. Зокрема, це стосується виявлення шахрайства, а також впровадження персоналізованих послуг та рекомендацій. Взаємозв'язок між даними користувачів, їх поведінкою та різними характеристиками часто є нелінійним, що вимагає застосування відповідного апарату моделювання для отримання адекватних аналітичних моделей.

У сучасному інформаційному суспільстві, обсяги збирання та аналізу даних постійно зростають. Це відкриває нові можливості для вирішення різноманітних завдань, таких як розробка систем ідентифікації користувачів, безпека даних, а також більш цілеспрямоване надання послуг та рекомендацій. У цьому контексті програмна система аналізу даних для ідентифікації користувачів стає ключовим інструментом для розробки ефективних технологій і методів роботи з користувачами.

Зважаючи на вищезазначене, кваліфікаційна робота присвячена розробці програмної системи аналізу даних для ідентифікації користувачів, забезпечуючи збільшення точності виявлення та аутентифікації користувачів на основі їх поведінки, взаємодії з сервісами та іншими параметрами. Метою роботи є проектування та реалізація програмної системи, яка буде використовуватись для забезпечення більш безпечного та ефективного використання ресурсів інтернету та інших послуг.

Для досягнення поставленої мети, в кваліфікаційній роботі будуть досліджені певні аспекти.

1. Огляд існуючих систем аналізу даних для ідентифікації користувачів та їх особливостей.

2. Вивчення теоретичних основ аналізу даних та ідентифікації користувачів.

3. Розробка методології та алгоритмів для ідентифікації користувачів заснованих на аналізі даних та реєстрації їх.

4. Проектування програмної системи для реалізації вибраних методів ідентифікації користувачів.

5. Тестування та оцінка ефективності розробленої програмної системи на практичних прикладах.

Об’єкт дослідження – процес ідентифікації користувачів.

Предмет дослідження – алгоритми аналізу даних для ідентифікації користувачів.

Практичне значення роботи – програмна система аналізу даних користувачів для їх ідентифікації у системах збору і обробки інформації.

У процесі виконання даної роботи будуть використані сучасні технічні засоби, методи та технології аналізу даних та машинного навчання, що дозволить розробити ефективну та надійну систему для ідентифікації користувачів. Результати отримані в ході виконання даної дипломної роботи матимуть важливе практичне застосування в різних галузях, таких як інформаційна безпека, електронна комерція, соціальні мережі та інші сфери, де використовуються системи ідентифікації користувачів.

У висновку дипломної роботи будуть представлені результати проведеного дослідження, аналіз ефективності розробленої програмної системи, а також рекомендації щодо можливості подальшого вдосконалення та розширення функціоналу системи. Виконання цього проекту сприятиме зростанню якості та безпеки послуг, що надаються користувачам, та сприяє розвитку технологій аналізу даних та машинного навчання в цілому.

Крім того, розроблена програмна система аналізу даних для ідентифікації користувачів буде спрямована на забезпечення прозорості та дотримання принципів конфіденційності персональних даних.

Також важливим аспектом кваліфікаційної роботи буде аналіз потенційних етичних питань, пов'язаних з використанням таких систем для ідентифікації користувачів. Особливий акцент буде зроблено на вивченні балансу між ефективністю розробленої системи та дотриманням принципів етики та конфіденційності.

Окрім того, під час розробки програмної системи буде проведено аналіз різних методів реєстрації та автентифікації, і також їх відповідності завданням ідентифікації користувачів. В результаті цього аналізу, буде вибрано найбільш ефективні методи, які забезпечать оптимальне поєднання продуктивності, точності та аналітичних можливостей.

Проведення досліджень у даному напрямку має великий потенціал для стимулювання інновацій та створення нових рішень в галузі аналізу даних, машинного навчання та штучного інтелекту. Розробка програмної системи аналізу даних для ідентифікації користувачів може послужити основою для майбутніх досліджень та розробок, спрямованих на створення більш продуктивних та безпечних систем, що відповідають вимогам сучасного інформаційного суспільства.

Основні результати дослідження опубліковано на VII науково-практичній конференції «Інтелектуальні комп’ютерні системи та мереж» [1]. Копії публікації наведено у додатку А.

1 АНАЛІЗ ЗАСОБІВ РЕЄСТРАЦІЇ, АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ТА КЕРУВАННЯ ДАНИМИ

1.1 Аналіз засобів реєстрації даних

Існує безліч методів реєстрації даних, тобто про користувача, і якщо розглядати елементарне записування інформації на листку паперу і те що зараз користувачі можуть реєструвати дані за допомогою комп’ютерної техніки, то другий варіант буде набагато надійнішим, ніж перший. Наприклад, той самий папір, раніше можливо було заховати в захищені архіви, то зараз не потрібні архіви, зараз все можливо зашифрувати в електронному вигляді. Існує багато видів та засобів реєстрації даних, наприклад, електронні таблиці.

Електронні таблиці – це інтерактивний, комп'ютерний застосунок для налагодження, аналізу та збереження даних у табличному форматі [55]. Найбільш відомі програми для створення електронних таблиць – це Microsoft Excel та Google Sheets. Електронні таблиці дозволяють користувачам створювати таблиці з даними, такі як числа, тексти, дати, формули та інші елементи. Користувачі можуть також створювати діаграми та графіки на основі даних, що дозволяє візуалізувати інформацію та зробити її більш зрозумілою. Окрім цього, електронні таблиці мають функції для обчислення, сортування та фільтрації даних. Наприклад, можна використовувати формули, щоб автоматично обчислити значення на основі даних у таблиці, сортувати та фільтрувати дані, щоб швидко знайти певні значення або групи даних. Електронні таблиці дуже корисні для збереження та організації великих обсягів даних, таких як дані про продажі, фінанси або інвентаризацію. Вони також дозволяють швидко та ефективно аналізувати дані та зробити висновки на основі цих даних.

Також існують CRM системи, наприклад такі як: SalesDrive, KeyCRM. Вони дозволяють зберігати інформацію про контакти з клієнтами, історію продажів, запити на обслуговування, контракти та іншу інформацію про

клієнтів. Вони також можуть включати функції для автоматизації маркетингових та продажних процесів, такі як автоматичні email-розсилки, кампанії на соціальних медіа та інші. CRM системи дозволяють підприємствам збільшувати продажі та покращувати обслуговування клієнтів, оскільки компанії можуть вести персоналізований підхід до кожного клієнта на основі інформації, що вони збирають в CRM системі. Наприклад, компанія може використовувати інформацію про попередні замовлення або запити на обслуговування, щоб зробити рекомендації клієнту щодо подальших покупок. CRM системи можуть бути корисні для різних типів компаній, від малих бізнесів до великих корпорацій. Вони дозволяють зберігати та організовувати великі обсяги даних про клієнтів та забезпечують зручний та ефективний спосіб управління відносинами з клієнтами. І головним із способів реєстрації та зберігання даних є бази даних те що і буде використовуватися у створенні програмного забезпечення для створення програмної системи аналізу даних для ідентифікації користувачів.

Бази даних – це організована колекція даних, яка зберігається та управляється за допомогою спеціального програмного забезпечення. Бази даних дозволяють зберігати та організовувати великі обсяги даних у логічній структурі. Вони дозволяють створювати таблиці, поля та відносини між ними, що дозволяє ефективно зберігати та організовувати дані. Бази даних також дозволяють швидко та легко знаходити та змінювати дані, забезпечуючи зручний спосіб управління даними. Існує безліч різних програм для створення та управління базами даних, таких як Microsoft Access, MySQL, Oracle та багато інших [2]. Кожна з цих програм має свої переваги та недоліки, але вони всі дозволяють зберігати та організовувати дані ефективним способом. Бази даних можуть бути корисні для різних типів компаній та організацій, від малих бізнесів до великих корпорацій. Наприклад, вони можуть використовуватися для зберігання даних про клієнтів, продукти, складський облік та інші види даних. Бази даних також можуть використовуватися для забезпечення безпеки даних та забезпечення їх захисту від несанкціонованого доступу. Бази даних

можуть мати різні типи організації, залежно від потреб користувачів і характеру даних. Один з найпоширеніших типів баз даних – реляційна модель.

У реляційній базі даних дані організовані у вигляді таблиць, де кожен стовпець представляє поле, а кожен рядок – запис. Відносини між таблицями встановлюються за допомогою ключів. Крім реляційних баз даних, існують також ієрархічні, мережеві, об'єктно-орієнтовані та інші типи баз даних, які підходять для специфічних сценаріїв та вимог. Окрім зберігання даних, бази даних також забезпечують можливість виконання різних операцій над ними, таких як додавання, вилучення, оновлення та пошук даних. Мови запитів, такі як SQL (Structured Query Language), використовуються для формулювання запитів до баз даних і отримання потрібної інформації [28]. Безпека баз даних є важливим аспектом. Для захисту від несанкціонованого доступу, бази даних можуть використовувати різні методи, такі як аутентифікація користувачів, авторизація доступу до даних, шифрування та аудит доступу. За допомогою баз даних компанії можуть зберігати і аналізувати великі обсяги даних для прийняття кращих управлінських рішень [27]. Вони також можуть підтримувати взаємодію з іншими програмними засобами, що дозволяє автоматизувати багато бізнес-процесів. Особливо важливим аспектом баз даних є резервне копіювання і відновлення даних. Регулярні резервні копії дозволяють запобігти втраті даних у разі випадкового видалення або пошкодження бази даних. Узагалі, бази даних є потужним інструментом для зберігання, організації та управління даними, що допомагає підвищити ефективність багатьох бізнес-процесів і забезпечити надійність даних.

Засоби реєстрації даних мають безліч переваг та допомагають підприємствам та організаціям збирати, зберігати, організовувати та аналізувати великі обсяги даних. Не тільки фізичний обсяг, але й інші визначні особливості великих даних відіграють ключову роль, підкреслюючи складність процесу їхньої обробки та аналізу. Набір критеріїв VVV (об'єм, швидкість накопичення даних і необхідність їх експрес-обробки, а також здатність переробляти дані різних форматів) було створено компанією Meta

Group у 2001 році, щоб вказати на рівну важливість управління даними в усіх трьох зазначених областях. [3]. Це дозволяє бізнесам приймати кращі управлінські рішення, покращувати взаємодію з клієнтами та забезпечувати більш ефективну роботу. Однак, важливо пам'ятати про необхідність захисту даних та забезпечення їх конфіденційності. У світі, де кіберзлочинці намагаються отримати доступ до конфіденційної інформації, важливо використовувати безпечні засоби для зберігання та обробки даних, а також дотримуватися найкращих практик з охорони даних. Крім того, варто пам'ятати про потребу у підтримці та оновленні засобів реєстрації даних. Нові програмні версії та інструменти можуть містити поліпшення, які допоможуть покращити продуктивність та ефективність роботи з даними. Також, варто навчати свій персонал користуватися засобами реєстрації даних, щоб забезпечити їх ефективне використання та оптимальне використання можливостей цих інструментів.

Однією з важливих переваг засобів реєстрації даних є можливість використання аналітики даних. Аналітика даних – це процес збору, обробки та аналізу даних з метою виявлення трендів, взаємозв'язків та патернів. Застосування аналітики даних може допомогти підприємствам зробити кращі управлінські рішення, покращити ефективність бізнесу та забезпечити більш якісну обслуговування клієнтів. Засоби реєстрації даних, такі як електронні таблиці,

CRM системи та бази даних, надають можливість використовувати аналітику даних для вивчення ринку, прогнозування продажів, аналізу потреб клієнтів та багато іншого [27]. Аналітика даних може також допомогти виявити проблемні моменти та недоліки в роботі бізнесу та надати рекомендації щодо їх вирішення [29]. Нарешті, важливо звернути увагу на розуміння та врахування етичних питань у роботі з даними. Збір, зберігання та використання даних повинні відбуватися в межах законів та рекомендацій з охорони приватності даних. Також, важливо дотримуватися етичних норм та

принципів у роботі з даними, щоб забезпечити захист прав та свобод людей та підтримати довіру до бізнесу та технологій реєстрації даних.

Таким чином, варто застосовувати засоби доступу до даних, зокрема такі, як ідентифікація та автентифікація користувача, детально які розглянемо у наступних розділах.

1.2 Аналіз програмних засобів ідентифікації користувача

Засоби ідентифікації користувача – це програмні та апаратні інструменти, які використовуються для перевірки та підтвердження ідентичності користувача перед тим, як дозволити доступ до певної системи, сервісу, додатку або пристрою. Ці засоби можуть включати різні технології, такі як логін та пароль, біометричні ідентифікатори, токени доступу, сертифікати та інші. Вони дозволяють забезпечити захист від несанкціонованого доступу до конфіденційної інформації, запобігти шахрайству та злому, а також забезпечити безпеку даних та інформації. Засоби ідентифікації користувача можуть використовуватися в багатьох різних галузях, включаючи банківський сектор, медичні установи, виробничі компанії та інші. Для кожної галузі можуть бути встановлені свої вимоги щодо захисту даних та ідентифікації користувачів. Забезпечення безпеки даних та захисту від несанкціонованого доступу є важливими аспектами для більшості компаній та організацій. Ідентифікація користувача дозволяє забезпечити безпеку даних та інформації, а також захистити бізнес від можливих фінансових втрат та втрати довіри клієнтів.

Існує декілька основних програмних засобів:

1. Логін та пароль – це найпростіший спосіб ідентифікації користувача, який використовують більшість сервісів та додатків. Користувач вводить свій логін та пароль, і система перевіряє їх правильність;

2. Біометричні ідентифікатори – такі як сканер відбитку пальця, сканер обличчя або розпізнавання голосу, дозволяють ідентифікувати користувача на основі його унікальних фізичних характеристик [26];

3. Токени доступу – це програмні засоби, які генерують унікальний код доступу, який надається користувачеві для входу в систему. Цей токен може бути надісланий на мобільний телефон або генеруватися на спеціальному пристрої;

4. Сертифікати – це електронні документи, що підтверджують ідентичність користувача. Вони можуть використовуватися для входу в захищені мережі та веб-сайти, а також для підпису електронних документів;

5. OAuth та OpenID – це стандарти, які дозволяють користувачам ідентифікуватися на додатках та веб-сайтах за допомогою своїх акаунтів на інших платформах, таких як Google, Facebook або Twitter [25];

Якщо розглядати логін та пароль то це один з найпоширеніших та найпростіших методів ідентифікації, який використовується в більшості сервісів та додатків. Користувачі створюють свій обліковий запис та вводять свій логін та пароль, щоб отримати доступ до системи, сервісу, додатку або пристрою. Система перевіряє введені дані на відповідність інформації в базі даних та надає користувачеві доступ до потрібних ресурсів. Логіни та паролі можуть бути унікальними для кожного користувача або використовуватися загальнодоступні для групи користувачів. Для забезпечення безпеки даних та інформації, важливо вимагати від користувачів складних паролів, які містять букви, цифри та символи, а також вимагати зміни пароля з регулярністю.

Однак, існують деякі недоліки цього методу ідентифікації, такі як можливість несанкціонованого доступу в разі незабезпеченості пароля, можливість підбору пароля шляхом використання спеціальних програм, а також можливість підміни логіну та пароля шляхом атак на систему. Тому важливо використовувати додаткові методи ідентифікації користувача, такі як біометричні ідентифікатори, токени доступу, сертифікати та інші, для забезпечення більш надійної та безпечної ідентифікації.

Другий пункт програмних засобів ідентифікації користувача – це біометричні ідентифікатори. Це спосіб ідентифікації на основі унікальних фізичних характеристик користувача, таких як відбиток пальця, обличчя або голос. Біометрична ідентифікація дозволяє забезпечити вищий рівень безпеки, оскільки такі фізичні характеристики унікальні для кожної людини та складно підробити. Крім того, вона зручна для користувачів, оскільки не потребує запам'ятовування паролів та інших ідентифікаційних даних. Основні види біометричних ідентифікаторів включають відбиток пальця, обличчя, розпізнавання голосу, сканер радужної оболонки ока та інші. Для біометричної ідентифікації використовуються спеціальні пристрої, які зчитують та аналізують фізичні характеристики користувача.

Однак, існують деякі проблеми з біометричною ідентифікацією, такі як невідповідність зчитаних даних, недостатня точність системи та можливість використання підроблених фізичних характеристик. Також, важливо забезпечувати захист біометричних даних та інформації, оскільки їх використання може бути дуже особистим та приватним. Біометрична ідентифікація зазвичай використовується в комплексі з іншими методами ідентифікації, такими як логін та пароль, токени доступу та інші, щоб забезпечити більш надійний та безпечний спосіб ідентифікації користувача.

Третій пункт програмних засобів ідентифікації користувача - це токени доступу. Токен доступу є програмним засобом, що генерує унікальний код, який надається користувачеві для входу в систему, сервіс, додаток або пристрій. Токен доступу може бути надісланий на мобільний телефон або генеруватися спеціальним пристроєм. Код доступу зазвичай має обмежений термін дії та може бути використаний лише один раз, що забезпечує високий рівень безпеки. Токени доступу дозволяють забезпечити безпеку даних та інформації, оскільки доступ до системи, сервісу, додатку або пристрою надається тільки за допомогою унікального токену. Крім того, вони зручні для користувачів, оскільки не потребують запам'ятовування паролів та інших ідентифікаційних даних.

Однак, існують деякі недоліки цього методу ідентифікації, такі як можливість втрати або крадіжки токену доступу, можливість перехоплення коду доступу при його надсиланні, а також можливість підроблення токену доступу. Тому важливо забезпечувати захист токенів доступу та контролювати їх використання. Токени доступу часто використовуються в комбінації з іншими методами ідентифікації, такими як логін та пароль, біометричні ідентифікатори, сертифікати та інші, щоб забезпечити більш надійний та безпечний спосіб ідентифікації користувача.

Використання токенів доступу має свої переваги. Вони дозволяють реалізувати одноразові та обмежені за часом сесії, що підвищує безпеку доступу до системи. Токени можуть також бути легко відкликані або змінювані, що дає можливість контролювати доступ користувачів до системи. Використання токенів також спрощує процес аутентифікації, оскільки користувачам не потрібно запам'ятовувати складні паролі.

У практиці, токени доступу широко використовуються в сучасних системах, сервісах та додатках для забезпечення безпеки та ефективного керування доступом користувачів.

Четвертий пункт програмних засобів ідентифікації користувача – це сертифікати. Сертифікат – це електронний документ, який містить інформацію про користувача та його публічний ключ. Сертифікати використовуються для забезпечення безпеки та конфіденційності даних, оскільки вони дозволяють користувачам передавати зашифровану інформацію один одному без можливості доступу третіх сторін до цієї інформації. Користувачі отримують сертифікати після проходження процедури перевірки особистості. Сертифікати можуть випускатися різними організаціями, такими як урядові установи, банки, комерційні організації та інші. Для перевірки дійсності сертифікату використовується публічний ключ, який міститься в сертифікаті.

Однак, існують деякі недоліки цього методу ідентифікації, такі як можливість крадіжки або підроблення сертифікату, можливість використання

старих або втрачених сертифікатів, а також можливість використання неправомірно випущених сертифікатів. Тому важливо забезпечувати захист сертифікатів та контролювати їх використання. Сертифікати часто використовуються в комбінації з іншими методами ідентифікації, такими як логін та пароль, біометричні ідентифікатори, токени доступу та інші, щоб забезпечити більш надійний та безпечний спосіб ідентифікації користувача.

