Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

“Розробка програмного модуля захисту від несанкціонованого копіювання даних CRM- системи підприємств”

Виконав: студент 2(4) курсу, гр.2КІТС-20б

спеціальності 125– Кібербезпека

Освітня програма – Кібербезпека інформаційних технологій та систем

(шифр і назва напряму підготовки, спеціальності)

Вахній Д.Д.

(прізвище та ініціали)

Керівник: к.т.н., доц., доцент каф. МБІС

Сачанюк-Кавецька Н.В.

(прізвище та ініціали)

«6» червня 2024 р.

Рецензент: к.т.н., доц., доцент каф. ОТ

Томчук М.А

(прізвище та ініціали)

«6» червня 2024 р.

Допущено до захисту

Голова секції УБ кафедри МБІС

д.т.н., проф. Яремчук Ю.Є.

“6” червня 2024р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

д.т.н., проф. Яремчук Ю.Є.
“ 22 ” березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Вахнію Денису Дмитровичу
(прізвище, ім`я, по-батькові)

1. Тема роботи - Розробка програмного модуля захисту від несанкціонованого копіювання даних CRM- системи підприємств
Керівник роботи: Сачанюк-Кавецька Наталія Василівна, к.ф.-м.н., доцент, затверджені наказом вищого навчального закладу від “ 11 ” березня 2024року № 80
2. Термін подання студентом роботи 10.06.2024 р.
3. Вихідні дані до роботи Windows: середовище програмування Visual Studio Code; мови програмування для веб-розробки - Python.

4. Зміст текстової частини:
1. Аналіз існуючих CRM систем та методів їх розробки
2. Розроблення алгоритму роботи модуля захисту CRM системи
3. Розроблення програмного модуля CRM системи згідно розробленого алгоритму

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
Презентація у форматі PowerPoint з ілюстраціями архітектури програмного модуля, скріншотами інтерфейсу та результатами тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ І- ІІІ	Сачанюк-Кавецька Н.В. к.т.н.,доцент	22.03.2024	22.03.2024

7.Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Ознайомлення з темою та постановкою завдання	22.03.2024	24.03.2024	
2	Пошук та вивчення літератури за темою дослідження	25.03.2024	31.03.2024	
3	Аналіз існуючих CRM систем	01.04.2024	07.04.2024	
4	Вибір мови програмування та середовища розробки	08.04.2024	14.04.2024	
5	Обґрунтування вибору фреймворку	15.04.2024	21.04.2024	
6	Розроблення функціоналу аутентифікації. Авторизації та шифрування	22.04.2024	28.04.2024	
7	Реалізація функціоналу моніторингу та системи ролей	29.04.2024	05.05.2024	
8	Тестування програмного модуля	06.05.2024	12.05.2024	
9	Оформлення текстової частини роботи	13.05.2024	22.05.2024	
10	Переддипломний захист			
11	Захист бакалаврської дипломної роботи			

Студент (підпис) **Вахній Д.Д.** (прізвище та ініціали)

Керівник роботи (підпис) **Сачанюк-Кавецька Н.В.** (прізвище та ініціали)

АНОТАЦІЯ

Дана робота присвячена розробці та впровадженню програмного модуля для захисту даних CRM-системи від несанкціонованого доступу та копіювання. Враховуючи зростаючі ризики кіберзлочинності та важливість захисту конфіденційної інформації, розробка надійних методів безпеки є критичною для будь-якої сучасної організації.

Мета дослідження

Основною метою дослідження є створення комплексного програмного рішення, яке забезпечить надійний захист даних CRM-системи шляхом впровадження механізмів аутентифікації, авторизації, шифрування, а також аудиту та моніторингу активності користувачів.

Завдання дослідження

1. Провести аналіз існуючих методів захисту даних CRM-систем та визначити найбільш ефективні з них.

2. Розробити алгоритми аутентифікації та авторизації користувачів, що відповідають сучасним вимогам безпеки.

3. Впровадити криптографічні методи шифрування даних для забезпечення їх цілісності та конфіденційності.

4. Інтегрувати системи аудиту та моніторингу для запису дій користувачів та виявлення підозрілої активності.

5. Провести тестування розробленого програмного модуля для оцінки його ефективності та відповідності вимогам до безпеки.

Методи дослідження

В процесі роботи були використані методи системного аналізу для оцінки існуючих підходів до захисту даних, а також методи програмування для реалізації запропонованих алгоритмів. Криптографічні методи шифрування та хешування були застосовані для забезпечення безпеки зберігання та передачі даних.

1. Розроблено та впроваджено комплексний програмний модуль для захисту даних CRM-системи, що включає механізми аутентифікації, авторизації, шифрування, аудиту та моніторингу.

2. Впроваджені механізми аутентифікації забезпечують захист від несанкціонованого доступу за допомогою криптографічних методів.

3. Авторизація користувачів здійснюється на основі ролей, що дозволяє контролювати доступ до різних ресурсів системи.

4. Дані шифруються надійними криптостійкими алгоритмами, що гарантує їх конфіденційність та цілісність.

5. Системи аудиту та моніторингу забезпечують запис всіх дій користувачів та виявлення підозрілої активності, що дозволяє своєчасно реагувати на потенційні загрози.

Розроблений програмний модуль успішно вирішує завдання захисту даних CRM-системи від несанкціонованого доступу та копіювання. Він забезпечує високий рівень безпеки та може бути використаний як основа для подальшого вдосконалення систем захисту інформації в різних галузях. Отримані результати підтверджують ефективність запропонованих методів та їх відповідність сучасним вимогам інформаційної безпеки.

Ключові слова

CRM-система, аутентифікація, авторизація, шифрування, аудит, моніторинг, інформаційна безпека, криптографія.

ABSTRACT.

This paper is devoted to the development and implementation of a software module for protecting CRM system data from unauthorized access and copying. Given the growing risks of cybercrime and the importance of protecting confidential information, the development of reliable security methods is critical for any modern organization.

The purpose of the study

The main goal of the study is to create a comprehensive software solution that will provide reliable protection of CRM system data by implementing authentication, authorization, encryption mechanisms, as well as auditing and monitoring user activity.

Research objectives.

1. To analyze the existing methods of protecting CRM system data and determine the most effective ones.

2. Develop user authentication and authorization algorithms that meet modern security requirements.

3. Implement cryptographic data encryption methods to ensure data integrity and confidentiality.

4. Integrate audit and monitoring systems to record user actions and detect suspicious activity.

5. Test the developed software module to evaluate its effectiveness and compliance with security requirements.

Research methods

In the course of the work, we used system analysis methods to evaluate existing approaches to data protection, as well as programming methods to implement the proposed algorithms. Cryptographic encryption and hashing methods were used to ensure the security of data storage and transmission.

1. A comprehensive software module for CRM system data protection, including authentication, authorization, encryption, audit and monitoring mechanisms, was developed and implemented.

2. The implemented authentication mechanisms provide protection against unauthorized access using cryptographic methods.

3. User authorization is based on roles, which allows you to control access to various system resources.

4. Data is encrypted with reliable cryptographic algorithms, which guarantees its confidentiality and integrity.

5. Audit and monitoring systems record all user actions and detect suspicious activity, which allows you to respond to potential threats in a timely manner.

The developed software module successfully solves the problem of protecting CRM system data from unauthorized access and copying. It provides a high level of security and can be used as a basis for further improvement of information security systems in various industries. The obtained results confirm the effectiveness of the proposed methods and their compliance with modern information security requirements.

Keywords.

CRM system, authentication, authorization, encryption, audit, monitoring, information security, cryptography.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де дані є одним з найцінніших активів підприємства, захист інформації від несанкціонованого доступу та копіювання набуває першочергового значення. Особливо це стосується CRM-систем (систем управління взаємовідносинами з клієнтами), які містять конфіденційні дані про клієнтів, продажі, маркетингові стратегії та інші важливі аспекти діяльності компанії. Витік такої інформації може призвести до значних фінансових втрат, підірвати репутацію компанії та спричинити юридичні проблеми.

Актуальність теми дослідження зумовлена зростаючою кількістю кібератак та випадків витоку даних, що підтверджує необхідність розробки надійних та ефективних методів захисту CRM-систем.

Об'єктом дослідження є процес захисту даних в CRM-системах підприємств від несанкціонованого копіювання.

Предметом дослідження виступають методи, алгоритми та програмні засоби, що забезпечують захист даних CRM-систем від несанкціонованого копіювання, а також методи оцінки їх ефективності.

Мета дослідження полягає у розробці та впровадженні програмного модуля, який забезпечить комплексний захист даних CRM-системи від несанкціонованого копіювання, відповідаючи при цьому таким ключовим вимогам:

- Ефективність: Гарантувати високий рівень захисту даних від несанкціонованого копіювання, використовуючи сучасні методи шифрування, контролю доступу та моніторингу активності користувачів.
- Простота використання: Забезпечити інтуїтивно зрозумілий інтерфейс та мінімальну кількість налаштувань для зручності використання співробітниками підприємства.

- Сумісність: Розробити модуль, який легко інтегрується з різними CRM-системами та операційними середовищами, забезпечуючи його універсальність та широке застосування.
- Економічність: Забезпечити оптимальне співвідношення вартості впровадження та експлуатації модуля з рівнем захисту, який він надає.

Завдання дослідження:

1. Провести комплексний аналіз існуючих методів та технологій захисту даних в CRM-системах, включаючи шифрування, контроль доступу, водяні знаки, аудит та інші. Виявити їх переваги, недоліки та обмеження.
2. Дослідити типові загрози та вразливості CRM-систем щодо несанкціонованого копіювання даних, включаючи атаки зловмисників, інсайдерські загрози та ненавмисні витоки інформації.
3. Розробити алгоритм захисту даних CRM-системи, який поєднуватиме різні методи захисту, враховуючи виявлені загрози та вразливості.
4. Реалізувати розроблений алгоритм у вигляді програмного модуля, використовуючи сучасні технології та інструменти розробки програмного забезпечення.
5. Провести тестування розробленого програмного модуля в різних умовах та середовищах, оцінити його ефективність, надійність та продуктивність.

Новизна дослідження полягає у створенні комплексного підходу до захисту даних CRM-систем від несанкціонованого копіювання, який враховує специфіку цих систем та сучасні загрози.

Практична значимість дослідження полягає у можливості використання розробленого програмного модуля для підвищення рівня інформаційної безпеки підприємств, запобігання витоку конфіденційних даних та мінімізації пов'язаних з цим ризиків.

1. АНАЛІЗ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1. Актуальність та значимість дослідження

Актуальність дослідження захисту даних CRM-систем від несанкціонованого копіювання обумовлюється низкою ключових факторів, що роблять цю тему вкрай важливою для сучасного бізнесу:

1. Стрімкий розвиток CRM-систем:

- CRM-системи стають невід'ємною частиною інформаційних систем підприємств, автоматизуючи та оптимізуючи процеси взаємодії з клієнтами [1].
- Їх широке впровадження призводить до зростання обсягів даних, що зберігаються: інформація про клієнтів, партнерів, угоди, маркетингові кампанії тощо[3].
- Ці дані мають значну цінність як для самого підприємства, так і для потенційних зловмисників[17].

2. Зростання кількості кібератак та викрадення даних:

Зловмисники постійно вдосконалюють свої методи, використовуючи різноманітні способи атак, включаючи фішинг, malware та social engineering[22]:

  - **Фішинг:** Розсилка шкідливих електронних листів, які спонукають користувачів розкрити конфіденційну інформацію або завантажити заражені файли.
  - **Malware:** Використання шкідливого програмного забезпечення для отримання доступу до комп'ютерних систем та даних.
  - **Social engineering:** Маніпулювання користувачами для розкриття конфіденційної інформації або виконання дій, які можуть призвести до витоку даних.
- Зростання кількості та складності кібератак робить захист даних CRM-систем вкрай важливим завданням[3].

3. Збільшення обсягу та цінності даних, що зберігаються в CRM-системах:

- Сучасні CRM-системи містять широкий спектр даних, які можуть бути використані зловмисниками для:
    - **Крадіжки особистої інформації:** Імена, адреси, номери телефонів, дані платіжних карток тощо.
    - **Промислового шпигунства:** Викрадення інформації про продукти, послуги, маркетингові стратегії та інші комерційні таємниці.
    - **Фінансового шахрайства:** Використання викрадених даних для здійснення шахрайських операцій, таких як крадіжка коштів з банківських рахунків або використання кредитних карток.
- Збільшення обсягу та цінності даних робить їх більш привабливою мішенню для зловмисників.

4. Недостатність існуючих методів захисту даних CRM-систем:

- Багато традиційних методів захисту даних, таких як брандмауери та антивіруси, не є достатньо ефективними проти сучасних кібератак.
- Необхідність розробки нових, більш ефективних методів захисту, які відповідають викликам сучасного цифрового середовища.

5. Необхідність розробки нових, більш ефективних методів захисту:

- Існуючі методи захисту даних CRM-систем часто не здатні забезпечити комплексний захист від усіх видів загроз.
- Необхідність розробки нових методів, які ґрунтуються на сучасних технологіях та підходах до кібербезпеки.

Значимість дослідження полягає в тому, що воно дозволить:

- **Розробити ефективний програмний модуль захисту даних CRM-систем від несанкціонованого копіювання:**

- Цей модуль буде блокувати спроби копіювання даних, шифрувати дані при їх зберіганні та передачі, а також забезпечувати контроль доступу до даних.
- Використання модуля дозволить підприємствам зберегти конфіденційність інформації, захистити себе від кібератак та підвищити рівень безпеки своїх даних.

- **Підвищити рівень кібербезпеки підприємств:**
  - Результати дослідження можуть бути використані для розробки та впровадження комплексних систем захисту даних CRM-систем, що включають програмні модулі, технічні та організаційні заходи.
  - Підвищення рівня кібербезпеки дозволить підприємствам зменшити ризики витоку даних, кібератак та інших загроз, що може призвести до значних фінансових втрат, втрати репутації та порушення нормативних вимог.
- **Захистити конфіденційну інформацію:**
  - Розроблений програмний модуль захисту даних CRM-систем від несанкціонованого копіювання дозволить зберегти конфіденційність інформації про клієнтів, партнерів, угоди та інші дані, що мають значну цінність як для самого підприємства, так і для потенційних зловмисників.
- **Збільшити довіру клієнтів:**
  - Захист даних клієнтів є важливою складовою формування довіри до підприємства.
  - Використання надійних методів захисту даних може стати конкурентною перевагою та сприяти залученню нових клієнтів.
- **Відповідати нормативним вимогам:**
  - Існує ряд нормативних актів, які встановлюють вимоги до захисту даних, наприклад, GDPR (Загальний регламент про захист даних) в Європейському Союзі.

- Розробка та впровадження програмного модуля захисту даних CRM-систем дозволить підприємствам відповідати цим вимогам та уникнути штрафних санкцій.

Очікувані результати дослідження:

- Розробка алгоритму захисту даних CRM-системи від несанкціонованого копіювання;
- Створення програмного модуля реалізації алгоритму;
- Тестування та оцінка ефективності програмного модуля;

Практична значимість дослідження:

- Розроблений програмний модуль може бути використаний підприємствами для захисту даних CRM-системи від несанкціонованого копіювання;
- Результати дослідження можуть бути використані для удосконалення існуючих методів захисту даних CRM-систем;
- Дослідження може стати базою для подальших досліджень в галузі захисту інформації.

1.2. Огляд існуючих методів захисту даних CRM-систем

Для захисту даних CRM-систем від несанкціонованого доступу та копіювання використовується широкий спектр методів, які можна умовно поділити на кілька категорій: технічні, організаційні, програмні та фізичні[5]:

1. Технічні методи:

- **Шифрування даних:** Шифрування даних робить їх нечитабельними для несанкціонованих користувачів, навіть якщо їм вдасться отримати доступ до них[29]. Існує багато різних алгоритмів шифрування, які використовуються для захисту даних CRM-систем, наприклад, AES, RSA, Blowfish тощо [7].
- **Контроль доступу:** Контроль доступу дозволяє регулювати, хто може отримувати доступ до даних CRM-системи та які дії вони можуть виконувати з ними[8]. Існує кілька методів контролю доступу, таких як:

- **Аутентифікація:** Перевірка особистості користувача перед наданням йому доступу до системи.
- **Авторизація:** Надання користувачеві дозволів на виконання певних дій з даними.
- **Мандатування:** Надання користувачеві доступу до даних лише на час виконання певного завдання.

- **Фільтрування трафіку:** Фільтрування трафіку дозволяє блокувати шкідливе програмне забезпечення, фішингові атаки та інші загрози.
- **Захисні стіни:** Захисні стіни створюють бар'єр між CRM-системою та Інтернетом, дозволяючи лише авторизований трафік.
- **Системи виявлення вторгнень (IDS):** IDS моніторять мережевий трафік та дії користувачів на предмет підозрілої активності, яка може свідчити про кібератаку.
- **Системи запобігання вторгненням (IPS):** IPS можуть автоматично блокувати підозрілу активність, щоб запобігти кібератакам.

