Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
освітній ступень бакалавр
(код і назва спеціальності)
освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Механізм захисту реєстрів медичних інформаційних систем»

Виконавець: студентка IV курсу, групи КБ-43

_____ **Юлія ДЕНИСЮК**
(підпис) (ім'я прізвище)

	Підпис	**Ім'я ПРІЗВИЩЕ**
Керівник		Сергій ДАКОВ
Нормоконтроль		Олександр ТОРОШАНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентці **КБ-43** **Денисюк Юлії Василівни**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Механізм захисту реєстрів медичних інформаційних систем

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Реєстри медичних даних, медичні інформаційні системи

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з циклом розробки програмного забезпечення, вразливостями, обрати метод шифрування даних та розробити програмне забезпечення для захисту реєстрів МІС.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблено програмне забезпечення, яке реалізує

механізм захисту авторських прав в програмних продуктах

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 28 листопада 2024 року

Завдання видав _____ (підпис) Сергій ДАКОВ (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Юлія ДЕНИСЮК (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 05.12.2024	*виконано*
2	Аналіз літератури	06.12.2024 – 15.12.2024	*виконано*
3	Обґрунтування вибору рішення	16.12.2024 – 20.12.2024	*виконано*
4	Розробка методів захисту реєстрів медичних даних	21.12.2024 – 31.12.2024	*виконано*
5	Проектування програмного забезпечення для захисту реєстрів	01.01.2025 – 15.01.2025	*виконано*
6	Реалізація та програмування механізмів захисту	16.01.2025 – 28.02.2025	*виконано*
7	Тестування програмного забезпечення та виправлення помилок	01.03.2025 – 30.04.2025	*виконано*
8	Оформлення пояснювальної записки	01.05.2025 – 31.05.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	01.06.2025 – 13.06.2025	*виконано*

Завдання видав _____ (підпис) Сергій ДАКОВ (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Юлія ДЕНИСЮК (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

УДК 004.056.5:614.2

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 51 сторінок, включає зміст, вступ, три розділи кваліфікаційної роботи, висновки та список літератури. Крім того, робота містить 6 додатків У пояснювальній записці міститься 29 ілюстрацій та 1 таблиці. Використано 20 джерел літератури.

Метою роботи є удосконалення механізмів захисту електронних реєстрів у медичних інформаційних системах шляхом розробки рекомендацій, спрямованих на підвищення рівня безпеки з урахуванням забезпечення конфіденційності, цілісності та доступності медичних даних.

Для досягнення поставленої мети виконано такі завдання:

1. Проведено аналіз існуючих методів захисту медичних інформаційних систем, зокрема реєстрів медичних даних.

2. Визначено основні кіберзагрози та внутрішні ризики, що можуть впливати на безпеку медичних реєстрів.

3. Розроблено програмне забезпечення для забезпечення захисту медичних реєстрів, яке включає шифрування даних, контроль доступу та систему моніторингу.

Об’єктом дослідження є процес захисту електронних реєстрів у медичних інформаційних системах.

Предметом дослідження є технічні та організаційні механізми забезпечення інформаційної безпеки в медичних системах.

Актуальність дослідження обумовлена зростанням кібератак на медичні системи, які оперують чутливою інформацією про здоров’я пацієнтів. Надійний захист цих даних є критично важливим для запобігання витоку,

несанкціонованого доступу або втрати медичної інформації, що має як правові, так і етичні наслідки.

Практична цінність полягає в можливості застосування розроблених рекомендацій для формування нових політик інформаційної безпеки в медичних закладах та вдосконалення існуючих заходів захисту.

Ключові слова: медичні інформаційні системи, захист даних, шифрування, автентифікація, багатофакторна автентифікація, контроль доступу, журнали аудиту, резервне копіювання, кіберзагрози.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

EMR	–	Electronic Medical Records
EHR	–	Electronic Health Records
RPM	–	Remote Patient Monitoring
MPI	–	Master Patient Index
НСЗУ	–	Національна служба здоров'я України
МІС	–	Медична інформаційна система
GDPR	–	General Data Protection Regulation
ЕСОЗ		Електронна система охорони здоров’я

ЗМІСТ

ВСТУП

Актуальність. Стрімкий розвиток цифрових технологій у сфері охорони здоров’я супроводжується широким впровадженням медичних інформаційних систем (МІС), які забезпечують автоматизацію процесів зберігання, обробки та передачі медичних даних. До таких систем належать електронні медичні картки, реєстри щеплень, бази даних пацієнтів, системи електронного рецепту та інші сервіси, що є критично важливими для ефективного функціонування сучасної медицини.

З одного боку, цифровізація охорони здоров’я забезпечує підвищення доступності медичних послуг, пришвидшує діагностику та покращує загальну якість лікування. З іншого боку, значні обсяги персональних медичних даних, що обробляються МІС, роблять ці системи об’єктами підвищеного ризику з точки зору інформаційної безпеки. Неавторизований доступ, викрадення або спотворення таких даних можуть призвести до серйозних наслідків – як для пацієнтів, так і для медичних установ: від порушення прав на конфіденційність до фінансових санкцій і втрати репутації.

Актуальність теми зумовлена зростанням кількості кібератак саме на медичну сферу, що свідчить про недостатню захищеність існуючих інформаційних систем. В умовах гібридних загроз, зростання кіберзлочинності та розвитку інструментів цифрового шпигунства забезпечення конфіденційності, цілісності та доступності медичних реєстрів стає ключовим завданням національної інформаційної безпеки.

Метою даної роботи є удосконалення механізмів захисту електронних реєстрів у медичних інформаційних системах шляхом розробки рекомендацій, спрямованих на підвищення рівня безпеки з урахуванням забезпечення конфіденційності, цілісності та доступності медичних даних

Для досягнення мети були поставлені такі завдання:

- охарактеризувати основні типи медичних інформаційних систем та їх структуру;
- проаналізувати можливі загрози та уразливості реєстрів медичних даних;
- дослідити сучасні механізми захисту, що використовуються в МІС;
- розробка програмного забезпечення для забезпечення захисту медичних реєстрів.

Об’єктом дослідження є процес захисту інформації в медичних інформаційні системи, що містять електронні реєстри пацієнтів.

Предметом дослідження є механізми забезпечення інформаційної безпеки в медичних реєстрах, зокрема засоби контролю доступу, шифрування, автентифікації, журналювання подій та резервного копіювання.

Методи дослідження:

- аналіз наукової літератури та нормативно-правових актів;
- структурно-функціональний аналіз інформаційних систем;
- порівняльний аналіз засобів інформаційного захисту;
- аналіз кейсів з практики впровадження ІБ у медичних установах.

Ключові слова: медичні інформаційні системи, захист даних, шифрування, автентифікація, багатофакторна автентифікація, контроль доступу, журнали аудиту, резервне копіювання, кіберзагрози.

РОЗДІЛ 1
АНАЛІЗ ЗАГАЛЬНИХ ПОЛОЖЕНЬ ЩОДО ЗАХИСТУ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Медичні інформаційні системи як об’єкт захисту

Системи медичної інформації являють собою складний комплекс зусиль щодо систематичного та сталого збору, аналізу та звітності про інформацію, а також отримання знань і доказів, що служать різним цілям з кінцевою метою покращення здоров'я. Термінологія, що використовується стосовно медичної інформації, включає такі поняття, як дані про здоров'я, медична інформація або статистика здоров'я , а також знання або докази . Історія статистики охорони здоров'я дає уявлення про те, як різноманітність даних зростала з часом і як краща медична інформація дозволила вченим і медичним працівникам досягати кращих результатів у сфері охорони здоров'я.

Основними цілями систем медичної інформації є забезпечення доказової бази для загальної політики та розподілу ресурсів, забезпечення доступу (особливо для вразливих груп), моніторинг якості медичної допомоги, що надається системою охорони здоров'я, розуміння загального стану здоров'я населення та факторів, що впливають на нього в різних умовах, а також забезпечення прозорості та підзвітності. Вивчення взаємодії між даними, що використовуються в національному та міжнародному контекстах, є важливим для підвищення ефективності медичних втручань.

Врахування соціальних детермінант здоров'я та їх міжгалузевого характеру спонукає до необхідності збору даних, отриманих в інших сферах і секторах, окрім охорони здоров'я. Системи реєстрації, такі як реєстрація актів цивільного стану та статистика життєвого циклу, є важливим джерелом медичної інформації. Залежно від контексту, для отримання статистики охорони здоров'я також використовуються такі ресурси, як реєстри, системи оповіщення,

опитування щодо здоров'я, джерела поза сектором охорони здоров'я . Інтеграція даних відіграє важливу роль у моніторингу політики охорони здоров'я, моніторингу та розумінні захворювань, а також у кращому відстеженні здоров'я та благополуччя людей. Однак етичні міркування та запобіжні заходи політики є життєво важливими при зборі, зберіганні, використанні та знищенні даних, їхній безпеці та згоді.

1.1.1 Види та цілі медичних інформаційних систем

Будь-яка уповноважена особа може використовувати системи управління інформацією в охороні здоров’я — це можуть бути як медичні працівники, так і пацієнти або представники сфери громадського здоров’я. Тепер, коли ми розуміємо, що таке медичні інформаційні системи та хто може ними користуватись, розглянемо основні приклади таких систем.

1. Електронні медичні записи (EMR) та електронні медичні картки (EHR)

Одними з найбільш поширених систем у медичній практиці є рішення EMR (Electronic Medical Records) та EHR (Electronic Health Records). Попри схожість, ці дві системи мають відмінності. Але спочатку розгляньмо їхню основну мету.

Системи EMR та EHR були впроваджені для заміни паперових медичних карток пацієнтів та для покращення результатів лікування. Ці записи зберігаються в електронному вигляді, щоб кожен уповноважений користувач мав до них швидкий доступ. До таких даних належать історія хвороби, результати лабораторних досліджень, життєві показники, історія оплат тощо.

Основна різниця між EMR та EHR полягає в масштабах і сфері використання. EMR містить медичні дані пацієнта в межах однієї клініки або медичної практики і не призначений для обміну з іншими установами. Натомість EHR дозволяє обмінюватися даними між різними медичними закладами.

2. Телемедицина та віддалений моніторинг пацієнтів (RPM)

Системи телемедицини та RPM (Remote Patient Monitoring) дозволяють лікарям надавати медичні послуги поза межами лікувального закладу. Ці рішення доповнюють одне одного, допомагаючи доставляти медичну допомогу в недостатньо забезпечені регіони та тим, хто її найбільше потребує.

Системи RPM дають змогу медичним працівникам дистанційно збирати медичні дані пацієнтів за допомогою носимих пристроїв на основі штучного інтелекту. Такі дані можуть включати життєві показники, патерни сну, рівень глюкози тощо. Це життєво важлива технологія, яка дозволяє зменшити кількість госпіталізацій, виявляти проблеми на ранньому етапі та надавати індивідуальну допомогу.

Системи телемедицини використовують телекомунікаційні технології для зв’язку між пацієнтами та медичними працівниками. Пацієнти можуть отримувати консультації за допомогою повідомлень, голосового або відеозв’язку.

3. Програмне забезпечення для управління медичною практикою

Програмне забезпечення для управління медичною практикою — ще один важливий приклад медичної інформаційної системи. Воно дозволяє оптимізувати повсякденні адміністративні та клінічні процеси медичного закладу. Системи такого типу допомагають з:

- плануванням прийомів,
- реєстрацією пацієнтів,
- виставленням рахунків і оплатою.

Основні переваги такого ПЗ:

- автоматизація адміністративних завдань;
- підвищення точності даних;
- покращення досвіду пацієнтів;
- ефективне управління фінансами.

Впровадження таких систем корисне як для невеликих приватних клінік, так і для великих медичних установ.

4. Пацієнтські портали

Сучасні пацієнти хочуть бути активними учасниками процесу лікування. Пацієнтські портали — це ефективні ІТ-рішення для підтримки залучення пацієнтів. Такі інформаційні системи дозволяють пацієнтам отримувати доступ до своєї медичної інформації, розкладу прийомів, результатів аналізів тощо.

Сучасні портали також надають функції:

- спілкування з лікарем у режимі онлайн,
- запиту на поновлення рецептів,
- самостійного запису на прийом.

Такі інструменти мотивують пацієнтів активніше дбати про своє здоров’я та сприяють покращенню результатів лікування.

5. Майстер-індекс пацієнтів (MPI)

Майстер-індекс пацієнтів — невід’ємна частина будь-якої інформаційної системи лікарні. Це централізована база даних, яка забезпечує ідентифікацію пацієнтів у межах одного медичного закладу або між різними установами охорони здоров’я.

Головна мета MPI — створення єдиного, надійного ідентифікатора для кожного пацієнта, який можна використовувати у всіх підсистемах. Це забезпечує узгодженість даних, покращує координацію медичної допомоги та сприяє прийняттю обґрунтованих клінічних рішень.

1.1.2 Особливості ведення та зберігання реєстрів у МІС

У контексті медичних інформаційних систем (МІС) поняття "реєстр" набуває форми електронної бази даних, призначеної для систематизованого обліку та зберігання інформації про пацієнтів, медичний персонал, надані послуги, лікарські засоби та інші важливі аспекти функціонування медичного закладу або системи охорони здоров'я загалом. Основними типами реєстрів, що використовуються в МІС, є: реєстри пацієнтів (що містять їхні персональні дані та історію звернень), реєстри медичних працівників, реєстри медичних послуг,

реєстри лікарських засобів, реєстри щеплень, листки непрацездатності тощо, приклад зображено на рис.1.1.

Рисунок 1.1. Діаграма структури медичних реєстрів

Особливості ведення реєстрів у МІС:

Нормативно-правові аспекти: Ведення медичної документації в електронній формі, включно з реєстрами, регулюється низкою нормативно-правових актів України. До них належать Закон України "Про електронні документи та електронний документообіг", Закон України "Про захист персональних даних", а також підзаконні акти Міністерства охорони здоров'я України, що визначають порядок ведення медичної облікової документації в електронному вигляді. Ці акти встановлюють вимоги до

юридичної сили електронних документів, порядку їх створення, зберігання, передачі та захисту.

Технологічні аспекти: Введення, редагування та оновлення даних у електронних реєстрах здійснюється за допомогою спеціалізованих інтерфейсів користувача, розроблених у складі МІС, на рис.1.2 зобразила процес внесення даних до реєстру. Ці інтерфейси повинні бути інтуїтивно зрозумілими та забезпечувати ефективне внесення інформації з мінімізацією помилок.

Важливим аспектом є інтеграція реєстрів з іншими модулями МІС, такими як електронна медична карта (ЕМК), системи призначення лікування, лабораторної та інструментальної діагностики. Це забезпечує автоматичне заповнення певних полів реєстрів на основі даних з інших підсистем, підвищуючи точність та зменшуючи необхідність ручного введення.

Процедурні аспекти: Стандартні процедури ведення реєстрів включають чітко визначені алгоритми внесення первинних даних (наприклад, при реєстрації пацієнта), оновлення інформації (наприклад, внесення діагнозу, наданих послуг), верифікації введених даних (перевірка на коректність та повноту), а також процедури контролю якості даних (періодичні перевірки на наявність помилок та дублювань). Важливим є документування всіх змін, що вносяться до реєстрів, із зазначенням автора та часу внесення.

Рисунок 1.2. Блок-схема процесу внесення даних до реєстру

Роль користувачів: У процесі ведення реєстрів беруть участь різні категорії медичного персоналу, кожна з яких має визначені права та обов'язки. Реєстратори відповідають за первинну реєстрацію пацієнтів, лікарі вносять дані про діагнози та надані послуги, медичні сестри фіксують виконання призначень тощо. Адміністратори системи відповідають за налаштування прав доступу та контроль за дотриманням процедур ведення реєстрів.

Особливості зберігання реєстрів у МІС:

Технічні аспекти: Електронні реєстри зберігаються в цифровому форматі на серверах медичних закладів або в захищених хмарних сховищах. Вибір технології зберігання залежить від розміру закладу, обсягу даних, вимог до безпеки та наявної інфраструктури. Використовуються різні типи баз даних, оптимізовані для зберігання та обробки великих обсягів структурованої інформації.

Вимоги до безпеки зберігання: Зберігання медичних даних, що містять персональну інформацію про стан здоров'я пацієнтів, вимагає забезпечення високого рівня конфіденційності, цілісності та доступності, можна побачити на рисунку 1.3. Для цього застосовуються такі заходи безпеки, як:

Шифрування даних: Застосування криптографічних методів для захисту даних як під час передачі, так і під час зберігання.

