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
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: «Засоби захисту персональних даних при дистанційній роботі»

Виконавець: студент IV курсу, групи КБ-43

_____ **Євгеній КОРОТИЧ**
(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 освітньо-професійної програми)

Студенту **КБ-43** (група) **Коротичу Євгенію Сергійовичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засоби захисту персональних даних при дистанційній роботі

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

Дані про сучасні методи захисту персональних даних, архітектура системи з авторизацією, реєстрацією, редагуванням і захистом даних.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Аналіз основних загроз для персональних даних при дистанційній роботі

Дослідження основних загроз та вразливостей при віддаленому доступі до даних

Та обґрунтування вимог до захисту персональних даних

Розробка рекомендацій та програмного проекту

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Полягає в розробці рекомендацій щодо захисту Персональних даних та розробці програмного проекту

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

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

Завдання прийняв до виконання _____ Євгеній КОРОТИЧ
(підпис) (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 – 20.01.2025	*виконано*
2	Аналіз літератури	21.01.2024 –12.03.2025	*виконано*
3	Аналіз особливостей віддаленої роботи	24.02.2025 – 04.03.2025	*виконано*
4	Визначення основних загроз і вразливостей	05.03.2024 – 21.03.2025	*виконано*
5	Вибір методів захисту персональних даних	22.03.2025 – 09.04.2025	*виконано*
6	Розробка структури захисної системи	10.04.2025 – 15.04.2025	*виконано*
7	Реалізація програмної частини (реєстрація, авторизація, безпека)	16.04.2025 – 30.04.2025	*виконано*
8	Тестування рішень і збір результатів, написання висновків	01.05. 2025 – 27.05.2025	*виконано*
9	Оформлення пояснювальної записки	28.05.2025 – 07.06.2025	*виконано*

Завдання видав			Сергій ДАКОВ
	(підпис)		(ім'я, прізвище)
Завдання прийняв до виконання			Євгеній КОРОТИЧ
	(підпис)		(ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 рок

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку джерел, додатків, має 69 сторінок основного тексту, 20 рисунків, 30 таблиць. Список джерел, які були використані, містить 20 найменування та займає 3 сторінки.

Мета роботи полягає в розробці рекомендацій для забезпечення захисту персональних даних при організації віддаленої роботи.

Для досягнення зазначеної мети поставлено наступні завдання:

- аналіз існуючих підходів до захисту персональних даних у контексті дистанційної роботи на основі вітчизняних і зарубіжних досліджень
- оцінка рівня безпеки сучасних технологій для забезпечення захисту інформації при віддаленому доступі до робочих ресурсів
- розробка рекомендацій щодо впровадження заходів безпеки для зниження ризиків витоку персональних даних при віддаленій роботі
- оцінка ефективності запропонованих рішень та їх тестування в реальних умовах

Об'єктом дослідження є процеси захисту персональних даних при віддаленій роботі.

Предметом дослідження виступають методи та технічні засоби, що забезпечують безпеку персональних даних у контексті використання віддаленого доступу до корпоративних ресурсів.

Методи дослідження:

- аналіз літературних джерел
- дослідження загроз та вразливостей, пов’язаних із використанням публічних мереж, хмарних сервісів і локальних пристроїв
- моделювання та реалізація програмного рішення для захисту персональних даних
- тестування ефективності заходів безпеки

Практичне значення роботи полягає в розробці рекомендацій та створенні програмного рішення, що дозволяє ефективно захищати персональні дані при віддаленій роботі. Запропоновані підходи можуть бути впроваджені в організаціях для підвищення рівня інформаційної безпеки та відповідності чинному законодавству.

Ключові слова: захист даних, віддалена робота, двофакторна автентифікація, шифрування, REST API, VPN, інформаційна безпека, персональні дані.

ЗМІСТ

ВСТУП

У сучасному світі зростання кількості користувачів Інтернету та популяризація віддаленої роботи поставили нові виклики перед забезпеченням безпеки персональних даних. Віддалена робота, що стала масовою практикою в умовах пандемії COVID-19, створює певні ризики для конфіденційності та цілісності інформації, що передається через мережу. Внаслідок цього виникає потреба у розробці нових рекомендацій і підходів до захисту персональних даних при організації віддаленого доступу до робочих ресурсів. Однією з основних проблем є недостатня обізнаність працівників та організацій щодо заходів безпеки при використанні Інтернету, що призводить до численних витоків інформації та компрометацій безпеки. Тому важливим напрямом сучасних наукових досліджень є вивчення механізмів і технологій для захисту персональних даних у контексті віддаленої роботи.

Проблема захисту персональних даних стає все більш актуальною в умовах глобалізації цифрових технологій та переходу значної частини економічної і соціальної діяльності в онлайн-простір. Віддалена робота потребує застосування новітніх технологій для забезпечення безпеки даних і належного захисту інформації при її обробці та передачі через Інтернет. За умов масштабного використання онлайн-сервісів та корпоративних мереж виникають нові загрози, такі як фішинг, вірусні атаки, перехоплення інформації, які можуть призвести до серйозних наслідків для організацій та окремих осіб. На основі даних аналізу вітчизняної та зарубіжної науково-технічної літератури, можна зазначити, що значна кількість досліджень зосереджена на розробці методів захисту інформації, проте існує потреба у більш детальному вивченні проблем безпеки саме в контексті віддаленої роботи, що є новим викликом для сучасної наукової спільноти.

Тема захисту персональних даних під час віддаленої роботи є надзвичайно важливою, оскільки недостатній рівень захисту може призвести до

значних фінансових та репутаційних втрат для компаній та організацій, а також до серйозних правових наслідків для працівників і керівників. Тому розробка ефективних рекомендацій щодо забезпечення безпеки персональних даних при віддаленій роботі набуває особливої важливості у нинішніх умовах. Питання безпеки інформації є не тільки технічною, а й соціальною проблемою, що потребує комплексного підходу з урахуванням як технічних, так і організаційних заходів.

Мета роботи полягає в розробці рекомендацій для забезпечення захисту персональних даних при організації віддаленої роботи. Для досягнення поставленої мети необхідно вирішити низку **завдань**:

- Аналіз існуючих підходів до захисту персональних даних у контексті віддаленої роботи на основі вітчизняних і зарубіжних досліджень.
- Оцінка рівня безпеки сучасних технологій для забезпечення захисту інформації при віддаленому доступі до робочих ресурсів.
- Розробка рекомендацій щодо впровадження заходів безпеки для зниження ризиків витоку персональних даних при віддаленій роботі.
- Оцінка ефективності запропонованих рішень та їх тестування в реальних умовах.

Об'єктом дослідження є процеси захисту персональних даних при віддаленій роботі.

Предметом дослідження виступають методи та технічні засоби, що забезпечують безпеку персональних даних у контексті використання віддаленого доступу до корпоративних ресурсів.

Результати дослідження можуть бути застосовані в організаціях, що практикують віддалену роботу, для підвищення рівня захисту персональних даних та інформаційної безпеки. Рекомендації можуть бути корисними для малого та середнього бізнесу, а також для великих компаній, що здійснюють транскордонні операції. Тестування запропонованих методів проводилася в рамках пілотних проєктів із застосуванням реальних корпоративних даних, що дозволило оцінити ефективність розроблених рішень та їх адаптацію до

конкретних умов.

Висновки та рекомендації, що містяться в даній роботі, можуть стати основою для подальших досліджень та розробок у сфері забезпечення інформаційної безпеки при віддаленій роботі та захисту персональних даних, зокрема для українських організацій, що активно використовують цифрові технології.

РОЗДІЛ 1
ОСОБЛИВОСТІ ВІДДАЛЕНОЇ РОБОТИ ТА ЗАХИСТ ПЕРСОНАЛЬНИХ ДАНИХ

1.1 Нормативно-правова база у сфері захисту персональних даних

У сучасних умовах віддалена робота стала невід'ємною частиною професійної діяльності, що підвищує актуальність питань захисту персональних даних. Забезпечення конфіденційності та безпеки інформації в дистанційному режимі вимагає чіткого розуміння та дотримання нормативно-правових актів, які регулюють цю сферу в Україні.

Основним законом держави є Конституція України [1], яка гарантує право на невтручання в особисте життя та захист персональних даних. Зокрема, стаття 32 Конституції визначає, що ніхто не може зазнавати втручання в його особисте та сімейне життя, окрім випадків, передбачених Конституцією. Це положення закладає фундамент для подальшого законодавчого регулювання захисту персональних даних.

Ключовим нормативно-правовим актом у цій сфері є Закон України "Про захист персональних даних" №2297-VI від 1 червня 2010 року. Цей закон регулює відносини, пов'язані з обробкою та захистом персональних даних, визначає права суб'єктів даних, обов'язки володільців та розпорядників баз даних, а також встановлює вимоги до обробки персональних даних. Закон підкреслює, що обробка персональних даних можлива лише за згодою суб'єкта або в інших випадках, передбачених законом [2].

Трудовий кодекс України містить положення, що стосуються захисту персональних даних працівників. Зокрема, стаття 2-1 визначає обов'язок роботодавця забезпечувати конфіденційність персональних даних працівників. Це особливо важливо в умовах віддаленої роботи, де обробка даних часто здійснюється через інформаційно-телекомунікаційні системи [3].

Закон України "Про інформацію" №2657-XII від 2 жовтня 1992 року встановлює загальні принципи інформаційних відносин, включаючи захист інформації з обмеженим доступом, до якої належать і персональні дані. Закон визначає права та обов'язки учасників інформаційних відносин щодо збору, зберігання, використання та поширення інформації [4].

У сучасних умовах віддаленої роботи захист персональних даних набуває особливого значення. Нормативно-правова база України у цій сфері формується з урахуванням міжнародних стандартів та національних особливостей.

Основні законодавчі акти України щодо захисту персональних даних, описані в таблиці 1.1.

Таблиця 1.1

Основні законодавчі акти України щодо захисту персональних даних

№	Назва	Опис
1	Закон України "Про захист персональних даних" №2297-VI від 1 червня 2010 року	визначає правові засади захисту персональних даних під час їх обробки, встановлює права суб'єктів даних та обов'язки володільців і розпорядників баз даних [5]
2	Кодекс законів про працю України	містить положення щодо трудових відносин, включаючи питання дистанційної роботи та захисту персональних даних працівників [6]

Міжнародні стандарти та їх імплементація в Україні описані в таблиці 1.2.

Таблиця 1.2

Міжнародні стандарти та їх імплементація в Україні

№	Назва	Опис
1	Конвенція Ради Європи №108 про захист осіб у зв'язку з автоматизованою обробкою персональних даних	Україна ратифікувала цю конвенцію, зобов'язуючись дотримуватися міжнародних стандартів у сфері захисту даних [7]
2	Загальний регламент про захист даних (GDPR)	хоча Україна не є членом Європейського Союзу, адаптація національного законодавства до вимог GDPR сприяє підвищенню рівня захисту персональних даних та інтеграції в європейський правовий простір [7]

Спеціальні нормативно-правові акти та рекомендації описані в таблиці 1.3.

Таблиця 1.3

Спеціальні нормативно-правові акти та рекомендації

№	Назва	Опис
1	Рекомендації Уповноваженого Верховної Ради України з прав людини щодо захисту персональних даних під час дистанційного надання освітніх послуг	містять поради щодо забезпечення конфіденційності та безпеки даних в умовах дистанційного навчання [8]
2	Рекомендації щодо застосування законодавства про захист персональних даних	надають роз'яснення щодо практичного впровадження норм закону в різних сферах діяльності [9]

Оновлення законодавства у сфері захисту персональних даних: у зв'язку з технологічним розвитком та цифровізацією, законодавство України потребує постійного оновлення. Наразі розроблено законопроєкти №8153 та №6177, які спрямовані на приведення національних норм у відповідність до міжнародних стандартів, зокрема GDPR [7].

Органи, відповідальні за контроль та нагляд описані в таблиці 1.4.

Таблиця 1.4

Органи, відповідальні за контроль та нагляд

№	Назва	Опис
1	Уповноважений Верховної Ради України з прав людини	здійснює нагляд за дотриманням прав у сфері захисту персональних даних, розглядає скарги та надає рекомендації щодо покращення законодавства [10]
2	Державна служба спеціального зв'язку та захисту інформації України	відповідає за технічний захист інформації та розробку відповідних стандартів і нормативних документів [11]

Захист персональних даних в умовах віддаленої роботи є критично

важливим питанням сучасних трудових відносин. Українська нормативно-правова база, спираючись на національні закони та міжнародні стандарти, забезпечує правові механізми для захисту інформації. Проте, з огляду на швидкий технологічний прогрес, необхідно постійно оновлювати та вдосконалювати законодавство, адаптуючи його до нових викликів та реалій цифрового світу.

1.2 Особливості віддаленої роботи

З розвитком технологій та глобалізацією ринку праці віддалена робота стала невід'ємною частиною сучасного професійного середовища. Цей формат праці надає працівникам гнучкість та можливість працювати з будь-якого місця, а роботодавцям — доступ до ширшого кола талантів. Однак віддалена робота має свої особливості, які впливають на організацію праці, комунікацію та захист персональних даних.

Організація робочого процесу описана в таблиці 1.5.

Таблиця 1.5

Особливості організації робочого процесу

№	Назва	Опис
1	Гнучкий графік роботи	Віддалена робота дозволяє працівникам самостійно планувати свій робочий час, що сприяє підвищенню продуктивності та балансу між роботою та особистим життям
2	Використання цифрових інструментів	Для ефективної організації роботи застосовуються різноманітні платформи для управління проєктами, такі як Trello або Asana, а також засоби для відеоконференцій, наприклад, Zoom або Microsoft Teams

Комунікація та взаємодія описані в таблиці 1.6.

Таблиця 1.6

Особливості комунікації та взаємодії

№	Назва	Опис
1	Віртуальні зустрічі	Відсутність фізичної присутності компенсується регулярними онлайн-зустрічами, що забезпечує командну взаємодію та обмін інформацією
2	Асинхронна комунікація	Через різницю в часових поясах або гнучкий графік працівники часто спілкуються через електронну пошту або месенджери, що дозволяє отримувати відповіді без негайної присутності співрозмовника

Технічне забезпечення описана в таблиці 1.7.

Таблиця 1.7

Особливості технічного забезпечення

№	Назва	Опис
1	Обладнання та програмне забезпечення	Працівники повинні мати надійні комп'ютери, стабільне інтернет-з'єднання та доступ до необхідних програм для виконання своїх обов'язків
2	Безпека інформації	Використання віртуальних приватних мереж (VPN), антивірусних програм та двофакторної аутентифікації є критично важливим для захисту корпоративних даних

Захист персональних даних описана в таблиці 1.8.

Таблиця 1.8

Особливості захисту персональних даних

№	Назва	Опис
1	**2**	**3**
1	Дотримання законодавства	Роботодавці зобов’язані забезпечити обробку та зберігання персональних даних відповідно до чинного законодавства, зокрема Закону України «Про захист персональних даних» [6]
2	Розробка внутрішніх політик	Необхідно впровадити чіткі внутрішні політики щодо обробки та захисту персональних даних, які враховують специфіку віддаленої роботи [12]

3	Навчання працівників	Регулярні тренінги допоможуть працівникам усвідомити важливість захисту даних та ознайомитися з найкращими практиками безпеки [8]

Правове регулювання описані в таблиці 1.9.

Таблиця 1.9

Особливості правового регулювання

№	Назва	Опис
1	Актуалізація законодавства	Верховна Рада України прийняла за основу законопроєкт №8153 "Про захист персональних даних", який спрямований на приведення національного законодавства у відповідність до європейських стандартів [13]
2	Контроль та відповідальність	Офіс Омбудсмана здійснює нагляд за дотриманням законодавства у сфері захисту персональних даних, забезпечуючи захист прав громадян [14]

Виклики та перспективи в таблиці 1.10.

Таблиця 1.10

Особливості викликів та перспектив

№	Назва	Опис
1	Технічні ризики	Збільшення кількості кібератак та витоків інформації вимагає постійного оновлення засобів захисту та підвищення обізнаності працівників щодо кібербезпеки
2	Міжнародні стандарти	Імплементація норм Загального регламенту захисту даних (GDPR) в українське законодавство сприятиме підвищенню рівня захисту персональних даних та інтеграції України до європейського правового простору [15]

Віддалена робота відкриває нові можливості для працівників та роботодавців, проте вимагає ретельного підходу до організації робочого процесу та захисту персональних даних. Дотримання законодавчих вимог, впровадження сучасних технічних засобів безпеки та регулярне навчання персоналу є ключовими елементами успішної та безпечної віддаленої роботи.

1.3 Основні загрози для персональних даних при віддаленій роботі

В умовах стрімкого переходу до віддаленої роботи захист персональних даних набуває особливої актуальності. Розуміння основних загроз, що виникають у цьому контексті, є ключовим для забезпечення безпеки інформації.

Основні загрози для персональних даних при віддаленій роботі зображені в таблиці 1.11.

Таблиця 1.11

Основні загрози для персональних даних при віддаленій роботі

№	Назва	Опис
1	**2**	**3**
1	Фішингові атаки	Збільшення кількості електронних комунікацій під час віддаленої роботи призводить до підвищеного ризику фішингових атак. Зловмисники надсилають підроблені повідомлення, що імітують офіційні джерела, з метою отримання конфіденційної інформації або встановлення шкідливого програмного забезпечення. Особливо небезпечними є такі атаки під час спонтанного переходу на дистанційну роботу, коли працівники ще не ознайомлені з новими протоколами безпеки [16]
2	Використання незахищених Wi-Fi мереж	Підключення до публічних або домашніх мереж Wi-Fi без належного захисту може призвести до перехоплення даних. Зловмисники можуть отримати доступ до переданої інформації, включаючи паролі та інші персональні дані. Це особливо актуально для працівників, які працюють з різних місць поза домом [17]
3	Недостатній захист кінцевих пристроїв	Відсутність оновлень операційних систем, антивірусного програмного забезпечення та використання слабких паролів на особистих пристроях створює вразливості. Це може призвести до несанкціонованого доступу до корпоративних даних через особисті комп'ютери або мобільні пристрої [18]
4	Використання особистих пристроїв для робочих цілей	Змішування особистих і робочих даних на одному пристрої підвищує ризик витоку інформації. Особисті пристрої можуть не відповідати корпоративним стандартам безпеки, що робить їх потенційною мішенню для атак [18]
5	Недостатній контроль доступу	Надання працівникам надмірних прав доступу до корпоративних систем може призвести до випадкового або навмисного витоку даних. Важливо регулярно

		переглядати та обмежувати права доступу відповідно до посадових обов'язків [19]

продовження таблиці 1.11

6	Відсутність шифрування даних	Передача та зберігання даних без належного шифрування робить їх вразливими до перехоплення та несанкціонованого доступу. Шифрування є критично важливим для захисту конфіденційної інформації [20]
7	Фізична втрата або крадіжка пристроїв	Під час віддаленої роботи працівники можуть використовувати ноутбуки, планшети та смартфони в різних місцях. Втрата або крадіжка таких пристроїв без належного захисту може призвести до компрометації персональних даних [18]
8	Недостатня обізнаність працівників щодо кібербезпеки	Брак навчання та обізнаності щодо сучасних кіберзагроз серед працівників може призвести до помилок, які компрометують безпеку даних. Регулярні тренінги та оновлення знань є необхідними для мінімізації ризиків [16]
9	Використання незахищених хмарних сервісів	Зберігання та обробка даних у хмарних сервісах без належних заходів безпеки може призвести до несанкціонованого доступу. Важливо обирати перевірених постачальників послуг та налаштовувати безпекові параметри відповідно до корпоративних стандартів [21]
10	Відсутність політик безпеки для віддаленої роботи	Без чітко визначених та впроваджених політик безпеки для віддаленої роботи працівники можуть несвідомо порушувати протоколи, що призводить до вразливостей. Розробка та комунікація таких політик є ключовими для захисту даних [13]

Віддалена робота відкриває нові можливості, але також приносить із собою низку загроз для персональних даних. Розуміння та ідентифікація цих загроз є першим кроком до їх нейтралізації. Впровадження комплексних заходів безпеки, регулярне навчання працівників та постійний моніторинг систем допоможуть мінімізувати ризики та забезпечити захист конфіденційної інформації в умовах дистанційної роботи.

1.4 Постановка завдання щодо забезпечення захисту персональних даних

Забезпечення захисту персональних даних є однією з найважливіших складових безпеки у сфері віддаленої роботи. В умовах активного впровадження технологій, зокрема для роботи в Інтернеті, велика кількість даних стає доступною для зловмисників, що ставить під загрозу конфіденційність та безпеку користувачів. Для належного захисту особистої інформації необхідно вжити комплексу заходів, які включають технологічні рішення, політики доступу, а також процеси верифікації та авторизації. Постановка завдання щодо забезпечення захисту персональних даних має бути продумана та комплексна, зокрема в контексті сучасних методів автентифікації, шифрування та моніторингу дій користувачів.

1.4.1. Підвищення рівня безпеки автентифікації користувачів

Один з основних етапів захисту персональних даних — це безпечна автентифікація користувачів. Для уникнення несанкціонованого доступу до системи важливо використовувати сильні методи верифікації. Найбільш ефективними є поєднання паролів з додатковими шарами захисту, такими як двофакторна автентифікація (2FA). Остання може включати генерацію одноразових кодів через спеціальні додатки або через електронну пошту. Такий підхід забезпечує додатковий рівень безпеки, навіть якщо пароль користувача став відомий зловмисникам.

У рамках системи захисту даних важливо застосовувати обов’язкове хешування паролів користувачів із використанням надійних алгоритмів, що унеможливлює їх відновлення навіть у разі компрометації бази даних. Також важливо забезпечити контроль активності користувачів, щоб запобігти несанкціонованим спробам входу та зафіксувати всі дії у системі, що дозволить відстежити потенційні загрози.

1.4.2. Захист персональних даних на всіх етапах обробки

Друге завдання — це забезпечення конфіденційності даних під час їх обробки, зберігання та передачі. Персональні дані повинні бути надійно захищені за допомогою шифрування на етапах їх зберігання та передачі через мережу Інтернет. Для цього використовуються криптографічні методи шифрування, які гарантують, що навіть при перехопленні даних вони не можуть бути розшифровані без наявності відповідного ключа.

Іншим важливим завданням є безпечне зберігання даних у базах даних. Всі чутливі дані, такі як паролі, адреси, номери телефонів, повинні зберігатися в зашифрованому вигляді, і доступ до них повинен бути обмежений лише авторизованим користувачам або службам. Система повинна реалізовувати політику мінімальних прав доступу, що дозволяє знижувати ризик витоку персональних даних.

1.4.3. Запобігання несанкціонованому доступу до особистих даних

Наступний важливий крок — це забезпечення належного контролю доступу до персональних даних. Для цього необхідно впровадити багаторівневу авторизацію, яка передбачає перевірку даних користувача на основі кількох факторів, що значно ускладнює спроби несанкціонованого доступу. Важливою частиною цієї політики є встановлення обмежень на доступ до різних типів даних залежно від ролей користувачів. Наприклад, адміністратор може мати доступ до більш широких даних порівняно зі звичайним користувачем. Усі спроби доступу повинні бути зафіксовані в журналі подій системи, щоб забезпечити можливість ретельного аудиту та перевірки будь-яких аномальних дій.

У системах з високими вимогами безпеки також рекомендовано застосовувати політику обмеженого часу доступу, де кожен сеанс користувача має обмежений термін дії, після чого відбувається автоматичний вихід з системи. Це знижує ймовірність несанкціонованого доступу до даних у разі,

якщо пристрій користувача потрапить до чужих рук.