П'ятий пункт програмних засобів ідентифікації користувача - це ідентифікація на основі поведінки користувача. Цей метод використовується для ідентифікації користувача на основі його унікальної поведінки, такої як стиль набору тексту, спосіб використання миші та клавіатури, швидкість набору тексту та інше. Ідентифікація на основі поведінки користувача є досить новим методом ідентифікації, який поки що мало використовується [24]. Проте він може забезпечити додатковий рівень безпеки для користувачів, оскільки така поведінка є унікальною для кожної людини і важко підробити. Ідентифікація на основі поведінки користувача відбувається шляхом збору та аналізу даних про поведінку користувача під час його взаємодії з системою, сервісом, додатком або пристроєм.

Ці дані можуть включати час роботи з системою, інтервали між діями, використання миші та клавіатури, спосіб вводу тексту та інше. Ідентифікація на основі поведінки користувача може використовуватися як самостійний метод ідентифікації або в поєднанні з іншими методами, такими як логін і пароль, біометричні ідентифікатори, токени доступу та інші, для забезпечення більш надійного та безпечного способу ідентифікації користувача. Однак, існують певні недоліки цього методу ідентифікації, такі як можливість спотворення даних про поведінку користувача у випадку зміни умов роботи з системою.

Незважаючи на це, ідентифікація на основі поведінки користувача є перспективним напрямком в галузі безпеки та ідентифікації, і може знайти широке застосування в майбутніх системах та технологіях. Ідентифікація на основі поведінки користувача має кілька переваг.

По-перше, цей метод не вимагає введення додаткових ідентифікаційних даних, таких як пароль чи пін-код. Він ґрунтується на унікальних особливостях поведінки кожного користувача, що робить його важкопідробним та складним для зловмисників.

По-друге, ідентифікація на основі поведінки користувача може бути проведена в режимі реального часу, дозволяючи виявляти аномальну або підозрілу активність та негайно реагувати на неї. Це може сприяти підвищенню безпеки системи та запобіганню несанкціонованому доступу.

Проте, ідентифікація на основі поведінки користувача також має свої обмеження та виклики. Попереднє навчання моделі поведінки користувача може бути складним завданням, оскільки необхідно зібрати достатню кількість даних про поведінку реальних користувачів для побудови точної моделі. Крім того, дані про поведінку користувача можуть бути залежні від контексту та змінюватися з часом, що може вплинути на ефективність методу [47]. Також виникає питання приватності та захисту даних, оскільки збір та аналіз поведінкових даних може порушити приватність користувача. Тому, використання цього методу повинно враховувати відповідні правові та етичні норми. В цілому, ідентифікація на основі поведінки користувача є перспективним напрямком в галузі безпеки та ідентифікації. Вона може доповнити та підсилити інші методи ідентифікації, забезпечуючи більш надійний та безпечний спосіб захисту інформації та ресурсів.

1.3 Аналіз засобів зберігання даних

Аналіз засобів зберігання даних є важливою складовою при розробці будь-якої програмної системи. Він дозволяє з'ясувати, які дані будуть зберігатися в системі, як вони будуть використовуватися та оброблятися, як будуть встановлюватися зв'язки між даними та як система буде взаємодіяти з іншими системами, що використовують ті ж самі дані. Найпоширенішим засобом зберігання даних є бази даних.

Першим кроком у процесі аналізу баз даних є визначення потреб користувачів в системі. Це дозволяє з'ясувати, які дані будуть потрібні для роботи системи, які операції будуть виконуватися з цими даними, які вимоги до безпеки та конфіденційності даних та які зв'язки між даними будуть потрібні для роботи системи.

Другим кроком є аналіз наявних даних та їх опис [23]. Це може бути виконано шляхом проведення досліджень на попередній етап розробки системи або шляхом аналізу наявної інформації в базах даних, які використовуються в даний час. Під час цього етапу слід визначити, які дані є доступними, як вони зберігаються та які зв'язки між даними вже існують.

Третім кроком є проектування бази даних. Це включає визначення структури бази даних, зв'язків між таблицями та типів даних, які будуть використовуватися для зберігання та обробки даних. Для проектування бази даних можуть бути використані різноманітні інструменти та технології, такі як ER-моделювання та UML-моделювання.

Четвертий крок полягає у створенні прототипу бази даних. Це може бути виконано шляхом створення тестової бази даних, яка дозволить випробувати та перевірити різні аспекти бази даних, такі як швидкість обробки даних, роботу з запитами та інші.

П'ятий крок полягає в аналізі результатів випробування прототипу та внесенні необхідних змін до бази даних. Це дозволить виявити та виправити

можливі помилки та недоліки прототипу, а також удосконалити його з точки зору швидкості та ефективності роботи з даними.

Шостий крок полягає у підготовці бази даних до впровадження в програмну систему. Це включає в себе створення скриптів для створення бази даних, наповнення бази даних даними та інші підготовчі роботи.

Сьомий крок – це тестування бази даних в реальних умовах роботи програмної системи. Під час цього етапу слід виконати різноманітні тестові сценарії для перевірки роботи бази даних в різних умовах. Останній крок полягає в аналізі результатів тестування та внесенні необхідних змін до бази даних для покращення її роботи в програмній системі.

Отже, аналіз баз даних для розробки програмної системи є важливим етапом у процесі розробки будь-якої програмної системи. Він дозволяє визначити необхідні дані та їх структуру, встановити зв'язки між даними та забезпечити безпеку та ефективність роботи з даними в програмній системі.

Після аналізу та проектування бази даних можна перейти до розробки програмної системи, що використовуватиме дану базу даних. Під час розробки програмної системи, розробники можуть використовувати різноманітні технології та інструменти, такі як реляційні СУБД, NoSQL бази даних, ORM фреймворки та інші. При розробці програмної системи слід звернути увагу на питання безпеки бази даних. Для забезпечення безпеки бази даних можна використовувати різноманітні методи та технології, такі як шифрування даних, контроль доступу до даних, забезпечення цілісності даних та інші. Важливим етапом у розробці програмної системи є тестування бази даних та програмної системи в цілому [46].

Тестування дозволяє виявити можливі помилки та недоліки програмної системи та внести необхідні зміни для покращення її роботи. Крім того, слід пам'ятати про питання масштабованості бази даних. Залежно від вимог до програмної системи та розміру даних, може знадобитися масштабування бази даних для забезпечення більшої продуктивності та ефективності роботи системи. Аналіз баз даних є важливим етапом у розробці будь-якої програмної

системи. Він дозволяє визначити структуру та зв'язки між даними, забезпечити безпеку та ефективність роботи з даними та підготувати базу даних до використання в програмній системі [50].

Після впровадження програмної системи важливо забезпечити моніторинг та підтримку бази даних. Моніторинг дозволяє відстежувати роботу бази даних та виявляти можливі проблеми, такі як затримки в роботі бази даних, помилки підключення та інші. Він забезпечує постійний контроль за станом бази даних, швидкістю її роботи та доступністю. Завдяки моніторингу можна вчасно виявляти аномалії та негаразди, що дозволяє попередити можливі проблеми та забезпечити стабільну та надійну роботу бази даних.

Підтримка бази даних є важливою складовою процесу після впровадження. Вона включає в себе регулярне проведення резервного копіювання бази даних, що дозволяє забезпечити збереження даних в разі непередбачуваних ситуацій, таких як видалення чи пошкодження даних. Крім того, підтримка бази даних включає в себе встановлення оновлень та патчів для програмного забезпечення бази даних, що дозволяє усунути можливі уразливості та вдосконалити безпеку та функціональність бази даних [38]. Також важливо виконувати профілактичні дії, такі як оптимізація запитів та налаштування бази даних для забезпечення оптимальної продуктивності та ефективності роботи. Моніторинг та підтримка бази даних є невід'ємною частиною її експлуатації після впровадження [20]. Вони дозволяють забезпечити стабільну та надійну роботу бази даних, мінімізувати ризики виникнення проблем та забезпечити оптимальну продуктивність системи, яка використовує дані з бази даних.

Також, слід пам'ятати про питання резервного копіювання бази даних. Резервне копіювання дозволяє зберегти дані бази даних у випадку непередбачуваної ситуації, такої як втрата даних чи пошкодження бази даних. Нарешті, при роботі з базами даних слід дотримуватися принципів нормалізації бази даних. Нормалізація бази даних дозволяє зменшити

дублювання даних, забезпечити їх цілісність та ефективність роботи з даними [45]. Тому, при проектуванні та використанні баз даних слід дотримуватися принципів нормалізації та враховувати їх під час моніторингу та підтримки баз даних.

Отже, робота з базами даних є складним та важливим процесом при розробці програмних систем. Вимагається велика увага до аналізу, проектування, тестування, моніторингу та підтримки баз даних для забезпечення безпеки, ефективності та масштабованості програмних систем.

Додатково, важливим аспектом баз даних є індексування. Індекси дозволяють прискорити пошук та виконання запитів до бази даних. Вони створюються для певних полів або комбінацій полів у таблицях бази даних і дозволяють швидко знаходити рядки з певними значеннями. Індекси допомагають зменшити час виконання запитів і підвищити продуктивність бази даних.

Також, транзакції є важливою концепцією в базах даних. Транзакція - це одинична операція або група операцій, які виконуються як єдине ціле. Вони забезпечують атомарність, цілісність і незалежність виконання операцій [21]. Якщо транзакція виконується успішно, зміни стають постійними (commit). У разі виникнення помилки або незавершеності операцій, транзакція може бути відкатана (rollback), відновлюючи базу даних до попереднього стану [49]. Це дозволяє забезпечити цілісність даних та уникнути втрати даних у випадку помилок або непередбачуваних ситуацій. Також варто згадати про реплікацію баз даних. Реплікація полягає в створенні копій бази даних і їх розподіленні на різних серверах [48]. Це дозволяє підвищити доступність даних та забезпечити більшу швидкодію бази даних шляхом розподілення навантаження на різні сервери.

Крім того, реплікація забезпечує забезпечення надійності та відновлення даних в разі відмови сервера. У кінцевому підсумку, бази даних є ключовим елементом багатьох сучасних програмних систем та бізнес-додатків. Вони дозволяють ефективно зберігати, організовувати та керувати великими

обсягами даних, забезпечуючи надійність, безпеку та продуктивність. Правильне проектування, використання та підтримка баз даних є важливими завданнями для розробників програмного забезпечення та адміністраторів баз даних [22].

1.4 Постановка задачі кваліфікаційної роботи

Розробка програмної системи для ідентифікації користувачів на основі логіну та паролю з використанням двофакторної автентифікації є важливим завданням забезпечення безпеки інформації та даних користувачів. Вона дозволяє підвищити рівень захисту системи від зловмисників та забезпечити користувачам доступ до ресурсів після успішної ідентифікації.

Окрім розробки програмної системи для ідентифікації користувачів на основі логіну та паролю з використанням двофакторної автентифікації, слід звернути увагу на важливість постійного моніторингу та оновлення системи забезпечення безпеки. Підтримка актуальної версії програмного забезпечення та системи баз даних, використання ефективних антивірусів та захисту від шкідливих програм, регулярне змінювання паролів та інші заходи забезпечення безпеки є важливими умовами для забезпечення безпеки користувачів та їх даних.

Також варто розглянути можливість використання біометричної автентифікації, яка полягає в ідентифікації користувача за його фізичними характеристиками, наприклад, за допомогою відбитків пальців. Це дозволяє забезпечити ще більш високий рівень безпеки та зручності користування системою.

Отже, розробка програмної системи для ідентифікації користувачів на основі логіну та паролю з використанням двофакторної автентифікації – це важлива задача, яка дозволяє забезпечити високий рівень безпеки

користувачів та їх даних. Варто пам'ятати, що постійний моніторинг та оновлення системи забезпечення безпеки, а також розгляд можливості використання біометричної автентифікації можуть додатково підвищити рівень безпеки інформації та даних користувачів.

Таким чином, метою кваліфікаційної роботи є розробка програмної системи для ідентифікації користувачів на основі логіну та паролю із використанням двофакторної автентифікації. Щоб досягнути мети у кваліфікаційній роботі слід виконати певні задачі.

1. Проектування та розробка бази даних для зберігання даних про користувачів та їх авторизацію з використанням двофакторної автентифікації.

2. Розробка програмного забезпечення для авторизації користувачів на основі введення логіну та паролю з використанням додаткового фактора автентифікації, наприклад, коду, який буде знаходитися в програмі для генерації шестизначних кодів, наприклад Google Authenticator

3. Розробка алгоритму реєстрації та аутентифікації, для сканування відбитку пальця, та розпізнавання його.

4. Забезпечення безпеки системи та користувачів за допомогою шифрування паролів, контролю доступу, методів захисту від атак та інших заходів.

5. Розробка функцій для керування доступом користувачів до ресурсів системи та їхніх прав доступу з врахуванням використання двофакторної автентифікації.

6. Тестування системи на різних тестових наборах даних та в реальних умовах експлуатації.

7. Розробка документації до програмної системи, включаючи технічний опис системи, інструкції користувача та інші документи.

Основним результатом кваліфікаційної роботи є програмна система для ідентифікації користувачів на основі логіну та паролю із використанням двофакторної автентифікації, яка забезпечує високий рівень безпеки користувачів.

Додатковою складовою задачі може бути розробка механізму для керування додатковим фактором автентифікації, наприклад, можливістю встановлювати термін дії та інші налаштування [19].

Слід забезпечити можливість відновлення доступу в разі втрати додаткового фактора автентифікації. Для цього можна використовувати додаткові методи ідентифікації, наприклад, відправку коду на електронну пошту користувача або відповіді на питання безпеки.

Також, як додатковою задачею кваліфікаційної роботи може бути аналіз застосування двофакторної автентифікації в різних сферах, а також розробка рекомендацій щодо впровадження даного методу автентифікації в організаціях.

Кваліфікаційна робота призначена для розробки програмної системи для ідентифікації користувачів на основі логіну та паролю із використанням двофакторної автентифікації.

Система, розроблена в ході цієї роботи, включає ряд ключових компонентів і функціоналів:

1. Автентифікація за допомогою логіну та паролю: Це перший шлях ідентифікації користувача, який потребує введення унікального імені користувача (логіну) та секретного паролю.

2. Двофакторна автентифікація (2FA): Після успішної перевірки логіну та паролю, система просить користувача надати другий "фактор" для підтвердження своєї особистості. Цей другий фактор може бути, наприклад, одноразовим паролем, який надсилається на мобільний телефон користувача, або ж кодом з аутентифікатора.

3. Біометрична автентифікація: Підтримка біометричної автентифікації, такої як відбитки пальців, може бути додатковим фактором безпеки. Це означає, що користувач може бути попрошений підтвердити свою особистість, наклавши палець на біометричний сканер. Така технологія забезпечує ще вищий рівень безпеки, оскільки відбитки пальців важко підробити.

4. Безпека: Система призначена так, щоб надавати високий рівень безпеки. Вона використовує сучасні криптографічні алгоритми для шифрування паролів та інших особистих даних користувачів.

5. Масштабованість: Система спроектована так, щоб легко масштабуватися для підтримки великого числа користувачів.

2 АЛГОРИТМ ІДЕНТИФІКАЦІЇ ТА АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

2.1 Алгоритм реєстрації користувачів та шифрування даних на основі AES 256

У цифрову еру дуже важливо забезпечити безпеку та конфіденційність особистої інформації користувачів. Реєстрація користувача та шифрування даних є двома важливими аспектами цього процесу. Цей розділ присвячено алгоритму реєстрації користувача та шифруванню даних на основі 256-бітного ключа Advanced Encryption Standard (AES), який забезпечує надійний та ефективний спосіб захисту інформації користувачів у системі [44].

Алгоритм реєстрації користувача – це ряд кроків, які користувач повинен пройти, щоб створити обліковий запис у системі. Основною метою алгоритму реєстрації користувачів є автентифікація користувача та безпечне зберігання його інформації.

Наступні кроки описують загальний алгоритм реєстрації користувача.

1. Користувач надсилає реєстраційну форму, включаючи вибране ім’я користувача, пароль та будь-яку іншу необхідну особисту інформацію.
2. Перевірки на стороні сервера виконуються, щоб переконатися, що надіслані дані відповідають вимогам системи, таким як мінімальна довжина пароля та унікальне ім’я користувача.
3. Система генерує значення, випадкову послідовність символів, яке буде використовуватися під час процесу хешування пароля.
4. Пароль користувача хешується за допомогою безпечного алгоритму хешування, наприклад bcrypt або scrypt, разом із згенерованою сіллю. Цей процес перетворює простий текстовий пароль на незрозумілий рядок символів.
5. Система шифрує особисту інформацію користувача, наприклад адресу електронної пошти, за допомогою алгоритму шифрування AES256.

6. Сервер зберігає зашифровану особисту інформацію, хешований пароль і сіль у базі даних.

Розширений стандарт шифрування (AES) — це симетричний алгоритм шифрування, який широко поширений завдяки своїм надійним функціям безпеки та ефективній продуктивності. AES256 конкретно стосується AES із 256-бітною довжиною ключа, що забезпечує найвищий рівень безпеки [15].

Для AES256 потрібен 256-бітний секретний ключ, який можна згенерувати за допомогою криптографічно захищеного генератора випадкових чисел або отримати з парольної фрази, наданої користувачем, за допомогою функції виведення ключа, наприклад PBKDF2 або Argon2.

Структуру функції шифрування AES можна розглядати як мережу заміщення перестановки. Структура такої мережі проілюстрована на Рис 2.1. 128-бітні круглі клавіші виробляються за розкладом ключів з головного ключа. Круглі клавіші змішуються у функції круглої форми із станом блоку за допомогою XOR.

Раундова функція виконує фазу заміщення, передаючи шістнадцять байт стану блоку через 8-бітну функцію S-box. S-поле AES виконує мультиплікативне обернене, зокрема, кінцеве поле GF (28) перетворення у вигляді множення бітової матриці.

Як результат, функція S-box - це бієкція, необхідна для побудови мережі перестановки. Зворотний блок S для 24 дешфрування побудований з оберненої трансформації та тієї ж мультиплікативної інверсії в GF (28). Фаза заміщення дешифрування, яка виконує шістнадцять паралельних зворотних операцій S-box, називається InvSubBytes [4].

Процес шифрування AES починається з початкової трансформації, що є додатковою операцією AddRoundKey. Дешифрування можна виконати, змінивши порядок [5].