Контроль доступу: Впровадження багаторівневої системи контролю доступу, що обмежує права користувачів відповідно до їхніх посадових обов'язків.

Резервне копіювання (бекап): Регулярне створення резервних копій баз даних реєстрів та їх зберігання у безпечному місці для забезпечення можливості відновлення даних у випадку збоїв або втрати основних даних.

Фізична безпека: Забезпечення фізичного захисту серверного обладнання від несанкціонованого доступу, пошкоджень або крадіжки.

Використання міжмережевих екранів (firewalls) та систем виявлення вторгнень (IDS/IPS) для захисту мережевої інфраструктури, на якій розміщені сервери з даними реєстрів.

Рисунок 1.3. Інфографіка для зберігання даних

Посилання на відповідні стандарти інформаційної безпеки, такі як ISO 27001, а також галузеві рекомендації щодо захисту медичної інформації.

Терміни зберігання медичних даних: Терміни зберігання різних видів медичної документації в електронній формі визначаються законодавством та внутрішніми нормативними актами медичного закладу. Важливо забезпечити дотримання цих термінів для уникнення правових наслідків та забезпечення можливості використання даних у майбутньому (наприклад, для статистичних досліджень).

Забезпечення безперервності доступу: Системи зберігання даних повинні бути надійними та відмовостійкими, щоб забезпечити безперебійний доступ медичного персоналу до необхідної інформації у будь-який час. Це може включати використання кластерних систем, дзеркалювання даних та інших технологій забезпечення високої доступності.

Аналіз переваг та недоліків електронного ведення та зберігання реєстрів порівняно з паперовими формами:

Електронне ведення та зберігання реєстрів у МІС(схему зберігання зобразила на рис.1.4) має значні переваги порівняно з традиційними паперовими формами, включаючи:

Підвищення ефективності: Швидкий доступ до інформації, усунення необхідності пошуку паперових документів, автоматизація рутинних операцій.

Зменшення кількості помилок: Автоматична перевірка даних, мінімізація ручного введення.

Покращення якості даних: Забезпечення цілісності та узгодженості інформації.

Покращення координації медичної допомоги: Миттєвий обмін інформацією між різними підрозділами та медичними працівниками.

Можливість аналізу та формування звітів: Зручний інструмент для статистичного аналізу та прийняття управлінських рішень.

Економія місця для зберігання: Відсутність потреби у великих архівах для паперових документів.

Рисунок 1.4. Схема резервного копіювання та відновлення даних

Однак існують і потенційні ризики та виклики:

Кіберзагрози: Ризик несанкціонованого доступу, викрадення, пошкодження або знищення даних.

Технічні збої: Можливість втрати даних внаслідок апаратних або програмних помилок.

Людський фактор: Помилки користувачів при введенні даних, порушення політик безпеки.

Необхідність інвестицій: Витрати на впровадження та підтримку МІС.

1.2 Нормативно-правове та концептуальне підґрунтя захисту

Цей розділ висвітлює основні законодавчі акти України та міжнародні стандарти, що регулюють захист персональних даних у сфері охорони здоров’я, а також ключові принципи інформаційної безпеки, які застосовуються в медичних інформаційних системах.

1.2.1 Вимоги до захисту персональних даних у медичних системах

Закон України «Про захист персональних даних» (2010):

Визначає основні поняття, такі як «персональні дані», «обробка персональних даних», «володілець» та «розпорядник» даних.

Встановлює принципи обробки персональних даних: законність, справедливість, точність, обмеження мети та зберігання.

Гарантує права суб’єктів персональних даних, включаючи право на доступ, виправлення, видалення та заперечення проти обробки даних.

Передбачає посилені вимоги до обробки особливих категорій даних, зокрема медичних даних.

Закон України «Основи законодавства України про охорону здоров’я»:

Регламентує питання лікарської таємниці та конфіденційності медичної інформації.

Визначає обов’язки медичних працівників щодо збереження конфіденційності пацієнтів.

Закон України «Про захист інформації в інформаційно-телекомунікаційних системах»:

Встановлює вимоги до захисту інформації в інформаційних системах, включаючи медичні інформаційні системи.

Передбачає необхідність впровадження технічних та організаційних заходів безпеки.

Міжнародні стандарти:

Загальний регламент захисту даних (GDPR): Надає рамки для обробки персональних даних у країнах ЄС та застосовується до українських організацій, що співпрацюють з ЄС.

Конвенція №108 Ради Європи: Спрямована на захист фізичних осіб у зв’язку з автоматизованою обробкою персональних даних.

1.2.2 Основні принципи інформаційної безпеки в медичних інформаційних системах

Медичні інформаційні системи (МІС) забезпечують збирання, зберігання, обробку та передачу персональних і медичних даних пацієнтів. Оскільки така інформація є конфіденційною та критично важливою для здоров’я і життя людей, до неї пред’являються підвищені вимоги безпеки.

У таблиці 1 в додатку А зображено інструменти та засоби для реалізації принципів інформаційної безпеки в МІС. Для забезпечення надійного функціонування МІС застосовуються такі основні принципи інформаційної безпеки(на рис.1.5 додатково зобразила діаграму Вена):

Основні принципи:

Конфіденційність (Confidentiality)

Забезпечення захисту даних від несанкціонованого доступу сторонніх осіб.

Приклад: лише лікар має доступ до історії хвороби свого пацієнта.

Цілісність (Integrity)

Гарантування того, що дані не будуть змінені або знищені без відповідного дозволу.

Приклад: будь-яке оновлення в електронній картці пацієнта фіксується в журналі змін.

Доступність (Availability)

Своєчасний і безперебійний доступ до інформації для уповноважених користувачів.

Приклад: система працює навіть при відмові одного з серверів.

Рисунок 1.5. Діаграма Вена

Аутентифікація та авторизація

Ідентифікація користувача та контроль його прав доступу.

Приклад: використання логіна, пароля та токена доступу для входу в систему.

Відповідальність (Audit & Accountability)

Можливість відслідкувати дії кожного користувача в системі.

Приклад: перегляд журналу подій показує, хто і коли вносив зміни до медичних записів.

Дотримання законодавства

Відповідність системи вимогам українського та міжнародного законодавства щодо захисту персональних даних (Закон України «Про захист персональних даних», GDPR, ISO/IEC 27001 та ін.).

Висновки за розділом 1

Медичні інформаційні системи (МІС) є критично важливими елементами сучасної медичної практики. Вони забезпечують ефективне зберігання, обробку та доступ до медичних даних, що дозволяє покращити якість медичних послуг, моніторинг здоров'я населення та оптимізацію управлінських процесів у медичних закладах. Водночас, для їх належного функціонування важливо забезпечити відповідний рівень безпеки, щоб захистити конфіденційність і цілісність медичних даних.

Інтеграція різноманітних видів медичних інформаційних систем (електронні медичні картки, телемедицина, пацієнтські портали, тощо) надає нові можливості для покращення доступу до медичних послуг, що особливо важливо для віддалених і недостатньо забезпечених регіонів. Проте це також підвищує ризики порушення безпеки, що вимагає застосування спеціальних технологічних рішень для забезпечення конфіденційності та цілісності даних.

Реєстри в МІС займають центральне місце в управлінні медичними даними, дозволяючи забезпечити точність, актуальність та доступність інформації. Водночас ведення реєстрів вимагає суворого дотримання нормативно-правових вимог, високої технологічної та процедурної точності, а також відповідної організаційної підготовки медичних працівників.

Захист персональних даних та інформаційна безпека є першочерговими завданнями при впровадженні та експлуатації медичних інформаційних систем.

З огляду на специфіку медичних даних, що містять персональну та чутливу інформацію, необхідно застосовувати принципи конфіденційності, цілісності, доступності та відповідальності, а також ретельно контролювати відповідність чинному законодавству.

Законодавче та нормативне підґрунтя захисту медичних даних в Україні забезпечує достатній рівень захисту персональних даних, однак, враховуючи глобальні тенденції в розвитку технологій, для забезпечення належного рівня безпеки важливо орієнтуватися на міжнародні стандарти, такі як GDPR та ISO/IEC 27001.

Технічні засоби захисту, такі як шифрування, двофакторна аутентифікація та резервне копіювання, є необхідними для забезпечення безпеки медичних інформаційних систем. Водночас важливо не забувати про людський фактор і необхідність навчання користувачів щодо дотримання політик безпеки, що значно знижує ризик помилок або зловмисних дій.

Подальший розвиток медичних інформаційних систем вимагає постійного вдосконалення технологій та підвищення кваліфікації медичних працівників, що працюють з такими системами. Адже для досягнення максимальної ефективності і мінімізації ризиків важливо забезпечити належну інтеграцію, безпеку та конфіденційність оброблюваних даних.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ СУЧАСНОГО СТАНУ ЗАХИСТУ РЕЄСТРІВ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Медичні інформаційні системи як об’єкт захисту

У зв'язку з постійним розвитком медичних інформаційних систем (МІС) та їх інтеграцією в національні та міжнародні медичні мережі, питання безпеки цих систем набуває надзвичайної важливості. Реєстри медичних інформаційних систем є особливо вразливими до ряду зовнішніх і внутрішніх загроз, що можуть призвести до серйозних наслідків, як для окремих пацієнтів, так і для медичних установ загалом. Визначення цих загроз та ризиків є важливим етапом для розробки ефективних механізмів захисту медичних реєстрів.

2.1.1 Кіберзагрози, що впливають на безпеку медичних реєстрів.

Медичні реєстри є мішенню для різних типів кіберзагроз. Серед найбільш розповсюджених можна виділити такі:

Атаки за допомогою програм-вимагачів (ransomware). Одна з найбільших загроз для медичних установ – це шкідливі програми, які шифрують файли та вимагають викуп за їх розшифрування.

DDoS-атаки перевантажують сервери медичних установ величезним обсягом трафіку, роблячи медичні реєстри та інші життєво важливі системи недоступними. Хоча DDoS-атаки зазвичай не призводять до крадіжки даних, вони можуть серйозно порушити роботу лікарень, затримуючи або повністю блокуючи доступ до медичних записів та надання невідкладної допомоги.

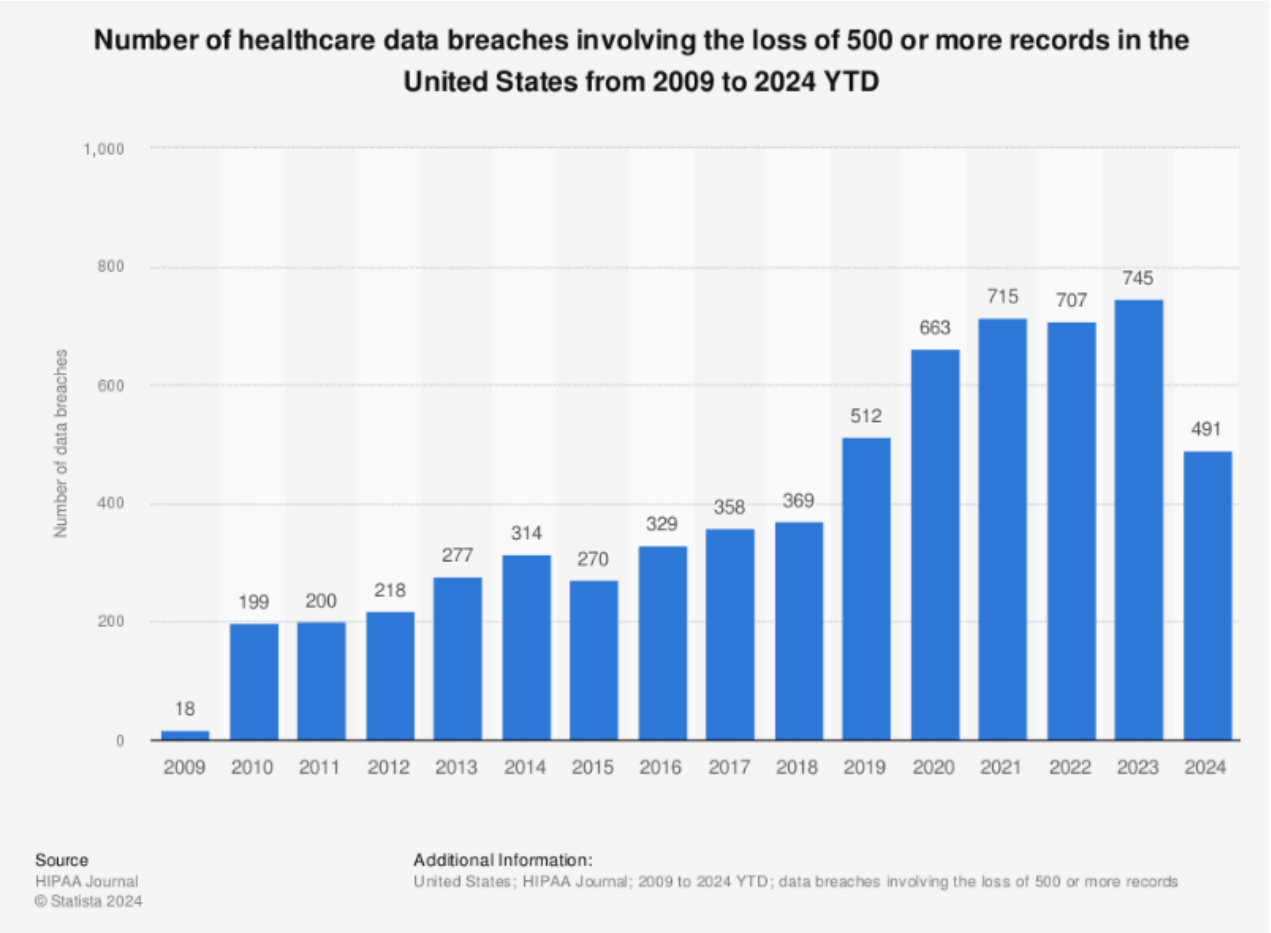

Рисунок 2.1. Зростання кількості витоків даних, що становили понад 500 записів у галузі охорони здоров'я, з 2009 по 2024 рік.

Наприклад, атакування лікарень у США та Великобританії в результаті яких були зупинені важливі медичні сервіси, а доступ до медичних записів став обмеженим, зобразила графік на рис.2.1. Такі атаки можуть призвести до серйозних наслідків для пацієнтів, зокрема до затримок в наданні медичних послуг.

Фішинг та соціальна інженерія. Часто медичні установи стають жертвами фішингових атак, де зловмисники маскуються під легітимних співробітників чи організації і намагаються отримати доступ до конфіденційних даних.

Приклад: Відомі випадки, коли працівники лікарень отримували електронні листи від «департаменту IT», що запитували пароль для «оновлення системи», і таким чином зловмисники отримували доступ до медичних реєстрів.

Незаконний доступ до даних. Зловмисники можуть використовувати вразливості в програмному забезпеченні або несанкціоновані пристрої для доступу до баз даних з медичною інформацією. Зазвичай це стосується слабкого контролю доступу та відсутності багаторівневого автентифікації.

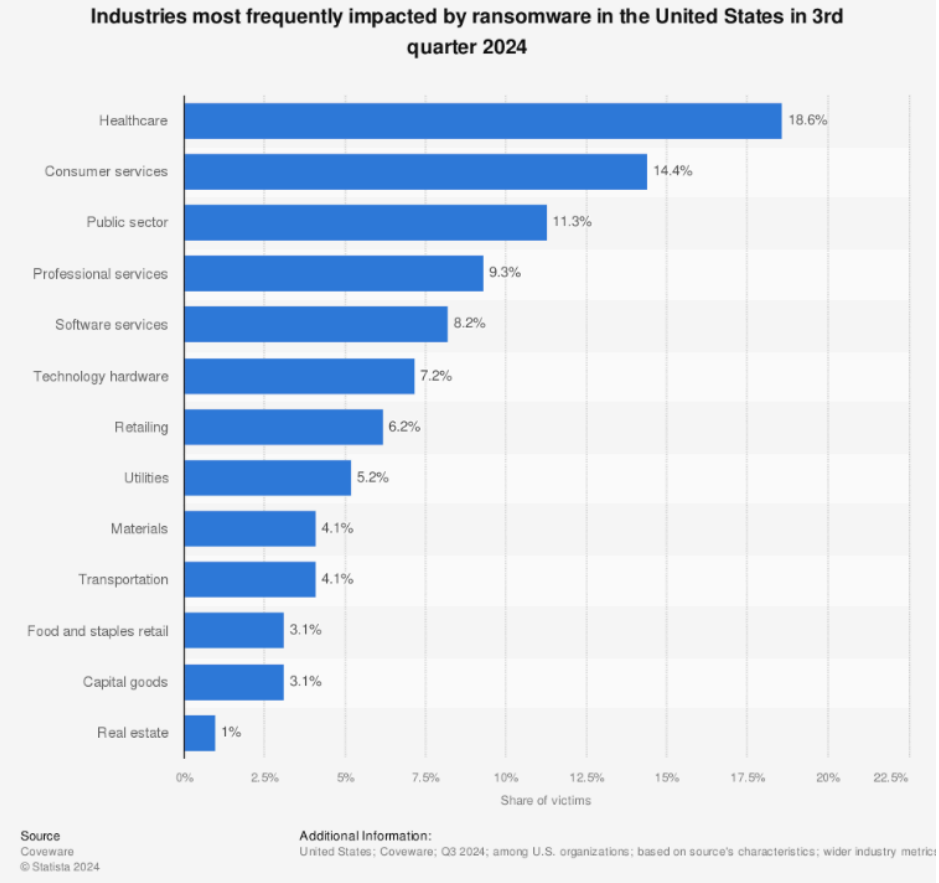

Рисунок 2.2. Графік, що показує поширеність атак програм-вимагачів, спрямованих на галузь охорони здоров'я

Медичні установи часто використовують сторонні сервіси та продукти (наприклад, системи управління пацієнтами, лабораторне обладнання, хмарні сховища). Атаки через ланцюг поставок відбуваються, коли зловмисники компрометують одного з цих сторонніх постачальників, щоб потім отримати доступ до систем їхніх клієнтів – у даному випадку, медичних реєстрів, зобразила графік на рис.2.2. Це може бути надзвичайно складно виявити, оскільки атака походить від довіреного джерела.