1.4.4. Забезпечення надійного моніторингу та аудиту

Щоб запобігти витоку даних або їх несанкціонованому використанню, необхідно впровадити систему моніторингу активності користувачів. Кожен крок, що здійснюється користувачем, має бути зафіксований у спеціалізованому журналі аудиту. Це дозволяє не тільки вчасно виявляти підозрілі дії, а й здійснювати ретроспективний аналіз у разі виявлення інцидентів безпеки. Логування повинно включати деталі про кожну дію, таку як вхід в систему, зміни особистих даних, спроби доступу до заборонених розділів.

Журнал активності має бути надійно захищений від підробки, а доступ до нього повинен бути обмежений лише для адміністраторів, які мають необхідні права для перевірки безпеки системи.

1.4.5. Використання механізмів для захисту від атак

У рамках захисту персональних даних важливо враховувати ризики, пов'язані з потенційними кібератаками, такими як SQL-ін’єкції, міжсайтові скриптові атаки (XSS) чи фішинг. Для запобігання таким атакам необхідно застосовувати методи обробки вхідних даних з використанням параметричних запитів, що дозволяє уникнути SQL-ін’єкцій. Важливим є також використання сучасних засобів захисту від XSS, які можуть виникати в результаті маніпуляцій із веб-формами.

Найбільш надійний захист забезпечується поєднанням кількох рівнів захисту, включаючи брандмауери, антишкідливе програмне забезпечення та регулярне оновлення всіх компонентів системи, щоб усунути вразливості.

Забезпечення належного захисту персональних даних під час віддаленої роботи є невід’ємною складовою безпеки будь-якої організації. Для досягнення

високого рівня захисту необхідно використовувати багаторівневі методи автентифікації, шифрування даних, обмеження доступу та ефективний моніторинг користувацької активності. Крім того, важливою є також здатність системи реагувати на потенційні загрози в реальному часі та забезпечити захист від зовнішніх атак. Тільки так можна досягти належного рівня безпеки персональних даних, що є важливою складовою для збереження довіри користувачів і стабільності бізнес-процесів.

Висновки за розділом 1

Захист персональних даних є критично важливим завданням у контексті віддаленої роботи. Оскільки технологічний прогрес змінює способи взаємодії працівників та роботодавців, важливість забезпечення належного рівня безпеки даних не можна недооцінювати. Система нормативно-правового регулювання в Україні забезпечує достатній рівень захисту персональних даних, але, з огляду на швидкий розвиток технологій та зміни в характері трудових відносин, законодавство потребує постійного вдосконалення та адаптації до нових реалій цифрового середовища. Віддалена робота дозволяє зручніше організувати робочий процес, проте вона також створює нові виклики для захисту персональних даних. Технічні рішення, які використовуються для забезпечення безпеки інформації, повинні бути комплексними та адаптованими до конкретних умов роботи на відстані.

Особливе значення має підвищення рівня безпеки через автентифікацію користувачів, шифрування даних та контролю доступу до конфіденційної інформації. Захист персональних даних на всіх етапах обробки є основним пріоритетом для будь-якої організації, яка прагне забезпечити безпеку своїх працівників і клієнтів. Постійний моніторинг систем і аудит дій користувачів допомагають своєчасно виявляти потенційні загрози та реагувати на них. Крім того, зростаюча кількість кіберзагроз вимагає від підприємств вжиття заходів для запобігання несанкціонованому доступу до даних і захисту від атак. Усі ці

елементи мають бути інтегровані в єдину систему безпеки, що дозволить не лише знизити ризики, а й гарантувати безпеку і конфіденційність персональних даних під час віддаленої роботи.

Забезпечення захисту персональних даних в умовах віддаленої роботи є важливою умовою для успішної та безпечної діяльності організацій. Реалізація цих заходів дозволяє не лише мінімізувати ризики витоку інформації, а й зміцнити довіру користувачів та партнерів, що, в свою чергу, сприяє стабільності і розвитку бізнес-процесів.

РОЗДІЛ 2
ОСНОВНІ ЗАГРОЗИ ТА ВРАЗЛИВОСТІ ПРИ ВІДДАЛЕНОМУ ДОСТУПІ ДО ДАНИХ

2.1 Загрози, пов’язані з аутентифікацією та авторизацією

Аутентифікація та авторизація є критично важливими елементами забезпечення безпеки при віддаленій роботі. Вони відповідають за правильне визначення користувачів та надання доступу до персональних даних і ресурсів системи. Неналежна реалізація цих процесів може призвести до серйозних загроз безпеці, що особливо актуально в умовах віддаленого доступу. З огляду на зростаючу кількість хакерських атак та різноманітних схем шахрайства, важливо забезпечити високий рівень захисту на кожному етапі процесу аутентифікації та авторизації. Далі розглядаються основні загрози, які можуть виникнути при реалізації цих процесів.

Загрози, пов’язані з аутентифікацією та авторизацією, представлені в таблиці 2.1.

Таблиця 2.1

Загрози, пов’язані з аутентифікацією та авторизацією

	Назва	**Опис**
	2	**3**
	Ненадійні паролі та їх злом	Однією з найбільших загроз, пов’язаних із процесом аутентифікації, є використання слабких паролів. Незважаючи на те, що багато систем вимагають від користувачів створювати паролі певної складності, часто вони не відповідають вимогам безпеки. Такі паролі можуть бути вкрадені через фішинг, атаки методом підбору (brute-force) або використання програм для автоматичного зламу паролів. Важливою практикою є використання складних паролів, що містять великі та малі літери, цифри та спеціальні символи, а також регулярна зміна паролів

продовження таблиці 2.1

	2	3
	Атаки типу "Man-in-the-Middle" (MitM)	Загроза "Man-in-the-Middle" виникає, коли атакувальник перехоплює та змінює комунікацію між користувачем і сервером, не будучи поміченим. У контексті аутентифікації та авторизації атака може здійснюватися на етапі передачі паролів або токенів доступу через незашифровані канали зв'язку. Якщо підключення не забезпечено належним рівнем шифрування (наприклад, HTTPS), зловмисники можуть перехопити чутливу інформацію і отримати несанкціонований доступ до облікових записів користувачів
	Використання токенів доступу	Використання токенів для аутентифікації та авторизації стало широко поширеною практикою для зручності користувачів. Однак це створює додаткові ризики, якщо токени зберігаються або передаються неналежним чином. Якщо токени не захищені належним чином, вони можуть бути вкрадені або підроблені, що дозволить зловмисникам отримати доступ до системи. Важливо використовувати механізми для захисту токенів, такі як їх зашифроване зберігання та короткий термін їх дії
	Проблеми з багатофакторною аутентифікацією (2FA)	Багатофакторна аутентифікація є важливим інструментом захисту, що додає додатковий рівень безпеки. Однак навіть вона не є повністю захищеною. Існує кілька видів атак, спрямованих на обхід 2FA, таких як фішинг для отримання одноразових кодів, перехоплення SMS або соціальна інженерія. Проте, при правильному налаштуванні та використанні додаткових засобів захисту (наприклад, апаратних токенів або додатків для генерації кодів) можна значно знизити ризики, пов'язані з багатофакторною аутентифікацією
	Використання слабких або вкрадених сесій	Під час віддаленого доступу до системи можуть виникнути загрози, пов'язані з використанням слабких або вкрадених сесій. Якщо система не має належних механізмів для перевірки автентичності сесії, зловмисники можуть скористатися вкраденими кукисами або сесіями для отримання доступу до користувацьких даних. Важливо впроваджувати механізми автоматичного завершення сесії після певного часу бездіяльності або перевірку активності сесії через постійну перевірку аутентифікації

завершення таблиці 2.1

	2	3
	Недостатній контроль доступу до чутливих даних	Авторизація повинна бути реалізована таким чином, щоб користувачі могли отримати доступ лише до тих ресурсів, які їм дійсно необхідні для виконання роботи. Недостатній контроль доступу до персональних даних або важливих ресурсів може призвести до витоку або несанкціонованого використання інформації. Важливо мати систему чіткої класифікації прав доступу, де кожен користувач має визначену роль і рівень доступу, що мінімізує можливості для зловмисників
	Атаки на сервери аутентифікації	Якщо сервер, що обробляє процес аутентифікації (наприклад, сервер бази даних або API-сервер), не захищений належним чином, це може призвести до компрометації всієї системи. Атаки на сервери можуть включати SQL-ін'єкції, DoS-атаки або інші методи, спрямовані на отримання доступу до чутливих даних користувачів, зокрема хешованих паролів або токенів. Потрібно впроваджувати належні заходи безпеки на рівні серверів, включаючи фільтрацію вводу, регулярні оновлення системи безпеки та використання додаткових засобів захисту, таких як обмеження кількості запитів

Аутентифікація та авторизація є важливими компонентами безпеки при віддаленій роботі. Однак реалізація цих процесів не без ризиків, і недостатньо ефективні механізми можуть спричинити серйозні загрози. Від слабких паролів до вкрадених сесій та атак на сервери — кожен етап аутентифікації та авторизації потребує ретельної уваги. Для мінімізації ризиків важливо використовувати сучасні методи захисту, такі як багатофакторна аутентифікація, безпечні канали зв'язку та постійний моніторинг активності користувачів. Всі ці заходи допоможуть створити більш надійне середовище для віддаленої роботи та захисту персональних даних.

2.2 Загрози, пов’язані з використанням незахищених мереж

У сучасному світі віддалена робота стає все більш популярною, і з цим пов’язані численні переваги, такі як гнучкість та доступ до ресурсів з будь-якої

точки світу. Однак, незважаючи на всі позитивні моменти, віддалена робота несе в собі значні ризики, зокрема пов’язані з використанням незахищених мереж для доступу до корпоративних систем та персональних даних. Порушення безпеки в таких мережах може призвести до серйозних наслідків, включаючи витік чутливої інформації, фінансові збитки та репутаційні втрати.

Загрози, пов’язані з використанням незахищених мереж представлені в таблиці 2.2.

Таблиця 2.2

Загрози, пов’язані з використанням незахищених мереж

	Назва	**Опис**
	2	**3**
	Вразливості при використанні публічних Wi-Fi мереж	Публічні Wi-Fi мережі, зокрема ті, що знаходяться у кафе, аеропортах або готелях, є основним каналом для віддалених працівників. Однак такі мережі зазвичай мають слабкий рівень захисту, що створює відкритий доступ для зловмисників. Хакери можуть легко налаштувати підроблену точку доступу, що дозволяє перехоплювати трафік користувачів, підслуховувати їхні дії та отримувати доступ до чутливих даних, зокрема паролів, логінів та інших особистих даних
	Перехоплення трафіку	Коли користувачі відправляють або отримують дані через незахищені мережі, зловмисники можуть легко перехопити трафік, який не зашифрований. Відсутність шифрування може призвести до того, що важлива інформація, зокрема паролі та конфіденційні дані, будуть доступні хакерам. Оскільки багато віддалених працівників часто використовують незахищені канали для доступу до систем, це збільшує ризик витоку даних
	Атаки типу "man-in-the-middle" (MITM)	Атака типу "man-in-the-middle" є однією з найбільш поширених загроз при використанні незахищених мереж. Вона полягає в тому, що зловмисник ставить себе між клієнтом та сервером, перехоплюючи, змінюючи та зчитуючи дані, що передаються. Наприклад, у випадку незащищеної мережі атакуючий може перенаправити з’єднання користувача на підроблений сервер, збираючи персональні дані, логіни та паролі, або навіть впроваджуючи шкідливий код

продовження таблиці 2.2

	2	3
	Витік даних через невідомі додатки	Використання незахищених мереж може також призвести до того, що віддалені працівники будуть підключатися до сторонніх додатків або несанкціонованих сервісів, що підвищує ймовірність витоку даних. Це особливо актуально, якщо користувачі не дотримуються принципів безпеки або використовують слабо захищені додатки для виконання робочих задач. Такі додатки можуть містити вразливості, які дозволяють зловмисникам проникати в систему та викрадати конфіденційну інформацію
	Використання незашифрованих з’єднань	Відсутність використання протоколів захищеного з’єднання (наприклад, HTTPS або VPN) може стати ще однією вразливістю при роботі через незахищену мережу. Протоколи захищеного з’єднання створюють шифровану тунель для передачі даних між пристроєм користувача та сервером. Без цього захисту зловмисники можуть легко прослуховувати трафік і отримувати доступ до переданої інформації. Більше того, багато корпоративних систем не мають відповідних механізмів для забезпечення такого шифрування, що збільшує вразливість під час підключення до мережі через публічні точки доступу
	Атаки через відкриті порти	Віддалений доступ до корпоративних систем через незахищені мережі може призвести до використання вразливих портів, які дозволяють зловмисникам здійснювати атаки через відкриті з'єднання. Використання таких портів без належного захисту може дати хакерам доступ до внутрішніх систем підприємства, дозволяючи їм викрадати або пошкоджувати конфіденційну інформацію. Відсутність належного фаєрволу або використання слабких налаштувань безпеки може збільшити ймовірність таких атак
	Ризики, пов’язані з недостатньою перевіркою автентичності	Зловмисники можуть використовувати незахищені мережі для проведення атак з метою обману користувачів за допомогою фальшивих запитів для авторизації або підробки особистих даних. Наприклад, уразливість у механізмах автентифікації або використання слабо захищених паролів може сприяти несанкціонованому доступу до даних або навіть до адміністраторських прав у системі

Отже, незахищені мережі створюють серйозні загрози безпеці персональних даних при віддаленій роботі. Від перехоплення трафіку до атак типу "man-in-the-middle" – кожна з цих загроз вимагає застосування відповідних заходів безпеки. Для забезпечення надійного захисту користувачі повинні використовувати захищені канали зв’язку, зокрема VPN, застосовувати шифрування даних та ретельно перевіряти автентичність підключень. В іншому випадку віддалена робота може стати значною загрозою для конфіденційності та безпеки даних.

2.3 Загрози, пов’язані з використанням хмарних сервісів

З розвитком технологій віддаленої роботи, організації все частіше звертаються до хмарних сервісів для зберігання, обробки та доступу до даних. Хмарні платформи надають безліч переваг, зокрема зручність, масштабованість і гнучкість, однак разом із цими перевагами виникають і значні загрози для захисту персональних даних. Враховуючи важливість збереження конфіденційності та цілісності інформації, аналіз загроз, пов’язаних із використанням хмарних сервісів, стає невід'ємною частиною забезпечення безпеки при віддаленому доступі до даних.

Загрози представлені в таблиці 2.3.

Таблиця 2.3

Загрози, пов’язані з використанням хмарних сервісів

	Назва	**Опис**
	2	**3**
	Невизначеність у місцезнаходженні даних	Один із головних ризиків при використанні хмарних сервісів полягає в невизначеності фізичного місцезнаходження серверів, на яких зберігаються дані. Хмари, як правило, використовують віддалені дата-центри, розташовані в різних країнах. Це призводить до потенційних проблем із дотриманням юридичних вимог, таких як захист персональних даних і відповідність нормам законодавства країни, де ці дані обробляються. Наявність кількох юрисдикцій може створювати ризики, якщо закони про захист даних у різних країнах не відповідають стандартам, прийнятим у державі, що надає послугу

продовження таблиці 2.3

	Вразливості інтерфейсів для програмного доступу (API)	Хмарні сервіси забезпечують програмний доступ до даних через API, що дозволяє автоматизувати робочі процеси. Однак інтерфейси API часто є уразливими місцями для зловмисників. Неякісно налаштовані або погано захищені API можуть стати вектором для атак, таких як несанкціонований доступ до персональних даних, маніпулювання або крадіжка важливої інформації. Атакуючи API, зловмисники можуть отримати доступ до величезних обсягів конфіденційних даних
	Недостатня безпека на стороні постачальника послуг	Незважаючи на те, що великі хмарні постачальники послуг намагаються забезпечити високий рівень безпеки, існують ситуації, коли самі постачальники не можуть гарантувати належний захист даних. Це може стосуватися як технічних проблем, таких як уразливості в програмному забезпеченні, так і людських помилок, коли адміністратори систем допускають помилки при налаштуванні безпеки або доступу до даних. Якщо постачальник хмарних послуг не виконує необхідних заходів щодо захисту даних, це може призвести до витоку персональної інформації
	Ризик витоку даних через спільний доступ	Багато хмарних платформ дозволяють користувачам спільно використовувати дані, що може стати джерелом ризиків. У разі ненавмисного надання доступу до конфіденційної інформації або використання незахищених каналів для передачі даних, можливий витік персональної інформації. Спільний доступ до документів або баз даних без належного контролю та моніторингу може призвести до доступу несанкціонованих осіб, що здатне завдати значної шкоди як організаціям, так і окремим особам
	Атаки типу «зловмисний інсайдер»	Хмарні сервіси часто надають доступ до персональних даних численним співробітникам постачальника послуг або стороннім постачальникам для виконання певних функцій. У разі наявності внутрішньої загрози — наприклад, коли співробітник має зловмисні наміри — існує ризик крадіжки або маніпулювання персональними даними. Існує також ймовірність, що доступ до даних можуть отримати співробітники, які мають надмірні привілеї для виконання своїх обов'язків, що також ставить під загрозу безпеку інформації

завершення таблиці 2.3

	Проблеми з автентифікацією та авторизацією	Однією з найбільших загроз є ненадійні механізми автентифікації та авторизації доступу до хмарних ресурсів. Якщо користувачі або системи не проходять належну перевірку на право доступу до даних, це може призвести до несанкціонованого доступу. Наприклад, використання слабких паролів або неактуальних методів автентифікації без двоетапної перевірки може створювати величезну вразливість. Недостатній контроль за доступом до даних підвищує ймовірність атак, таких як фішинг або зловмисне використання облікових записів
	Неадекватне шифрування даних	Невід'ємною частиною захисту персональних даних є шифрування. У разі використання хмарних платформ для зберігання чутливої інформації, неадекватне або ненадійне шифрування даних може стати причиною їхнього витоку. Хмарні постачальники часто застосовують стандартні методи шифрування, але якщо вони не налаштовані правильно, або якщо вони використовують застарілі алгоритми, це може поставити під загрозу цілісність і конфіденційність персональних даних
	Загроза через зовнішні атаки	Хмари можуть стати мішенню для численних атак ззовні, таких як DDoS-атаки, коли зловмисники намагаються перевантажити сервери або мережі хмарного провайдера. Такі атаки можуть призвести до тимчасових або тривалих відключень сервісів, а в деяких випадках — до втрати даних, особливо якщо резервні копії не виконуються належним чином

Використання хмарних сервісів для зберігання та обробки персональних даних значно підвищує зручність і ефективність віддаленої роботи, проте наявні значні загрози, які потребують ретельного контролю та захисту. Ключовими питаннями захисту є вибір надійного постачальника послуг, забезпечення належного рівня шифрування даних, регулярне оновлення методів автентифікації та моніторинг активності в хмарному середовищі. Необхідно також враховувати юридичні та організаційні питання, такі як відповідність міжнародним стандартам і законодавству, що регулює захист персональних даних.

2.4 Загрози, пов’язані зі зловмисним програмним забезпеченням

Захист персональних даних у контексті віддаленої роботи є надзвичайно важливою задачею, оскільки зловмисне програмне забезпечення (зазвичай позначене абревіатурою "ЗП") може серйозно загрожувати конфіденційності, цілісності та доступності інформації. Віддалений доступ до корпоративних або особистих даних відкриває нові можливості для зловмисників, оскільки зростає кількість каналів для атак, які вони можуть використовувати. У цьому контексті важливо детально розглянути основні типи зловмисного програмного забезпечення та їхню здатність ставити під загрозу захист персональних даних при віддаленому доступі.

Загрози представлені в таблиці 2.4.

Таблиця 2.4

Загрози, пов’язані зі зловмисним програмним забезпеченням

	Назва	**Опис**	**Ризики для персональних даних Ризики для персональних даних**
	2	**3**	**4**
	Віруси та черви	Віруси та черви є одними з найбільш поширених видів зловмисного програмного забезпечення. Вони можуть поширюватися через незащищені канали комунікації, наприклад, через електронну пошту, незахищені мережі або вразливі веб-ресурси. У випадку віддаленого доступу, ці програми можуть автоматично інфікувати комп'ютери, які підключаються до корпоративної мережі, або пристрої користувачів. Зокрема, віруси можуть не лише пошкодити дані, але й спричинити втрату важливої інформації або дозволити зловмисникам отримати доступ до конфіденційних даних	Пошкодження або втрату персональних даних, що зберігаються на пристроях користувачів. Можливість несанкціонованого доступу до облікових записів і конфіденційної інформації. Інфікування іншого програмного забезпечення, що може призвести до витоку даних

продовження таблиці 2.4

	Троянські програми	Троянські програми, або "трояни", є одним з найбільш небезпечних типів зловмисного програмного забезпечення, оскільки вони маскуються під корисні програми або файли, що користувачі можуть самостійно завантажувати. У віддаленому середовищі трояни можуть проникати через відкриті порти або вразливі додатки, які використовуються для доступу до корпоративних мереж. Коли троян потрапляє на пристрій, він може здійснювати несанкціоновані дії, наприклад, передавати конфіденційну інформацію до зловмисників	Крадіжка або витік персональних даних, таких як адреси електронної пошти, номери телефонів та інші чутливі відомості Встановлення бекдорів, які дозволяють зловмисникам контролювати систему користувача на відстані Можливість крадіжки облікових даних для доступу до важливих акаунтів
	Шкідливе програмне забезпечення для перехоплення клавіатурного вводу (Keyloggers)	Keyloggers – це спеціалізоване програмне забезпечення, яке фіксує введення з клавіатури. Це один з найбільш підступних видів атак, оскільки він дозволяє зловмисникам отримати доступ до всіх введених користувачем даних, включаючи паролі, номери кредитних карток і іншу чутливу інформацію. Використання таких програм є особливо небезпечним для віддалених працівників, оскільки вони часто працюють у невідповідно захищених середовищах, таких як особисті комп'ютери або незахищені мережі Wi-Fi.	Перехоплення паролів і іншої конфіденційної інформації при вході в систему Викрадення банківських реквізитів та особистої фінансової інформації Зловмисники можуть отримати доступ до корпоративних ресурсів через захоплені облікові дані

	Атаки типу "чоловік посередині" (Man-in-the-Middle, MITM)	Атаки типу "чоловік посередині" використовуються для перехоплення і змінювання даних, що передаються між користувачем і сервером. У віддаленому середовищі такі атаки можуть відбуватися через незахищені або нешифровані канали зв'язку, особливо в умовах використання публічних або слабко захищених мереж. Зловмисник може втручатися в комунікації, отримуючи доступ до чутливої інформації або навіть модифікуючи її без відома користувача	Перехоплення логінів, паролів та інших чутливих даних Модифікація або фальсифікація даних, що передаються, наприклад, при здійсненні фінансових транзакцій або зміні особистої інформації Підмова користувача до виконання дій, які можуть призвести до зловживання або шахрайства

продовження таблиці 2.4

	Рансоумери	Рансоумери (від англ. ransomware) є одним із найбільш руйнівних видів шкідливого програмного забезпечення, яке може заблокувати доступ до файлів або зашифрувати їх, вимагаючи викуп за відновлення доступу. В умовах віддаленої роботи ці програми можуть бути особливо небезпечними, оскільки вони можуть поширюватися через невідомі або підроблені додатки, а також через соціальні інженерії. У разі потрапляння в систему, рансоматери можуть зашкодити персональним даним, ускладнюючи або унеможливлюючи доступ до них	Втрата або блокування доступу до важливих файлів і баз даних Викрадення даних після їх дешифрування, оскільки деякі варіанти рансоматерів можуть не лише шифрувати, а й копіювати дані перед блокуванням Фінансові витрати на викуп для відновлення доступу до даних

Зловмисне програмне забезпечення є серйозною загрозою для захисту персональних даних при віддаленому доступі. Від вірусів та червів до більш складних атак типу "чоловік посередині" або рансоматерів, усі ці програми здатні спричинити серйозні наслідки для конфіденційності, цілісності та доступності даних. Використання сучасних засобів захисту, таких як шифрування з'єднань, регулярне оновлення програмного забезпечення, а також впровадження багатофакторної автентифікації є важливими заходами для

мінімізації ризиків, пов'язаних із зловмисним програмним забезпеченням в умовах віддаленої роботи.

