Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська дипломна робота на тему:

«Засіб захисту чутливих даних»

Виконав: студент 4 курсу групи 1БС-19 б
спеціальності 125 Кібербезпека
_____ Попова І. С.
Керівник: к. т. н., доцент каф. ЗІ
_____ Лукічов В. В.
« 19 » червня 2023 р.
Рецензент: к. т. н., доц., доцент каф. ОТ
_____ Майданюк В. П.
« 19 » червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«19» червня 2023 р.

Вінниця ВНТУ – 2023 року

4. Зміст текстової частини: Вступ. 1. Аналіз потенційних проблем та загроз чутливих даних. 2. Розробка засобу захисту чутливих даних. 3. Експлуатація та тестування розробленого програмного забезпечення. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: схеми, рисунки, скріншоти.

6. Консультанти розділів дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Лукічов В. В., к. т. н., доц. каф. ЗІ	20.03.23	19.06.23
2	Лукічов В. В., к. т. н., доц. каф. ЗІ	20.03.23	19.06.23
3	Лукічов В. В., к. т. н., доц. каф. ЗІ	20.03.23	19.06.23

7. Дата видачі завдання: «20» березня 2023 р.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

№	Назва етапів комплексної бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	20.03.23 – 26.03.23	
2	Аналіз джерел за напрямком комплексної бакалаврської дипломної роботи	27.03.23 – 09.04.23	
3	Розробка архітектури та алгоритму засобу	10.04.23 – 23.04.23	
4	Програмна реалізація та тестування	24.04.23 – 21.05.23	
5	Аналіз результатів тестування, висновки	22.05.23 – 24.05.23	
6	Оформлення пояснювальної записки	25.05.23 – 31.05.23	
7	Попередній захист та доопрацювання БДР	01.06.23 – 15.06.23	
8	Представлення БДР на захист	16.06.23 – 19.06.23	
9	Захист БДР	20.06.23 – 23.06.23	

Студент Попова І. С

Керівник роботи Лукічов В.В

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., проф.
_____ В. А. Лужецький
«20» березня 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Поповій Ірині Сергіївні

1. Тема роботи: «Засіб захисту чутливих даних»
керівник роботи: Лукічов Віталій Володимирович, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 20 березня 2023 року №67.
2. Строк подання студентом роботи 19 червня 2023 р.
3. Вихідні дані до роботи:
- визначення чутливих даних;
- наслідки втрати чутливих даних;
- потенційні загрози безпеки;
- алгоритми і способи шифрування даних.

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 84 сторінок формату А4, на яких є 12 рисунків, 5 таблиць, список використаних джерел містить 27 найменувань.

Ця бакалаврська робота присвячена розробці програмного засобу для захисту чутливих даних. В процесі дослідження алгоритмів та методів шифрування даних був обраний алгоритм Advanced Encryption Standard (AES) як найбільш підходящий для забезпечення безпеки і конфіденційності інформації. Реалізація розробленого програмного засобу включає систему реєстрації і авторизації користувачів з шифруванням введених даних. Остаточний програмний засіб надає надійний рівень захисту чутливих даних, забезпечуючи їхню цілісність і конфіденційність.

Ключові слова: захист програмного забезпечення, захист даних, конфіденційність, інформаційна безпека, шифрування, аутентифікація, авторизація, криптографія.

ABSTRACT

The bachelor 's diploma work consists of 84 pages of A4 format, on which there are 12 figures, 5 tables, the list of used sources contains 27 titles.

This bachelor thesis is devoted to the development of a software tool for the protection of sensitive data. In the process of researching data encryption algorithms and methods, the Advanced Encryption Standard (AES) algorithm was chosen as the most suitable for ensuring the security and confidentiality of information. The implementation of the developed software includes a user registration and authorization system with encryption of entered data. The ultimate software tool provides a reliable level of protection for sensitive data, ensuring its integrity and confidentiality.

Keywords: software protection, data protection, privacy, information security, encryption, authentication, authorization, cryptography.

ЗМІСТ

ВСТУП

У сучасному цифровому світі, обмін та збереження інформації за допомогою різних електронних засобів стали невід'ємною частиною нашого повсякденного життя. Комунікація, фінансові транзакції, медичні записи, персональні дані — усе це передається та зберігається в електронному форматі.

Наприклад, електронна пошта дозволяє нам легко і швидко передавати повідомлення та документи за допомогою Інтернету. Банківські та фінансові операції проводяться через онлайн-банкінг, де надійність та конфіденційність даних є вирішальними факторами. Також, соціальні мережі дозволяють нам обмінюватися особистою інформацією та фотографіями з родиною та друзями. Однак, разом зі зручністю та ефективністю, цей цифровий ландшафт приносить і серйозні ризики щодо безпеки чутливих даних.

Захист чутливих даних є надзвичайно актуальною темою в сучасному інформаційному суспільстві. З розвитком технологій та зростанням цифрової комунікації з'являються нові можливості для зловмисників, які намагаються отримати доступ до цінної інформації. Це стосується як персональних даних, так і конфіденційної інформації, що належить організаціям, урядовим структурам, медичним установам та іншим суб'єктам. Несанкціонований доступ до таких даних може мати серйозні наслідки, включаючи фінансові втрати, порушення приватності та пошкодження репутації.

Актуальність теми даної бакалаврської дипломної роботи полягає в необхідності розробки ефективного засобу захисту чутливих даних, які б дозволяли запобігти несанкціонованому доступу, зміні та розголошенню цих даних. Захист чутливих даних стає ключовим фактором для забезпечення конфіденційності, цілісності та доступності інформації, а також виконання вимог щодо захисту персональних даних, встановлених законодавством.

Метою даної роботи є підвищення ефективності захисту чутливих даних під час авторизації та реєстрації.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- Визначити типи чутливих даних та їх важливість.
- Охарактеризувати потенційні загрози безпеки даних.
- Визначити основні способи захисту персональної інформації.
- Описати архітектуру та технологій для реалізації засобу захисту.
- Розробити систему аутентифікації та авторизації у програмі.
- Впровадити криптографічні методи захисту чутливих даних.
- Описати інтерфейсу та функціонал розробленого програмного засобу.
- Провести тестування програмного засобу та визначити його ефективність та безпеку.

1 АНАЛІЗ ПОТЕНЦІЙНИХ ПРОБЛЕМ ТА ЗАГРОЗ ЧУТЛИВИХ ДАНИХ

1.1 Визначення типів чутливих даних та їх важливості

Персональні дані — відомості чи сукупність відомостей за допомогою яких фізична особа може бути ідентифікована. Персональні дані відносяться до інформації, яка стосується конкретної фізичної особи. Це можуть бути будь-які дані, які дозволяють прямо або опосередковано ідентифікувати особу, такі як ім'я, прізвище, адреса, номер телефону, електронна пошта, фотографія, номер паспорта, медична інформація, геолокація тощо [1].

Персональні дані поділяються на дві категорії: загальні та особливі, що також відомі як чутливі дані.

Загальна категорія персональних даних включає такі відомості, як прізвище та ім'я, дату та місце народження, громадянство, сімейний стан, псевдонім, дані, зазначені в посвідченні водія, економічний і фінансовий стан, дані про майно, банківські реквізити, підпис, дані з актів цивільного стану, номер пенсійної справи, адресу місця проживання, дипломи про освіту, професійну підготовку та інші подібні відомості.

Особлива категорія персональних даних (чутливі дані) охоплює інформацію, яка вважається особливо чутливою та приватною. До неї належать дані про расове, етнічне та національне походження, політичні, релігійні та світоглядні переконання, членство в політичних партіях, професійних спілках, релігійних організаціях чи громадських організаціях світоглядної спрямованості, стан здоров'я (медичні дані), статеве життя, біометричні дані, генетичні дані, інформація про притягнення до адміністративної чи кримінальної відповідальності, застосування до особи заходів у рамках досудового розслідування, вжиття щодо особи заходів, передбачених законодавством щодо оперативно-розшукової діяльності, вчинення щодо особи насильства та інші подібні дані [2].

Різні типи конфіденційних даних можуть завдати величезної шкоди особі, компанії чи державній установі, якщо їх зламати. Далі розглянемо кілька типових прикладів персональних даних.

Інформація про статок і статус доходу організації або дані фінансового рахунку. Це включає номери банківських рахунків і маршрутизації, дані кредитної/дебетової картки, визначені стандартом безпеки даних індустрії платіжних карток (PCI DSS), записи кредитної історії та податкові декларації. Розкриття фінансової інформації може загрожувати фінансовими втратами або крадіжкою особистих даних у разі її зламу.

Будь-яка інформація, як-от стан здоров'я особи, умови, догляд, лікування та інформація, пов'язана з медичним страхуванням. Якщо особисту медичну інформацію буде зламано, конфіденційність жертви буде під загрозою [3].

Інформація, необхідна для доступу до системи, програми, пристрою чи фізичного розташування, як-от імена користувачів, паролі та персональні ідентифікаційні номери (PIN). Він також включає дані, що зберігаються у фізичних пристроях автентифікації, таких як картки-брелоки та брелоки, і біометричні дані, отримані за допомогою сканування обличчя чи відбитків пальців. Крадіжка облікових даних поставила б під загрозу інформаційну безпеку та конфіденційність.

Дані клієнтів, як-от імена, адреси, дії веб-перегляду, і контактна інформація, як-от номери телефонів і адреси електронної пошти, які не включають їхні фінансові дані, PHI або облікові дані. Недотримання конфіденційності клієнтів може призвести до штрафів і судових позовів проти компаній, які керують їхньою інформацією.

Інформація, яка надає та зберігає переваги для бізнесу чи державного органу, як-от інтелектуальна власність, військова таємниця або дані бізнес-аналітики. У разі компрометації з боку супротивника або конкурента жертва ризикує втратити свою конкурентну перевагу на ринку або в геополітичних і військових конфліктах.

Особисті дані, які часто називають інформацією, яка дозволяє ідентифікувати особу, — це інформація, яка може бути унікально використана для ідентифікації або перевірки особи чи організації. Персональні дані можуть бути конфіденційними або неконфіденційними. Наприклад, імена та номери телефонів можна легко знайти в публічних архівах, і зловмиснику буде важко завдати шкоди людині, маючи лише цю інформацію. Крім того, номер соціального страхування людини може бути використаний для викрадення її особи, тому він вважається конфіденційною ідентифікаційною інформацією [3].

Обробка персональних даних відноситься до процесу збирання, захисту, збереження, використання, передачі та видалення інформації, яка ідентифікує конкретну особу або може бути пов'язана з нею. Цей процес є невід'ємною частиною багатьох діяльностей організацій, установ та індивідуальних користувачів, які займаються обробкою даних.

Обробка персональних даних включає такі етапи:

1) Збір даних: отримання персональних даних від особи, до якої вони належать, або з інших джерел, які мають право передавати ці дані.

2) Захист даних: застосування відповідних заходів безпеки для захисту персональних даних від несанкціонованого доступу, втрати або пошкодження.

3) Збереження даних: збереження персональних даних протягом визначеного періоду відповідно до правових вимог та цілей обробки.

4) Використання даних: використання персональних даних відповідно до підстави, для якої вони були зібрані, таких як виконання договору, згода особи, виконання правових обов'язків або законних інтересів.

5) Передача даних: передача персональних даних третім особам або органам згідно з встановленими правилами та обмеженнями.

6) Видалення даних: видалення або анонімізація персональних даних після закінчення строку їх збереження або після втрати юридичної підстави для їх обробки [4].

Внутрішні політики безпеки даних — це внутрішні нормативні документи, які встановлюють правила та процедури щодо захисту персональних даних в рамках організації. Ці політики розробляються власниками та операторами персональних даних з урахуванням вимог законодавства та специфіки їх діяльності [2].

Внутрішні політики безпеки даних визначають:

1) Процедури збирання, збереження та обробки персональних даних: правила збирання даних, встановлення строків їх збереження, управління доступом до даних та використання внутрішніх систем для обробки даних.

2) Заходи безпеки: політика щодо захисту даних від несанкціонованого доступу, втрати або пошкодження, використання шифрування, встановлення паролів, контроль доступу до інформаційних систем, резервне копіювання даних тощо.

3) Внутрішню організацію: розподіл обов'язків та відповідальності в рамках організації, проведення навчань з питань безпеки даних, ведення реєстру обробки персональних даних, контроль виконання політик та процедур тощо.

4) Політику відповідності: забезпечення виконання внутрішніх політик безпеки даних з урахуванням вимог законодавства та нормативних актів, а також встановлення процедур моніторингу та аудиту для перевірки відповідності.

Важливість чутливих даних полягає в тому, що їх незаконне збирання, використання або розголошення можуть мати серйозні наслідки для особи, що стосується цих даних. Вони можуть бути використані для дискримінації, шантажу, порушення приватності, а також для злочинів, таких як крадіжка особистості, шахрайство, кібератаки тощо.

Законодавство в сфері захисту приватних даних має на меті регулювання збору, обробки, збереження та передачі персональних даних з метою захисту приватності та прав осіб, чиї дані обробляються. Різні країни та регіони мають

власні закони та нормативні акти, які встановлюють правила та вимоги для організацій та індивідуалів, що займаються обробкою персональних даних.

Одним з найважливіших і впливових законодавчих актів в сфері захисту персональних даних в Європі є Загальний регламент про захист персональних даних (General Data Protection Regulation - GDPR). Введений в дію 25 травня 2018 року, GDPR є обов'язковим для всіх країн-членів Європейського Союзу і має застосовність до всіх організацій, які збирають, обробляють та зберігають персональні дані громадян ЄС, незалежно від місцезнаходження таких організацій.

Регламент GDPR встановлює положення і вимоги щодо обробки особистих даних суб'єктів даних всередині Європейського Союзу. З метою забезпечення приватності, бізнес-процеси, які опрацьовують персональні дані, повинні бути побудовані з урахуванням принципу «приватність за призначенням і за замовчуванням». Це означає, що персональні дані повинні бути збережені за допомогою псевдонімів або повної анонімізації, а також застосовувати налаштування найвищого рівня приватності за замовчуванням, щоб дані не стали доступними публічно без явної згоди та не могли бути використані для ідентифікації суб'єкта без додаткової інформації, яка зберігається окремо.

Обробка особистих даних можлива лише за наявності законних підстав, визначених регламентом, або з явної, очевидної згоди власника даних. В будь-який час користувачі мають право відкликати цю згоду.

Контролер персональних даних має чітко вказати, які дані збираються, яким чином і з якою метою вони опрацьовуються, тривалість зберігання та чи передаються вони третім сторонам. Користувачі мають право запитати мобільну копію своїх персональних даних, які зібрані контролером, у загальноприйнятому форматі, а також право на вилучення своїх даних за певних обставин.

Державні органи та підприємства, які систематично обробляють персональні дані, повинні призначити посаду співробітника з питань захисту

даних (DPO), який відповідає за виконання GDPR. Підприємства повинні повідомляти про будь-які порушення захисту даних, які мають негативний вплив на конфіденційність користувачів, протягом 72 годин.

Регламент встановлює санкції для порушень в області захисту персональних даних. Основні типи санкцій, які можуть бути застосовані, включають:

1) Фінансові санкції: GDPR передбачає значні адміністративні штрафи для порушників. Залежно від характеру порушення, максимальна сума штрафу може становити до 20 мільйонів євро або 4% річного глобального обороту підприємства.

2) Попередження та застереження: Уповноважений орган з захисту даних може накласти попередження на порушника та вимагати виправлення недоліків у системі обробки даних.

3) Тимчасове або постійне обмеження обробки даних: Якщо порушення є серйозним, уповноважений орган може заборонити або обмежити обробку персональних даних підприємством або організацією.

4) Припинення діяльності: У випадках серйозних порушень GDPR, орган з контролю може заборонити підприємству займатися обробкою персональних даних або скасувати його ліцензію.

5) Відшкодування збитків: Суб'єкти даних, які постраждали в результаті порушення GDPR, мають право на відшкодування збитків, спричинених витоком або неправомірним використанням їхніх персональних даних [5].

Закон України «Про захист персональних даних» було прийнято 1 червня 2010 року, він набрав чинності з 1 січня 2011 року. Однак, цей закон був предметом значної критики і вважався недосконалим через відсутність розрізнення між вразливими та звичайними персональними даними.

З метою поліпшення системи захисту персональних даних, 1 січня 2014 року набрав чинності Закон України "Про внесення змін до деяких законодавчих актів України щодо удосконалення системи захисту

персональних даних". Цим законом повноваження контролю за дотриманням законодавства про захист персональних даних були покладені на Уповноваженого Верховної Ради України з прав людини, згідно з вимогами Конвенції Ради Європи про захист осіб у зв'язку з автоматизованою обробкою персональних даних.

Згідно з положеннями статті 8 Закон України "Про захист персональних даних" суб'єктом персональних даних є фізична особа, персональні дані якої обробляються і яка має право:

1) знати про джерела збирання, місцезнаходження своїх персональних даних, мету їх обробки, місцезнаходження або місце проживання (перебування) володільця чи розпорядника персональних даних або дати відповідне доручення щодо отримання цієї інформації уповноваженим ним особам, крім випадків, встановлених законом;

2) отримувати інформацію про умови надання доступу до персональних даних, зокрема інформацію про третіх осіб, яким передаються його персональні дані;

3) на доступ до своїх персональних даних;

4) отримувати не пізніш як за 30 календарних днів з дня надходження запиту, крім випадків, передбачених законом, відповідь про те, чи обробляються його персональні дані, а також отримувати зміст таких персональних даних;

5) пред'являти вмотивовану вимогу володільцю персональних даних із запереченням проти обробки своїх персональних даних;

6) пред'являти вмотивовану вимогу щодо зміни або знищення своїх персональних даних будь-яким володільцем та розпорядником персональних даних, якщо ці дані обробляються незаконно чи є недостовірними;

7) на захист своїх персональних даних від незаконної обробки та випадкової втрати, знищення, пошкодження у зв'язку з умисним приховуванням, ненаданням чи несвоєчасним їх наданням, а також на захист

від надання відомостей, що є недостовірними чи ганьблять честь, гідність та ділову репутацію фізичної особи;

8) звертатися із скаргами на обробку своїх персональних даних до Уповноваженого або до суду;

9) застосовувати засоби правового захисту в разі порушення законодавства про захист персональних даних;

10) вносити застереження стосовно обмеження права на обробку своїх персональних даних під час надання згоди;

11) відкликати згоду на обробку персональних даних;

12) знати механізм автоматичної обробки персональних даних;

13) на захист від автоматизованого рішення, яке має для нього правові наслідки [6].