2. Організаційні методи:

- **Політики безпеки:** Політики безпеки визначають правила та процедури, які повинні дотримуватися співробітники при роботі з даними CRM-системи.
- **Навчання персоналу:** Навчання персоналу з питань кібербезпеки допомагає їм розуміти загрози та знати, як їх уникнути.
- **Управління інцидентами:** Управління інцидентами передбачає наявність плану дій на випадок кібератаки або іншого інциденту інформаційної безпеки.

3. Програмні методи:

- **Антивіруси:** Антивірусні програми сканують комп'ютери та сервери на наявність шкідливого програмного забезпечення [5].

- **Захист від програм-вимагачів:** Захист від програм-вимагачів допомагає захистити дані від шифрування та вимагання викупу зловмисниками[5]
- **Захист від DDoS-атак:** Захист від DDoS-атак допомагає захистити CRM-систему від перевантаження трафіком, який може призвести до її недоступності[5].
- **Системи резервного копіювання та відновлення:** Системи резервного копіювання та відновлення дозволяють відновити дані CRM-системи у разі їх втрати або пошкодження.

4. Фізичні методи:

Контроль фізичного доступу:

- **Обмеження доступу до приміщення:** Доступ до приміщення, де знаходяться комп'ютери та сервери CRM-системи, має бути обмежений лише авторизованим співробітникам[5].
- **Використання пропускної системи:** Пропускна система дозволяє контролювати, хто може входити до приміщення та коли.
- **Установка біометричних систем:** Біометричні системи, такі як сканери відбитків пальців або розпізнавання облич, дозволяють ідентифікувати людей за їхніми унікальними фізичними характеристиками.
- **Шифрування жорстких дисків:** Шифрування жорстких дисків робить їх нечитабельними для несанкціонованих користувачів, навіть якщо їм вдасться викрасти комп'ютери або сервери.

Відеоспостереження:

- **Установка камер відеоспостереження:** Камери відеоспостереження дозволяють фіксувати все, що відбувається в приміщенні, де знаходяться комп'ютери та сервери CRM-системи[5].
- **Запис відео:** Записане відео можна використовувати для розслідування інцидентів інформаційної безпеки.

- **Віддалений доступ до відео:** Відео з камер спостереження можна переглядати віддалено через Інтернет.

Системи сигналізації:

- **Системи сигналізації про проникнення:** Системи сигналізації про проникнення попереджають про несанкціоноване проникнення на територію або в приміщення, де знаходяться комп'ютери та сервери CRM-системи[5].
- **Системи сигналізації про пожежу:** Системи сигналізації про пожежу попереджають про пожежу в приміщенні, де знаходяться комп'ютери та сервери CRM-системи.
- **Системи сигналізації про затоплення:** Системи сигналізації про затоплення попереджають про затоплення в приміщенні, де знаходяться комп'ютери та сервери CRM-системи.

Вибір методів захисту даних CRM-системи залежить від:

- **Розміру та типу підприємства:**
  - Малі та середні підприємства (МСП) можуть використовувати менш складні та дорогі методи захисту, такі як контроль фізичного доступу та системи сигналізації.
  - Великі підприємства можуть використовувати більш складні та дорогі методи захисту, такі як шифрування даних, контроль доступу та системи виявлення вторгнень.
- **Галузі діяльності підприємства:**
  - Підприємства, які працюють у галузях, де використовуються чутливі дані, наприклад, у фінансовій або медичній сфері, можуть використовувати більш жорсткі методи захисту даних.
- **Бюджету підприємства:**
  - Вибір методів захисту даних також залежить від бюджету, який підприємство може виділити на цю статтю витрат.

Важливо використовувати комплексний підхід до захисту даних CRM-системи, який включає в себе:

- Використання технічних, організаційних, програмних та фізичних методів захисту.
- Регулярне оновлення методів захисту даних відповідно до нових загроз.
- Навчання персоналу з питань кібербезпеки.
- Проведення регулярних тестувань на проникнення.
- Розробку плану дій на випадок кібератаки або іншого інциденту інформаційної безпеки.

Захист даних CRM-системи є важливим завданням для будь-якого підприємства. Використання ефективних методів захисту даних може допомогти захистити інформацію від несанкціонованого доступу, копіювання та витоку, що може призвести до значних фінансових втрат, втрати репутації та порушення нормативних вимог.

1.3. Очікувані результати

Очікуваними результатами дослідження є:

1. Розробка програмного модуля захисту даних CRM-системи від несанкціонованого копіювання:

- Модуль повинен блокувати будь-які спроби копіювання даних CRM-системи, включаючи копіювання до сховища, друк на принтері, створення скріншотів та інші подібні дії.
- Модуль повинен шифрувати дані CRM-системи при їх зберіганні та передачі. Це робить їх нечитабельними для несанкціонованих користувачів, навіть якщо їм вдасться отримати доступ до них.
- Модуль повинен забезпечувати контроль доступу до даних CRM-системи. Це означає, що лише авторизовані користувачі повинні мати можливість переглядати, редагувати та копіювати дані.

- Модуль повинен вести журнал всіх дій, які виконуються з даними CRM-системи. Це дозволить відстежувати, хто і коли отримував доступ до даних, а також які дії були виконані.

2. Програмний модуль повинен відповідати всім поставленим вимогам:

- **Функціональні вимоги:**
  - Блокування копіювання даних.
  - Шифрування даних.
  - Контроль доступу.
  - Аудит.
- **Нефункціональні вимоги:**
  - Ефективність.
  - Надійність.
  - Безпека.
  - Простота використання.
  - Сумісність.

3. Модуль буде простим у використанні та налаштуванні:

- Інтерфейс користувача модуля повинен бути простим та інтуїтивно зрозумілим.
- Налаштування модуля повинні бути простими та не вимагати спеціальних знань.

4. Модуль буде сумісний з різними CRM-системами:

- Модуль повинен бути розроблений таким чином, щоб його можна було легко інтегрувати з різними CRM-системами.
- Модуль повинен підтримувати різні протоколи та стандарти обміну даними.

5. Додаткові результати:

- **Визначення рівня критичності даних:**

- Розробка методики визначення рівня критичності даних CRM-системи.
- Рівень критичності даних визначає, наскільки важливою є інформація та які заходи захисту необхідно застосовувати.

- **Інтеграція з існуючими системами безпеки:**
  - Дослідження можливостей інтеграції розробленого програмного модуля з існуючими системами безпеки підприємства.
  - Це дозволить створити більш комплексну систему захисту даних.

Очікується, що розробка програмного модуля захисту даних CRM-системи від несанкціонованого копіювання дозволить:

- Підвищити рівень захищеності даних підприємств.
- Зменшити ризики витоку даних та кібератак.
- Підвищити довіру клієнтів до підприємств.
- Розробити методичні рекомендації щодо захисту даних CRM-систем.

Практична значимість дослідження полягає в тому, що його результати можуть бути використані:

- Підприємствами для захисту даних CRM-систем від несанкціонованого копіювання.
- Розробниками CRM-систем для підвищення безпеки своїх продуктів.
- Фахівцями з інформаційної безпеки для розробки комплексних систем захисту даних.
- Державними органами для розробки стандартів та регуляторних вимог щодо захисту даних.
- Навчальними закладами для підготовки фахівців з інформаційної безпеки.

Вимірювання успіху дослідження:

Успіх дослідження буде вимірюватися за наступними показниками:

- Функціональність програмного модуля;
- Ефективність програмного модуля;

- Надійність програмного модуля;
- Безпека програмного модуля;
- Простота використання програмного модуля;
- Сумісність програмного модуля;

1.4. Висновок

У першому розділі цієї роботи було розглянуто проблему захисту даних CRM-систем від несанкціонованого копіювання. Було проведено огляд існуючих методів захисту даних, а також сформульовані завдання дослідження. Очікуваними результатами дослідження є розробка програмного модуля захисту даних CRM-системи від несанкціонованого копіювання, який буде відповідати всім поставленим вимогам, а також буде простим у використанні та налаштуванні.

Основні висновки першого розділу:

- Захист даних CRM-систем є важливим завданням для будь-якого підприємства.
- Існує багато різних методів захисту даних CRM-систем, кожен з яких має свої переваги та недоліки.
- Вибір методів захисту даних залежить від розміру та типу підприємства, галузі його діяльності, бюджету, а також від існуючих загроз.
- Для ефективного захисту даних CRM-систем необхідно використовувати комплексний підхід, який включає в себе використання технічних, організаційних, програмних та фізичних методів захисту.

Очікується, що результати цього дослідження допоможуть:

- Підвищити рівень захищеності даних CRM-систем на підприємствах.
- Зменшити ризики витоку даних та кібератак.
- Підвищити довіру клієнтів до підприємств.
- Розробити методичні рекомендації щодо захисту даних CRM-систем.

Практична значимість дослідження полягає в тому, що його результати можуть бути використані:

- Підприємствами для захисту даних CRM-систем від несанкціонованого копіювання.
- Розробниками CRM-систем для підвищення безпеки своїх продуктів.
- Фахівцями з інформаційної безпеки для розробки комплексних систем захисту даних.
- Державними органами для розробки стандартів та регуляторних вимог щодо захисту даних.
- Навчальними закладами для підготовки фахівців з інформаційної безпеки.

У наступних розділах цієї роботи буде детально розглянуто розробку програмного модуля захисту даних CRM-системи від несанкціонованого копіювання. Буде проведено аналіз існуючих методів захисту даних, розроблено алгоритм захисту даних, створено програмний модуль реалізації алгоритму, а також проведено тестування та оцінку програмного модуля.

2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО МОДУЛЯ

2.1. Алгоритм модуля захисту даних CRM-системи

Цей алгоритм розроблений для захисту даних CRM-системи від несанкціонованого копіювання. Алгоритм включає в себе наступні етапи:

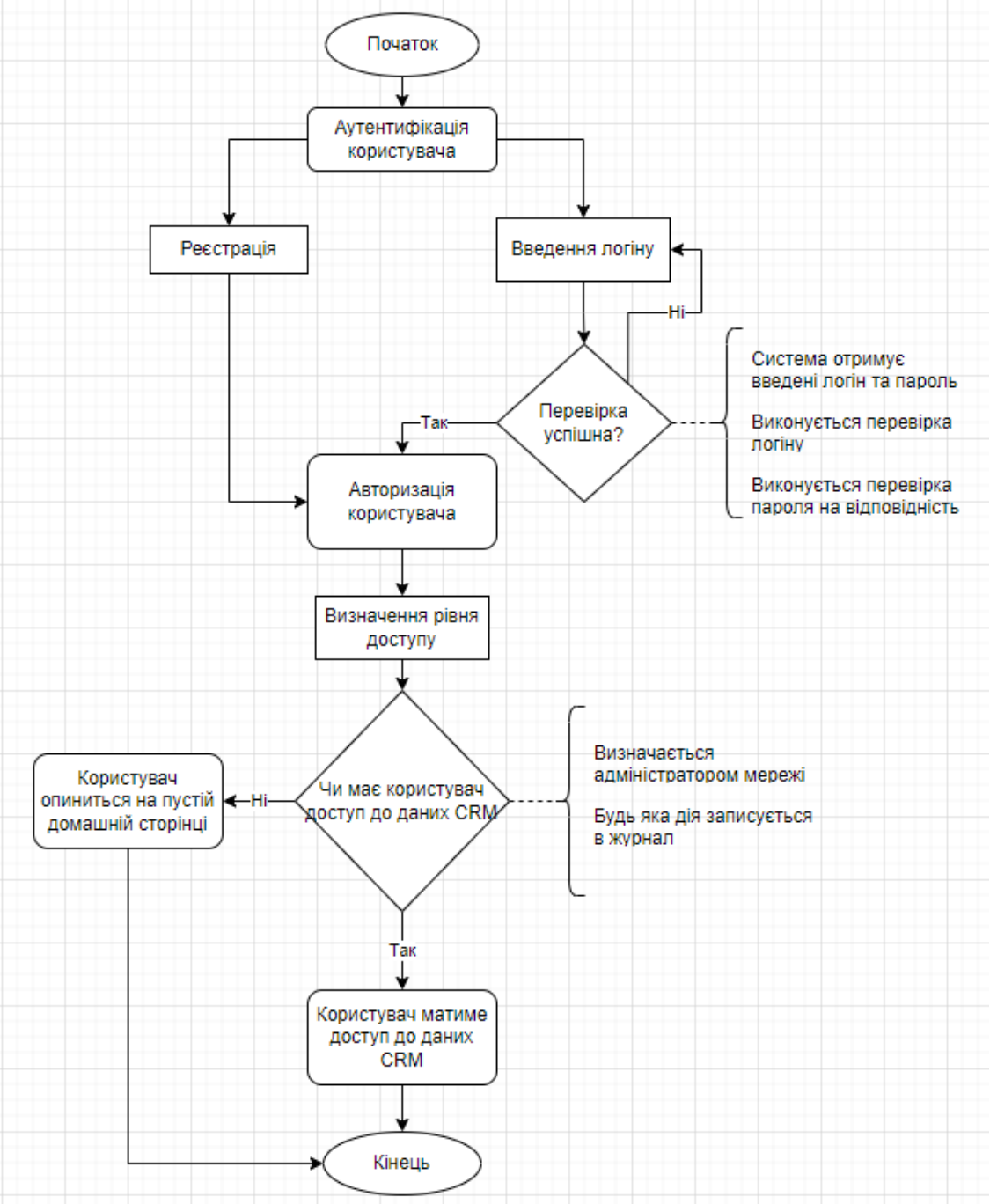


Рисунок. 2.1 Блокова схема алгоритму програмного модуля CRM

Крок перший. Аутентифікація користувача

1. Введення логіну та пароля:
    - Користувач вводить свій логін та пароль у відповідні поля на сторінці входу CRM-системи.
    - Логін та пароль повинні відповідати формату та мінімальним вимогам до складності, встановленим системою.
2. Перевірка даних:
    - Система отримує введені логін та пароль.
    - Виконується перевірка логіну
    - Виконується перевірка пароля на відповідність мінімальним вимогам до складності (довжина, наявність спеціальних символів, тощо).
3. Аутентифікація:
    - Система порівнює введений логін та пароль з даними, що зберігаються в безпечному сховищі користувачів CRM-системи.
    - Використовуються криптографічні методи для порівняння паролів, щоб уникнути витоку хешів паролів у разі компрометації системи.
4. Успішна аутентифікація:
    - Якщо логін та пароль збігаються, система:
        - Генерує унікальний сеансовий ключ для користувача.
        - Зашифровує сеансовий ключ за допомогою симетричного алгоритму шифрування.
        - Зберігає зашифрований сеансовий ключ у файлі cookie на клієнтському комп'ютері.
        - Перенаправляє користувача на головну сторінку CRM-системи.
5. Невдала аутентифікація:

- Якщо логін та/або пароль не відповідають даним у сховищі користувачів, система:
  - Відображає повідомлення про помилку, що логін та/або пароль невірні.
  - Записує інформацію про спробу невдалої аутентифікації до журналу подій.

Крок 2. Авторизація користувача

1. Визначення рівня доступу:
   - Система отримує зашифрований сеансовий ключ користувача з файлу cookie.
   - Розшифровує сеансовий ключ за допомогою симетричного алгоритму шифрування.
   - Використовує сеансовий ключ для пошуку запису про поточного користувача в сховищі користувачів.
   - З запису про користувача система визначає його роль та рівень доступу до даних CRM-системи.
2. Контроль доступу:
   - Система перевіряє доступ користувача до кожного запитуваного ним ресурсу (сторінки, функції, даних) CRM-системи.
   - Доступ надається лише до тих ресурсів, які відповідають рівню доступу користувача.
   - Для ресурсів, до яких доступ заборонено, система:
     - Відображає повідомлення про помилку, що користувач не має дозволу на доступ до даного ресурсу.
     - Записує інформацію про спробу несанкціонованого доступу до журналу подій.

Крок 3. Шифрування даних

Перед тим, як дані CRM-системи будуть передані на клієнтський комп'ютер, вони шифруються за допомогою симетричного або асиметричного алгоритму шифрування.

1. Шифрування.
    - Шифрування з допомогою надійного криптостійкого алгоритму
2. Зберігання ключів шифрування:
    - Ключі шифрування зберігаються в безпечному місці, недоступному для несанкціонованого доступу.
3. Контроль цілісності даних
    - Для забезпечення цілісності даних під час передачі та зберігання використовується метод криптографічного хешування.

Крок 4. Аудит та моніторинг

1. Аудит доступу: Система записує всі дії користувачів у CRM-системі
2. Моніторинг активності: Система постійно моніторить активність користувачів та виявляє підозрілу поведінку, таку як несанкціонований доступ до даних або спроби злому.
3. Оповіщення: У разі виявлення підозрілої активності система генерує оповіщення для адміністраторів.

2.2. Вибір мови програмування та інструментальних засобів

Для розробки модуля захисту CRM-системи я буду використовувати мову програмування Python.