2.5 Загрози, пов’язані зі збереженням даних на локальних пристроях

В умовах віддаленої роботи збереження персональних даних на локальних пристроях створює численні виклики в сфері інформаційної безпеки. Під час використання особистих комп'ютерів, ноутбуків та мобільних пристроїв для доступу до корпоративних або чутливих даних існують суттєві загрози, що можуть призвести до витоку, втрати або несанкціонованого доступу до інформації. Зберігання таких даних без належного захисту на локальних пристроях може бути вразливим місцем для різноманітних атак, що підривають безпеку інформації.

Загрози представлені в таблиці 2.5.

Таблиця 2.5

Загрози, пов’язані зі збереженням даних на локальних пристроях

	Назва	**Опис**
	2	**3**
	Несанкціонований доступ до даних	Однією з головних загроз є можливість несанкціонованого доступу до локальних пристроїв, що містять чутливу інформацію. Якщо пристрій не захищений належним чином (наприклад, відсутність складних паролів чи біометричної аутентифікації), зловмисники можуть отримати доступ до даних, фізично маніпулюючи пристроєм. Особливо це актуально, коли пристрій залишається без нагляду або використовується кількома людьми. Вразливість до такої загрози можна зменшити за допомогою надійних паролів, багатоетапної аутентифікації та системи шифрування, що забезпечить захист даних навіть у разі втрати пристрою
	Втрата або крадіжка пристрою	У разі втрати або крадіжки пристрою з важливою інформацією може статися витік персональних даних. Локальні пристрої часто не мають достатнього захисту для захисту даних, коли вони потрапляють в чужі руки. Зловмисники, які отримують фізичний доступ до комп'ютера чи мобільного пристрою, можуть без зусиль скористатися збереженими даними або навіть доступом до корпоративних ресурсів через відсутність належних

		механізмів захисту. Використання системи шифрування та регулярне оновлення програмного забезпечення значно знижує ризик витоку інформації у випадку втрати пристрою
	Вразливості програмного забезпечення	Локальні пристрої часто зберігають старі версії програмного забезпечення, які можуть мати невиправлені вразливості. Відсутність актуальних оновлень операційних систем та програмних продуктів на таких пристроях може створювати можливості для зловмисників використовувати їх для виконання атак. Це може бути використано для інсталяції шкідливих програм (наприклад, вірусів, троянів чи шпигунських програм), що дозволяють перехоплювати або модифікувати дані, що зберігаються на пристрої. Регулярне оновлення програмного забезпечення та використання сучасних антивірусних рішень є необхідними для забезпечення базової безпеки даних

продовження таблиці 2.5

	Зберігання даних на незашифрованих носіях	Існує ризик зберігання персональних даних на незашифрованих зовнішніх носіях, таких як USB-флешки або зовнішні жорсткі диски. Якщо такі носії не захищені шифруванням, дані можуть бути легко вкрадені або доступні стороннім особам. Це є особливо актуальним, коли користувачі переносить інформацію між різними пристроями, не враховуючи можливі ризики. Для запобігання цим загрозам рекомендується використовувати шифрування як на зовнішніх носіях, так і на самих пристроях
	Залишкові дані на пристроях	Крім того, важливим нюансом є залишкові дані, що можуть залишатися на пристрої після видалення файлів. Невикористовувані або видалені файли можуть залишатися на дисках, і їх можна відновити за допомогою спеціалізованих інструментів. Зловмисники можуть скористатися цією вразливістю для відновлення інформації, що була на пристрої. Використання програм для безпечного стирання даних та регулярна очистка пристроїв допомагає мінімізувати цей ризик
	Вразливість до атак на локальну мережу	Коли пристрій підключається до незахищених або недостатньо захищених локальних мереж (наприклад, Wi-Fi мережі в публічних місцях), він може стати мішенню для атак. Це може бути як перехоплення трафіку, так і більш складні атаки, що дозволяють отримати доступ до даних. Для зниження ризику рекомендується використовувати

		VPN-з'єднання для захисту трафіку та з’єднуватися лише з надійними мережами

Загрози, пов’язані зі збереженням даних на локальних пристроях, є одними з найбільш актуальних для віддаленої роботи. Несанкціонований доступ, крадіжка пристроїв, вразливості програмного забезпечення, незашифровані носії та залишкові дані можуть призвести до серйозних наслідків для безпеки персональних та корпоративних даних. Забезпечення належного рівня захисту даних, таких як шифрування, оновлення програмного забезпечення, використання двоетапної аутентифікації та обережність при підключенні до мереж, є важливими кроками для мінімізації ризиків та захисту інформації в умовах віддаленої роботи.

Висновки за розділом 2

Загрози безпеці персональних даних при віддаленій роботі є численними і різноманітними, охоплюючи як технічні, так і організаційні складові захисту інформації. У процесі аутентифікації та авторизації кожен етап вимагає особливої уваги для запобігання атак, що можуть призвести до компрометації даних. Використання незахищених мереж ставить під загрозу конфіденційність інформації, адже атаки на мережевому рівні можуть бути непомітними, але вкрай небезпечними. Хмарні сервіси, хоча й спрощують робочі процеси, також відкривають нові можливості для зловмисників, вимагаючи ретельного контролю та надійних заходів захисту. Водночас, зловмисне програмне забезпечення залишається одним із найбільших ворогів безпеки в умовах віддаленої роботи, здатним завдати серйозної шкоди через порушення цілісності та доступності даних. Окремої уваги заслуговує збереження даних на локальних пристроях, де кожен фактор, від крадіжки до неконтрольованого доступу, може призвести до серйозних наслідків для організації та її співробітників.

Таким чином, кожна загроза вимагає застосування комплексних заходів

для її нейтралізації, починаючи від базових методів аутентифікації і закінчуючи використанням сучасних технологій шифрування та моніторингу. Усі ці питання вимагають постійної уваги і систематичного оновлення практик безпеки для забезпечення захисту персональних даних в умовах віддаленої роботи. Задля підтримки високого рівня безпеки важливо забезпечити інтеграцію технічних рішень із правильними організаційними політиками, що дозволяють запобігти і швидко реагувати на можливі загрози.

РОЗДІЛ 3
РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО ЗАХИСТУ ПЕРСОНАЛЬНИХ ДАНИХ ПРИ ВІДДАЛЕНОЇ РОБОТІ

3.1 Аналіз вимог до захисту персональних даних при віддаленій роботі

В умовах цифровізації та швидкого розвитку технологій віддалена робота стала важливою складовою сучасного робочого процесу. Разом з тим, цей формат праці ставить під загрозу безпеку персональних даних, які необхідно захищати від несанкціонованого доступу та потенційних витоків. Враховуючи значення персональних даних для забезпечення конфіденційності та довіри в бізнесі, важливо чітко окреслити вимоги до їх захисту під час віддаленої роботи. У цьому контексті важливим є дотримання законодавчих норм та застосування сучасних технологічних рішень, що гарантують безпеку обробки та зберігання таких даних.

3.1.1. Вимоги до автентифікації та авторизації

Для ефективного захисту персональних даних при віддаленій роботі особливо важливою є безпека процесів автентифікації та авторизації користувачів. Це дозволяє уникнути несанкціонованого доступу до систем та баз даних. Один з найбільш надійних методів захисту — двофакторна автентифікація (2FA), яка включає використання як пароля, так і додаткового коду, що генерується на мобільному пристрої користувача. Впровадження цього механізму дозволяє суттєво зменшити ризик зламу облікових записів.

Належна автентифікація передбачає також використання складних паролів, які повинні відповідати вимогам до довжини, складності та регулярної зміни. Для забезпечення додаткового захисту паролів їх хешують за допомогою алгоритмів, які унеможливлюють доступ до відкритого тексту пароля, навіть у

разі компрометації бази даних.

3.1.2. Захист передавання даних

В умовах віддаленої роботи передавання персональних даних через інтернет є потенційно вразливим етапом. Для того, щоб забезпечити безпечне передавання даних, використовують технології шифрування. Використання протоколу HTTPS для захисту комунікацій між клієнтом та сервером є стандартом для забезпечення конфіденційності та цілісності даних.

Шифрування даних на етапі передавання дозволяє уникнути їх перехоплення чи модифікації зловмисниками. Застосування SSL/TLS сертифікатів забезпечує надійний захист передачі конфіденційної інформації, що особливо важливо при роботі з персональними даними, які можуть бути використані для несанкціонованих дій, таких як шахрайство чи викрадення особистої інформації.

3.1.3. Контроль доступу та моніторинг активності

Не менш важливою вимогою є здійснення належного контролю доступу до інформаційних систем. Це передбачає обмеження доступу до персональних даних виключно для уповноважених осіб, а також моніторинг та аудит всіх операцій з даними. Встановлення прав доступу на рівні ролей, де кожен користувач отримує доступ лише до тих даних, які необхідні для виконання його обов'язків, є важливим етапом у забезпеченні безпеки.

Щоб вчасно виявляти потенційні загрози, слід запровадити систему моніторингу та аудиту всіх дій користувачів. Це дозволить виявити будь-які несанкціоновані спроби доступу чи маніпуляцій з персональними даними. Збереження журналів активності користувачів є необхідним для подальшого розслідування інцидентів безпеки та для документування всіх змін.

3.1.4. Захист від фізичних загроз

Незважаючи на те, що основний фокус у забезпеченні захисту персональних даних при віддаленій роботі зазвичай ставиться на програмні засоби, важливо не забувати і про фізичні елементи безпеки. Це стосується захисту пристроїв, на яких зберігаються або обробляються персональні дані. Працівники, які працюють віддалено, повинні дотримуватися основних принципів фізичної безпеки, зокрема уникати роботи з конфіденційною інформацією на публічних чи незахищених мережах, використовувати шифрування на своїх пристроях та регулярно їх оновлювати.

Для зменшення ризиків потрібно забезпечити належний рівень захисту кінцевих пристроїв (комп'ютерів, смартфонів тощо), де можуть зберігатися чутливі дані. Використання програм для шифрування жорстких дисків та резервне копіювання даних є необхідними заходами для збереження даних від фізичних загроз.

3.1.5. Відповідність законодавчим вимогам

Захист персональних даних регулюється низкою законодавчих актів, зокрема такими, як Загальний регламент захисту даних (GDPR) в Європейському Союзі, Закон України "Про захист персональних даних" та інші. Важливо, щоб організації, що здійснюють обробку персональних даних, відповідали вимогам цих нормативних актів.

Враховуючи високий рівень довіри до компаній, які виконують ці вимоги, це є важливим питанням не лише з точки зору безпеки, а й для збереження репутації організації. Для забезпечення відповідності законодавству слід регулярно проводити аудити безпеки та реалізувати відповідні заходи, спрямовані на захист персональних даних.

Аналіз вимог до захисту персональних даних при віддаленій роботі показує, що для забезпечення безпеки даних необхідно враховувати

комплексний підхід, який включає автентифікацію користувачів, шифрування передавання даних, контроль доступу, фізичну безпеку пристроїв, а також відповідність законодавчим вимогам. Застосування цих заходів дозволяє мінімізувати ризики витоку інформації та забезпечити належний рівень захисту персональних даних при віддаленій роботі.

3.2 Розробка концепції програмного проекту

Захист персональних даних є критично важливою складовою безпеки інформаційних систем, особливо в умовах віддаленої роботи, яка стрімко набирає популярності. З ростом числа інтернет-загроз та зловмисних дій, які можуть спричинити витік особистої інформації, важливою є розробка надійних механізмів безпеки, які гарантують конфіденційність, цілісність і доступність персональних даних. Концепція програмного проекту, спрямованого на захист таких даних, повинна включати методи автентифікації, забезпечення конфіденційності через шифрування, а також ефективні способи моніторингу та аудиту діяльності користувачів.

Безпека персональних даних у процесі віддаленої роботи має враховувати низку важливих питань, представлені в таблиці 3.1.

Таблиця 3.1

Питання безпеки персональних даних у процесі віддаленої роботи

№	Назва	Опис
1	**2**	**3**
1	Конфіденційність	Персональні дані повинні бути захищені від несанкціонованого доступу як в процесі зберігання, так і при передачі
2	Ідентифікація та автентифікація	Процес доступу до даних має ґрунтуватися на надійних механізмах ідентифікації користувачів, що дозволяє мінімізувати ймовірність використання підроблених облікових записів
3	Цілісність даних	Важливо забезпечити збереження точності та повноти даних без змін, викликаних зовнішніми факторами

продовження таблиці 3.1

1	2	3
4	Доступність	Всі дані, що використовуються у віддаленій роботі, повинні бути доступні лише авторизованим користувачам у встановлені терміни
5	Моніторинг та аудит	Необхідно постійно відстежувати дії користувачів та вести журнал активності для своєчасного виявлення порушень

Для забезпечення захисту даних необхідно розробити систему автентифікації, що включає фактори, описані в таблиці 3.2.

Таблиця 3.2

Фактори системи автентифікації та управління доступом

№	Назва	Опис
1	Міцні паролі	Оскільки використання слабких паролів є однією з основних причин витоків даних, важливо реалізувати функцію валідації паролів, яка не дозволяє користувачам обирати небезпечні паролі. Крім того, застосування технік хешування та соління паролів значно підвищує рівень безпеки
2	Двофакторна автентифікація (2FA)	Для підвищення рівня безпеки доступу до персональних даних необхідно інтегрувати механізм двоетапної автентифікації, що знижує ризик несанкціонованого доступу навіть у випадку компрометації пароля
3	Рольова модель доступу	Система повинна бути побудована таким чином, щоб доступ до даних та функцій був обмежений відповідно до ролі користувача в організації, тим самим мінімізуючи потенційні ризики

Один із найефективніших способів захисту конфіденційності даних — це їх шифрування, що включає напрямки описані в таблиці 3.3.

Таблиця 3.3

Напрямки шифрування даних

№	Назва	Опис

1	Шифрування даних при передачі	Використання протоколів, таких як HTTPS (SSL/TLS), гарантує, що всі дані, що передаються між користувачем та сервером, будуть зашифровані і захищені від перехоплення

продовження таблиці 3.3

2	Шифрування даних на стороні зберігання	Для збереження даних в базах даних також необхідно впровадити методи їх шифрування, щоб навіть у разі несанкціонованого доступу до серверу зловмисники не могли безпосередньо прочитати чутливу інформацію

Одним із основних методів виявлення порушень безпеки є ведення журналу активності користувачів, що включають компоненти описані в таблиці 3.4.

Таблиця 3.4

Компоненти Журналу активності користувачів

№	Назва	Опис
1	Журналювання дій користувачів	Система повинна реєструвати всі значущі дії користувачів, зокрема спроби входу, зміни паролів, додавання чи оновлення персональних даних. Це дозволяє не лише відстежувати потенційні спроби несанкціонованого доступу, але й зберігати важливу інформацію для подальшого аналізу в разі інциденту
2	Аудит активності	Окрім журналу дій, необхідно створити систему аудиту, яка дозволить на етапі розслідування виявити порушення безпеки та можливі точки уразливості. Це також допомагає в розробці аналітики для запобігання майбутнім інцидентам

Незважаючи на впроваджені механізми захисту, система має бути здатною протистояти різним типам атак, що описані в таблиці 3.5.

Таблиця 3.5

Типи атак

№	Назва	Опис
1	Атаки на автентифікацію	Наприклад, атаки методом підбору пароля (brute-force). Для їх запобігання система повинна мати обмеження на кількість спроб введення неправильного пароля, а також застосовувати капчу для захисту від автоматичних атак

2	Атаки на веб-додатки	Важливим є захист від SQL-ін'єкцій, міжсайтових атак (XSS), атак CSRF та інших вразливостей веб-додатків. Для цього необхідно використовувати захищені методи доступу до бази даних і коректну обробку вводу користувача

Важливо не лише передбачити механізми захисту, але й забезпечити ефективне реагування на інциденти, що представлені в таблиці 3.6.

Таблиця 3.6

Інциденти та відновлення після атак

№	Назва	Опис
1	Виявлення вторгнень	Впровадження систем моніторингу для виявлення підозрілої активності (наприклад, невдалі спроби входу або зміни даних
2	Відновлення даних після атак	Для відновлення даних у разі їх втрати чи пошкодження необхідно регулярно створювати резервні копії, що зберігаються в захищених місцях, та проводити тести відновлення

Розробка концепції програмного проекту для захисту персональних даних при віддаленій роботі повинна ґрунтуватися на комплексному підході, що включає використання ефективних механізмів автентифікації, шифрування даних, аудит і моніторинг активності користувачів, а також захист від атак та налаштування відновлення після інцидентів. Інтеграція цих компонентів створює основу для надійного захисту персональних даних, що є критично важливим в умовах постійного розвитку технологій і зростаючих кіберзагроз.

3.3 Реалізація організаційних заходів у програмному проекті

В умовах поширення віддаленої роботи організаціям необхідно забезпечувати надійний захист персональних даних своїх співробітників і користувачів. Це вимагає комплексного підходу, що включає як технічні, так і організаційні заходи, спрямовані на мінімізацію ризиків витоку або несанкціонованого доступу до чутливої інформації. Організаційні заходи в

межах програмного проекту забезпечують ефективне управління доступом до даних, контролюють безпеку інформаційних систем і гарантують виконання стандартів та регламентів, які стосуються захисту персональних даних.

Організаційні заходи представлені в таблиці 3.7.

Таблиця 3.7

Організаційні заходи у програмному проекті

№	Назва	Опис
1	**2**	**3**
1	Визначення політики доступу та управління ролями користувачів	Одним із ключових елементів організаційних заходів є визначення чіткої політики доступу до персональних даних. Для цього необхідно впровадити механізм, що регулює рівні доступу до інформації на основі ролей користувачів. Кожному користувачеві надаються лише ті права, які необхідні для виконання його робочих обов’язків. Ролі користувачів мають бути детально прописані й обмежені доступом до чутливої інформації, що запобігає несанкціонованому доступу до персональних даних. Так, для адміністратора можуть бути передбачені розширені права доступу, тоді як для звичайного користувача — лише можливість переглядати власні дані
2	Управління автентифікацією та авторизацією користувачів	Для забезпечення безпеки віддаленого доступу до системи необхідно впровадити багаторівневу автентифікацію та авторизацію. Одним із таких заходів є використання двофакторної автентифікації (2FA). Вона дозволяє додатково захистити користувацькі акаунти шляхом введення одноразового коду, що генерується спеціальним додатком або надсилається через SMS. Крім того, важливо забезпечити процедуру зміни пароля, що має бути регулярною і обов'язковою для користувачів. Всі ці заходи забезпечують високий рівень безпеки та знижують ризик несанкціонованого доступу до персональних даних
3	Логування та моніторинг активності користувачів	Важливим організаційним заходом є створення системи логування та моніторингу всіх операцій з персональними даними. Це дозволяє фіксувати всі дії користувачів у системі та своєчасно виявляти будь-які підозрілі або несанкціоновані операції. Журналювання входів і виходів, зміни даних та інших критичних дій допомагає не лише контролювати доступ до даних, а й виконувати аудит безпеки системи. У випадку виникнення інцидентів безпеки моніторинг дає змогу

		оперативно реагувати та відновлювати контроль над даними

продовження таблиці 3.7

4	Реалізація заходів щодо захисту даних при передачі та зберіганні	Організація безпечної передачі та зберігання персональних даних є важливим напрямком у реалізації організаційних заходів. Для захисту даних під час передачі необхідно використовувати протоколи захищеного з'єднання, такі як HTTPS, що забезпечує шифрування переданої інформації. Також необхідно вжити заходів для захисту даних, що зберігаються в базах даних, за допомогою методів шифрування. Важливо, щоб доступ до зашифрованих даних був можливий тільки за умови наявності відповідних прав доступу
5	Резервне копіювання та відновлення даних	Регулярне резервне копіювання даних є важливим заходом для забезпечення їх безпеки в разі несанкціонованого доступу, помилок або збоїв у системі. Організація повинна встановити чіткий графік для створення резервних копій персональних даних, зберігаючи їх у безпечному місці, яке є доступним лише для авторизованих осіб. Важливо також мати план відновлення даних у випадку їх втрати або пошкодження
6	Навчання та підвищення обізнаності користувачів	Забезпечення безпеки персональних даних не обмежується лише технологічними заходами, важливим елементом є також навчання та підвищення обізнаності користувачів щодо найкращих практик безпеки. Користувачі повинні бути ознайомлені з політиками безпеки, правилами захисту даних та важливістю збереження конфіденційності інформації. Регулярні тренінги та тестування на знання правил безпеки допомагають створити культуру відповідальності за захист персональних даних серед співробітників
7	Регулярні перевірки та аудит безпеки	Щоб гарантувати відповідність установленим стандартам і виявляти можливі вразливості, організація повинна проводити регулярні перевірки та аудит безпеки системи. Аудит включає в себе оцінку ефективності поточних заходів захисту, перевірку відповідності політик безпеки та виявлення потенційних загроз. Важливо, щоб аудит проводився не тільки для оцінки безпеки системи, а й для коригування заходів, що мають на меті підвищення захисту персональних даних

8	Управління інцидентами та реагування на загрози	Для забезпечення швидкої та ефективної реакції на можливі інциденти безпеки необхідно створити процедури управління інцидентами. Ці процедури повинні включати чітке визначення ролей та відповідальностей, а також поетапний план дій для реагування на загрози. У разі виявлення загрози чи порушення безпеки персональних даних, організація повинна негайно вжити заходів для мінімізації шкоди, включаючи інформування постраждалих користувачів та відповідних органів

Забезпечення захисту персональних даних при віддаленій роботі є складним і багатоетапним процесом, що вимагає чіткої організації заходів безпеки на всіх етапах обробки даних. Реалізація організаційних заходів дозволяє не лише гарантувати безпеку інформації, але й створює атмосферу довіри серед користувачів. Ретельно сплановані та реалізовані заходи забезпечують захист від несанкціонованого доступу, знижують ризики витоку даних і сприяють відповідальній обробці персональних даних у середовищі віддаленої роботи.

3.4 Реалізація технічних заходів у програмному проекті

Захист персональних даних є ключовим компонентом будь-якої сучасної інформаційної системи, особливо в контексті віддаленої роботи, де взаємодія з користувачами відбувається через мережу. Зважаючи на зростаючу кількість загроз і атак на персональні дані, необхідно впроваджувати надійні технічні заходи для їх захисту. Це забезпечує не лише захист конфіденційності користувачів, але й підтримку довіри до системи. Оскільки в сучасних умовах особисті дані часто стають ціллю для кіберзлочинців, реалізація надійних методів їх захисту є критично важливою.

У програмному проекті, що передбачає обробку персональних даних при віддаленій роботі, застосовано низку технічних заходів для забезпечення їх захисту. Далі будуть розглянуті основні технічні засоби, що були реалізовані у цьому проекті.

Реалізація технічних заходів представлена в таблиці 3.8.