2.1.2 Внутрішні ризики та людський фактор

Не менш важливими є внутрішні загрози, пов'язані з людським фактором, який є однією з основних причин витоків даних або помилок у роботі медичних систем:

Недостатня підготовка персоналу. Медичний персонал може не бути належним чином навчений щодо стандартів безпеки та процедур обробки даних. Це може призводити до таких помилок, як випадковий витік інформації чи небезпечні практики в обробці персональних даних.

Приклад можна побачити на рис.2.3: Поширеною помилкою є використання слабких паролів або збереження паролів у відкритому доступі, що підвищує ризик несанкціонованого доступу.

Рисунок 2.3. Приклад використання слабкого паролю

Незабезпечений фізичний доступ. Часто проблеми з безпекою виникають через фізичний доступ до серверів і комп'ютерів, що містять медичні дані. Відсутність належних систем контролю доступу до серверних кімнат або

робочих станцій може призвести до крадіжки даних або установки шкідливого ПЗ.

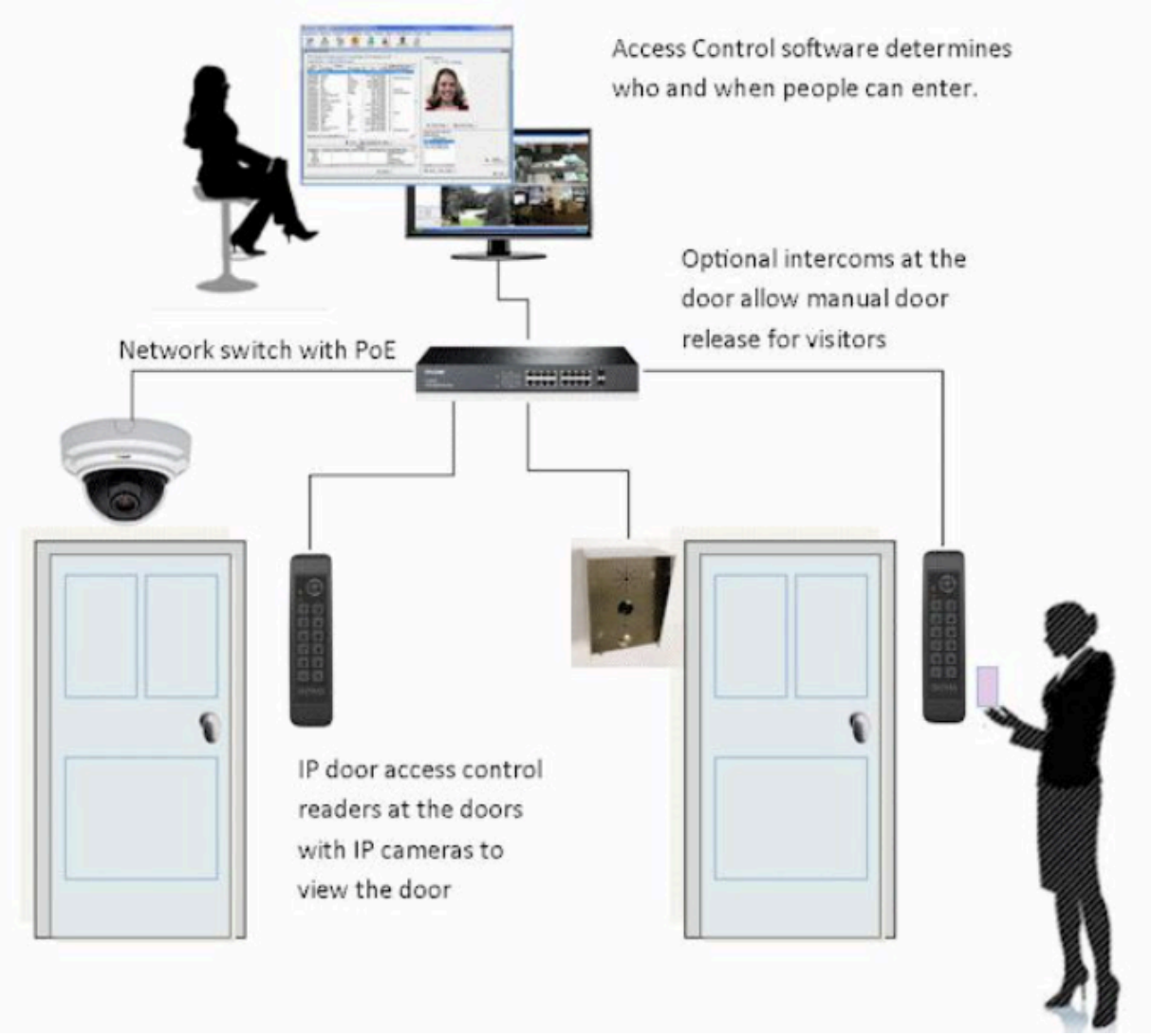

Рисунок 2.4. Система контролю доступу з IP-камерами та мережею PoE

Ця система, приклад зображений на рис.2.4, забезпечує високий рівень фізичної безпеки приміщень, де зберігаються сервіри з медичними даними. IP-камери дозволяють здійснювати віддалене відеоспостереження, а технологія Power over Ethernet (PoE) забезпечує зручність у монтажі, передаючи як дані, так і живлення через один мережевий кабель. Це допомагає знизити ризик фізичного втручання або крадіжки інформації.

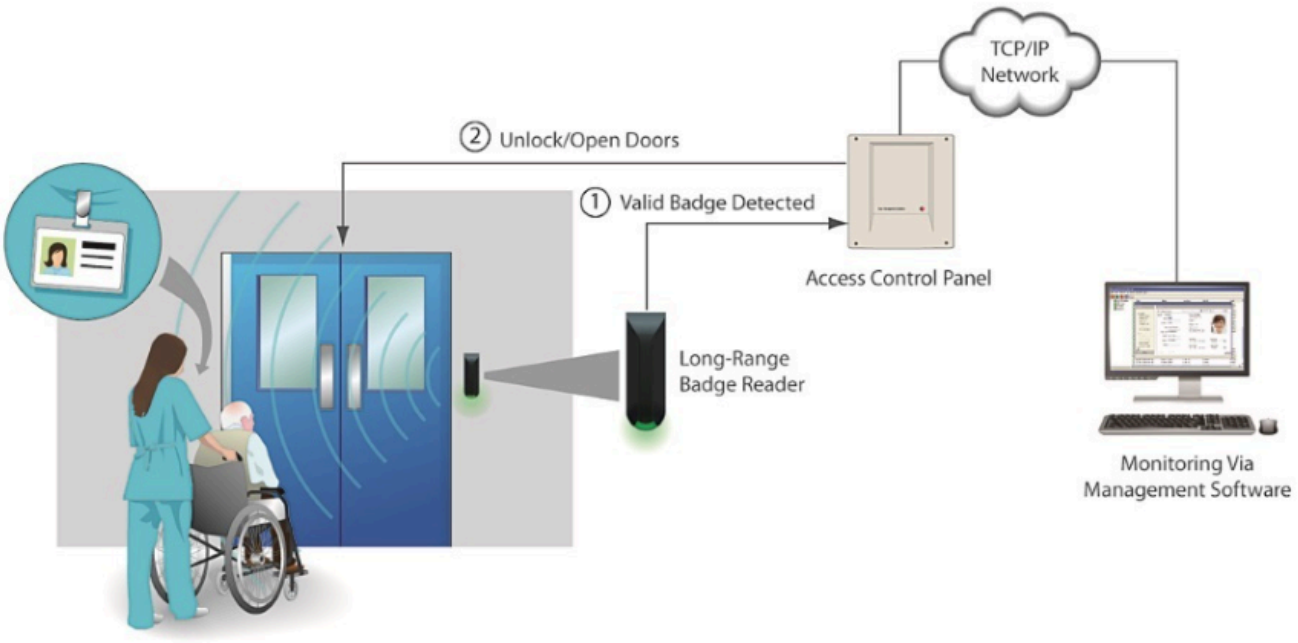

Рисунок 2.5. Безконтактна система контролю доступу з довготривалим зчитуванням бейджів

Безконтактна система контролю доступу дозволяє обмежити доступ до важливих серверних кімнат або кабінетів тільки тим персоналом, який має відповідний дозвіл. Цей метод, зображений на рис.2.5, також дозволяє легко відслідковувати, хто і коли заходив в обмежену зону, що є важливим для забезпечення безпеки медичних даних.

Важливою перевагою є те, що система працює без необхідності фізичного контакту з пристроєм, що робить її зручнішою та швидшою.

ID-картки з вбудованими чіпами, приклад зображений на рис.2.6, або RFID технологією забезпечують високий рівень контролю за доступом до важливих приміщень. Цей метод дозволяє швидко і безпечно перевіряти права доступу та забезпечує документування кожного входу/виходу.

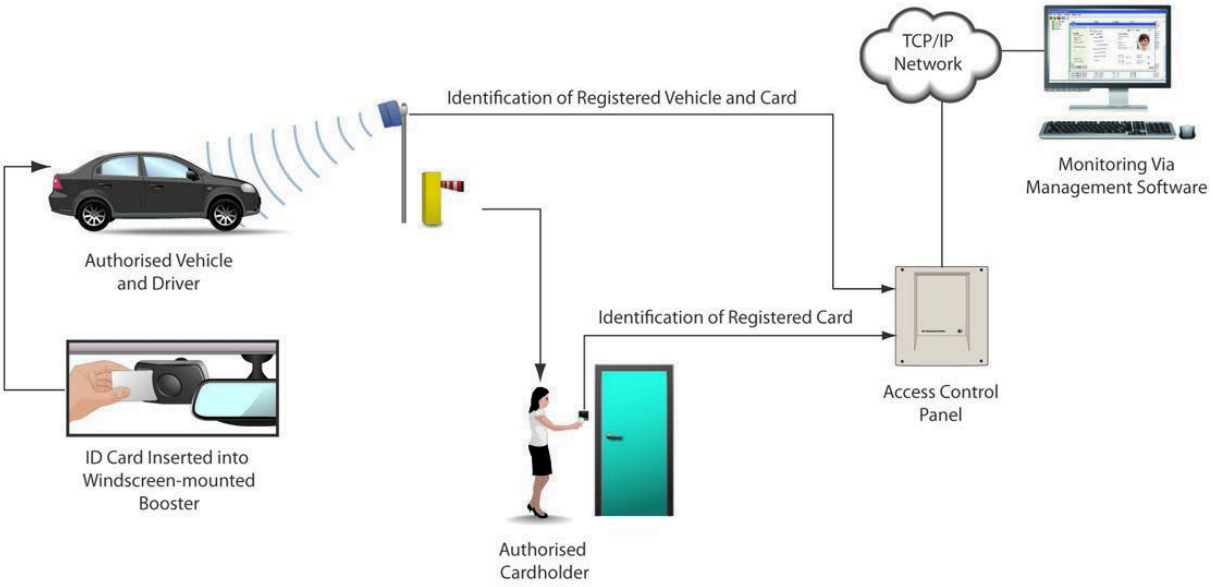

Рисунок 2.6. Автоматизований контроль в’їзду та входу за ID-картами

Це дозволяє уникнути несанкціонованого доступу до медичних серверів та баз даних. Додатково, система може бути інтегрована з програмним забезпеченням для моніторингу та створення звітів про активність у приміщеннях з високим рівнем безпеки.

2.2 Існуючі механізми захисту

Для мінімізації кіберзагроз та зменшення внутрішніх ризиків, медичні установи впроваджують різноманітні механізми захисту, серед яких важливими є технічні засоби та організаційні заходи.

2.2.1 Технічні засоби: шифрування, контроль доступу, аудит

Шифрування. Це один з основних методів захисту медичних даних. Всі чутливі дані, включаючи медичні записи пацієнтів, повинні бути зашифровані як на етапі зберігання, так і під час передачі через мережу. Застосування шифрування, таких як AES-256 для зберігання даних і TLS/SSL для захищеної передачі даних, є стандартом в індустрії.

Контроль доступу. Один з найбільш ефективних способів захисту медичних реєстрів – це обмеження доступу до даних. Кожен співробітник має отримати доступ лише до тих даних, які необхідні для виконання його професійних обов'язків. Також важливим є використання мультирівневого контролю доступу та многофакторної автентифікації для більшої безпеки.

Аудит та моніторинг. Постійний моніторинг і аудит дій користувачів є необхідним для виявлення будь-яких аномалій або потенційних спроб несанкціонованого доступу. Ведення журналів аудиту дає можливість відстежувати, хто, коли і до яких даних мав доступ.

Рисунок 2.7. Шарова структура механізмів захисту медичних даних

Це діаграма, зображена на рис.2.7, яка показує, як технічні та організаційні заходи працюють разом для забезпечення безпеки медичних даних. Можна розділити діаграму на дві основні частини: Технічні заходи (шифрування, контроль доступу, аудит) та Організаційні заходи (політика безпеки, підготовка персоналу).

Технічні заходи можуть бути на одному рівні: шифрування, контроль доступу, аудит, моніторинг.

Організаційні заходи — на іншому рівні: політика безпеки, підготовка персоналу, планування відновлення, регулярний аудит.

2.2.2 Організаційні заходи: політики безпеки, підготовка персоналу

Організаційні заходи є важливою частиною забезпечення захисту медичних інформаційних систем. Вони передбачають створення відповідної політики безпеки та ефективну підготовку персоналу для збереження цілісності, конфіденційності та доступності медичних даних.

Основні аспекти організаційних заходів

1. Політика безпеки

Політика безпеки — це сукупність принципів, правил та процедур, які визначають, як організація забезпечує захист своїх медичних інформаційних систем та даних. Вона повинна бути чітко сформульована, зрозуміла для всіх співробітників та відповідати національним та міжнародним стандартам.

Основні елементи політики безпеки:

Контроль доступу: визначення прав доступу до систем, баз даних, а також контролю за використанням облікових записів.

Безпека комунікацій: захист передавання даних між медичними установами, через Інтернет та інші канали зв'язку.

Резервне копіювання та відновлення: створення планів на випадок втрати або пошкодження медичних даних.

Управління інцидентами: розробка процедур для реагування на інциденти безпеки, такі як атаки, порушення конфіденційності чи збої систем.

2. Підготовка персоналу

Захист медичних інформаційних систем неможливий без належної підготовки персоналу. Всі співробітники, від медиків до IT-фахівців, повинні бути навчені основам кібербезпеки та стандартам роботи з медичними даними.

Ключові аспекти підготовки персоналу:

Освітні тренінги: регулярні навчання та тренінги, що висвітлюють новітні загрози, методи захисту даних і процедури в разі порушення безпеки.

Інструкції та протоколи: детальні інструкції щодо безпечного використання медичних інформаційних систем, а також дії у разі виявлення порушення безпеки.

Оцінка рівня знань: періодичні тести та оцінка знань співробітників для визначення рівня їх підготовленості та здатності реагувати на можливі інциденти.