Згідно цього закону, обробка персональних даних здійснюється відкрито і прозоро, відповідно до визначених цілей такої обробки. Для обробки персональних даних потрібна згода суб'єкта персональних даних, яка є добровільним волевиявленням фізичної особи після інформування про мету обробки. Згода може бути висловлена письмово або в іншій формі, що дає підстави вважати, що згода надана. Винятки з цього правила можуть бути передбачені законами України.

Персональні дані обробляються у такій формі, що дозволяє ідентифікувати особу, яка стосується цих даних, і зберігаються протягом необхідного періоду для законних цілей, з яких вони були зібрані або подальшої обробки.

Обробка персональних даних без згоди фізичної особи або без згоди уповноваженої нею особи заборонена, за винятком випадків, передбачених законом, і тільки в інтересах національної безпеки, економічного добробуту, прав людини або проведення Всеукраїнського перепису населення. У таких випадках обробка персональних даних без згоди суб'єкта може здійснюватися до отримання згоди, якщо це стає можливим.

За порушення недоторканості приватного життя, а саме за незаконне збирання, зберігання, використання, знищення, поширення конфіденційної інформації про особу або незаконна зміна такої інформації винна особа притягується до кримінальної відповідальності (стаття 182 Кримінального кодексу України) [7].

1.2 Характеристика потенційних загроз безпеки даних

Потенційні загрози безпеки даних можуть мати серйозні наслідки для індивідуальної приватності, фінансової стабільності, репутації та безпеки особистості. Далі детально розглянемо деякі характеристики потенційних загроз безпеки даних.

Несанкціонований доступ означає отримання особами доступу до даних, мереж, кінцевих точок, програм або пристроїв організації без дозволу. Це тісно пов’язане з автентифікацією – процесом, який перевіряє особу користувача під час доступу до системи. Зламані або неправильно налаштовані механізми автентифікації є основною причиною доступу неавторизованих сторін.

Інші поширені причини несанкціонованого доступу:

– Ненадійні паролі, вибрані користувачами, або паролі, спільні для різних служб.

– Атаки соціальної інженерії, насамперед фішинг, під час яких зловмисники надсилають повідомлення, видаючи себе за законних осіб, часто з метою викрадення облікових даних користувача.

– Зламані облікові записи — зловмисники часто шукають вразливу систему, компрометують її та використовують для отримання доступу до інших, більш безпечних систем.

– Інсайдерські загрози — зловмисний інсайдер може використовувати своє становище для отримання несанкціонованого доступу до систем компанії.

– Зловмисне програмне забезпечення Zeus — використовує ботнети для отримання несанкціонованого доступу до фінансових систем шляхом крадіжки облікових даних, банківської інформації та фінансових даних.

– Cobalt strike — комерційний інструмент тестування на проникнення, який використовується для фішингу та отримання несанкціонованого доступу до систем.

Блокування несанкціонованого доступу відіграє центральну роль у запобіганні витоку даних. Однак надійна програма безпеки використовує «поглиблений захист» — кілька рівнів захисту безпеки, намагаючись пом’якшити атаки задовго до того, як зловмисники досягнуть чутливої системи. Додаткові рівні безпеки включають захист мережі, захист кінцевої точки та захист даних.

Типове порушення безпеки відбувається в три етапи:

1) Дослідження — зловмисник шукає слабкі місця або вразливі місця в організаційних системах, людях або процесах.

2) Мережева/соціальна атака — зловмисник намагається проникнути на периметр мережі, уникаючи захисту мережі або використовуючи соціальну інженерію, щоб обманом змусити людей надати доступ, дані чи облікові дані.

3) Ексфільтрація — як тільки зловмиснику вдається отримати доступ, він може викрасти цінні активи або завдати шкоди в точці входу, а також виконати бічний рух, щоб отримати доступ до додаткових, більш цінних систем.

Отримавши несанкціонований доступ до організаційних систем або облікових записів користувачів, зловмисники можуть:

– Викрасти або знищити особисті дані.

– Викрасти гроші або товари.

– Викрасти ідентифікаторів користувачів.

– Компрометувати системи та використовуйте їх для нелегітимної чи злочинної діяльності.

– Саботувати організаційні системи або псувати веб-сайти.

– Заподіяти фізичні збитки – отримавши доступ до підключених пристроїв [8].

Крадіжка особистих даних відбувається, коли хтось використовує особисту ідентифікаційну інформацію іншої особи, як-от її ім’я, ідентифікаційний номер або номер кредитної картки, без її дозволу, для вчинення шахрайства чи інших злочинів. Термін крадіжка особистих даних був введений у 1964 році. З того часу визначення крадіжки особистих даних було законодавчо визначено як у Великобританії, так і в США як крадіжка особистої інформації.

Зловмисник навмисно використовує чужу особу як спосіб отримати фінансову вигоду чи отримати кредит та інші вигоди і, можливо, завдати невигідного становища чи збитків іншій особі. Особа, чию особу було викрадено, може зазнати несприятливих наслідків, особливо якщо її помилково притягнуть до відповідальності за дії злочинця.

Особиста інформація зазвичай включає ім’я людини, дату народження, номер соціального страхування, номер водійського посвідчення, номер банківського рахунку або кредитної картки, PIN-коди, електронні підписи, відбитки пальців, паролі або будь-яку іншу інформацію, яка може бути використана для доступу до фінансової інформації особи.

Визначити зв’язок між витоком даних і крадіжкою особистих даних є складним завданням, насамперед через те, що жертви крадіжки особистих даних часто не знають, як була отримана їх особиста інформація. Відповідно до звіту, зробленого для FTC, крадіжка особистих даних не завжди виявляється окремими жертвами. Шахрайство з особистими даними часто, але не обов’язково є наслідком крадіжки особистих даних. Хтось може викрасти або незаконно привласнити особисту інформацію, не вчиняючи потім крадіжки особистих даних, використовуючи інформацію про кожну особу, наприклад, коли відбувається серйозне порушення даних.

Ґюнтер Оллманн, головний технічний директор відділу безпеки Microsoft: «Вас цікавить крадіжка кредитних карток? Для цього є додаток». Ця

заява підсумовує легкість, з якою ці хакери отримують доступ до будь-якої інформації в Інтернеті. Нову програму для зараження комп'ютерів користувачів назвали Zeus, і програма настільки зручна для хакерів, що з нею може працювати навіть недосвідчений хакер.

Хоча хакерська програма проста у використанні, цей факт не применшує руйнівних наслідків, які Zeus (або інше програмне забезпечення, подібне до Zeus) може завдати комп’ютеру та користувачеві. Наприклад, такі програми, як Zeus, можуть викрадати дані кредитних карток, важливі документи та навіть документи, необхідні для внутрішньої безпеки. Якби хакер отримав цю інформацію, це означало б крадіжку особистих даних або навіть можливу терористичну атаку [9].

Крадіжки особистих даних поділяються на наступні типи:

– Крадіжка особи злочинця (видання за іншу особу під час затримання за злочин);

– Крадіжка фінансових даних (використання особистих даних іншої особи для отримання кредиту, товарів і послуг);

– Клонування ідентичності (використання інформації про іншу людину для підтвердження її особи в повсякденному житті);

– Крадіжка медичних даних (використання особистих даних іншої особи для отримання медичної допомоги або ліків).

Крадіжка особистих даних може бути використана для сприяння або фінансування інших злочинів, зокрема нелегальної імміграції, тероризму, фішингу та шпигунства. Бувають випадки клонування ідентифікаційних даних для атаки на платіжні системи, включаючи онлайн-обробку кредитних карток і медичне страхування.

Методи отримання та використання особистої інформації:

– Отримання особистих даних із зайвого IT-обладнання та носіїв інформації, включаючи ПК, сервери, КПК, мобільні телефони, USB-накопичувачі та жорсткі диски, які були необережно утилізовані на громадських сміттєзвалищах, віддані або продані без належної дезобробки;

– Використання публічних записів про окремих громадян, опублікованих в офіційних реєстрах, таких як списки виборців;

– Крадіжка банківських або кредитних карток, ідентифікаційних карток, паспортів, токенів автентифікації зазвичай шляхом кишенькової крадіжки, злому або крадіжки пошти;

– Загальновідомі схеми опитування, які пропонують перевірку облікового запису, наприклад «Яке дівоче прізвище вашої матері?», «Яка була ваша перша модель автомобіля?» або «Як звали вашу першу домашню тварину?»;

– Збирання інформації з банківських або кредитних карток за допомогою скомпрометованих або портативних пристроїв для зчитування карток і створення карток-клонів;

– Викрадення особистої інформації з комп'ютерів за допомогою проломів у безпеці веб-переглядача або зловмисного програмного забезпечення, наприклад троянських програм для реєстрації натискань клавіш або інших форм шпигунського програмного забезпечення;

– Злом комп'ютерних мереж, систем і баз даних для отримання персональних даних, часто у великих кількостях;

– Використання порушень, які призводять до публікації або більш обмеженого розкриття особистої інформації, такої як імена, адреси, номер соціального страхування або номери кредитних карток;

– Реклама фіктивних пропозицій про роботу для накопичення резюме та заявок, які зазвичай розкривають імена кандидатів, домашні та електронні адреси, номери телефонів, а іноді й їхні банківські реквізити;

– Використання внутрішнього доступу та зловживання правами привілейованих ІТ-користувачів на доступ до персональних даних у системах їхніх роботодавців;

– Проникнення в організації, які зберігають і обробляють великі обсяги або особливо цінну особисту інформацію;

– Видання себе за довірену організацію в електронних листах, SMS-повідомленнях, телефонних дзвінках або інших формах зв’язку, щоб змусити жертв розкрити свою особисту інформацію чи облікові дані для входу, як правило, на підробленому корпоративному веб-сайті чи у формі збору даних (фішинг);

– Атака грубою силою на слабкі паролі та використання натхненних здогадок для компрометації слабких запитань щодо скидання пароля.

– Отримання зліпків пальців для фальсифікації ідентифікації за відбитками пальців.

– Перегляд веб-сайтів соціальних мереж для пошуку особистих даних, опублікованих користувачами, часто використання цієї інформації, щоб виглядати більш достовірною в подальших діях соціальної інженерії.

– Перенаправлення електронної пошти чи пошти жертви для отримання особистої інформації та облікових даних, як-от дані кредитних карток, рахунки та виписки з банківських/кредитних карток, або для затримки виявлення нових рахунків і кредитних угод, відкритих зловмисниками на імена жертв.

– Використання фальшивих приводів для обману окремих осіб, представників служби підтримки клієнтів і працівників служби підтримки, щоб вони розкрили особисту інформацію та дані для входу або зміни паролів користувачів/прав доступу.

– Низький рівень безпеки/захисту конфіденційності на фотографіях, які легко натискати та завантажувати на сайтах соціальних мереж.

– Спілкування з незнайомцями в соціальних мережах і використання їхньою довіри для надання приватної інформації (Соціальна інженерія) [9].

Кібератаки спрямовані на пошкодження важливих документів і систем у корпоративній або персональній комп’ютерній мережі, а також отримання доступу до них. Кібератаки здійснюють як окремі особи, так і цілі організації в політичних, кримінальних або особистих цілях для знищення засекреченої інформації чи отримання доступу до неї [10].

Шкідливе програмне забезпечення, також відоме як зловмисне програмне забезпечення, є одним із найпоширеніших засобів для отримання персональної інформації. Це програми, створені зловмисниками з метою завдати шкоди комп'ютерам, мережам або пристроям користувачів, включаючи крадіжку персональних даних.

Деякі типи шкідливого програмного забезпечення, які можуть бути використані для отримання персональної інформації, включають:

– Віруси: це програми, які можуть розмножуватися і поширюватися, прикріплюючись до інших файлів або програм. Вони можуть спричинити руйнування даних і виконувати шкідливі дії, такі як збір персональних даних.

– Троянські програми: вони можуть приховуються від користувачів, здаючись за надійні або корисні програми. Вони можуть створювати зворотні двері в системі, які дозволяють зловмиснику отримувати доступ до персональних даних.

– Кейлогери: це програми, які записують натискання клавіш на комп'ютері або мобільному пристрої. Зловмисники можуть використовувати кейлогери для отримання паролів, особистих повідомлень і іншої персональної інформації, яку користувач вводить з клавіатури.

Розподілені атаки на відмову в обслуговуванні (DDoS-атаки) є ще одним типом кібератак, які можуть бути використані для отримання персональної інформації. Ці атаки спрямовані на перевантаження ресурсів мережі, серверів або інших компонентів, що призводить до перебоїв у роботі та відмови в обслуговуванні легітимних користувачів.

DDoS-атаки зазвичай виконуються за допомогою ботнетів, що складаються з великої кількості комп'ютерів або пристроїв, які були заражені шкідливим програмним забезпеченням або отримали неправомірний доступ. Зловмисники використовують ці ботнети для одночасного надсилання великого обсягу запитів до цільового об'єкта, перевантажуючи його ресурси і заважаючи нормальній роботі.

DDoS-атаки можуть бути використані як відволікаючий маневр для зловмисників, щоб виконати інші типи кібератак, такі як крадіжка персональних даних або вторгнення.

Фішинг — це метод кібератак, який використовується для отримання конфіденційної інформації, такої як логіни, паролі, номери кредитних карток чи інші особисті дані, шляхом підробки довіреної сторони. Зловмисники, які виконують фішинг-атаки, вдаються за допомогою електронних листів, повідомлень в соціальних мережах або підроблених веб-сайтів, щоб переконати потенційну жертву надати свої особисті дані або виконати певні дії.

Основна ідея фішингу полягає в тому, щоб створити вигляд легітимної організації або особи, з якою потенційна жертва взаємодіє регулярно. Наприклад, зловмисники надсилають електронний лист, який видавлюється за повідомлення від банку, компанії чи іншої довіреної сторони. Цей лист може містити привабливий заголовок або вимагати негайних дій, наприклад, оновлення облікового запису або підтвердження інформації. Посилання або вкладення в такому листі можуть направляти жертву на підроблену веб-сторінку, де їм буде запропоновано ввести свої особисті дані.

Зловмисники також можуть створювати підроблені профілі в соціальних мережах, використовуючи логотипи та інформацію реальних компаній або організацій. Вони можуть надсилати повідомлення або залишати коментарі, щоб залучити користувачів і впровадити фішинг-схему.

SQL-ін'єкція є одним з типів кібератак, що використовується для отримання несанкціонованого доступу до бази даних або виконання шкідливого коду на веб-сайтах або додатках, що використовують SQL-запити для взаємодії з базою даних. Ця атака використовується для впровадження шкідливого SQL-коду в SQL-запити, що виконуються на веб-сервері.

Основна ідея SQL-ін'єкції полягає в тому, щоб вставити шкідливий SQL-код у вхідні дані, які передаються до бази даних. Цей код може модифікувати або видаляти дані в базі даних, виконувати небезпечні запити або навіть

отримувати конфіденційну інформацію з бази даних. Атаки SQL-ін'єкції можуть мати наслідки, такі як виток конфіденційних даних, внесення змін у базу даних, пошкодження або втрата даних.

Зловмисні програми з вимогою викупу, відомі також як розшифровувальні віруси або рансомваре (ransomware), є одним із видів шкідливого програмного забезпечення, яке шифрує файли на комп'ютері або мережі і вимагає викупу за їх розблокування. Ці атаки є способом шантажу, де зловмисники намагаються використовувати цінні дані користувачів або організацій як засіб заробітку.

Основний сценарій дії зловмисних програм з вимогою викупу такий:

1) Шифрування файлів. Зловмисні програми з вимогою викупу шифрують файли на комп'ютері або мережі, застосовуючи сильні алгоритми шифрування. Це робить файли недоступними для легітимних користувачів, оскільки вони не можуть розшифрувати їх без спеціального ключа.

2) Вимога викупу. Після шифрування файлів, зловмисники повідомляють потерпілих про наявність шифрувального вірусу і вимагають викупу за розблокування файлів. Ця вимога може бути представлена у вигляді сповіщення на екрані, текстового файлу або через повідомлення електронної пошти. Зазвичай, зловмисники вимагають викупу у криптовалюті, такій як Bitcoin, оскільки це дає їм можливість приховати свою ідентичність.

3) Розблокування файлів. Якщо потерпілий погоджується на викуп, зловмисники надають йому спеціальний ключ або програму для розшифрування файлів. Після отримання розблокувального засобу, потерпілий може відновити доступ до своїх файлів, проте немає гарантії, що зловмисники дійсно відновлять доступ після отримання викупу [10].

1.3 Основні способи захисту персональної інформації

Для захисту персональної інформації існує кілька важливих способів. Далі розглянемо основні з них.

Використання сильних паролів є важливою складовою безпеки чутливих даних. Сильний пароль має бути унікальним, складатися з комбінації різних символів і бути достатньо довгим. Використовуйте поєднання великих і малих літер, цифр та спеціальних символів для створення паролю, який важко вгадати або підібрати методом перебору. Уникайте використання очевидних паролів, таких як "password" або "123456", а також особистої інформації, такої як імена членів сім'ї чи дати народження.

Рекомендується також використовувати унікальні паролі для кожного облікового запису, щоб запобігти розповсюдженню в разі витоку даних. Використання менеджера паролів може сприяти збереженню та керуванню безпечними паролями для різних сервісів.