Таблиця 3.8

Технічні заходи у програмному проекті

№	Назва	Опис
1	**2**	**3**
1	Хешування паролів та використання солей	Один із найважливіших технічних заходів для захисту персональних даних користувачів — це хешування паролів. У разі збереження паролів у відкритому вигляді створюється суттєва загроза їх крадіжки та подальшого використання зловмисниками. Тому в рамках проекту використовується хешування паролів із додаванням унікальної солі, що ускладнює можливість відновлення початкового пароля навіть за умови компрометації бази даних. Для кожного користувача генерується соляний ключ, що додається до пароля перед його хешуванням. Це означає, що навіть якщо два користувачі обирають однакові паролі, їх хеші в базі даних будуть різними завдяки унікальним солям. Таким чином, цей метод ефективно протидіє атакам за допомогою попередньо обчислених хешів (наприклад, використання таблиць для злому паролів типу "rainbow tables").
2	Використання багатофакторної автентифікації (2FA)	Для посилення безпеки доступу до системи важливою частиною є багатофакторна автентифікація. Цей метод забезпечує додатковий рівень захисту, знижуючи ризик несанкціонованого доступу навіть у разі компрометації пароля. В рамках даного проекту реалізовано налаштування двофакторної автентифікації (2FA), що використовує одноразові коди, згенеровані за допомогою спеціального додатку, такого як Google Authenticator або Authy. Під час налаштування 2FA користувач вводить секретний ключ, який використовується для генерації кодів доступу. Тільки після успішного введення цього коду система надає доступ до ресурсів. Це значно ускладнює можливість отримання доступу до

		облікових записів без фізичного доступу до пристрою користувача

продовження таблиці 3.8

1	2	3
3	Протокол HTTPS для безпечної передачі даних	Для захисту персональних даних під час передачі по мережі використовуються зашифровані канали зв'язку. Це досягається за допомогою протоколу HTTPS (Hypertext Transfer Protocol Secure), що є безпечним варіантом стандартного HTTP. Використання цього протоколу дозволяє забезпечити шифрування даних, які передаються між клієнтом та сервером, що запобігає їх перехопленню або зміні під час передачі через незахищені мережі. Протокол HTTPS гарантує, що інформація, включаючи особисті дані, передається в зашифрованому вигляді, що робить її недоступною для сторонніх осіб, навіть якщо вони мають доступ до каналу зв'язку. Це забезпечує важливу безпеку при роботі в умовах віддаленого доступу
4	Логування та аудит активності	Для забезпечення цілісності та прозорості доступу до даних використовуються механізми журналювання та аудиту дій користувачів. У проекті реалізовано ведення аудиту, що фіксує всі важливі дії користувачів — реєстрацію, входи в систему, зміни паролів, налаштування двофакторної автентифікації тощо. Це дозволяє виявити будь-які несанкціоновані дії та спроби зловживань, а також допомагає зберігати історію змін для подальшого аналізу та розслідувань. Логи активності зберігаються в спеціальних таблицях бази даних і містять важливу інформацію, таку як тип дії, IP-адреса користувача, дані про пристрій та браузер, з якого виконувалась дія. Такі логи також є корисними для моніторингу безпеки та виявлення підозрілих активностей
5	Обмеження доступу до персональних даних	Ще одним важливим технічним заходом є обмеження доступу до персональних даних на рівні бази даних. У проекті реалізовано механізм авторизації, що забезпечує доступ до даних лише тим користувачам, які мають

		відповідні права. Це гарантує, що користувачі можуть отримувати або змінювати тільки свої особисті дані, а не чужі. Крім того, система дозволяє реалізувати різні рівні доступу, що дає можливість адміністратору управляти правами доступу для різних груп користувачів. Це є важливим фактором для зниження ризику витоку інформації або випадкового порушення конфіденційності даних

завершення таблиці 3.8

6	Шифрування особистих даних	Шифрування є одним з основних методів забезпечення конфіденційності даних. У проекті передбачено шифрування особистих даних користувачів на рівні бази даних. Це означає, що навіть у разі компрометації бази даних зловмисник не зможе прочитати дані без відповідного ключа шифрування. Використання шифрування є одним із найефективніших способів захисту конфіденційної інформації, оскільки шифровані дані не можуть бути доступні або використані без відповідного ключа для їх розшифровки

Запровадження технічних заходів захисту персональних даних є необхідною умовою для забезпечення їх безпеки, особливо в умовах віддаленої роботи. Використання таких механізмів, як хешування паролів, багатофакторна автентифікація, протокол HTTPS, аудит активності, обмеження доступу та шифрування, забезпечує надійний захист особистих даних користувачів. Всі ці заходи не лише сприяють підвищенню безпеки, але й підтримують довіру користувачів до системи, гарантуючи, що їх дані залишаються захищеними від несанкціонованого доступу та використання.

3.5 Тестування та впровадження програмного проекту

Процес тестування та впровадження програмного проекту є важливою частиною розробки програмного забезпечення, що сприяє забезпеченню безпеки та функціональності продукту. Особливо це стосується проектів, які

опрацьовують чутливу інформацію, таку як персональні дані, в умовах віддаленої роботи. Тестування дозволяє виявити потенційні вразливості в коді, а впровадження забезпечує безпечний та ефективний перехід до використання програмного забезпечення в реальному середовищі.

Тестування безпеки є ключовим етапом для забезпечення надійного захисту персональних даних (таблиця 3.9).

Таблиця 3.9

Напрямки тестування безпеки

№	Назва	Опис
1	Аутентифікація та авторизація користувачів	тестується система перевірки користувачів, зокрема наявність таких механізмів, як перевірка правильності пароля, реєстрація нових користувачів і підтвердження електронної пошти. Важливим є тестування механізмів блокування або обмеження доступу до даних для користувачів, які не авторизовані або не мають відповідних прав доступу
2	Захист від атак	тестуються різні види атак, зокрема SQL-ін'єкції, CSRF (Cross-Site Request Forgery), XSS (Cross-Site Scripting) та інші. Для цього застосовуються автоматизовані сканери та ручне тестування для перевірки наявності уразливих місць, де дані можуть бути компрометовані
3	Захист даних	перевіряється механізм шифрування паролів та персональних даних. Зокрема, необхідно впевнитися, що пароль користувача зберігається в зашифрованому вигляді, а також перевіряються механізми зберігання та передачі персональних даних (наприклад, використання HTTPS для передачі даних між клієнтом і сервером)

Функціональне тестування полягає у перевірці всіх основних функцій системи (таблиця 3.10).

Таблиця 3.10

Напрямки функціонального тестування

№	Назва	Опис
1	Реєстрація та авторизація	тестування процесу реєстрації нових користувачів, коректності введення обов'язкових даних (логін,

		пароль, електронна пошта), перевірка правильності повідомлень про помилки
2	Додавання та оновлення персональних даних	перевірка процесу введення, збереження та редагування персональних даних, а також правильності роботи механізмів валідації (наприклад, правильність номеру телефону чи імені)
3	Зміна пароля та налаштування двофакторної аутентифікації (2FA)	перевірка всіх сценаріїв, що пов'язані зі зміною пароля, зокрема правильність обробки поточного та нового пароля, перевірка працездатності 2FA при налаштуванні

Навантажувальне тестування перевіряє, як система реагує на підвищене навантаження (таблиця 3.11). Це особливо важливо в контексті віддаленої роботи, коли одночасно можуть працювати десятки або навіть сотні користувачів.

Таблиця 3.11

Напрямки навантажувального тестування

№	**Назва**	**Опис**
1	**2**	**3**
1	Масштабованість	тестується здатність системи витримувати великі обсяги запитів без значних втрат у швидкості обробки. Перевіряється коректність роботи при великій кількості одночасних користувачів, що виконують операції з реєстрацією, входом, зміною пароля, додаванням персональних даних
2	Перевірка на стійкість до помилок	система повинна правильно обробляти помилки, такі як збої під час зберігання даних або неправильно введені дані користувачем. Це включає автоматичні перевірки на наявність можливих помилок у програмному коді, а також перевірки для забезпечення стабільності при збої в мережі чи інших непередбачених ситуаціях

Хоча безпека та функціональність є основними критеріями для програмного забезпечення, тестування зручності користування також має велике значення (таблиця 3.12). Користувачі, які працюють віддалено, повинні мати інтуїтивно зрозумілий інтерфейс і можливість швидко виконувати свої завдання без складних або незрозумілих кроків.

Таблиця 3.12

Напрямки тестування зручності та юзабіліті

№	Назва	Опис
1	Інтерфейс користувача	тестується зручність і доступність інтерфейсу для користувачів, зокрема перевіряється логіка навігації між сторінками, коректність виведення помилок, а також оптимізація інтерфейсу для різних пристроїв

продовження таблиці 3.12

2	Процес реєстрації та входу	перевіряється зручність і простота процесу реєстрації нових користувачів та входу в систему. Користувач повинен отримувати чіткі інструкції щодо необхідних дій, а також відповідні повідомлення про помилки

Процес впровадження є етапом, коли програмне забезпечення стає доступним для користувачів. Він включає кілька важливих кроків, що описані в таблиці 3.13.

Таблиця 3.13

Кроки впровадження в реальне середовище

№	Назва	Опис
1	Розгортання програмного забезпечення	після тестування програма розгортається на сервері, підключеному до продуктивного середовища. Це може включати налаштування веб-сервера, бази даних, а також забезпечення відповідних мережевих з'єднань для забезпечення доступу
2	Налаштування системи безпеки	після розгортання слід переконатися, що всі безпекові механізми працюють належним чином. Це включає налаштування брандмауерів, сертифікатів SSL для захищеного з'єднання, а також перевірку наявності механізмів моніторингу за діяльністю користувачів
3	Міграція даних	якщо програмне забезпечення інтегрується з існуючими системами, необхідно перевірити, чи всі дані було коректно перенесено з попередніх баз даних. Це включає як тестування даних, так і перевірку сумісності нових і старих систем

Після впровадження проекту важливим є постійний моніторинг його роботи, виявлення потенційних вразливостей і надання оновлень для покращення безпеки та функціональності. Це може включати напрямки, описані в таблиці 3.14.

Таблиця 3.14

Напрямки підтримки та оновлення

№	Назва	Опис
1	Моніторинг логів та аналіз активності	регулярний перегляд журналів аудиту дозволяє виявити підозрілі дії або атаки на систему, а також забезпечити своєчасне реагування на загрози
2	Оновлення програмного забезпечення	регулярні оновлення програми для покращення безпеки, виправлення помилок або додавання нових функцій

Важливим елементом тестування є покрокове виконання програми.

Крок 1: Запуск програми

Програма розпочинає свою роботу з виконання головного модуля. Операційна система виділяє пам'ять для виконання коду, а також ініціалізує середовище виконання, необхідне для роботи інтерпретатора Python. Після цього виконується імпорт необхідних бібліотек, зокрема `socket`, `threading`, `cryptography`, які забезпечують мережеву взаємодію, багатозадачність і шифрування даних.

Крок 2: Ініціалізація налаштувань безпеки

Програма генерує асиметричні ключі за допомогою бібліотеки `cryptography`. Використовуються алгоритми RSA для створення пари відкритого та закритого ключів, які забезпечують шифрування та розшифрування даних. Закритий ключ зберігається локально, тоді як відкритий ключ надсилається клієнтам для безпечного обміну даними.

Крок 3: Створення серверного сокета

На сервері створюється сокет з використанням протоколу TCP, який забезпечує надійну передачу даних. Сокет прив'язується до IP-адреси та порту, визначених у налаштуваннях. Сервер переводиться у режим очікування вхідних з’єднань, використовуючи метод `listen()`.

Крок 4: Очікування вхідних з'єднань

Сервер очікує на вхідні з'єднання від клієнтів (рис. 3.1, рис.3.2, рис. 3.3, рис. 3.4, рис. 3.5, рис. 3.6, рис. 3.7, рис 3.8, рис. 3.9, рис. 3.10, рис. 3.11). Після отримання запиту від клієнта створюється новий потік для обробки цього підключення. Це забезпечує багатозадачність та дає можливість обслуговувати кілька клієнтів одночасно.

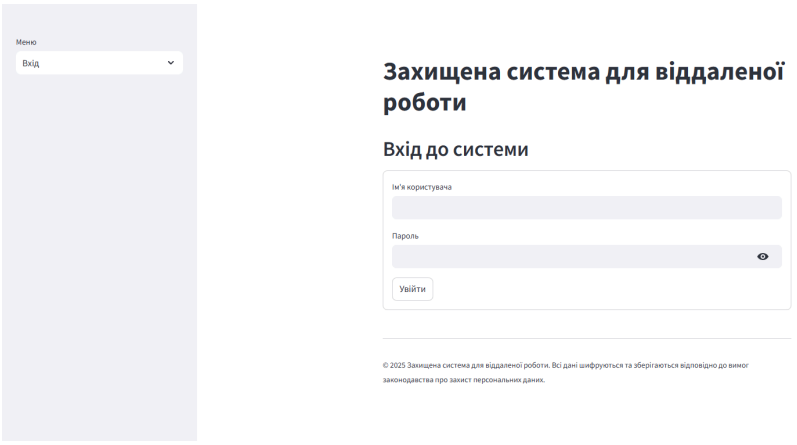


Рис. 3.1. Форма входу в систему

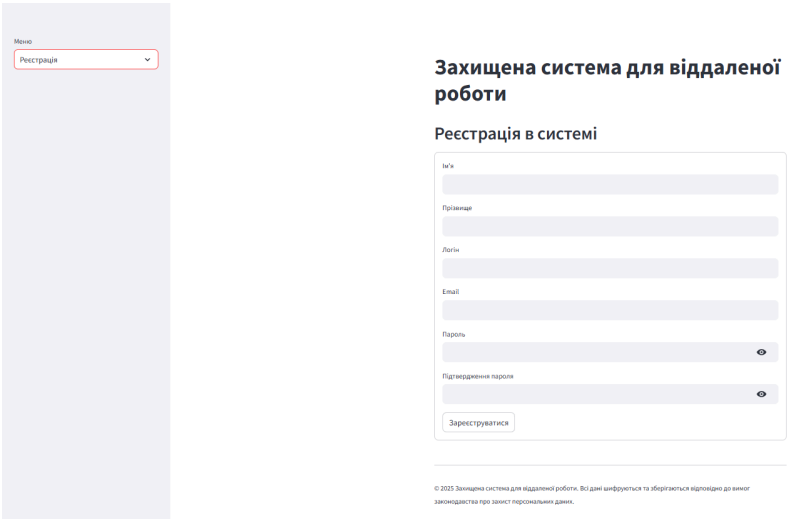


Рис. 3.2. Форма реєстрації до системи

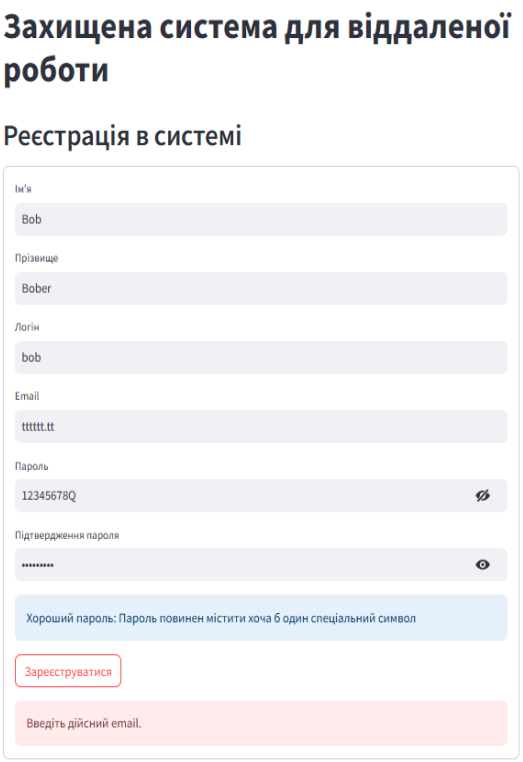


Рис. 3.3. Помилка при реєстрації – введено нереальну адресу електронної пошти

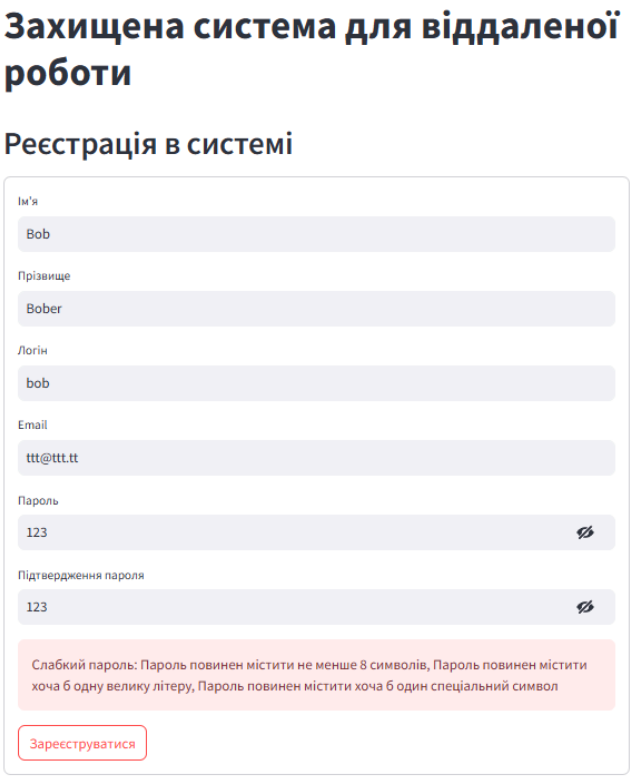


Рис. 3.4. Помилка при реєстрації – слабкий пароль, з рекомендаціями до правил формування паролю

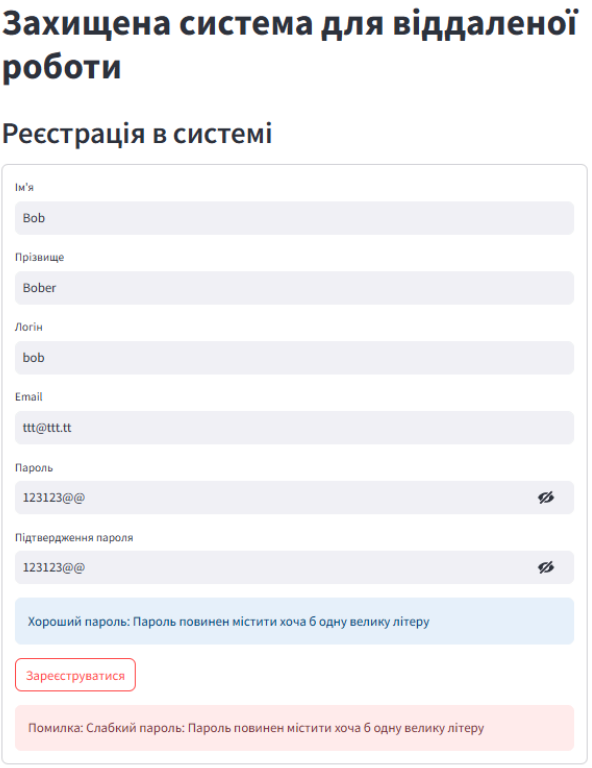


Рис. 3.5. Помилка при реєстрації – слабкий пароль: відсутність в паролі великої букви

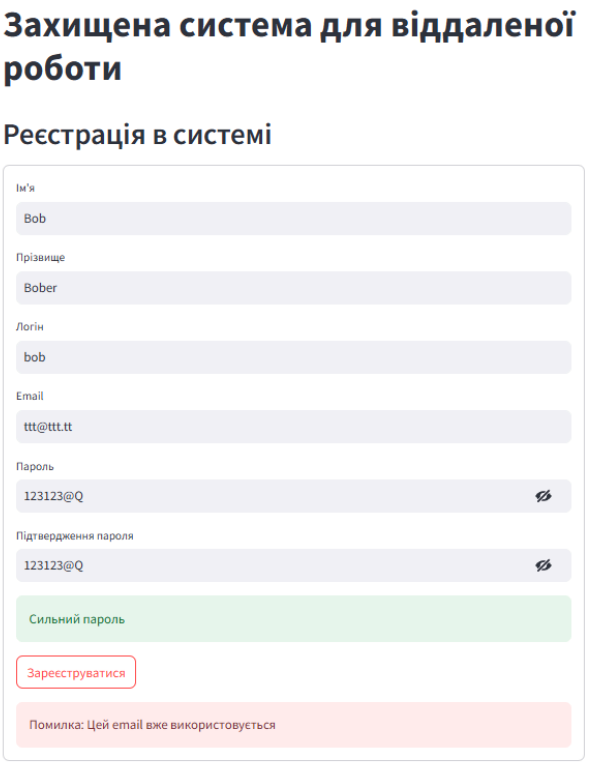


Рис. 3.6. Помилка при реєстрації – використано email, вже зареєстрований у системі

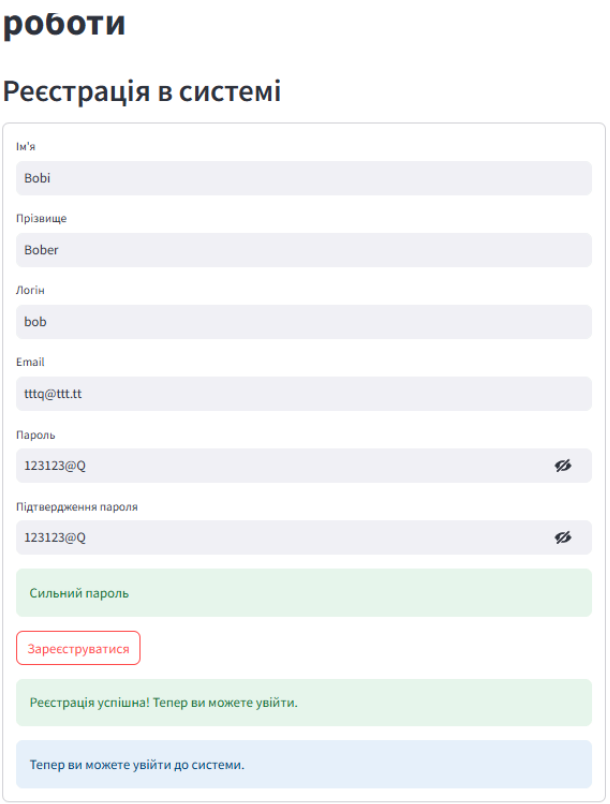


Рис. 3.7. Успіх реєстрації

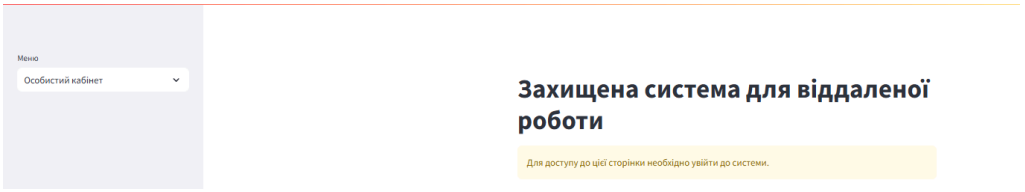


Рис. 3.8. Попередження, що для перегляду особистого кабінету, необхідно увійти в систему

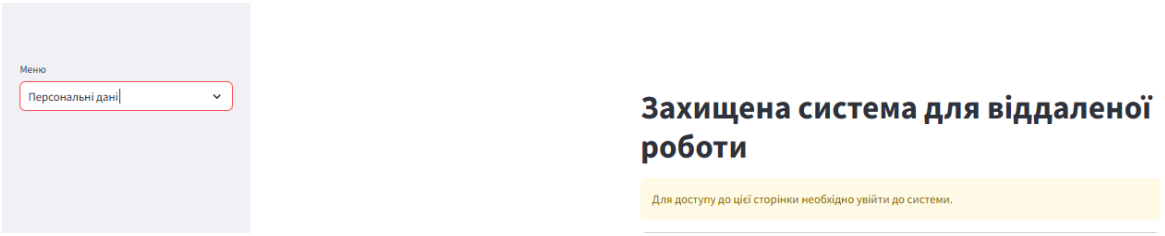


Рис. 3.9. Попередження, що для перегляду персональних даних, необхідно увійти в систему