3. Регулярний аудит та моніторинг

Для ефективного управління безпекою необхідно здійснювати регулярний аудит і моніторинг політик і практик безпеки. Це дозволяє виявити слабкі місця в системі захисту, своєчасно виявити інциденти та порушення безпеки, а також коригувати заходи безпеки.

4. Планування на випадок надзвичайних ситуацій

Існує необхідність у розробці плану відновлення після збоїв або кібератак. Він включає чітке визначення ролей та обов'язків персоналу, порядок дій у разі порушення безпеки або збоїв у роботі системи.

Рисунок 2.8. Структура політики безпеки медичних інформаційних систем.

Можна підсумувати, що політика безпеки та підготовка персоналу є фундаментом ефективного захисту медичних інформаційних систем. Вони повинні бути інтегровані в загальну стратегію безпеки установи, приклад такої структури зображений а рис.2.8, забезпечуючи належний рівень захисту та надійності роботи з медичними даними.

Висновки за розділом 2

Захист медичних інформаційних систем є важливою складовою забезпечення безпеки персональних даних пацієнтів. В умовах постійного зростання кіберзагроз та внутрішніх ризиків, таких як людський фактор і помилки персоналу, медичні установи повинні впроваджувати комплексні заходи безпеки, що поєднують як технічні, так і організаційні рішення.

Основними технічними заходами є шифрування даних, контроль доступу та постійний моніторинг діяльності користувачів, що дозволяють забезпечити конфіденційність і цілісність медичних записів. У свою чергу, організаційні заходи, такі як чітко сформульовані політики безпеки, управління інцидентами

та регулярне навчання персоналу, грають ключову роль у запобіганні помилок і зменшенні людських факторів ризику.

Забезпечення належного рівня безпеки медичних даних потребує постійного вдосконалення існуючих механізмів захисту та адаптації до нових загроз і технологій. Лише інтеграція сучасних технічних засобів з ефективними організаційними політиками забезпечує надійний захист чутливих даних і дозволяє мінімізувати потенційні ризики для пацієнтів та медичних установ.

РОЗДІЛ 3
РОЗРОБКА МЕХАНІЗМІВ ДЛЯ ЗАХИСТУ ДОСТУПУ ДО РЕЄСТРІВ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ (МІС)

3.1 Архітектура та функціональні можливості МІС

3.1.1 Робота з базою даних та облік медичної інформації.

Цей проєкт присвячений розробці механізмів для захисту доступу до реєстрів медичних інформаційних систем (МІС). В умовах зростаючих кіберзагроз та суворих вимог до конфіденційності медичної інформації, надійний захист доступу до електронних реєстрів пацієнтів є невід'ємним елементом сучасної інфраструктури охорони здоров'я.

В рамках цього проєкту було спроєктовано та реалізовано прототип системи, що являє собою надійний програмний бар'єр для захисту критично важливого доступу. Програмне рішення розроблене на мові Python та спирається на перевірені часом бібліотеки: bcrypt для безпечного хешування паролів, pyotp для реалізації TOTP-алгоритму, qrcode для зручної інтеграції з мобільними автентифікаторами, а також вбудовані модулі для логування та роботи з даними у форматі JSON.

Архітектурно система є модульною, що забезпечує її гнучкість, масштабованість та легкість у підтримці. Ключовими компонентами, що формують її ядро, є:

Модуль первинної автентифікації. Перший ешелон захисту, що відповідає за верифікацію пари логін/пароль. Для унеможливлення компрометації облікових даних паролі не зберігаються у відкритому вигляді.

Натомість система використовує криптографічний алгоритм bcrypt для створення безпечних хешів з додаванням унікальної "солі" для кожного

користувача, що робить процес зворотного відновлення пароля обчислювально неможливим.

Модуль двофакторної автентифікації (2FA). Серце системи, що реалізує другий, динамічний рівень захисту на основі алгоритму TOTP (Time-based One-Time Password), можна побачити на рис.3.1. Завдяки бібліотеці pyotp, система генерує тимчасові коди, повністю сумісні з популярними мобільними додатками, такими як Google Authenticator, забезпечуючи перевірений та зручний для користувача метод підтвердження особи.

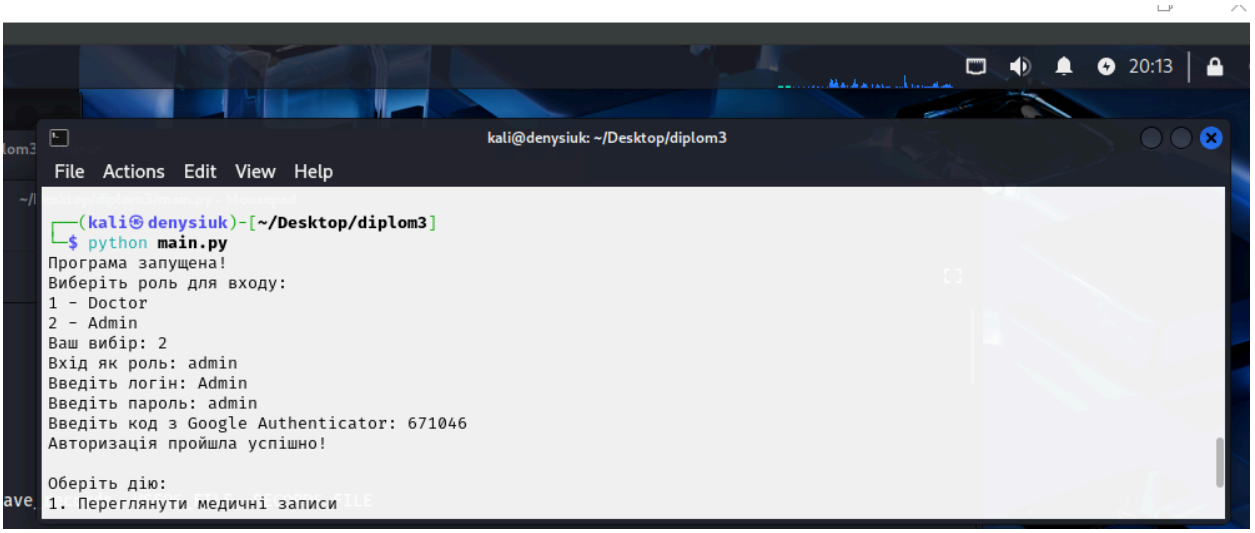

Рисунок 3.1. Приклад підтвердження коду з Google Authenticator

Для максимальної зручності користувачів, система автоматично генерує QR-код під час реєстрації. Сканування цього коду миттєво налаштовує мобільний додаток для генерації OTP-кодів, роблячи процес налаштування 2FA інтуїтивно зрозумілим. Це значно спрощує введення другого фактора захисту, усуваючи необхідність ручного введення довгих секретних ключів. Користувачеві достатньо один раз відсканувати код, і його мобільний пристрій буде готовий генерувати унікальні одноразові паролі кожні 30-60 секунд, додаючи суттєвий рівень безпеки до облікових записів. Інтерфейс додатку Google Authenticator зображено на рис. 3.2.

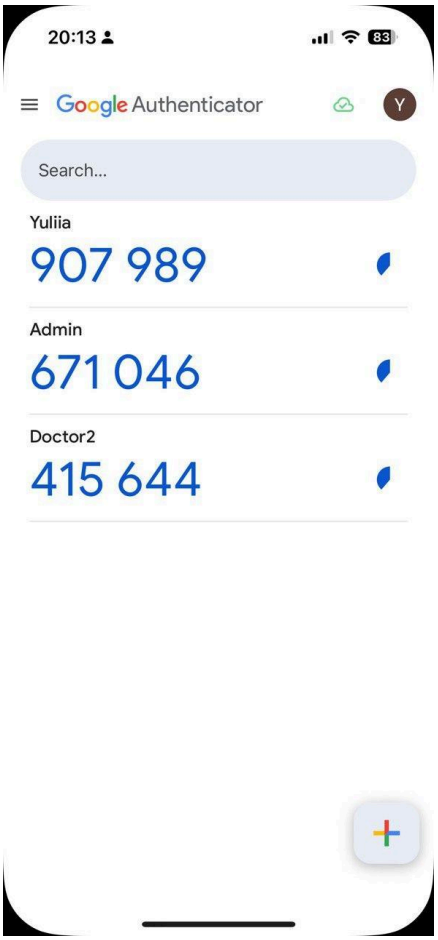

Рисунок 3.2. Інтерфейс додатку Google Authenticator

На рис. 3.1 та рис. 3.2 можна переконатися, що синхронізація мобільного додатку Google Authenticator та програмної реалізації системи пройшла успішно. Це свідчить про повну сумісність та надійність впровадженого механізму двофакторної автентифікації. Успішна синхронізація є ключовим етапом, адже вона забезпечує безперебійну генерацію та верифікацію одноразових паролів, що є фундаментом для ефективного захисту доступу до чутливих даних. Така інтеграція не тільки підсилює безпеку, але й зберігає високий рівень зручності для користувачів, які вже звикли до подібних рішень у повсякденному житті.

Модуль управління реєстром пацієнтів, приклад зобразила на рис.3.3. Захищене сховище даних (реалізоване у вигляді JSON-файлу), доступ до якого відкривається лише після успішного проходження обох етапів автентифікації. Гнучка система ролей дозволяє лікарям редагувати записи, тоді як

адміністратори мають повноваження лише для перегляду або управління правами доступу.

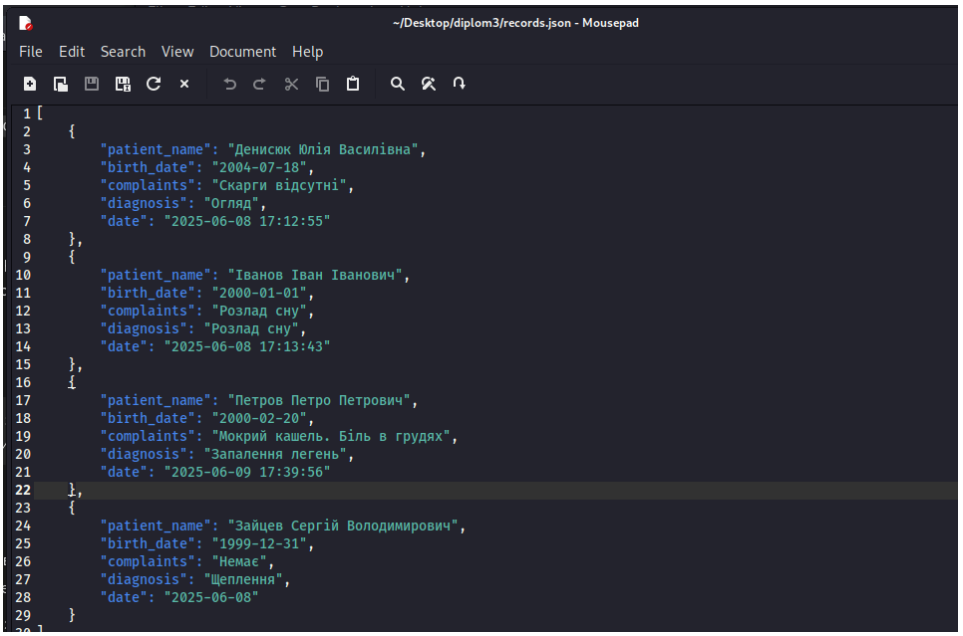

Рисунок 3.3. Модуль управління реєстром пацієнтів

Модуль аудиту та логування, зобразила на рис.3.5 та рис.3.4. Для забезпечення повної підзвітності та аналізу потенційних інцидентів безпеки, цей компонент ретельно фіксує всі значущі події: успішні та невдалі спроби входу, зміни в реєстрі та дії адміністраторів. Журнали подій зберігаються в окремих файлах (.txt або .csv) для подальшого аналізу.

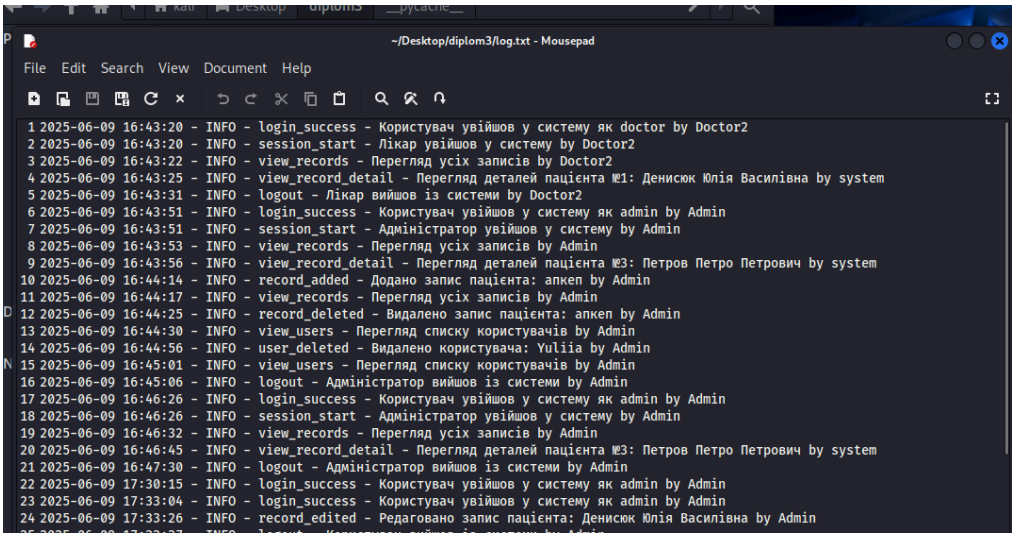

Рисунок 3.4. Модуль логування

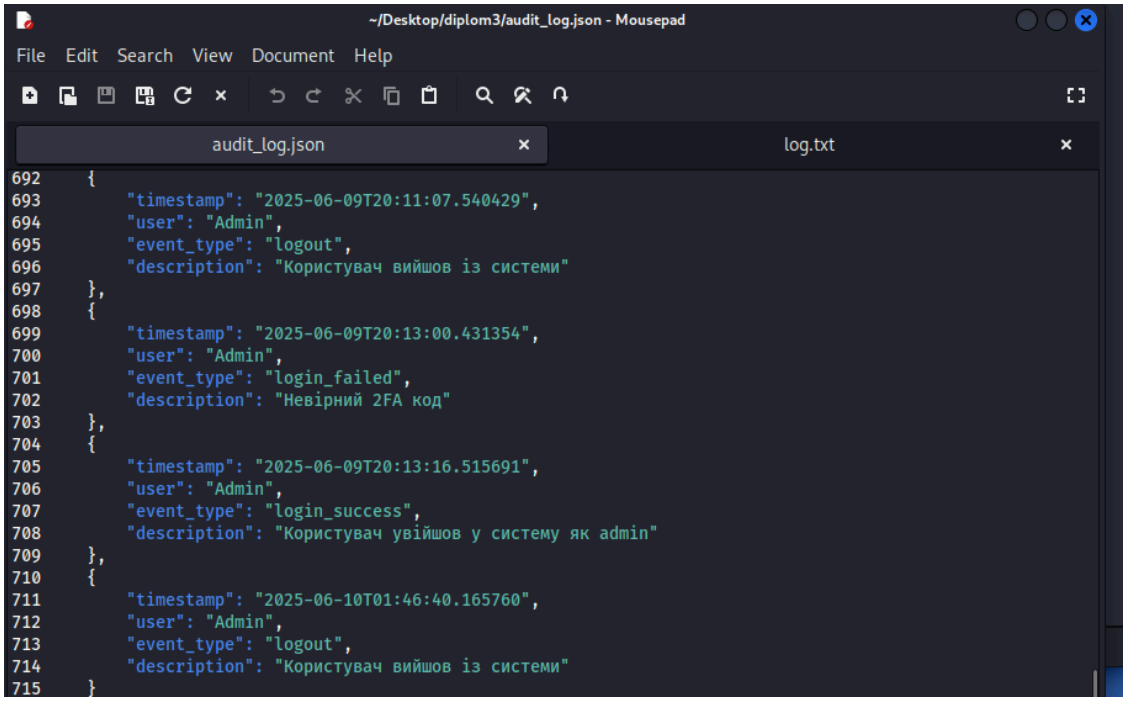

Рисунок 3.5. Модуль аудиту

Модуль адміністрування користувачів. Надає адміністраторам повний контроль над життєвим циклом облікових записів: від створення нових користувачів та призначення їм ролей до блокування чи видалення існуючих.

На рис.3.7 зображено файл з відкритими даними користувачів. Також було створено файл із зашифрованими даними користувачів, зображено на рис.3.6.