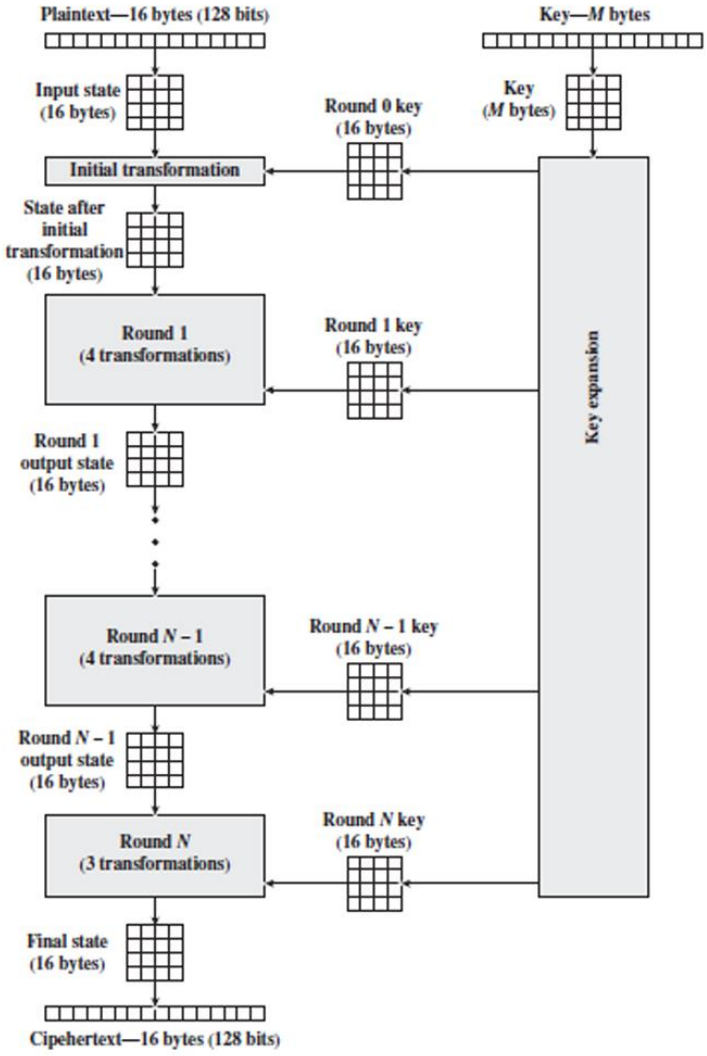

Рисунок 2.1 – Схема роботи AES алгоритму

Процес шифрування AES256 складається з певних кроків:

1. поділ даних відкритого тексту на 128-бітні блоки;
2. виконання початкового додавання ключа за допомогою 256-бітного секретного ключа.
3. 4 раунди процесу шифрування, який включає:
   a. суббайти: заміна коженого байту у блоці даних за допомогою S-box, попередньо визначеної таблиці підстановки;
   b. ShiftRows: змінює порядок байтів у кожному рядку блоку даних;
   c. AddRoundKey: XOR блок даних із круглим ключем,

отриманим із секретного ключа;

4. Виконаня останнього раунду процесу шифрування, який включає операції SubBytes, ShiftRows і AddRoundKey;

5. Об’єднання зашифрованих блоків даних, щоб сформувати остаточний зашифрований текст.

Алгоритм реєстрації користувачів і шифрування AES256 забезпечують надійний і безпечний метод захисту особистої інформації користувачів у цифровій системі. Впроваджуючи ці методи, системні адміністратори можуть забезпечити конфіденційність і цілісність даних користувача, запобігаючи несанкціонованому доступу та зберігаючи довіру користувачів.

2.2 Проектування бази даних

Створення бази даних є критичним елементом у розробці продуктивних та масштабуємих програм. Адекватно розроблена база даних сприяє цілісності даних, обмежує зайве дублювання і підвищує продуктивність обробки запитів. Ця секція фокусується на побудові баз даних використовуючи Amazon Web Services (AWS) і програмне забезпечення MySQL Workbench, презентуючи перегляд інструментарію та методик для створення і керування базами даних в межах AWS.

Amazon Web Services (AWS) надає багатий асортимент управлінських послуг баз даних для різноманітних сценаріїв використання та вимог. Дві ключові служби баз даних AWS, призначені для розробки реляційних баз даних, - це Amazon RDS (Relational Database Service) та Amazon Aurora.

Amazon RDS представляє собою управлінський сервіс реляційних баз даних, який підтримує декілька відомих двигунів баз даних, таких як MySQL, PostgreSQL, Oracle та Microsoft SQL Server. RDS автоматизує завдання, як-то створення резервних копій, відновлення і масштабування, що дозволяє

розробникам концентруватися на проектировані і оптимізації їхніх баз даних.

Amazon Aurora – це повністю управлінський сервіс реляційних баз даних, сумісний з MySQL і PostgreSQL. Його створено з урахуванням високої продуктивності та надійності. Aurora автоматично налаштовує обсяги зберігання і гарантує вбудовану стійкість до відмов, що робить його ідеальним для високопродуктивних застосунків, які вимагають мінімальної затримки для операцій читання та запису.

MySQL Workbench – це потужний візуальний інструмент з відкритим кодом для створення, управління та обслуговування баз даних MySQL. За його допомогою розробники мають змогу створювати та модифікувати схеми баз даних, управляти об'єктами бази даних і ефективно виконувати SQL-запити завдяки інтуїтивно зрозумілому графічному інтерфейсу MySQL Workbench. [6].

Додатковою вигодою використання баз даних AWS є їхня масштабованість. AWS надає зручність легкого масштабування реляційних баз даних відповідно до збільшуючихся потреб вашої організації [43]. Використовуючи AWS, ви можете збільшувати обсяги баз даних, підвищувати продуктивність та резервувати ресурси для подолання високого навантаження і забезпечення неперервної роботи вашої системи.

Також AWS надає високий ступінь безпеки для ваших баз даних. Ви маєте можливість використовувати різноманітні механізми захисту, такі як контроль доступу, шифрування даних та моніторинг дій. AWS також пропонує можливість автоматичного резервного копіювання та відновлення даних, що забезпечує захист від потенційної втрати даних.

В цілому, використання баз даних AWS, таких як Amazon RDS і Amazon Aurora, в комбінації з інструментами, такими як MySQL Workbench, дозволяє розробникам створювати надійні, масштабовані та безпечні системи управління базами даних. Це надає можливість фокусуватися на розробці програмного забезпечення та гарантує ефективне використання даних в організації.

Основні характеристики включають:

– Візуальне проектування схем: створення та модифікація таблиць бази даних, індексів та зв'язків через графічний інтерфейс;

– Редактор SQL: створення, виконання та оптимізація SQL-запитів з підсвічуванням синтаксису, авто-завершенням та форматуванням коду;

– Адміністрування сервера: управління екземплярами сервера MySQL, користувачами та параметрами безпеки;

– Моделювання даних: створення та підтримка діаграми сутності-зв'язку для візуального представлення схеми бази даних.

Процес проектування бази даних з використанням AWS і MySQL Workbench починається з налаштування екземпляра бази даних AWS.

Щоб створити базу даних за допомогою AWS і MySQL Workbench, спочатку необхідно розробити екземпляр бази даних Amazon RDS або Aurora MySQL:

1. входження у консоль керування AWS і перейдіть до інформаційної панелі RDS або Aurora;

2. натискання «Створити базу даних» і вибрати потрібну базу даних (наприклад, MySQL для Amazon RDS або Amazon Aurora);

3. налаштування параметрів екземпляра, наприклад, тип екземпляра, сховище та групу безпеки;

4. запуск екземпляра бази даних.

Виконання підключення MySQL Workbench до екземпляра бази даних AWS.

1. У MySQL Workbench потрібно нажати піктограму «+» у розділі «З’єднання MySQL», щоб створити нове з’єднання.

2. Потрібно ввести кінцеву точку примірника бази даних AWS, порт, ім’я користувача та пароль у відповідних полях.

3. Натиснути «Тестувати підключення», щоб перевірити підключення, а потім натиснути «ОК», щоб зберегти його.

Потрібно спроектувати схему бази даних. Після підключення до екземпляра бази даних AWS розроблено схему бази даних за допомогою функцій проектування візуальної схеми MySQL Workbench і моделювання даних:

– створення таблиці, визначаючи стовпці з відповідними типами даних, обмеженнями та значеннями за замовчуванням;

– призначення первинних ключів для кожної таблиці, щоб унікально ідентифікувати записи;

– створення зовнішніх ключів для встановлення зв'язків між таблицями та забезпечення посилальної цілісності;

– додавання індексів для оптимізації продуктивності запитів для поширених випадків використання;

– за потреби потрібно створити представлення даних, збережені процедури та тригери для інкапсуляції складної логіки або забезпечення дотримання правил цілісності даних.

Створення та виконання сценаріїв SQL. Після розробки схеми бази даних потрібно сористуватися MySQL Workbench для створення сценаріїв SQL, які визначають структуру та об'єкти бази даних: У MySQL Workbench потірбно відкрити меню «Database» і обрати «Forward Engineer». Виберати розроблену вами схему та натисніть «Далі». На вкладці «Параметри» налаштуйте потрібні параметри, наприклад «Генерувати оператори DROP» або «Опустити кваліфікатори схеми», і натисніть «Далі». Перегляньте згенерований сценарій SQL на вкладці «Переглянути сценарій SQL». За потреби внесіть необхідні зміни безпосередньо в сценарій. Натиснути «Далі», щоб виконати сценарій SQL на підключеному екземплярі бази даних AWS, створюючи об'єкти бази даних відповідно до вашого проекту.

Заповнення та керування даними. Після створення схеми бази даних скористайтеся редактором SQL MySQL Workbench та інструментами керування даними, щоб наповнити базу даних даними та ефективно керувати

ними.

Імпортуйте дані із зовнішніх джерел, наприклад файлів CSV або інших баз даних, за допомогою «Майстра імпорту даних таблиці», доступного в меню «Сервер». Використовуйте редактор SQL для написання та виконання операторів INSERT, UPDATE, DELETE та SELECT для маніпулювання даними в базі даних. Переглядайте та редагуйте дані таблиці за допомогою вкладки «Дані таблиці» в поданні схеми, яка забезпечує зручний інтерфейс, схожий на електронну таблицю, для керування даними [17]. Відстежуйте продуктивність вашої бази даних за допомогою інформаційної панелі продуктивності MySQL Workbench, яка надає інформацію в реальному часі про ключові показники ефективності, такі як пропускна здатність запитів, час відповіді та використання ресурсів.

Нормалізація схеми бази даних допомагає зменшити надмірність даних та підвищити цілісність даних. Для цього можна застосувати наступні нормальні форми.

– Перша нормальна форма (1НФ): Кожен стовпець в таблиці містить тільки атомарні значення, тобто значення, які не можна розбити на менші компоненти.

– Друга нормальна форма (2НФ): Кожен стовпець в таблиці залежить від всього первинного ключа, а не від частини його. Якщо є стовпець, який залежить від підмножини первинного ключа, його слід виділити в окрему таблицю.

– Третя нормальна форма (3НФ): Всі стовпці, які не є безпосередньо пов'язаними з первинним ключем, повинні залежати тільки від первинного ключа. Якщо є стовпці, які залежать від інших стовпців, їх слід виділити в окрему таблицю.

– Четверта нормальна форма (4НФ): У таблиці не повинно бути множинних залежностей, тобто стовпці не повинні залежати від інших незалежних стовпців.

– П'ята нормальна форма (5НФ): У таблиці не повинно бути залежностей між незалежними наборами стовпців, які мають ту саму ключову інформацію.

Створення індексів для стовпців, які часто використовуються в пошукових умовах або операціях об'єднання, може покращити продуктивність запитів. Однак, слід уникати надмірного індексування, щоб уникнути збільшення витрат на операції запису та споживання пам'яті.

Встановлення узгоджених правил іменування об'єктів бази даних, таких як таблиці, стовпці, індекси та зовнішні ключі, покращує читабельність та зручність обслуговування схеми бази даних.

Проектування бази даних з використанням баз даних AWS та MySQL Workbench дозволяє розробникам створювати ефективні, масштабовані та безпечні рішення для баз даних. Застосування керованих баз даних AWS та потужних інструментів візуального дизайну MySQL Workbench спрощує процес проектування, впровадження та керування базами даних. Використання найкращих практик проектування баз даних та функціональностей AWS та MySQL Workbench допомагає забезпечити цілісність, продуктивність та зручність обслуговування баз даних.

Ще одним важливим аспектом проектування бази даних є використання зовнішніх ключів для встановлення зв'язків між таблицями. Зовнішній ключ є посиланням на первинний ключ іншої таблиці і дозволяє створювати зв'язки між даними в різних таблицях [42]. Це допомагає забезпечити цілісність даних та зменшити дублювання інформації. При проектуванні бази даних важливо правильно визначити зовнішні ключі та їх зв'язки між таблицями для забезпечення правильної роботи бази даних та збереження цілісності даних.

Крім того, використання транзакцій є важливим аспектом управління базою даних. Транзакція – це логічна одиниця роботи з базою даних, яка включає одну або кілька операцій. Вона має властивості ACID (атомарність, консистентність, ізольованість, стійкість). Використання транзакцій дозволяє забезпечити цілісність та надійність бази даних, оскільки зміни в базі

виконуються атомарно - або повністю, або не виконуються зовсім. Нарешті, забезпечення резервного копіювання бази даних є важливим кроком для забезпечення безпеки та відновлення даних в разі втрати або пошкодження. Резервне копіювання дозволяє створювати копії бази даних, які можна використовувати для відновлення даних у разі потреби. Важливо регулярно створювати резервні копії бази даних і зберігати їх в безпечному місці з обмеженим доступом. Загалом, проектування бази даних вимагає уважного аналізу, нормалізації схеми, встановлення зв'язків, використання індексів, правильного іменування об'єктів та врахування різних аспектів безпеки та продуктивності. Знання та використання найкращих практик допоможуть розробникам створити ефективну та надійну базу даних для своїх програмних систем.

2.3 Алгоритм ідентифікації та автентифікації користувача на основі методів 2FA

Двофакторна автентифікація (2FA) – це важливий механізм безпеки, який додає додатковий рівень захисту облікових записів користувачів, вимагаючи від них підтверджувати свою особу за допомогою комбінації двох різних факторів. Цей метод ідентифікації та автентифікації стає все більш поширеним та рекомендованим для захисту від несанкціонованого доступу до облікових записів. Одним з популярних алгоритмів 2FA є використання Google Authenticator [41]. У цьому методі система просить користувача ввести одноразовий пароль (TOTP), який генерується додатком Google Authenticator на їх мобільному пристрої. Користувач вводить цей пароль, підтверджуючи свою особу. Система перевіряє введений TOTP на очікуване значення, згенероване за допомогою збереженого секретного ключа та поточного часу.

Цей метод забезпечує високий рівень безпеки, оскільки генерація одноразових паролів залежить від унікального секретного ключа, який відомий лише користувачеві та системі. На нижче наведеному рисунку 2.2 ви можете бачити алгоритм роботи методу TOTP.

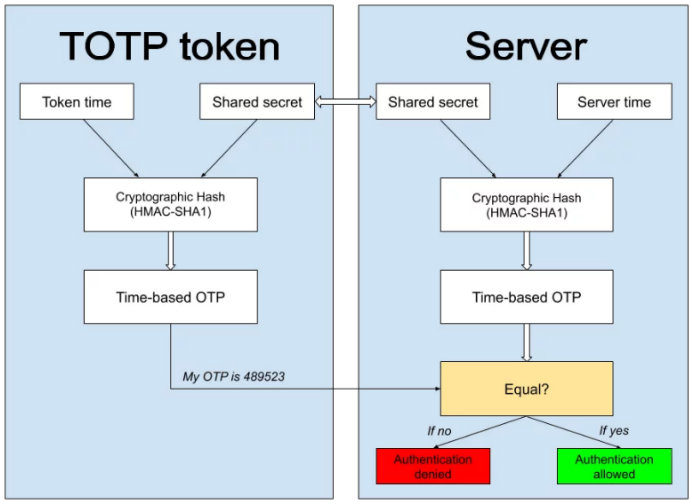

Рисунок 2.2 – Блок схема методу TOTP token

Ще одним методом 2FA є використання сканерів відбитків пальців. У такому випадку система пропонує користувачеві сканувати свій відбиток пальця за допомогою спеціального пристрою. Користувач прикладає палець до сканера, і система порівнює сканований відбиток пальця зі збереженими даними відбитків пальців для перевірки відповідності [7].

Третім методом 2FA є використання простих запитань для перевірки. Система пропонує користувачеві одне або кілька попередньо вибраних таємних питань, на які користувач надає відповіді. Система порівнює надані відповіді зі збереженими відповідями для підтвердження особи користувача.

Використання комбінації цих методів 2FA дозволяє створити більш надійний та безпечний спосіб ідентифікації та автентифікації користувачів [36]. Ще одним важливим аспектом 2FA є можливість налаштування альтернативних методів підтвердження особи. Наприклад, окрім Google Authenticator, можуть бути використані інші додатки або пристрої, які генерують одноразові паролі або здійснюють ідентифікацію на основі

біометричних даних, наприклад, сканування обличчя або розпізнавання відбитків пальців. При використанні 2FA важливо забезпечити правильне управління та зберігання факторів ідентифікації [32]. Користувачі повинні мати можливість безпечно зберігати та керувати своїми паролями, а також здійснювати належне управління своїми біометричними даними. Забезпечення конфіденційності та безпеки цих факторів є важливим аспектом успішної реалізації 2FA.

2FA є ефективним інструментом для захисту облікових записів користувачів від несанкціонованого доступу та зламу [33]. Використання двох різних факторів ідентифікації значно підвищує рівень безпеки та ускладнює заволодіння обліковими записами зловмисниками. Рекомендується впровадження 2FA для всіх облікових записів, особливо тих, які містять чутливу інформацію, щоб забезпечити надійний захист даних та забезпечити безпеку користувачів.

2.3.1 Аналіз методів 2FA

Google Authenticator – це програма, яка генерує одноразові паролі на основі часу (TOTP) для використання в двофакторній автентифікації [30]. Алгоритм TOTP генерує унікальний чутливий до часу код на основі спільного секретного ключа між користувачем і системою [31]. Користувач повинен ввести цей код разом із паролем, щоб отримати доступ до свого облікового запису.

Сканери відбитків пальців – це біометричні пристрої, які знімають і аналізують відбитки пальців користувача для ідентифікації їхньої особи. Цей метод базується на унікальності шаблону відбитка пальця людини, що забезпечує високий рівень безпеки [8].

Прості запитання для перевірки, також відомі як питання безпеки, є особистими запитаннями, на які користувач відповідає під час реєстрації

облікового запису. Пізніше ці запитання використовуються для перевірки особи користувача під час автентифікації або відновлення облікового запису. Користувач повинен надати правильні відповіді, щоб отримати доступ до свого облікового запису.

2.3.2 Алгоритм ідентифікації та автентифікації користувача

Наступний алгоритм описує комплексний процес ідентифікації та автентифікації користувача. Іноді для аутентифікації можуть використовуватися всі три фактори разом, тоді її називають багатофакторною чи мультифакторною (multi-factor authentication, MFA) [39]. Користувач вводить своє ім'я користувача та пароль. Система перевіряє введені облікові дані. Якщо облікові дані правильні, система переходить до процесу перевірки 2FA, також на рисунку 2.3 зображені методи аутентифікації користувача, вони представляють з себе методи MFA.