У контексті розробки модуля захисту для CRM-системи, мова програмування Python виступає привабливим варіантом завдяки своїй гнучкості, широкому спектру інструментів та активній спільноті розробників. Однак, як і будь-яка технологія, Python має свої сильні та слабкі сторони, які необхідно враховувати при прийнятті рішення.

Переваги Python

1. **Простота використання та швидкість розробки:** Python є популярною мовою програмування для розробки веб-додатків, завдяки своїй простоті, читабельності та великій кількості бібліотек[14], що значно спрощує процес розробки та підтримки коду. Це особливо важливо при створенні модуля захисту, де помилки можуть мати серйозні наслідки.
2. **Багата екосистема бібліотек:** Python має величезну кількість бібліотек, спеціально розроблених для завдань безпеки. Наприклад, бібліотека cryptography надає високорівневі інтерфейси для криптографічних операцій, таких як шифрування, дешифрування та хешування. Бібліотека PyNaCl пропонує безпечні реалізації криптографічних примітивів, а bcrypt та scrypt дозволяють надійно зберігати паролі.
3. **Активна спільнота та підтримка:** Велика спільнота Python забезпечує постійну підтримку, оновлення та виправлення помилок у бібліотеках безпеки. Це допомагає підтримувати модуль захисту в актуальному стані та відповідати сучасним загрозам.
4. **Кросплатформність:** Python є кросплатформною мовою, що дозволяє розгорнути модуль захисту на різних операційних системах без значних змін у коді[6].

Обмеження Python

1. **Продуктивність:** Як інтерпретована мова, Python може поступатися компільованим мовам, таким як C або C++, у швидкості виконання. Однак, для більшості CRM-систем, де продуктивність модуля захисту не є критичним фактором, це не є суттєвим обмеженням.
2. **Вбудована безпека:** У порівнянні з деякими іншими мовами, Python не має такого ж рівня вбудованих механізмів безпеки[6]. Це означає, що розробники повинні бути особливо уважними до питань безпеки та використовувати відповідні бібліотеки та практики.

Як результат, Python є ефективним інструментом для розробки модуля захисту CRM-системи, особливо якщо вимоги до безпеки є середніми. Його простота, гнучкість та багата екосистема бібліотек дозволяють швидко створювати та підтримувати надійний модуль захисту.

Якщо CRM-системі потрібні високорівневі методи захисту даних та необхідний дуже високий рівень безпеки, то краще використовувати іншу мову програмування, наприклад C++ або Java[6]. Але в даному досліджені, як приклад CRM системи, я буду створювати просту систему, яка написана на мові Python з використанням фреймворку Django.

Django - це високорівневий веб-фреймворк на Python, який дозволяє швидко створювати складні веб-додатки[15]. Він полегшує швидке створення складних веб-додатків з акцентом на чистоту дизайну та прагматизм.

Особливості Django:

- **Швидка розробка:** Django пропонує широкий спектр компонентів та інструментів, які значно прискорюють процес розробки веб-додатків.
- **Чистий та прагматичний дизайн:** Django заохочує до написання чіткого, читабельного та елегантного коду, що робить ваші веб-додатки легкими для розуміння та обслуговування.
- **Багатство функцій:** Django постачається з широким набором вбудованих функцій, таких як система аутентифікації користувачів, авторизація, адміністрування сайту, шаблони та багато іншого.
- **Безпека:** Django розроблений з урахуванням безпеки, що допомагає вам уникнути поширених помилок та вразливостей. Django має вбудовані засоби безпеки, такі як захист від CSRF-атак, XSS-атак та SQL-ін'єкцій[28].
- **Масштабованість:** Django здатний підтримувати веб-додатки будь-якого розміру, від невеликих сайтів до багатокористувацьких платформ з мільйонами користувачів.

- **Гнучкість:** Django можна налаштувати та розширити для задоволення будь-яких потреб вашого веб-додатку.
- **Спільнота:** Django має велику та активну спільноту розробників, які готові допомогти та поділитися своїми знаннями.

Архітектура Django:

Django побудований на основі моделі MVC (Model-View-Controller), яка розділяє логіку веб-додатку на три чітко визначені частини:

- **Моделі:** Представляють дані вашого веб-додатку.
- **Вигляди:** Створюють HTML-відповіді на запити користувачів.
- **Контролери:** Поєднують моделі та погляди, обробляючи логіку запитів.

Використання Django:

Django використовується для створення широкого спектру веб-додатків, включаючи:

- **Веб-сайти:** Блоги, форуми, інтернет-магазини, новинні сайти тощо.
- **Веб-додатки:** Системи управління контентом (CMS), CRM, ERP тощо.
- **API:** RESTful API для мобільних та веб-додатків.

Бібліотеки Python

У даній роботі будуть використані вбудовані у фреймворк програмні рішення, включаючи наступні зовнішні бібліотеки:

- **PyNaCl:** Ця бібліотека Python, яка надає високорівневі інтерфейси для криптографічних операцій, таких як шифрування, дешифрування та хешування[16]. Вона використовує криптографічно стійкі алгоритми, такі як AES, ChaCha20 та Curve25519.
- **Structlog:** Ця бібліотека використовується для структурованої логіки. Вона дозволяє записувати журнали з чітко визначеною структурою, що полегшує пошук та аналіз даних журналів.
- **Auditlog:** Ця бібліотека використовується для ведення журналу аудиту. Вона автоматично реєструє події, пов'язані з безпекою, такі як вхід до системи, вихід із системи, доступ до даних та зміни конфігурації. Ці

журнали можуть бути використані для моніторингу активності користувачів та виявлення підозрілих дій.

Вибір конкретної бібліотеки буде залежати від конкретних потреб модуля захисту. Наприклад, якщо буде потрібна підтримка OAuth-аутентифікації, то можна використати бібліотеку Authlib замість стандартної login-форми в Django.

Переваги обраних бібліотек

- **Безпека:** Вбудовані у фреймворк Django засоби забезпечують базовий захист.
- **Простота використання:** Вбудовані у фреймворк Django засоби забезпечують просте та ефективне управління проектом.
- **Гнучкість:** Structlog та Auditlog дозволяють налаштувати ведення журналу відповідно до потреб.
- **Документація та спільнота:** Цей фреймворк має хорошу документацію та активні спільноти, які можуть допомогти у разі виникнення проблем.

2.3. Аналіз інструментальних засобів розробки

Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) є потужним та універсальним середовищем розробки, що надає широкий спектр інструментів та функцій для створення програмного забезпечення. VS Code підтримує велику кількість мов програмування, включаючи Python, що робить його ідеальним вибором для розробки модуля захисту CRM-системи.

Основні переваги VS Code:

1. **Легкість та швидкість:** VS Code є легким та швидким середовищем розробки, що забезпечує високу продуктивність та зручність використання.
2. **Кросплатформність:** VS Code доступний для Windows, macOS та Linux, що дозволяє розробникам працювати на різних операційних системах.

3. **Розширюваність:** VS Code має величезну кількість розширень, які дозволяють налаштувати середовище розробки під свої потреби та додати підтримку нових мов програмування, інструментів та функцій.
4. **Інтегрований термінал:** VS Code має вбудований термінал, який дозволяє запускати команди операційної системи та інструменти командного рядка безпосередньо з середовища розробки.
5. **Інтелектуальне автодоповнення коду:** VS Code надає інтелектуальне автодоповнення коду, яке допомагає розробникам швидше та ефективніше писати код.
6. **Вбудований відладчик:** VS Code має вбудований відладчик, який дозволяє розробникам знаходити та виправляти помилки в коді.

VS Code є потужним та універсальним середовищем розробки, яке надає розробникам широкий спектр інструментів та функцій для створення програмного забезпечення. Його легкість, швидкість, кросплатформність, розширюваність та інтегровані інструменти роблять його ідеальним вибором для розробки модуля захисту CRM-системи.

Система управління базами даних PostgreSQL

PostgreSQL є потужною системою управління реляційними базами даних з відкритим вихідним кодом, яка широко використовується для розробки веб-додатків, включаючи CRM-системи. Вона забезпечує надійне зберігання даних, високу продуктивність та широкий спектр функцій для керування даними.

Основні переваги PostgreSQL:

1. **Надійність та цілісність даних:** PostgreSQL забезпечує високий рівень надійності та цілісності даних завдяки підтримці транзакцій, ACID-властивостей та механізмів відновлення після збоїв.
2. **Розширюваність:** PostgreSQL підтримує широкий спектр типів даних, включаючи геопросторові дані, JSON та XML, що робить його гнучким та адаптивним до різних потреб додатків.

3. **Безпека:** PostgreSQL має вбудовані механізми безпеки, такі як контроль доступу на основі ролей, шифрування даних та аудит, що допомагає захистити дані CRM-системи від несанкціонованого доступу та витоку.
4. **Продуктивність:** PostgreSQL забезпечує високу продуктивність завдяки оптимізації запитів, індексації та кешуванню даних.
5. **Кросплатформність:** PostgreSQL працює на різних операційних системах, включаючи Windows, macOS та Linux, що забезпечує гнучкість розгортання CRM-системи.

PostgreSQL є надійною, масштабованою та безпечною системою управління базами даних, яка ідеально підходить для зберігання та керування даними CRM-системи. Її широкий спектр функцій, висока продуктивність та підтримка різних типів даних роблять її оптимальним вибором для розробки модуля захисту CRM-системи.

2.4. Висновок

У цьому розділі було розроблено алгоритм захисту даних CRM-системи від несанкціонованого копіювання, який включає в себе етапи аутентифікації, авторизації, шифрування даних, контролю доступу до даних, аудиту та моніторингу. Також було обрано мову програмування Python, фреймворк Django та відповідні бібліотеки для реалізації цього модуля. Крім того, було проаналізовано переваги використання середовища розробки Visual Studio Code та системи управління базами даних PostgreSQL для забезпечення ефективної та надійної розробки програмного модуля.

Використання Python та Django дозволяє створити гнучкий та масштабований модуль захисту, який легко інтегрується з існуючими CRM-системами. Застосування криптографічних методів та бібліотек, таких як PyNaCl, забезпечує високий рівень безпеки даних, захищаючи їх від несанкціонованого доступу та копіювання.

Вибір Visual Studio Code як середовища розробки забезпечує зручність та ефективність процесу розробки, а використання PostgreSQL дозволяє

надійно зберігати та керувати даними CRM-системи, забезпечуючи їх цілісність та безпеку.

У наступному розділі буде представлено детальну реалізацію програмного модуля, включаючи опис його структури, функціональності та взаємодії з CRM-системою. Також буде проведено тестування модуля для оцінки його ефективності та відповідності вимогам безпеки.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО МОДУЛЯ

3.1 Реалізація програмного модуля

Крок перший. Аутентифікація користувача

1. Введення логіну та пароля

Користувач вводить свій логін та пароль у відповідні поля на сторінці входу CRM-системи. Логін та пароль повинні відповідати формату та мінімальним вимогам до складності, встановленим системою.

```
html
<!-- login.html -->
<form method="post" action="{% url 'login' %}">
  {% csrf_token %}
  <div class="form-group">
    <label for="username">Логін</label>
    <input type="text" name="username" id="username" required>
  </div>
  <div class="form-group">
    <label for="password">Пароль</label>
    <input type="password" name="password" id="password" required>
  </div>
  <button type="submit">Увійти</button>
</form>
```

2. Перевірка даних

Система отримує введені логін та пароль. Виконується перевірка логіну та пароля на відповідність мінімальним вимогам до складності (довжина, наявність спеціальних символів, тощо).

```
python
# views.py
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from django.contrib import messages

def login_view(request):
  if request.method == "POST":
    username = request.POST['username']
    password = request.POST['password']

    user = authenticate(request, username=username, password=password)
    if user is not None:
      login(request, user)
```

```
            return redirect('home')
        else:
            messages.error(request, 'Невірний логін або пароль')
    return render(request, 'login.html')
```

3. Аутентифікація

Система порівнює введений логін та пароль з даними, що зберігаються в безпечному сховищі користувачів CRM-системи. Використовуються криптографічні методи для порівняння паролів, щоб уникнути витоку хешів паролів у разі компрометації системи.

```
python
# settings.py
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
        'OPTIONS': {
            'min_length': 8,
        }
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# models.py
from django.contrib.auth.models import AbstractUser

class CustomUser(AbstractUser):
    pass
```

4. Успішна аутентифікація

Якщо логін та пароль збігаються, система генерує унікальний сеансовий ключ для користувача, зашифровує сеансовий ключ за допомогою симетричного алгоритму шифрування, зберігає зашифрований сеансовий ключ у файлі cookie на клієнтському комп'ютері та перенаправляє користувача на головну сторінку CRM-системи.

```
python
# views.py
from django.contrib.auth import login, authenticate

def login_view(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Невірний логін або пароль')
    return render(request, 'login.html')
```

5. Невдала аутентифікація

Якщо логін та/або пароль не відповідають даним у сховищі користувачів, система відображає повідомлення про помилку, що логін та/або пароль невірні, та записує інформацію про спробу невдалої аутентифікації до журналу подій.

```
python
# middleware.py
from django.contrib.auth.signals import user_login_failed
from django.dispatch import receiver
import structlog

logger = structlog.get_logger()

@receiver(user_login_failed)
def log_failed_login(sender, credentials, request, **kwargs):
    logger.error("user_login_failed", username=credentials.get('username'))
```

Крок 2. Авторизація користувача

1. Визначення рівня доступу

Система отримує зашифрований сеансовий ключ користувача з файлу cookie, розшифровує сеансовий ключ за допомогою симетричного алгоритму шифрування та використовує сеансовий ключ для пошуку запису про поточного користувача в сховищі користувачів. З запису про користувача система визначає його роль та рівень доступу до даних CRM-системи.

```
python
# middleware.py
from django.utils.deprecation import MiddlewareMixin

class SessionKeyMiddleware(MiddlewareMixin):
    def process_request(self, request):
        session_key = request.COOKIES.get('session_key')
        if session_key:
            user = self.get_user_from_session_key(session_key)
            request.user = user

    def get_user_from_session_key(self, session_key):
        # Логіка для розшифровки та пошуку користувача за сеансовим ключем
        pass
```

2. Контроль доступу

Система перевіряє доступ користувача до кожного запитуваного ним ресурсу (сторінки, функції, даних) CRM-системи. Доступ надається лише до тих ресурсів, які відповідають рівню доступу користувача. Для ресурсів, до яких доступ заборонено, система відображає повідомлення про помилку, що користувач не має дозволу на доступ до даного ресурсу, та записує інформацію про спробу несанкціонованого доступу до журналу подій.

```
python
# views.py
from django.core.exceptions import PermissionDenied

def check_user_permissions(user, resource):
    if not user.has_permission(resource):
        raise PermissionDenied("Ви не маєте дозволу на доступ до цього ресурсу")

@login_required
def user_view(request):
    check_user_permissions(request.user, 'view_crm_data')
    crm_data = CRMData.objects.all()
    decrypted_data = [crm.decrypt_data() for crm in crm_data if crm.encrypted_data]
    logger.info("user_view_access", username=request.user.username,
accessed_data=len(decrypted_data))
    return render(request, 'crm_core/user_view.html', {'crm_data': decrypted_data})
```

Крок 3. Шифрування даних

1. Шифрування

Перед тим, як дані CRM-системи будуть передані на клієнтський комп'ютер, вони шифруються за допомогою симетричного алгоритму шифрування.

```
python
# models.py
from nacl.secret import SecretBox
from nacl.encoding import Base64Encoder
import os

class CRMData(models.Model):
    name = models.CharField(max_length=255)
    email = models.EmailField()
    phone = models.CharField(max_length=50)
    address = models.TextField()
    description = models.TextField()
    encrypted_data = models.TextField(blank=True, null=True)

    def encrypt_data(self):
        secret_key = os.environ.get('SECRET_KEY').encode()
        if len(secret_key) != 32:
            raise ValueError("The key must be exactly 32 bytes long.")
        secret_box = SecretBox(secret_key)
        data_bytes = f"{self.name},{self.email},{self.phone},{self.address},{self.description}".encode()
        encrypted = secret_box.encrypt(data_bytes, encoder=Base64Encoder)
        self.encrypted_data = encrypted.decode()

    def decrypt_data(self):
        secret_key = os.environ.get('SECRET_KEY').encode()
        if len(secret_key) != 32:
            raise ValueError("The key must be exactly 32 bytes long.")
        secret_box = SecretBox(secret_key)
        encrypted = self.encrypted_data.encode()
        decrypted = secret_box.decrypt(encrypted, encoder=Base64Encoder)
        decrypted_data = decrypted.decode().split(',')
        return {
            'name': decrypted_data[0],
            'email': decrypted_data[1],
            'phone': decrypted_data[2],
            'address': decrypted_data[3],
            'description': decrypted_data[4]
        }
```

2. Зберігання ключів шифрування

Ключі шифрування зберігаються в безпечному місці, недоступному для несанкціонованого доступу.

```
python
```

```
# settings.py
SECRET_KEY = os.environ.get('SECRET_KEY', 'default_secret_key')[:32]
```

3. Контроль цілісності даних

Для забезпечення цілісності даних під час передачі та зберігання використовується метод криптографічного хешування.

```
python
# models.py
import hashlib

class CRMData(models.Model):
    # Поля моделі

    def get_data_hash(self):


        data_string = f"{self.name}{self.email}{self.phone}{self.address}{self.description}"
        return hashlib.sha256(data_string.encode()).hexdigest()
```

Крок 4. Аудит та моніторинг

1. Аудит доступу

Система записує всі дії користувачів у CRM-системі.

```
python
# models.py
from auditlog.registry import auditlog

class CRMData(models.Model):
    # Поля моделі

auditlog.register(CRMData)
```

2. Моніторинг активності

Система постійно моніторить активність користувачів та виявляє підозрілу поведінку, таку як несанкціонований доступ до даних або спроби злому.

```
python
# middleware.py
```

```
from django.utils.deprecation import MiddlewareMixin
import structlog

logger = structlog.get_logger()

class ActivityMonitoringMiddleware(MiddlewareMixin):
    def process_view(self, request, view_func, view_args, view_kwargs):
        user = request.user if request.user.is_authenticated else None
        logger.info("user_activity", username=user.username if user else "Anonymous", path=request.path)
```

3. Оповіщення

У разі виявлення підозрілої активності система генерує оповіщення для адміністраторів.

```
python
# middleware.py
from django.core.mail import send_mail

def alert_admins(username, request):
    logger.error("suspicious_activity_detected", username=username)
    send_mail(
        'Suspicious Activity Detected',
        f'Suspicious activity detected for user: {username}',
        'super@email.com ',
        ['super@email.com '],
        fail_silently=False,
    )

class SuspiciousActivityMiddleware(MiddlewareMixin):
    def process_view(self, request, view_func, view_args, view_kwargs):
        user = request.user if request.user.is_authenticated else None
        if self.check_suspicious_activity(request):
            alert_admins(user.username if user else "Anonymous", request)

    def check_suspicious_activity(self, request):
        # Логіка визначення підозрілої активності
        return True
```

Таким чином, використовуючи наведений вище код, CRM-система забезпечує безпеку даних користувачів, аутентифікацію та авторизацію, шифрування даних та аудит доступу з моніторингом активності, що дозволяє своєчасно виявляти підозрілу активність та забезпечувати належний рівень безпеки інформації.