Будьте обережні, не надсилайте свої паролі поштою чи повідомленнями, і не зберігайте їх у небезпечних місцях, таких як відкриті файли чи записки на комп'ютері. Регулярно змінюйте свої паролі, особливо після виявлення порушень безпеки або сумнівних активностей. Пам'ятайте, що сильний пароль є першим рядком захисту вашої персональної інформації та допомагає унеможливити несанкціонований доступ до ваших чутливих даних.

Багатофакторна аутентифікація є ефективним інструментом для забезпечення безпеки чутливих даних. У традиційній однофакторній аутентифікації користувач має представити лише один фактор, зазвичай пароль, для підтвердження своєї ідентичності. Однак, це може бути недостатньо, оскільки паролі можуть бути скомпрометовані або вгадані.

Багатофакторна аутентифікація вимагає від користувача представити не лише пароль, але й додатковий фактор, такий як фізичний об'єкт (такий як токен або смарт-карта) або біологічний атрибут (такий як відбиток пальця або розпізнавання обличчя). Це означає, що для успішної аутентифікації потрібно мати щонайменше два незалежних фактори.

Багатофакторна аутентифікація ускладнює завдання зловмисників, оскільки навіть якщо вони володіють паролем, вони не зможуть отримати

доступ до системи без додаткового фактора. Це значно підвищує безпеку і унеможливлює несанкціонований доступ до чутливих даних.

Популярні методи багатофакторної аутентифікації включають використання одноразових паролів, смс-повідомлень з кодами, біометричних даних та апаратних токенів. Інтеграція цих додаткових факторів з паролем дозволяє створити надійну систему аутентифікації, яка складніша для зловмисників [8].

Уважність при використанні Інтернету є критично важливим аспектом для забезпечення безпеки чутливих даних. З ростом кількості кіберзлочинів та загроз в Інтернеті, користувачі повинні бути обережні та свідомі своїх дій.

При перегляді веб-сайтів чи використанні онлайн-послуг, важливо бути уважним до надання особистої інформації. Необхідно уникати надання особистих даних, таких як номери соціального страхування, банківські реквізити або паролі, на ненадійних або неперевірених веб-сайтах.

Уважність також передбачає обережне ставлення до електронної пошти та повідомлень. Користувачі повинні бути обережними при відкриванні вкладок, що містять посилання або вкладення, особливо від невідомих або підозрілих відправників. Фішингові атаки широко використовуються для викрадення особистої інформації, тому користувачі повинні бути пильними та не відкривати підозрілі посилання чи надавати особисті дані без обдумування.

Крім того, важливо мати оновлені програми та антивірусне програмне забезпечення на своєму комп'ютері або пристрої. Періодичні оновлення допомагають виправити вразливості програм і запобігти злому чи атакам. Також рекомендується використовувати безпечні мережі Wi-Fi та уникати підключення до ненадійних або відкритих мереж, які можуть бути легко скомпрометовані [8].

Шифрування – це процес перетворення звичайного тексту (відкритого тексту) в незрозумілий формат (шифротекст) за допомогою спеціального алгоритму або ключа. Цей процес забезпечує конфіденційність інформації,

оскільки шифрований текст стає нечитабельним без використання правильного ключа або алгоритму розшифрування.

Шифрування використовується для захисту чутливих даних під час їх збереження, передачі або обробки. Воно забезпечує захист від несанкціонованого доступу та зловмисних дій, таких як перехоплення інформації або підробка даних.

Для детальнішого порівняння способів шифрування розробимо таблицю:

Таблиця 1.1 — Порівняння симетричного та асиметричного шифрування

Характеристика	Симетричне шифрування	Асиметричне шифрування
Ключі	Використовується один ключ для шифрування і розшифрування	Використовуються два різних ключі - публічний та приватний
Безпека ключа	Ключ потрібно безпечно обмінювати між сторонами, які спілкуються	Публічний ключ може бути розповсюджений, приватний ключ залишається виключно у власника
Швидкість шифрування	Швидше за рахунок використання одного ключа	Повільніше за рахунок використання більш складних алгоритмів
Ключовий обмін	Вимагає безпечного обміну ключем між сторонами	Не потребує обміну ключами між сторонами
Застосування	Часто використовується для шифрування великих обсягів даних, таких як файлів	Часто використовується для захисту комунікації та обміну ключами
Безпека	Залежить від безпеки самого ключа і його обміну	Забезпечує вищий рівень безпеки за рахунок використання двох ключів
Приклади алгоритмів	DES, AES, 3DES, Blowfish	RSA, ECC, DSA, ElGamal

Шифрування важливо в контексті захисту особистої інформації, фінансових даних, медичної інформації та інших чутливих даних, які можуть бути використані проти власника.

Існують різні методи шифрування, включаючи симетричне шифрування і асиметричне шифрування. У симетричному шифруванні використовується один і той же ключ для шифрування і розшифрування даних. У асиметричному шифруванні використовуються два різних ключі – публічний і приватний. Публічний ключ використовується для шифрування, а приватний ключ - для розшифрування. Асиметричне шифрування забезпечує більшу безпеку і дозволяє безпечний обмін ключами між комунікуючими сторонами [11].

У цьому розділі було проведено дослідження засобів захисту чутливих даних, починаючи з їх визначення та типів. Виявлено, що чутливі дані можуть включати різноманітну інформацію, таку як особисті дані, фінансову і медичну інформацію, комерційні та конфіденційні дані підприємств. Втрата чутливих даних або незаконний доступ до них можуть мати серйозні наслідки, такі як порушення приватності, фінансові втрати або пошкодження репутації організацій.

Далі, були розглянуті потенційні загрози безпеки даних, серед яких несанкціонований доступ до даних, використання шкідливого програмного забезпечення, фішинг, SQL-ін'єкції, зловмисні програми з вимогою викупу та розподілені атаки на відмову в обслуговуванні (DDoS-атаки). Кожна з цих загроз може стати причиною порушення безпеки даних і викликати серйозні наслідки для організацій та користувачів.

Також були розглянуті основні способи захисту персональної інформації. Ці способи включають використання сильних паролів, багатофакторну аутентифікацію, шифрування даних, встановлення оновлень програмного забезпечення та уважне ставлення до використання Інтернету. Використання цих заходів допомагає забезпечити конфіденційність, цілісність та доступність персональної інформації.

2 РОЗРОБКА ЗАСОБУ ЗАХИСТУ ЧУТЛИВИХ ДАНИХ

2.1 Вибір архітектури та технологій для реалізації засобу захисту

2.1.1 Архітектура

Архітектура програмного забезпечення — це розроблена структура додатка, яка включає визначення взаємодії компонентів інтерфейсу з внутрішніми процесами програми. Простіше кажучи, це своєрідний підхід, який визначає які функції за що відповідають та як вони взаємодіють між собою.

Архітектура програмного рішення виконує низку важливих завдань:

– визначає структуру додатка та дозволяє зрозуміти, як вона влаштована, на яких рівнях виконуються ті чи інші завдання та функції;

– визначає поведінку та взаємодію елементів, завдяки чому стає зрозуміло, що відбувається, якщо виконується певна дія;

– визначає значні та другорядні елементи, що дозволяє оцінити вартість розробки, зрозуміти, які елементи обов'язково впроваджувати, а від яких можна відмовитися з міркувань економії;

– допомагає зрозуміти, наскільки додаток масштабується, як складно буде впроваджувати нові функції та який стек технологій використовувати;

– дозволяє задовольнити потреби клієнта та адаптувати додаток під взаємовиключні вимоги, наприклад, високий рівень функціональності та визначення меж часу, у такому випадку стає зрозуміло, як це реалізувати;

– дозволяє зрозуміти логічні взаємозв'язки у додатку;

– дає можливість коректно вести документацію та чітко розписувати функціонал, що суттєво спрощує подальше обслуговування додатка, внесення змін та роботу з існуючим функціоналом [12].

Програма захисту чутливих даних — це програмне забезпечення, яке призначене для захисту конфіденційної і важливої інформації від

несанкціонованого досупу, втрати або пошкодження. Вона використовує криптографічні методи та інші техніки для забезпечення безпеки даних.

Основна мета програми захисту чутливих даних – забезпечити конфіденційність, цілісність та доступність інформації. Це досягається за допомогою таких функцій:

1) Шифрування даних. Програма використовує криптографічні алгоритми для перетворення чутливої інформації в зашифрований формат, який може бути розшифрований тільки з допомогою правильного ключа. Шифрування забезпечує конфіденційність даних, навіть якщо вони потрапляють в несанкціоновані руки.

2) Аутентифікація та авторизація. Програма вимагає, щоб користувачі представляли свідчення своєї ідентичності, такі як логіни та паролі, для доступу до чутливої інформації.

3) Захист від несанкціонованого доступу: Програма використовує заходи безпеки, такі як обмеження фізичного доступу до серверів або обмеження мережевого доступу до баз даних.

Далі представлена UML діаграма варіантів використання програмного засобу: (рис. 2.1)

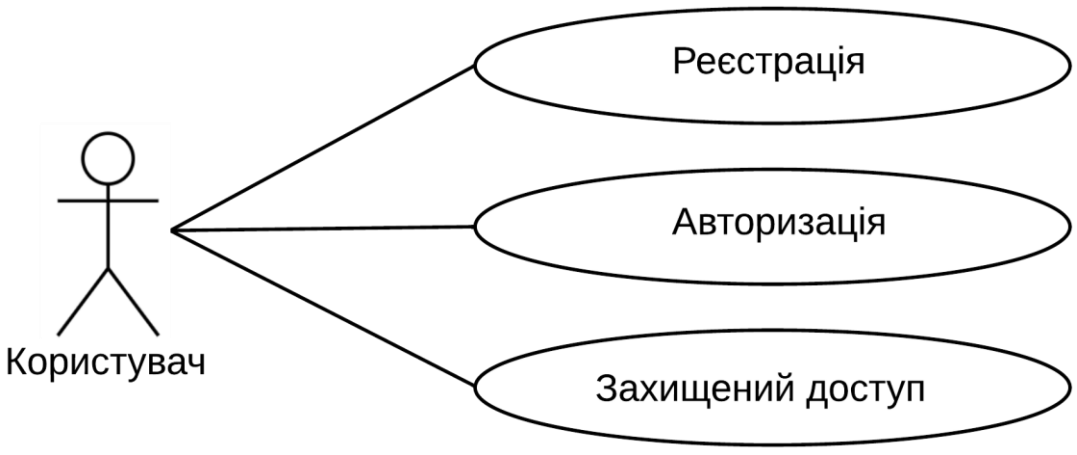


Рисунок 2.1 — UML діаграма варіантів використання програмного засобу

2.1.2 Мова програмування

Мова програмування — це формальний набір правил та синтаксису, який використовується для написання програмного коду. Вона визначає набір конструкцій, операторів та синтаксичних правил, які програмісти використовують для створення програм і взаємодії з комп'ютером.

Існує багато мов програмування, які можна використовувати для написання програми з захистом даних, авторизацією та реєстрацією. Далі детальніше розглянемо деякі з них.

1) Python — це високорівнева, інтерпретована мова програмування, яка часто використовується для розробки програм з захистом даних, авторизацією та реєстрацією. Ось кілька способів, якими Python може бути використана для цих цілей:

– Python має різноманітні бібліотеки для роботи з криптографічними алгоритмами. Наприклад, бібліотека cryptography дозволяє здійснювати шифрування, дешифрування, генерацію ключів та багато іншого. Вона підтримує різні алгоритми шифрування, такі як AES, RSA, і HMAC.

– Ця мова програмування має багато фреймворків, які сприяють реалізації систем авторизації та аутентифікації. Наприклад, фреймворк Django надає вбудовану підтримку для реалізації системи користувачів, яка включає функції реєстрації, входу, перевірки прав доступу та багато іншого.

– Python має різні фреймворки для веб-розробки, такі як Flask або Pyramid, які дозволяють створювати веб-додатки з функціями авторизації та реєстрації. Ці фреймворки надають розширюваність та дозволяють легко налаштувати механізми авторизації та аутентифікації.

– Дана мова програмування має багато бібліотек для роботи з базами даних. Ви можете використовувати SQLite, MySQL або PostgreSQL

для зберігання та обробки даних, пов'язаних з авторизацією та реєстрацією [13].

2) Java є високорівневою, об'єктно-орієнтованою мовою програмування, яка широко використовується для розробки програм з захистом даних, авторизацією та реєстрацією. Ось кілька способів, якими Java може бути використана для цих цілей:

– Java має багато вбудованих класів та бібліотек для роботи з криптографією. Наприклад, класи javax.crypto дозволяють здійснювати шифрування, дешифрування та генерацію ключів для різних криптографічних алгоритмів, таких як AES, RSA, DES та інші.

– Мова програмування має різні фреймворки та бібліотеки, які допомагають в реалізації систем авторизації та аутентифікації. Наприклад, фреймворк Spring Security надає широкі можливості для налаштування механізмів авторизації та аутентифікації в веб-додатках.

– Java має популярні фреймворки, такі як Spring або Java EE, які дозволяють створювати веб-додатки з функціями авторизації та реєстрації. Ці фреймворки надають вбудовану підтримку для обробки аутентифікації, авторизації, керування користувачами та багато іншого.

– Дана мова має широкий вибір бібліотек та фреймворків для роботи з різними системами керування базами даних (СКБД). Ви можете використовувати JDBC для підключення до реляційних баз даних, таких як MySQL або PostgreSQL, або використовувати ORM-фреймворки, такі як Hibernate, для спрощеної роботи з базами даних [14].

3) C# (C Sharp) є об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft. Вона також є популярним вибором для розробки програм з захистом даних, авторизацією та реєстрацією. Ось деякі способи, якими C# може бути використана для цих цілей:

– C# надає вбудовану підтримку для криптографічних операцій. Класи, такі як System.Security.Cryptography, дозволяють здійснювати

шифрування, дешифрування та генерацію ключів для різних алгоритмів шифрування, таких як AES, RSA, DES та інші.

– Мова C# має різні фреймворки та бібліотеки для реалізації систем авторизації та аутентифікації. Наприклад, фреймворк ASP.NET Identity надає засоби для управління користувачами, реєстрації, входу, ролей та дозволів.

– Дана мова програмування використовується в контексті фреймворків, таких як ASP.NET або ASP.NET Core, для створення веб-додатків з функціями авторизації та реєстрації. Ці фреймворки надають інструменти для розробки захищених веб-додатків з можливістю налаштування прав доступу, аутентифікації та авторизації користувачів.

– C# має підтримку для різних систем керування базами даних (СКБД). Ви можете використовувати ADO.NET або Entity Framework для взаємодії з реляційними базами даних, такими як MySQL, SQL Server, PostgreSQL та інші. Це дозволяє зберігати та оброблювати дані, пов'язані з авторизацією та реєстрацією [15].

4) PHP є скриптовою мовою програмування, що використовується для розробки веб-додатків. Вона також може бути використана для написання програм з захистом даних, авторизацією та реєстрацією. Ось кілька способів, якими PHP може бути використана для цих цілей:

– PHP має вбудовані функції для роботи з шифруванням. Ви можете використовувати функції, такі як openssl_encrypt() та openssl_decrypt(), для здійснення шифрування та дешифрування даних за допомогою різних криптографічних алгоритмів.

– Ця мова має багато фреймворків та бібліотек, які допомагають в реалізації систем авторизації та аутентифікації. Наприклад, фреймворк Laravel надає вбудовану підтримку для аутентифікації користувачів, включаючи функції реєстрації, входу, керування ролями та правами доступу.

– PHP має різні фреймворки, такі як Laravel, Symfony або CodeIgniter, які дозволяють створювати веб-додатки з функціями авторизації та реєстрації. Ці фреймворки надають готові компоненти для обробки аутентифікації, авторизації та управління користувачами.

– Дана мова програмування має підтримку для різних типів баз даних, включаючи MySQL, PostgreSQL, SQLite та багато інших. Ви можете використовувати розширення PDO (PHP Data Objects) для взаємодії з базами даних, зберігання та обробки даних, пов'язаних з авторизацією та реєстрацією [16].

Для детальнішого порівняння характеристик даних мов програмування розробимо наступну таблицю:

Була обрана саме мова Python тому що вона має декілька ключових переваг, які роблять її кращим рішенням для розробки засобу шифрування даних з реєстрацією і авторизацією. По-перше, Python є простою і зрозумілою мовою програмування, що дозволяє розробникам швидко створювати і тестувати програмний код. Вона має зрозумілий синтаксис і велику кількість документації, що сприяє ефективній роботі над проектом.

По-друге, Python має потужну стандартну бібліотеку, яка включає різноманітні модулі для криптографічних операцій, що полегшує розробку функцій шифрування і розшифрування даних. Крім того, Python має велику кількість сторонніх бібліотек, таких як cryptography, які надають розширені можливості для криптографічного захисту даних.

Таблиця 2.1 — Порівняння мов програмування

	Python	**Java**	**C#**	**PHP**
Тип мови	Інтерпретована	Компільована	Компільована	Інтерпретована
Синтаксис	Простий і зрозумілий	Суворий і строгий	Схожий до Java	Синтаксис PHP
Використання	Універсальна	Універсальна	Розробка Windows	Веб-розробка
Орієнтованість	Об'єктно-орієнтована	Об'єктно-орієнтована	Об'єктно-орієнтована	Об'єктно-орієнтована
Платформа	Крос-платформена	Крос-платформена	Windows	Крос-платформена
Використання в веб.	Широке використання	Популярна у корпораціях	Менш популярна веброзробці	Основна мова веб-розробки
Підтримка шифрування	бібліотеки для шифрування, таких як cryptography, PyCrypto, PyNaCl	Підтримка шифрування через різні алгоритми, такі як AES, DES, RSA	Підтримка шифрування через різні алгоритми, такі як AES, DES, RSA	Має вбудовані функції для шифрування, такі як OpenSSL, Mcrypt

2.1.3 Середовище розробки

Середовище розробки або IDE (скор. англ. Integrated Development Environment — інтегроване середовище розробки) — спеціальний програмний комплекс, призначений для повного циклу написання та тестування програм певною мовою.