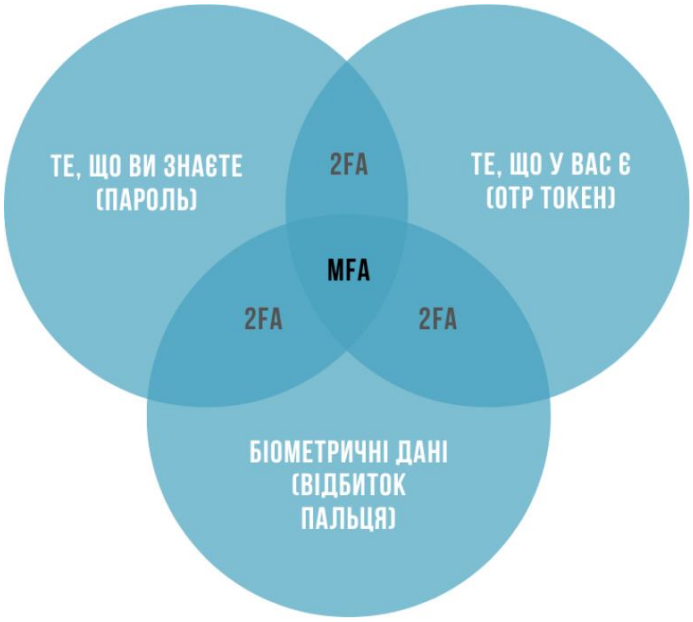

Рисунок 2.3 – Методи MFA (multi-factor authentication)

a) Google Authenticator:

1) Система просить користувача ввести TOTP, який відображається в додатку Google Authenticator;

2) Користувач вводить TOTP;

3) Система перевіряє TOTP на очікуване значення, згенероване за допомогою збереженого секретного ключа та поточного часу;

b) Сканер відбитків пальців:

1) Система пропонує користувачеві сканувати відбиток пальця за допомогою сканера відбитків пальців;

2) Користувач прикладає палець до сканера;

3) Система порівнює сканований відбиток пальця зі збереженими даними відбитків пальців та перевіряє їх відповідність;

c) Прості запитання для перевірки:

1) Система пропонує користувачеві одне або кілька попередньо обраних таємних питань;

2) Користувач надає відповіді на контрольні запитання;

3) Система порівнює надані відповіді зі збереженими відповідями.

Якщо перевірка 2FA пройшла успішно, користувач отримує доступ до свого облікового запису. У разі невдалої перевірки користувачеві буде відмовлено в доступі.

Впровадження системи багатофакторної автентифікації. Для підвищення безпеки організації можуть вибрати систему багатофакторної автентифікації, яка вимагає від користувачів пройти кілька методів 2FA перед наданням доступу. У такій системі процес автентифікації відбувається наступним чином [40]:

Користувач вводить своє ім'я користувача та пароль, які перевіряються системою. Система випадковим чином вибирає два або більше методів 2FA для використання в процесі автентифікації (наприклад, Google Authenticator і сканер відбитків пальців). Користувач повинен успішно виконати кожен вибраний метод 2FA. Якщо всі вибрані методи 2FA успішно виконано,

користувач отримує доступ до свого облікового запису. У разі невдалої перевірки будь-якого методу 2FA користувачеві буде відмовлено в доступі.

Включення кількох методів 2FA, таких як алгоритм Google Authenticator, сканери відбитків пальців і прості запитання для перевірки, до процесу ідентифікації та автентифікації користувача значно підвищує безпеку облікового запису [35]. Застосування надійного алгоритму автентифікації, який використовує комбінацію методів 2FA, дозволяє організаціям захищати облікові записи користувачів від несанкціонованого доступу, зберігати довіру користувачів і забезпечувати загальну безпеку своїх систем [16].

2.3.3 Переваги та недоліки методів 2FA

Деякі з ключових переваг використання методів 2FA у процесі ідентифікації та автентифікації користувачів включають:

– деякі з ключових переваг використання методів 2FA у процесі ідентифікації та автентифікації користувачів включають:

– покращена безпека: методи 2FA додають додатковий рівень безпеки, ускладнюючи зловмисникам отримання несанкціонованого доступу до облікових записів користувачів;

– більша впевненість користувачів: користувачі більш впевнені в безпеці своїх облікових записів, коли потрібні кілька факторів автентифікації;

– відповідність галузевим стандартам: багато галузей і регуляторних органів вимагають використання 2FA для захисту облікових записів користувачів і конфіденційних даних.

Незважаючи на переваги, є деякі потенційні недоліки, пов’язані з використанням методів 2FA:

– підвищена складність: впровадження методів 2FA може ускладнити процес автентифікації як для користувачів, так і для системних

адміністраторів;

– проблеми щодо зручності використання: деякі користувачі можуть вважати методи 2FA громіздкими або незручними, особливо якщо їм треба використовувати кілька пристроїв і відповідати на таємні запитання;

– потенціал для хибних спрацьовувань/негативів: хоча методи 2FA загалом покращують безпеку, все ще існує ризик хибних спрацьовувань (наприклад, дійсному користувачеві заборонено доступ) або хибно-негативних (наприклад, зловмисник успішно обійшов 2FA) [13].

Щоб підвищити ефективність методів 2FA, потрібно звернути увагу на такі рекомендації:

– оцінити вимоги організації до безпеки та потреби користувачів, щоб визначити найбільш прийнятні методи 2FA.;

– забезпечення навчання користувачів використанню методів 2FA, щоб мінімізувати проблеми з зручністю використання та просувати найкращі методи безпеки;

– регулярний перегляд та оновлення питання безпеки, щоб переконатися, що вони залишаються актуальними та ефективними;

– відстеження продуктивності і ефективності методів 2FA, щоб визначити потенційні області для вдосконалення або оптимізації;

– розгляд можливості впровадження адаптивної автентифікації, яка регулює необхідні фактори автентифікації на основі поведінки користувача та рівня ризику.

Таким чином, інтеграція кількох методів 2FA, таких як алгоритм Google Authenticator, сканери відбитків пальців і прості запитання для перевірки, у процес ідентифікації та автентифікації користувача значно підвищує безпеку облікового запису. Ретельно вибираючи та впроваджуючи відповідні методи 2FA, організації можуть збалансувати безпеку та зручність використання, захищаючи облікові записи користувачів від несанкціонованого доступу.

3 РЕАЛІЗАЦІЯ АЛГОРИТМУ ІДЕНТИФІКАЦІЇ ТА АВТЕНТИФІКАЦІЇ КОРИСТУВАЧА

3.1 Аналіз програмного середовища та інструментарію

Проведене дослідження базується на розробці застосунку в середовищі Python, використовуючи крос-платформену бібліотеку PyQt для створення графічного інтерфейсу користувача. Під час розробки використовувалась база данихSQLite для збереження інформації про користувачів і їх відбитки пальців, а також бібліотека pyotp для реалізації двофакторної автентифікації за допомогою Google Authenticator.

Python – це високорівнева мова програмування, яка була обрана для цього проекту через її простоту, гнучкість та потужні бібліотеки, які дозволяють швидко розробляти та випробовувати програмне забезпечення. Python також відомий своєю читабельністю та зручністю для навчання, що робить його відмінним вибором для розробки складних проектів [34].

PyQt – це потужний набір біндінгів мови програмування Python для набору інструментів для створення графічного інтерфейсу користувача, відомого як Qt. Ця бібліотека створена для того, щоб дозволити розробникам Python використовувати всю потужність та гнучкість Qt у своїх програмах. Однією з основних переваг PyQt є те, що вона дає можливість розробникам створювати додатки з багатим, інтерактивним і професійно виглядаючим графічним інтерфейсом користувача. Завдяки PyQt, програми, написані на Python, можуть мати такий самий високий рівень вигляду і відчуття, як і програми, написані за допомогою традиційних бібліотек графічного інтерфейсу користувача. PyQt надає велику кількість класів, які можна використовувати для створення різноманітних елементів інтерфейсу, включаючи вікна, діалогові вікна, кнопки, текстові поля, списки, меню і багато іншого.

Також PyQt має потужну систему сигналів і слотів, яка дозволяє легко організувати взаємодію між різними компонентами програми. Крім того, PyQt включає ряд інструментів для автоматизації процесу розробки, включаючи дизайнер форм Qt (Qt Designer), який дозволяє розробникам створювати інтерфейси за допомогою перетягування елементів, і мову розмітки інтерфейсу Qt (Qt Markup Language, QML), що дозволяє описувати інтерфейси в декларативному стилі [37].

Таким чином, за допомогою PyQt розробники можуть легко створювати складні, високоякісні графічні інтерфейси користувача на Python, використовуючи багSQLite – це вбудована система керування базами даних, яка є ідеальним рішенням для цього проекту через її простоту в використанні та гнучкість. SQLite не вимагає окремого сервера для роботи і може зберігати всі дані в одному файлі, що спрощує розповсюдження програми.

PyOTP – це бібліотека Python для генерації та перевірки одноразових паролів. Вона використовується для реалізації двофакторної автентифікації за допомогою Google Authenticator. PyOTP може генерувати секретні ключі для кожного користувача, а потім використовувати ці ключі для генерації одноразових паролів, що забезпечують високий рівень безпеки для користувачів [14].

PySide6 – це офіційний набір для Python від Qt Company, який включає в себе бібліотеку для розробки графічного інтерфейсу користувача Qt6. PySide6 та PyQt5/6 є дуже схожими, оскільки обидві надають доступ до Qt API з Python, але є важливі різниці, які можуть вплинути на вибір між ними [9]. Інтерфейс PySide6 ми можемо бачити на рисунку 3.1. Однією з важливих особливостей PySide6 є те, що вона поставляється з LGPL (Lesser General Public License), що означає, що ви можете використовувати її в комерційних проектах без необхідності відкривати вихідний код вашого додатка. PySide6 надає доступ до всіх основних класів і функцій Qt6, що дозволяє створювати високоякісні, професійно виглядаючі графічні інтерфейси користувача.

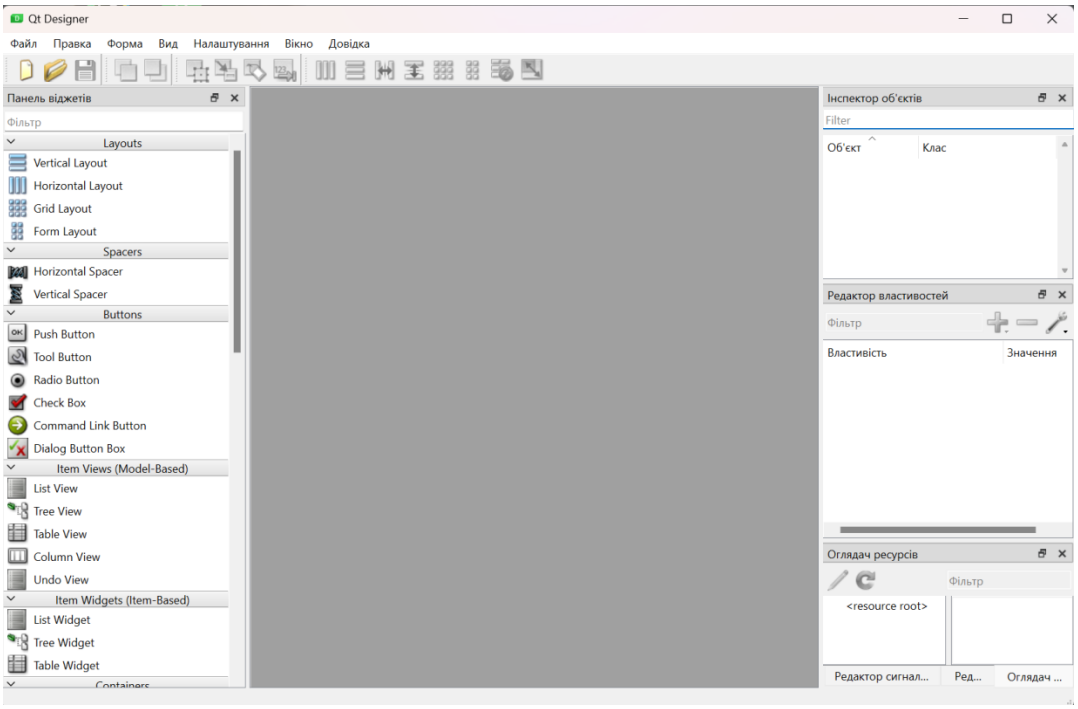

Рисунок 3.1 – Інтерфейс Qt Designer який включає в себе PySide 6

Ця бібліотека включає в себе інструменти для роботи з вікнами, діалогами, кнопками, списками, меню, і багато іншого. Подібно до PyQt, PySide6 має потужну систему сигналів і слотів, що дозволяє легко організувати взаємодію між різними компонентами програми. Ця бібліотека також включає в себе набір інструментів для автоматизації процесу розробки, включаючи Qt Designer і QML. Важливо відмітити, що PySide6 регулярно оновлюється і підтримується Qt Company, тому вона забезпечує доступ до найновіших функцій і вдосконалень Qt6.

Для роботи програмного забезпечення та розпізнавання відбитків пальців використовується сканер відбитків пальців, який є якісним пристроєм для аутентифікації та ідентифікації користувача за допомогою сканування відбитків пальців. Ці сенсори включають DSP чіп, який обробляє зображення, виробляє необхідні розрахунки для виявлення відповідності між записаними і поточними даними. Датчик відбитків пальців дозволяє записати до 150 різних відбитків пальців. При використанні бази даних можливо зберігати більше.

Технічні характеристики сканера AS608 [10]:

– напруга живлення: 3.3 В (постійний струм);
– Робочий струм споживання: 40 мА (Режим очікування);
– Максимальний струм споживання: 60 мА (режим сканування);
– Час обробки зображення відбитка: < 1.0 секунд;
– Розмір вікна: 15.3 мм x 18.2 мм;
– Кількість файлів, що записуються: 150 файлів (відбитків);
– Рівень безпеки (від 1 до 5);
– Інтерфейс (підключення): TTL послідовний;
– Швидкість передачі (Baud rate): 9600, 19200, 28800, 38400, 57600 (за замовчуванням 57600);
– Робочий діапазон температур: від -20 °C до +50 °C;
– Допустимий рівень вологості: 40 % - 85 % RH;
– Габаритні розміри: 56 x 20 x 21.5 мм;
– Маса модуля: 20 грам.

Вигляд сканера зображений на рисунку 3.2.

Рисунок 3.2 – Сканера відбитків AS608

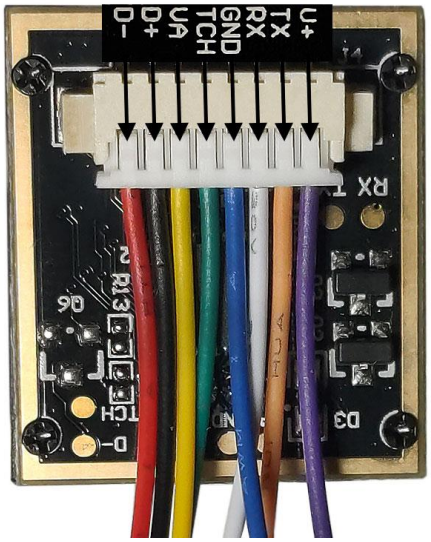

Рисунок 3.3 – Нижня частина сканера відбитків пальця

Піни сканера відбитків пальця:

1. U+ – Подача напруги (позитивний полюс);
2. TX – Передача даних (вихідний сигнал);
3. RX - Прийом даних (вхідний сигнал);
4. GND - Земля (загальний контакт);
5. TCH - Вхід для дотику (сенсорний сигнал);
6. VA - Напруга для аналогового живлення;
7. D+ - USB D+ сигнал (для USB-підключення);
8. D- - USB D- сигнал (для USB-підключення).

Також для підключення сканера відбитків пальців використовувався модуль CP2102. Вигляд модуля можливо побачити на рисунку 3.4.

CP2102 – це USB-серійний перетворювач, розроблений компанією Silicon Labs. Він дозволяє підключати пристрої з інтерфейсом RS232 до комп'ютера або іншого пристрою через USB-порт. Цей модуль широко використовується в різних галузях, включаючи електроніку, робототехніку та програмування мікроконтролерів [11].

CP2102 має компактний розмір і зручний для використання. Він підтримує швидкість передачі даних до 921,600 біт/с, що дозволяє передавати

великі обсяги інформації з високою швидкістю. Модуль також підтримує різні режими передачі даних, включаючи асинхронний режим, режим RTS/CTS (Request to Send/Clear to Send) і режим XON/XOFF (Software Flow Control). CP2102 працює з різними операційними системами, включаючи Windows, macOS та Linux. Для його використання потрібно встановити драйвери, які забезпечують правильну роботу модуля з комп'ютером. Драйвери зазвичай надаються виробником модуля і доступні для завантаження з їх веб-сайту [14]. CP2102 також має різні вихідні піни, які можна підключити до інших пристроїв або мікроконтролерів для обміну даними. Це дозволяє використовувати модуль в різних проектах, де потрібна комунікація зі зовнішніми пристроями через USB.

У загальному, модуль CP2102 є потужним і зручним інструментом для підключення пристроїв з інтерфейсом RS232 до комп'ютера чи іншого пристрою через USB-порт. Він широко використовується в електроніці та робототехніці і дозволяє забезпечити швидку і надійну передачу даних [12].

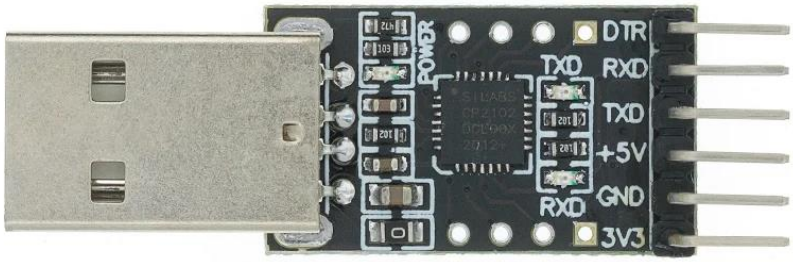

Рисунок 3.4 – USB-серійний перетворювачCP2102

Піни сканера серійного перетворювач CP2102:

1. DTR – Вихід для сигналу DTR (Data Terminal Ready);
2. RXD - Прийом даних (вхідний сигнал);
3. TXD - Передача даних (вихідний сигнал);
4. 5V - Подача напруги 5 В (позитивний полюс);

5. GND - Земля (загальний контакт);

6. 3V3 - Подача напруги 3.3 В (позитивний полюс).

Для підключення сканера відбитків пальців використовувалася бібліотека pyfingerprint. Бібліотека pyfingerprint – це бібліотека програмного забезпечення для роботи зі сканерами відбитків пальців, яка надає зручний інтерфейс для інтеграції та управління цими пристроями з використанням мови програмування Python. Основні функції та можливості бібліотеки pyfingerprint включають:

– запуск та підключення до сканера відбитків пальців;

– зчитування відбитків пальців з пристрою та збереження їх у вигляді зображень або шаблонів;

– пошук збережених відбитків пальців у базі даних;

– верифікація або ідентифікація відбитків пальців на основі порівняння збережених шаблонів;

– встановлення та управління параметрами сканера відбитків пальців, такими як якість зображення, розмір області сканування тощо;

– робота з командами сканера, такими як створення, видалення або оновлення шаблонів.

Pyfingerprint робить процес роботи зі сканерами відбитків пальців простим та зручним, надаючи розширені можливості для розробки програм, що використовують ці пристрої. Вона дозволяє розробникам ефективно використовувати функціональність сканера відбитків пальців у своїх проектах, пов'язаних з аутентифікацією осіб, безпекою або контролем доступу. Окрім цього, Pyfingerprint підтримує багатий набір команд, що позволяють створювати, зберігати та пошук відбитків пальців. Це робить цю бібліотеку особливо корисною для розробників, які прагнуть інтегрувати біометричну ідентифікацію в свої додатки або системи.