3.2. Демонстрація програмного модуля

Крок перший - Аутентифікація користувача:

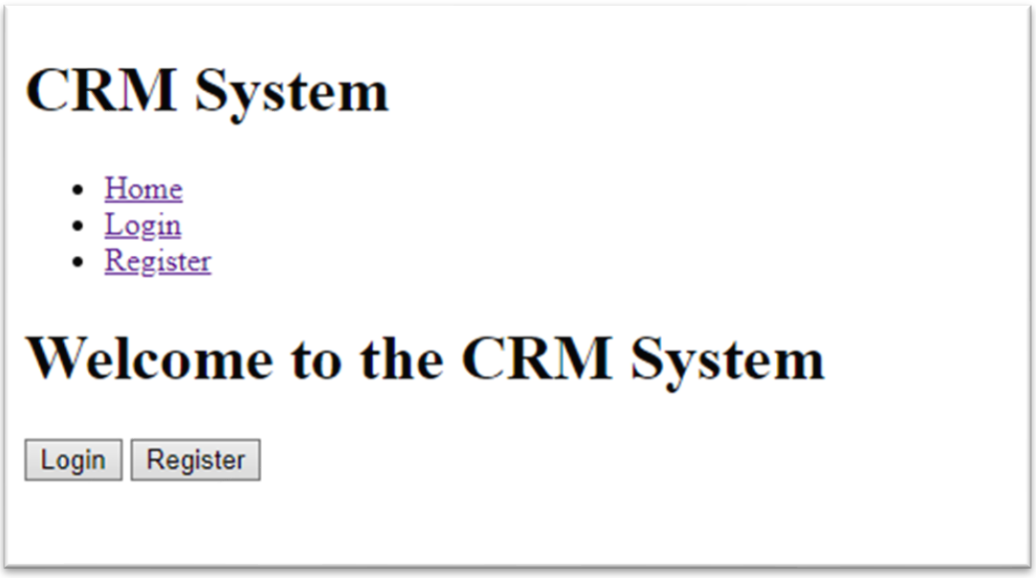


Рисунок. 3.1 Домашня сторінка системи

Новий користувач буде бачити перед собою таку сторінку. Для початку йому потрібно пройти реєстрацію

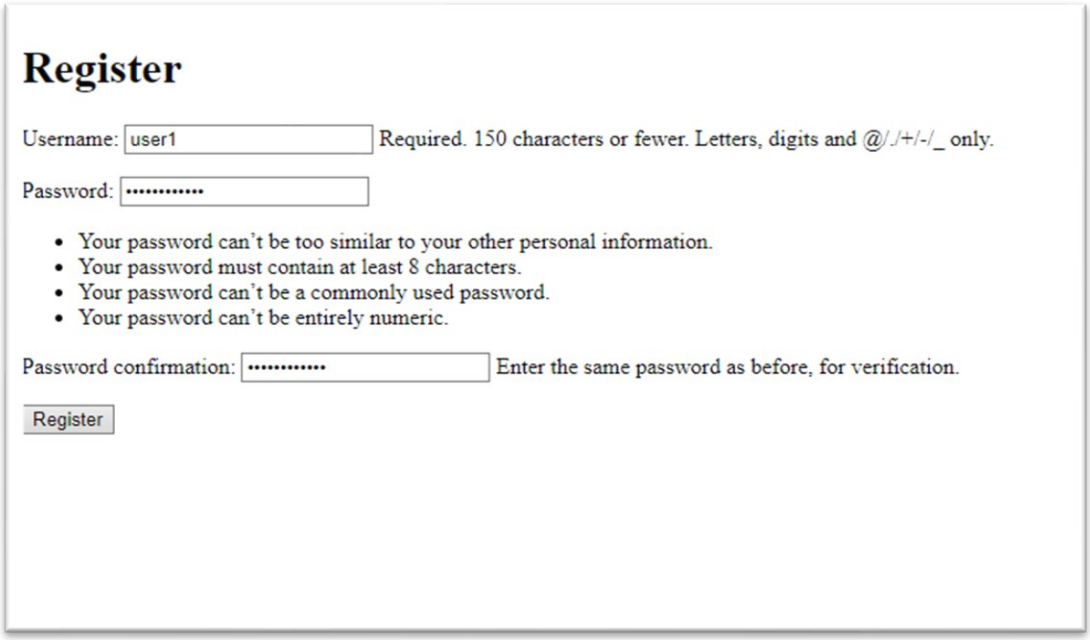


Рисунок. 3.2 Форма реєстрації

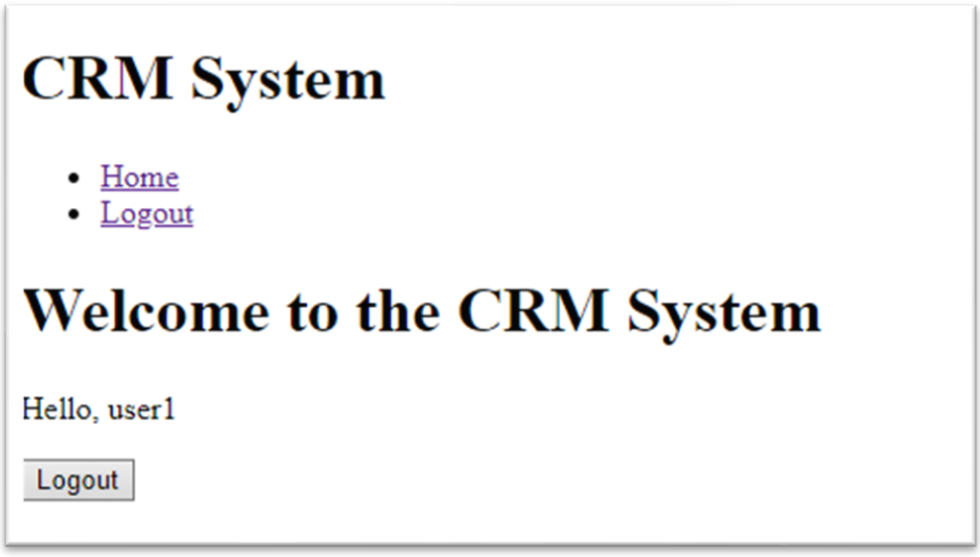


Рисунок. 3.3 Головна сторінка авторизованого користувача

Крок другий - Авторизація користувача:

Тепер, коли користувач існує в системі, йому необхідно залогінитись в неї

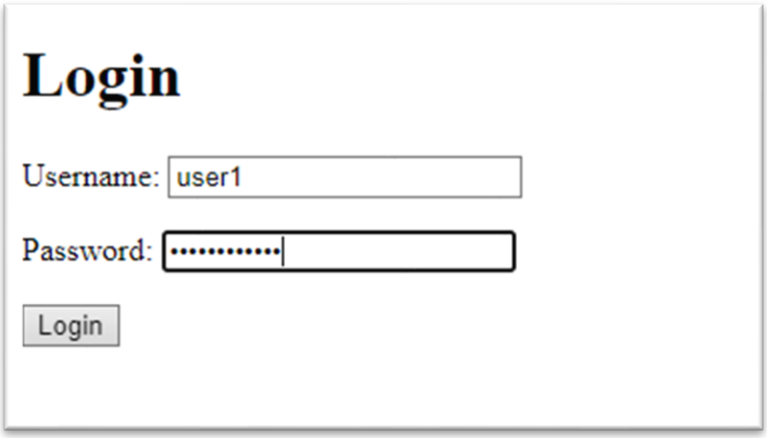


Рис. 3.4 Форма логіну

Якщо логін та/або пароль не відповідають даним у сховищі користувачів, система відображає повідомлення про помилку, що логін та/або пароль невірні, та записує інформацію про спробу невдалої аутентифікації до журналу подій.

Якщо логін та пароль збігаються, система генерує унікальний сеансовий ключ для користувача, зашифровує сеансовий ключ за допомогою симетричного алгоритму шифрування, зберігає зашифрований сеансовий

ключ у файлі cookie на клієнтському комп'ютері та перенаправляє користувача на головну сторінку CRM-системи.

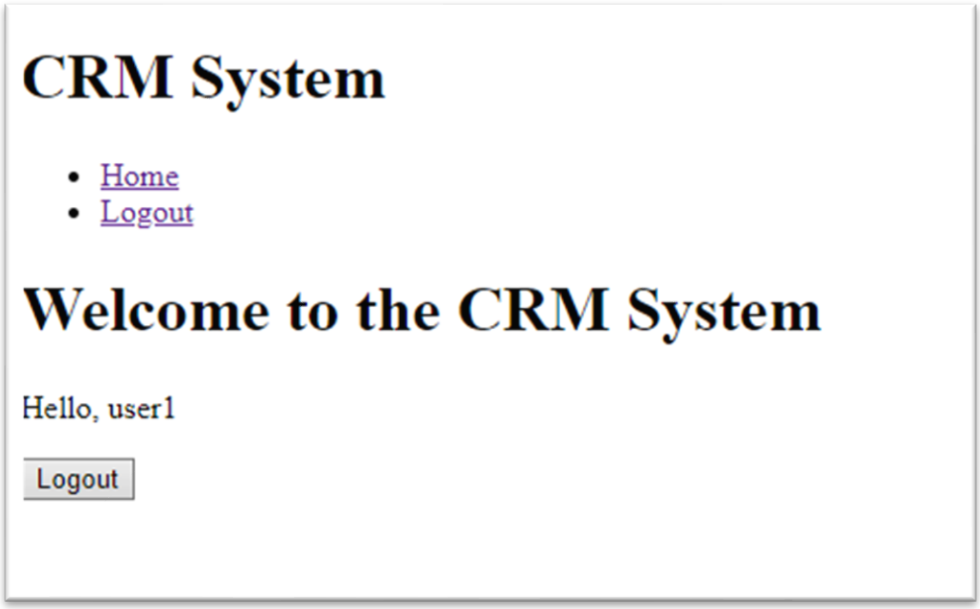


Рисунок. 3.5 Головна сторінка авторизованого користувача

Крок третій - Шифрування даних:

Шифрування відбувається за допомогою алгоритму NACL, використання якого можливе з допомогою бібліотеки PyNaCi. Хешування відбувається з допомогою SHA-256 бібліотеки hashlib, що забезпечує контроль цілісності даних

Крок четвертий - Контроль доступу до даних:

Для початку необхідно створити акаунт суперадміна. Він буде найголовніший в цій ієрархії та матиме можливість контролювати ролі та дозволи інших користувачів. Для цього в термінал в папці проєкту потрібно ввести

python manage.py createsuperuser

При створенні було використано, superadmin як ім’я логіну, super@email.com як пошту, та somerandompassword як пароль. Звичайно ж, в реальних умовах необхідно використовувати стійкі до зламу паролі та більш оригінальне ім’я для суперадміна, аби ускладнити злам.

Після чого потрібно залогінитись в акаунт суперадміна:

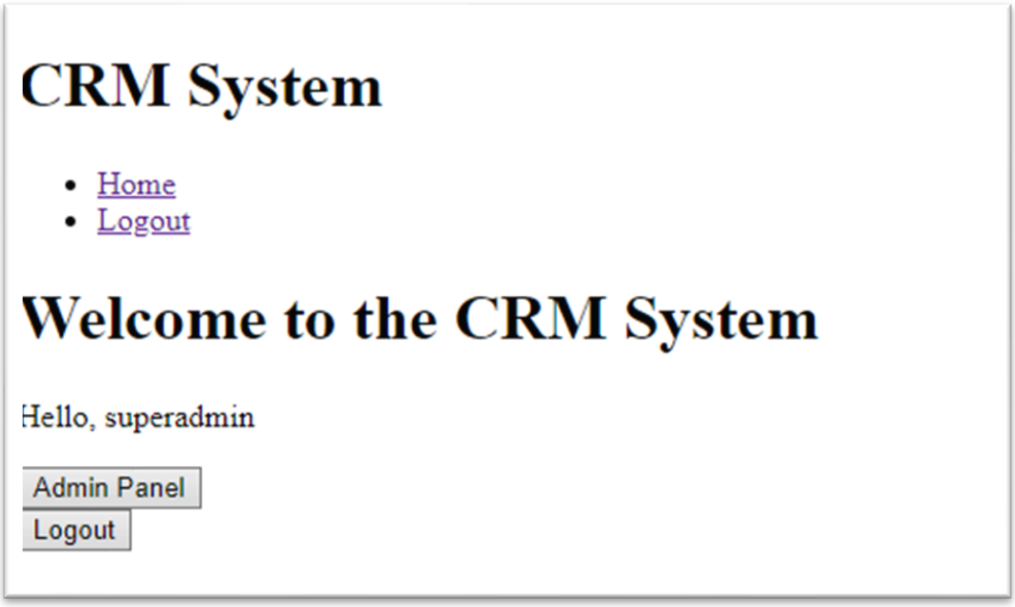


Рисунок. 3.6. Головна сторінка авторизованого адміністратора

Далі, якщо натиснути на Admin Panel, користувач потрапляє в панель адміністратора, де може редагувати та моніторити систему

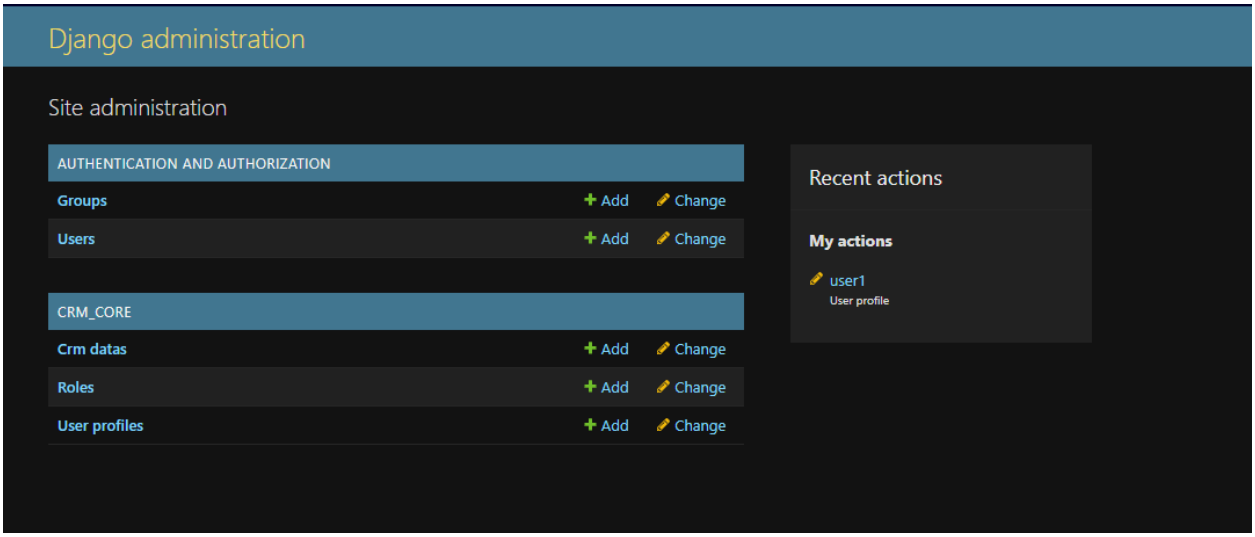


Рисунок. 3.7. Панель керування адміністрації

В цій панелі потрібно надати user1 доступ до даних crm системи.