Типове середовище розробки містить:

– Текстовий редактор із підсвічуванням синтаксису мови, для якого розроблено середовище.

– Менеджер файлів та/або об’єктів.

– компілятор чи інтерпретатор команд мови програмування.

– Засіб обробки помилок (дебагера).

– Інструменти для складання проекту програми у готову програму [17].

Далі розглянемо декілька найпопулярніших програм для розробки на Python.

PyCharm — це інтегроване середовище розробки (IDE) для програмування на мові Python, розроблене компанією JetBrains. Воно надає широкий спектр функціональності та інструментів, спрямованих на полегшення розробки, налагодження та управління проектами на Python.

Основні переваги PyCharm включають автодоповнення коду, інспектування коду для виявлення помилок та покращення якості коду, підтримку систем контролю версій, вбудований відлагоджувач, інструменти для автоматичного форматування коду, підтримку рефакторингу, можливість роботи з віртуальними середовищами та пакетним менеджером.

PyCharm також надає підтримку для інших технологій, що використовуються в екосистемі Python, таких як Django, Flask, SQLAlchemy, тестування одиниць, наука даних, інтерпретатор IPython та багато іншого. Крім того, PyCharm доступний як Community Edition (безкоштовна версія) і Professional Edition (платна версія з додатковими функціями).

Завдяки своїм потужним можливостям та зручному інтерфейсу, PyCharm є популярним вибором серед розробників Python для ефективної роботи над проектами будь-якого розміру і складності [18].

Visual Studio Code (VS Code) — це безкоштовне та легке використання інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Воно надає розширену підтримку для програмування на різних мовах, включаючи Python.

Основні переваги Visual Studio Code включають розширену підтримку мови Python з автодоповненням коду, інтегрованим відлагоджувачем,

підтримкою систем контролю версій, інструментами для форматування коду, пошуку та виправлення помилок, інтеграцію з пакетним менеджером, вбудований термінал і багато іншого.

VS Code також має широкий вибір розширень, які дозволяють розширити його функціональність і адаптувати до ваших потреб. Наприклад, є розширення для роботи з фреймворками Python, відладки тестів, роботи з базами даних та багато інших.

Один з основних переваг Visual Studio Code полягає в його переносимості — він підтримується на різних платформах, включаючи Windows, macOS та Linux [19].

Jupyter Notebook — це інтерактивне середовище розробки, яке дозволяє створювати та виконувати код у вигляді окремих клітинок, називаних "комірками". Він підтримує багато мов програмування, включаючи Python.

Основна перевага Jupyter Notebook полягає в його здатності поєднувати виконуваний код, текстовий опис та результати виконання коду в одному документі. Кожна комірка може містити код Python, який можна виконати прямо у середовищі, а результати виконання будуть відображені безпосередньо під коміркою. Це робить Jupyter Notebook дуже зручним для експериментування, візуалізації даних, проведення досліджень та демонстрації коду.

Jupyter Notebook також підтримує розширення, що дозволяють додавати додатковий функціонал, такий як підтримка різних мов програмування, інтерактивні візуалізації, інтеграція з системами керування версіями та інше. Крім того, Jupyter Notebook є популярним інструментом для обміну знанням та спільної роботи, оскільки документи можуть бути легко експортовані у різні формати, такі як HTML, PDF або презентації [20].

Anaconda — це дистрибутив Python та платформа для управління пакетами та середовищем для наукових обчислень та аналізу даних. Він включає в себе встановлення Python, багатий набір наукових бібліотек та інструменти для роботи з даними.

Одна з головних переваг Anaconda полягає у його здатності до управління пакетами. Він надає можливість швидко встановлювати, оновлювати та керувати пакетами Python та інших мов програмування, які використовуються в області наукових обчислень. Також Anaconda надає можливість створювати окремі середовища (environments), що дозволяє ізолювати різні проекти та їх залежності один від одного.

Крім того, Anaconda має вбудовані інструменти для роботи з даними, включаючи Jupyter Notebook, що дозволяє проводити аналіз даних, виконувати обчислення та створювати візуалізації прямо у середовищі. Він також містить багато наукових бібліотек, таких як NumPy, pandas, matplotlib, scikit-learn та інші, які роблять Anaconda потужним інструментарієм для розв'язання завдань з обробки даних та наукових обчислень.

Загалом, Anaconda є популярним вибором для розробників та дослідників, які працюють з Python у сфері наукових обчислень та аналізу даних. Він забезпечує зручне управління пакетами, ізоляцію середовищ та має вбудовані інструменти для роботи з даними, що дозволяє зосередитися на розробці та аналізі, замість управління конфігурацією та залежностями [21].

Для роботи було обрано середовище PyCharm, адже воно надає розширений функціонал та спеціалізацію саме на розробку Python-програм. PyCharm пропонує широкий набір інструментів, таких як автодоповнення коду, відлагодження, системи керування версіями та інтегрована підтримка для роботи з віртуальними середовищами. Він також підтримує візуалізацію даних, має можливості для роботи в команді та надає зручність налаштування через плагіни.

Для детальнішого порівняння складемо наступну таблицю:

Таблиця 2.2 — Порівняння середовищ розробки

Функціонал	**PyCharm**	**VS Code**	**Jupyter Notebook**	**Anaconda**
Редагування коду	Так	Так	Частково	Так
Підсвічування синтаксису	Так	Так	Так	Так
Автодоповнення	Так	Так	Так	Так
Відлагодження	Так	Так	Ні	Так
Управління версіями	Так	Так	Ні	Ні
Підтримка віртуальних середовищ	Так	Так	Ні	Так
Інтеграція з Git	Так	Так	Так	Так
Вбудована консоль	Так	Так	Ні	Так
Підтримка розширень	Так	Так	Ні	Ні
Інтерактивна робота з даними	Ні	Ні	Так	Так
Управління пакетами Python	Частково	Частково	Ні	Так
Візуалізація даних	Ні	Частково	Ні	Так

2.1.4 Фреймворки

Фреймворки. Кращі Python фреймворки містять набори інструментів, бібліотек, корисних для розробника модулів. У них реалізовані завдання, які найчастіше зустрічаються і закладена чітка структура коду. Це полегшує роботу: дозволяє зосередитися на логіці проєкту, не витрачати час на повторювані дії, які вже є в асортименті інструментів.

Наприклад, у фреймворки спочатку можуть бути закладені:

– шаблони дизайну з маршрутизацією URL-адрес;
– валідація введення форм;
– міграція БД;
– ORM;
– механізми шаблонізації виведених форм;
– захист від підробки міжсайтових запитів і різних типів ін'єкцій;

– авторизація, аутентифікація, зберігання та вилучення сеансів;

– конфігурація підключення до баз даних.

Важливо, що фреймворки забезпечують незалежність від систем управління базами даних: один і той же код може працювати без внесення змін з різними серверами баз даних [22].

Kivy — це безкоштовний та відкритий фреймворк для розробки мобільних додатків та інших програм з багатоконтактним інтерфейсом користувача (NUI) на мові Python. Він поширюється за ліцензією MIT і може працювати на платформах Android, iOS, Linux, macOS та Windows.

Kivy є основним фреймворком, розробленим організацією Kivy, наряду з Python for Android, Kivy iOS та кількома іншими бібліотеками, які можуть використовуватися на всіх платформах. У 2012 році Kivy отримав грант у розмірі 5000 доларів від Python Software Foundation для портування до Python 3.3. Крім того, Kivy підтримує Raspberry Pi, фінансування якого було забезпечено через Bountysource.

Фреймворк містить всі необхідні елементи для побудови додатку, такі як:

– широкі можливості введення для миші, клавіатури, TUIO та специфічних для ОС подій багатоконтактного введення,

– графічну бібліотеку, яка використовує лише OpenGL ES 2 і базується на об'єктах вершинного буфера і шейдерах,

– широкий набір віджетів, що підтримують багатоконтактний ввід,

– проміжну мову (Kv), яка використовується для легкого проектування власних віджетів [23].

Kivy фреймворк можна використати в програмі для захисту чутливих даних з реєстрацією і авторизацією шляхом створення графічного інтерфейсу користувача (GUI) і реалізації відповідних функцій. Завдяки своїй природній підтримці багатодотикового взаємодії та мультисенсорних подій, Kivy забезпечує зручний спосіб взаємодії з додатком на різних пристроях, включаючи мобільні пристрої.

Використовуючи Kivy, можна створити інтуїтивно зрозумілий інтерфейс для реєстрації нових користувачів, введення та збереження їхніх особистих даних, а також реалізувати механізми авторизації, такі як вхід за допомогою логіна та пароля, аутентифікація через соціальні мережі або використання біометричних даних.

Крім того, Kivy має великий набір вбудованих віджетів та можливостей для створення належного інтерфейсу користувача, включаючи кнопки, текстові поля, списки, вкладки тощо, що дозволяє легко розробляти функціональність реєстрації, авторизації та управління чутливими даними в додатку. Крім того, Kivy має гнучку систему розмітки і можливості стилізації, що дозволяє налаштовувати зовнішній вигляд елементів інтерфейсу відповідно до потреб вашої програми для забезпечення належного рівня безпеки та естетичного зовнішнього вигляду.

Основна частина коду, де використовується фреймворк Kivy, це клас MyApp:

```
class MyApp(App):
    def build(self):
        Window.clearcolor = (0.95, 0.95, 0.95, 1)
        return MyScreenManager()
```

Цей клас є похідним від класу App з Kivy і використовується для визначення функціональності додатку. У методі build створюється екземпляр MyScreenManager і повертається як головний кореневий віджет додатку.

```
class MyScreenManager(ScreenManager):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

        self.login_screen = Screen(name='login')
        self.registration_screen = Screen(name='registration')

        login_form = LoginForm()
        self.login_screen.add_widget(login_form)

        registration_form = RegistrationForm()
        self.registration_screen.add_widget(registration_form)

        self.add_widget(self.login_screen)
        self.add_widget(self.registration_screen)

        self.current = 'login'
```

Клас MyScreenManager також є похідним від класу ScreenManager з Kivy і використовується для управління екранами додатку. У конструкторі створюються екземпляри Screen для екранів входу та реєстрації. Далі створюються екземпляри LoginForm і RegistrationForm, які додаються до відповідних екранів. Екрани додаються до MyScreenManager, і поточний екран встановлюється на екран входу ('login').

2.1.5 База даних

База даних (англ. database) — сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім самих даних, містить їх опис та може містити засоби для їх обробки.

Для розробки засобу захисту чутливих даних була обрана база даних SQLite. Вона є вбудовуваною, легкою у використанні базою даних, яка забезпечує збереження даних у локальних файлових сховищах. Ця база даних має деякі переваги, такі як простота використання, підтримка шифрування та підтримка на різних платформах, таких як Android, iOS, Linux, macOS та Windows.

Особливості SQLite:

– Транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення.

– Встановлення без конфігурації — не потребує ані установки, ані адміністрування.

– Реалізує значну частину стандарту SQL92.

– База даних зберігається в одному крос-платформовому файлі на диску.

– Підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів.

– Малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями.

– Швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій.

– Простий, легкий у використанні API.

– Написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов.

– Добре прокоментований початковий код зі 100 % тестовий покриттям гілок.

– Доступний як єдиний файл початкового коду на ANSI C, який можна легко вставити в інший проєкт.

– Автономність: немає зовнішніх залежностей.

– Крос-платформовість: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE).

– Легко переноситься на інші системи [24].

SQLite дозволяє зберігати конфіденційні дані, такі як користувачі, паролі та інша важлива інформація, у безпечному форматі. Можна використовувати шифрування для збереження конфіденційних даних та виконувати різні операції для забезпечення безпеки даних.

Крім того, SQLite забезпечує швидкий доступ до даних, що є важливим аспектом при роботі з чутливими даними. Можна легко виконувати запити до бази даних, оновлювати інформацію та забезпечувати цілісність даних за допомогою транзакцій.

Створення та обслуговування БД можуть здійснюватись через текстову консоль SQL-командами або через спеціальні інструменти, у тому числі — з графічним інтерфейсом користувача.

У даному проекті засобу для захисту чутливих даних передаються наступні дані в базу даних SQLite:

Для реєстрації нового або авторизації існуючого користувача:

– Логін (username): дані, введені користувачем при реєстрації або авторизації, зберігаються в полі username таблиці users.

– Пароль (password): дані, введені користувачем при реєстрації або авторизації, після шифрування зберігаються в полі password таблиці users (рис. 2.2).

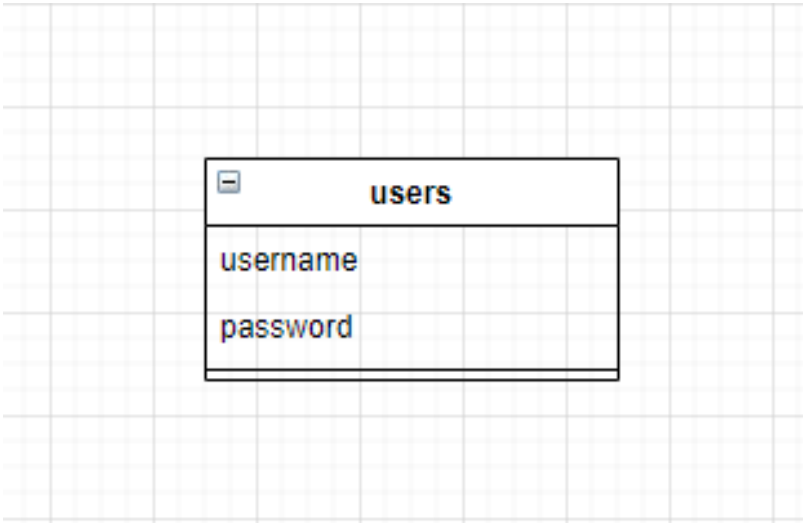


Рисунок 2.2 — Схема бази даних проекту

Далі розглянемо реалізацію через код:

import sqlite3

З'єднання з базою даних SQLite

```
conn = sqlite3.connect('database.db')
c = conn.cursor()
```

Створення таблиці для зберігання користувачів (якщо вона не існує)

```
c.execute('''CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT NOT NULL,
                password TEXT NOT NULL
                )''')
conn.commit()
```

У цьому фрагменті коду спочатку викликається функція connect() модуля sqlite3, передаючи їй назву бази даних 'database.db'. Ця функція створює з'єднання з базою даних або відкриває існуючу базу даних з такою назвою. Далі створюється курсор за допомогою функції cursor(), який використовується для виконання SQL-запитів до бази даних.

Після цього виконується SQL-запит за допомогою методу execute(), який створює таблицю users, якщо вона не існує. Структура таблиці включає поля id, username та password. Поле id є первинним ключем, яке автоматично збільшується при вставці нових записів. Поля username та password визначені як текстові та обов'язкові (NOT NULL).

Команда conn.commit() застосовує зміни до бази даних та зберігає їх.

У цьому коді створюється лише одна таблиця для зберігання користувачів зі специфічною структурою. Якщо потрібно створити додаткові таблиці або розширити структуру існуючих таблиць, можна викликати відповідні команди SQL за допомогою об'єкту c.

Загальну схему роботи програми захисту чутливих даних можна описати наступним чином:

1) Користувач запускає програму і бачить екран авторизації.
2) Користувач вводить свій логін та пароль в відповідні поля.
3) Користувач натискає кнопку "Увійти".
4) Програма перевіряє введені дані:
   – Зчитує користувача з бази даних за введеним логіном.
   – Шифрує введений пароль за допомогою ключа шифрування.
   – Перевіряє, чи співпадає шифрований пароль з паролем, збереженим у базі даних для відповідного користувача.
5) Якщо введені дані є вірними, користувач отримує повідомлення про успішну авторизацію.
6) Якщо введені дані неправильні, користувач отримує повідомлення про помилку авторизації.
7) Користувач може натиснути кнопку "Реєстрація" для переходу до екрану реєстрації.
8) На екрані реєстрації користувач вводить новий логін та пароль в відповідні поля.
9) Користувач натискає кнопку "Зареєструватися".

10) Програма шифрує введений пароль за допомогою ключа шифрування та зберігає нового користувача з введеними даними в базі даних.

11) Користувач отримує повідомлення про успішну реєстрацію.

12) Користувач може повернутися до екрану авторизації, натиснувши кнопку "Назад".

Цей алгоритм забезпечує базову функціональність програми, що включає авторизацію та реєстрацію користувачів з використанням шифрування паролів. Залежно від потреб проекту, цей алгоритм може бути розширений або модифікований для додавання.

Далі розглянемо структурну схему роботи даного програмного забезпечення (рис. 2.3).

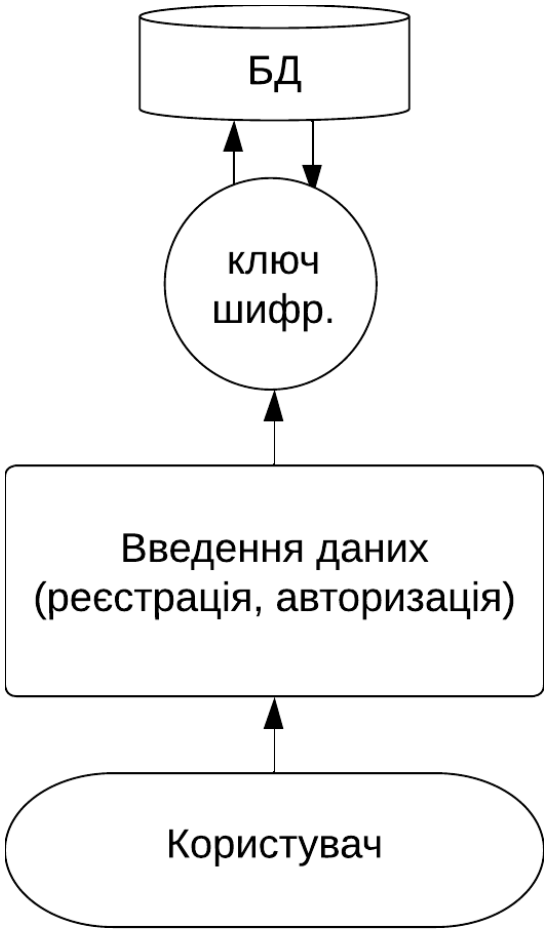


Рисунок 2.3 — Загальна схема роботи програмного продукту

2.2 Розробка системи реєстрації та авторизації

Дана програма для захисту чутливих даних пропонує механізми реєстрації та авторизації користувачів.