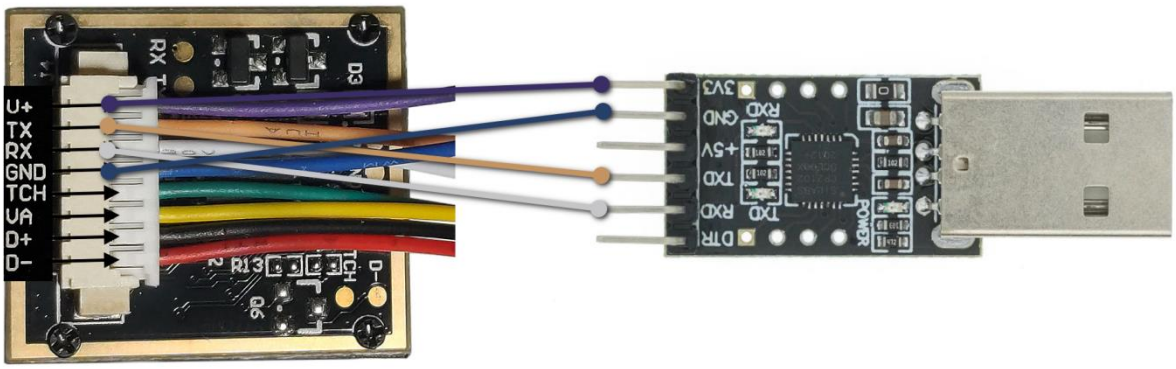

Рисунок 3.5 – Спосіб під’єднання сканера відбитків до модуля CP2102

Підключення сканера до перетворювача здійснюється за допомогою кабелів які ідуть в комплекті, або їх можна придбати.

3.2 Архітектура програмного забезпечення

ПЗ базується на моделі "Модель видавництва-підписник" (Publish-Subscribe model) з використанням шаблону проектування "Спостерігач" (Observer pattern). Основне вікно програми використовує об'єкт Ui_BasicWindow, який служить для налаштування графічного інтерфейсу користувача. У цьому вікні використовується віджет QStackedWidget для керування різними сторінками програми, залежно від дій користувача. Крім того, програма використовує декілька віджетів, таких як QDialog і QMainWindow, для відображення додаткових вікон та діалогових вікон. У програмі також присутні кілька власних класів, таких як UserDatabaseManager, EncryptionManager, FingerprintRegistrationDialogWIDGET, MessageWindowWIDGET та інші, які реалізують певну логіку програми.

За допомогою сигналів і слотів, програма встановлює зв'язки між різними класами і реагує на події, такі як натискання кнопок або зміна значень полів введення.

Програма також використовує сторонні бібліотеки, такі як PyFingerprint для роботи з датчиком відбитків пальців та pyotp для реалізації двофакторної аутентифікації Google.

В цілому, архітектура програми заснована на розділенні відповідальностей між різними класами та використанні шаблонів проектування для забезпечення модульності, розширюваності та підтримки принципів SOLID.

Продовжуючи опис архітектури програми, можна зазначити, що вона використовує парадигму об'єктно-орієнтованого програмування. Кожен клас в програмі виконує певну функціональність і має свої властивості та методи для взаємодії з іншими об'єктами.

Деякі з ключових компонентів програми:

1. Клас UserDatabaseManager: Відповідає за керування базою даних користувачів. Містить методи для збереження, отримання та перевірки даних користувачів;

2. Клас EncryptionManager: Відповідає за шифрування та розшифрування паролів користувачів. Забезпечує безпеку збереження паролів у базі даних;

3. Клас ExpenseTracker: Головний клас програми, який успадковує від QMainWindow. Він виконує роль контролера, координуючи взаємодію між графічним інтерфейсом користувача та різними компонентами програми. Містить обробники подій, методи для перевірки логінів користувачів, налаштування відображення вікон та інші функції;

4. Класи, пов'язані з графічним інтерфейсом: Ui_BasicWindow, Ui_FingerprintRegistrationDialogWIDGET, Ui_MessageWindowWIDGET і т.д. Ці класи відповідають за налаштування графічного інтерфейсу користувача за допомогою фреймворку Qt.

Рисунок 3.6 – Головне меню (ExpenseTracker - BasicWindow)

Архітектура програми базується на подієво-орієнтованому підході, де класи спостерігачів (наприклад, FingerprintRegistrationDialogWIDGET і MessageWindowWIDGET) реагують на події, які виникають у головному класі ExpenseTracker або інших компонентах програми.

Взаємодія між класами здійснюється за допомогою передачі сигналів і підключення слотів. Наприклад, подія натискання кнопки викликає відповідний слот, який обробляє цю подію та виконує необхідні дії.

Загалом, архітектура програми спроектована з урахуванням принципів модульності, розширюваності і зручності управління. Це дозволяє програмі ефективно виконувати завдання обліку витрат та забезпечує зручний інтерфейс для користувачів.

Клас MessageWindowWIDGET: Відповідає за відображення повідомлень користувачу. Містить стековий віджет для переключення між різними сторінками повідомлень. Це можна побачити на рисунку 3.7.

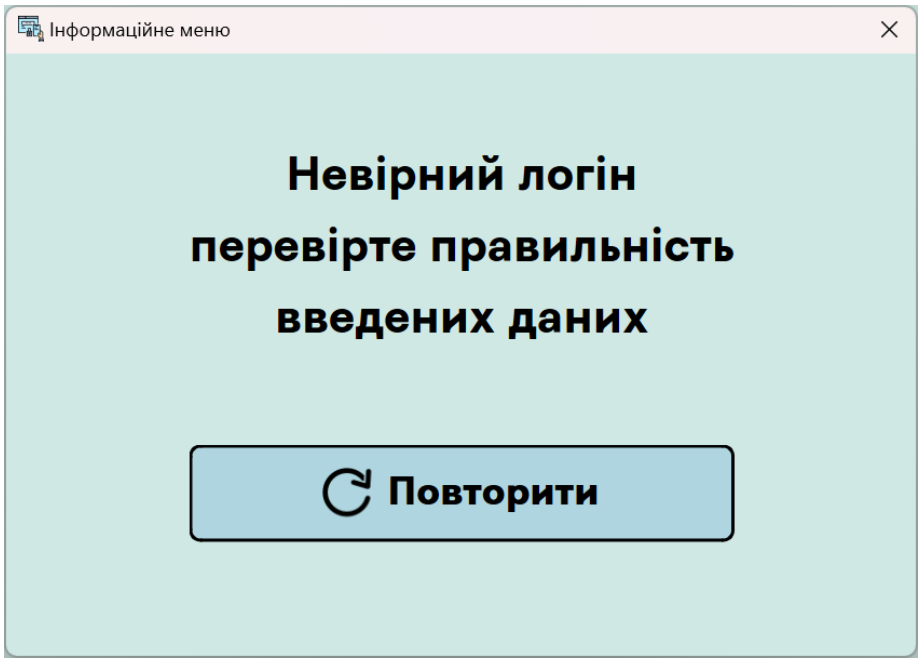

Рисунок 3.7 – Інформаційне вікно (MessageWindowWIDGET)

Клас FingerprintRegistrationDialogWIDGET: Представляє діалогове вікно для реєстрації відбитків пальців. Використовує бібліотеку PyFingerprint для взаємодії з датчиком відбитків пальців. Зображено на риснуку 3.8.

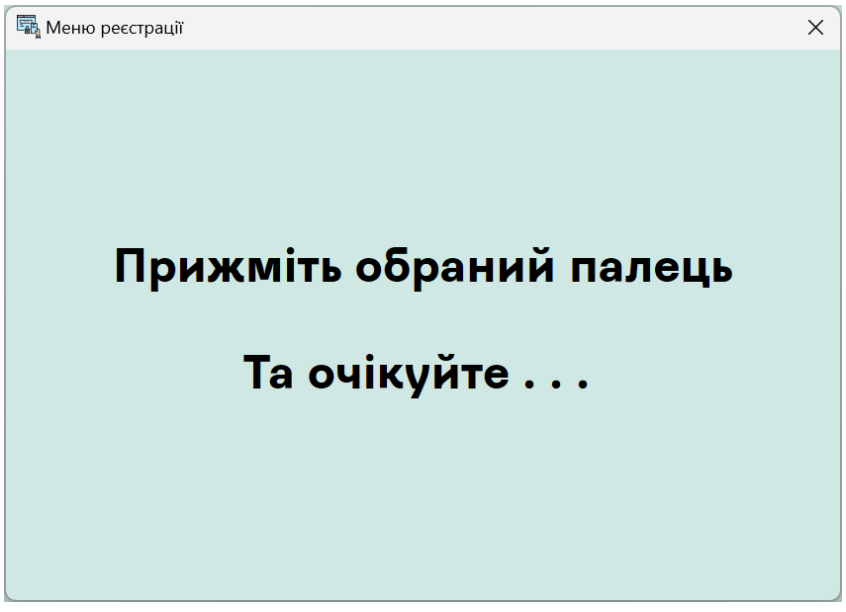

Рисунок 3.8 – Вікно інформації про реєстрації (FingerprintRegistrationDialogWIDGET)

Продовжуючи опис архітектури програми, варто зазначити про використання модель-представлення-контролер (MVC) або модель-вид-контролер (MVP) патерну. Цей підхід дозволяє відокремити логіку програми (модель) від графічного інтерфейсу (представлення) і від обробки дій користувача (контролер).

У програмі ExpenseTracker модель відповідає за роботу з базою даних користувачів, шифрування паролів і збереження відбитків пальців. Вона представлена класом UserDatabaseManager і EncryptionManager. Ці класи забезпечують збереження та доступ до даних користувачів.

Представлення програми включає графічний інтерфейс, який реалізований за допомогою класів Ui_BasicWindow, Ui_FingerprintRegistrationDialogWIDGET, Ui_MessageWindowWIDGET та інших. Ці класи відповідають за налаштування та відображення елементів інтерфейсу, таких як кнопки, поля введення та повідомлення.

Контролер представлений класом ExpenseTracker. Він обробляє події, які виникають у графічному інтерфейсі, такі як натискання кнопок або введення даних користувачем. Контролер забезпечує взаємодію між моделлю і представленням, виконує необхідні перевірки, обробляє запити користувача та забезпечує відображення відповідних сторінок повідомлень.

Така архітектура дозволяє розділити логіку програми на окремі компоненти, що спрощує розробку, тестування та підтримку програмного забезпечення. Крім того, це дозволяє змінювати або розширювати окремі компоненти без впливу на інші частини програми.

Наступні класи GoogleAuthenticatorWIDGET та GoogleAuthenticatorCheckWIDGET також внесуть внесок у структуру програми.

GoogleAuthenticatorWIDGET відповідає за генерацію та відображення коду для Google Authenticator, зображено на рисунку 3.9. Він має залежність від зовнішньої бібліотеки pyotp, яка дозволяє генерувати та перевіряти Google 2FA-коди. Клас виконує такі функції:

– ініціалізує інтерфейс користувача та генерує випадковий ключ Google 2FA;

– генерує QR-код, який відображається у вікні;

– перевіряє введений користувачем код з Google Authenticator;

– виділяє сигнал fingerprint_saved4, який вказує на успішне завершення процесу аутентифікації з допомогою Google 2FA.

Рисунок 3.9 – Вікно реєстрації Google аутенитифікатора (GoogleAuthenticatorWIDGET)

GoogleAuthenticatorCheckWIDGET використовується для перевірки коду, введеного користувачем з Google Authenticator. Цей клас має наступний функціонал:

– ініціалізує інтерфейс користувача для введення коду;

– Перевіряє введений користувачем код та виділяє сигнал google_auth_passed, передаючи введений код як аргумент сигналу.

Ці класи додають можливість використовувати двофакторну аутентифікацію за допомогою Google Authenticator який зображений на рисунку 3.10 у програмі ExpenseTracker. Класи інтегруються зі структурою інших класів і взаємодіють з ними за допомогою сигналів і слотів, що дозволяє розширити функціональність програми та покращити безпеку доступу.

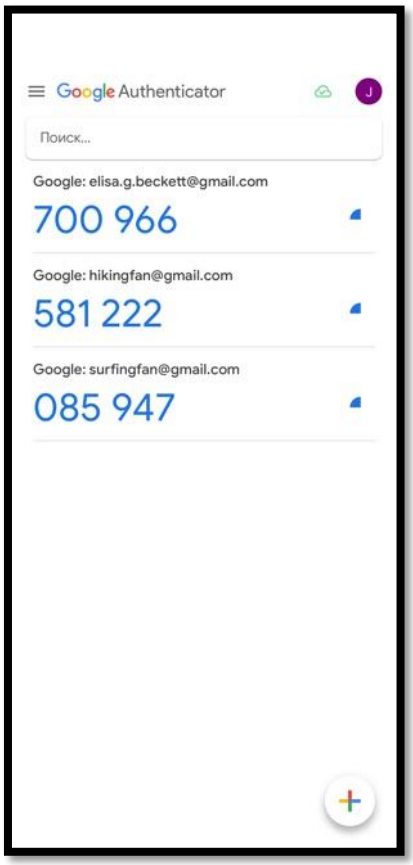

Рисунок 3.10 – Програмне забезпечення для створення тимчасових кодів, за допомогою яких можливо виконати аутентифікації користувача за допомогою Google Authenticator

Також є вікно налаштувань користувача, яке можна бачити на рисунку 3.10 за допомогою якого користувач самостійно зможе змінити дані, такі як пароль, або ім’я користувача, також відключити аутентифікації користувача при потребі. Або це Google Authenticator або сканування відбитку пальця.

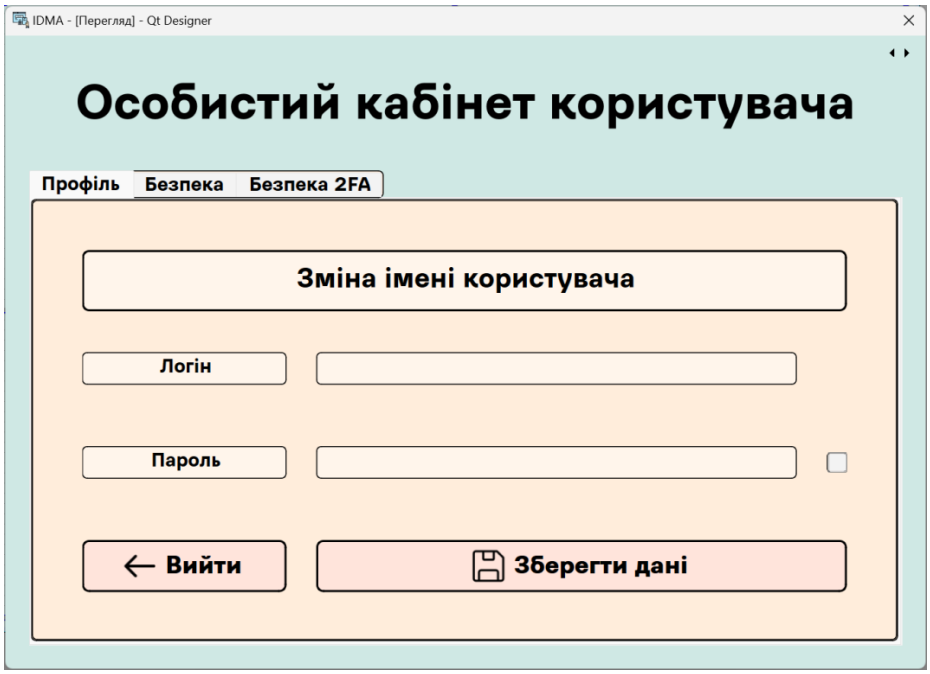

Рисунок 3.11 – Особистий кабінет користувача (BasicWindow)

Вкладка «Безпека» яка зображення на рисунку 3.12 у вікні BasicWindow, дає можливість змінити пароль користувача безпосередньо без участі адміністратора, за допомогою вводу старого паролю, наступним кроком введення нового паролю, з підтвердженням його, методом повтору. Метод прийняття даних у вигляді коду зображений на рисунку 3.11.

```
def on_save_password_data_button_clicked(self):
    old_password = self.ui.lineEdit_5.text()
    new_password = self.ui.lineEdit_7.text()
    confirm_new_password = self.ui.lineEdit_6.text()
```

Рисунок 3.12 – Приклад коду для збору інформації для зміни паролю

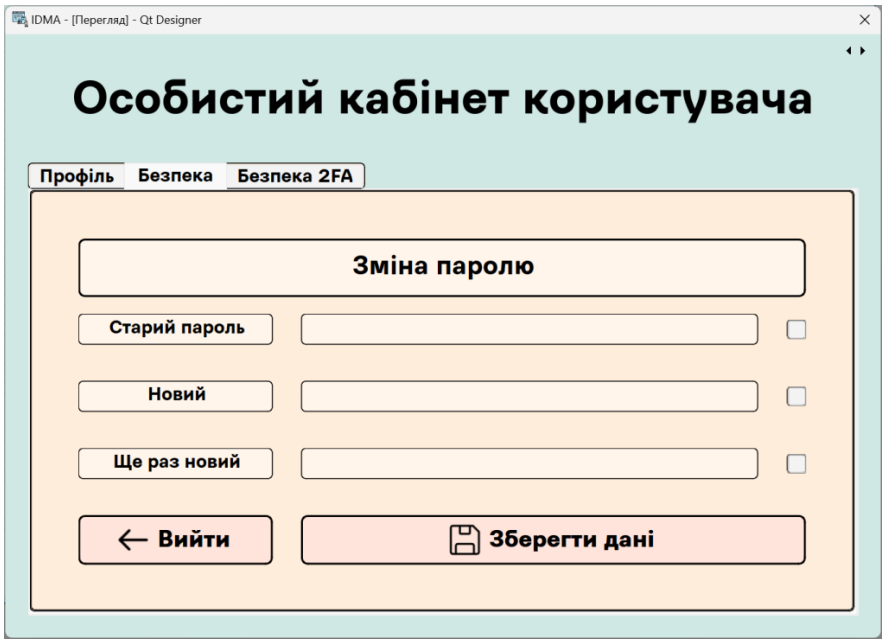

Рисунок 3.13 – Особистий кабінет користувача (BasicWindow)

Також існує вікно адміністратора яке зображено на рисунку 3.13, може показувати дані які занесені в базу даних користувача. Адміністратор може змінювати дані користувачів напряму, без доступу до стороннього ПО.

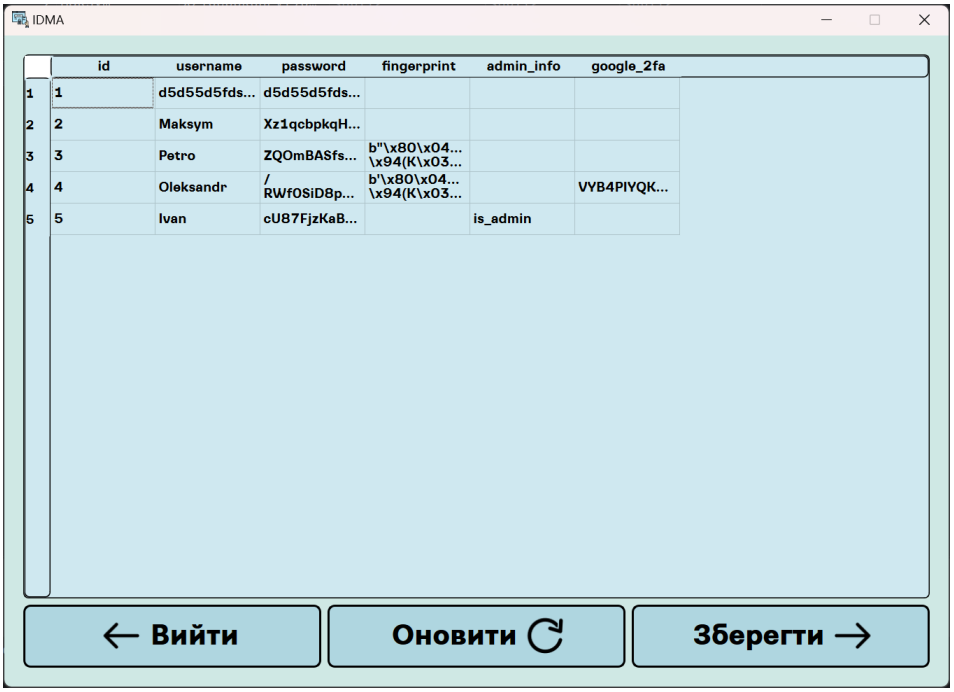

Рисунок 3.14 – Вікно адміністратора

3.3 Реалізація алгоритму ідентифікації та автентифікації користувача

Алгоритм ідентифікації та автентифікації користувача, який використовується в наданому коді, базується на наступних кроках:

Вхідні дані: Користувач вводить своє ім'я користувача (логін) та пароль через графічний інтерфейс.