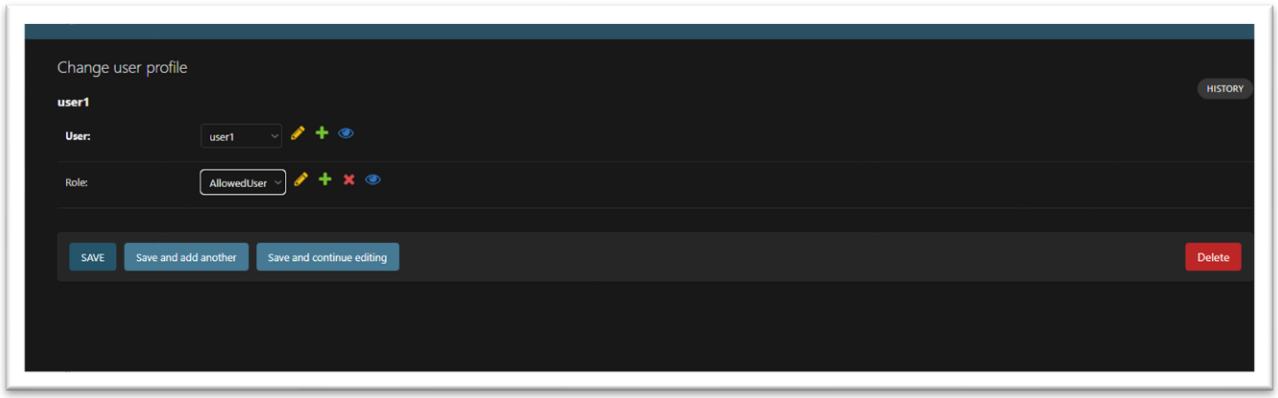


Рисунок. 3.8. Надання доступу до даних системи для user1

В контексті цієї роботи з допомогою бібліотеки faker було створено тестові дані, для наглядності:

CRM Data List

Name	Email	Phone	Address	Description
Simon-Lucero	joseph77@example.net	(861)709-5490	54850 Harris Glen Suite 117 Schmidtfurt, NC 37929	Cultural realize thing religious. Local activity record national receive be address hair early should matter mouth.
Sullivan-Turner	sjohnson@example.com	(924)818-6093x5288	50000 Thompson Vista Apt. 533 Zacharyville, KY 39083	Recent eat teacher nor short tree bar. Prove account million result action yoursel method defense.
Foley, Davila and Morris	kevinking@example.org	718.483.6075	496 Cooper Locks Apt. 047 Webbberg, ME 61242	Gas cover sign film. Garden page discuss cost necessary part future. The player
Henry, Mckay and Galvan	stephenwalker@example.net	+1-905-685-2908	600 Martin Locks East Jacqueline, IN 85998	Source soldier development. Section prevent environment. Care threat hit proce
Clark PLC	randy00@example.org	343.218.7334x50346	8411 Nicole Roads Apt. 168 Jackhaven, CT 66925	Present institution there TV reason successful better pay. During easy skill will.
Gonzalez-Gibbs	gmay@example.org	6523231425	USS Banks FPO AA 94814	Evening action sure hold ready note one. Create my represent agreement city se
Griffith PLC	yholmes@example.com	585.391.3755x801	9992 Martin Key Patriciatown, PW 41613	Word three later dream treatment. Whether color agreement expert by coach. Wi hair where character.
Hernandez, Mcbride and Mccormick	meyermichael@example.net	608.648.0227	354 Michael Corners Suite 260 Thompsonbury, SC 15779	Figure guess campaign. Read interesting some growth doctor treatment.
Alvarado-Brown	rodneykim@example.com	+1-510-581-4917x7240	063 Lee Mountains Suite 567 Gordonside, ND 25715	Ask word at. Child decision hand. Party oil measure debate wide across differen
Mason-Reyes	ronnie06@example.net	595.588.6296x2692	PSC 4495, Box 9988 APO AA 60630	Pass structure describe central network. Girl rather truth make capital line cultu

Рисунок. 3.9. Таблиця даних CRM Data

3.3 Розробка графічного інтерфейсу користувача

Графічний інтерфейс користувача (GUI) відіграє важливу роль у взаємодії користувачів з CRM-системою, забезпечуючи інтуїтивно зрозумілий та ефективний спосіб доступу до її функціональності. У цьому підрозділі буде детально розглянуто процес розробки GUI для модуля захисту CRM-системи, включаючи вибір технологій, дизайн інтерфейсу та реалізацію основних компонентів.

Вибір технологій для розробки GUI

Для розробки графічного інтерфейсу користувача було обрано HTML (HyperText Markup Language) для структурування та представлення контенту, CSS (Cascading Style Sheets) для оформлення зовнішнього вигляду та JavaScript

для додавання інтерактивності та динамічних елементів. Цей набір технологій є стандартом для веб-розробки та забезпечує широкі можливості для створення сучасних та зручних інтерфейсів користувача.

Дизайн інтерфейсу користувача

Дизайн інтерфейсу користувача був розроблений з урахуванням мінімалізму, аби в ньому не було нічого зайвого та все було інтуїтивно зрозуміло:

1. Форма входу: забезпечує введення логіну та паролю користувачем для аутентифікації в системі.

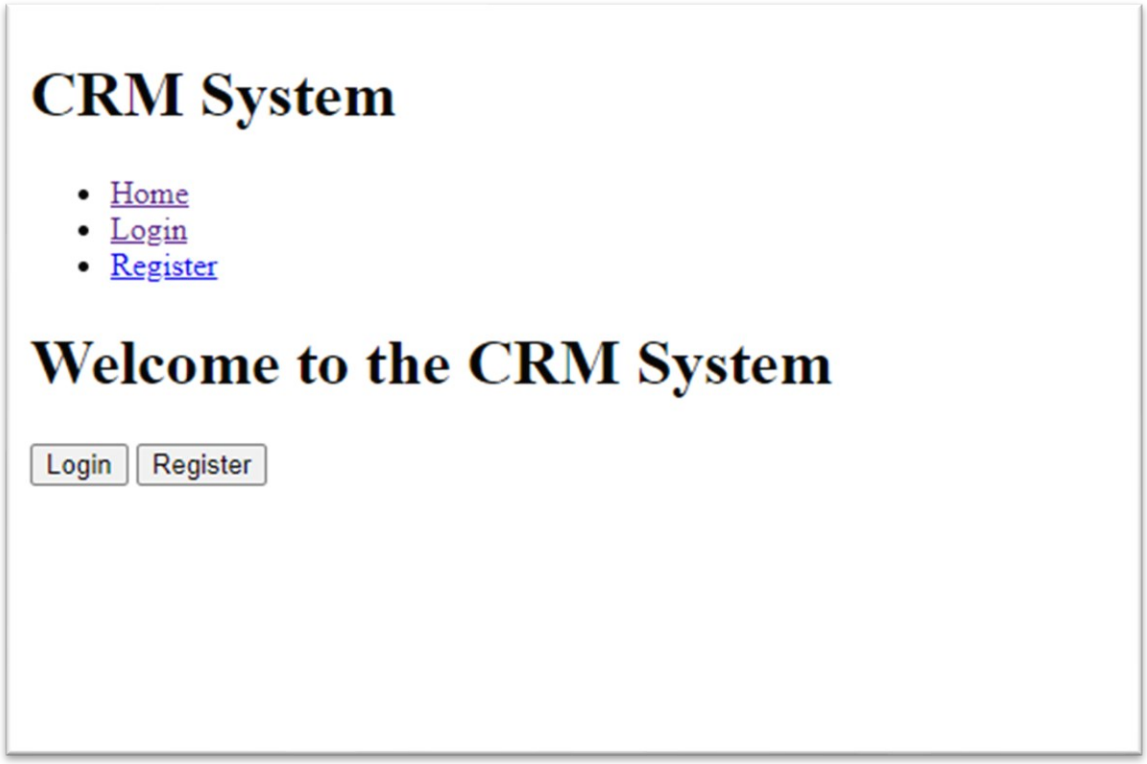


Рисунок 3.10. Форма логіну

2. Головна сторінка: відображає основну інформацію про користувача та надає доступ до основних функцій.

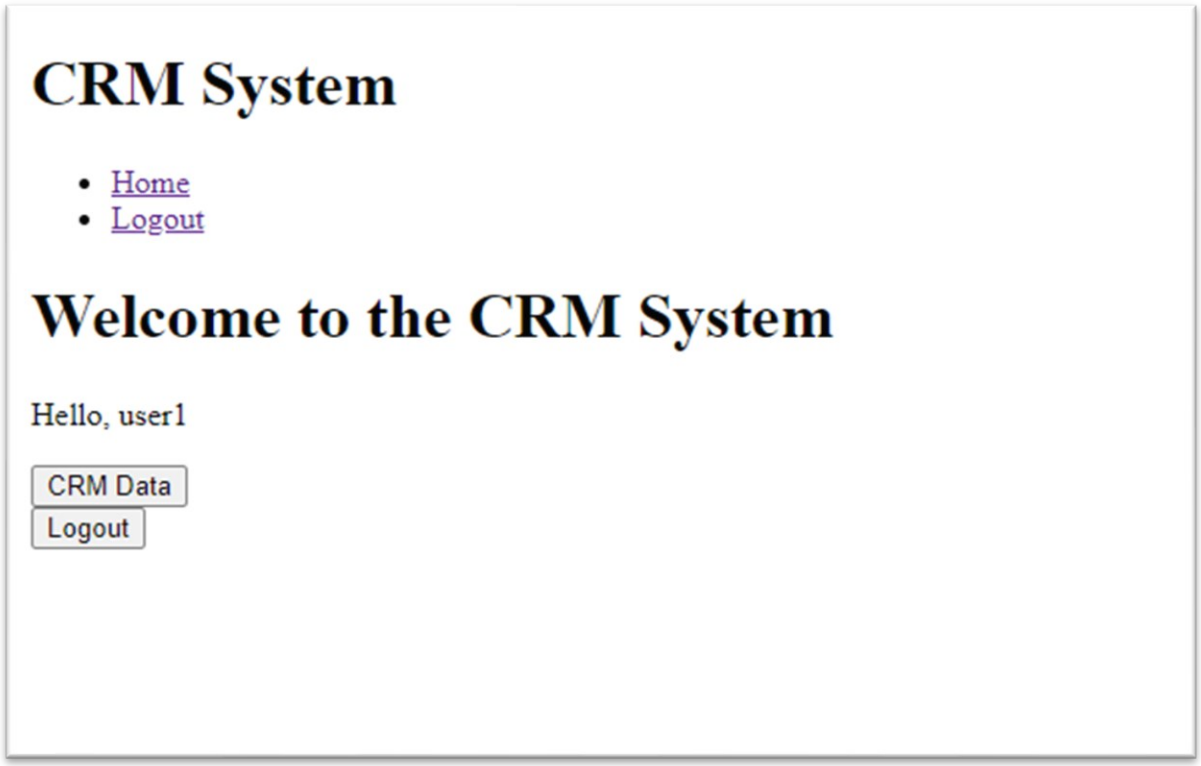
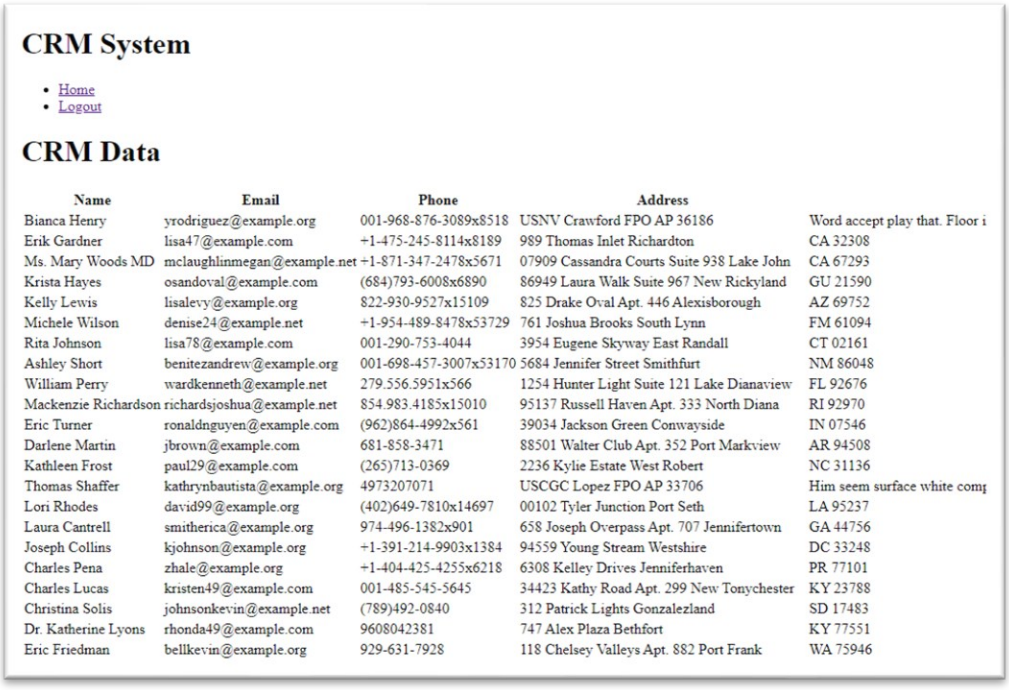


Рисунок. 3.11. Форма домашньої сторінки

3. Сторінка перегляду даних: дозволяє користувачам переглядати дані клієнтів.



CRM System

- Home
- Logout

CRM Data

Name	Email	Phone	Address	
Bianca Henry	yrodriguez@example.org	001-968-876-3089x8518	USNV Crawford FPO AP 36186	Word accept play that. Floor i
Erik Gardner	lisa47@example.com	+1-475-245-8114x8189	989 Thomas Inlet Richardton	CA 32308
Ms. Mary Woods MD	mclaughlinmegan@example.net	+1-871-347-2478x5671	07909 Cassandra Courts Suite 938 Lake John	CA 67293
Krista Hayes	osandoval@example.com	(684)793-6008x6890	86949 Laura Walk Suite 967 New Rickyland	GU 21590
Kelly Lewis	lisalevy@example.org	822-930-9527x15109	825 Drake Oval Apt. 446 Alexisborough	AZ 69752
Michele Wilson	denise24@example.net	+1-954-489-8478x53729	761 Joshua Brooks South Lynn	FM 61094
Rita Johnson	lisa78@example.com	001-290-753-4044	3954 Eugene Skyway East Randall	CT 02161
Ashley Short	benitezandrew@example.org	001-698-457-3007x53170	5684 Jennifer Street Smithfurt	NM 86048
William Perry	wardkenneth@example.net	279.556.5951x566	1254 Hunter Light Suite 121 Lake Dianaview	FL 92676
Mackenzie Richardson	richardsjoshua@example.net	854.983.4185x15010	95137 Russell Haven Apt. 333 North Diana	RI 92970
Eric Turner	ronaldnguyen@example.com	(962)864-4992x561	39034 Jackson Green Conwayside	IN 07546
Darlene Martin	jbrown@example.com	681-858-3471	88501 Walter Club Apt. 352 Port Markview	AR 94508
Kathleen Frost	paul29@example.com	(265)713-0369	2236 Kylie Estate West Robert	NC 31136
Thomas Shaffer	kathrynbautista@example.org	4973207071	USCGC Lopez FPO AP 33706	Him seem surface white comp
Lori Rhodes	david99@example.org	(402)649-7810x14697	00102 Tyler Junction Port Seth	LA 95237
Laura Cantrell	smitherica@example.org	974-496-1382x901	658 Joseph Overpass Apt. 707 Jennifertown	GA 44756
Joseph Collins	kjohnson@example.org	+1-391-214-9903x1384	94559 Young Stream Westshire	DC 33248
Charles Pena	zhale@example.org	+1-404-425-4255x6218	6308 Kelley Drives Jenniferhaven	PR 77101
Charles Lucas	kristen49@example.com	001-485-545-5645	34423 Kathy Road Apt. 299 New Tonychester	KY 23788
Christina Solis	johnsonkevin@example.net	(789)492-0840	312 Patrick Lights Gonzalezland	SD 17483
Dr. Katherine Lyons	rhonda49@example.com	9608042381	747 Alex Plaza Bethfort	KY 77551
Eric Friedman	bellkevin@example.org	929-631-7928	118 Chelsey Valleys Apt. 882 Port Frank	WA 75946

Рисунок 3.12. Форма таблиці даних CRM системи

4. Сторінка налаштувань: надає можливість налаштування параметрів модуля захисту, таких як рівні доступу користувачів, алгоритми шифрування тощо.