2.2.1. Реєстрація

При реєстрації, користувач вводить свій логін та пароль, які зберігаються в базі даних. Пароль шифрується з використанням ключа шифрування та зберігається у зашифрованому вигляді. Таким чином, навіть якщо база даних буде скомпрометована, паролі будуть захищені від несанкціонованого доступу.

Процес реєстрації полягає в наступному:

1) Користувач вводить логін і пароль у відповідні поля.

2) Код шифрує пароль за допомогою модуля cryptography.fernet, використовуючи попередньо визначений ключ шифрування.

3) Зашифрований пароль та логін зберігаються у базі даних.

Основний шматок коду, в якому реалізована функціональність авторизації користувача, це клас LoginForm:

```
class LoginForm(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.orientation = 'vertical'
        self.padding = (50, 100)
        self.spacing = 20
        self.size_hint = (None, None)
        self.size = self.minimum_size
        self.pos_hint = {'center_x': 0.5, 'center_y': 0.5}

        self.username_label = Label(text='Логін', color=(0, 0, 0,
1), size_hint=(1, None), height=40)
        self.username_input = CustomTextInput(multiline=False)

        self.password_label = Label(text='Пароль', color=(0, 0, 0,
1), size_hint=(1, None), height=40)
        self.password_input   =   CustomTextInput(multiline=False,
password=True)

        self.login_button = CustomButton(text='Увійти')
        self.login_button.bind(on_press=self.login)

        self.add_widget(self.username_label)
        self.add_widget(self.username_input)
        self.add_widget(self.password_label)
        self.add_widget(self.password_input)
```

```
            self.add_widget(self.login_button)

        def login(self, instance):
            username = self.username_input.text
            password = self.password_input.text
```

Отримання зашифрованого паролю з бази даних

```
            c.execute("SELECT password FROM users WHERE username=?",
(username,))
            result = c.fetchone()

            if result:
                encrypted_password = result[0]
```

Розшифрування паролю

```
                cipher_suite = Fernet(encryption_key)
                decrypted_password                                    =
cipher_suite.decrypt(encrypted_password).decode('utf-8')

                if password == decrypted_password:
                    popup = Popup(title='Успішна авторизація',
                                  content=Label(text='Ви      успішно
авторизовані!', color=(0, 0, 0, 1)),
                                  size_hint=(None,  None),  size=(400,
200))
                    popup.open()
                else:
                    popup = Popup(title='Помилка авторизації',
                                  content=Label(text='Невірний  логін
або пароль', color=(0, 0, 0, 1)),
                                  size_hint=(None,  None),  size=(400,
200))
                    popup.open()
            else:
                popup = Popup(title='Помилка авторизації',
                              content=Label(text='Невірний  логін  або
пароль', color=(0, 0, 0, 1)),
                              size_hint=(None, None), size=(400, 200))
                popup.open()
```

Клас LoginForm є підкласом BoxLayout з Kivy і використовується для створення форми авторизації користувача. У конструкторі встановлюються розмір, положення та оформлення віджетів форми.

На формі авторизації присутні наступні елементи:

– username_label і password_label — етикетки для полів вводу логіну та паролю відповідно.

– username_input і password_input — поля вводу для логіну та паролю відповідно.

– login_button — кнопка "Увійти", при натисканні на яку виконується процес авторизації.

Метод login викликається при натисканні кнопки "Увійти". Він отримує значення введеного логіну та пароля і перевіряє їх на відповідність збереженим даним в базі даних. За допомогою SQL-запиту отримується зашифрований пароль з бази даних, після чого за допомогою криптографічного алгоритму розшифровується пароль. Порівнюються введений пароль і розшифрований пароль. Якщо вони збігаються, відображається спливаюче вікно з повідомленням про успішну авторизацію. В іншому випадку відображається повідомлення про помилку авторизації.

Цей код використовує глобальну змінну encryption_key, яка представляє ключ шифрування для криптографічного алгоритму. Цей ключ повинен бути збережений в безпечному місці і недоступним для сторонніх осіб, оскільки він використовується для шифрування та розшифрування паролів користувачів.

2.2.2 Авторизація

Процес авторизації полягає в наступному:

1) Користувач вводить логін і пароль у відповідні поля.

2) Код перевіряє ці дані у базі даних. Використовується модуль sqlite3 для взаємодії з базою даних SQLite.

3) Якщо користувач з таким логіном знайдений, виконується розшифрування пароля та порівняння з введеним користувачем.

4) Якщо авторизація успішна, відображається вікно з повідомленням про успішну авторизацію. Використовується клас Popup з бібліотеки Kivy для створення спливаючого вікна.

Основна частина коду, в якому реалізована функціональність реєстрації користувача, це клас RegistrationForm:

```
class RegistrationForm(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.orientation = 'vertical'
        self.padding = (50, 100)
        self.spacing = 20
```

```
            self.size_hint = (None, None)
            self.size = self.minimum_size
            self.pos_hint = {'center_x': 0.5, 'center_y': 0.5}
            self.username_label = Label(text='Логін', color=(0, 0, 0,
1), size_hint=(1, None), height=40)
            self.username_input = CustomTextInput(multiline=False)
            self.password_label = Label(text='Пароль', color=(0, 0, 0,
1), size_hint=(1, None), height=40)
            self.password_input   =   CustomTextInput(multiline=False,
password=True)
            self.back_button = CustomButton(text='Назад')
            self.back_button.bind(on_press=self.go_to_login)

            self.register_button                                    =
CustomButton(text='Зареєструватися')
            self.register_button.bind(on_press=self.register)
            self.add_widget(self.username_label)
            self.add_widget(self.username_input)
            self.add_widget(self.password_label)
            self.add_widget(self.password_input)
            self.add_widget(self.back_button)
            self.add_widget(self.register_button)

        def go_to_login(self, instance):
            app = App.get_running_app()
            app.root.current = 'login'

        def register(self, instance):
            username = self.username_input.text
            password = self.password_input.text
```

Шифрування паролю

```
        cipher_suite = Fernet(encryption_key)

         encrypted_password=cipher_suite.encrypt(password.encode('utf
         -8'))
```

Збереження користувача в базі даних

```
            c.execute("INSERT INTO users (username, password) VALUES (?,
?)", (username, encrypted_password))
            conn.commit()

            popup = Popup(title='Успішна реєстрація',
                          content=Label(text='Ви              успішно
зареєстровані!', color=(0, 0, 0, 1)),
                          size_hint=(None, None), size=(400, 200))
            popup.open()

            app = App.get_running_app()
            app.root.current = 'login'
```

Клас RegistrationForm є підкласом BoxLayout з Kivy і використовується для створення форми реєстрації користувача. У конструкторі встановлюються розмір, положення та оформлення віджетів форми.

На формі реєстрації присутні наступні елементи:

– username_label — мітка для поля вводу логіну користувача.

– username_input — поле вводу логіну користувача.

– password_label — мітка для поля вводу пароля користувача.

– password_input — поле вводу пароля користувача.

– back_button — кнопка "Назад", яка перенаправляє користувача до екрану входу.

– register_button — кнопка "Зареєструватися", при натисканні на яку відбувається реєстрація користувача.

Метод go_to_login викликається при натисканні кнопки "Назад" і змінює поточний екран на екран входу.

Метод register викликається при натисканні кнопки "Зареєструватися". Він отримує значення введеного логіну та пароля, шифрує пароль за допомогою cryptography.fernet і зберігає дані користувача в базі даних SQLite. Після успішної реєстрації відображається спливаюче вікно з повідомленням про успішну реєстрацію, а потім користувач перенаправляється до екрану входу.

У цьому коді використовується також глобальна змінна encryption_key, яка представляє ключ шифрування для криптографічного алгоритму. Цей ключ повинен бути збережений в безпечному місці і недоступним для сторонніх осіб, оскільки він використовується для шифрування та розшифрування паролів користувачів.

При авторизації, користувач вводить свій логін та пароль. Програма зчитує відповідний запис з бази даних та розшифровує збережений пароль. Порівнюється розшифрований пароль з введеним користувачем паролем. Якщо паролі співпадають, користувач успішно авторизується. В протилежному випадку, виводиться повідомлення про помилку авторизації.

Загальна структура розробленого програмного продукту з функцією шифрування даних при реєстрації та авторизації має наступний вигляд (рис. 2.4):

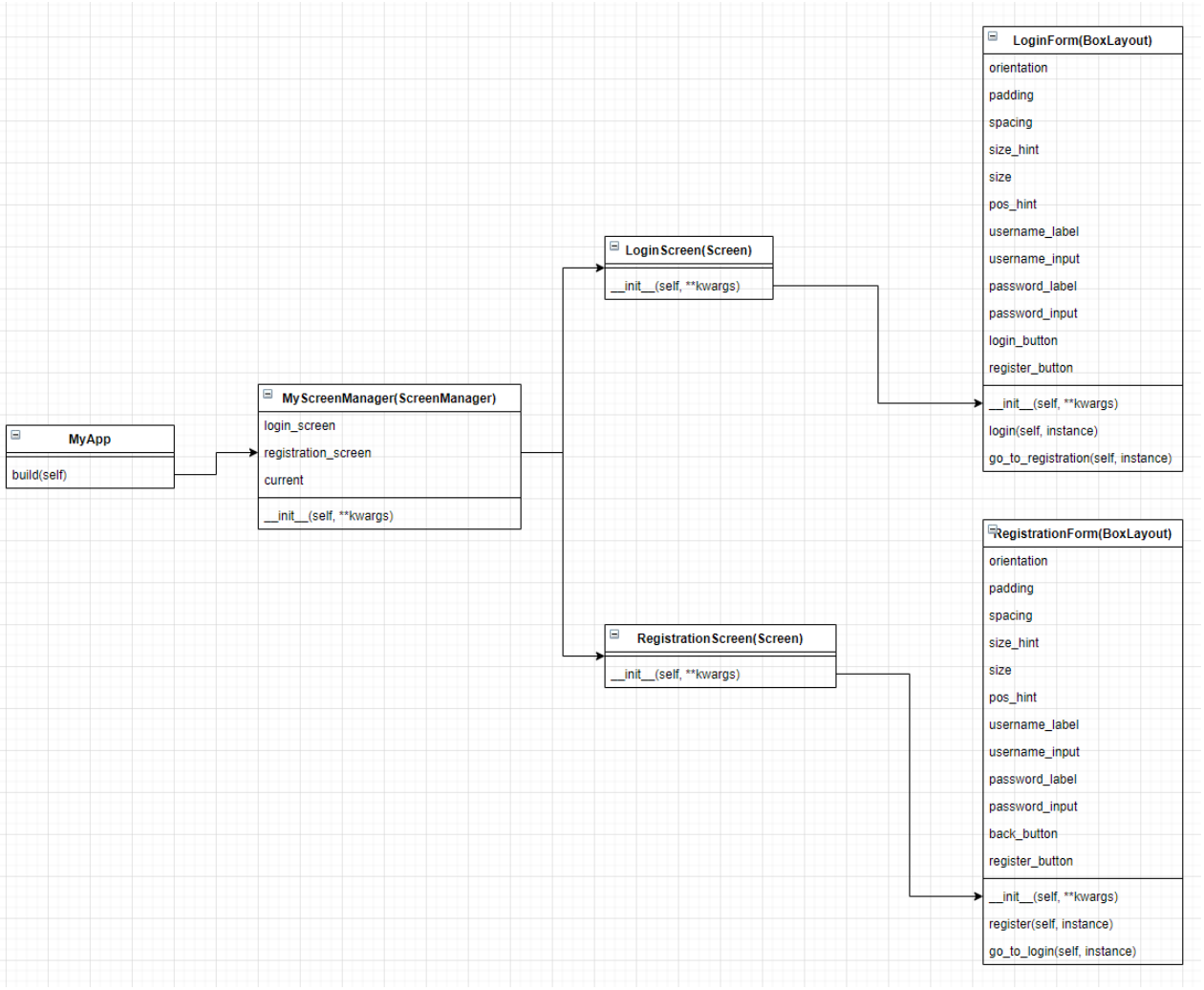


Рисунок 2.4 — Структура розробленого програмного забезпечення

2.3 Впровадження криптографічних методів захисту даних

Криптографічний алгоритм, або шифр — це математична формула, що описує процеси шифрування і розшифрування. Щоб зашифрувати відкритий текст, криптоалгоритм працює в сполученні з ключем - словом, числом або фразою [25].

Впровадження криптографічних методів захисту даних є важливою складовою будь-якої системи або програмного продукту, що працює з чутливою інформацією. Криптографічні методи дозволяють шифрувати дані, щоб забезпечити їх конфіденційність, цілісність та доступність.

Існує багато алгоритмів шифрування, таких як AES, RSA, SHA, і багато інших. Далі розробимо таблицю для їх детального порівняння.

Таблиця 2.3 — Порівняння алгоритмів шифрування

Алгоритм	Тип шифрування	Розмір ключа	Використання
AES	Симетричний	128, 192, 256 біт	Широко використовується для захисту конфіденційності даних, включаючи шифрування файлів, мережевого трафіку тощо.
RSA	Асиметричний	Залежить від реалізації	Використовується для шифрування та цифрового підпису даних, а також для реалізації протоколів обміну ключами та забезпечення конфіденційності та цілісності інформації.
SHA	Хеш-функція	224, 256, 384, 512 біт	Використовується для створення унікального хеш-коду для валідації цілісності даних, створення цифрових підписів та інших криптографічних операцій.

Для програми захисту чутливих даних, зокрема для функцій реєстрації та авторизації, обраний алгоритм AES з вагомих причин. Перш за все, AES (Advanced Encryption Standard) є одним з найбільш надійних і стійких симетричних алгоритмів шифрування, який використовується у багатьох критичних застосунках. Він має доведену стійкість до атак і використовує ключі довжиною 128, 192 або 256 біт, що забезпечує високий рівень конфіденційності і захисту даних.

AES також є широко підтримуваним алгоритмом і має високу поширеність у мовах програмування та криптографічних бібліотеках. Це дає змогу легко інтегрувати AES у програму та забезпечити сумісність з різними платформами [26].

Для ключа довжиною 128 біт алгоритм AES використовує 10 раундів, в кожному з яких послідовно виконуються наступні операції:

– subBytes(): операція замінює кожний байт в стані на відповідний байт з таблиці замін (S-Box). Це забезпечує необхідний рівень нелінійності в алгоритмі і допомагає запобігти простим атакам (рис. 2.5)

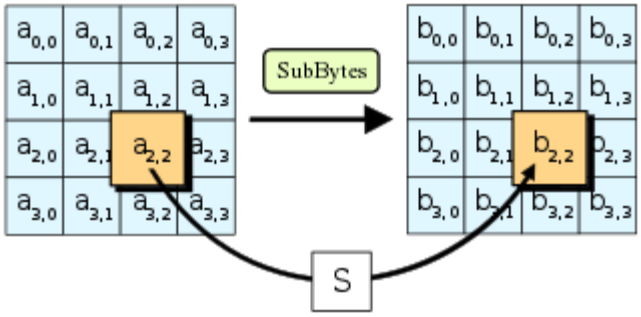


Рисунок 2.5 — Схема операції subBytes() алгоритму AES

- shiftRows(): в цій операції байти в кожному рядку стану циклічно зсуваються вліво. Це допомагає змішати дані між рядками і забезпечити додатковий рівень дифузії (рис. 2.6).

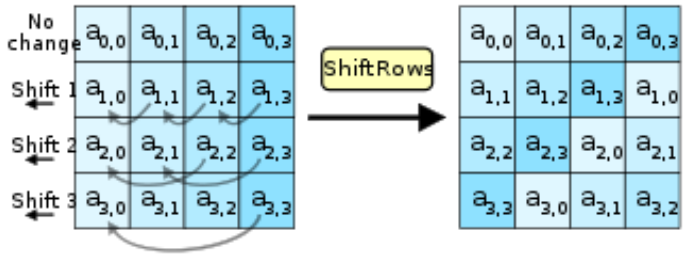


Рисунок 2.6 — Схема операції shiftRows() алгоритму AES

- mixColumns(): у даній операції кожний стовпець стану перемножається на фіксову матрицю, що змішує байти. Це допомагає змішати дані в стані і забезпечити додатковий рівень дифузії. У 10-му раунді ця операція пропускається (рис. 2.7).

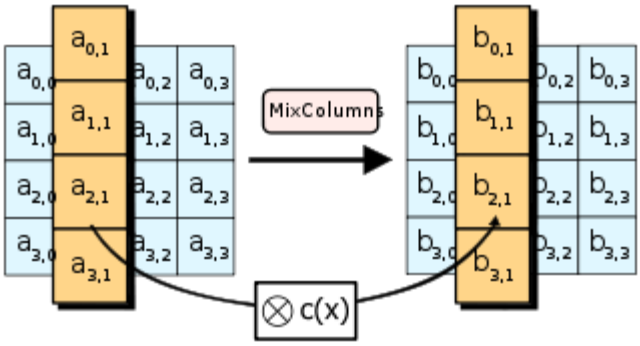


Рисунок 2.7 — Схема операції mixColumns() алгоритму AES

- xorRoundKey(): стан XOR-ується з раундовим ключем, який залежить від початкового ключа і раунду. Це додає додатковий шар ключа для зміцнення безпеки і забезпечує унікальність шифротекстів (рис. 2.8) [26].