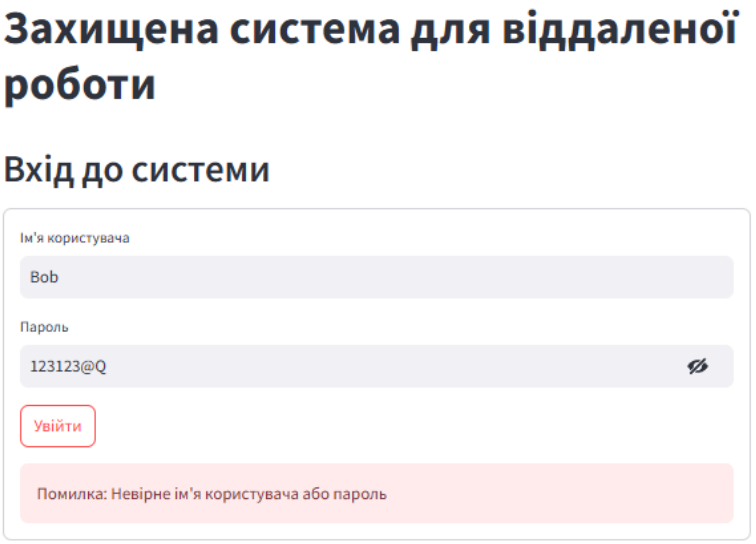


Рис. 3.10. Помилка входу в систему – невірне ім’я користувача

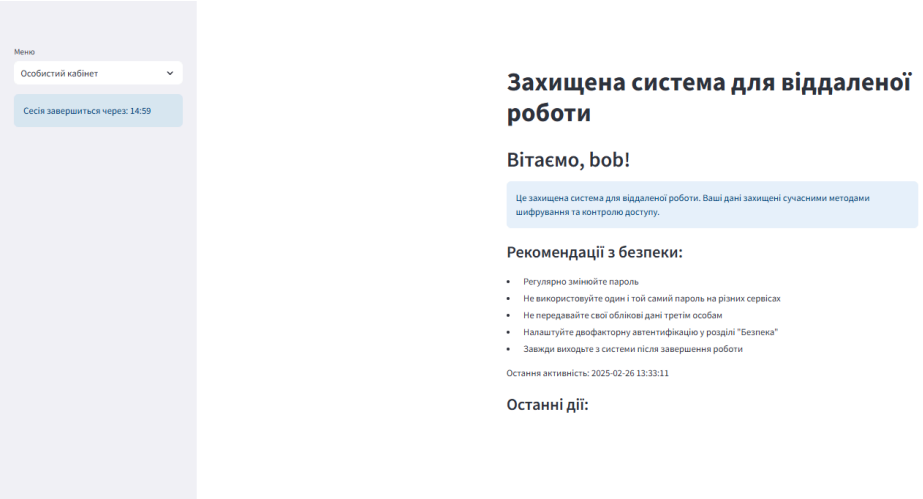


Рис. 3.11. Особистий кабінет користувача

Крок 5: Встановлення з'єднання з клієнтом

У новоствореному потоці сервер приймає вхідне з'єднання за допомогою методу `accept()` та отримує інформацію про клієнта (IP-адресу та порт). Після цього сервер надсилає клієнту свій відкритий ключ для подальшого безпечного обміну даними.

Крок 6: Ініціалізація клієнтського підключення

Клієнтська програма створює сокет для підключення до сервера. Відбувається з'єднання з сервером за вказаною IP-адресою та портом. Після встановлення з'єднання клієнт отримує відкритий ключ сервера. Після чого клієнт може редагувати персональні дані (рис. 3.12, рис. 3.13, рис. 3.14) та налаштовувати безпеку (рис. 3.15).

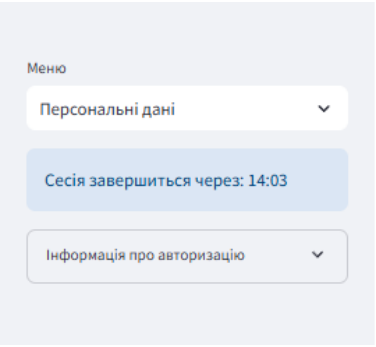


Рис. 3.12. Меню «Персональні дані»

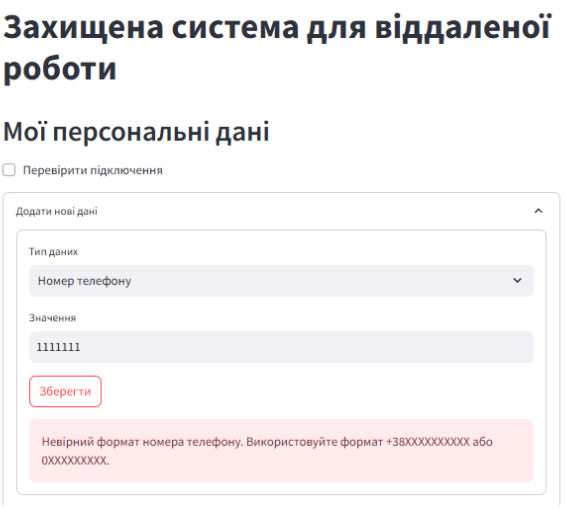


Рис. 3.13. Зміна номеру телефона. Помилка – Невірний формат номера телефону

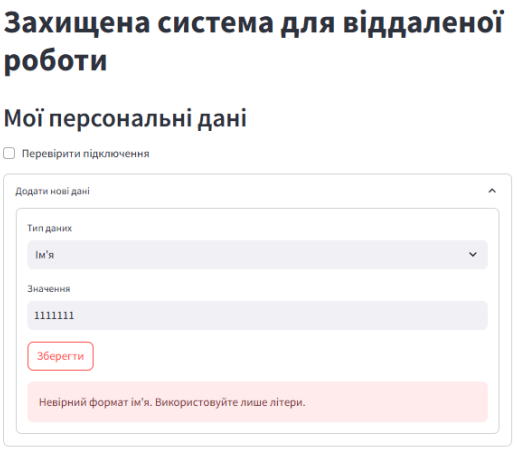


Рис. 3.14. Додавання ім’я. Помилка – невірний формат імені

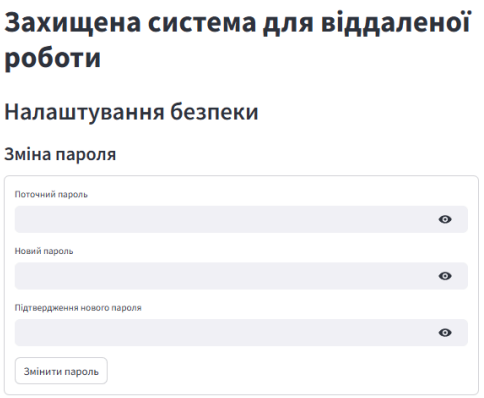


Рис. 3.15. Налаштування безпеки – зміна паролю

Крок 7: Шифрування даних клієнтом

Клієнт шифрує свої дані за допомогою отриманого відкритого ключа сервера (рис. 3.16). Використовується алгоритм RSA, що гарантує конфіденційність даних під час передачі через мережу (рис. 3.17).

Рис. 3.16. Отримання секретного ключа

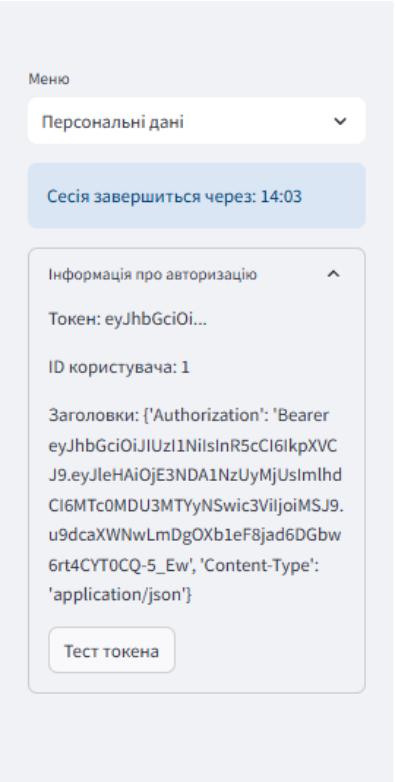


Рис. 3.17. Зашифрована інформація користувача

Крок 8: Передача зашифрованих даних на сервер

Клієнт надсилає зашифровані дані на сервер через TCP-з'єднання. Завдяки використанню протоколу TCP забезпечується надійна передача даних без втрат і пошкоджень.

Крок 9: Отримання та розшифрування даних сервером

Сервер отримує зашифровані дані від клієнта. За допомогою закритого ключа, який зберігається локально, сервер розшифровує ці дані. Таким чином, забезпечується конфіденційність інформації, оскільки лише сервер має доступ до закритого ключа.

Крок 10: Обробка отриманих даних

Сервер обробляє отримані дані відповідно до визначеної логіки програми. Це може включати їх збереження у базі даних, подальший аналіз або передачу іншим модулям системи.

Крок 11: Відправка відповіді клієнту

Після обробки даних сервер формує відповідь та шифрує її за допомогою відкритого ключа клієнта. Це гарантує, що відповідь може бути розшифрована лише клієнтом, який має відповідний закритий ключ (рис. 3.18).

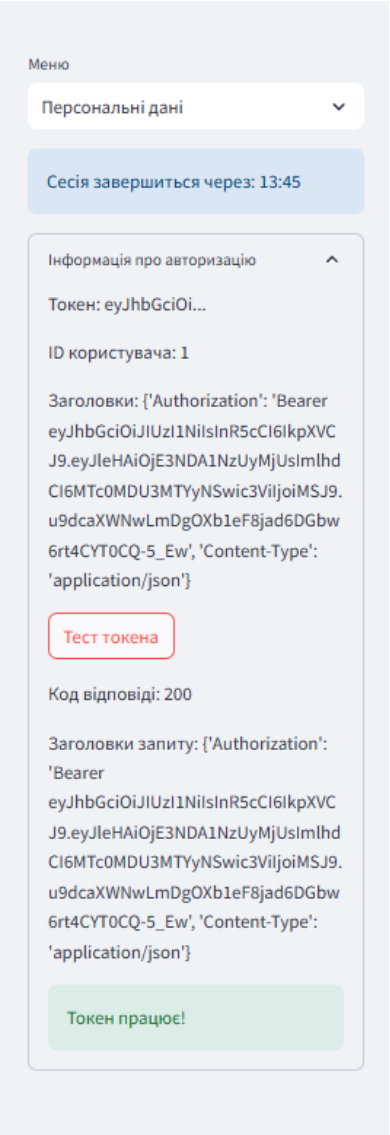


Рис. 3.18. Отримання зашифрованої відповіді

Крок 12: Розшифрування відповіді клієнтом

Клієнт отримує зашифровану відповідь від сервера. За допомогою свого закритого ключа клієнт розшифровує відповідь та виводить її на екран.

Крок 13: Завершення сеансу зв'язку

Після успішного обміну даними сервер і клієнт закривають сокети, що забезпечує звільнення ресурсів мережі та пам'яті. Також можна переглянути журанал активності (рис. 3.19). З’єднання вважається завершеним.

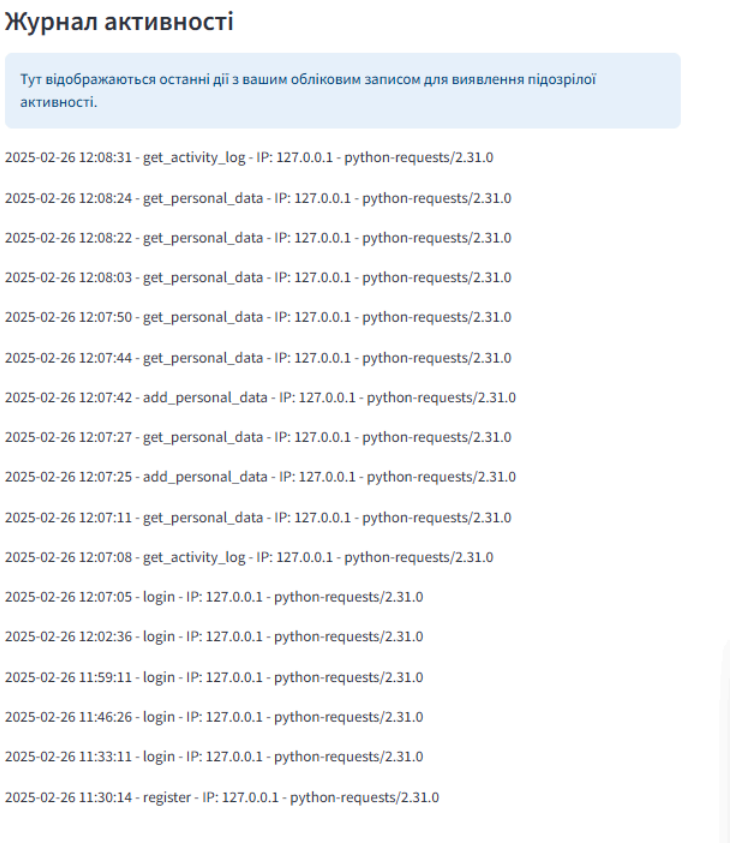


Рис. 3.19. Перегляд журналу активності

Крок 14: Завершення роботи програми

Після закриття з'єднання усі потоки завершують свою роботу (рис. 3.20). Інтерпретатор Python звільняє використані ресурси та завершує виконання програми.

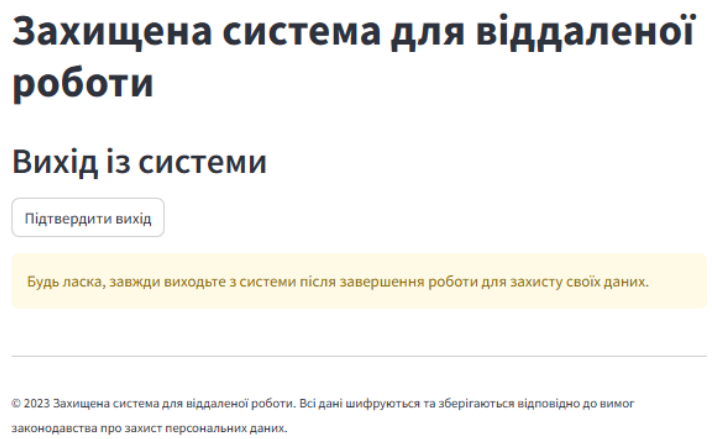


Рис. 3.20. Вихід із системи

Покроковий аналіз виконання програми дозволяє чітко простежити послідовність дій від запуску до завершення роботи, а також виявити ключові етапи забезпечення безпеки персональних даних під час віддаленої роботи. Програма реалізує надійний механізм захисту даних шляхом використання асиметричного шифрування, що гарантує конфіденційність та цілісність інформації. Дотримання описаних кроків дозволяє ефективно захистити персональні дані від несанкціонованого доступу під час їх передачі у мережі.

Процес тестування та впровадження є критично важливим етапом в розробці програмного проекту, особливо коли мова йде про захист персональних даних при віддаленій роботі. Успішне виконання цього етапу гарантує, що система буде безпечною, стабільною та ефективною, а користувачі зможуть безпечно працювати з чутливою інформацією в умовах віддаленого доступу.

Висновки за розділом 3

Захист персональних даних при віддаленій роботі є надзвичайно важливим завданням, оскільки зростаюча популярність дистанційних форматів взаємодії створює нові виклики для забезпечення конфіденційності та цілісності інформації. В умовах глобальної діджиталізації ризики несанкціонованого доступу до даних, їх втрати або викрадення суттєво зростають, що вимагає впровадження ефективних заходів безпеки. Відтак, важливо забезпечити комплексний підхід до захисту персональних даних, який охоплює як організаційні, так і технічні методи.

Забезпечення належного рівня безпеки вимагає реалізації багаторівневої автентифікації користувачів, шифрування даних під час їх передавання, а також контролю доступу до інформаційних ресурсів. Використання сучасних методів захисту, таких як хешування паролів, асиметричне шифрування та багатофакторна автентифікація, дозволяє мінімізувати ризики витоку даних.

Важливу роль відіграє також аудит і моніторинг активності користувачів, що дозволяє вчасно виявляти потенційні загрози та реагувати на них. Окрім цього, фізична безпека пристроїв та відповідність законодавчим вимогам є невід'ємними складовими ефективного захисту даних.

Комплексний підхід до розробки програмного забезпечення з урахуванням вимог до захисту персональних даних включає не лише технічні заходи, але й чітко сплановані організаційні заходи. Це дозволяє створити середовище довіри серед користувачів та забезпечити відповідальну обробку даних. Важливим напрямком є також тестування та впровадження програмних рішень, що гарантує стабільність, безпеку та ефективність системи під час віддаленої роботи.

Загалом, забезпечення захисту персональних даних при віддаленій роботі потребує системного підходу, який об'єднує сучасні технічні рішення та ефективні організаційні заходи. Реалізація комплексної стратегії безпеки сприяє мінімізації ризиків витоку інформації та створенню надійного середовища для обробки персональних даних. Це не лише підвищує рівень безпеки, але й зміцнює довіру користувачів до цифрових рішень, що є критично важливим в умовах сучасного інформаційного суспільства.

ВИСНОВКИ

В умовах стрімкого розвитку цифрових технологій та поширення практики віддаленої роботи питання захисту персональних даних набуває особливої актуальності. Зростання обсягів інформації, що передається через мережу, а також збільшення кількості кіберзагроз створюють серйозні виклики для організацій та окремих користувачів. У контексті віддаленої роботи виникає необхідність забезпечення безпеки персональних даних на всіх етапах їх обробки та передачі, що вимагає впровадження ефективних технічних і організаційних заходів.

Дослідження, проведене в межах цього проєкту, охоплює комплексний аналіз особливостей віддаленої роботи та загроз, які виникають у процесі обробки персональних даних. Проаналізовано нормативно-правову базу у сфері захисту інформації, що дозволило визначити ключові вимоги до безпеки даних у контексті віддаленого доступу. Виявлено, що більшість існуючих нормативних актів зосереджені на загальних вимогах до захисту інформації, проте недостатньо враховують специфіку віддаленої роботи, яка передбачає використання хмарних сервісів, публічних мереж та особистих пристроїв працівників. Це вимагає адаптації існуючих стандартів та розробки нових підходів до безпеки даних.

Особливу увагу приділено аналізу загроз, які виникають при віддаленій роботі, зокрема проблемам автентифікації та авторизації, використанню незахищених мереж, збереженню даних на локальних пристроях, а також небезпекам, пов'язаним зі зловмисним програмним забезпеченням. Виявлено, що традиційні методи захисту не завжди є ефективними у контексті віддаленої роботи, оскільки вони не враховують динамічний характер загроз та вразливості, пов'язані з використанням особистих пристроїв працівників. Тому важливим аспектом забезпечення безпеки є використання багатофакторної автентифікації, шифрування даних під час передачі та зберігання, а також

контроль доступу на основі ролей.

На основі аналізу загроз і вразливостей розроблено рекомендації щодо захисту персональних даних при віддаленій роботі. Вони включають комплекс технічних і організаційних заходів, спрямованих на мінімізацію ризиків витоку інформації та несанкціонованого доступу. Серед технічних рішень рекомендовано використання сучасних засобів шифрування, віртуальних приватних мереж (VPN), систем моніторингу та виявлення вторгнень, а також антивірусного програмного забезпечення з можливістю аналізу поведінкових характеристик. Особливу увагу приділено організаційним заходам, серед яких — навчання працівників правилам інформаційної безпеки, розробка політик доступу до корпоративних ресурсів, а також впровадження процедур управління інцидентами.

Розроблені рекомендації мають практичну цінність, оскільки вони адаптовані до умов віддаленої роботи та враховують специфіку використання сучасних цифрових технологій. Вони спрямовані не лише на технічний захист даних, але й на формування культури безпеки серед працівників, що дозволяє мінімізувати ризики, пов'язані з людським фактором. Запропоновані методики можуть бути легко адаптовані до різних типів організацій, включаючи малі та середні підприємства, що робить їх універсальними та гнучкими у застосуванні.

У процесі дослідження було розроблено концепцію програмного проєкту, який реалізує запропоновані рекомендації. Проведено тестування розробленого рішення в реальних умовах, що дозволило підтвердити його ефективність у захисті персональних даних при віддаленій роботі. Впровадження даного проєкту сприяє підвищенню рівня інформаційної безпеки, забезпечуючи цілісність, конфіденційність та доступність даних.