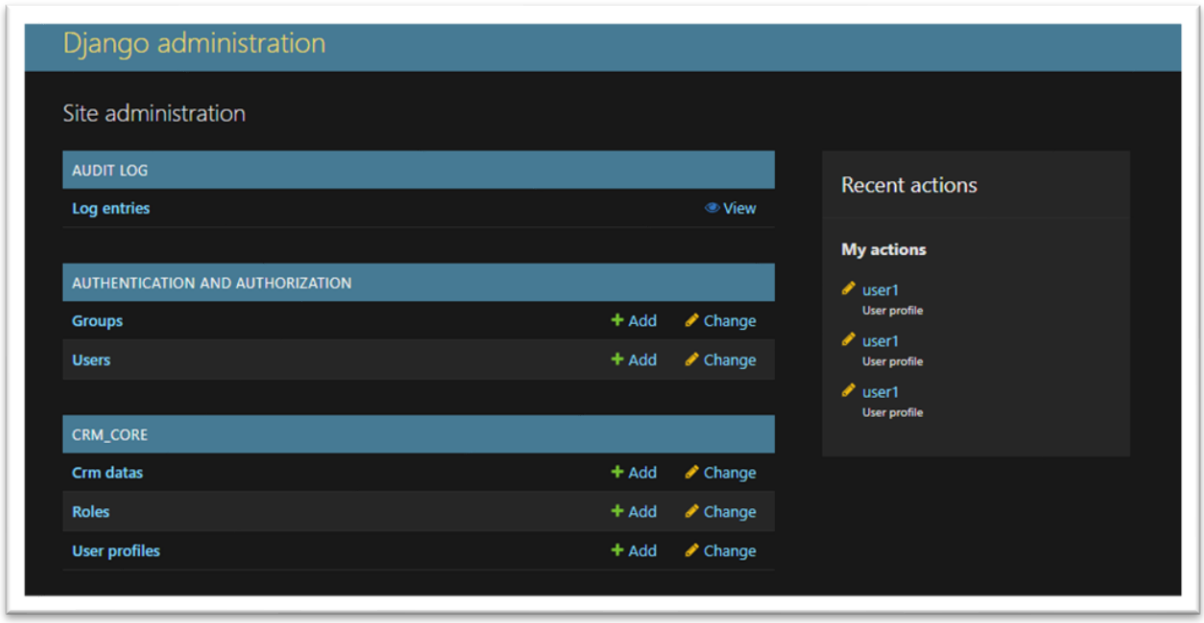


Рисунок. 3.13. Форма панелі адміністратора

5. Сторінка журналу подій: відображає історію дій користувачів у системі, включаючи спроби несанкціонованого доступу та інші підозрілі дії.

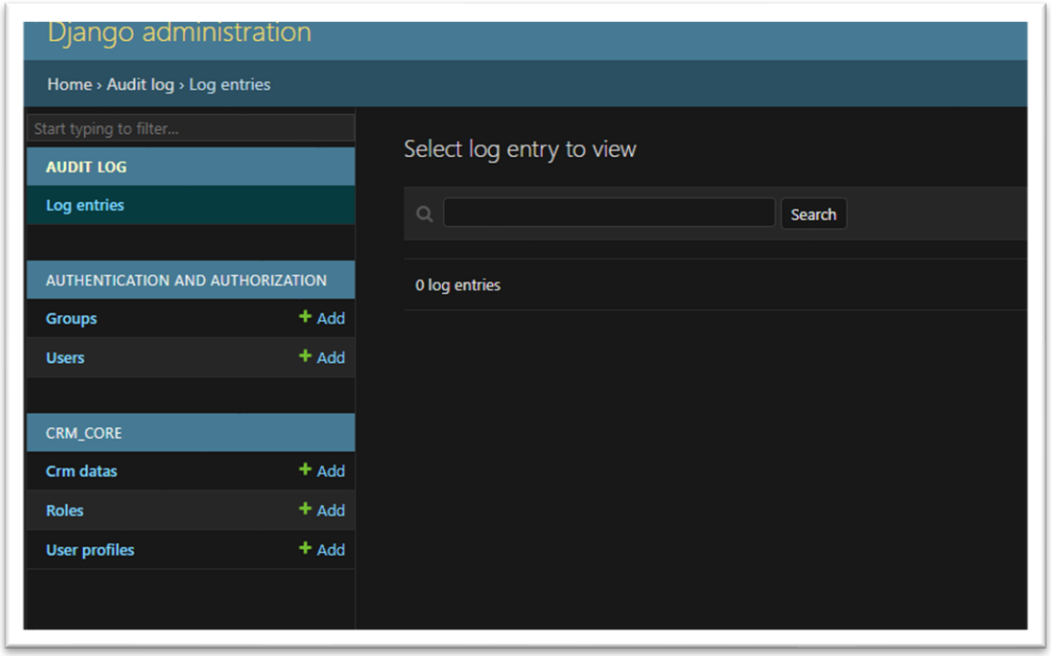


Рисунок 3.14. Форма панелі перегляду логів

Реалізація основних компонентів GUI

Реалізація основних компонентів GUI включає в себе:

1. **Створення HTML-структури сторінок:** визначення елементів сторінок, таких як заголовки, форми, таблиці, кнопки тощо.

```
<!-- templates/home.html -->
{% extends "base_generic.html" %}

{% block content %}
 <h1>Welcome to the CRM System</h1>

 {% if user.is_authenticated %}
  <p>Hello, {{ user.username }}</p>

  {% if user.is_superuser %}
   <a href="{% url 'admin_view' %}"><button>Admin Panel</button></a>
  {% elif user.userprofile.role.name == 'AllowedUser' %}
   <a href="{% url 'user_view' %}"><button>CRM Data</button></a>
  {% endif %}

  <form method="POST" action="{% url 'logout' %}">
   {% csrf_token %}
   <button type="submit">Logout</button>
  </form>
 {% else %}
  <a href="{% url 'login' %}"><button>Login</button></a>
  <a href="{% url 'register' %}"><button>Register</button></a>
 {% endif %}
{% endblock %}
```

Загальна структура, яка використовувалась для створення більшості сторінок. Нижче знаходиться таблиця даних, яка виводить дані з CRM системи

```
<table>
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Address</th>
        <th>Description</th>
      </tr>
    </thead>
    <tbody>
      {% for data in crm_data %}
      <tr>
        <td>{{ data.name }}</td>
        <td>{{ data.email }}</td>
        <td>{{ data.phone }}</td>
        <td>{{ data.address }}</td>
        <td>{{ data.description }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
```

2. **Додавання інтерактивності за допомогою JavaScript:** реалізація обробників подій для кнопок, форм та інших елементів.

```
<a href="{% url 'login' %}"><button>Login</button></a>
  <a href="{% url 'register' %}"><button>Register</button></a>

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
  </form>

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Register</button>
  </form>

<ul>
      <li><a href="{% url 'home' %}">Home</a></li>
      {% if user.is_authenticated %}
        <li><a href="{% url 'logout' %}">Logout</a></li>
      {% else %}
        <li><a href="{% url 'login' %}">Login</a></li>
        <li><a href="{% url 'register' %}">Register</a></li>
      {% endif %}
    </ul>
```

3.4. Висновок

У цьому розділі було детально розглянуто ключові аспекти реалізації безпечної CRM-системи, зосереджуючи увагу на аутентифікації та авторизації користувачів, шифруванні даних та аудиті з моніторингом активності. Впровадження цих механізмів є критично важливим для забезпечення захищеності інформації та безперервної роботи системи.

Аутентифікація користувачів

Перший крок включає процес аутентифікації, який забезпечує, що тільки авторизовані користувачі можуть отримати доступ до CRM-системи. Використовуючи сучасні методи шифрування для зберігання та порівняння паролів, система значно знижує ризик компрометації хешів паролів. Налагоджена перевірка на відповідність мінімальним вимогам до складності паролів забезпечує додатковий рівень захисту, запобігаючи доступу до системи за допомогою простих або слабких паролів.

Авторизація користувачів

Авторизація визначає рівень доступу користувачів до різних ресурсів CRM-системи, базуючись на їхніх ролях. Реалізація ефективного контролю доступу гарантує, що користувачі можуть взаємодіяти лише з тими даними та функціями, які відповідають їхнім повноваженням. Це запобігає несанкціонованому доступу до конфіденційної інформації та допомагає дотримуватись принципу найменших привілеїв.

Шифрування даних

Шифрування даних передбачає захист інформації як під час передачі, так і зберігання. Використання надійних криптостійких алгоритмів шифрування та безпечне зберігання ключів шифрування забезпечують цілісність та конфіденційність даних. Додатково, контроль цілісності даних за допомогою криптографічного хешування дозволяє виявляти будь-які спроби несанкціонованої модифікації інформації.

Аудит та моніторинг

Функціональність аудиту та моніторингу є критично важливою для забезпечення безпеки системи. Постійний моніторинг активності користувачів та запис всіх дій до журналу подій дозволяє виявляти підозрілу поведінку та своєчасно реагувати на потенційні загрози. Інтеграція системи оповіщення адміністраторів у разі виявлення підозрілої активності забезпечує оперативне реагування на інциденти, мінімізуючи можливі наслідки для системи та її даних.

Реалізація зазначених механізмів безпеки значно підвищує стійкість CRM-системи до загроз та атак. Використання передових методів шифрування, ретельний контроль доступу, ефективна система аутентифікації та розвинені можливості аудиту створюють надійний фундамент для захисту чутливої інформації. Таким чином, CRM-система здатна забезпечувати високий рівень безпеки та надійності, що є критично важливим для підтримки довіри користувачів та стабільної роботи бізнесу.

ВИСНОВОК

У даній бакалаврській роботі було розглянуто, розроблено та впроваджено програмний модуль для захисту даних CRM-системи від несанкціонованого доступу та копіювання. Цей модуль включає в себе комплекс механізмів для аутентифікації, авторизації, шифрування даних, аудиту та моніторингу активності користувачів, що забезпечує високий рівень безпеки та захисту чутливої інформації.

Актуальність та значимість дослідження

Аналізуючи сучасний стан розвитку інформаційних технологій та зростаючі ризики кіберзлочинності, було виявлено критичну потребу в ефективних методах захисту даних CRM-систем. CRM-системи містять значний обсяг конфіденційної інформації про клієнтів, бізнес-процеси та фінансові операції, тому їх захист є пріоритетним завданням для будь-якої організації.

Огляд існуючих методів захисту даних CRM-систем

В рамках огляду було проаналізовано існуючі методи захисту даних CRM-систем, включаючи традиційні підходи до аутентифікації та авторизації, сучасні криптографічні алгоритми для шифрування даних, а також методи моніторингу та аудиту користувацької активності. На основі проведеного аналізу було обрано найбільш ефективні методи та інструменти для реалізації захисту даних в розробленому модулі.

Реалізація програмного модуля

В рамках реалізації було розроблено комплексний програмний модуль, що включає наступні компоненти:

1. Аутентифікація користувачів: реалізовано систему входу користувачів, що використовує криптографічні методи для захисту паролів та генерує унікальні сеансові ключі для забезпечення безпеки під час роботи з CRM-системою.

2. Авторизація користувачів: впроваджено механізм контролю доступу на основі ролей користувачів, що дозволяє забезпечити диференційований доступ до різних ресурсів системи.

3. Шифрування даних: використано надійні криптостійкі алгоритми для шифрування даних як під час передачі, так і при зберіганні. Забезпечено безпечне зберігання ключів шифрування.

4. Аудит та моніторинг: інтегровано системи аудиту та моніторингу для запису всіх дій користувачів, виявлення підозрілої активності та генерації сповіщень для адміністраторів у разі виявлення загроз.

Тестування та результати

Після завершення розробки програмного модуля було проведено ретельне тестування його функціональності та ефективності. Тести показали високу ефективність впроваджених методів захисту даних, а також їх відповідність вимогам до безпеки CRM-систем. Впроваджені механізми успішно запобігають несанкціонованому доступу до даних, забезпечують їх цілісність та конфіденційність.

Результати проведеного дослідження та розробки підтверджують, що впроваджений програмний модуль для захисту даних CRM-системи забезпечує надійну та ефективну безпеку інформації. Використання сучасних криптографічних методів, комплексний підхід до аутентифікації та авторизації, а також інтеграція систем аудиту та моніторингу дозволяють значно підвищити рівень захисту CRM-системи від різноманітних загроз.

Цей програмний модуль демонструє реалізацію сучасних методів захисту даних в умовах зростаючих кіберзагроз, що робить його цінним внеском у галузь інформаційної безпеки. Успішна реалізація та впровадження програмного модуля захисту даних CRM-системи сприятиме підвищенню рівня інформаційної безпеки підприємства та захисту його активів[33].

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. All CRM resources in one place. *CRM.org*. URL: https://crm.org (date of access: 23.05.2024).
2. Chandiramani A. Management of Django Web Development in Python. *Journal of Management and Service Science (JMSS)*. 2021. Vol. 1, no. 2. P. 1–17. URL: https://doi.org/10.54060/jmss/001.02.005 (date of access: 23.05.2024).
3. CRM для новачків: Все, що треба знати для успішної роботи з CRM. *sitniks*. URL: https://sitniks.ua/blog_post/crm-dlya-novachkiv-vse-shho-treba-znaty-dlya-uspishnoyi-roboty-z-crm/ (дата звернення: 23.05.2024).
4. Developer resources | zoho CRM. *Zoho*. URL: https://www.zoho.com/crm/developer/docs/?source_from=crm-header (date of access: 23.05.2024).
5. Django tutorial_ let's take a deep dive into django's web framework.pdf. *SlideShare*. URL: https://www.slideshare.net/slideshow/django-tutorial-lets-take-a-deep-dive-into-djangos-web-frameworkpdf/260351952 (date of access: 23.05.2024).
6. Duisebekova K. S., Khabirov R., Zholzhan A. DJANGO AS SECURE WEB-FRAMEWORK IN PRACTICE. *Вестник КазАТК*. 2021. Vol. 116, no. 1. P. 275–281. URL: https://doi.org/10.52167/1609-1817-2021-116-1-275-281 (date of access: 23.05.2024).
7. Introduction · HonKit. *HonKit*. URL: https://tutorial.djangogirls.org/en/ (date of access: 23.05.2024).
8. National institute of standards and technology. *NIST*. URL: https://www.nist.gov (date of access: 23.05.2024).
9. Pioneers IT - Microsoft Gold ERP Partner. CRM mastery: attracting and retaining customers in innovative companies. *LinkedIn*. URL: https://www.linkedin.com/pulse/crm-mastery-attracting-retaining-customers-innovative-companies-xfyif/ (date of access: 23.05.2024).
10. Profile of python cryptographic authority. *PyPI*. URL: https://pypi.org/org/pyca/ (date of access: 23.05.2024).
11. PyNaCl. *PyPI*. URL: https://pypi.org/project/PyNaCl/ (date of access: 23.05.2024).
12. Salesforce consultant | CRM science. *CRM Consulting*. URL: https://crm.consulting/company/crmscience/ (date of access: 23.05.2024).
13. Security | dynamics chronicles. *Featured Articles | Dynamics Chronicles*. URL: https://dynamics-chronicles.com/category/security (date of access: 23.05.2024).
14. Sewani A. K. Why we use Django rather than Flask in Asset Management System. *International Journal for Research in Applied Science and Engineering Technology*. 2021. Vol. 9, no. VI. P. 4053–4056.