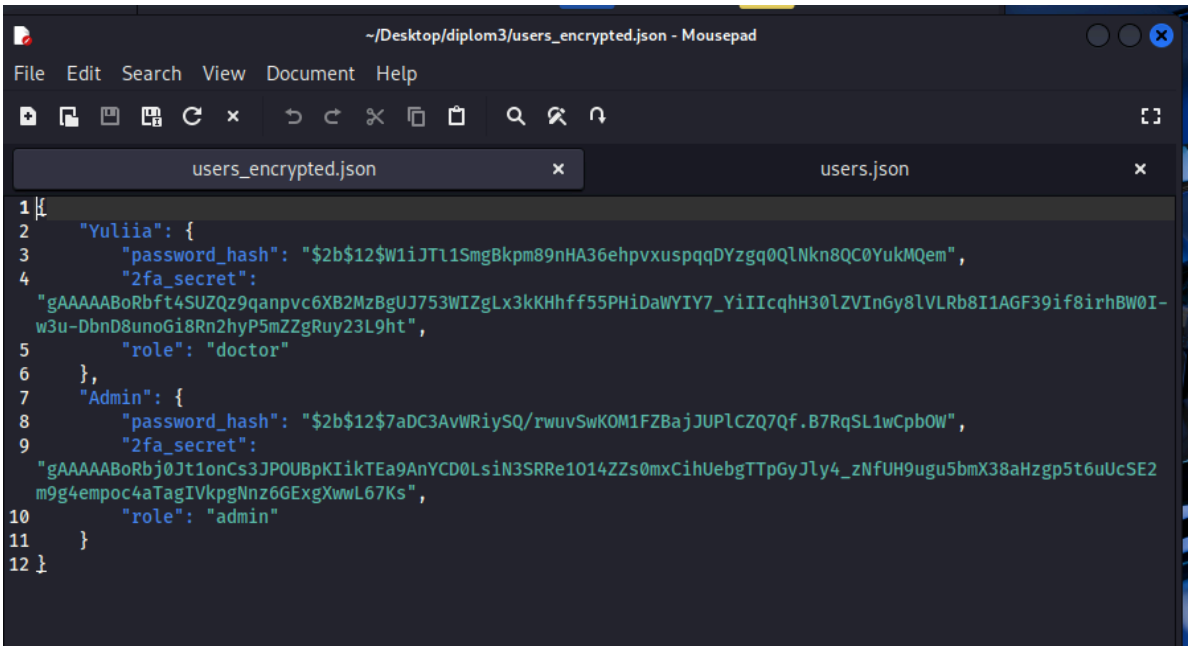

Рисунок 3.6. Зашифрований файл із данними користувачів

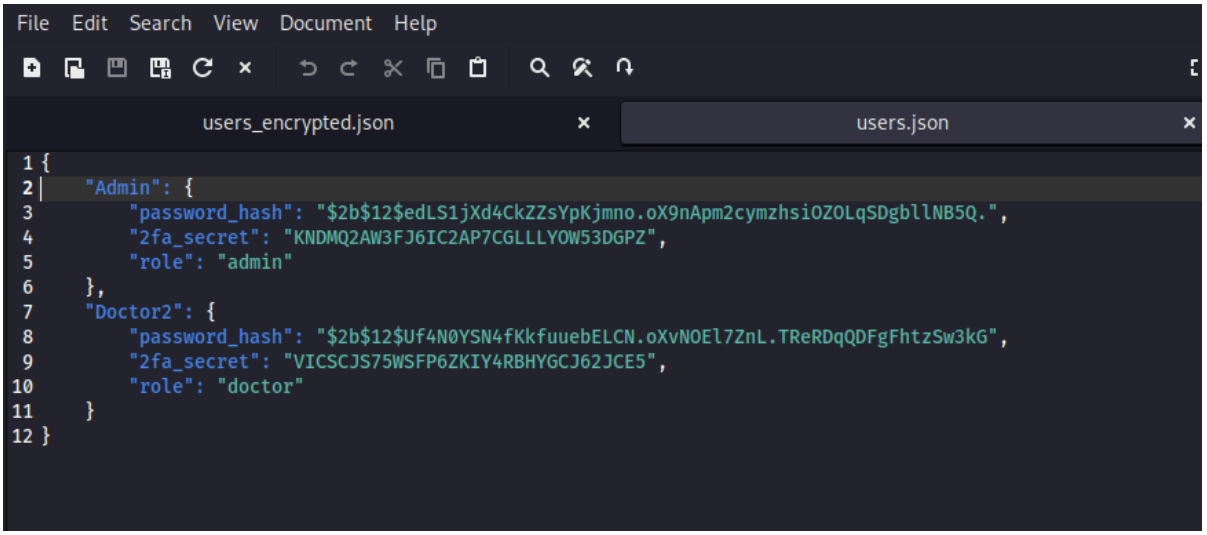

Рисунок 3.7. Файл із відкритим 2fa_secret

3.1.2 **Функціональні можливості**

Розроблена система пропонує комплексний набір функцій для забезпечення всебічного захисту та керованості:

Безпечна реєстрація: Створення користувачів з унікальним логіном, надійним паролем та персональним секретним ключем для 2FA.

Криптографічний захист паролів: Використання bcrypt гарантує, що навіть у випадку витоку бази даних, паролі залишаться захищеними.

Стандартна TOTP-автентифікація: Генерація та валідація одноразових паролів за допомогою pyotp забезпечує сумісність та надійність.

Швидке налаштування 2FA: Автоматична генерація QR-коду для миттєвої прив'язки до Google Authenticator чи аналогічних додатків.

Двоетапна процедура входу: Послідовна перевірка статичного пароля та динамічного OTP-коду для підтвердження особи.

Гнучка рольова модель доступу:

Лікар: має повноваження для перегляду, додавання, редагування та видалення записів про пацієнтів, зображено на рис.3.8, рис.3.9 та рис.3.10.

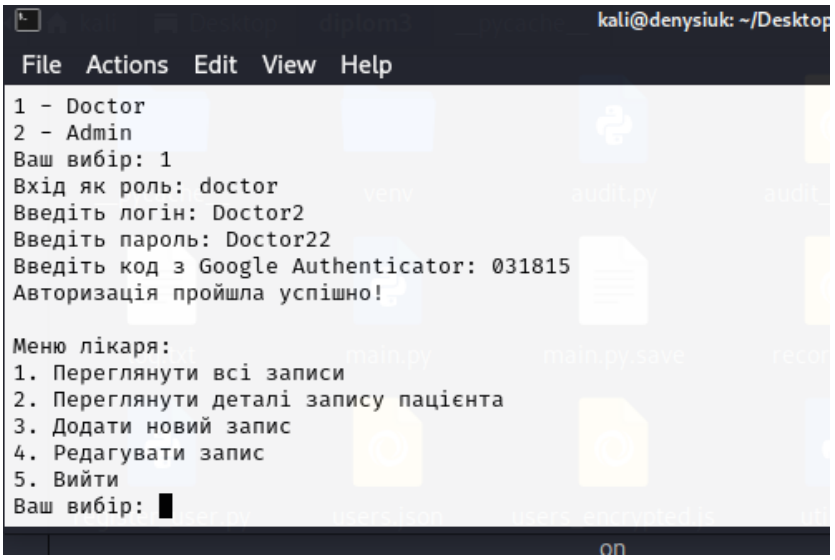

Рисунок 3.8. Вигляд меню під роллю лікаря

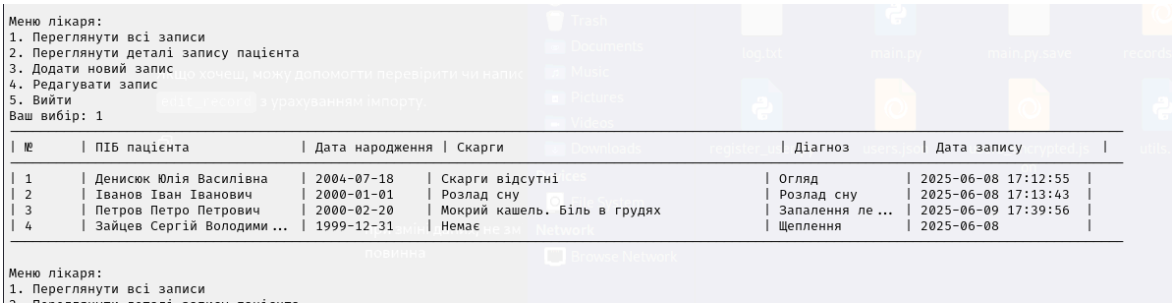

Рисунок 3.9. Перегляд записів пацієнта

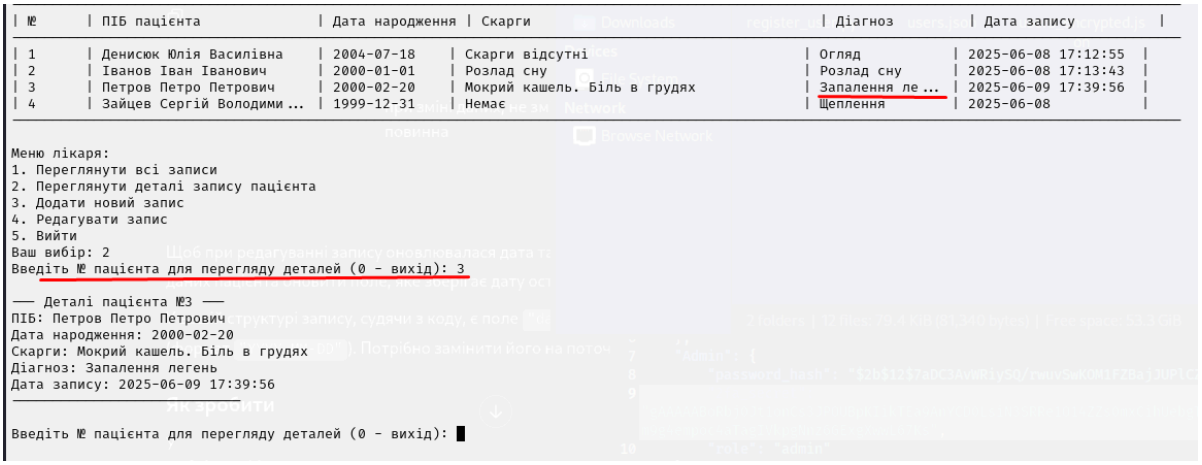

Рисунок 3.10. Перегляд деталей записів пацієнта

Адміністратор: володіє розширеними правами, включаючи управління користувачами(додавати, видаляти та переглядати), управління журналом записів(додавати, видаляти, редагувати, переглядати детальніше), зображено на рис.3.11.

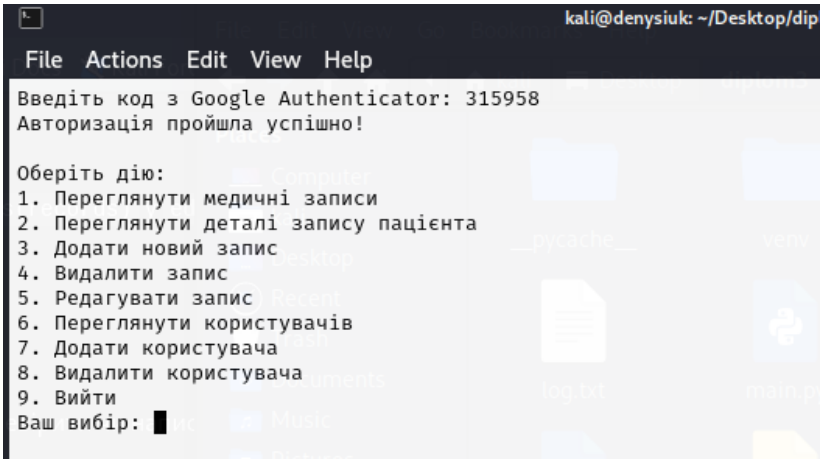

Рисунок 3.11. Меню дій адміністратора

Проактивний захист: Автоматичне блокування облікового запису після трьох невдалих спроб входу для запобігання атакам методом перебору (брутфорс). Кількість залишених спроб зображено на рис.3.12.

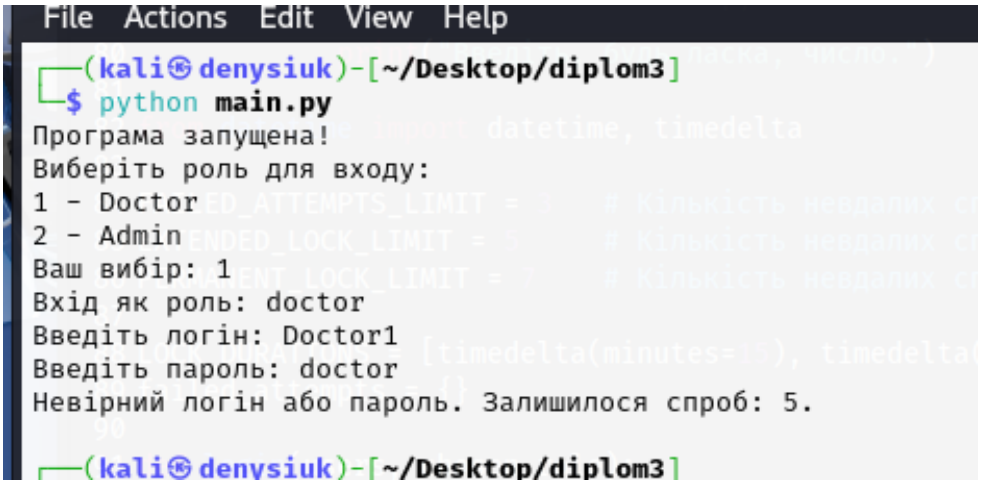

Рисунок 3.12. Зображення кількості невдалих спроб

Для блокування акаунта необхідні такі умови щодо кількісті невдалих спроб введення паролю:

- Якщо 3, то блокування облікових записів на 15 хв, приклад зображено на рис.3.13;

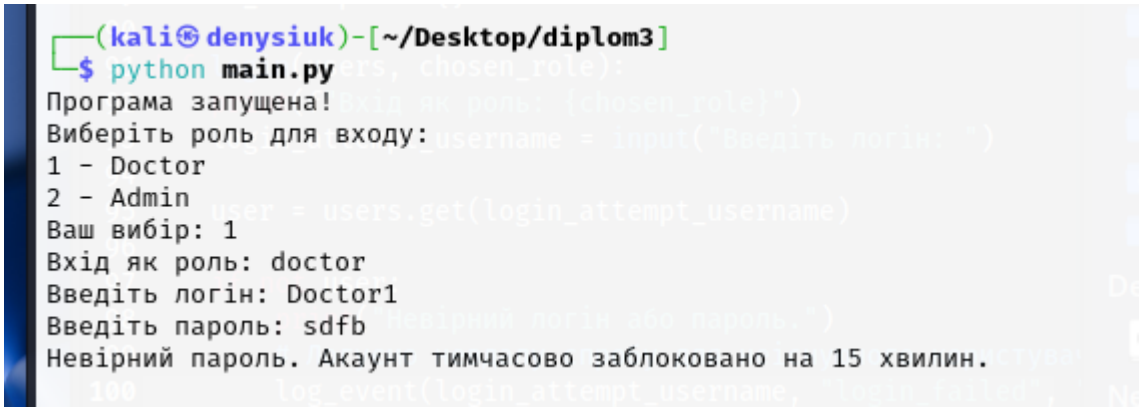

Рисунок 3.13. Зображення блокування акаунта на 15 хвилин

- Якщо 5, то блокування облікових записів на 30 хв;
- Якщо 7, то повне блокування облікового запису.

Встановлення застосунку: Користувач встановлює на свій смартфон додаток Google Authenticator, Authy, FreeOTP або будь-який інший, що підтримує стандарт TOTP.

Генерація секретного ключа: У момент створення облікового запису система генерує унікальний секретний ключ (2fa_secret) у форматі base32, який є основою для генерації OTP-кодів.

Створення QR-коду: Система формує спеціальний URL та перетворює його на QR-код за допомогою бібліотеки qrcode.

Сканування та синхронізація: Користувач сканує QR-код через мобільний додаток, після чого обліковий запис MIC миттєво додається і починає генерувати 6-значні коди, що оновлюються кожні 30 секунд.

Верифікація: Під час кожного наступного входу, після введення пароля, система запитує актуальний OTP-код з додатка, завершуючи процес безпечної автентифікації. При неправильному введені OTP-коду, система також блокує профіль користувача, зображено на рис.3.14.