Таким чином, результати дослідження демонструють необхідність комплексного підходу до захисту персональних даних при віддаленій роботі, що включає технічні, організаційні та освітні заходи. Розроблені рекомендації та програмне рішення можуть бути використані як основа для подальших досліджень та розробок у цій сфері, а також для впровадження на практиці в

організаціях, що використовують віддалену роботу. Вони забезпечують ефективний захист інформації в умовах динамічних кіберзагроз і сприяють формуванню культури інформаційної безпеки, що є важливим чинником успішного функціонування сучасних організацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конституція України Із змінами, внесеними Законами України
2. ЗАКОН УКРАЇНИ №2297–VI Про захист персональних даних URL: https://www.president.gov.ua/documents/2297vi-11567 (дата звернення: 17.02.2025)
3. Кодекс законів про працю України. Затверджується Законом № 322-VIII від 10.12.71 ВВР, 1971, додаток до № 50, ст. 375. Із змінами URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 17.02.2025)
4. ЗАКОН УКРАЇНИ Про інформацію (Відомості Верховної Ради України (ВВР), 1992, № 48, ст.650) Із змінами. URL: https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення: 17.02.2025)
5. ЗАКОН УКРАЇНИ №2297–VI Про захист персональних даних URL: https://www.president.gov.ua/documents/2297vi-11567 (дата звернення: 17.02.2025)
6. Регулювання віддаленої роботи: правові аспекти, нормативне регулювання та практичні рекомендації URL: https://consultant.net.ua/consultant-article/10240 (дата звернення: 17.02.2025)
7. Рухаємось до європейських стандартів у питанні захисту персональних даних, - Комітет з питань цифрової трансформації Прес-служба Апарату Верховної Ради України Опубліковано 21 лютого 2024, о 10:02 URL: https://www.rada.gov.ua/news/news_kom/246753.html (дата звернення: 17.02.2025)
8. Рекомендації Уповноваженого Верховної Ради України з прав людини щодо захисту персональних даних під час дистанційного надання освітніх послуг URL: https://mon.gov.ua/osvita-2/pozashkilna-osvita/vikhovna-robota-ta-zakhist-prav-ditini/bezpeka-ditey-v-interneti/rekomendatsii-upovnovazhenogo-verkhovnoi-radi-ukraini-z-prav-lyudini-shchodo-zakhistu-personalnikh-danikh-pid-chas-distantsiynogo-nadannya-osvitnikh-poslug (дата звернення: 17.02.2025)

9. Уповноважений Верховної Ради України з прав людини РЕКОМЕНДАЦІЇ щодо застосування законодавства про захист персональних даних для органів реєстрації (виконавчих органів сільської, селищної або міської ради, центрів надання адміністративних послуг) URL: https://ombudsman.gov.ua/storage/app/media/uploaded-files/Рекомендації%20щодо%20застосування%20законодавства%20про%20захист%20персональних%20даних.pdf (дата звернення: 17.02.2025)

10. Щорічна доповідь про стан додержання та захисту прав і свобод людини і громадянина в Україні у 2023 році URL: https://ombudsman.gov.ua/report-2023/rozdil-9-informatsiini-prava (дата звернення: 17.02.2025)

11. Список нормативних документів щодо інформаційної безпеки в Україні URL: https://uk.wikipedia.org/wiki/Список_нормативних_документів_щодо_інформаційної_безпеки_в_Україні (дата звернення: 17.02.2025)

12. Захист персональних даних працівників та забезпечення права на приватність у контексті дистанційної роботи в Україні URL: https://consultant.net.ua/consultant-article/3827 (дата звернення: 17.02.2025)

13. Захист персональних даних: ВРУ схвалила законопроєкт «Повне або часткове копіювання будь-яких матеріалів сайту, цитування, публікація їх анотованих оглядів допускаються лише за письмового дозволу редакції сайту Служба охорони праці URL: https://pro-op.com.ua/news/86170-zakhist-personalnikh-danikh-vru-skhvalila-zakonoproekt» (дата звернення: 17.02.2025)

14. Сучасні виклики та майбутнє захисту персональних даних – круглий стіл URL: https://www.coe.int/uk/web/kyiv/-/round-table-on-challenges-and-future-of-data-protection (дата звернення: 17.02.2025)

15. Нові правила захисту персональних даних: як відповідати стандартам ЄС? URL: https://yur-gazeta.com/dumka-eksperta/novi-pravila-zahistu-personalnih-danih-yak-vidpovidati-standartam-es-.html (дата звернення:

17.02.2025) Секрети кібербезпеки: протидія цифровим загрозам при віддаленій роботі URL: https://eba.com.ua/sekrety-kiberbezpeky-protydiya-tsyfrovym-zagrozam-pry-viddalenij-roboti/ (дата звернення: 17.02.2025)

16. Кібербезпека та безпека віддаленої роботи. Ризики кібербезпеки під час роботи з будь-якого місця URL: https://www.ranktracker.com/uk/blog/cybersecurity-risks-while-working-from-anywhere/ (дата звернення: 17.02.2025)

17. Дистанційна кібербезпека: як захистити інформацію під час віддаленого режиму роботи URL: https://mind.ua/openmind/20210952-distancijna-kiberbezpeka-yak-zahistiti-informaciyu-pid-chas-viddalenogo-rezhimu-roboti (дата звернення: 17.02.2025)

18. 10 правил кібербезпеки під час дистанційної роботи офісу URL: https://raiffeisen.ua/biznesu/blog/10-pravil-kiberbezpeki-pid-chas-distanciyno-roboti-ofisu-224 (дата звернення: 17.02.2025)

19. Загальні рекомендації щодо підвищення рівня захищеності інформаційних ресурсів при віддаленій роботі співробітників установи URL: https://mihajlivska-gromada.gov.ua/news/1589527120/ (дата звернення: 17.02.2025)

20. Шість найкращих практик забезпечення безпеки даних при віддаленій роботі URL: https://oberig-it.com/statti/shist-najkrashhyh-praktyk-zabezpechennya-bezpeky-danyh-pry-viddalenij-roboti/ (дата звернення: 17.02.2025)

ДОДАТКИ

Додаток А

Код програми

client.py

```
# Функція для входу
def login(username, password): try:
        response = requests.post( f"{API_URL}/login",
            json={"username": username, "password": password}, headers={"Content-Type":
            "application/json"}
        )
        if response.status_code == 200: data = response.json()
            st.session_state.token = data['token'] st.session_state.user_id =
            data['user_id'] st.session_state.username = username
            st.session_state.last_activity = datetime.now() # Встановлюємо час закінчення
            сесії (15 хвилин)
            st.session_state.session_expiry = datetime.now() + timedelta(minutes=15) #
            Виведення деталей токена для відлагодження
            print(f"Отримано токен: {st.session_state.token[:15]}...") print(f"User ID:
            {st.session_state.user_id}")
            # Тест токена - виконуємо запит до API print("Тестуємо токен з заголовками:")
            headers = get_auth_headers() print(f"Headers: {headers}")
            return True else:
            error_msg = response.json().get('error', 'Невідома помилка при вході')
            st.error(f"Помилка: {error_msg}")
            return False except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False# client.py
import streamlit as st import requests import json
import os import re
from datetime import datetime, timedelta import time
import pyotp import qrcode
from io import BytesIO # URL API сервера
API_URL = "http://localhost:5000/api" # Налаштування стану сесії
if 'token' not in st.session_state: st.session_state.token = None
if 'user_id' not in st.session_state: st.session_state.user_id = None
if 'username' not in st.session_state: st.session_state.username = None
if 'last_activity' not in st.session_state: st.session_state.last_activity = datetime.now()
if 'session_expiry' not in st.session_state: st.session_state.session_expiry = None
# Функція для валідації номера телефону def validate_phone_number(phone):# Видалення всіх
нецифрових символів, крім + cleaned_phone = re.sub(r'[^\d+]', '', phone) # Перевірка формату
+38XXXXXXXXXX
    if cleaned_phone.startswith('+38') and len(cleaned_phone) == 13 and
cleaned_phone[1:].isdigit():
        return True, cleaned_phone
    # Якщо номер починається з 0 і містить 10 цифр, додаємо +38
    elif cleaned_phone.startswith('0') and len(cleaned_phone) == 10 and
cleaned_phone.isdigit():
        fixed_phone = '+38' + cleaned_phone return True, fixed_phone
    return False, None
# Функція для валідації імені та прізвища def validate_name(name):
    # Дозволяємо лише літери, пробіли та деякі спеціальні символи (апостроф, дефіс) if
    re.match(r'^[A-Za-zA-Яа-яІЇЇЄєҐґ\s\'\-]+$', name):
        return True return False
# Функція для перевірки надійності пароля def validate_password(password):
    strength = 0 feedback = []
    # Мінімум 8 символів if len(password) >= 8:
```

```
        strength += 1 else:
        feedback.append("Пароль повинен містити не менше 8 символів") # Наявність цифри
    if any(c.isdigit() for c in password): strength += 1
    else:
        feedback.append("Пароль повинен містити хоча б одну цифру") # Наявність великої літери
    if any(c.isupper() for c in password): strength += 1
    else:
        feedback.append("Пароль повинен містити хоча б одну велику літеру") # Наявність
    спеціального символу
    if any(c in "!@#$%^&*()-_=+[]{}|;:'\",.<>/?`~" for c in password): strength += 1
    else:
        feedback.append("Пароль повинен містити хоча б один спеціальний символ") return
    strength, feedback
# Функція для перевірки часу неактивності та автоматичного виходу def check_session():
    if st.session_state.token and st.session_state.session_expiry: current_time =
        datetime.now()
        # Відображення залишку часу сесії
        time_left = st.session_state.session_expiry - current_time minutes_left =
        int(time_left.total_seconds() // 60) seconds_left = int(time_left.total_seconds() %
        60)
        # Якщо час сесії закінчився
        if time_left.total_seconds() <= 0: st.session_state.token = None
            st.session_state.user_id = None st.session_state.username = None
            st.session_state.session_expiry = None
            st.warning("Сесія закінчилася через неактивність. Будь ласка, увійдіть знову.")
            time.sleep(1)
            st.experimental_rerun()
        # Оновлення часу сесії при взаємодії з додатком
        if current_time - st.session_state.last_activity > timedelta(seconds=5): #
перевіряємо кожні 5 секунд
            st.session_state.last_activity = current_time
        return f"Сесія завершиться через: {minutes_left:02d}:{seconds_left:02d}"
        return None
# Функція для створення заголовків з токеном def get_auth_headers():
    if not st.session_state.token:
        return {"Content-Type": "application/json"} # Додаємо логування на стороні клієнта
    token_part = st.session_state.token[:10] + "..." if st.session_state.token else "None"
    print(f"Використовуємо токен: {token_part}")
    # Перевіряємо наявність пробілів у токені if ' ' in st.session_state.token:
        print(f"УВАГА! Токен містить пробіли") # Прибираємо зайві пробіли
        token = st.session_state.token.replace(' ', '') else:
        token = st.session_state.token return {
        "Authorization": f"Bearer {token}", "Content-Type": "application/json"
    }
# Функція для реєстрації
def register(username, password, email, first_name, last_name): try:
        response = requests.post( f"{API_URL}/register", json={
                "username": username, "password": password, "email": email, "first_name":
                first_name, "last_name": last_name
            },
            headers={"Content-Type": "application/json"}
        )
        if response.status_code == 201:
            st.success("Реєстрація успішна! Тепер ви можете увійти.") return True
        else:
            error_msg = response.json().get('error', 'Невідома помилка при реєстрації')
            st.error(f"Помилка: {error_msg}")
            return False except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для виходу def logout():
    if st.session_state.token: try:
            response = requests.post( f"{API_URL}/logout", headers=get_auth_headers()
            )
            # Очищення стану сесії незалежно від відповіді сервера st.session_state.token =
            None
```

```
            st.session_state.user_id = None st.session_state.username = None
            st.session_state.session_expiry = None if response.status_code == 200:
                st.success("Ви успішно вийшли з системи.") else:
                st.warning("Вихід з системи, але виникла помилка на сервері.") except
        Exception as e:
            st.warning(f"Локальний вихід з системи. Помилка з'єднання з сервером: {str(e)}")
            st.session_state.token = None
            st.session_state.user_id = None st.session_state.username = None
            st.session_state.session_expiry = None # Функція для додавання персональних
            даних
def add_personal_data(data_type, data_value): try:
        response = requests.post( f"{API_URL}/personal-data",
            json={"data_type": data_type, "data_value": data_value},
            headers=get_auth_headers()
                                                )
                                                if response.status_code == 201:
                                                    st.success("Персональні дані успішно
                                                    додано.") return True
                                                elif response.status_code == 401:
                                                    st.error("Помилка авторизації. Будь ласка,
                                                    увійдіть знову.") time.sleep(2)
                                                    logout() st.rerun()
                                                else:
                                                    try:
                                                        error_msg =
                                                        response.json().get('error',
даних')                                                     'Невідома помилка при додаванні

                                                    except:
                                                        error_msg = f"Код відповіді:
                                                    {response.status_code}"
                                                    st.error(f"Помилка: {error_msg}")
                                                    return False
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для отримання персональних даних def get_personal_data():
    try:
        # Виведемо заголовки для відлагодження headers = get_auth_headers()
        print(f"Headers in get_personal_data: {headers}") response = requests.get(
            f"{API_URL}/personal-data", headers=headers
        )
        print(f"Response status: {response.status_code}")

                                                if response.status_code == 200: return
                                                    response.json()
                                                elif response.status_code == 401:
                                                    st.error("Помилка авторизації. Будь ласка,
                                                    увійдіть знову.") time.sleep(2)
                                                    logout() st.rerun()
                                                else:
                                                    try:
                                                        error_msg =
                                                        response.json().get('error',
даних')                                                     'Невідома помилка при отриманні

                                                    except:
                                                        error_msg = f"Код відповіді:
                                                    {response.status_code}"
                                                    st.error(f"Помилка: {error_msg}")
                                                    return []
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return []
# Функція для оновлення персональних даних
def update_personal_data(data_id, data_value): try:
        response = requests.put( f"{API_URL}/personal-data/{data_id}", json={"data_value":
            data_value}, headers=get_auth_headers()
```

```
                                                )
                                                if response.status_code == 200:
                                                    st.success("Персональні дані успішно
                                                    оновлено.") return True
                                                elif response.status_code == 401:
                                                    st.error("Помилка авторизації. Будь ласка,
                                                    увійдіть знову.") time.sleep(2)
                                                    logout() st.rerun()
                                                else:
                                                    try:
                                                        error_msg =
                                                        response.json().get('error',
                                                        'Невідома помилка при оновленні
даних')

                                                    except:
                                                        error_msg = f"Код відповіді:
                                                    {response.status_code}"
                                                    st.error(f"Помилка: {error_msg}")
                                                    return False
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для видалення персональних даних def delete_personal_data(data_id):
    try:
        response = requests.delete( f"{API_URL}/personal-data/{data_id}",
            headers=get_auth_headers()
                                                )
                                                if response.status_code == 200:
                                                    st.success("Персональні дані успішно
                                                    видалено.") return True
                                                elif response.status_code == 401:
                                                    st.error("Помилка авторизації. Будь ласка,
                                                    увійдіть знову.") time.sleep(2)
                                                    logout() st.rerun()
                                                else:
                                                    try:
                                                        error_msg =
                                                        response.json().get('error',
                                                        'Невідома помилка при видаленні
даних')

                                                    except:
                                                        error_msg = f"Код відповіді:
                                                    {response.status_code}"
                                                    st.error(f"Помилка: {error_msg}")
                                                    return False
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для зміни пароля
def change_password(current_password, new_password): try:
        response = requests.post( f"{API_URL}/change-password",
            json={"current_password": current_password, "new_password": new_password},
            headers=get_auth_headers()
        )
        if response.status_code == 200: st.success("Пароль успішно змінено.") return True
        else:
            error_msg = response.json().get('error', 'Невідома помилка при зміні пароля')
            st.error(f"Помилка: {error_msg}")
            return False except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для генерації QR-коду для двофакторної автентифікації
```

```
def generate_2fa_qrcode(username): # Генерація секретного ключа secret =
    pyotp.random_base32() # Створення URL для QR-коду totp = pyotp.TOTP(secret)
    provisioning_url = totp.provisioning_uri(name=username,
issuer_name="SecureRemoteWorkApp")
                        # Генерація QR-коду qr = qrcode.QRCode( version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L, box_size=10,
        border=4,
    )
    qr.add_data(provisioning_url) qr.make(fit=True)
    img = qr.make_image(fill_color="black", back_color="white") buffer = BytesIO()
    img.save(buffer) buffer.seek(0) return secret, buffer
# Функція для налаштування двофакторної автентифікації def setup_2fa(secret, code):
    try:
        response = requests.post( f"{API_URL}/setup-2fa", json={"secret": secret, "code":
            code}, headers=get_auth_headers()
                                                )
                                                if response.status_code == 200:
                                                    st.success("Двофакторну автентифікацію
                                                    успішно налаштовано.") return True
                                                else:
                                                    error_msg = response.json().get('error',
2FA')                                               'Невідома помилка при налаштуванні

                                                    st.error(f"Помилка: {error_msg}") return
                                                    False
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return False
# Функція для отримання журналу активності def get_activity_log():
    try:
        response = requests.get( f"{API_URL}/get-activity-log", headers=get_auth_headers()
        )
        if response.status_code == 200: return response.json()
        else:
            error_msg = response.json().get('error', 'Невідома помилка при отриманні журналу
активності')
            st.error(f"Помилка: {error_msg}") return []
    except Exception as e:
        st.error(f"Помилка підключення до сервера: {str(e)}") return []
# Виведення токена для debug def debug_token():
    if st.session_state.token:
        st.sidebar.text_area("Поточний токен", st.session_state.token, height=100) headers =
        get_auth_headers()
        st.sidebar.text_area("Заголовки", str(headers), height=100) else:
        st.sidebar.error("Токен відсутній")
```

```
# Головний заголовок програми
st.title("Захищена система для віддаленої роботи") # Бічна панель для навігації
menu = st.sidebar.selectbox( "Меню",
    ["Вхід", "Реєстрація", "Особистий кабінет", "Персональні дані", "Безпека"] if not
    st.session_state.token else
    ["Особистий кабінет", "Персональні дані", "Безпека", "Вихід"]
)
# Відображення таймера сесії if st.session_state.token:
    session_info = check_session() if session_info:
        st.sidebar.info(session_info) # Відображення інформації про токен
    with st.sidebar.expander("Інформація про авторизацію"): st.text(f"Токен:
        {st.session_state.token[:10]}...") st.text(f"ID користувача:
        {st.session_state.user_id}") st.text(f"Заголовки: {get_auth_headers()}")
        if st.button("Тест токена"): try:
                response = requests.get(f"{API_URL}/personal-data",
headers=get_auth_headers())
                st.text(f"Код відповіді: {response.status_code}") st.text(f"Заголовки запиту:
                {get_auth_headers()}") if response.status_code == 200:
                    st.success("Токен працює!") else:
                    st.error(f"Помилка авторизації: {response.status_code}") except Exception
            as e:
                st.error(f"Помилка запиту: {str(e)}")
# Відображення різних сторінок в залежності від вибору меню if menu == "Вхід" and not
st.session_state.token:
    st.header("Вхід до системи") with st.form("login_form"):
        username = st.text_input("Ім'я користувача") password = st.text_input("Пароль",
        type="password") submit_button = st.form_submit_button("Увійти")
        if submit_button:
            if username and password:
                if login(username, password): st.success("Вхід успішний!") time.sleep(1)
                    st.rerun()
            else:
                st.error("Будь ласка, заповніть всі поля.") elif menu == "Реєстрація" and not
st.session_state.token:
    st.header("Реєстрація в системі") with st.form("register_form"):
        first_name = st.text_input("Ім'я") last_name = st.text_input("Прізвище") username =
        st.text_input("Логін") email = st.text_input("Email")
        password = st.text_input("Пароль", type="password")
        password_confirm = st.text_input("Підтвердження пароля", type="password") # Валідація
        даних
        validation_passed = True
        # Перевірка імені та прізвища
        if first_name and not validate_name(first_name):
            st.error("Ім'я може містити лише літери, пробіли, апостроф та дефіс.")
            validation_passed = False
        if last_name and not validate_name(last_name):
            st.error("Прізвище може містити лише літери, пробіли, апостроф та дефіс.")
            validation_passed = False
        # Перевірка надійності пароля
```

```
        if password:
            strength, feedback = validate_password(password) if strength == 0:
                st.error("Дуже слабкий пароль: " + ", ".join(feedback)) validation_passed =
                False
            elif strength == 1:
                st.error("Слабкий пароль: " + ", ".join(feedback)) validation_passed = False
            elif strength == 2:
                st.warning("Середній пароль: " + ", ".join(feedback)) validation_passed =
                False
            elif strength == 3:
                st.info("Хороший пароль: " + ", ".join(feedback)) elif strength == 4:
                st.success("Сильний пароль")
        submit_button = st.form_submit_button("Зареєструватися") if submit_button:
            if first_name and last_name and username and email and password and
password_confirm:
                if password != password_confirm: st.error("Паролі не співпадають.")
                elif "@" not in email or "." not in email: st.error("Введіть дійсний email.")
                elif validation_passed:
                    if register(username, password, email, first_name, last_name):
                        st.info("Тепер ви можете увійти до системи.")
            else:
                st.error("Будь ласка, заповніть всі поля.") elif menu == "Особистий кабінет"
and st.session_state.token:
    st.header(f"Вітаємо, {st.session_state.username}!") st.info("""
    Це захищена система для віддаленої роботи.
    Ваші дані захищені сучасними методами шифрування та контролю доступу. """)
    st.subheader("Рекомендації з безпеки:") st.markdown("""
    * Регулярно змінюйте пароль
    * Не використовуйте один і той самий пароль на різних сервісах
    * Не передавайте свої облікові дані третім особам
    * Налаштуйте двофакторну автентифікацію у розділі "Безпека"
    *      Завжди виходьте з системи після завершення роботи """)
    # Відображення часу останньої активності
    st.text(f"Остання активність: {st.session_state.last_activity.strftime('%Y-%m-%d %H:%M:
%S')}")
    # Журнал активності st.subheader("Останні дії:") activity_log = get_activity_log() if
    activity_log:
        for i, activity in enumerate(activity_log[:5]): # Показуємо лише 5 останніх дій
            st.text(f"{activity['timestamp']} - {activity['action']} - IP:
{activity['ip_address']}") else:
        st.info("Журнал активності порожній.")
elif menu == "Персональні дані" and st.session_state.token: st.header("Мої персональні дані")
    # Тест підключення
    test_connection = st.checkbox("Перевірити підключення") if test_connection:
        try:
            test_response = requests.get(f"{API_URL}/personal-data",
headers=get_auth_headers())
            st.write(f"Код відповіді: {test_response.status_code}") st.write(f"Токен:
            {st.session_state.token[:10]}...") st.write(f"Заголовки запиту:
            {get_auth_headers()}")
```

```
            try:
                st.write(f"Відповідь: {test_response.json()}")
            except:
                st.write(f"Відповідь не в форматі JSON: {test_response.text}") except
        Exception as e:
            st.error(f"Помилка підключення: {str(e)}") # Форма для додавання персональних
    даних
    with st.expander("Додати нові дані", expanded=True): with st.form("personal_data_form"):
            data_type = st.selectbox( "Тип даних",
                ["Номер телефону", "Ім'я", "Прізвище", "Адреса", "Паспортні дані", "ІПН",
"Банківські реквізити", "Інше"]
            )
            data_value = st.text_input("Значення") submit_button =
            st.form_submit_button("Зберегти") if submit_button:
                if data_type and data_value:
                    # Валідація даних в залежності від типу if data_type == "Номер телефону":
                        is_valid, validated_phone = validate_phone_number(data_value) if not
                        is_valid:
                            st.error("Невірний формат номера телефону. Використовуйте
                            формат
+38XXXXXXXXXX або 0XXXXXXXXX.")
                        else:
                            add_personal_data(data_type, validated_phone) st.rerun()
                    elif data_type in ["Ім'я", "Прізвище"]: if not
                        validate_name(data_value):
                            st.error(f"Невірний формат {data_type.lower()}. Використовуйте
лише літери.")                                          else:
                                add_personal_data(data_type,
                                data_value) st.rerun()
                    else:
                        add_personal_data(data_type, data_value) st.rerun()
                else:
                    st.error("Будь ласка, заповніть всі поля.") # Відображення існуючих
    персональних даних st.subheader("Мої збережені дані")
    personal_data = get_personal_data() if personal_data:
        for item in personal_data:
            with st.expander(f"{item['data_type']} (додано: {item['created_at']})"):
                current_value = item['data_value']
                st.text(f"Поточне значення: {current_value}") st.text(f"Оновлено:
                {item['updated_at']}")
                # Редагування даних
                new_value = st.text_input(f"Нове значення для {item['data_type']}",
key=f"edit_{item['id']}")
                col1, col2 = st.columns(2) with col1:
                    if st.button("Оновити", key=f"update_{item['id']}"): if new_value:
                            # Валідація даних в залежності від типу if item['data_type'] ==
                            "Номер телефону":
                                is_valid, validated_phone = validate_phone_number(new_value)
                                if not is_valid:
                                    st.error("Невірний формат номера телефону.
Використовуйте формат +38XXXXXXXXXX або 0XXXXXXXXX.")
                                else:
                                    update_personal_data(item['id'], validated_phone)
                                    st.rerun()
                            elif item['data_type'] in ["Ім'я", "Прізвище"]:
```

```
                                                    if not validate_name(new_value):
Використовуйте лише літери.")                           st.error(f"Невірний формат
                                                        {item['data_type'].lower()}.

                                                    else:
                                                        update_personal_data(item['id'],
                                                        new_value) st.rerun()
                                                    else:
                                                        update_personal_data(item['id'],
                                                        new_value) st.rerun()
                    with col2:
```

```
                    if st.button("Видалити", key=f"delete_{item['id']}"):
                        delete_personal_data(item['id'])
                        st.rerun()
    else:
        st.info("У вас поки немає збережених персональних даних.") elif menu == "Безпека"
and st.session_state.token:
    st.header("Налаштування безпеки") # Зміна пароля st.subheader("Зміна пароля")
    with st.form("change_password_form"):
        current_password = st.text_input("Поточний пароль", type="password") new_password =
        st.text_input("Новий пароль", type="password")
        confirm_password = st.text_input("Підтвердження нового пароля", type="password") #
        Перевірка надійності пароля
        if new_password:
            strength, feedback = validate_password(new_password) if strength < 3:
                st.error("Пароль недостатньо надійний: " + ", ".join(feedback))
                is_strong_password = False
            else:
                is_strong_password = True if strength == 3:
                    st.info("Хороший пароль") elif strength == 4:
                    st.success("Сильний пароль")
        submit_button = st.form_submit_button("Змінити пароль") if submit_button:
            if not current_password or not new_password or not confirm_password:
                st.error("Будь ласка, заповніть всі поля.")
            elif new_password != confirm_password:
                st.error("Новий пароль та підтвердження не співпадають.") elif not
            is_strong_password:
                st.error("Новий пароль недостатньо надійний. Підвищіть його надійність.") else:
                if change_password(current_password, new_password): st.success("Пароль
                    успішно змінено!")
    # Налаштування двофакторної автентифікації st.subheader("Двофакторна автентифікація")
    if st.button("Налаштувати двофакторну автентифікацію"):
        secret, qr_code_img = generate_2fa_qrcode(st.session_state.username)
        st.image(qr_code_img, caption="QR-код для Google Authenticator або іншого додатку
для 2FA")
        st.info(f"Секретний ключ: {secret}") st.markdown("""
        1.    Відскануйте QR-код за допомогою Google Authenticator або іншого додатку для
двофакторної автентифікації
        2. Збережіть секретний ключ у надійному місці на випадок втрати доступу до додатку
        3.     Введіть код із додатку для підтвердження налаштування """)
        verification_code = st.text_input("Введіть код із додатку для 2FA",
key="2fa_setup_code")
        if st.button("Підтвердити"): if verification_code:
                setup_2fa(secret, verification_code) else:


                st.error("Будь ласка, введіть код із додатку.") # Журнал активності
    st.subheader("Журнал активності")
    st.info("Тут відображаються останні дії з вашим обліковим записом для виявлення підозрілої
активності.")
    activity_log = get_activity_log() if activity_log:
        for activity in activity_log:
            st.text(f"{activity['timestamp']} - {activity['action']} - IP:
{activity['ip_address']} - {activity['user_agent']}") else:
        st.info("Журнал активності порожній.") elif menu == "Вихід" and
st.session_state.token:
    st.header("Вихід із системи")
    if st.button("Підтвердити вихід"): logout()
        st.rerun()
    st.warning("Будь ласка, завжди виходьте з системи після завершення роботи для захисту
своїх даних.")
else:
    # Якщо користувач не авторизований і намагається отримати доступ до захищених сторінок if
    not st.session_state.token and menu in ["Особистий кабінет", "Персональні дані",
"Безпека", "Вихід"]:
        st.warning("Для доступу до цієї сторінки необхідно увійти до системи.") st.rerun()
# Інформація про захист даних у нижньому колонтитулі st.markdown("---")
```

```
st.markdown("""
<small>© 2025 Захищена система для віддаленої роботи. Всі дані шифруються та зберігаються відповідно до вимог законодавства про захист персональних даних.</small>
       """, unsafe_allow_html=True)
```