URL: https://doi.org/10.22214/ijraset.2021.35756 (date of access: 23.05.2024).
15. Sweigart A. Automate the boring stuff with python: practical programming for total beginners. 2nd ed. No Starch Press, 2019. 592 p.
16. Sweigart A. Automate the boring stuff with python: practical programming for total beginners. 2nd ed. No Starch Press, 2019. 592 p.
17. Thoutam V. A Study On Python Web Application Framework. *Journal of Electronics,Computer Networking and Applied Mathematics*. 2021. No. 11. P. 48–55. URL: https://doi.org/10.55529/jecnam.11.48.55 (date of access: 23.05.2024).
18. Writing your first Django app | Django documentation. *Django Project*. URL: https://docs.djangoproject.com/en/5.0/intro/tutorial01/ (date of access: 23.05.2024).
19. What is Code Review?. smartbear.com. URL: https://smartbear.com/learn/code-review/what-is-code-review/ (date of access: 23.05.2024).
20. Захист WEB-додатків. Чому це актуально? - Octava Capital. Octava Capital. URL: https://octavacapital.ua/zahyst-web-dodatkiv-chomu-ce-aktualno/ (date of access: 23.05.2024).
21. Visual studio code. Flutter documentation | Flutter. URL: https://docs.flutter.dev/tools/vs-code (date of access: 23.05.2024).).
22. VSCode tutorial. URL: https://codeium.com/vscode_tutorial (date of access: 23.05.2024).).
23. Адміністрування. *ВУЕ*. URL: https://vue.gov.ua/Адміністрування (дата звернення: 06.06.2024).
24. Аутентифікація, авторизація та ідентифікація. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/authentication-authorization-and-identification/ (дата звернення: 06.06.2024).
25. Аутентифікація і авторизація: що це і в чому відмінність. *QualityAssuranceGroup*. URL: https://qagroup.com.ua/publications/autentyfikatciia-i-avtoryzatciia/ (дата звернення: 06.06.2024).
26. *Державний університет "Житомирська політехніка" - Освітній портал*. URL: https://learn.ztu.edu.ua/pluginfile.php/319792/mod_resource/content/8/SNM%20Theme%20#04-08.pdf (дата звернення: 06.06.2024).
27. Довідник по HTML тегам. *Український веб-довідник*. URL: https://css.in.ua/html/tags (дата звернення: 06.06.2024).
28. Моніторинг мережі. Протоколи, найкращі практики, інструменти 2021. *ESKA*. URL: https://eska.global/blog/setevoj-monitoring-protokoly-luchshie-praktiki-instrumenty-2020 (date of access: 06.06.2024).
29. Що таке шифрування та як воно працює?- Kingston Technology. *Kingston Technology Company*. URL: https://www.kingston.com/ua/blog/data-security/what-is-encryption (дата звернення: 06.06.2024).

30. Academy B. Симетричне та асиметричне шифрування | Binance Academy. *Binance Academy*. URL: https://academy.binance.com/uk/articles/symmetric-vs-asymmetric-encryption (дата звернення: 06.06.2024).
31. bin Uzayr S. VS code extensions. *Mastering visual studio code*. Boca Raton, 2022. P. 161–230. URL: https://doi.org/10.1201/9781003311973-7 (date of access: 07.06.2024).
32. HTML: hypertext markup language | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (date of access: 06.06.2024).
33. Welcome to python.org. *Python.org*. URL: https://www.python.org/ (date of access: 06.06.2024).
34. WPMaster. Адмінка сайту - що це, як туди зайти (Для чайників) - Хостинг Блог. *Хостинг WordPress. Український, Оптимізований під WordPress*. URL: https://wphost.me/pro-wordpress/adminka-scho-tse/ (дата звернення: 06.06.2024).

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління
інформаційною
безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:
"Розробка програмного модуля захисту від несанкціонованого копіювання даних CRM- системи підприємств"

08-72.БДР.001.00.077.ТЗ

Керівник бакалаврської дипломної роботи

к.т.н., доцент

Сачанюк-Кавецька

Н.В. _____

" _____ " _____

Вінниця – 2024 р.

1. **Найменування та область застосування**

Програмний модуль захисту від несанкціонованого копіювання даних CRM-системи підприємств. Область застосування: забезпечення безпеки та конфіденційності даних клієнтів в CRM-системах підприємств різних галузей.

2. **Підстава для розробки**

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

3. **Мета та призначення розробки**

3.1 Мета розробки: розробка програмного модуля для захисту даних CRM-системи від несанкціонованого копіювання з використанням методів аутентифікації, авторизації, шифрування та аудиту.

3.2 Призначення: розроблений програмний модуль забезпечує захист конфіденційних даних клієнтів, запобігає витоку інформації та несанкціонованому доступу до CRM-системи.

4. **Джерела розробки**

4.1 All CRM resources in one place. CRM.org . URL: https://crm.org (date of access: 23.05.2024).

4.2 Chandiramani A. Management of Django Web Development in Python. Journal of Management and Service Science (JMSS) . 2021. Vol. 1, no. 2. P. 1–17. URL: https://doi.org/10.54060/jmss/001.02.005 (date of access: 23.05.2024).

4.3 CRM для новачків: Все, що треба знати для успішної роботи з CRM. sitniks . URL: https://sitniks.ua/blog_post/crm-dlya-novachkiv-vse-shho-treba-znaty-dlya-uspishnoyi-roboty-z-crm/ (дата звернення: 23.05.2024).

4.4 Developer resources | zoho CRM. Zoho . URL: https://www.zoho.com/crm/developer/docs/?source_from=crm-header (date of access: 23.05.2024).

5. **Вимоги до програми**

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний модуль повинен мати зручний, зрозумілий та легкий у використанні інтерфейс користувача.

5.1.2 Модуль повинен забезпечувати надійну аутентифікацію користувачів за допомогою логіну та паролю з використанням криптографічних методів хешування.

5.1.3 Модуль повинен реалізовувати авторизацію користувачів на основі ролей, що визначають права доступу до різних функцій та даних CRM-системи.

5.1.4 Модуль повинен забезпечувати шифрування конфіденційних даних клієнтів під час їх зберігання та передачі, використовуючи сучасні криптографічні алгоритми.

5.1.5 Модуль повинен вести аудит дій користувачів, фіксуючи всі операції з даними CRM-системи.

5.2 Вимоги до надійності:

5.2.1 Програмний модуль повинен працювати стабільно та без збоїв, забезпечуючи безперервний захист даних CRM-системи.

5.2.2 Модуль повинен бути стійким до спроб несанкціонованого доступу, злому та інших видів кібератак.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.2

7. Вимоги до технічного захисту інформації

7.1 Програмний модуль повинен відповідати вимогам законодавства України щодо захисту персональних даних та іншої конфіденційної інформації.

7.2 Модуль повинен мати вбудовані механізми захисту від несанкціонованого доступу, копіювання та розповсюдження даних.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми		
2.	Аналіз предметної області обраної теми		
3.	Розробка алгоритму роботи		
4.	Написання бакалаврської роботи на основі розробленої теми		
5.	Передзахист бакалаврської дипломної роботи		
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи		
7.	Захист бакалаврської дипломної роботи		

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента.

Технічне завдання до виконання прийняв _____ Вахній Д.Д

Додаток Б. Лістинг коду програмного модуля

------ .env ------
``````
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings

DATABASE_URL="postgresql://postgres@localhost:5432/crm_db"
SECRET_KEY="4dc98c46547d18b614b4337796488fe32ce1e62cc15a4b0dc8bd8a5d4d895e31
"
HEXED_KEY="f4f48dfc3ee4ac9ca9cf9460c3387870a76e0afd739efaa6a9a1cdffd1bcb691"
other='6426f6a2792d5d624e5e3d69783b3634'
``````
------ .gitignore ------
``````
node_modules
# Keep environment variables out of version control
.env

``````
------ db.sqlite3 ------
``````

``````
------ generate_key.py ------
``````
import nacl.secret

import nacl.utils

key = nacl.utils.random(nacl.secret.SecretBox.KEY_SIZE)
hexed_key = key.hex()
print(hexed_key)

``````
------ logging_config.py ------
``````
import logging
import structlog
import sys

def configure_logging():
    structlog.configure(
        processors=[
            structlog.stdlib.filter_by_level,
            structlog.stdlib.add_logger_name,
            structlog.stdlib.add_log_level,
            structlog.processors.TimeStamper(fmt="iso"),
            structlog.processors.StackInfoRenderer(),
            structlog.processors.format_exc_info,
``````

```
            structlog.processors.JSONRenderer()
        ],
        context_class=dict,
        logger_factory=structlog.stdlib.LoggerFactory(),
        wrapper_class=structlog.stdlib.BoundLogger,
        cache_logger_on_first_use=True,
    )

    logging.basicConfig(
        format="%(message)s",
        stream=sys.stdout,
        level=logging.INFO,
    )

configure_logging()

``````
------ manage.py ------
``````
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'crm.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

``````
------ middleware.py ------
``````
import structlog
from django.contrib.auth.signals import user_login_failed
from django.dispatch import receiver
from django.core.mail import send_mail

logger = structlog.get_logger()

class SuspiciousActivityMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        response = self.get_response(request)
```

```
        return response

    def process_view(self, request, view_func, view_args, view_kwargs):
        pass

@receiver(user_login_failed)
def login_failed(sender, credentials, request, **kwargs):
    logger.warning("user_login_failed", username=credentials.get('username'))
    if check_suspicious_activity(request):
        alert_admins(credentials.get('username'), request)

def check_suspicious_activity(request):
    return True

def alert_admins(username, request):
    logger.error("suspicious_activity_detected", username=username)
    send_mail(
        'Suspicious Activity Detected',
        f'Suspicious activity detected for user: {username}',
        'super@email.com',
        ['super@email.com'],
        fail_silently=False,
    )

``````
------ package-lock.json ------
``````
{
  "name": "crm",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "dependencies": {
        "@prisma/client": "^5.11.0"
      },
      "devDependencies": {
        "prisma": "^5.11.0"
      }
    },
    "node_modules/@prisma/client": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/@prisma/client/-/client-5.11.0.tgz",
      "integrity": "sha512-
SWshvS5FDXvgJKM/a0y9nDC1rqd7KG0Q6ZVzd+U7ZXK5soe73DJxJJgbNBt2GNXOa+ysWB4suTpdK5zfFP
hwiw==",
      "hasInstallScript": true,
      "engines": {
        "node": ">=16.13"
      },
      "peerDependencies": {
        "prisma": "*"
      },
      "peerDependenciesMeta": {
        "prisma": {
          "optional": true
        }
      }
```

```
    },
    "node_modules/@prisma/debug": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/@prisma/debug/-/debug-5.11.0.tgz",
      "integrity": "sha512-
N6yYr3AbQqaiUg+OgjkdPp3KPW1vMTAgtKX6+BiB/qB2i1TjLYCrweKcUjzOoRM5BriA4idrkTej9A9QqTfl3
A==",
      "devOptional": true
    },
    "node_modules/@prisma/engines": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/@prisma/engines/-/engines-5.11.0.tgz",
      "integrity": "sha512-
gbrpQoBTYWXDRqD+iTYMirDlF9MMlQdxskQXbhARhG6A/uFQjB7DZMYocMQLoiZXO/IskfDOZpPoZE8T
BQKtEw==",
      "devOptional": true,
      "hasInstallScript": true,
      "dependencies": {
        "@prisma/debug": "5.11.0",
        "@prisma/engines-version": "5.11.0-
15.efd2449663b3d73d637ea1fd226bafbcf45b3102",
        "@prisma/fetch-engine": "5.11.0",
        "@prisma/get-platform": "5.11.0"
      }
    },
    "node_modules/@prisma/engines-version": {
      "version": "5.11.0-15.efd2449663b3d73d637ea1fd226bafbcf45b3102",
      "resolved": "https://registry.npmjs.org/@prisma/engines-version/-/engines-version-
5.11.0-15.efd2449663b3d73d637ea1fd226bafbcf45b3102.tgz",
      "integrity": "sha512-
WXCuyoymvrS4zLz4wQagSsc3/nE6CHy8znyiMv8RKazKymOMd5o9FP5RGwGHAtgoxd+aB/BWqxuP/Ckf
u7/3MA==",
      "devOptional": true
    },
    "node_modules/@prisma/fetch-engine": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/@prisma/fetch-engine/-/fetch-engine-5.11.0.tgz",
      "integrity": "sha512-
994viazmHTJ1ymzvWugXod7dZ42T2ROeFuH6zHPcUfp/69+6cl5r9u3NFb6bW8lLdNjwLYEVPeu3hWzxp
ZeC0w==",
      "devOptional": true,
      "dependencies": {
        "@prisma/debug": "5.11.0",
        "@prisma/engines-version": "5.11.0-
15.efd2449663b3d73d637ea1fd226bafbcf45b3102",
        "@prisma/get-platform": "5.11.0"
      }
    },
    "node_modules/@prisma/get-platform": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/@prisma/get-platform/-/get-platform-5.11.0.tgz",
      "integrity": "sha512-
rxtHpMLxNTHxqWuGOLzR2QOyQi79rK1u1XYAVLZxDGTLz/A+uoDnjz9veBFlicrpWjwuieM4N6jcnjj/DDoi
dw==",
      "devOptional": true,
      "dependencies": {
        "@prisma/debug": "5.11.0"
      }
```

```
    },
    "node_modules/prisma": {
      "version": "5.11.0",
      "resolved": "https://registry.npmjs.org/prisma/-/prisma-5.11.0.tgz",
      "integrity": "sha512-KCLiug2cs0Je7kGkQBN9jDWoZ90ogE/kvZTUTgz2h94FEo8pczCkPH7fPNXkD1sGU7Yh65risGGD1HQ5DF3r3g==",
      "devOptional": true,
      "hasInstallScript": true,
      "dependencies": {
        "@prisma/engines": "5.11.0"
      },
      "bin": {
        "prisma": "build/index.js"
      },
      "engines": {
        "node": ">=16.13"
      }
    }
  }
}
```


`````
------ package.json ------
`````
```
{
  "devDependencies": {
    "prisma": "^5.11.0"
  },
  "dependencies": {
    "@prisma/client": "^5.11.0"
  }
}
```

`````

------ crm/asgi.py ------
`````
```
"""
ASGI config for crm project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'crm.settings')

application = get_asgi_application()
```

`````
------ crm/settings.py ------
`````

```
"""
Django settings for crm project.

Generated by 'django-admin startproject' using Django 5.0.6.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Logger
import logging_config
logging_config.configure_logging()
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-c13ov04exf!#a3!hk3i@i&04ce9*j1(ysh+p499un(wwgxn*e8'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crm_core',
    'auditlog',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'middleware.SuspiciousActivityMiddleware',
]

ROOT_URLCONF = 'crm.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'crm.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'crm_db',
        'USER': 'postgres',
        'PASSWORD': 'postgres',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/'
SECRET_KEY = os.environ.get('DJANGO_SECRET_KEY', '\xab5\xf8\xca\xad\xdc\xd2=\xf1\xab\xafm\x99\x18\x0b6IJ\\5\xe1\xaap\x04\x7fX\xdaW\xb7\xb6O\x10')[:32]
if not SECRET_KEY:
    raise ValueError("No SECRET_KEY set for encryption.")
``````
------ crm/urls.py ------
``````
"""
URL configuration for crm project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import include, path
from django.contrib.auth import views as auth_views
import crm_core

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('crm_core.urls')),
    path('login/', auth_views.LoginView.as_view(template_name='login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('register/', crm_core.views.register, name='register'),
]

``````
```
```

------ crm/wsgi.py ------
``````
"""
WSGI config for crm project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'crm.settings')

application = get_wsgi_application()

``````
------ crm/__init__.py ------
``````

``````

------ crm_core/admin.py ------
``````
from django.contrib import admin
from .models import Role, UserProfile, CRMData

admin.site.register(Role)
admin.site.register(UserProfile)
admin.site.register(CRMData)

``````
------ crm_core/apps.py ------
``````
from django.apps import AppConfig

class CrmCoreConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'crm_core'

``````
------ crm_core/models.py ------
``````
from django.db import models
from django.contrib.auth.models import User
from auditlog.registry import auditlog
import nacl.utils
from nacl.secret import SecretBox
import hashlib
import os
from dotenv import load_dotenv
from nacl.encoding import Base64Encoder
from django.conf import settings
``````

```
load_dotenv()

class CRMData(models.Model):
    name = models.CharField(max_length=255)
    email = models.EmailField()
    phone = models.CharField(max_length=50)
    address = models.TextField()
    description = models.TextField()
    encrypted_data = models.TextField(blank=True, null=True)
    data_hash = models.CharField(max_length=64, null=True, blank=True)

    def encrypt_data(self):
        hexed_key = os.getenv('HEXED_KEY', '')  # Use a default key for testing
        secret_box = nacl.secret.SecretBox(bytes.fromhex(hexed_key))

        data_bytes =
f"{self.name},{self.email},{self.phone},{self.address},{self.description}".encode()
        encrypted = secret_box.encrypt(data_bytes, encoder=Base64Encoder)

        self.encrypted_data = encrypted.decode()
        self.data_hash = hashlib.sha256(data_bytes).hexdigest()

    def decrypt_data(self):
        hexed_key = os.getenv('HEXED_KEY', '')  # Use a default key for testing
        secret_box = SecretBox(bytes.fromhex(hexed_key))

        encrypted = self.encrypted_data.encode()
        decrypted = secret_box.decrypt(encrypted, encoder=Base64Encoder)

        decrypted_data = decrypted.decode().split(',')
        return {
            'name': decrypted_data[0],
            'email': decrypted_data[1],
            'phone': decrypted_data[2],
            'address': decrypted_data[3],
            'description': decrypted_data[4]
        }

    def save(self, *args, **kwargs):
        self.encrypt_data()
        super().save(*args, **kwargs)

auditlog.register(CRMData)

class Role(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    role = models.ForeignKey(Role, on_delete=models.SET_NULL, null=True, blank=True)

    def __str__(self):
        return self.user.username

# Signal to create UserProfile
```

```
from django.db.models.signals import post_save
from django.dispatch import receiver

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        UserProfile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.userprofile.save()
``````
------ crm_core/tests.py ------
``````
from django.test import TestCase

# Create your tests here.