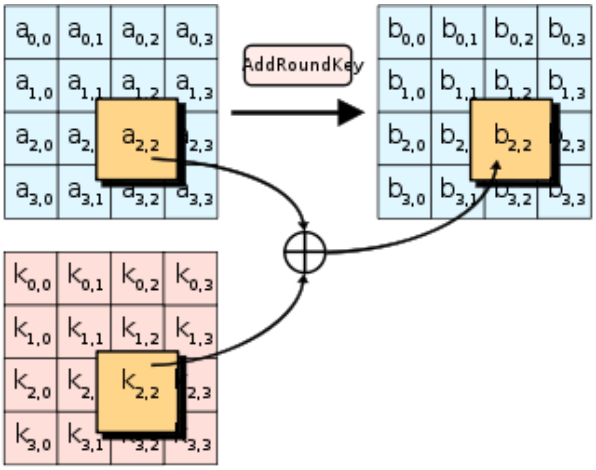


Рисунок 2.8 — Схема операції xorRoundKey() алгоритму AES

Далі опишемо схему криптографічного методу захисту паролів у програмному продукті для захисту даних:

1) Генерація ключа шифрування:

– Використання криптографічно стійкої функції генерації випадкових чисел для отримання ключа шифрування.

– Збереження ключа шифрування в безпечному місці. Не розголошення його і не зберігання разом з іншими конфіденційними даними.

2) Шифрування пароля перед збереженням:

– Використання криптографічного алгоритму AES (Advanced Encryption Standard) для шифрування пароля.

– Шифрування паролю з використанням згенерованого ключа шифрування.

– Збереження зашифрованого пароля в базі даних.

3) Розшифрування пароля під час перевірки авторизації:

– При отриманні вхідного пароля під час перевірки авторизації необхідно розшифрувати зашифрований пароль з використанням ключа шифрування.

– Порівняння отриманого розшифрованого пароля зі вхідним паролем, який ввів користувач.

– Якщо паролі збігаються, авторизація вважається успішною.

Зрештою розглянемо розробку даного методу шифрування в коді програми. Для його реалізації використовує бібліотеку cryptography для шифрування та розшифрування паролів.

1) Генерація ключа шифрування:

```
encryption_key = Fernet.generate_key()
```

2) Шифрування даних:

```
cipher_suite = Fernet(encryption_key)
encrypted_data = cipher_suite.encrypt(data)
```

У коді пароль шифрується перед збереженням в базі даних. Використовується об'єкт Fernet, який створюється з ключем шифрування. Потім за допомогою методу encrypt() шифрується вхідний рядок data.

3) Розшифрування даних:

```
cipher_suite = Fernet(encryption_key)
decrypted_data = cipher_suite.decrypt(data)
```

Тут розшифровується збережений в базі даних зашифрований пароль. Знову створюється об'єкт Fernet з ключем шифрування, а потім за допомогою методу decrypt() розшифровується вхідний зашифрований рядок data.

В даному розділі було проведено комплексні заходи для розробки засобу захисту чутливих даних з функціями реєстрації та авторизації.

Під час вибору архітектури та технологій для реалізації програмного продукту було обрано Python як мову програмування, Kivy як фреймворк для інтерфейсу користувача, SQLite як базу даних та PyCharm як середовище розробки. Ці технології відповідають вимогам проекту і забезпечують ефективну реалізацію засобу захисту.

Далі була розроблена система аутентифікації та авторизації, що дозволяє контролювати доступ до системи і гарантує ідентифікацію користувачів та керування їхніми правами доступу. Ця система забезпечує безпеку обробки даних і запобігає несанкціонованому доступу до чутливої інформації.

Важливим кроком було впровадження криптографічних методів захисту даних, зокрема використання алгоритму AES для шифрування. AES забезпечує високий рівень криптографічної стійкості і ефективну обробку даних, що забезпечує надійний захист чутливої інформації в системі.

В результаті проведених заходів з розробки засобу захисту чутливих даних було досягнуто важливої мети — забезпечення безпеки і конфіденційності інформації. Розроблені архітектура, система аутентифікації та авторизації, а також впроваджені криптографічні методи забезпечують надійний рівень захисту чутливих даних в системі.

3 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис інтерфейсу та функціоналу програмного засобу

3.1.1 Функціонал програмного забезпечення

Програма захисту чутливих даних має на меті забезпечити безпеку та конфіденційність важливої інформації шляхом реалізації функціоналу, спрямованого на захист введених даних.

Основний функціонал розробленої програми включає наступне:

1) Авторизація користувача:
   – Користувач вводить логін та пароль.
   – Введені дані перевіряються з базою даних.
   – Якщо введені дані вірні, відображається повідомлення про успішну авторизацію.
   – Якщо введені дані невірні, відображається повідомлення про помилку авторизації.
2) Реєстрація користувача:
   – Користувач вводить логін та пароль для реєстрації нового облікового запису.
   – Введені дані зберігаються в базі даних.
   – Після успішної реєстрації відображається повідомлення про успішну реєстрацію.
3) Збереження даних:
   – Для збереження користувачів використовується база даних SQLite.
   – При реєстрації пароль користувача шифрується за допомогою бібліотеки cryptography.fernet.
   – При авторизації введений пароль розшифровується та порівнюється зі збереженим в базі даних.

3.1.2 Інтерфейс програмного забезпечення

Засіб захисту даних пропонує інтуїтивно зрозумілий та зручний інтерфейс, що дозволяє користувачам легко орієнтуватися та використовувати всі функціональні можливості програми для ефективного захисту їх чутливої інформації. Далі розглянемо детальніше інтерфейс розробленої програми.

1) Екран авторизації (рис. 3.1):

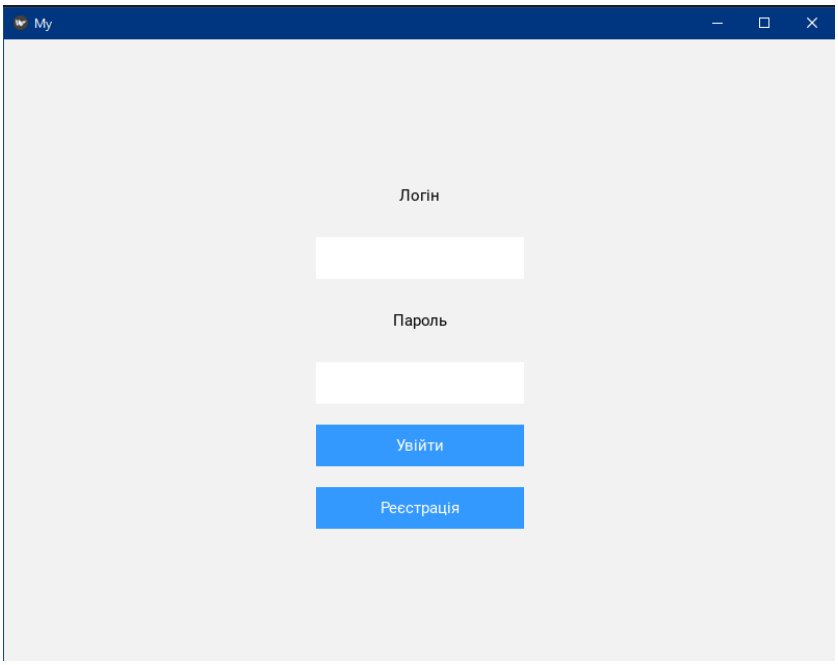


Рисунок 3.1 — Скріншот екрану авторизації

– Зверху розміщений лейбл з написом "Логін".

– Під лейблом розміщене текстове поле, де користувач може ввести свій логін.

– Після логіну розташований другий лейбл з написом "Пароль".

– Під ним розміщене текстове поле для введення пароля, де символи приховані.

– Далі йде кнопка "Увійти", яку користувач натискає для авторизації.

– Під кнопкою "Увійти" знаходиться кнопка "Реєстрація", що веде до екрану реєстрації.

2) Екран реєстрації (рис. 3.2):

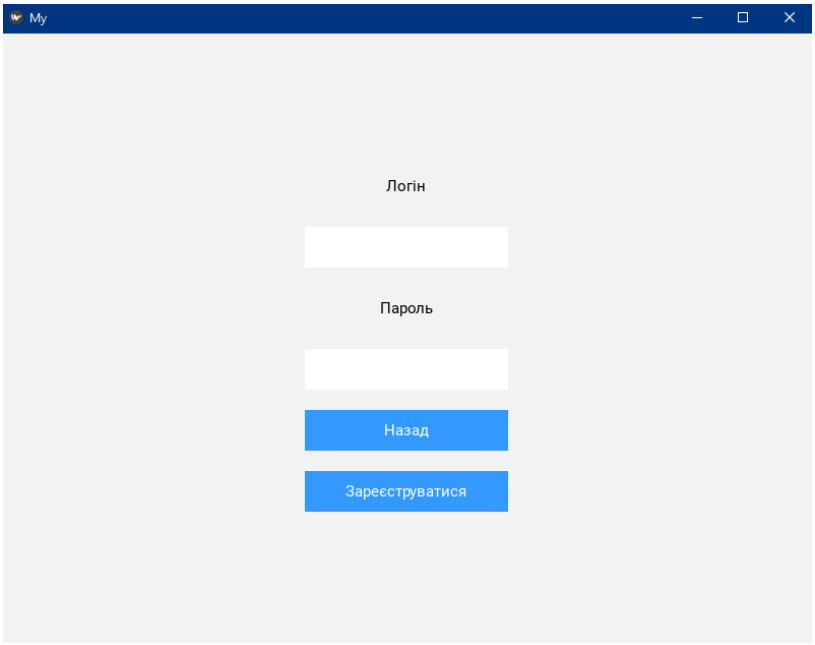


Рисунок 3.2 — Скріншот екрану реєстрації

– Починається з лейбла "Логін" та текстового поля для введення нового логіна.

– Після цього йде лейбл "Пароль" та текстове поле для введення нового пароля.

– Далі є кнопка "Назад", яку користувач натискає для повернення до екрану авторизації.

– Після неї знаходиться кнопка "Зареєструватися", яку користувач натискає для завершення процесу реєстрації.

3) Повідомлення:

– Після спроби авторизації або реєстрації вище описані екрани відображають спливаючі вікна-повідомлення.

– Успішна авторизація відображає повідомлення з заголовком "Успішна авторизація" та текстом "Ви успішно авторизовані!" (рис. 3.3).

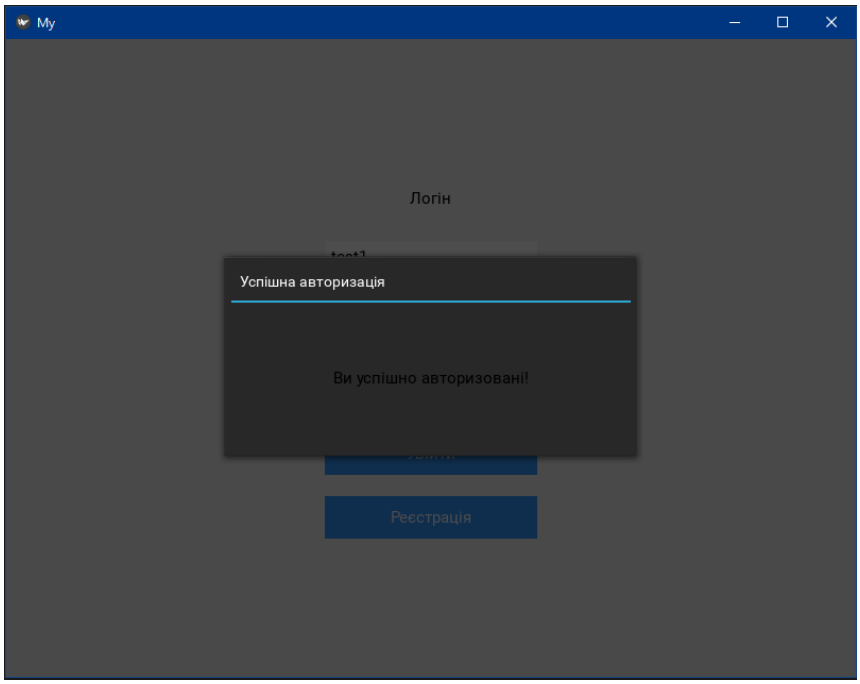


Рисунок 3.3 — Скріншот успішної авторизації

– Невірні дані для авторизації призводять до повідомлення з заголовком "Помилка авторизації" та текстом "Неправильний логін або пароль!".

– Успішна реєстрація відображає повідомлення з заголовком "Успішна реєстрація" та текстом "Ви успішно зареєстровані!" (рис. 3.4).

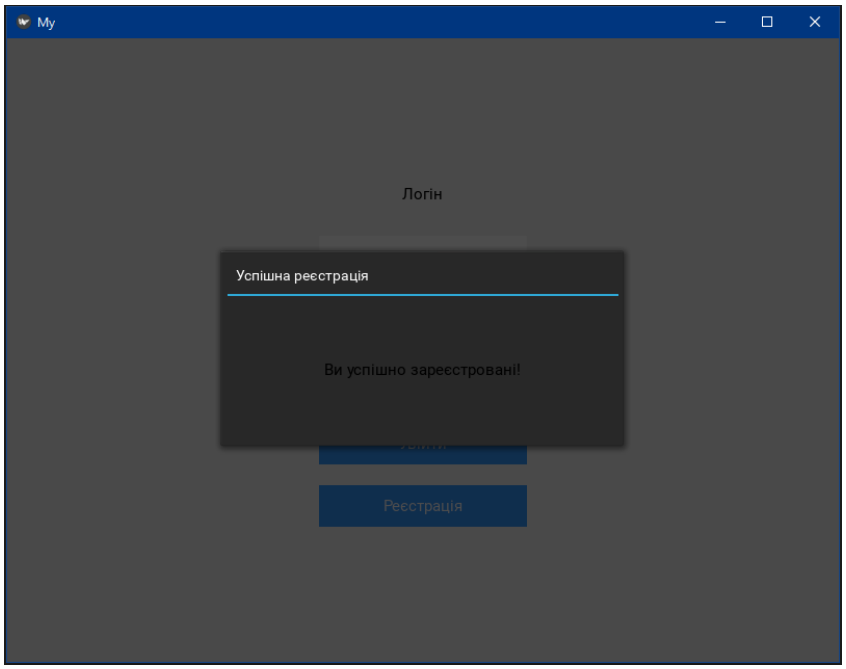


Рисунок 3.3 — Скріншот успішної реєстрації

3.2 Методика тестування та оцінка ефективності та безпеки розробленої програми

Тестування програми проводиться з метою перевірки її функціональності, якості і надійності перед її випуском. Це допомагає виявити та виправити помилки, а також забезпечити відповідність програми поставленим вимогам і очікуванням. Тестування дозволяє переконатися, що програма працює належним чином, і допомагає забезпечити якість і надійність ПП.

У відповідності з індивідуальним завданням були створені сценарії тестування, для кожного з яких було в кінці виконано виклик і перевірку отриманого результату, реалізовуючи ці дії щонайменше раз на кожну функцію. При перевірці співставлявся очікуваний результат з отриманим. Усі результати в кінцевому варіанті співпали. Тому тестування в підсумку можна вважати успішним, а значить і реалізація ПП в підсумку виявилась успішною.

За результатами була розроблена таблиця тестування програми (табл. 3.1).

У цьому розділі було представлено детальний опис інтерфейсу та функціоналу розробленого програмного засобу. Виявлено, що програмний засіб захисту чутливих даних надає зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Функціонал програми дозволяє ефективно захищати чутливі дані під час авторизації та реєстрації.

УСП — Успішно

НСП — Не успішно

Таблиця 3.1 — Таблиця тестування програми

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Авторизація з правильними даними	Логін: "user1", Пароль: "password123"	Відображення повідомлення про успішну авторизацію	Повідомлення про успішну авторизацію відображається	УСП
2	Авторизація з неправильним логіном	Логін: "incorrect_user", Пароль: "password123"	Відображення повідомлення про помилку авторизації	Повідомлення про помилку авторизації відображається	УСП
3	Авторизація з неправильним паролем	Логін: "user1", Пароль: "incorrect_password"	Відображення повідомлення про помилку авторизації	Повідомлення про помилку авторизації відображається	УСП
4	Реєстрація нового користувача	Логін: "new_user", Пароль: "new_password"	Відображення повідомлення про успішну реєстрацію	Повідомлення про успішну реєстрацію відображається	УСП
5	Реєстрація користувача з вже існуючим логіном	Логін: "user1", Пароль: "password123"	Відображення повідомлення про помилку реєстрації	Повідомлення про помилку реєстрації відображається	УСП
6	Авторизація без введення даних	Логін: "", Пароль: ""	Відображення повідомлення про помилку авторизації	Повідомлення про помилку авторизації відображається	УСП
7	Реєстрація з введенням пустого логіна	Логін: "", Пароль: "password123"	Відображення повідомлення про помилку реєстрації	Повідомлення про помилку реєстрації відображається	УСП
10	Завершення програми	Натиснення на кнопку "Вихід"	Закриття програми	Програма закривається	УСП

В ході розробки програмного засобу для захисту чутливих даних було проведене тестування, спрямоване на перевірку функціональності, якості та безпеки програми. Під час тестування було визначено, що програмний засіб відповідає поставленим вимогам і очікуванням, забезпечує ефективний захист чутливих даних та виявляється надійним у використанні.

ВИСНОВКИ

У ході даної бакалаврської дипломної роботи була створена програма для авторизації і реєстрації користувачів. Цей програмний продукт має зовнішній вигляд, що складається з екрану авторизації, екрану реєстрації та повідомлень. Він дозволяє користувачам зручно і безпечно авторизуватися та реєструватися в системі.