Перевірка наявності користувача: За допомогою методу get_user_from_database з об'єкта user_db_manager перевіряється наявність користувача в базі даних. Якщо користувач з таким ім'ям існує, отримуються його дані.

Розшифрування паролю: З отриманих даних користувача витягується зашифрований пароль. За допомогою encryption_manager виконується розшифрування цього паролю.

Порівняння паролів: Розшифрований пароль, введений користувачем, порівнюється з розшифрованим паролем з бази даних. Якщо вони співпадають, користувач успішно пройшов автентифікацію.

Двофакторна автентифікація (Google Authenticator): Якщо користувач має активовану двофакторну автентифікацію за допомогою Google Authenticator, виконується наступний крок.

Google Authenticator є двофакторною системою автентифікації, яка використовує протокол Time-Based One-Time Password (TOTP) для генерації і відображення одноразових паролів. Ось приблизна блок-схема, яка описує його роботу:

– реєстрація: Користувач встановлює Google Authenticator на свій пристрій і з'єднує його з обліковим записом на веб-сайті, використовуючи QR-код або секретний ключ, який надає веб-сайт;

– генерація OTP: Google Authenticator використовує поточний час і зареєстрований секретний ключ для генерації одноразового паролю (OTP). OTP змінюється кожні 30 секунд;

– введення OTP: Коли користувач намагається увійти на веб-сайт, він вводить свій звичайний пароль і потім OTP, який відображається в Google Authenticator;

– перевірка OTP: Веб-сайт також генерує OTP, використовуючи зареєстрований секретний ключ і поточний час. Якщо OTP, введений користувачем, співпадає з OTP, який генерує веб-сайт, вхід дозволяється.

– Користувач - [Встановлює Google Authenticator]-> Google Authenticator;

– Користувач - [Сканує QR-код або вводить секретний ключ]-> Google Authenticator;

– Google Authenticator - [Генерує OTP на основі часу і секретного ключа]-> Користувач - [Вводить OTP на вікні програми]-> Вікно програми;

– Вікно програми - [Генерує свій OTP на основі часу і секретного ключа]-> Вікно програми - [Порівнює два OTP]-> Якщо вони співпадають, вхід дозволено;

Алгоритм розпізнавання відбитків пальців у даному коді включає такі кроки:

– збір відбитка пальця: Користувач реєструє свій відбиток пальця за допомогою біометричного сенсора або пристрою. У даному випадку, цей процес відбувається у спеціальному вікні реєстрації відбитків пальців;

– збереження відбитка пальця: Отримані характеристики відбитка пальця зберігаються у базі даних SQLite. Ця інформація використовується для майбутнього порівняння з введеним відбитком пальця при автентифікації;

– автентифікація відбитка пальця: При автентифікації користувача, відбиток пальця, який був зчитаний біометричним сенсором або пристроєм, порівнюється з збереженими в базі даних характеристиками відбитків пальців. Використовуючи алгоритм порівняння, визначається ступінь відповідності між введеним відбитком пальця та збереженими даними.

– результат автентифікації: Залежно від ступеня відповідності, визначеного алгоритмом розпізнавання відбитків пальців, визначається, чи

була автентифікація успішною. Якщо відбиток пальця відповідає збереженим даним, користувач отримує доступ до системи. В іншому випадку, автентифікація вважається невдалим і користувачу може бути відмовлено у доступі.

Алгоритм розпізнавання відбитків пальців який зображений на рисунку 3.14, може використовувати різні методи та алгоритми, такі як аналіз текстур, порівняння особливостей або нейронні мережі. У конкретному коді можуть бути використані конкретні алгоритми, залежно від використовуваних бібліотек та пристроїв для розпізнавання відбитків пальців. Алгоритм реєстрації у коді:

```
class FingerprintRegistrationDialogWIDGET(QDialog):
    fingerprint_saved2 = Signal()
    def __init__(self, parent=None):
        super().__init__(parent)
        self.ui = Ui_FingerprintRegistrationDialogWIDGET()
        self.ui.setupUi(self)
        self.scan_fingerprint()
self.ui.pushButtonRegistrationNEXT_4.clicked.connect(self.on_next_button_clicked)
    def on_next_button_clicked(self):
        self.fingerprint_saved2.emit()
        self.close()
    def scan_fingerprint(self):
        QTimer.singleShot(1000, self.start_scan)
    def start_scan(self):
        try:
            fingerprint = PyFingerprint('COM5', 57600, 0xFFFFFFFF, 0x00000000)
            if fingerprint.verifyPassword() == False:
                raise ValueError('The given fingerprint sensor password is wrong!')
        except Exception as e:
            print('The fingerprint sensor could not be initialized!')
            print('Exception message: ' + str(e))
            return None
        print('Waiting for finger...')
        while fingerprint.readImage() == False:
            pass
        fingerprint.convertImage(0x01)
        self.fingerprint_characteristics = fingerprint.downloadCharacteristics()
```

Рисунок 3.15 – Приклад коду розпізнавання відбитків

Після виклику FingerprintRegistrationDialogWIDGET, починається сканування відбитку пальців, через затримку в 1 секунду, для того щоб сканер

запустив сканування. Далі відбиток зберігається в базу даних у стовпець «fingerprint».

Коли потрібно перевірити користувача викликається метод check_login_data_users1, на рисунку 3.15 можемо бачити код який реалізований для цього методу. Після чого він перевіряє логін пароль, відбиток пальця якщо є, та гугл ідентифікатор також якщо є. Якщо існують і перше і друге, то перевіряють тільки відбиток. Якщо тільки аутентифікатор, то перевіряється аутентифікатор, якщо нічого немає, то вхід виконується тільки за допомогою логіну та паролю.

```
def check_login_data_users1(self):  # Користувач1 Основний логін
    username = self.ui.lineEdit_4Login_3.text()
    password = self.ui.lineEdit_3Password_3.text()
    user_data = self.user_db_manager.get_user_from_database(username)
    if user_data is None:
        self.message_window = MessageWindowWIDGET(self)
        self.message_window.ui.stackedWidget.setCurrentIndex(6)  # Перейти на сторінку "FailedLogin"
        self.message_window.show()
        return
    decrypted_password = self.encryption_manager.decrypt(user_data['password'])
    if decrypted_password != password:
        self.message_window = MessageWindowWIDGET(self)
        self.message_window.ui.stackedWidget.setCurrentIndex(2)  # Перейти на сторінку "FailedPassword"
        self.message_window.show()
        return
    if user_data['fingerprint'] is not None:
        self.open_fingerprint_login_dialog2()
    # Якщо користувач має Google 2FA
    elif user_data['google_2fa'] is not None:
    # Якщо користувач має Google 2FA
        self.open_fingerprint_login_dialog2()
    elif user_data['google_2fa'] is not None:
        # Відкрити вікно для введення коду Google 2FA
        self.google_auth_check_widget = GoogleAuthenticatorCheckWIDGET(self, username)
        self.google_auth_check_widget.show()
        # Приєднати функцію on_button_check_google_2fa_clicked до сигналу google_auth_passed
        self.google_auth_check_widget.google_auth_passed.connect(
            self.on_button_check_google_2fa_clicked_users1)
    else:
        self.ui.stackedWidget.setCurrentIndex(6)  # Перейти на сторінку "page_3_AfterLogin"
        self.ui.lineEdit_4Login_3.clear()
        self.ui.lineEdit_3Password_3.clear()
        self.ui.lineEdit_4Login_3.clear()
        self.ui.lineEdit_3Password_3.clear()
```

Рисунок 3.16 - приклад коду для перевірки входу користувача

3.4 Тестування програмного забезпечення на основі експериментів

Перед початком проведення тестування наведено дані користувачів які зареєстровані в системі. Також дані зображені у рисунку 3.17.

Таблиця 3.1 – Дані зареєстрованих користувачів в системі

Дані зареєстрованого користувача №1
Логін: Maksym
Пароль: Maksym123456789
Відбиток: Ні
Google аутентифікатор: Ні
Дані зареєстрованого користувача №2
Логін: Petro
Пароль: Petro8665266
Відбиток: Так
Google аутентифікатор: Ні
Дані зареєстрованого користувача №3
Логін: Oleksandr
Пароль: Oleksandr866gfg
Відбиток: Так
Google аутентифікатор: Так (Додано через особистий кабінет)
Адміністратор: Ні
Дані зареєстрованого користувача №4
Логін: Ivan
Пароль: Ivand68f14s6d8fs
Відбиток: Ні
Google аутентифікатор: Ні
Адміністратор: Так

	id	username	password	fingerprint	admin_info	google_2fa
1	1	d5d55d5fds51…	d5d55d5fds51f65…	<null>	<null>	<null>
2	2	Maksym	Xz1qcbpkqHEVl9G…	<null>	<null>	<null>
3	3	Petro	ZQOmBASfsRL2iVw…	0x80049505060000…	<null>	<null>
4	4	Oleksandr	/RWf0SiD8pZtg6R…	0x80049505060000…	<null>	VYB4PIYQK46IHX…
5	5	Ivan	cU87FjzKaB2lQWO…	<null>	is_admin	<null>

Рисунок 3.17 – Дані зареєстрованих користувачів в систему (PyCharm)

Перше тестування – чи пропустить метод реєстрації, дані які введені не на англійській мові. Результати тестування можна бачити на рисунку 3.18.

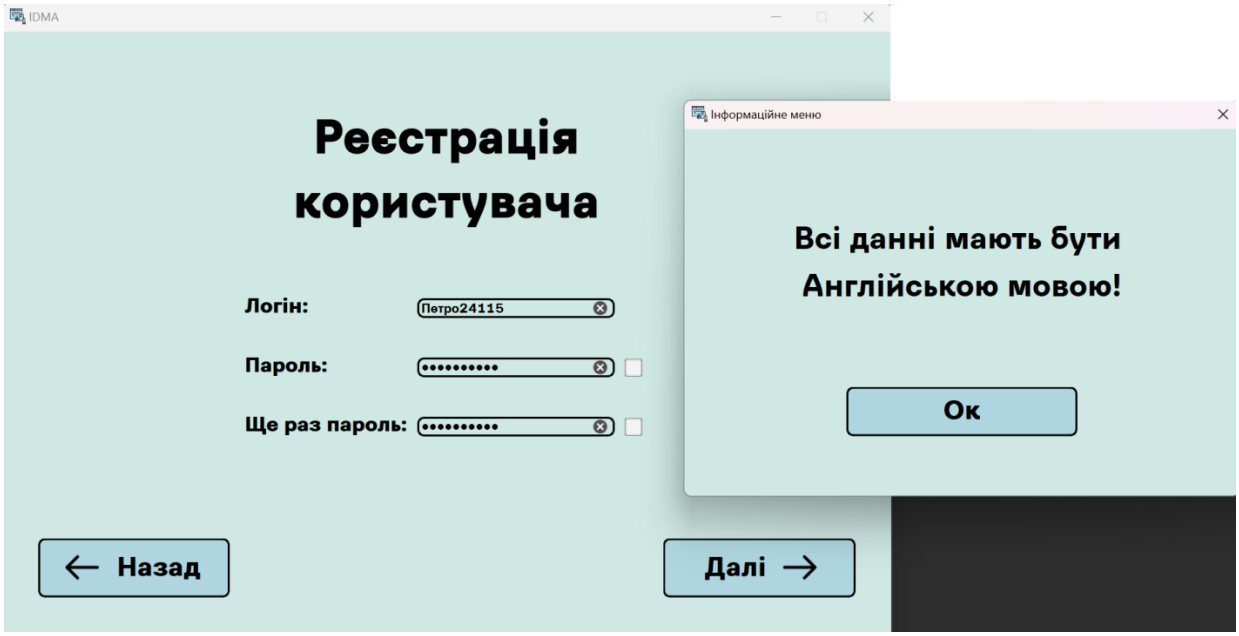

Рисунок 3.18 – Перевірка даних на мову реєстрації

Друге тестування – чи пропустить система пароль без великої літери або як мінімум однієї цифри. Результати зображені на рисунку 3.19.

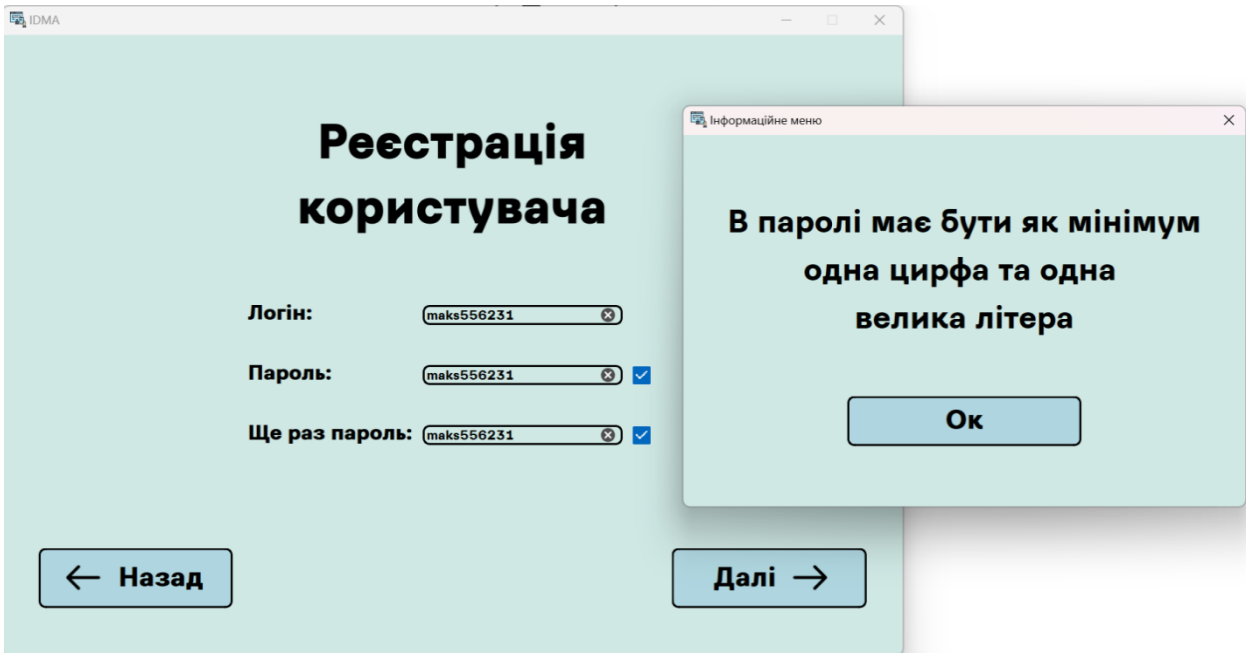

Рисунок 3.19 – Перевірка даних на потрібні дані для реєстрації такі як: мінімум одна цифра, та одна велика літера

Третє тестування буде на те, чи продовжить програма ідентифікацію якщо користувача немає в базі даних (наявні дані показано вище). Результати зображенні на рисунку 3.20.

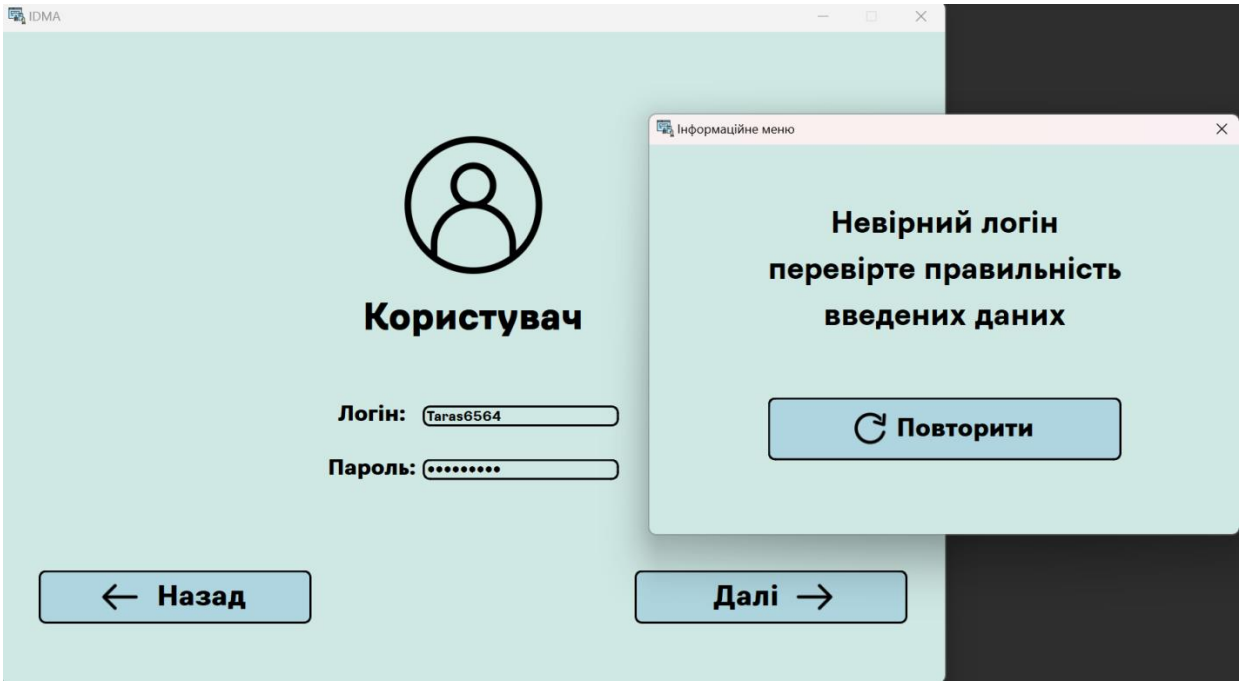

Рисунок 3.20 – Перевірка наявності користувача

Четверте тестування на те, чи пропустить користувача якщо логін існує, але пароль невірний. Результати зображені на рисунку 3.21.