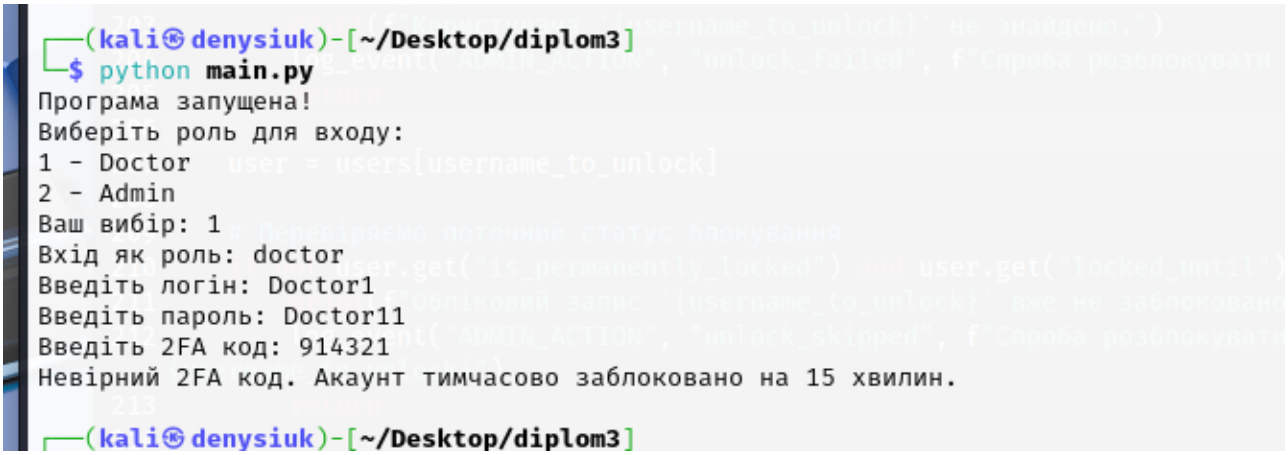

Рисунок 3.14. Зображено блокування акаунта через неправильний 2FA код

Окрім блокування акаунта через неправильно введений пароль, також реалізовано захист акаунта від неправильно введеного 2FA коду. Приклад блокування акаунта через неправильний 2FA код зображено на рис. 3.14. Цей додатковий рівень захисту запобігає спробам підбору або компрометації облікових даних навіть у випадку, якщо основний пароль користувача було скомпрометовано.

Для підвищення зручності користувачів було реалізовано можливість запуску програми через ярлик на робочому столі. Приклад ярлика програми показано на рис. 3.15.

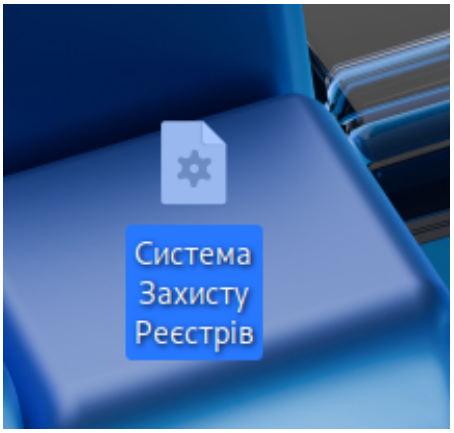

Рисунок 3.15. Ярлик програми

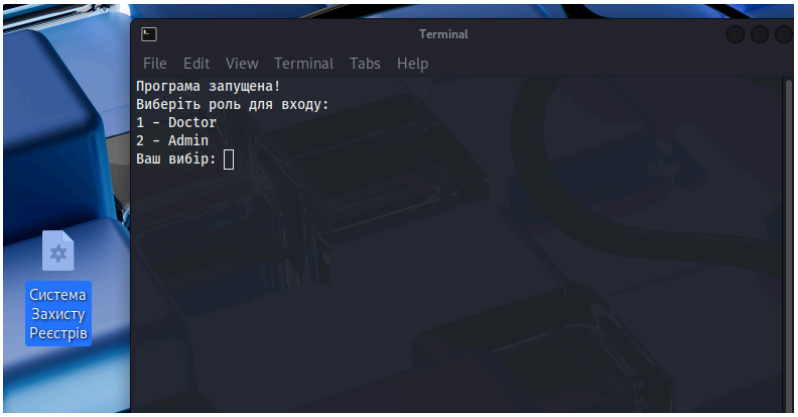

Рисунок 3.16. Запуск програми через ярлик

Таким чином, окрім традиційного запуску програми через термінал за допомогою команди python main.py, користувачі мають можливість запустити її через створений ярлик, що значно спрощує доступ до системи. Приклад запуску програми через ярлик зображений на рис. 3.16. Це робить використання системи більш інтуїтивним та швидким, особливо для менш технічно обізнаних користувачів.

3.2 Забезпечення інформаційної безпеки

Інформаційна безпека в медичних інформаційних системах (МІС) має критичне значення, адже вона забезпечує захист конфіденційних даних

пацієнтів, гарантує їх цілісність та недоступність для несанкціонованих осіб. Даний проєкт був завантажений на github.com. Переглянути можна за посиланням: https://github.com/Yuliia31/prototype-protecting-medical-registers.git.

3.2.1 Механізми автентифікації та авторизації

Для забезпечення безпеки системи необхідно застосовувати багаторівневий підхід до захисту даних:

Хешування паролів: Використання bcrypt з унікальною "сіллю" для кожного запису.

Динамічна автентифікація: Перевірка OTP-кодів, секретні ключі для яких ніколи не передаються у відкритому вигляді після реєстрації.

Одноразова передача секрету: QR-код для налаштування 2FA надається користувачу лише один раз, що мінімізує ризик його перехоплення.

Ізоляція даних: Прямий доступ до файлу з реєстрами пацієнтів блокується; взаємодія можлива виключно через програмний інтерфейс після повної авторизації.

Розмежування повноважень: Чітка рольова модель не дозволяє користувачам виконувати дії, що виходять за межі їхніх компетенцій.

3.2.2 Захист персональних даних у системі

У даному проекті було реалізовано важливі кроки захисту прототипу реєстру медичної інформаційної системи. Більше за все було приділено увагу забезпечення конфіденційності за допомогою таких механізмів:

Контроль доступу на основі ролей: Гарантує, що лише авторизований медичний персонал може переглядати та змінювати записи пацієнтів.

Аудит дій: Детальне логування всіх операцій з даними та спроб доступу дозволяє відстежувати будь-які аномалії.

Захист від атак: Механізм тимчасового блокування ефективно протидіє спробам підбору паролів.

Цілісність даних: Система підтримує створення резервних копій для швидкого відновлення у випадку збоїв.

Висновки за розділом 3

У було детально розглянуто розробку механізмів для забезпечення захищеного доступу до реєстрів медичних інформаційних систем (МІС). Враховуючи критичне значення конфіденційності та цілісності медичної інформації в умовах постійного зростання кіберзагроз, основним завданням було створення надійного програмного бар'єру для захисту електронних реєстрів пацієнтів.

В рамках цього розділу було успішно спроєктовано та реалізовано прототип системи, розроблений на мові Python з використанням перевірених бібліотек, таких як bcrypt для хешування паролів, pyotp для двофакторної автентифікації та qrcode для зручної інтеграції. Модульна архітектура системи забезпечила її гнучкість, масштабованість та легкість у підтримці.

Ключовими результатами реалізації є:

Надійний Модуль Первинної Автентифікації: Забезпечує безпечну верифікацію логін/пароль з використанням криптографічного хешування bcrypt та унікальної "солі", що унеможливлює зворотне відновлення паролів.

Ефективний Модуль Двофакторної Автентифікації (2FA): Реалізовано на основі алгоритму TOTP, що забезпечує додатковий рівень захисту та сумісність з популярними мобільними автентифікаторами, такими як Google Authenticator, завдяки генерації QR-кодів.

Гнучкий Модуль Управління Реєстром Пацієнтів: Захищене сховище даних, доступ до якого контролюється ролями користувачів, дозволяючи лікарям керувати записами, а адміністраторам – повноваженнями.

Модуль Аудиту та Логування: Забезпечує повну підзвітність, фіксуючи всі значущі події системи для аналізу безпеки та виявлення аномалій.

Розширені Функції Адміністрування Користувачів: Надано адміністраторам повний контроль над життєвим циклом облікових записів.

Крім того, було реалізовано комплексний набір функцій для забезпечення всебічного захисту: від безпечної реєстрації та криптографічного захисту паролів до проактивного блокування акаунтів від атак брутфорс, що включає тимчасові (на 15 та 30 хвилин) та повне блокування при перевищенні ліміту невдалих спроб введення пароля та OTP-коду. Чітка рольова модель та ізоляція даних додатково підсилюють інформаційну безпеку.

Загалом, розроблений прототип демонструє ефективний багаторівневий підхід до захисту даних у МІС, забезпечуючи конфіденційність, цілісність та доступність лише для авторизованого персоналу. Його функціональні можливості та архітектурні рішення підкреслюють готовність до подальшої масштабованості та інтеграції у реальні медичні інформаційні системи.

Варто зазначити, що реалізація проєкту була успішно завантажена на платформу GitHub, що забезпечує його доступність для перегляду та подальшого розвитку за посиланням: https://github.com/Yuliia31/prototype-protecting-medical-registers.git.

ВИСНОВКИ

Медичні інформаційні системи (МІС) є ключовим елементом сучасної охорони здоров'я, забезпечуючи ефективне зберігання, обробку та доступ до медичних даних. Вони сприяють підвищенню якості медичних послуг, поліпшенню моніторингу здоров'я населення та оптимізації управлінських процесів у медичних закладах. Водночас, для їх успішного функціонування вкрай важливо забезпечити високий рівень інформаційної безпеки, щоб захистити конфіденційність, цілісність та доступність медичних даних.

Аналіз архітектури МІС показав, що такі системи складаються з кількох основних компонентів, включаючи роботу з реєстрами пацієнтів, лікарів та медичних записів. Вибір технологій для зберігання інформації (зокрема, використання SQLite у розглянутому проєкті) забезпечує зручну та ефективну роботу з медичними даними. Водночас реєстри вимагають дотримання нормативно-правових вимог і організаційної дисципліни для підтримки актуальності і точності інформації.

Особливу увагу приділено питанням захисту інформації в медичних системах. Забезпечення автентифікації та авторизації користувачів, а також розмежування прав доступу є базовими механізмами контролю безпеки. Використання шифрування та інших технічних засобів дозволяє зменшити ризики несанкціонованого доступу або витоку даних. Водночас, не менш важливим є людський фактор — підготовка персоналу, дотримання політик безпеки і регулярний аудит системи.

Проаналізований приклад реалізації медичної інформаційної системи на основі відкритого проєкту демонструє базові можливості і механізми захисту, які можуть бути впроваджені у практичних рішеннях. Проте існують значні перспективи для подальшого вдосконалення систем, зокрема шляхом впровадження багатофакторної автентифікації, сучасних протоколів

шифрування, а також адаптації до вимог національного та міжнародного законодавства, таких як GDPR та стандарти ISO/IEC 27001.

Отже, для забезпечення безпеки медичних інформаційних систем необхідний комплексний підхід, який включає технічні, організаційні та правові заходи. Такий підхід гарантує захист персональних медичних даних, підвищує довіру користувачів і сприяє сталому розвитку охорони здоров’я на основі сучасних інформаційних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Health Organization. Health statistics and information systems. https://www.who.int/data/gho — доступ 2024.

2. Коваленко, І. В. Медичні інформаційні системи: структура, типи, функції / І. В. Коваленко // Вісник медичної інформатики. — 2021. — Вип. 3. — С. 45-52.

3. Петрова, О. М. Особливості захисту електронних медичних записів / О. М. Петрова // Інформаційна безпека та захист інформації. — 2022. — № 2. — С. 78-84.

4. Державні стандарти України: ДСТУ ISO/IEC 27001:2023 «Інформаційна безпека, кібербезпека та захист конфіденційності. Системи керування інформаційною безпекою.» / Національний орган стандартизації України. — Київ, 2016.

5. Gartner IT Glossary. Electronic Health Record (EHR). https://www.gartner.com/en — доступ 2024.

6. Міністерство охорони здоров’я України. Національна стратегія розвитку електронної охорони здоров’я на період 2020-2025 рр. — Київ, 2020.

7. Розумний, В. А., & Сидоренко, І. П. Системи телемедицини: принципи, технології, застосування / В. А. Розумний, І. П. Сидоренко // Журнал медичної інформатики. — 2021. — Т. 8, № 1. — С. 23-34.

8. Sittig, D. F., & Singh, H. A socio-technical approach to preventing, mitigating, and recovering from ransomware attacks on healthcare delivery organizations. BMJ Health & Care Informatics, 26(1), 2020. DOI:10.1136/bmjhci-2020-100156.

9. Медична інформаційна система Helsi. https://helsi.pro/.

10. Медичний стандарт ISO/HL7. https://www.siemens-healthineers.com/ua/services/it-standards/hl7.

11. Створення ярлика https://losst.pro/dobavlenie-yarlyka-v-ubuntu.

12. Статистика по кіберзлочинам та захисту https://www.statista.com/markets/424/topic/1065/cyber-crime-security/#overview.

13. Інструкція по налаштування Google Authenticator https://support.google.com/accounts/answer/1066447?sjid=14195069477067073912-EU/.

14. https://www.embs.org/jbhi/past-special-issues/.

15. ЕСОЗ https://ehealth.gov.ua/.

16. НСЗУ https://nszu.gov.ua/.

17. Створення зображень https://sora.chatgpt.com/explore.

18. MIC Health24 https://blog.h24.ua/uk/shho-take-mis/.

19. Ismail Keshta, Ammar Odeh. Security and privacy of electronic health records: Concerns and challenges https://www.sciencedirect.com/science/article/pii/S1110866520301365.

20. https://www.igi-global.com/gateway/article/full-text-html/297076&riu=true.

ДОДАТОК А

Таблиця.1. Приклади інструментів та засобів для реалізації принципів інформаційної безпеки в МІС

Принцип	Мета	Приклади реалізації / інструментів
Конфіденційність	Запобігання доступу до даних сторонніми або неавторизованими особами	Розмежування доступу; шифрування даних; використання VPN; політики конфіденційності; аномізація/псевдонімізація
Цілісність	Захист від несанкціонованих змін або спотворення даних	Електронний журнал змін (audit trail); контроль версій; цифрові підписи; контроль цілісності БД; хешування даних
Доступність	Забезпечення постійного доступу до даних для уповноважених користувачів	Надійна IT-інфраструктура; кластери серверів; резервне копіювання; система аварійного відновлення (Disaster Recovery)
Аутентифікація	Ідентифікація користувачів	Використання логінів, паролів, багатофакторної аутентифікації (MFA), токенів, біометричної ідентифікації
Авторизація	Надання доступу до функцій згідно з повноваженнями користувача	Ролі користувачів у системі; політики доступу (RBAC — Role-Based Access Control)

Продовження табл.1

Принцип	Мета	Приклади реалізації / інструментів
Відповідальність	Можливість простежити дії користувачів у системі МІС	Журнали активності користувачів МІС (хто, коли, що робив). Система аудиту (audit trail) для відстеження всіх змін та доступу до медичних даних. Налаштування детального логування подій у МІС та на серверах. Регулярний аналіз логів безпеки та формування звітів.
Дотримання законодавства	Відповідність національним та	Впровадження політик відповідно до GDPR, ISO/IEC 27001, Законів