Для реалізації програми були використані Python як мова програмування, Kivy як фреймворк для інтерфейсу користувача, SQLite як база даних та PyCharm як середовище розробки. Ці технології відповідають вимогам проекту і забезпечують ефективну реалізацію засобу захисту.

Розроблена система аутентифікації та авторизації контролює доступ до системи і гарантує ідентифікацію користувачів та керування їхніми правами доступу. Вона забезпечує безпеку обробки даних і запобігає несанкціонованому доступу до чутливої інформації. Також було використано алгоритм AES для шифрування даних, що забезпечує високий рівень криптографічної стійкості і ефективну обробку інформації. Цей криптографічний метод гарантує надійний захист чутливих даних в системі.

Під час тестування було визначено, що програмний засіб відповідає поставленим вимогам і очікуванням, забезпечує ефективний захист чутливих даних та виявляється надійним у використанні.

Результати розробки засобу захисту чутливих даних підтверджують досягнення важливої мети — забезпечення безпеки і конфіденційності інформації. Розроблені архітектура, система аутентифікації та авторизації, а також впроваджені криптографічні методи забезпечують надійний рівень захисту чутливих даних в системі.

У цілому, розроблений засіб захисту чутливих даних є ефективним інструментом для забезпечення безпеки і конфіденційності інформації. Його можна використовувати як основу для подальшого розширення і вдосконалення в галузі захисту даних.

Оцінка результатів і відповідність сучасному рівню наукових і технічних знань: результати роботи підтверджують високий рівень наукових і технічних знань, оскільки було розроблено ефективну систему захисту чутливих даних з використанням сучасних технологій. Програма авторизації і реєстрації користувачів, розроблена в рамках проекту, відповідає сучасним стандартам і вимогам безпеки даних.

Створення нової програми і розроблення методики роботи: в результаті роботи була створена програма для авторизації і реєстрації користувачів, яка може бути використана в різних галузях і сферах діяльності. Розроблена методика роботи програмних засобів забезпечує зручну та безпечну обробку даних.

Значущість роботи: розроблений засіб захисту чутливих даних має велику наукову, науково-технічну та соціально-економічну значущість, оскільки забезпечує безпеку і конфіденційність інформації для різних суб'єктів, включаючи компанії, організації та користувачів.

Практичні рекомендації: рекомендується використання розробленого засобу захисту чутливих даних в організаціях, де вимагається надійний рівень безпеки і конфіденційності інформації. Також рекомендується подальше вдосконалення і розширення розробленої системи захисту для врахування специфічних потреб та вимог різних галузей і сфер діяльності.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Захист персональних даних: правове регулювання та практичні аспекти науково-практичний посібник // Маркіян Бем, Іван Городиський, 2021

2. ТОП-10 питань у сфері захисту персональних даних. Експертний Центр з Прав Людини [Електронний ресурс] URL: https://ecpl.com.ua/news/top-10-pytan-u-sferi-zakhystu-personal-nykh-danykh/

3. What is Sensitive Data? Definition, Examples, and More. StrongDM [Електронний ресурс] URL: https://www.strongdm.com/blog/sensitive-data

4. Згода на обробку персональних даних: у яких випадках вона не вимагається. Юридична Газета [Електронний ресурс] URL: https://yur-gazeta.com/dumka-eksperta/zgoda-na-obrobku-personalnih-danih-u-yakih-vipadkah-vona-ne-vimagaetsya.html

5. Загальний регламент про захист даних (GDPR). GDPR-Text.com. Газета [Електронний ресурс] URL: https://gdpr-text.com/uk/

6. Про захист персональних даних : Закон України від 01.06.2010 р. № 2297-VI : станом на 27 жовт. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text

7. Захист персональних даних — WikiLegalAid [Електронний ресурс] URL:https://wiki.legalaid.gov.ua/index.php/Захист_персональних_даних

8. Unauthorized Access: 5 Best Practices to Avoid Data Breaches. Cynet [Електронний ресурс] URL: https://www.cynet.com/network-attacks/unauthorized-access-5-best-practices-to-avoid-the-next-data-breach/

9. Identity theft – Wikipedia [Електронний ресурс] URL: https://en.wikipedia.org/wiki/Identity_theft

10. Що таке кібератака? Microsoft [Електронний ресурс] URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-a-cyberattack

11. Шифрування: типи і алгоритми. Hostpro Wiki. [Електронний ресурс] URL: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms

12. Архітектура програмного забезпечення. [Електронний ресурс] URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya

13. Python [Електронний ресурс] URL: https://www.python.org/

14. What is Java? [Електронний ресурс] URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-programming-language/

15. Документация по C# Python [Електронний ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/

16. PHP Manual [Електронний ресурс] URL: https://www.php.net/manual/en/index.php

17. Що таке середовище розробки і навіщо воно потрібне [Електронний ресурс] URL: https://blogchain.com.ua/shcho-take-seredovyshche-rozrobky-i-navishcho-vono-potribne/

18. Pycharm [Електронний ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/

19. Visual Studio Code [Електронний ресурс] URL: https://code.visualstudio.com/

20. JupyterNotebook [Електронний ресурс] URL: https://jupyter.org/

21. Anaconda [Електронний ресурс] URL: https://docs.anaconda.com/

22. Що таке фреймворки Python? [Електронний ресурс] URL: https://dev.ua/news/top-7-freimvorkiv-python-dlia-rozrobky-veb-dodatkiv

23. Kivy (framework) [Електронний ресурс] URL: https://en.wikipedia.org/wiki/Kivy_%28framework%29

24. SQLite. Вікіпедія. [Електронний ресурс] URL:

25. https://uk.wikipedia.org/wiki/SQLite

26. Класифікація криптоалгоритмів. Wiki ТНТУ. [Електронний ресурс] URL: https://wiki.tntu.edu.ua/Класифікація_криптоалгоритмів

27. Advanced Encryption Standard. Вікіпедія [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard

ДОДАТКИ

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ
БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб захисту чутливих даних

Автор роботи: Попова Ірина Сергіївна

Тип роботи: бакалаврська дипломна робота .

Підрозділ кафедра захисту інформації ФІТКІ .

Показники звіту подібності Unicheck

Оригінальність – 89,0%. Схожість – 11,0%.

Аналіз звіту подібності (відмітити потрібне):

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

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

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

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

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

Автор роботи _____ Попова І. С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Лукічов В. В.
(підпис) (прізвище, ініціали)

Додаток Б
КОД ПРОГРАМИ

Main.py

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.popup import Popup
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.core.window import Window
from kivy.lang import Builder
from cryptography.fernet import Fernet
import sqlite3


# Змінна, що містить ключ шифрування (зберігайте його в безпечному
місці!)
encryption_key = b'PfGgDS3PHvzlDbT7PTXBa9KZzy2oO39m6ubbH79j-qs='

# З'єднання з базою даних SQLite
conn = sqlite3.connect('database.db')
c = conn.cursor()

# Створення таблиці для зберігання користувачів (якщо вона не існує)
c.execute('''CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT NOT NULL,
                password TEXT NOT NULL
                )''')
conn.commit()

Builder.load_string('''
<CustomTextInput>:
    background_normal: ''
    background_color: 1, 1, 1, 1
    foreground_color: 0, 0, 0, 1
    size_hint: (None, None)
    size: 200, 40

<CustomButton>:
    background_normal: ''
    background_color: 0.2, 0.6, 1, 1
    color: 1, 1, 1, 1
    size_hint: (None, None)
    size: 200, 40

<LoginForm>:
    orientation: 'vertical'
    padding: 50
```

```
    spacing: 20
    size_hint: (None, None)
    size: self.minimum_size

<RegistrationForm>:
    orientation: 'vertical'
    padding: 50
    spacing: 20
    size_hint: (None, None)
    size: self.minimum_size
''')


class CustomTextInput(TextInput):
    pass


class CustomButton(Button):
    pass


class LoginForm(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.orientation = 'vertical'
        self.padding = (50, 100)
        self.spacing = 20
        self.size_hint = (None, None)
        self.size = self.minimum_size
        self.pos_hint = {'center_x': 0.5, 'center_y': 0.5}

        self.username_label = Label(text='Логін', color=(0, 0, 0, 1),
size_hint=(1, None), height=40)
        self.username_input = CustomTextInput(multiline=False)

        self.password_label = Label(text='Пароль', color=(0, 0, 0, 1),
size_hint=(1, None), height=40)
        self.password_input = CustomTextInput(multiline=False,
password=True)

        self.login_button = CustomButton(text='Увійти')
        self.login_button.bind(on_press=self.login)

        self.register_button = CustomButton(text='Реєстрація')
        self.register_button.bind(on_press=self.go_to_registration)

        self.add_widget(self.username_label)
        self.add_widget(self.username_input)
        self.add_widget(self.password_label)
        self.add_widget(self.password_input)
        self.add_widget(self.login_button)
```

```
        self.add_widget(self.register_button)

    def login(self, instance):
        username = self.username_input.text
        password = self.password_input.text

        # Зчитування користувача з бази даних
        c.execute("SELECT * FROM users WHERE username=?", (username,))
        user = c.fetchone()

        if user:
            # Розшифрування та перевірка паролю
            cipher_suite = Fernet(encryption_key)
            decrypted_password = cipher_suite.decrypt(user[2])
            decrypted_password = decrypted_password.decode('utf-8')

            if password == decrypted_password:
                popup = Popup(title='Успішна авторизація',
                              content=Label(text='Ви успішно
авторизовані!', color=(0, 0, 0, 1)),
                              size_hint=(None, None), size=(400, 200))
                popup.open()
                return

        popup = Popup(title='Помилка авторизації',
                      content=Label(text='Неправильний логін або
пароль!', color=(0, 0, 0, 1)),
                      size_hint=(None, None), size=(400, 200))
        popup.open()

    def go_to_registration(self, instance):
        app = App.get_running_app()
        app.root.current = 'registration'


class RegistrationForm(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.orientation = 'vertical'
        self.padding = (50, 100)
        self.spacing = 20
        self.size_hint = (None, None)
        self.size = self.minimum_size
        self.pos_hint = {'center_x': 0.5, 'center_y': 0.5}

        self.username_label = Label(text='Логін', color=(0, 0, 0, 1),
size_hint=(1, None), height=40)
        self.username_input = CustomTextInput(multiline=False)
```

```
        self.password_label = Label(text='Пароль', color=(0, 0, 0, 1),
size_hint=(1, None), height=40)
        self.password_input = CustomTextInput(multiline=False,
password=True)

        self.back_button = CustomButton(text='Назад')
        self.back_button.bind(on_press=self.go_to_login)

        self.register_button = CustomButton(text='Зареєструватися')
        self.register_button.bind(on_press=self.register)

        self.add_widget(self.username_label)
        self.add_widget(self.username_input)
        self.add_widget(self.password_label)
        self.add_widget(self.password_input)
        self.add_widget(self.back_button)
        self.add_widget(self.register_button)

    def go_to_login(self, instance):
        app = App.get_running_app()
        app.root.current = 'login'

    def register(self, instance):
        username = self.username_input.text
        password = self.password_input.text

        # Шифрування паролю
        cipher_suite = Fernet(encryption_key)
        encrypted_password =
cipher_suite.encrypt(password.encode('utf-8'))

        # Збереження користувача в базі даних
        c.execute("INSERT INTO users (username, password) VALUES (?,
?)", (username, encrypted_password))
        conn.commit()

        popup = Popup(title='Успішна реєстрація',
                      content=Label(text='Ви успішно зареєстровані!',
color=(0, 0, 0, 1)),
                      size_hint=(None, None), size=(400, 200))
        popup.open()

        app = App.get_running_app()
        app.root.current = 'login'


class MyScreenManager(ScreenManager):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
```

```
        self.login_screen = Screen(name='login')
        self.registration_screen = Screen(name='registration')

        login_form = LoginForm()
        self.login_screen.add_widget(login_form)

        registration_form = RegistrationForm()
        self.registration_screen.add_widget(registration_form)

        self.add_widget(self.login_screen)
        self.add_widget(self.registration_screen)

        self.current = 'login'


class MyScreenManager(ScreenManager):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

        self.login_screen = LoginScreen(name='login')
        self.registration_screen =
RegistrationScreen(name='registration')

        self.add_widget(self.login_screen)
        self.add_widget(self.registration_screen)

        self.current = 'login'

class LoginScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.add_widget(LoginForm())


class RegistrationScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.add_widget(RegistrationForm())


class MyApp(App):
    def build(self):
        Window.clearcolor = (0.95, 0.95, 0.95, 1)
        return MyScreenManager()