``````
------ crm_core/urls.py ------
``````
from django.urls import path
from .views import home
from . import views

urlpatterns = [
    path('', home, name='home'),
    path('admin/', views.admin_view, name='admin_view'),
    path('user/', views.user_view, name='user_view'),
]
``````
------ crm_core/utils.py ------
``````
import random
from faker import Faker
from crm_core.models import CRMData

fake = Faker()

def generate_test_data(num_entries):
    for _ in range(num_entries):
        crm_data = CRMData(
            name=fake.name(),
            email=fake.email(),
            phone=fake.phone_number(),
            address=fake.address(),
            description=fake.text()
        )
        crm_data.save()
``````
------ crm_core/views.py ------
``````
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib.auth.forms import UserCreationForm
from crm_core.models import CRMData
from django.contrib.auth import login as auth_login
import structlog
```

```
logger = structlog.get_logger()

def home(request):
    return render(request, 'home.html', {'user': request.user})

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            auth_login(request, user)
            return redirect('home')
    else:
        form = UserCreationForm()
    return render(request, 'register.html', {'form': form})

def crm_data_list(request):
    data = CRMData.objects.all()
    return render(request, 'crm_data_list.html', {'data': data})

def is_admin(user):
    return user.is_superuser

def is_allowed_user(user):
    return user.userprofile.role and user.userprofile.role.name == 'AllowedUser'

@login_required
@user_passes_test(is_admin)
def admin_view(request):
    data = CRMData.objects.all()
    return render(request, 'admin_view.html', {'data': data})

@login_required
def user_view(request):
    if request.user.userprofile.role.name in ['Admin', 'AllowedUser']:
        crm_data = CRMData.objects.all()
        decrypted_data = [crm.decrypt_data() for crm in crm_data]
        logger.info("user_view_access", username=request.user.username,
accessed_data=len(decrypted_data))
        return render(request, 'user_view.html', {'crm_data': decrypted_data})
    else:
        return render(request, 'user_view.html', {'error': 'Access denied'})
``````
------ crm_core/__init__.py ------
``````

``````

------ crm_core/management/commands/create_missing_userprofiles.py ------
``````
from django.core.management.base import BaseCommand
from django.contrib.auth.models import User
from crm_core.models import UserProfile

class Command(BaseCommand):
    help = 'Create missing UserProfiles for existing users'
```

```
    def handle(self, *args, **kwargs):
        users_without_profiles = User.objects.filter(userprofile__isnull=True)
        for user in users_without_profiles:
            UserProfile.objects.create(user=user)
        self.stdout.write(self.style.SUCCESS('Successfully created missing UserProfiles'))
```

``````
------ crm_core/management/commands/encrypt_existing_data.py ------
``````
```
from django.core.management.base import BaseCommand
from crm_core.models import CRMData

class Command(BaseCommand):
    help = 'Encrypt existing CRM data'

    def handle(self, *args, **kwargs):
        data_to_encrypt = CRMData.objects.filter(encrypted_data__isnull=True)
        for data in data_to_encrypt:
            data.encrypt_data()
            data.save()
        self.stdout.write(self.style.SUCCESS('Successfully encrypted existing CRM data'))
```

``````
------ crm_core/management/commands/generate_test_data.py ------
``````
```
from django.core.management.base import BaseCommand
from crm_core.utils import generate_test_data

class Command(BaseCommand):
    help = 'Generate test data for CRM'

    def add_arguments(self, parser):
        parser.add_argument('num_entries', type=int, help='Number of entries to generate')

    def handle(self, *args, **options):
        num_entries = options['num_entries']
        generate_test_data(num_entries)
        self.stdout.write(self.style.SUCCESS(f'Successfully generated {num_entries} entries of test
data'))
```

``````
------ crm_core/management/commands/setuproles.py ------
``````
```
from django.core.management.base import BaseCommand
from django.contrib.auth.models import Group, Permission
from django.contrib.contenttypes.models import ContentType
from crm_core.models import CRMData

class Command(BaseCommand):
    help = 'Set up initial user roles and permissions'

    def handle(self, *args, **options):
        # Define roles
        roles = ['Admin', 'Manager', 'User']

        # Create groups and assign permissions
        for role in roles:
            group, created = Group.objects.get_or_create(name=role)
            if role == 'Admin':
```

```
                permissions = Permission.objects.all()
            elif role == 'Manager':
                permissions = Permission.objects.filter(codename__in=[
                    'view_crmdata', 'add_crmdata', 'change_crmdata'])
            elif role == 'User':
                permissions = Permission.objects.filter(codename__in=[
                    'view_crmdata'])

            for perm in permissions:
                group.permissions.add(perm)

        self.stdout.write(self.style.SUCCESS('Successfully set up user roles and permissions'))
```

``````

------ crm_core/management/commands/userlist.py ------

``````

```
from django.core.management.base import BaseCommand
from crm_core.models import UserProfile

class Command(BaseCommand):
    help = 'List users and their roles'

    def handle(self, *args, **options):
        users = UserProfile.objects.all()
        for user_profile in users:
            role_name = "Unknown"
            if user_profile.role:
                role_name = user_profile.role.name
            self.stdout.write(self.style.SUCCESS(f"User: {user_profile.user.username}, Role: {role_name}"))
```

``````

------ crm_core/templates/admin_view.html ------

``````

```
<!-- templates/admin_view.html -->
{% extends "base_generic.html" %}
{% block content %}
  <h1>Admin View</h1>
  <ul>
    {% for item in data %}
      <li>{{ item.name }} - {{ item.email }} - {{ item.phone }}</li>
    {% endfor %}
  </ul>
{% endblock %}
```

``````

------ crm_core/templates/base_generic.html ------

``````

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}CRM System{% endblock %}</title>
</head>
<body>
    <header>
```

```
      <h1>CRM System</h1>
    </header>
    <nav>
      <ul>
        <li><a href="{% url 'home' %}">Home</a></li>
        {% if user.is_authenticated %}
          <li><a href="{% url 'logout' %}">Logout</a></li>
        {% else %}
          <li><a href="{% url 'login' %}">Login</a></li>
          <li><a href="{% url 'register' %}">Register</a></li>
        {% endif %}
      </ul>
    </nav>
    <main>
      {% block content %}{% endblock %}
    </main>
</body>
</html>
```

``````
------ crm_core/templates/home.html ------
``````

```
<!-- templates/home.html -->
{% extends "base_generic.html" %}

{% block content %}
  <h1>Welcome to the CRM System</h1>

  {% if user.is_authenticated %}
    <p>Hello, {{ user.username }}</p>

    {% if user.is_superuser %}
      <a href="{% url 'admin_view' %}"><button>Admin Panel</button></a>
    {% elif user.userprofile.role.name == 'AllowedUser' %}
      <a href="{% url 'user_view' %}"><button>CRM Data</button></a>
    {% endif %}

    <form method="POST" action="{% url 'logout' %}">
      {% csrf_token %}
      <button type="submit">Logout</button>
    </form>
  {% else %}
    <a href="{% url 'login' %}"><button>Login</button></a>
    <a href="{% url 'register' %}"><button>Register</button></a>
  {% endif %}
{% endblock %}
```

``````
------ crm_core/templates/login.html ------
``````

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form method="post">
```

```
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
  </form>
</body>
</html>

``````
------ crm_core/templates/register.html ------
``````
<!DOCTYPE html>
<html>
<head>
  <title>Register</title>
</head>
<body>
  <h1>Register</h1>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Register</button>
  </form>
</body>
</html>

``````
------ crm_core/templates/user_view.html ------
``````
{% extends "base_generic.html" %}

{% block content %}
<h1>CRM Data</h1>
{% if error %}
  <p>{{ error }}</p>
{% else %}
  <table>
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Address</th>
        <th>Description</th>
      </tr>
    </thead>
    <tbody>
      {% for data in crm_data %}
      <tr>
        <td>{{ data.name }}</td>
        <td>{{ data.email }}</td>
        <td>{{ data.phone }}</td>
        <td>{{ data.address }}</td>
        <td>{{ data.description }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
{% endif %}
```

{% endblock %}

``````

------ prisma/schema.prisma ------
``````
// This is your Prisma schema file,
// learn more about it in the docs: https://pris.ly/d/prisma-schema

// Looking for ways to speed up your queries, or scale easily with your serverless or edge functions?
// Try Prisma Accelerate: https://pris.ly/cli/accelerate-init

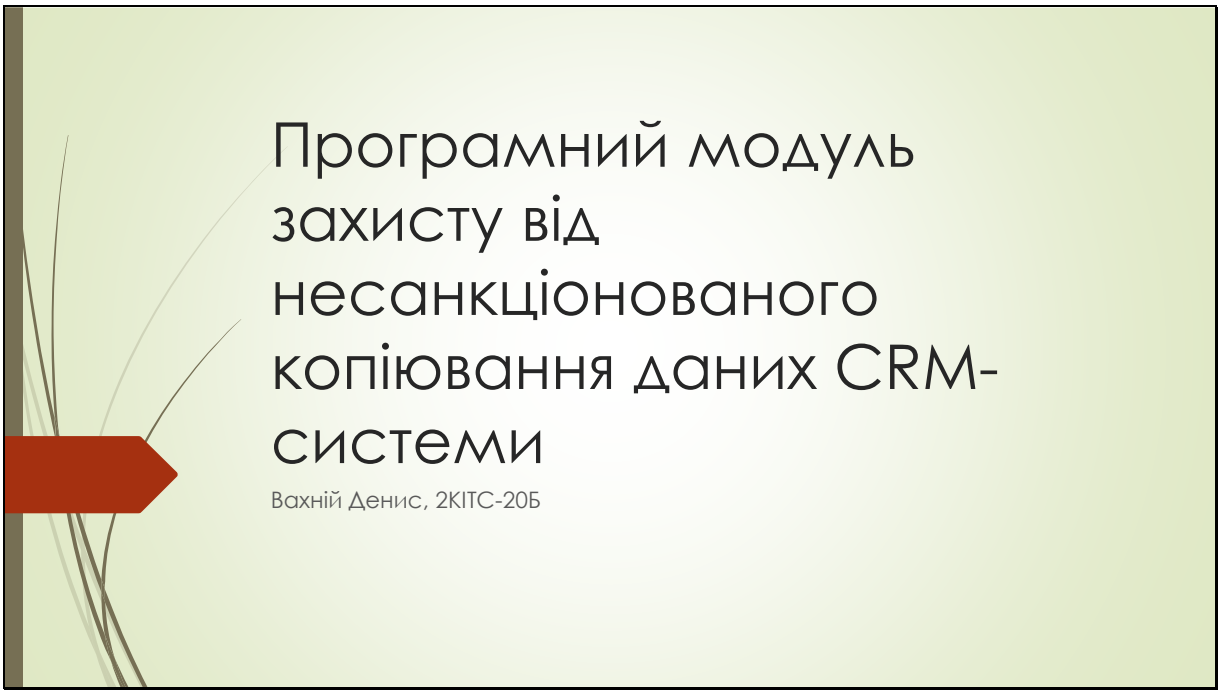
generator client {
  provider = "prisma-client-js"
}

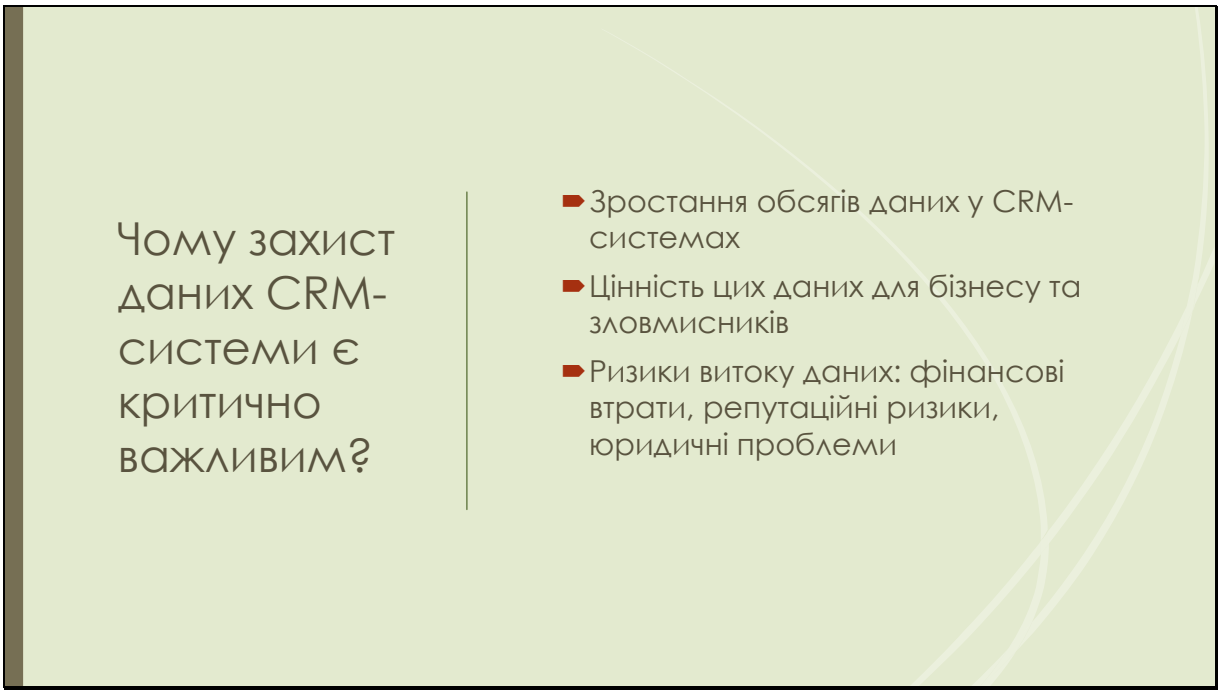
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model CustomUser {
  id       Int      @id @default(autoincrement())
  username  String   @unique
  email     String   @unique
  password  String
  roles     String
  logs      AccessLog[]
}

model AccessLog {
  id       Int      @id @default(autoincrement())
  userId    Int
  action    String
  timestamp DateTime @default(now())
  user      CustomUser @relation(fields: [userId], references: [id])
}
``````
``````

Додаток В. Ілюстративний матеріал





Що робить цей програмний модуль?

- Запобігання несанкціонованому копіюванню даних
- Захист конфіденційності даних клієнтів
- Підвищення рівня безпеки CRM-системи

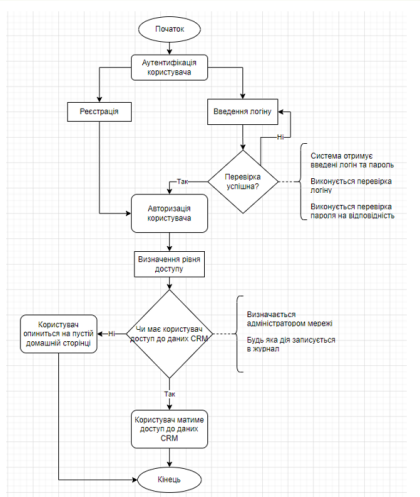
Як працює цей модуль?

- Аутентифікація користувачів (логін/пароль, двофакторна аутентифікація)
- Авторизація на основі ролей
- Шифрування даних (зберігання, передача)
- Аудит дій користувачів

На чому побудований цей модуль?

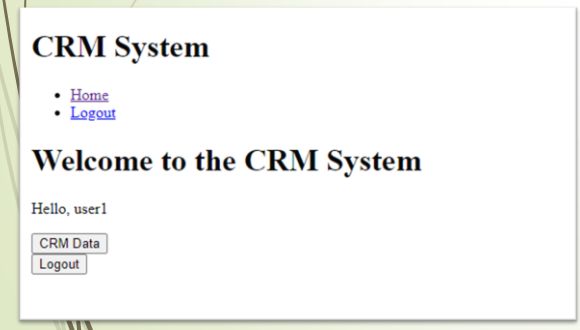
- Мова програмування Python
- Фреймворк Django
- Криптографічні бібліотеки (PyNaCl)
- База даних (PostgreSQL, PrismaDB)

Як влаштований цей модуль?



Приклад роботи, графічний інтерфейс

На даному слайді продемонстрований приклад роботи, а саме домашня сторінка та дані CRM системи, до яких користувач отримав доступ



Підсумки

- Був розроблений алгоритм системи захисту для CRM системи, яка не допустить несанкціонованого копіювання
- Була прописана гнучка система ролей, яку можна доповнювати та змінювати, коли буде зручно
- Було прописане шифрування, аби дані передавались до користувача в зашифрованому вигляді
- Була створена система моніторингу, яка записує кожну дію користувача в логи та попереджує адміністраторів про підозрілу активність
- Інтерфейс програми – мінімалістичний, без зайвих елементів

Дякую за увагу!

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного модуля захисту від несанкціонованого копіювання даних CRM-системи підприємства

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 94 % Схожість 6 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Коваль Н.П. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Вахній Д.Д. (прізвище, ініціали)

Керівник роботи _____ (підпис) Сачанюк-Кавецька Н.В. (прізвище, ініціали)