server.py

```
# server.py import os import sqlite3 import hashlib import secrets import jwt
from datetime import datetime, timedelta from flask import Flask, request, jsonify from flask_cors import CORS
from dotenv import load_dotenv import logging
from logging.handlers import RotatingFileHandler from functools import wraps
# Завантаження змінних середовища load_dotenv()
app = Flask(_name_)
CORS(app) # Включення CORS для взаємодії з клієнтом # Налаштування логгера для моніторингу подій безпеки if not os.path.exists('logs'):
    os.makedirs('logs')
file_handler = RotatingFileHandler('logs/security.log', maxBytes=10240, backupCount=10)
file_handler.setFormatter(logging.Formatter(
    '%(asctime)s %(levelname)s: %(message)s [in %(pathname)s:%(lineno)d]'
))
file_handler.setLevel(logging.INFO)                    app.logger.addHandler(file_handler)
app.logger.setLevel(logging.INFO)
# Секретний ключ для JWT токенів (в реальному додатку використовувати змінні середовища)
SECRET_KEY = os.getenv('SECRET_KEY', secrets.token_hex(32)) # Ініціалізація бази даних
def init_db():
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE NOT NULL, password_hash TEXT NOT NULL, salt TEXT NOT NULL,
        email TEXT UNIQUE NOT NULL,
        first_name TEXT NOT NULL, last_name TEXT NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        last_login TIMESTAMP, two_factor_enabled BOOLEAN DEFAULT 0, two_factor_secret TEXT,
        is_active BOOLEAN DEFAULT 1
    ) ''')
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS personal_data ( id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER, data_type TEXT NOT NULL, data_value TEXT NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP,
        FOREIGN KEY (user_id) REFERENCES users (id)
    ) ''')
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS audit_log ( id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER, action TEXT NOT NULL,
        timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        ip_address TEXT, user_agent TEXT,
        FOREIGN KEY (user_id) REFERENCES users (id)
    ) ''')
    conn.commit() conn.close()
# Виклик ініціалізації бази даних init_db()
# Функція для валідації номера телефону def validate_phone_number(phone):
    import re
    # Видалення всіх нецифрових символів, крім + cleaned_phone = re.sub(r'[^\d+]', '', phone)
    # Перевірка формату +38XXXXXXXXXX
    if cleaned_phone.startswith('+38') and len(cleaned_phone) == 13 and
cleaned_phone[1:].isdigit():
        return True, cleaned_phone
    # Якщо номер починається з 0 і містить 10 цифр, додаємо +38
    elif cleaned_phone.startswith('0') and len(cleaned_phone) == 10 and
cleaned_phone.isdigit():
```

```
        fixed_phone = '+38' + cleaned_phone return True, fixed_phone
    return False, None
# Функція для валідації імені та прізвища def validate_name(name):
    # Перевіряємо кожен символ у імені/прізвищі
    allowed_chars =
    set("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzАБВГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦ
    ЧШЩЬЮЯабв гдеєжзиіїйклмнопрстуфхцчшщьюяґ '-")
    for char in name:
        if char not in allowed_chars: return False
    return True
# Функція для валідації пароля def validate_password(password):
    # Мінімум 8 символів if len(password) < 8:
        return False, "Пароль повинен містити не менше 8 символів" # Наявність цифри
    if not any(c.isdigit() for c in password):
        return False, "Пароль повинен містити хоча б одну цифру" # Наявність великої літери
    if not any(c.isupper() for c in password):
        return False, "Пароль повинен містити хоча б одну велику літеру" # Наявність
    спеціального символу
    if not any(c in "!@#$%^&*()-_=+[]{}|;:'\",.<>/?`~" for c in password): return False,
        "Пароль повинен містити хоча б один спеціальний символ"
    return True, ""
# Функція для хешування паролів
def hash_password(password, salt=None): if salt is None:
        salt = secrets.token_hex(16)
    # Використання PBKDF2 через hashlib для безпечного хешування паролів hash_obj =
    hashlib.pbkdf2_hmac('sha256', password.encode('utf-8'),
                                salt.encode('utf-8'), 100000)
    password_hash = hash_obj.hex() return password_hash, salt
# Створення JWT токенів def create_token(user_id):
    # Використовуємо datetime.now() з UTC try:
        # Для новіших версій Python (3.11+) from datetime import UTC current_time =
        datetime.now(UTC)
        expiration = current_time + timedelta(hours=1) # Токен дійсний 1 годину except
    ImportError:
        # Для старіших версій Python import datetime as dt
        current_time = datetime.now(dt.timezone.utc)
        expiration = current_time + timedelta(hours=1) # Токен дійсний 1 годину # Важливо!
    Переконуємося що user_id це рядок
    user_id_str = str(user_id) payload = {
        'exp': expiration, 'iat': current_time,
        'sub': user_id_str # Використовуємо рядкове представлення
    }
    app.logger.info(f"Створюємо токен для user_id: {user_id_str} (тип:
{type(user_id_str)._name_})")
    token = jwt.encode(payload, SECRET_KEY, algorithm='HS256') app.logger.info(f"Токен
    створено: {token[:15]}...")
    return token
# Перевірка токенів
def verify_token(token): try:
        # Для відлагодження - виведемо токен app.logger.info(f"Верифікація токена (початок):
        {token[:10]}...") # Декодування токена
        payload = jwt.decode(token, SECRET_KEY, algorithms=['HS256']) # Перевіримо тип поля
        sub
        sub_value = payload.get('sub')
        sub_type = type(sub_value)._name_
        app.logger.info(f"Поле 'sub' в токені: {sub_value} (тип: {sub_type})")
        app.logger.info(f"Токен успішно декодовано. User ID: {payload['sub']}") return
        payload['sub']
    except jwt.ExpiredSignatureError: app.logger.warning("Токен прострочений") return None #
        Токен прострочений
    except jwt.InvalidTokenError as e: app.logger.warning(f"Недійсний токен: {str(e)}")
        return None # Недійсний токен
    except Exception as e:
        app.logger.error(f"Помилка при перевірці токена: {str(e)}") return None
# Декоратор для захисту маршрутів def token_required(f):
```

```
    @wraps(f)
    def decorated(*args, **kwargs): token = None
        auth_header = request.headers.get('Authorization') # Для відлагодження
        app.logger.info(f"Заголовок Authorization: {auth_header}") if auth_header:
            try:
                # Перевіряємо чи є префікс "Bearer " if "Bearer " in auth_header:
                    # Вилучаємо токен без пробілів
                    token = auth_header.split("Bearer ")[1].strip() else:
                    token = auth_header.strip() # Виявлення пробілів у токені if ' ' in token:
                    app.logger.warning(f"Токен містить пробіли, очищаємо: '{token}'") token =
                    token.replace(' ', '')
                app.logger.info(f"Отриманий токен (після обробки): {token[:10]}...") except
            IndexError:
                app.logger.warning("Невірний формат токена")
                return jsonify({'message': 'Невірний формат токена'}), 401 except Exception
            as e:
                app.logger.error(f"Помилка при обробці заголовка: {str(e)}") return
                jsonify({'message': 'Помилка при обробці заголовка'}), 401
        if not token:
            app.logger.warning("Токен автентифікації відсутній")
            return jsonify({'message': 'Токен автентифікації відсутній'}), 401 user_id =
        verify_token(token)
        if not user_id:
            app.logger.warning("Токен недійсний або прострочений")
            return jsonify({'message': 'Токен недійсний або прострочений'}), 401 #
        Конвертуємо user_id назад у число, оскільки в БД це INT
        try:
            user_id = int(user_id) except ValueError:
            app.logger.error(f"Неможливо конвертувати user_id '{user_id}' в число") return
            jsonify({'message': 'Помилка автентифікації'}), 401
        # Запис у журнал аудиту
        conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
        cursor.execute('''
        INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
        ''', (user_id, f._name_, request.remote_addr, request.user_agent.string))
        conn.commit()
        conn.close()
        app.logger.info(f"Успішна авторизація для user_id: {user_id}") return f(user_id,
        *args, **kwargs)
    return decorated
    # Маршрути API
@app.route('/api/register', methods=['POST']) def register():
    data = request.get_json()
    # Перевірка наявності всіх необхідних полів
    required_fields = ['username', 'password', 'email', 'first_name', 'last_name'] if not
    all(field in data for field in required_fields):
        return jsonify({'error': 'Відсутні обов\'язкові поля'}), 400 # Валідація імені та
    прізвища
    if not validate_name(data['first_name']):
        return jsonify({'error': 'Ім\'я містить неприпустимі символи. Використовуйте лише
літери.'}), 400
    if not validate_name(data['last_name']):
        return jsonify({'error': 'Прізвище містить неприпустимі символи. Використовуйте лише
літери.'}), 400
    # Валідація пароля
    is_valid_password, password_error = validate_password(data['password']) if not
    is_valid_password:
        return jsonify({'error': f'Слабкий пароль: {password_error}'}), 400 # Підключення до
    БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    # Перевірка, чи існує вже користувач
    cursor.execute('SELECT id FROM users WHERE username = ?', (data['username'],)) if
    cursor.fetchone():
        conn.close()
        return jsonify({'error': 'Користувач з таким ім\'ям вже існує'}), 409 # Перевірка, чи
```

```
    існує вже email
    cursor.execute('SELECT id FROM users WHERE email = ?', (data['email'],)) if
    cursor.fetchone():
        conn.close()
        return jsonify({'error': 'Цей email вже використовується'}), 409 # Хешування пароля
    password_hash, salt = hash_password(data['password']) # Збереження користувача в БД
    try:
        cursor.execute('''
        INSERT INTO users (username, password_hash, salt, email, first_name, last_name,
created_at)
        VALUES (?, ?, ?, ?, ?, ?, datetime('now'))
        ''', (data['username'], password_hash, salt, data['email'], data['first_name'],
data['last_name']))
        user_id = cursor.lastrowid # Запис у журнал аудиту cursor.execute('''
        INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
        ''', (user_id, 'register', request.remote_addr, request.user_agent.string))
        conn.commit()
        app.logger.info(f'Новий користувач зареєстрований: {data["username"]}') return
        jsonify({
            'message': 'Користувач успішно зареєстрований', 'user_id': user_id
        }), 201
    except Exception as e: conn.rollback()
        app.logger.error(f'Помилка реєстрації: {str(e)}') return jsonify({'error': str(e)}),
        500
    finally:
        conn.close() @app.route('/api/login', methods=['POST']) def login():
    data = request.get_json()
    # Перевірка наявності всіх необхідних полів
    if not data or 'username' not in data or 'password' not in data:
    return jsonify({'error': 'Відсутні обов\'язкові поля'}), 400 # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    # Пошук користувача
    cursor.execute('SELECT id, password_hash, salt, is_active FROM users WHERE username
= ?',
                   (data['username'],)) user = cursor.fetchone()
    if not user:
        # Запис у журнал про невдалу спробу входу
        app.logger.warning(f'Невдала спроба входу для неіснуючого користувача:
                                                                {data["username"]}')
                                                                        conn.close()
        return jsonify({'error': 'Невірне ім\'я користувача або пароль'}), 401 user_id,
    stored_hash, salt, is_active = user
    if not is_active:
        app.logger.warning(f'Спроба входу для деактивованого користувача:
                                                                {data["username"]}')
                                                                        conn.close()
        return jsonify({'error': 'Обліковий запис деактивовано'}), 403 # Перевірка пароля
    password_hash, _ = hash_password(data['password'], salt) if password_hash !=
    stored_hash:
        # Запис у журнал про невдалу спробу входу
        app.logger.warning(f'Невдала спроба входу для користувача: {data["username"]}')
        cursor.execute('''
        INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
        ''', (user_id, 'failed_login', request.remote_addr, request.user_agent.string))
        conn.commit()
        conn.close()
        return jsonify({'error': 'Невірне ім\'я користувача або пароль'}), 401 # Оновлення
    часу останнього входу
    cursor.execute('UPDATE users SET last_login = datetime("now") WHERE id = ?', (user_id,))
    # Запис у журнал про успішний вхід
    cursor.execute('''
    INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
    ''', (user_id, 'login', request.remote_addr, request.user_agent.string)) conn.commit()
    conn.close()
    # Створення токена
```

```
    token = create_token(user_id)
    app.logger.info(f'Користувач увійшов: {data["username"]}') return jsonify({
        'message': 'Вхід успішний', 'token': token,
        'user_id': user_id
    }), 200
@app.route('/api/personal-data', methods=['POST']) @token_required
def add_personal_data(user_id): data = request.get_json()
    if not data or 'data_type' not in data or 'data_value' not in data: return
        jsonify({'error': 'Відсутні обов\'язкові поля'}), 400
    # Валідація даних в залежності від типу if data['data_type'] == 'Номер телефону':
        is_valid, validated_phone = validate_phone_number(data['data_value']) if not
        is_valid:
            return jsonify({'error': 'Невірний формат номера телефону. Використовуйте
            формат
+38XXXXXXXXXX або 0XXXXXXXXX.'}), 400
        data['data_value'] = validated_phone
    elif data['data_type'] in ['Ім\'я', 'Прізвище']: if not
        validate_name(data['data_value']):
    return jsonify({'error': f'Невірний формат {data["data_type"].lower()}.
Використовуйте лише літери.'}), 400 # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        cursor.execute('''
        INSERT INTO personal_data (user_id, data_type, data_value, created_at, updated_at)
        VALUES (?, ?, ?, datetime('now'), datetime('now'))
        ''', (user_id, data['data_type'], data['data_value'])) data_id = cursor.lastrowid
        conn.commit()
        app.logger.info(f'Додано персональні дані для користувача ID {user_id}, тип:
{data["data_type"]}')
        return jsonify({
            'message': 'Персональні дані успішно додано', 'data_id': data_id
        }), 201
    except Exception as e: conn.rollback()
        app.logger.error(f'Помилка при додаванні персональних даних: {str(e)}') return
        jsonify({'error': str(e)}), 500
    finally:
        conn.close()
@app.route('/api/personal-data', methods=['GET']) @token_required
def get_personal_data(user_id): # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        cursor.execute('''
        SELECT id, data_type, data_value, created_at, updated_at FROM personal_data
        WHERE user_id = ? ''', (user_id,))
        rows = cursor.fetchall() personal_data = []
        for row in rows: personal_data.append({
                'id': row[0],
                'data_type': row[1], 'data_value': row[2], 'created_at': row[3],
                'updated_at': row[4]
            })
        app.logger.info(f'Користувач ID {user_id} отримав персональні дані') return
        jsonify(personal_data), 200
    except Exception as e:
        app.logger.error(f'Помилка при отриманні персональних даних: {str(e)}') return
        jsonify({'error': str(e)}), 500
    finally:
        conn.close()
@app.route('/api/personal-data/<int:data_id>', methods=['PUT']) @token_required
def update_personal_data(user_id, data_id): data = request.get_json()
    if not data or 'data_value' not in data:
        return jsonify({'error': 'Відсутні обов\'язкові поля'}), 400 # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        # Перевірка, чи є дані і чи належать вони даному користувачу cursor.execute('''SELECT
        data_type FROM personal_data WHERE id = ? AND user_id = ?
```

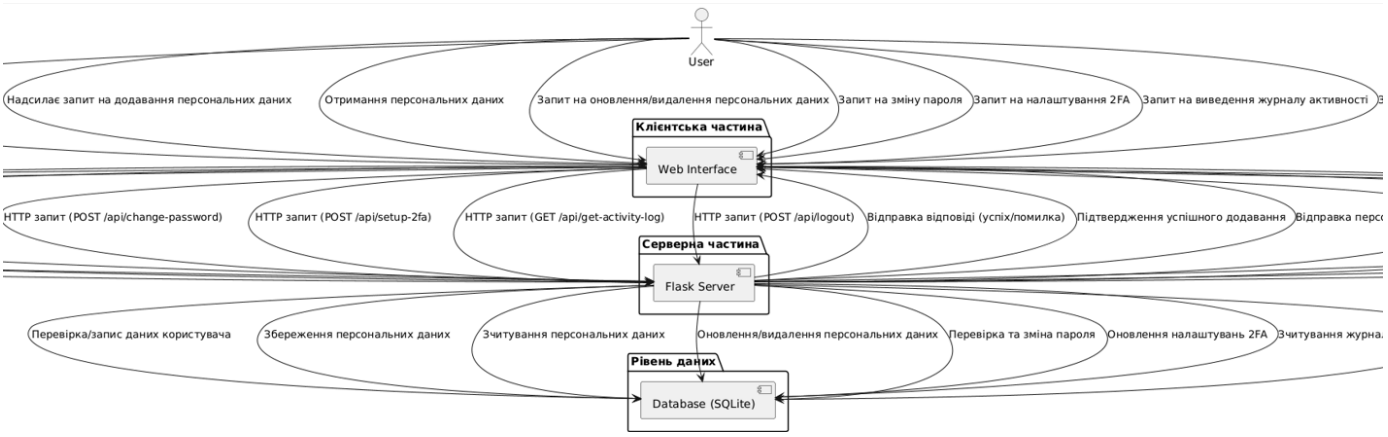
```
        ''', (data_id, user_id)) result = cursor.fetchone() if not result:
            conn.close()
            return jsonify({'error': 'Персональні дані не знайдено або доступ заборонено'}),
404
        data_type = result[0]
        # Валідація даних в залежності від типу if data_type == 'Номер телефону':
            is_valid, validated_phone = validate_phone_number(data['data_value']) if not
            is_valid:
                return jsonify({'error': 'Невірний формат номера телефону. Використовуйте
формат +38XXXXXXXXXX або 0XXXXXXXXX.'}), 400
            data['data_value'] = validated_phone elif data_type in ['Ім\'я', 'Прізвище']:
            if not validate_name(data['data_value']):
                return jsonify({'error': f'Невірний формат {data_type.lower()}.
Використовуйте лише літери.'}), 400 # Оновлення даних cursor.execute('''
        UPDATE personal_data
        SET data_value = ?, updated_at = datetime('now') WHERE id = ? AND user_id = ?
        ''', (data['data_value'], data_id, user_id)) conn.commit()
        app.logger.info(f'Оновлено персональні дані ID {data_id} для користувача ID
{user_id}, тип: {data_type}') return jsonify({
            'message': 'Персональні дані успішно оновлено', 'data_id': data_id
        }), 200
    except Exception as e: conn.rollback()
        app.logger.error(f'Помилка при оновленні персональних даних: {str(e)}') return
        jsonify({'error': str(e)}), 500
    finally:
        conn.close()
@app.route('/api/personal-data/<int:data_id>', methods=['DELETE']) @token_required
def delete_personal_data(user_id, data_id): # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        # Перевірка, чи є дані і чи належать вони даному користувачу cursor.execute('''
        SELECT id FROM personal_data WHERE id = ? AND user_id = ? ''', (data_id, user_id))
        if not cursor.fetchone(): conn.close()
            return jsonify({'error': 'Персональні дані не знайдено або доступ заборонено'}),
404
        # Видалення даних cursor.execute(''' DELETE FROM personal_data
        WHERE id = ? AND user_id = ? ''', (data_id, user_id)) conn.commit()
        app.logger.info(f'Видалено персональні дані ID {data_id} для користувача ID
{user_id}')
        return jsonify({
            'message': 'Персональні дані успішно видалено'}), 200
    except Exception as e: conn.rollback()
        app.logger.error(f'Помилка при видаленні персональних даних: {str(e)}') return
        jsonify({'error': str(e)}), 500
    finally:
        conn.close()
@app.route('/api/change-password', methods=['POST']) @token_required
def change_password(user_id): data = request.get_json()
    if not data or 'current_password' not in data or 'new_password' not in data: return
        jsonify({'error': 'Відсутні обов\'язкові поля'}), 400
    # Валідація нового пароля
    is_valid_password, password_error = validate_password(data['new_password']) if not
    is_valid_password:
        return jsonify({'error': f'Слабкий пароль: {password_error}'}), 400 # Підключення до
    БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        # Отримання поточного хешу та солі
        cursor.execute('SELECT password_hash, salt FROM users WHERE id = ?', (user_id,))
        user = cursor.fetchone()
        if not user:
            conn.close()
            return jsonify({'error': 'Користувача не знайдено'}), 404 stored_hash, salt =
        user
        # Перевірка поточного пароля
```