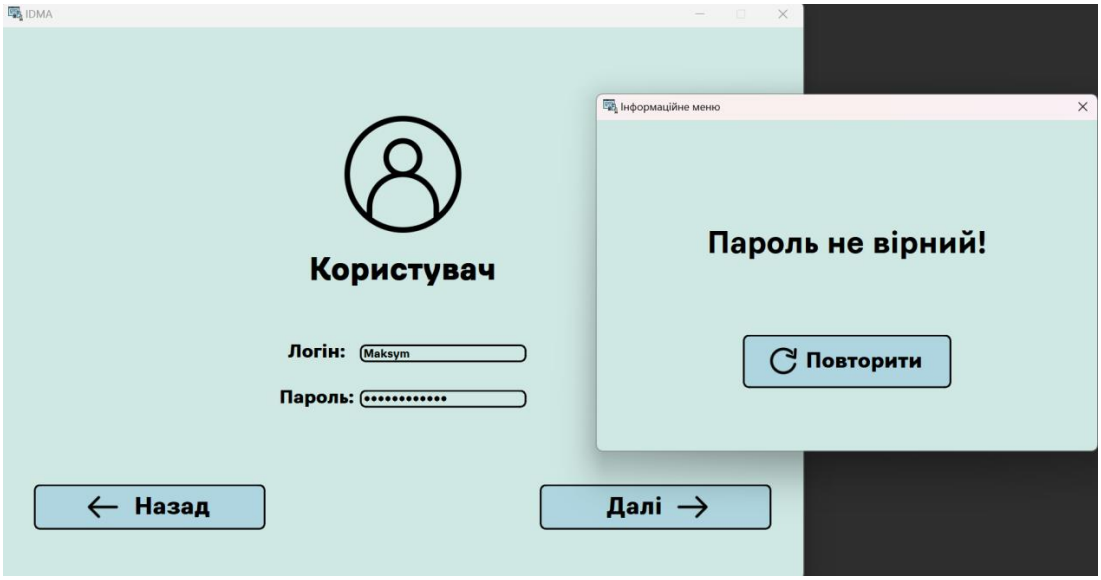

Рисунок 3.21 – Перевірка паролю користувача

П’яте тестування – чи зможе увійти користувач, якщо в нього є права адміністратора (відмітка в базі даних у стовпці admin_info --- значення is_admin). Результати на рисунку 3.22.

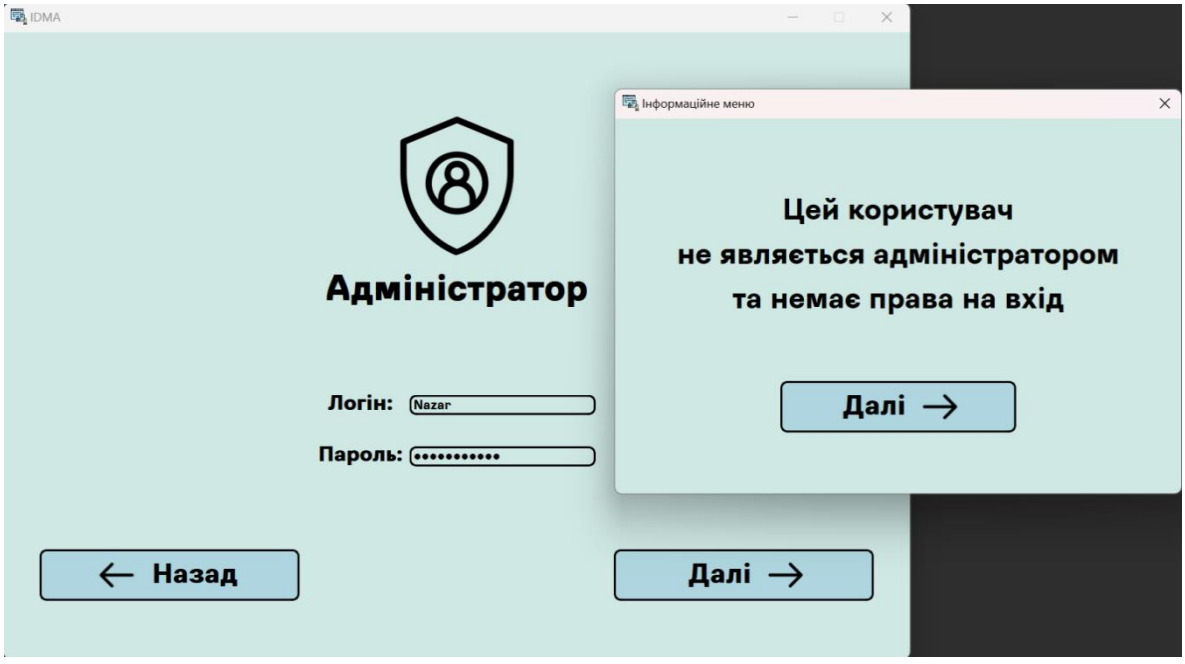

Рисунок 3.22 – Перевірка адміністратора

Шоста перевірка на відбиток пальця, чи працює цей метод, і чи розпізнає він відбиток. Відбиток відхилено, тобто перевірено і він не вірний, результат зображений на рисунку 3.23.

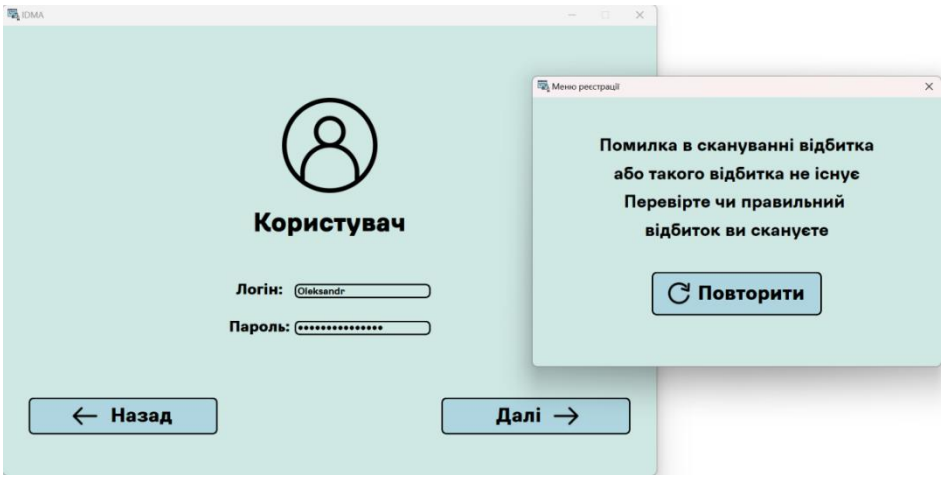

Рисунок 3.23 – Перевірка відбитку пальця користувача

Сьома перевірка буде перевіряти чи дійсний код чи не дійсний в методі Google аутентифікатора. Результат на рисунку 3.24.

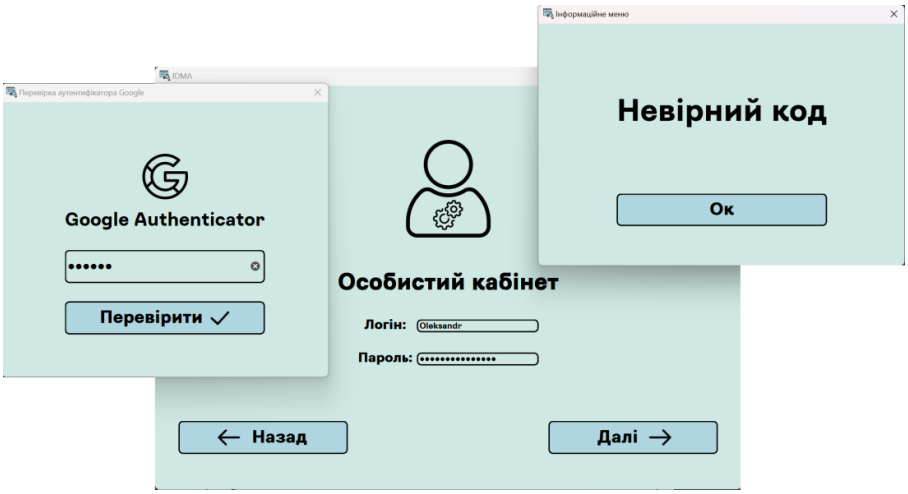

Рисунок 3.24 – Перевірка Google аутентифікатора

Виконуючи ці перевірки та тестування, було виявлено що всі основні алгоритми програмного забезпечення виконують свою ціль, помилок немає, все працює без перебоїв.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Визначення витрат на розробку програмної системи

Обчислення трудомісткості процесу розробки програми.

Трудомісткість розробки програми розраховують за формулою:

$$t = t_o + t_u + t_a + t_n + t_н + t_д, \tag{4.1}$$

де:

t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_н$ - витрати праці на налагодження програми на ПК;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмі, яка розробляється. Умовне число операторів (підпрограм) розраховують за формулою:

$$Q = q \cdot C \cdot (1 + p), \tag{4.2}$$

де:

q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

Коефіцієнт складності програми (C) характеризує відносну складність програми з відношення до так званої типовий завданню, реалізує стандартні на методи вирішення, складність якої прийнята що дорівнює одиниці (величина лежать у межах від 1,25 до 2).

Коефіцієнт кореляції програми в ході її розробки (p) - збільшення обсягу робіт з допомогою внесення змін - у алгоритм чи програму з результатам уточнення постановок і описів її, зміни складу і структури інформації, і навіть уточнень, внесених розробниками підвищення якості самої програми без зміни постановки завдання (величина у межах 0,05...0,1).

Розрахунок складових витрат:

$$Q = 1206 \cdot 1,7 \cdot (1 + 0,5) = 3075,3.$$

Трудові витрати на підготовку й опис поставленої задачі (t_0) точній оцінці не піддаються, оскільки це пов'язано з творчим характером роботи (приймається 50).

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста, розраховують за формулою:

$$t_u = \frac{Q \cdot B}{S \cdot k}, \qquad (4.3)$$

де:

В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, уточнення та деякого доопрацювання - якість постановки завдання, виданого у розроблення, у зв'язку з тим, що завдання, зазвичай, вимагають уточнень і відзначаються певним доопрацюванням (цей коефіцієнт залежно від складності завдання приймається від 1,2 до 1,5).

S- коефіцієнт, який визначається складністю завдання, у даному випадку S = (75...85).

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та становить:

- для працюючих до 2 років - 0,8;
- від 2 до 3 років - 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

Розрахунок витрат праці на вивчення опису задачі:

$$t_u = \frac{3075,3 \cdot 1,5}{81 \cdot 1,15} = 49,52 год .$$

Витрати праці на розробку алгоритму рішення задачі розраховують за формулою:

$$t_a = \frac{Q}{S \cdot k}, \qquad (4.4)$$

де:

S- коефіцієнт, який визначається складністю завдання, у даному випадку S = (20…25).

Розрахунок:

$$t_a = \frac{3075,3}{25 \cdot 1,15} = 107 год .$$

Витрати на складання програми по готовій блок-схемі розраховують за формулою:

$$t_n = \frac{Q}{S \cdot k}, \tag{4.5}$$

де:

S- коефіцієнт, який визначається складністю завдання, у даному випадку S = (20…25).

Розрахунок:

$$t_n = \frac{3075,3}{20 \cdot 1,15} = 133,7 год.$$

Витрати праці на налагодження програми на ПК розраховують:

За умови автономного налагодження одного завдання:

$$t_н = \frac{Q}{S \cdot k}, \tag{4.6}$$

де:

S- коефіцієнт, який визначається складністю завдання, у даному випадку S = (4…5).

Розрахунок:

За умови автономного налагодження одного завдання:

$$t_н = \frac{3075,3}{5 \cdot 1,15} = 535 год.$$

За умови комплексного налагодження завдання:

$$t_н^k = 1,5 \cdot t_н . \quad (4.7)$$

Розрахунок:

За умови комплексного налагодження завдання:

$$t_н^k = 1,5 \cdot 535 = 802,5 год ,$$

$$t_н = 802,5^{1,15} = 2188,2 год .$$

Витрати праці на підготовку документації розраховують за формулою:

$$t_д = t_{др} + t_{до} , \quad (4.8)$$

де:

$t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{S \cdot k} , \quad (4.9)$$

де:

S- коефіцієнт, який визначається складністю завдання, у даному випадку S = (15…20).

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др} . \quad (4.10)$$

Розрахунок:

$$t_{др} = \frac{3075,3}{15 \cdot 1,15} = 178,3 год,$$

$$t_{до} = 0,75 \cdot 178,3 = 133,7 год,$$

підготовка документації:

$$t_{д} = 178,3 + 133,7 = 312 год,$$

трудомісткість розробки програми буде розраховуватись за формулою:

$$t = t_{о} + t_{и} + t_{а} + t_{п} + t_{н} + t_{д}. \quad (4.11)$$

Розрахунок:

$$t = 50 + 49,52 + 107 + 133,7 + 535 + 312 = 1187 год,$$

$$t = 1187 год.$$

Розрахунок витрат на оплату праці розробників

Розрахуємо середньо годинну оплату програміста. Для цього необхідно спочатку визначити його річний фонд грошового забезпечення. Це можна зробити, знаючи місячне грошове забезпечення програміста. Воно складає приблизно 42000,00 гривень. Таким чином, річний фонд грошового забезпечення 504000 гривень.

Кількість робочих годин у році розраховуємо за формулою:

$$N_р = (N - N_п - N_в) \cdot 8, \quad (4.12)$$

де N - загальна кількість днів у році,

$N_п$ - кількість святкових днів у році,

$N_в$ - кількість вихідних днів у році.

Приймається, що кількість святкових днів у році – 14, а вихідних – 104.

Середньо годинна оплата праці програміста визначається за формулою:

$$C_п = \frac{Ф_р}{N_р}, \quad (4.13)$$

де:

$Ф_р$ річний фонд грошового забезпечення

Витрати на оплату праці розробників програми складають:

$$B_{оп} = C_п \cdot T_з, \quad (4.14)$$

де:

$T_з$ - загальна кількість годин роботи програміста над проектом.

Розрахунок:

$$N_р = (365 - 11 - 105) \cdot 8 = 1992 год.$$

$$C_п = \frac{504000}{1992} = 253 грн,$$

$$B_{оп} = 253 \cdot 1187 = 300311 грн .$$

Витрати, пов'язані з розробкою програми на ПК розраховуються:

$$B_{ПК} = T_{ПК} \cdot t_c , \quad (4.15)$$

де:

$T_{ПК}$ - час використання ПК для розробки програми,

$C_{ПК}$ - собівартість машинного часу обчислювальної техніки (розраховує бухгалтерія підприємства).

Собівартість однієї години роботи ПК дорівнює:

$$C_{ПК} = \frac{B_e}{Ф_{ПК}} , \quad (4.16)$$

де:

B_e - річні поточні витрати на експлуатацію ПК,

$Ф_{ПК}$ - річний фонд часу корисної роботи ПК.

Розрахуємо річний фонд часу роботи ПК. Визначивши дійсний річний фонд часу ПК у годинах, отримаємо можливість оцінити собівартість годин машинного часу.

Дійсний річний фонд часу ПК дорівнює:

$$Ф_д = N_р - (Ф_м + Ф_{річ}) , \quad (4.17)$$

де:

$\Phi_м$ - місячний фонд часу на профілактику і ремонт ПК (час профілактики щомісячно – 5 годин),

$\Phi_{річ}$ - річний фонд часу на профілактику і ремонт ПК (час профілактики щорічно – 6 діб) .

Розрахунок:

$$\Phi_д = 1992 - (5 + 144) = 1843 год .$$

Розрахунок електроенергії:

$$B_e = \sum_{i=1}^{n} P_i \cdot k_i \cdot T_i \cdot Ц , \qquad (4.18)$$

де:

P_i - паспортна потужність і-го електрообладнання, кВт;

k_i - коефіцієнт використання потужності і-го електрообладнання (приймається 0.7…0.9);

T_i - час роботи і-го устаткування за весь період розробки, год;

$Ц$ - ціна електроенергії, грн / кВт*год;

i - тип електрообладнання;

n - кількість електрообладнання.

Річні поточні витрати на експлуатацію ПК:

$$B_e = \sum_{i=1}^{1} 0,3 \cdot 0,9 \cdot 1992 \cdot 1,44 = 774,48 грн .$$

Собівартість машинного часу обчислювальної техніки (розраховує бухгалтерія підприємства):

$$C_{ПК} = \frac{774.48}{1843} = 0,42 грн / год.$$

Витрати, пов'язані з розробкою програми на ПК:

$$B_{ПК} = 1187 \cdot 0,42 = 498,54 грн.$$

Таблиця 4.1 – кошторис витрат на розробку ПЗ

Статті витрат	Сума, грн
1. Витрати на електроенергію	774,8 грн
3. Витрати на оплату праці	42000 грн
4. Відрахування у соціальні фонди	8610 грн
5. Амортизація основних фондів	840 грн
Разом	52225,2 грн

У таблиці 4.1 було розглянуто розрахунок витрат на розробку продукту, було розрахованно час розробки та погодинну оплату.

4.2 Розрахунок ціни проєкту

Річні поточні витрати на експлуатацію програмного забезпечення визначаються за формулою:

$$B_{ПК_р} = B_{E_р} + B_{A_р} + B_{РЕМ_р} + B_{ДК_р} + B_{I_р}, \quad (4.19)$$

де:

$B_{A_р}$ – річні відрахування на амортизацію,

$B_{E_р}$ – річні витрати на електроенергію для ПК,

$B_{РЕМ_р}$ – річні витрати на ремонт ПК,

$B_{ДК_р}$ – річні витрати на додаткові комплектуючі ПК,

$B_{I_р}$ – інші витрати.

Суму річних амортизаційних відрахувань визначаємо за такою формулою:

$$B_{A_р} = Ц_{ПК} \cdot Н_A, \tag{4.20}$$

де:

$Ц_{ПК}$ – балансова вартість ПК,

$Н_A$ – норма амортизаційних відрахувань (дорівнює 15% у квартал).

Балансову вартість ПК розраховуємо за формулою:

$$Ц_{ПК} = Ц_р \cdot (1 + К_{УН}), \tag{4.21}$$

де:

$Ц_р$ – ринкова вартість ПК,

$К_{УН}$ – коефіцієнт, що враховує витрати на установку й налагодження ПК (приймається рівним 12%).

Витрати на електроенергію, що споживає ПК, визначаємо за формулою:

$$B_{E_р} = Р_{ПК} \cdot Ф_{ПК} \cdot Ц_E \cdot К_{ІВ}, \tag{4.22}$$

де:

$P_{ПК}$ – паспортна потужність ПК,

$Ф_{ПК}$ – річний фонд корисного часу роботи ПК,

$Ц_{Е}$ – вартість 1 кВт/год електроенергії,

$P_{ІВ}$ – коефіцієнт інтенсивного використання ПК (0,7 - 1).