	міжнародним нормам	України про захист інформації та персональних даних

ДОДАТОК Б

Лістинг коду main.py:

```
import json
import bcrypt
import pyotp
from datetime import datetime, timedelta
import re
from cryptography.fernet import Fernet
from audit import log_event
from register_user import register_user
from records_menu import delete_user_flow
from utils import load_users, save_users, load_records, save_records, USERS_FILE, RECORDS_FILE
from records_menu import edit_record

# Ключ для шифрування
key = b'3qw2__Ky1g2cakycxyhlbuU2mo45J-TZL5F2oTGOaF8='
fernet = Fernet(key)

# Функції перевірки паролю і 2FA
def verify_password(password, hashed):
    return bcrypt.checkpw(password.encode(), hashed.encode())

def verify_2fa(secret, code):
    totp = pyotp.TOTP(secret)
    return totp.verify(code)

# Функції для роботи з українським текстом
def check_ukrainian_text(text):
```

```
    pattern = re.compile(r"^[А-Яа-яЇїІіЄєҐґ'’\s\-]+$")
    return bool(pattern.match(text))

def truncate(text, max_len):
    if len(text) <= max_len:
        return text
    return text[:max_len-3] + "..."

def print_records_table(records):
    if not records:
        print("Медичних записів немає.")
        return

     headers = ["№", "ПІБ пацієнта", "Дата народження", "Скарги", "Діагноз", "Дата запису"]
    col_widths = [6, 25, 13, 40, 15, 20]
    row_format = "| {:<6} | {:<25} | {:<13} | {:<40} | {:<15} | {:<20} |"
    total_width = sum(col_widths) + 3 * len(col_widths) + 1 + 4

    print("-" * total_width)
    print(row_format.format(*headers))
    print("-" * total_width)

    for i, rec in enumerate(records, 1):
        print(row_format.format(
            i,
            truncate(rec['patient_name'], col_widths[1]),
            rec['birth_date'],
            truncate(rec['complaints'], col_widths[3]),
            truncate(rec['diagnosis'], col_widths[4]),
            rec['date']
```

```
        ))

    print("-" * total_width)

def show_record_detail(records):
    while True:
        try:
            num = int(input("Введіть № пацієнта для перегляду деталей (0 - вихід): "))
            if num == 0:
                break
            if 1 <= num <= len(records):
                rec = records[num - 1]
                print("\n--- Деталі пацієнта №{} ---".format(num))
                print(f"ПІБ: {rec['patient_name']}")
                print(f"Дата народження: {rec['birth_date']}")
                print(f"Скарги: {rec['complaints']}")
                print(f"Діагноз: {rec['diagnosis']}")
                print(f"Дата запису: {rec['date']}")
                print("----------------------------\n")
                    log_event("system", "view_record_detail", f"Перегляд деталей пацієнта №{num}: {rec['patient_name']}")
            else:
                print("Невірний номер. Спробуйте ще.")
        except ValueError:
            print("Введіть, будь ласка, число.")

from datetime import datetime, timedelta

FAILED_ATTEMPTS_LIMIT = 3      # Кількість невдалих спроб до першого тимчасового блокування (15 хвилин)
EXTENDED_LOCK_LIMIT = 5        # Кількість невдалих спроб до розширеного
```

```
тимчасового блокування (30 хвилин)
PERMANENT_LOCK_LIMIT = 7        # Кількість невдалих спроб до повного блокування облікового запису

LOCK_DURATIONS = [timedelta(minutes=15), timedelta(minutes=30)]

def login(users, chosen_role):
    print(f"Вхід як роль: {chosen_role}")
    login_attempt_username = input("Введіть логін: ")

    user = users.get(login_attempt_username)

    if not user:
        print("Невірний логін або пароль.")
            log_event(login_attempt_username, "login_failed", "Користувача не знайдено (або невірний пароль)")
        return None

    # 1. Перевірка на постійне блокування
    if user.get("is_permanently_locked"):
        print("Акаунт повністю заблоковано. Зверніться до адміністратора.")
                log_event(login_attempt_username, "login_blocked", "Акаунт повністю заблоковано")
        return None

    # 2. Перевірка на тимчасове блокування
    locked_until_str = user.get("locked_until")
    if locked_until_str:
        locked_until = datetime.fromisoformat(locked_until_str)
        if datetime.now() < locked_until:
            print(f"Акаунт тимчасово заблоковано до {locked_until.strftime('%Y-%m-%d
```

```
%H:%M:%S')}")
                log_event(login_attempt_username, "login_blocked", f"Акаунт тимчасово заблоковано до {locked_until}")
            return None
        else:
            # Час блокування минув, скидаємо лише статус блокування.
            # Лічильник невдалих спроб залишається для прогресивного блокування.
            user["locked_until"] = None
            save_users(users) # Зберігаємо зміни

    # 3. Перевірка ролі (якщо користувач існує і не заблокований)
    if user["role"] != chosen_role:
        print(f"У вас немає доступу з роллю '{chosen_role}'.")
            log_event(login_attempt_username, "login_failed", f"Немає доступу для ролі {chosen_role}")
        return None

    password = input("Введіть пароль: ")

    # 4. Перевірка пароля
    if not verify_password(password, user["password_hash"]):
        user["failed_login_attempts"] = user.get("failed_login_attempts", 0) + 1

            # Логіка блокування на основі лічильника (від найсуворішого до найменш суворого)
        if user["failed_login_attempts"] >= PERMANENT_LOCK_LIMIT:
            user["is_permanently_locked"] = True
                log_event(login_attempt_username, "account_locked_permanently", f"Акаунт заблоковано назавжди після {user['failed_login_attempts']} спроб")
            print("Акаунт повністю заблоковано через багато невдалих спроб.")
        elif user["failed_login_attempts"] >= EXTENDED_LOCK_LIMIT:
```

```
            # Застосовуємо розширене блокування, якщо акаунт наразі не заблокований
              if user["locked_until"] is None: # Перевірка, щоб не перезаписувати активне блокування
                duration = LOCK_DURATIONS[1] # 30 хвилин
                user["locked_until"] = (datetime.now() + duration).isoformat()
                  log_event(login_attempt_username, "account_locked", f"Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин (розширене блокування)")
                                print(f"Невірний пароль. Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин.")
          elif user["failed_login_attempts"] >= FAILED_ATTEMPTS_LIMIT:
                 # Застосовуємо перше тимчасове блокування, якщо акаунт наразі не заблокований
              if user["locked_until"] is None: # Перевірка, щоб не перезаписувати активне блокування
                duration = LOCK_DURATIONS[0] # 15 хвилин
                user["locked_until"] = (datetime.now() + duration).isoformat()
                  log_event(login_attempt_username, "account_locked", f"Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин")
                                print(f"Невірний пароль. Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин.")
          else:
             log_event(login_attempt_username, "login_failed", "Невірний пароль")
                                 print(f"Невірний логін або пароль. Залишилося спроб: {PERMANENT_LOCK_LIMIT - user['failed_login_attempts']}.")

          save_users(users) # Зберігаємо зміни після кожної невдалої спроби
          return None

       # 5. Перевірка 2FA (якщо є)
       if user.get("2fa_secret"):
          code_2fa = input("Введіть 2FA код: ")
```

```
        if not verify_2fa(user["2fa_secret"], code_2fa):
            user["failed_login_attempts"] = user.get("failed_login_attempts", 0) + 1

            if user["failed_login_attempts"] >= PERMANENT_LOCK_LIMIT:
                user["is_permanently_locked"] = True
                log_event(login_attempt_username, "account_locked_permanently", f"Акаунт заблоковано назавжди після {user['failed_login_attempts']} спроб 2FA")
                print("Невірний 2FA код. Акаунт повністю заблоковано через багато невдалих спроб.")
            elif user["failed_login_attempts"] >= EXTENDED_LOCK_LIMIT:
                if user["locked_until"] is None: # Перевірка, щоб не перезаписувати активне блокування
                    duration = LOCK_DURATIONS[1]
                    user["locked_until"] = (datetime.now() + duration).isoformat()
                    log_event(login_attempt_username, "account_locked", f"Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин (розширене блокування 2FA)")
                    print(f"Невірний 2FA код. Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин.")
            elif user["failed_login_attempts"] >= FAILED_ATTEMPTS_LIMIT:
                if user["locked_until"] is None: # Перевірка, щоб не перезаписувати активне блокування
                    duration = LOCK_DURATIONS[0]
                    user["locked_until"] = (datetime.now() + duration).isoformat()
                    log_event(login_attempt_username, "account_locked", f"Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин (2FA)")
                    print(f"Невірний 2FA код. Акаунт тимчасово заблоковано на {duration.seconds // 60} хвилин.")
            else:
                log_event(login_attempt_username, "login_failed", "Невірний 2FA код")
                print(f"Невірний 2FA код. Залишилося спроб:
```

```
{PERMANENT_LOCK_LIMIT - user['failed_login_attempts']}.")

            save_users(users) # Зберігаємо зміни після кожної невдалої спроби 2FA
            return None

    # 6. Успішний вхід (пароль та 2FA) — скидання всіх блокувань та лічильників
    user["failed_login_attempts"] = 0
    user["locked_until"] = None
    user["is_permanently_locked"] = False
    save_users(users) # Зберігаємо зміни після успішного входу

    log_event(login_attempt_username, "login_success", "Успішний вхід")
    return login_attempt_username

def unlock_user_account(users):

    print("\n--- Розблокування облікового запису ---")
        username_to_unlock = input("Введіть логін користувача, якого потрібно розблокувати: ")

    if username_to_unlock not in users:
        print(f"Користувача '{username_to_unlock}' не знайдено.")
            log_event("ADMIN_ACTION", "unlock_failed", f"Спроба розблокувати неіснуючого користувача: {username_to_unlock}")
        return

    user = users[username_to_unlock]

    # Перевіряємо поточний статус блокування
    if not user.get("is_permanently_locked") and user.get("locked_until") is None:
        print(f"Обліковий запис '{username_to_unlock}' вже не заблоковано.")
```

```
        log_event("ADMIN_ACTION", "unlock_skipped", f"Спроба розблокувати вже розблокований обліковий запис: {username_to_unlock}")
        return

    # Знімаємо всі статуси блокування та скидаємо лічильник
    user["is_permanently_locked"] = False
    user["locked_until"] = None
    user["failed_login_attempts"] = 0

    save_users(users)
    print(f"Обліковий запис '{username_to_unlock}' успішно розблоковано.")
     log_event("ADMIN_ACTION", "account_unlocked", f"Адміністратор розблокував обліковий запис: {username_to_unlock}")

def unlock_user(users, login):
    if login in users:
        users[login]["is_permanently_locked"] = False
        users[login]["locked_until"] = None
        failed_attempts[login] = 0
        log_event(login, "admin_unlock", "Акаунт розблоковано адміністратором")
        print(f"Користувача {login} розблоковано.")
    else:
        print("Користувача не знайдено.")



def show_menu(role):
    print("\nОберіть дію:")
    if role == "doctor":
        print("1. Переглянути медичні записи")
        print("2. Переглянути деталі запису пацієнта")
        print("3. Додати новий запис")
```

```
        print("4. Редагувати запис")
        print("5. Вийти")
    elif role == "admin":
        print("1. Переглянути медичні записи")
        print("2. Переглянути деталі запису пацієнта")
        print("3. Додати новий запис")
        print("4. Видалити запис")
        print("5. Редагувати запис")
        print("6. Переглянути користувачів")
        print("7. Додати користувача")
        print("8. Видалити користувача")
        print("9. Розблокувати користувача")
        print("10. Вийти")


def add_new_record(records, current_user=None):
    patient_name = input("Введіть ПІБ пацієнта (українською): ")
    if not check_ukrainian_text(patient_name):
        print("ПІБ має містити лише українські літери, пробіли або дефіси.")
        return

    while True:
        birth_date = input("Введіть дату народження (формат YYYY-MM-DD): ")
        try:
            datetime.strptime(birth_date, "%Y-%m-%d")
            break
        except ValueError:
            print("Невірна дата. Спробуйте ще раз. Наприклад: 1990-12-31")

    while True:
        complaints = input("Введіть скарги (українською): ")
```

```
        if check_ukrainian_text(complaints):
            break
        else:
                print("Скарги мають містити лише українські літери, пробіли, дефіси, крапки або коми.")

    while True:
        diagnosis = input("Введіть діагноз (українською): ")
        if check_ukrainian_text(diagnosis):
            break
        else:
            print("Діагноз має містити лише українські літери, пробіли, дефіси, крапки або коми.")

    date = datetime.now().strftime("%Y-%m-%d")
    records.append({
        "patient_name": patient_name,
        "birth_date": birth_date,
        "complaints": complaints,
        "diagnosis": diagnosis,
        "date": date
    })
    save_records(records)
    print("Запис успішно додано.")
    if current_user:
                log_event(current_user, "record_added", f"Додано запис пацієнта: {patient_name}")

def doctor_actions(records, login_user):
    log_event(login_user, "session_start", "Лікар увійшов у систему")
    while True:
```

```
        print("\nМеню лікаря:")
        print("1. Переглянути всі записи")
        print("2. Переглянути деталі запису пацієнта")
        print("3. Додати новий запис")
        print("4. Редагувати запис")
        print("5. Вийти")

        choice = input("Ваш вибір: ")

        if choice == "1":
            print_records_table(records)
            log_event(login_user, "view_records", "Перегляд усіх записів")
        elif choice == "2":
            show_record_detail(records)
        elif choice == "3":
            add_new_record(records, login_user)
        elif choice == "4":
            edit_record(records, login_user)
        elif choice == "5":
            log_event(login_user, "logout", "Користувач вийшов із системи")
            print("Вихід із системи.")
            break
        else:
            print("Невірний вибір.")

def admin_actions(users, records, login_user):
    while True:
        show_menu("admin") # Показуємо оновлене меню адміністратора
        choice = input("Ваш вибір: ")

        if choice == "1":
```

```
            print_records_table(records)
        elif choice == "2":
            show_record_detail(records)
        elif choice == "3":
            add_new_record(records, login_user)
        elif choice == "4":
            if not records:
                print("Записів немає.")
            else:
                print_records_table(records)
                to_del = input("Введіть номер запису для видалення: ")
                if to_del.isdigit() and 1 <= int(to_del) <= len(records):
                    deleted = records.pop(int(to_del) - 1)
                    save_records(records)
                    print(f"Запис '{deleted['patient_name']}' видалено.")
                        log_event(login_user, "record_deleted", f"Видалено запис пацієнта:
{deleted['patient_name']}")
                else:
                    print("Невірний номер запису.")
        elif choice == "5":
            edit_record(records, login_user)
        elif choice == "6":
            print("\n--- Користувачі ---")
            for username, data in users.items():
                print(f"Логін: {username}, Роль: {data['role']}")
        elif choice == "7":
            add_user(users, login_user)
            users = load_users() # Оновлюємо список користувачів після додавання
        elif choice == "8":
            delete_user_flow(users, login_user)
            users = load_users() # Оновлюємо список користувачів після видалення
```

```
        elif choice == "9":
            # Виклик функції для розблокування облікового запису
            unlock_user_account(users) # Передаємо словник users до функції
                users = load_users() # Оновлюємо список користувачів після потенційних
змін
        elif choice == "10": # Новий номер для виходу
            print("Вихід із системи.")
            log_event(login_user, "logout", "Користувач вийшов із системи")
            break
        else:
            print("Невірний вибір.")



def add_user(users, current_user):
    print("\nДодавання нового користувача:")
    register_user()
    log_event(current_user, "user_added", "Додано нового користувача")
    print("Користувача успішно додано!")

def main():
    print("Програма запущена!")

    users = load_users()
    records = load_records()

    print("Виберіть роль для входу:")
    print("1 - Doctor")
    print("2 - Admin")
    role_choice = input("Ваш вибір: ")

    if role_choice == "1":
```

```
        role = "doctor"
    elif role_choice == "2":
        role = "admin"
    else:
        print("Невірний вибір ролі.")
        return

    login_user = login(users, role)
    if not login_user:
        return

    if role == "doctor":
        doctor_actions(records, login_user)
    elif role == "admin":
        admin_actions(users, records, login_user)

if __name__ == "__main__":
    main()
```

ДОДАТОК В

Лістинг коду audit.py:

```
#audit.py
import logging
import datetime
import json
import os

AUDIT_LOG_FILE_JSON = "audit_log.json"
AUDIT_LOG_FILE_TXT = "log.txt"

# Налаштування логування в текстовий файл log.txt
logging.basicConfig(
    filename=AUDIT_LOG_FILE_TXT,
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

# Кеш для логів у пам'яті
logs_cache = []

# Завантаження логів з JSON файлу (якщо існує)
def load_logs():
    global logs_cache
    if os.path.exists(AUDIT_LOG_FILE_JSON):
        try:
            with open(AUDIT_LOG_FILE_JSON, "r", encoding="utf-8") as f:
                logs_cache = json.load(f)
        except json.JSONDecodeError as e:
            logging.error(f"Error reading JSON log file: {e}")
            logs_cache = []
        except Exception as e:
            logging.error(f"Unexpected error when reading the log file: {e}")
            logs_cache = []

# Збереження логів в JSON файл
def save_logs():
```

```
    try:
        with open(AUDIT_LOG_FILE_JSON, "w", encoding="utf-8") as f:
            json.dump(logs_cache, f, indent=4, ensure_ascii=False)
    except Exception as e:
        logging.error(f"Error saving log file: {e}")

# Логування події
def log_event(username, event_type, description):
    timestamp = datetime.datetime.now().isoformat()
    event = {
        "timestamp": timestamp,
        "user": username,
        "event_type": event_type,
        "description": description
    }

    logs_cache.append(event)

    # Запис у текстовий лог-файл (log.txt)
    logging.info(f"{event_type} - {description} by {username}")

    # Збереження в JSON
    save_logs()

# Ініціалізація логів при старті програми
load_logs()
```