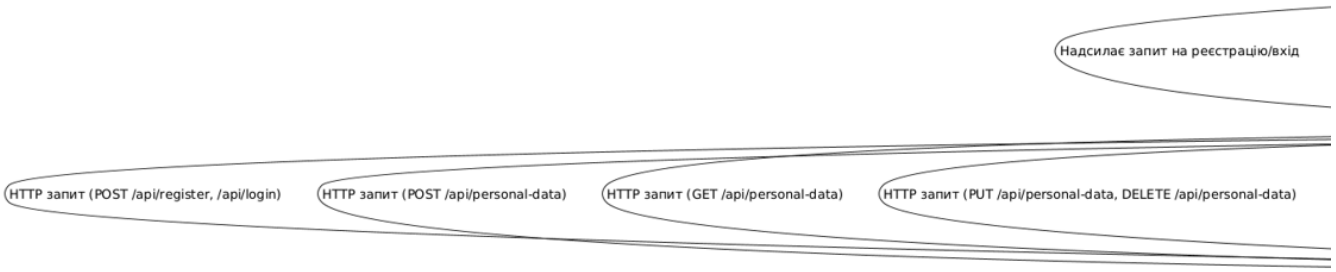
```
        current_password_hash, _ = hash_password(data['current_password'], salt) if
        current_password_hash != stored_hash:
            conn.close()
            return jsonify({'error': 'Невірний поточний пароль'}), 401 # Хешування нового
        пароля
        new_password_hash, new_salt = hash_password(data['new_password']) # Оновлення
        пароля в БД
        cursor.execute(''' UPDATE users
        SET password_hash = ?, salt = ? WHERE id = ?
        ''', (new_password_hash, new_salt, user_id)) # Запис у журнал аудиту
        cursor.execute('''
        INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
        ''', (user_id, 'change_password', request.remote_addr, request.user_agent.string))
        conn.commit()
        app.logger.info(f'Користувач ID {user_id} змінив пароль') return jsonify({'message':
        'Пароль успішно змінено'}), 200
    except Exception as e: conn.rollback()
        app.logger.error(f'Помилка при зміні пароля: {str(e)}') return jsonify({'error':
        str(e)}), 500
    finally:
        conn.close()
@app.route('/api/setup-2fa', methods=['POST']) @token_required
def setup_2fa(user_id):
    data = request.get_json()
    if not data or 'secret' not in data or 'code' not in data: return jsonify({'error':
        'Відсутні обов\'язкові поля'}), 400
    # Перевірка коду import pyotp
    totp = pyotp.TOTP(data['secret']) if not totp.verify(data['code']):
    return jsonify({'error': 'Невірний код 2FA'}), 400 # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        # Оновлення налаштувань 2FA cursor.execute('''
        UPDATE users
        SET two_factor_enabled = 1, two_factor_secret = ? WHERE id = ?
        ''', (data['secret'], user_id)) # Запис у журнал аудиту cursor.execute('''
        INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
        ''', (user_id, 'setup_2fa', request.remote_addr, request.user_agent.string))
        conn.commit()
        app.logger.info(f'Користувач ID {user_id} налаштував 2FA')
        return jsonify({'message': 'Двофакторну автентифікацію успішно налаштовано'}), 200
    except Exception as e:
        conn.rollback()
        app.logger.error(f'Помилка при налаштуванні 2FA: {str(e)}') return jsonify({'error':
        str(e)}), 500
    finally:
        conn.close() @app.route('/api/logout', methods=['POST']) @token_required
def logout(user_id):
    # Записуємо у журнал вихід користувача
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    cursor.execute('''
    INSERT INTO audit_log (user_id, action, ip_address, user_agent) VALUES (?, ?, ?, ?)
    ''', (user_id, 'logout', request.remote_addr, request.user_agent.string)) conn.commit()
    conn.close()
    app.logger.info(f'Користувач ID {user_id} вийшов з системи') # У реальному додатку можна
    додати токен у чорний список return jsonify({'message': 'Вихід успішний'}), 200
@app.route('/api/get-activity-log', methods=['GET']) @token_required
def get_activity_log(user_id): # Підключення до БД
    conn = sqlite3.connect('secure_remote_work.db') cursor = conn.cursor()
    try:
        cursor.execute('''
        SELECT action, timestamp, ip_address, user_agent FROM audit_log
        WHERE user_id = ? ORDER BY timestamp DESC LIMIT 50
        ''', (user_id,))
        rows = cursor.fetchall() activity_log = []
        for row in rows: activity_log.append({
```

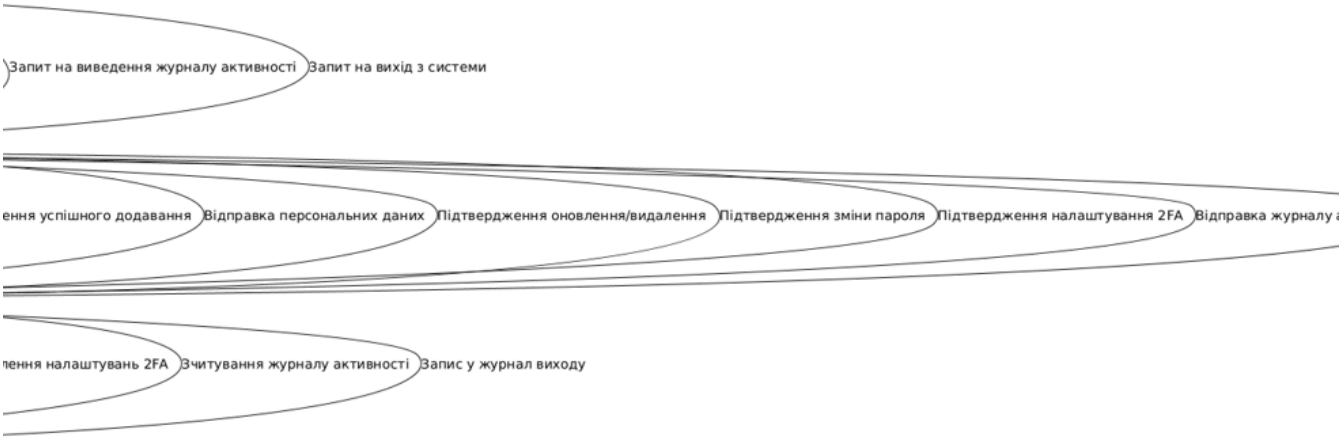
```
                'action': row[0],
                'timestamp': row[1], 'ip_address': row[2], 'user_agent': row[3]
            })
        app.logger.info(f'Користувач ID {user_id} отримав журнал активності') return
        jsonify(activity_log), 200
        except Exception as e:
        app.logger.error(f'Помилка при отриманні журналу активності: {str(e)}') return
        jsonify({'error': str(e)}), 500
    finally:
        conn.close() # Запуск веб-сервера
if _name___ == '_main_':
    # В продакшн використовувати WSGI сервер, як Gunicorn # В продакшн використовувати HTTPS
    app.run(debug=False, host='0.0.0.0', port=5000)
```

Додаток Б

Діаграма компонентів

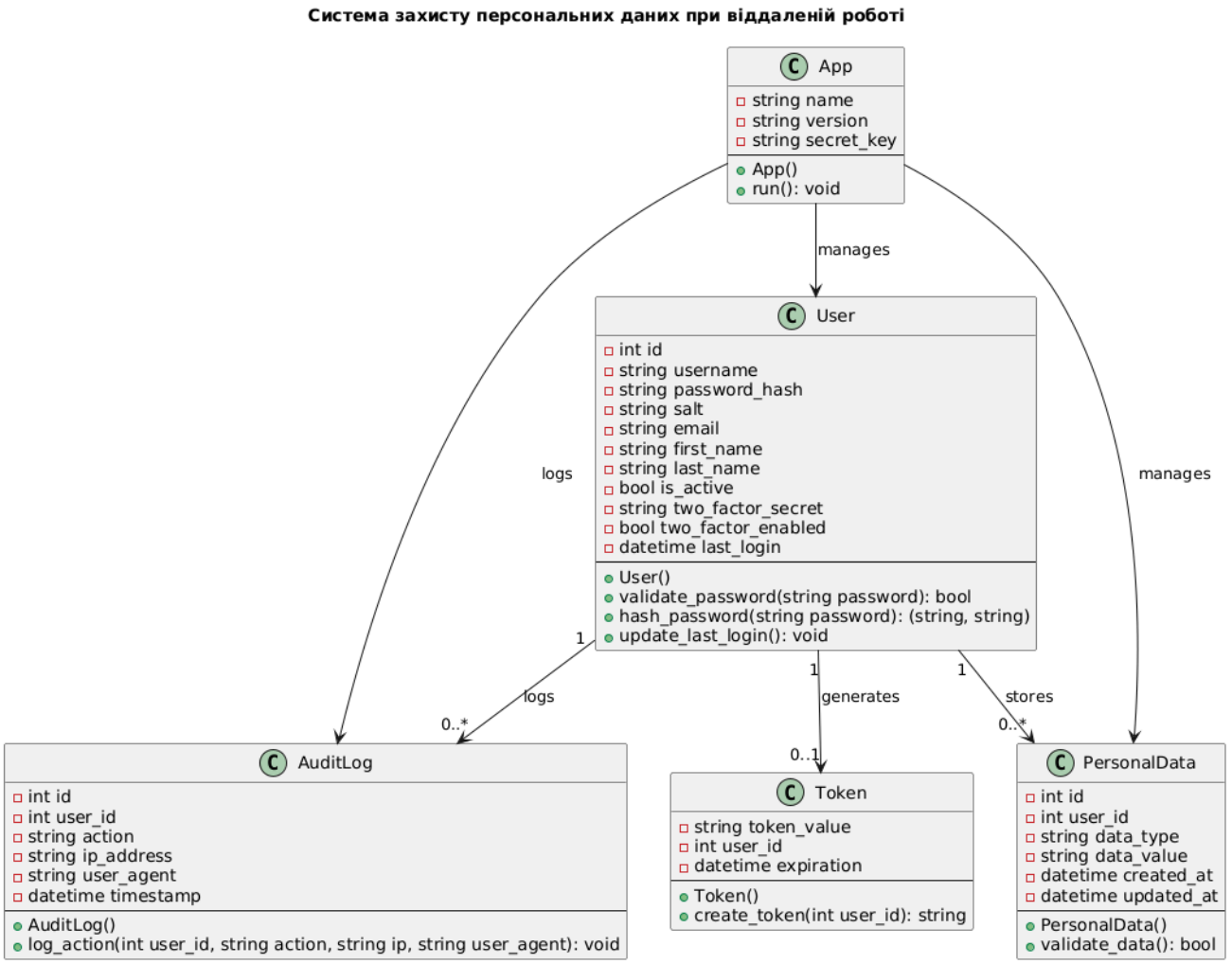






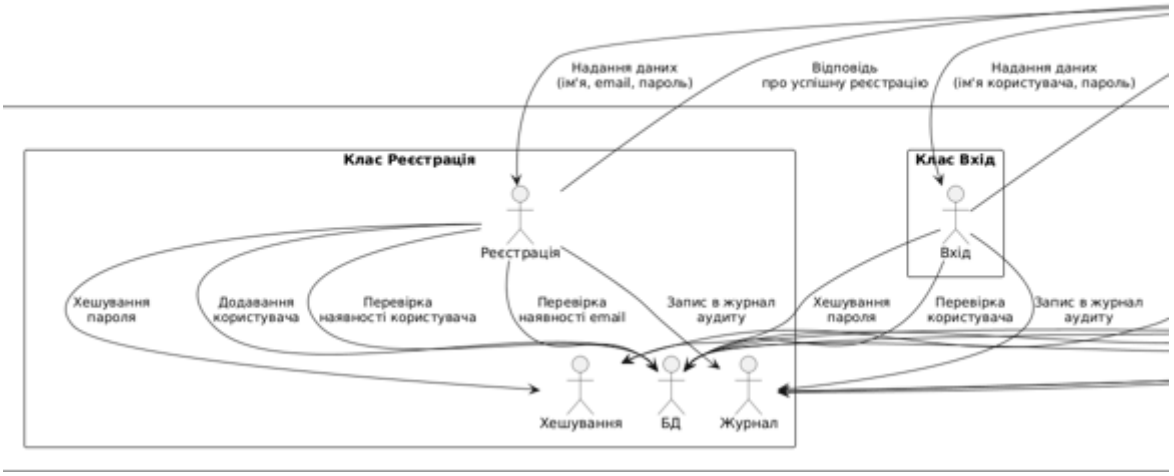
Додаток В

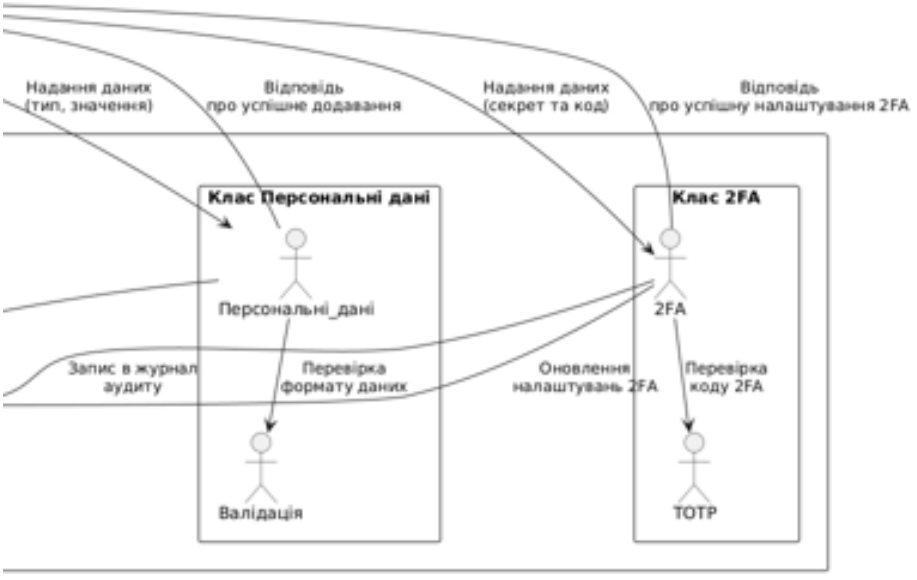
Діаграма класів



Додаток Г

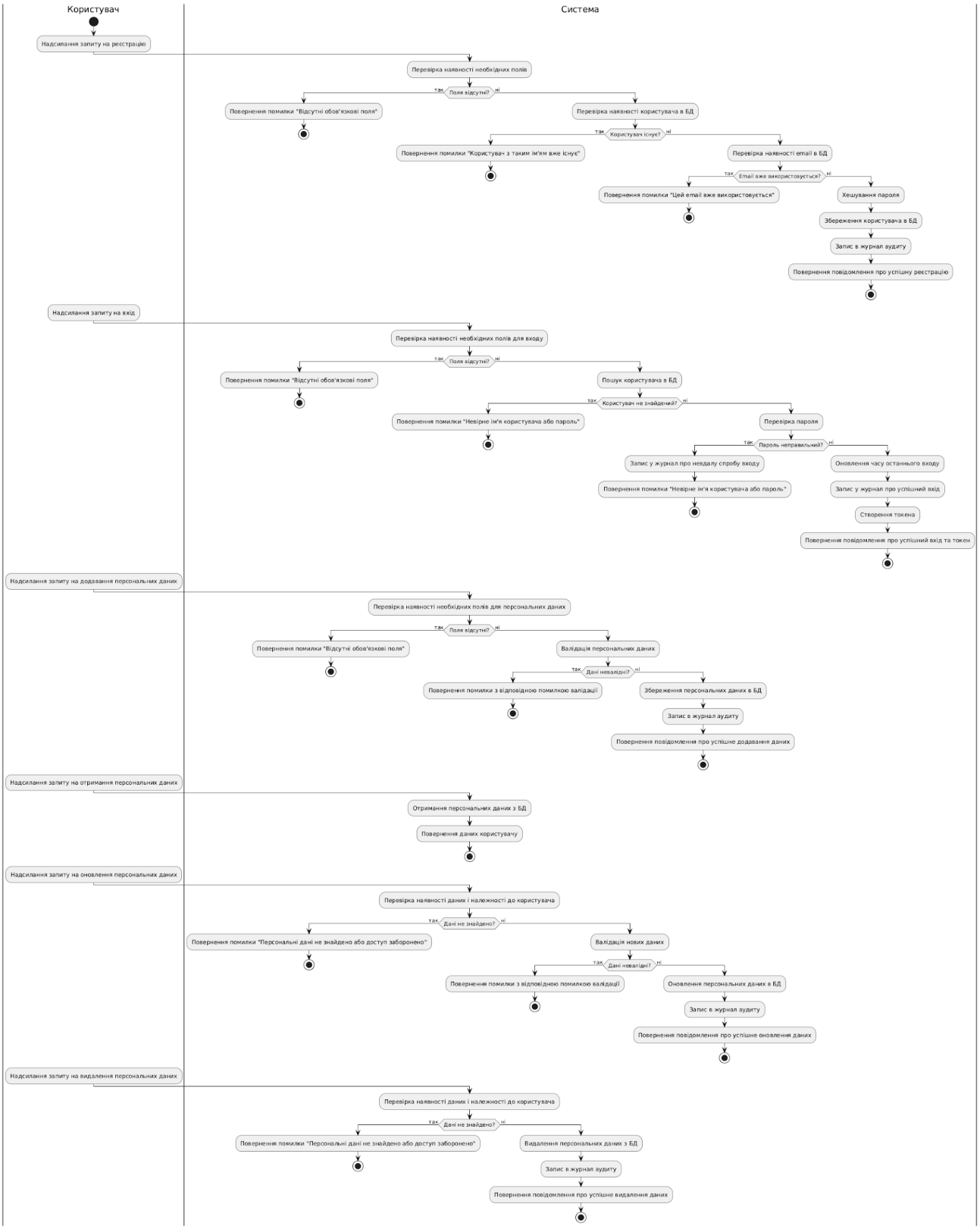
Діаграма кооперацій





Додаток Д

Діаграма діяльності

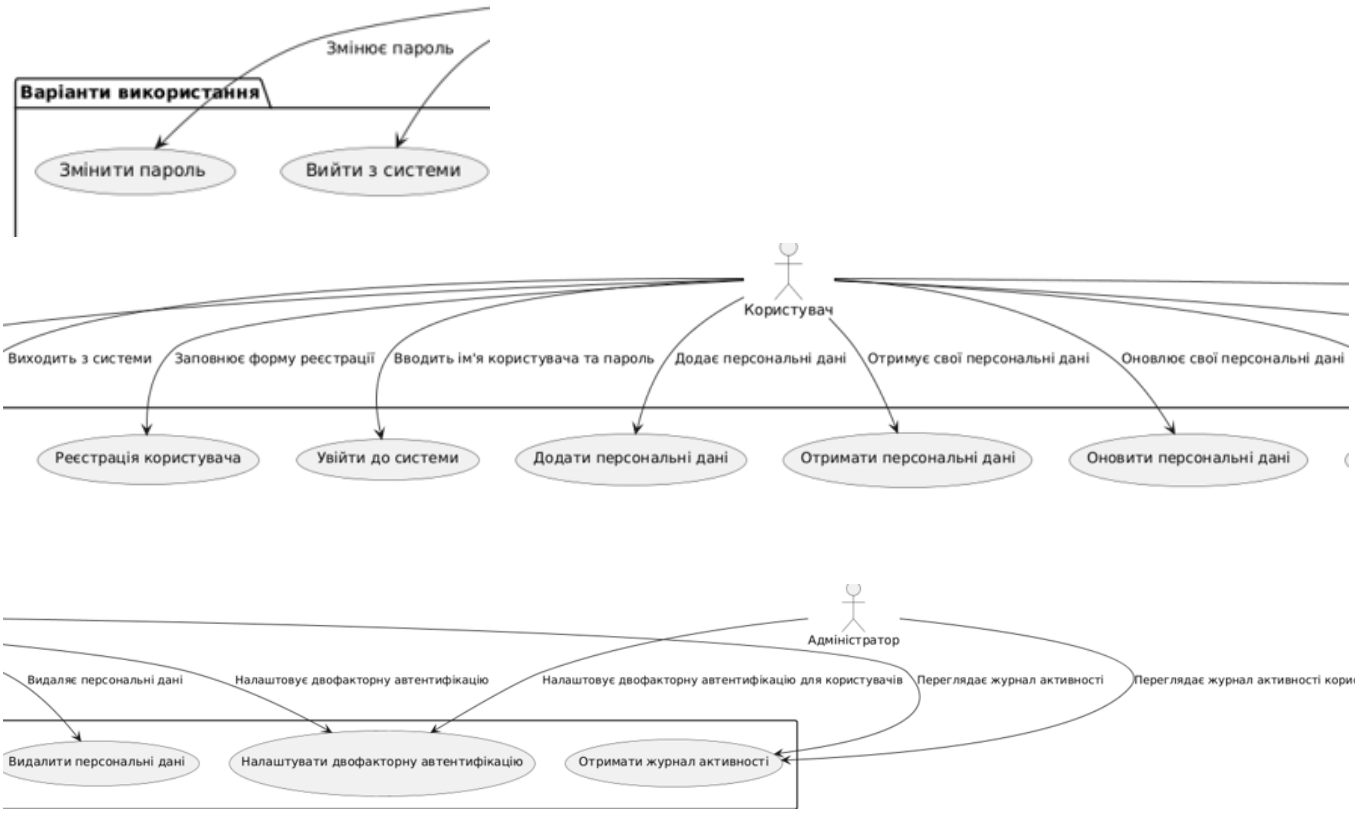


Додаток Е

Діаграма послідовностей

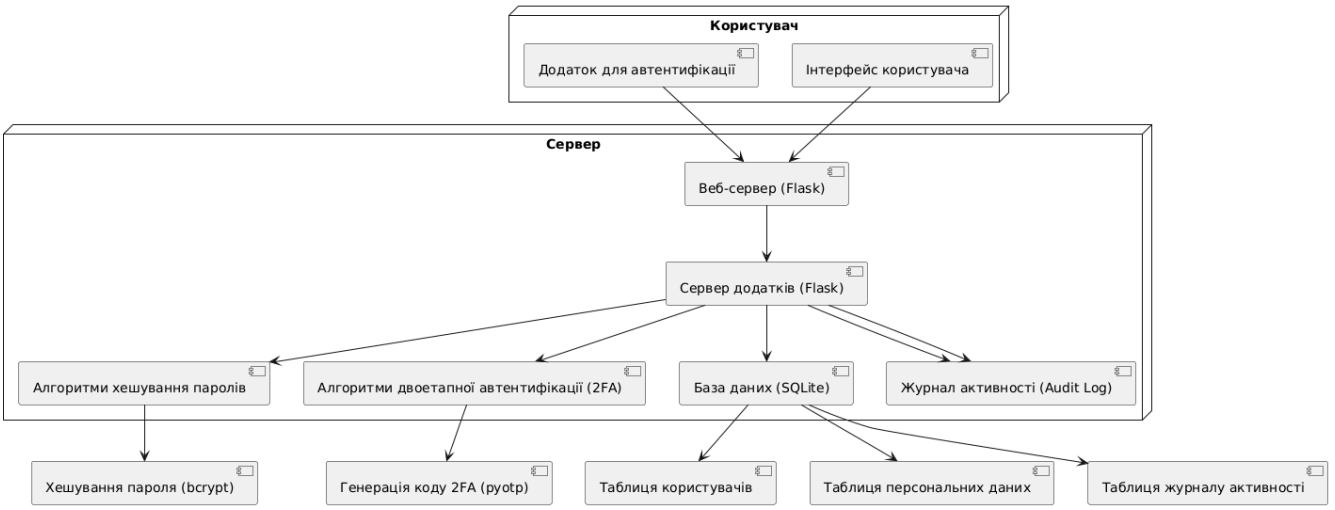
Додаток Ж

Діаграма варіантів використання



Додаток З

Діаграма розгортання



Додаток І

Діаграма станів