Таким чином, розрахункове значення витрат на електроенергію, що споживає ПК, складає:

-витрати на поточний і профілактичний ремонт (приймаються рівними 6% від вартості ПК):

$$B_{РЕМ_р} = Ц_{ПК} \cdot 0,06, \quad (4.23)$$

- витрати на додаткові комплектуючі

- витрати необхідні для забезпечення експлуатації ПК (приймаються рівними 2% від вартості ПК):

$$B_{ДК_р} = Ц_{ПК} \cdot 0,02, \quad (4.24)$$

- інші витрати, тобто непрямі витрати пов'язані з експлуатацією ПК (приймаються рівними 5-10% від вартості ПК):

$$B_{І_р} = Ц_{ПК} \cdot 0,05. \quad (4.25)$$

Розрахунок:

Балансова вартість ПК:

$$Ц_{ПК} = 5000 \cdot (1 + 0{,}12) = 5600 грн .$$

Річні відрахування на амортизацію:

$$B_{A_р} = 5600 \cdot 0{,}15 = 840 грн .$$

Інші витрати:

$$B_{I_р} = 5600 \cdot 0{,}05 = 280 грн .$$

Річні витрати на додаткові комплектуючі ПК:

$$B_{ДК_р} = 5600 \cdot 0{,}02 = 112 грн .$$

Річні витрати на ремонт ПК:

$$B_{РЕМ_р} = 5600 \cdot 0{,}06 = 336 грн .$$

Річні витрати на електроенергію для ПК:

$$B_{E_р} = 0{,}3 \cdot 1843 \cdot 1{,}44 \cdot 0{,}9 = 716{,}55 грн .$$

Річні поточні витрати на експлуатацію програмного забезпечення:

$$B_{ПК_р} = 716{,}55 + 840 + 336 + 112 + 280 = 2284{,}55 грн .$$

Таким чином, витрати машинного часу склали ($t_{маш}$):

$$t_{маш} = t_{п} + t_{отл}^{k} + t_{д}. \quad (4.26)$$

Витрати на оплату машинного часу розраховуємо за формулою:

$$B_{маш} = t_{маш} \cdot C_{ПК}. \quad (4.27)$$

Витрати на оплату машинного часу розраховуємо за формулою:

$$B_{заг} = B_{оп} + B_{маш} \quad (4.28)$$

Розрахунки:

Витрати машинного часу склали:

$$t_{маш} = 133,7 + 535 + 312 = 980,7 год.$$

Витрати на оплату машинного часу розраховуємо за формулою:

$$B_{маш} = 980,7 \cdot 0,42 = 412 грн.$$

Витрати на оплату машинного часу розраховуємо за формулою:

$$B_{заг} = 300311 + 412 = 300723 грн.$$

4.3Визначення економічної ефективності розробки проєкту

Розрахунок показників економічної ефективності розробки програмного продукту.

За міжнародним стандартам для оцінки ефективності розробки ПЗ застосовують такі показники:

- внутрішня норма дохідності;
- чистий приведений дохід;
- рентабельність;
- термін окупності.

Показник внутрішньої дохідності характеризує величину чистого прибутку (чистого валового доходу), що припадає на одиницю інвестиційних вкладень у кожному часовому інтервалі життєвого циклу проекту. Розрахунок цього показника виконується за такою формулою:

$$\sum_{i=0}^{T} \frac{D_i}{(1+q)^i} - \sum_{i=0}^{T} \frac{K_i}{(1+q)^i} = 0 \quad , \qquad (4.29)$$

де:

D_i - дохід (прибуток) у і-му періоді;

K_i - інвестиційні вкладення в і-му періоді з урахуванням інфляційних процесів; i - періоди виконання і впровадження проекту;

T - загальний період (тривалість) життєвого циклу проекту;

q - показник внутрішньої норми дохідності.

Показник інвестиційних вкладень з урахуванням інфляційних процесів обчислюємо за формулою:

$$K_i = \varphi_i \cdot R_i, \quad (4.30)$$

де:

φ_i - коефіцієнт інфляції на поточний період;

R_i - інвестиційні платежі в і-му періоді (капітальні вкладення).

Дохід від розробки ПЗ у і-му періоді розраховуємо за формулою:

$$D_i = J_i(B_i - C_i), \quad (4.31)$$

де:

B_i - ціна продажу програмного продукту в і-му періоді;

C_i - собівартість програмного продукту (фактично дорівнює сумі витрат на розробку ПЗ);

J_i - кількість ПЗ.

Вартість продажу розробленого продукту розраховують за формулою:

$$B_i = B_{заг} \cdot \left(1 + \frac{p}{100}\right), \quad (4.32)$$

де:

p - середній рівень рентабельності на поточний період.

Розрахунок:

Витрати електроенергії на розробки проекту:

$$B_{ПК} = 0,42 \cdot 1187 = 498,54 грн.$$

Формула розрахунку собівартістості проекту:

$$C_i = B_{ПК} + B_{оп}. \quad (4.33)$$

Собівартістості проекту:

$$C_i = 498,54 + 300723 = 301221,54 грн.$$

Вартість продажу розробленого продукту:

$$B_i = 301221,54 \cdot \left(1 + \frac{23}{100}\right) = 370502,5 грн.$$

Дохід (прибуток) у і-му періоді:

$$D_i = 25 \cdot (370502,5 - 300723) = 1744535 грн.$$

Інвестиційні вкладення в і-му періоді з урахуванням інфляційних:

$$K_i = 1,01500 \cdot 705161,4 = 1770705 грн.$$

Показник рентабельності інвестицій. У практиці середнього бізнесу для визначення ефективності проектних рішень широко використовується показник рентабельності інвестицій.

$$p = \frac{\sum_{i=0}^{T} \frac{D_i}{(1+q_n)^i}}{\sum_{i=0}^{T} \frac{K^i}{(1+q_n)^i}} - 1 > 0 \quad . \tag{4.34}$$

Терміну окупності визначається на
основі величини капітальних витрат по періодах розробки програмного продукту та величини фактичних чи прогнозних доходів:

$$\sum_{i=0}^{T} K_i = \sum_{i=0}^{T} D_i \quad , \tag{4.35}$$

де:

T - термін окупності,

D_i - дохід (прибуток) у поточному періоді,

K_i - капітальні витрати у поточному періоді.

Економічна ефективність полягає у відношенні результату від розробленого програмного продукту до затрачених ресурсів:

$$E = \frac{D_i}{B_{заг}} . \tag{4.36}$$

Тоді термін окупності можна розрахувати за такою формулою:

$$T = \frac{1}{E} . \tag{4.37}$$

Розрахунок:

Економічна ефективність:

$$E = \frac{1744535}{301221,54} = 5,79 .$$

Термін окупності проекту:

$$T = \frac{1}{5,79} = 0,17 = 365 \cdot 0,17 = 62,05 - доби$$

Висновок: Узагальнюючи усі вищенаведені данні можна сказати що цей проект окупиться за 2 місяці 6 годин, що є економічно ефективним проектом, так як проект який окупляться терміном до 10-и років є вигідним.

ВИСНОВКИ

У рамках проведеної кваліфікаційної роботи, було успішно впроваджено і виконано методику, яка дозволяє ідентифікувати користувачів з допомогою спеціально розробленого та налаштованого програмного забезпечення. Для цього було залучено сучасні технології та передовий досвід у цій сфері. Не зупинившись на досягнутому, було також розроблено та впроваджено додатковий метод аутентифікації користувача. Цей метод базується на використанні унікальних відбитків пальців користувача, що дозволяє забезпечити найвищий рівень безпеки й конфіденційності. Таким чином, у результаті виконання кваліфікаційної роботи сформовано висновки.

1. Проаналізовано сучасні засобів реєстрації, автентифікації користувачів та управління даними. Вивчено різні методики та інструменти для реєстрації даних та ідентифікації користувачів, зокрема за допомогою біометричних технологій, як от сканування відбитків пальців. Було проаналізовано актуальні засоби зберігання даних, з особливим акцентом на безпеку і приватність інформації. За результатами аналізу було поставлено задачу для кваліфікаційної роботи, що передбачає розробку системи з використанням найкращих засобів реєстрації, автентифікації користувачів та управління даними.

2. Охарактеризовано алгоритм ідентифікації та автентифікації користувачів. Було розроблено алгоритм реєстрації користувачів та шифрування даних на основі AES256, який дозволяє забезпечити високий рівень безпеки інформації. Розроблено структуру бази даних для зберігання інформації про користувачів та їхніх відбитках пальців. Для забезпечення надійної ідентифікації та автентифікації користувачів було запропоновано використовувати методи 2FA. Було проаналізовано переваги та недоліки цих методів, на основі чого було визначено оптимальний підхід до автентифікації користувачів. В результаті, розроблено комплексний алгоритм ідентифікації

та автентифікації користувача, який об'єднує в собі найефективніші методи та технології.

3. Здійснено реалізацію алгоритму ідентифікації та автентифікації користувача. Проведено аналіз програмного середовища та інструментарію, який дозволив визначити найефективніші технології та платформи для реалізації. Було розроблено архітектуру програмного забезпечення, що дозволяє оптимально інтегрувати необхідні алгоритми та технології. Реалізовано алгоритм ідентифікації та автентифікації користувача, використовуючи вибрані методи та інструменти. Для перевірки ефективності та надійності програмного забезпечення було проведено тестування на основі експериментів. Результати тестування демонструють високу ефективність та надійність розробленого алгоритму ідентифікації та автентифікації користувача.

4. Проведено техніко-економічне обґрунтування розробки проекту. Було визначено витрати на розробку програмної системи, що включають вартість використаного обладнання, оплату праці розробників, витрати на тестування, а також інші витрати, пов'язані з процесом розробки. Враховуючи вартість виробничих ресурсів, також було обчислено вартість розробки програмної системи. На основі оцінки витрат та потенційного доходу було визначено економічну ефективність розробки проекту. Вивчення показало, що розробка проекту є економічно виправданим, та проект окупиться за 2 місяці 6 годин, що є економічно ефективним проектом, враховуючи можливість впровадження системи на ринку та її високий потенціал для використання у сфері безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Домбровський М. О. Вдосконалення системи реєстрації та ідентифікації осіб в Україні. *VII Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі»*. 23 травня 2023 р. Тернопіль. Україна - с. 12.

2 Програми для роботи з базами даних. *Soringpcrepair*. 2023: веб-сайт. URL: https://uk.soringpcrepair.com/database-software/. (дата звернення 27.03.2023).

3 Big Data. *IT Interprice*. 2023: веб-сайт. URL: https://www.it.ua/knowledge-base/technology-innovation/big-data-bolshie-dannye. (дата звернення 30.03.2023).

4 Daemen, J. Rijmen, V. AES - The Advanced Encryption Standard. *Journal of Information Security.* 2002. 300 p.

5 Faiz Muqorrir Kaffah. E-Mail Message Encryption Using Advanced Encryption Standard (AES) and Huffman Compression Engineering. *2020 6th International Conference on Wireless and Telematics (ICWT)*. 2020. 6 p.

6 Cohn D. Hull R. A data-centric approach to modeling business operations and processes. *Business artifacts*. 2009. 7 p.

7 Aleksandr Ometov. Multi-Factor Authentication: A Survey. *Cryptography*. 2018. 31 p.

8 Abdul Razaque, Syed S. Rizvi. Triangular data privacy-preserving model for authenticating all key stakeholders in a cloud environment. *Division of Business, Engineering, and Information Sciences & Technology (Altoona)*. 2016. 20 p.

9 Qt for Python. *Qt for Python and Pyside*. 2023: веб-сайт. URL: https://doc.qt.io/qtforpython-6/index.html. (дата звернення 01.04.2023).

10 Модуль датчика відбитків пальців AS608. *3V3.COM.UA*. веб-сайт. URL: https://3v3.com.ua/product_8452.html. (дата звернення 03.04.2023).

11 USB - a brief tutorial for embedded engineers. *COMSOL*. 2015: веб-сайт. URL: Computer-solutions.co.uk.

12 Liu Xiaoyue and Li Xing. Serial Communication System of Mobile Devices and Embedded Computer Based on C/S Structure. *Future Computer Science and Education (ICFCSE) 2011 International Conference*. 2011. 20 p.

13 O. Ellahi. M. Umer. A. Raza. K. Rehman. Analyzing 2FA Phishing Attacks and Their Prevention Techniques. *2022 International Conference on Smart Information Systems and Technologies (SIST)*. 2022. 7 p.

14 PyOTP - The Python One-Time Password Library. *PyOTP documentation.* 2023: веб-сайт.URL: https://pyauth.github.io/pyotp/. (дата звернення 06.04.2023).

15 S. Dhalwar. S. Ashtekar, A. Pasupathy and N. Poojary. Computer Vision Based Vehicle Intension Finding by Understanding Driver Hand Signal. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*. 2021. 5 p.

16 M. N. Hossain. S. F. U. Zaman. T. Z. Khan. S. A. Katha. M. T. Anwar and M. I. Hossain. Implementing Biometric or Graphical Password Authentication in a Universal Three-Factor Authentication System. *2022 4th International Conference on Computer Communication and the Internet (ICCCI).* 2022. 5 p.

17 X. Zhang. F. Yin. G. Ma. B. Ge and W. Xiao. F-SQL: Fuse Table Schema and Table Content for Single-Table Text2SQL Generation. *IEEE Access*. 2020. 11 p.

18 My SQL. *My SQL TM.* 2023*:* веб-сайт. URL: https://www.mysql.com/products/workbench/. (дата звернення 06.04.2023).

19 O. Ellahi. M. Umer. A. Raza and K. Rehman. Analyzing 2FA Phishing Attacks and Their Prevention Techniques. 2022 International Conference on Smart Information Systems and Technologies (SIST). 2022. 6 p.

20 How to Maintain Data Integrity: 6 Best Practices. *HEVO*. 2021: веб-сайт. URL: https://hevodata.com/learn/data-integrity/. (дата звернення 09.04.2023).

21 Data Base Transaction. *Tutorialspoint.* 2022: веб-сайт. URL: https://www.tutorialspoint.com/dbms/dbms_transaction.html. (дата звернення 15.04.2023).

22 Основи розробки баз даних. *Microsoft.* 2021: веб-сайт. URL: https://support.microsoft.com/uk-ua/office/. (дата звернення 21.04.2023).

23 The five stages of the data analysis process. *Lighthouse Labs.* 2023: веб-сайт. URL: https://www.lighthouselabs.ca/en/blog/the-five-stages-of-data-analysis. (дата звернення 23.04.2023).

24 O. Bradeanu. D. Munteanu. Significant Location Identification Based on User Behavior Modeling. *2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*. 2008. 6 p.

25 Michael Cobb. OAuth. *TechTarget.* 2020: веб-сайт. URL: https://www.techtarget.com/searchapparchitecture/definition/OAuth. (дата звернення 24.04.2023).

26 Y. Wu and B. Qiu. Transforming a pattern identifier into biometric key generators. *2010 IEEE International Conference on Multimedia and Expo*. 2010. 4p.

27 S. Bimantoro. R. Jayadi. N. Legowo. Analysis and Design of CRM System for PT. Askrindo. *2021 International Conference on Information Management and Technology (ICIMTech).* 2021. 5p.

28 D. S. Peduru Hewa. C. Farook. A Sinhala Natural Language Interface for Querying Databases Using Natural Language Processing. *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter).* 2021, 5 p.

29 Hennig et al. Big social data analytics of changes in consumer behaviour and opinion of a TV broadcaster. *2016 IEEE International Conference on Big Data (Big Data).* 2016, 9 p.

30 Google Authenticator. *Wikipedia*. 2021: веб-сайт. URL: https://uk.wikipedia.org/wiki/Google_Authenticator. (дата звернення 26.04.2023).

31 Linda Rosencrance. Two-factor authentication (2FA). *TechTarget*. 2021: веб-сайт.URL https://www.techtarget.com/searchsecurity/definition/two-factor-authentication. (дата звернення 29.04.2023).

32 G. Arrieta et al. Increasing security in the teaching/learning process with 2FA. *2021 XI International Conference on Virtual Campus (JICV)*. 2021. 4 p.

33 Two-Factor Authentication (2FA). *Cisco*. 2023: веб-сайт. URL: https://duo.com/product/multi-factor-authentication-mfa/two-factor-authentication-2fa. (дата звернення 31.04.2023).

34 Python. *Python Software Foundation.* 2023: веб-сайт. URL: https://www.python.org/. (дата звернення 01.05.2023).

35 Dan Moore. Multi-Factor Authentication For Developers. *Fusionauth Docs*. 2021: веб-сайт. URL: https://fusionauth.io/articles/authentication/multi-factor-authentication. (дата звернення 03.05.2023).

36 S. Ghorbani Lyastani. M. Schilling. M. Neumayr. M. Backes. S. Bugiel. Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. *2020 IEEE Symposium on Security and Privacy (SP)*. 2020. 17 p.

37 Qt Style Sheets. *Qt Company*. 2023: веб-сайт. URL: https://doc.qt.io/qt-6/stylesheet.html. (дата звернення 09.05.2023).

38 What is Azure SQL Database?. *Microsoft*. 2023: веб-сайт. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql. (дата звернення 13.05.2023).

39 Як працює двофакторна аутентифікація та чому ви маєте активувати 2FA вже сьогодні. *AIN.UA*. 2022: веб-сайт. URL: https://ain.ua/2022/11/09/yak-praczyuye-dvofaktorna-autentyfikacziya-ta-chomu-vy-mayete-aktyvuvaty-2fa-vzhe-sogodni/. (дата звернення 19.05.2023).

40 Mary E. Shacklett. What is multifactor authentication and how does it work?. *TechTarget*. 2023: веб-сайт. URL:

https://www.techtarget.com/searchsecurity/definition/multifactor-authentication-MFA. (дата звернення 22.05.2023).

41 Олексій Мельник. Google Authenticator відтепер синхронізується з вашим акаунтом Google. *Mezha media.* 2023: веб-сайт. URL: https://mezha.media/2023/04/25/google-authenticator-vidteper-synkhronizuietsia-z-vashym-akauntom-google/. (дата звернення 23.05.2023).

42 Primary and Foreign Key Constraints. *Microsoft.* 2023: веб-сайт. URL: https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16. (дата звернення 24.05.2023).

43 Amazon RDS Features. *Amazon Web Services, Inc.* 2023: веб-сайт. URL: https://aws.amazon.com/rds/features/?nc1=h_ls. (дата звернення 25.05.2023).

44 Advanced Encryption Standard. *Wikipedia*. 2021: веб-сайт. URL: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard. (дата звернення 26.05.2023).

45 N. Mendjoge. A. R. Joshi. M. Narvekar. Intelligent tutoring system for Database Normalization. *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*. 2016. 6 p.

46 System testing. *TechTarget*. 2023: веб-сайт. URL: https://www.techtarget.com/searchsoftwarequality/definition/system-testing (дата звернення 26.05.2023).

47 D. Xiang and Y. Wu. Analysis and Research of Internet User Behaviors under the Context of Big Data. *2022 International Conference on Big Data, Information and Computer Network (BDICN).* 2022. 4 p.

48 Розподілені бази даних. *СумГУ*. 2022: веб-сайт. URL: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151030211833/44605/index.html. (дата звернення 27.05.2023).

49 Revert a Database to a Database Snapshot. *Microsoft*. 2023: веб-сайт. URL: https://learn.microsoft.com/en-us/sql/relational-databases/databases/revert-a-

database-to-a-database-snapshot?view=sql-server-ver16. (дата звернення 28.05.2023).

50 Craig Stedman. What is data management and why is it important? *TechTarget*. 2022: веб-сайт. URL: https://www.techtarget.com/searchdatamanagement/definition/data-management. (дата звернення 30.05.2023).

51 ДСТУ 3008:2015. Національний стандарт України. Інформація та документація. *Звіти у сфері науки і техніки. Структура та правила оформлювання.* Введ. 01.07.2017. К.: ДП "УкрНДНЦ, 2016. 25 с

52 ДСТУ 8302:2015. Інформація та документація. *Бібліографічне посилання. Загальні положення та правила складання*. Введ. 01.07.2016. К.: ДП «УкрНДНЦ», 2017. 16 с.

53 Методичні вказівки до випускних кваліфікаційних робіт освітнього рівня “Бакалавр” спеціальності “Комп’ютерна інженерія”/ О.М. Березький, Г.М. Мельник, Л.О.Дубчак, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2021. – 52 с.

54 Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп’ютерних систем”/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

55 Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

56 Електронна таблиця. *Вікіпедія.* 2022: веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8F. (дата звернення 25.03.2023).