ДОДАТОК Г

Лістинг коду records_menu.py:

```
import json
from datetime import datetime
from audit import log_event
from utils import save_users, USERS_FILE, check_ukrainian_text, save_records, load_records
from audit import log_event

USERS_FILE = "users.json"
RECORDS_FILE = "records.json"
# records_menu.py
def delete_user_flow(users, current_user):
    username = input("Введіть логін користувача для видалення (або Enter для виходу): ")
    if username == "":
        print("Відмінено.")
        return

    if username not in users:
        print("Користувача з таким логіном не знайдено.")
        return

    if username == current_user:
        print("Ви не можете видалити себе.")
        return

    del users[username]
    save_users(users)
```

```
    print(f"Користувача '{username}' успішно видалено.")
    log_event(current_user, "user_deleted", f"Видалено користувача: {username}")


# Функція для виведення таблиці записів
def print_records_table(records):
    if not records:
        print("Медичних записів немає.")
        return

        headers = ["№", "ПІБ пацієнта", "Дата народження", "Скарги", "Діагноз",
"Дата запису"]
    col_widths = [6, 25, 13, 40, 15, 20]
    row_format = "| {:<6} | {:<25} | {:<13} | {:<40} | {:<15} | {:<20} |"
    total_width = sum(col_widths) + 3 * len(col_widths) + 1 + 4

    print("-" * total_width)
    print(row_format.format(*headers))
    print("-" * total_width)

    for i, rec in enumerate(records, 1):
        print(row_format.format(
            i,
            rec['patient_name'][:col_widths[1]],
            rec['birth_date'],
            rec['complaints'][:col_widths[3]],
            rec['diagnosis'][:col_widths[4]],
            rec['date']
        ))
```

```
    print("-" * total_width)

def edit_record(records, login_user):
    if not records:
        print("Записів немає для редагування.")
        return
    print_records_table(records)
    try:
        num = int(input("Введіть номер запису для редагування (0 - вихід): "))
        if num == 0:
            return
        if not (1 <= num <= len(records)):
            print("Невірний номер запису.")
            return
    except ValueError:
        print("Введіть, будь ласка, число.")
        return

    rec = records[num - 1]

    print(f"Редагування запису пацієнта: {rec['patient_name']}")
    new_name = input(f"ПІБ пацієнта [{rec['patient_name']}]: ")
    if new_name.strip():
        if check_ukrainian_text(new_name):
            rec['patient_name'] = new_name
        else:
                print("ПІБ має містити лише українські літери, пробіли або дефіси.
Залишено старе значення.")

    new_birth_date = input(f"Дата народження [{rec['birth_date']}]: ")
```

```
    if new_birth_date.strip():
        try:
            datetime.strptime(new_birth_date, "%Y-%m-%d")
            rec['birth_date'] = new_birth_date
        except ValueError:
            print("Невірна дата. Залишено старе значення.")

    new_complaints = input(f"Скарги [{rec['complaints']}]: ")
    if new_complaints.strip():
        if check_ukrainian_text(new_complaints):
            rec['complaints'] = new_complaints
        else:
            print("Скарги мають містити лише українські літери, пробіли або дефіси.
Залишено старе значення.")

    new_diagnosis = input(f"Діагноз [{rec['diagnosis']}]: ")
    if new_diagnosis.strip():
        if check_ukrainian_text(new_diagnosis):
            rec['diagnosis'] = new_diagnosis
        else:
            print("Діагноз має містити лише українські літери, пробіли або дефіси.
Залишено старе значення.")

    rec['date'] = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    save_records(records)
    print("Запис успішно оновлено.")
            log_event(login_user, "record_edited", f"Редаговано запис пацієнта:
{rec['patient_name']}")
```

```
# Функція для перегляду деталей конкретного запису
def show_record_detail(records):
    while True:
        try:
            num = int(input("Введіть № пацієнта для перегляду деталей (0 - вихід): "))
            if num == 0:
                break
            if 1 <= num <= len(records):
                rec = records[num - 1]
                print("\n--- Деталі пацієнта №{} ---".format(num))
                print(f"ПІБ: {rec['patient_name']}")
                print(f"Дата народження: {rec['birth_date']}")
                print(f"Скарги: {rec['complaints']}")
                print(f"Діагноз: {rec['diagnosis']}")
                print(f"Дата запису: {rec['date']}")
                print("----------------------------\n")
            else:
                print("Невірний номер. Спробуйте ще.")
        except ValueError:
            print("Введіть, будь ласка, число.")

# Функція для видалення записів
def delete_record(records):
    while True:
        print("\nМеню видалення запису:")
        if not records:
            print("Медичних записів немає.")
            return
        print_records_table(records)
```

```
        to_del = input("Введіть номер запису для видалення: ")
        if to_del.isdigit() and 1 <= int(to_del) <= len(records):
            deleted = records.pop(int(to_del) - 1)
            save_records(records)
            print(f"Запис '{deleted['patient_name']}' видалено.")
            return deleted
        else:
            print("Невірний номер запису. Спробуйте ще.")

# Функція для додавання нового запису
def add_record(records, current_user=None):
    patient_name = input("Введіть ПІБ пацієнта (українською): ")
    if not check_ukrainian_text(patient_name):
        print("ПІБ має містити лише українські літери, пробіли або дефіси.")
        return

    while True:
        birth_date = input("Введіть дату народження (формат YYYY-MM-DD): ")
        try:
            datetime.strptime(birth_date, "%Y-%m-%d")
            break
        except ValueError:
            print("Невірна дата. Спробуйте ще раз. Наприклад: 1990-12-31")

    while True:
        complaints = input("Введіть скарги (українською): ")
        if check_ukrainian_text(complaints):
            break
        else:
                print("Скарги мають містити лише українські літери, пробіли, дефіси,
```

```
крапки або коми.")

    while True:
        diagnosis = input("Введіть діагноз (українською): ")
        if check_ukrainian_text(diagnosis):
            break
        else:
                print("Діагноз має містити лише українські літери, пробіли, дефіси, крапки або коми.")

    date = datetime.now().strftime("%Y-%m-%d")
    records.append({
        "patient_name": patient_name,
        "birth_date": birth_date,
        "complaints": complaints,
        "diagnosis": diagnosis,
        "date": date
    })
    save_records(records)
    print("Запис успішно додано.")
    if current_user:
                log_event(current_user, "record_added", f"Додано запис пацієнта: {patient_name}")

# Функція для взаємодії з користувачем в меню медичних записів
def records_menu(records, role, current_user=None):
    while True:
        if role == "doctor":
            print("\nМеню лікаря:")
            print("1. Переглянути всі записи")
```

```
        print("2. Переглянути деталі запису пацієнта")
        print("3. Додати новий запис")
        print("4. Вийти")
        choice = input("Ваш вибір: ")
        if choice == "1":
            print_records_table(records)
        elif choice == "2":
            show_record_detail(records)
        elif choice == "3":
            add_record(records, current_user)
        elif choice == "4":
            print("Вихід із системи.")
            break
        else:
            print("Невірний вибір.")

    elif role == "admin":
        print("\nМеню адміністратора:")
        print("1. Переглянути всі записи")
        print("2. Переглянути деталі запису пацієнта")
        print("3. Додати новий запис")
        print("4. Видалити запис")
        print("5. Вийти")
        choice = input("Ваш вибір: ")
        if choice == "1":
            print_records_table(records)
        elif choice == "2":
            show_record_detail(records)
        elif choice == "3":
            add_record(records, current_user)
```

```
        elif choice == "4":
            deleted_record = delete_record(records)
            if deleted_record:
                log_event(current_user, "record_deleted", f"Видалено запис пацієнта: {deleted_record['patient_name']}")
        elif choice == "5":
            print("Вихід із системи.")
            break
        else:
            print("Невірний вибір.")
```

ДОДАТОК Д

Лістинг коду register_user.py:

```
#register_user.py
import bcrypt
import pyotp
import json
import os
from utils import load_users, save_users, check_password_strength

USERS_FILE = "users.json"
LOG_FILE = "log.txt"

def check_password_strength(password):
    """Перевірка на мінімальну складність паролю (наприклад, мінімум 8 символів, цифра, велика і мала літера)"""
    if len(password) < 8:
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char.islower() for char in password):
        return False
    if not any(char.isupper() for char in password):
        return False
    return True

def load_users():
    """Завантаження користувачів з файлу JSON"""
    if os.path.exists(USERS_FILE):
```

```
        try:
            with open(USERS_FILE, "r", encoding="utf-8") as f:
                return json.load(f)
        except json.JSONDecodeError:
                print("Помилка при читанні файлу користувачів. Файл може бути пошкоджений.")
            return {}
        except Exception as e:
            print(f"Не вдалося завантажити користувачів: {e}")
            return {}
    return {}

def save_users(users):
    """Збереження користувачів у файл JSON"""
    try:
        with open(USERS_FILE, "w", encoding="utf-8") as f:
            json.dump(users, f, indent=4)

    except Exception as e:
        print(f"Не вдалося зберегти користувачів: {e}")

def log_action(message):
    """Проста функція логування для демонстрації."""
    with open("audit.log", "a", encoding="utf-8") as log_file:
                log_file.write(f"[{datetime.now().strftime('%Y-%m-%d  %H:%M:%S')}] {message}\n")

def register_user():
    # Завантажуємо наявних користувачів
    users = load_users()
```

```
    # Введення логіну
    while True:
        username = input("Введіть логін нового користувача: ")
        if username in users:
            print("Цей логін вже існує. Спробуйте інший.")
        else:
            break

    # Введення паролю
    while True:
        password = input("Введіть пароль: ")
        if check_password_strength(password):
            break
        else:
             print("Пароль має бути не менше 8 символів і містити хоча б одну цифру, одну велику та одну малу літеру.")

    # Вибір ролі
    while True:
        role = input("Введіть роль (doctor/admin): ").strip().lower()
        if role in ["doctor", "admin"]:
            break
        else:
            print("Невірна роль. Спробуйте знову (doctor або admin).")

    # Хешування паролю
    password_hash = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()

    # Генерація секрету для 2FA
```

```
    secret = pyotp.random_base32()
    print(f"\nСекрет для Google Authenticator (додай в додаток вручну):\n{secret}")

    # Вивід QR-коду в терміналі
    qr = qrcode.QRCode()
    qr.add_data(uri)
    qr.make(fit=True)
    qr.print_ascii()

    # Генерація QR-коду
    totp = pyotp.TOTP(secret)
    uri = totp.provisioning_uri(name=username, issuer_name="MyMedicalApp")
    print(f"\nQR-код для Google Authenticator (не обов'язково): {uri}")

    # Додавання нового користувача
    users[username] = {
        "password_hash": password_hash,
        "2fa_secret": secret,
        "role": role,
        "locked_until": None,              # Додано: тимчасове блокування (дата/час)
          "is_permanently_locked": False,      # Додано: постійне блокування (булеве
значення)
        "failed_login_attempts": 0         # Додано: лічильник невдалих спроб входу

    }

    # Збереження користувачів
    save_users(users)

    print("\nКористувача успішно додано!")
```

```
    # Логування дії
    log_action(f"Користувач {username} доданий з роллю {role}")

if __name__ == "__main__":
    register_user()
```

ДОДАТОК Е

Лістинг коду utils.py:

```
# utils.py
import re
import json
import os
import bcrypt
import pyotp

USERS_FILE = "users.json"
RECORDS_FILE = "records.json"

def load_records():
    try:
        with open(RECORDS_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    except FileNotFoundError:
        return []

def save_users(users):
    with open(USERS_FILE, "w") as f:
        json.dump(users, f, indent=4)

def load_users():
    try:
        with open(USERS_FILE, "r", encoding="utf-8") as f:
            data = json.load(f)
    except FileNotFoundError:
```

```
        return {}
    except json.JSONDecodeError:
            print(f"Помилка читання JSON з файлу {USERS_FILE}. Файл може бути пошкоджений.")
        return {}
    for user_data in data.values():
        user_data.setdefault("locked_until", None)
        user_data.setdefault("is_permanently_locked", False)
                user_data.setdefault("failed_login_attempts", 0) # Додаємо лічильник невдалих спроб
    return data

def save_records(records, filename="records.json"):
    """Зберігає записи в JSON файл."""
    with open(filename, "w", encoding="utf-8") as file:
        json.dump(records, file, indent=4, ensure_ascii=False)

def check_ukrainian_text(text):
        pattern = re.compile(r"^[А-Яа-яЇїІіЄєҐґ'’\s\-.,]+$")  # + крапки і коми, якщо треба
    return bool(pattern.match(text))

def check_password_strength(password):
        """Перевірка на мінімальну складність паролю (наприклад, мінімум 8 символів, цифра, велика і мала літера)"""
    if len(password) < 8:
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char.islower() for char in password):
```

```
        return False
    if not any(char.isupper() for char in password):
        return False
    return True

def delete_users(users, username):
    if username in users:
        del users[username]
        return True
    return False

def admin_actions():
    """Дії адміністратора (залежно від вашої логіки)."""
    print("Admin actions placeholder")

def add_record(records, record):
    """Додає новий запис."""
    records.append(record)
    return records

def register_user():
    """Реєстрація нового користувача."""
    if os.path.exists("users.json"):
        with open("users.json", "r", encoding="utf-8") as f:
            users = json.load(f)
    else:
        users = {}

    # Введення логіну
    while True:
```

```
        username = input("Введіть логін нового користувача: ")
        if username in users:
            print("Цей логін вже існує. Спробуйте інший.")
        else:
            break

    # Введення паролю
    while True:
        password = input("Введіть пароль: ")
        if check_password_strength(password):
            break
        else:
            print("Пароль має бути не менше 8 символів і містити хоча б одну цифру, одну велику та одну малу літеру.")

    # Вибір ролі
    role = input("Введіть роль (doctor/admin): ").strip().lower()
    if role not in ["doctor", "admin"]:
        print("Невірна роль, встановлено 'doctor' за замовчуванням.")
        role = "doctor"

    # Хешування паролю
    password_hash = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()

    # Генерація секрету для 2FA
    secret = pyotp.random_base32()
    print(f"\nСекрет для Google Authenticator (додай в додаток вручну):\n{secret}")

    # Генерація QR-коду
    totp = pyotp.TOTP(secret)
```

```
uri = totp.provisioning_uri(name=username, issuer_name="MyMedicalApp")
print(f"\n(не обов'язково) QR-лінк: {uri}")

# Додавання нового користувача
users[username] = {
    "password_hash": password_hash,
    "2fa_secret": secret,
    "role": role
}

# Збереження користувачів
with open("users.json", "w", encoding="utf-8") as f:
    json.dump(users, f, indent=4, ensure_ascii=False)

print("\nКористувача успішно додано!")

# Логування дії (якщо потрібно)
with open("log.txt", "a", encoding="utf-8") as log_file:
    log_file.write(f"Користувач {username} доданий з роллю {role}\n")
```

ДОДАТОК Є

Лістинг коду crypt_key.py:

```
import json
from cryptography.fernet import Fernet

key = b'3qw2__Ky1g2cakycxyhlbuU2mo45J-TZL5F2oTGOaF8='  # той самий ключ, що використовуєш для розшифрування
fernet = Fernet(key)

with open("users.json", "r", encoding="utf-8") as f:
    users = json.load(f)

for user_data in users.values():
    secret = user_data["2fa_secret"]
    # Спробуємо розшифрувати - якщо вийде, значить вже зашифровано
    try:
        decrypted = fernet.decrypt(secret.encode()).decode()
        # Якщо розшифрування пройшло, то секрет уже зашифрований, залишаємо як є
    except Exception:
        # Якщо не вдалося розшифрувати - зашифруємо зараз
        encrypted_secret = fernet.encrypt(secret.encode()).decode()
        user_data["2fa_secret"] = encrypted_secret

with open("users.json", "w", encoding="utf-8") as f:
    json.dump(users, f, ensure_ascii=False, indent=4)
```

```
print("Усі 2fa_secret тепер зашифровані.")

"""
from cryptography.fernet import Fernet

key = b'3qw2__Ky1g2cakycxyhlbuU2mo45J-TZL5F2oTGOaF8='
fernet = Fernet(key)
secret = "CKRS3KQT6HADLQWIU5Q7T5FMBGLW6W7Y"

enc1 = fernet.encrypt(secret.encode())
enc2 = fernet.encrypt(secret.encode())

print("Зашифрований секрет 1:", enc1.decode())
print("Зашифрований секрет 2:", enc2.decode())
print("Чи однакові:", enc1 == enc2)
```