if __name__ == '__main__':
    MyApp().run ()
```

ІЛЮСТРАТИВНА ЧАСТИНА

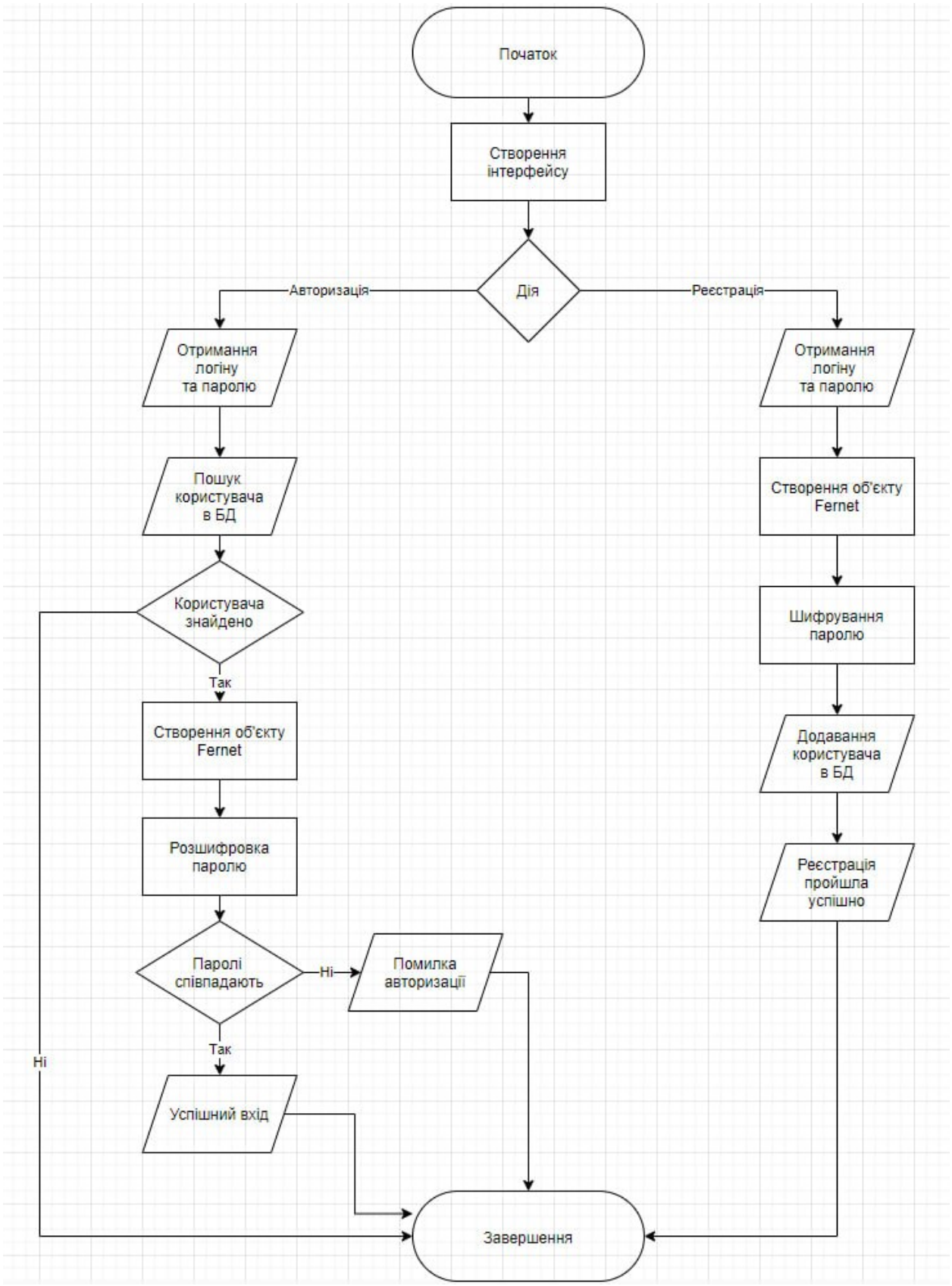
ЗАСІБ ЗАХИСТУ ЧУТЛИВИХ ДАНИХ

(Назва бакалаврської кваліфікаційної роботи)

Виконала: студентка 4 курсу групи 1БС-19 б
спеціальності 125 Кібербезпека
_____ Попова І. С.
19 червня 2023 р.

Керівник: к. т. н., доцент каф. ЗІ
_____ Лукічов В. В.
19 червня 2023 р.

Блок-схема програмного засобу



UML діаграма варіантів використання програмного засобу

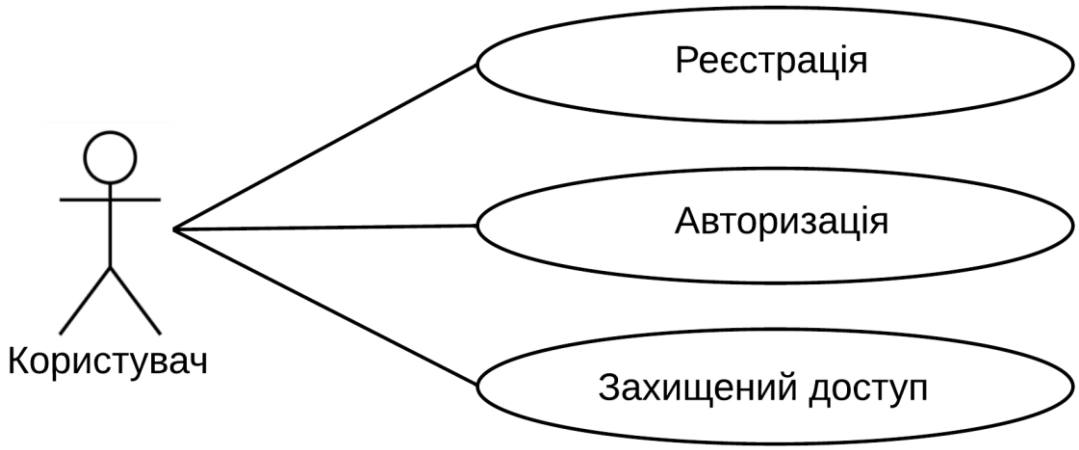
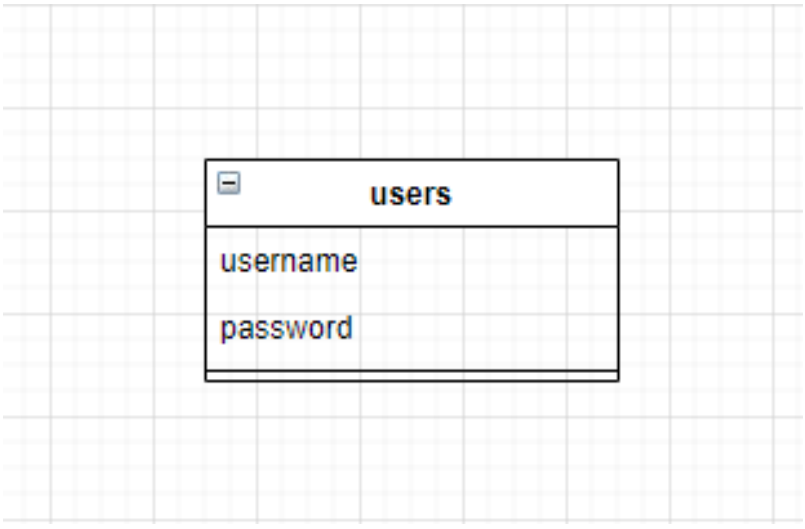
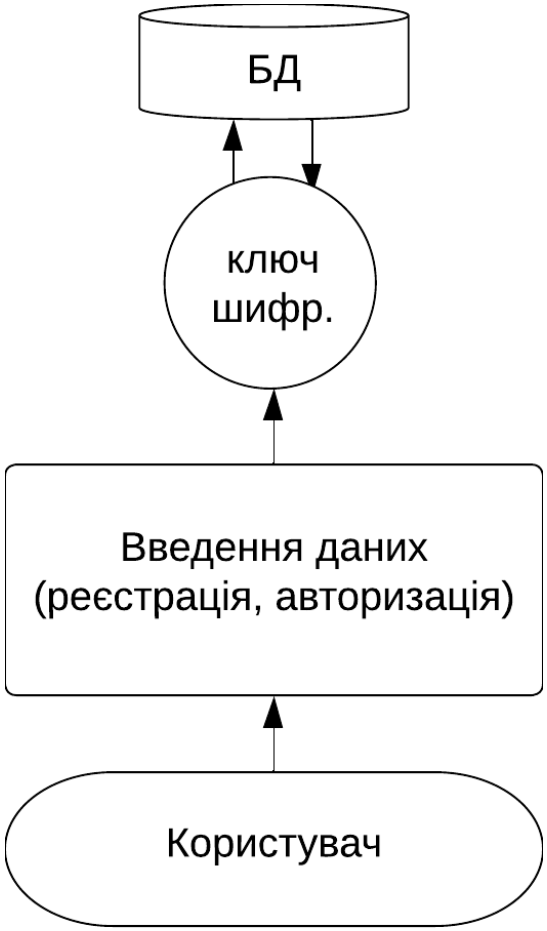


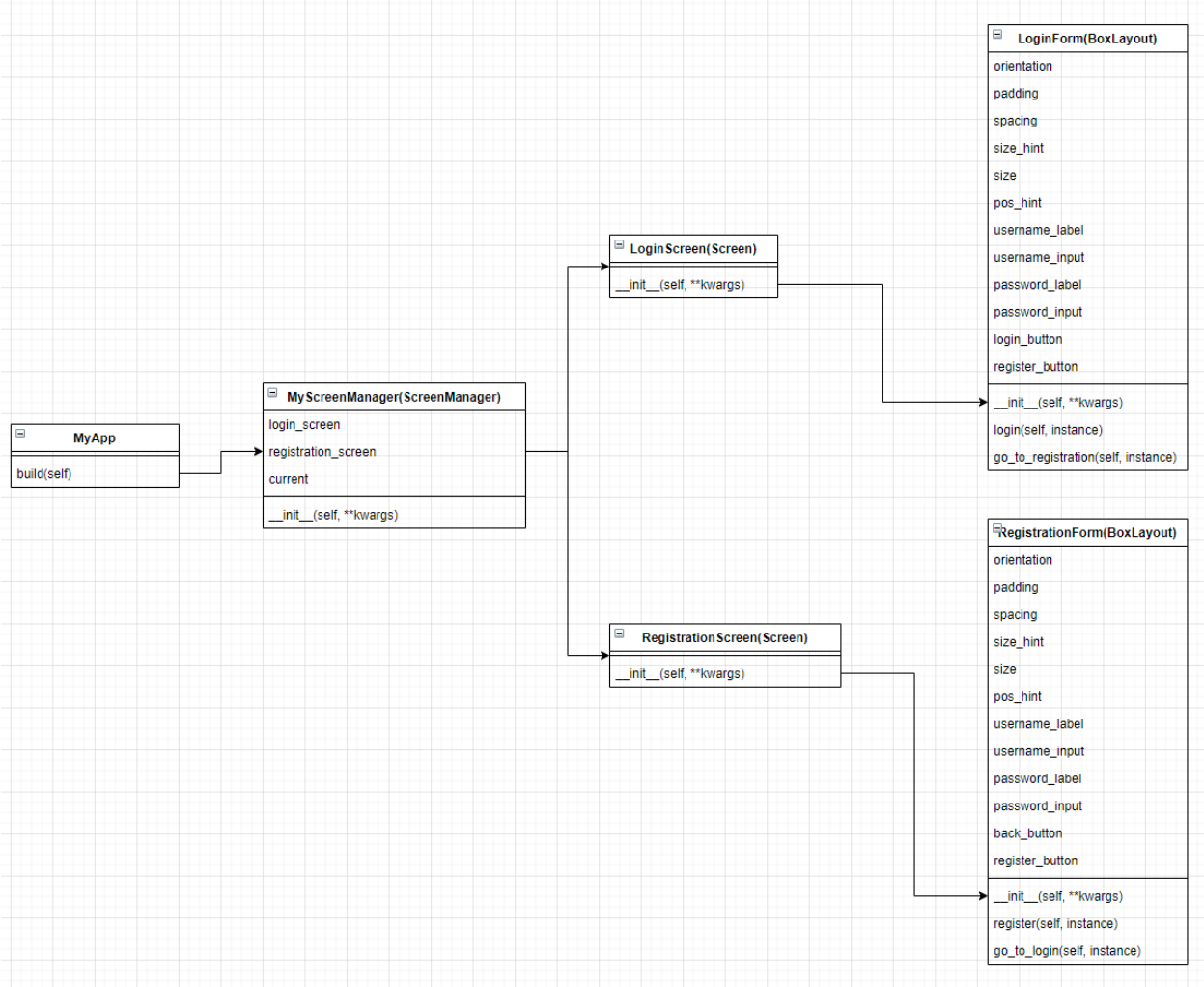
Схема бази даних застосунку



Загальна схема роботи програмного засобу



Структура розробленого програмного забезпечення



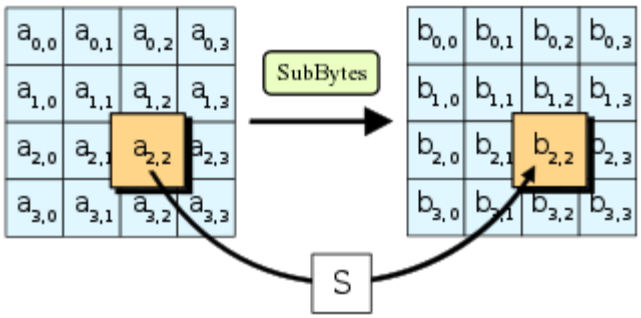


Схема операції subBytes() алгоритму AES

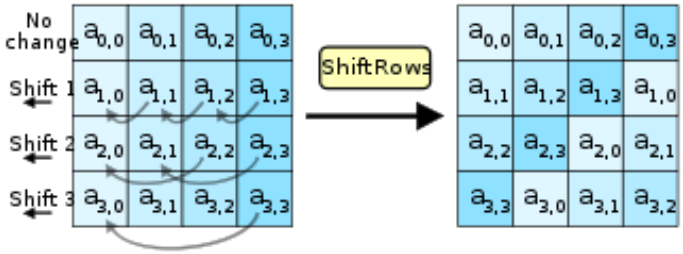


Схема операції shiftRows() алгоритму AES

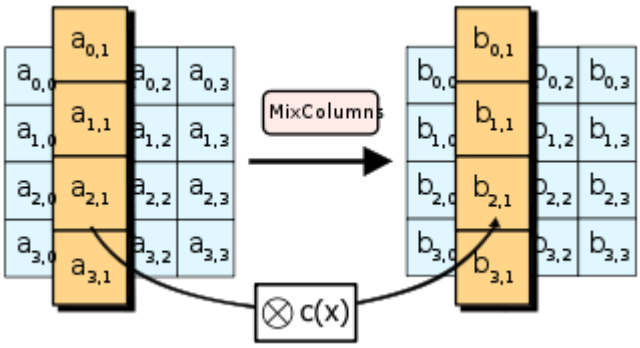


Схема операції mixColumns() алгоритму AES

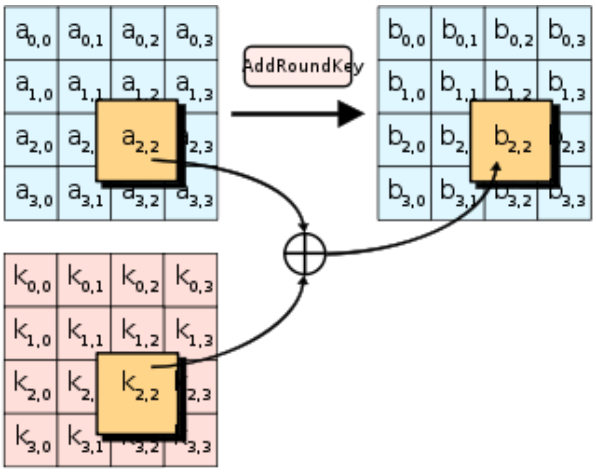
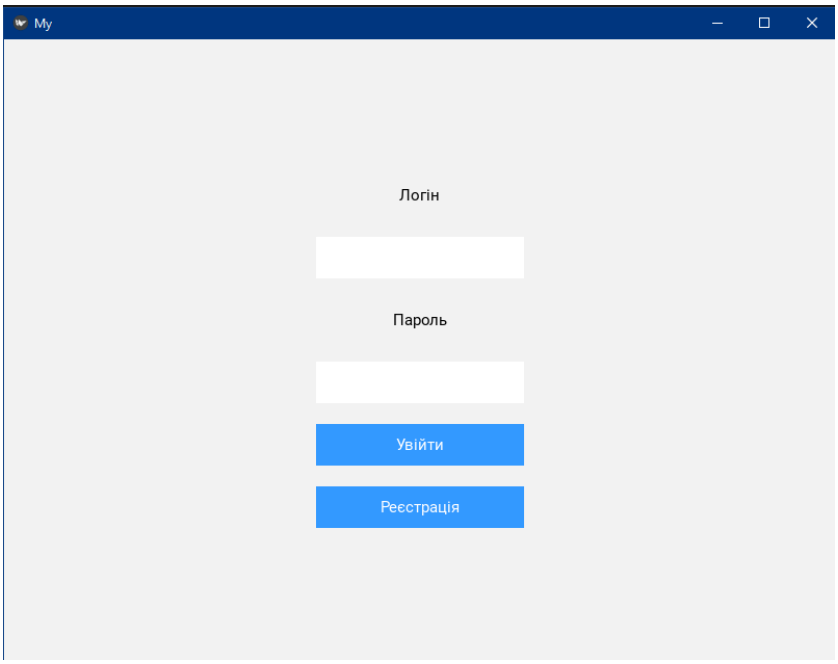
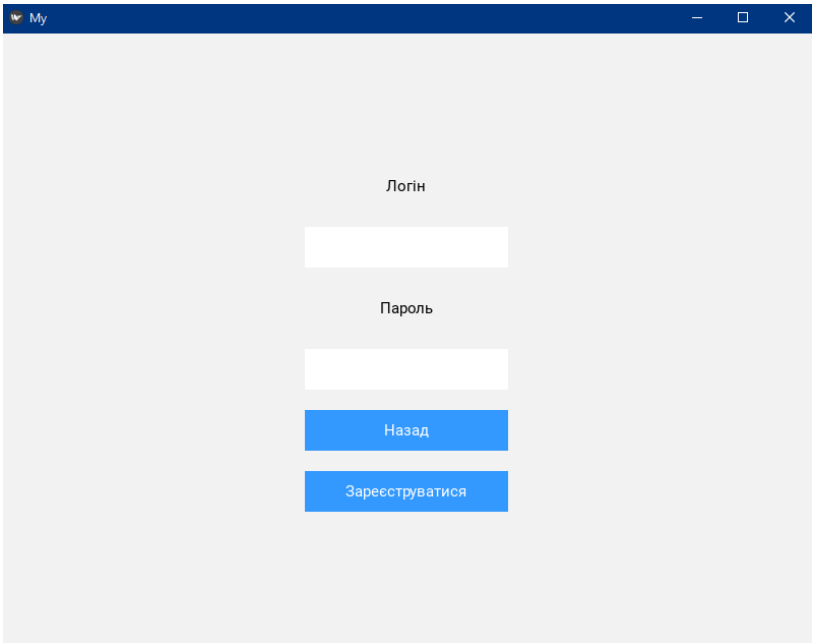


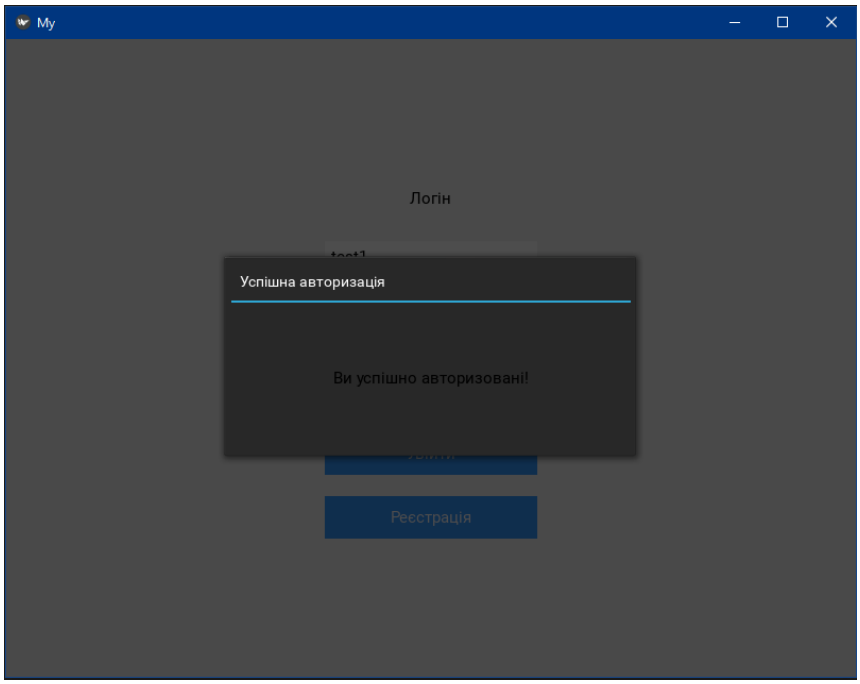
Схема операції xorRoundKey() алгоритму AES



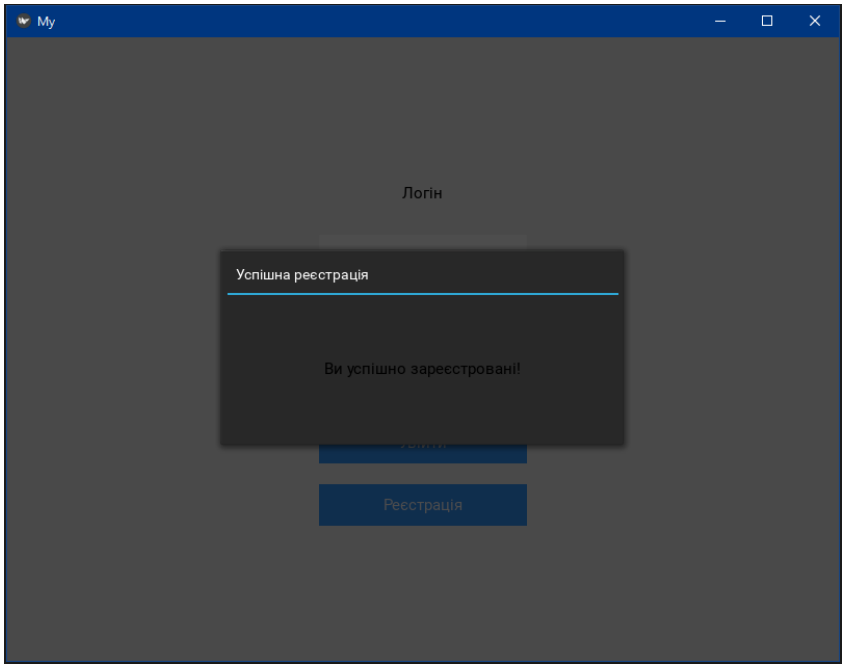
Скріншот екрану авторизації



Скріншот екрану реєстрації



Скріншот успішної авторизації



Скріншот успішної реєстрації