Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: «Розробка методів та програмного додатку управління паролями і їх шифрування»

Виконала: студентка II курсу
групи 1ПІ-19м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Штокал А.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(прізвище та ініціали)

Рецензент: д.т.н., проф. каф. КН Василевський О.М.
(прізвище та ініціали)

ВНТУ – 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 - «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
17 лютого 2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Штокал Аллі Сергіївні

1. Тема роботи: «Розробка методів та програмного додатку управління паролями і їх шифрування»
керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 9 березня 2021 року № 65
2. Строк подання студентом роботи 1 червня 2021 року
3. Вихідні дані до роботи : метод оцінювання часу нагадування для заміни старого паролю на новий та метод оцінювання надійності паролю, повний UX/UI дизайн для клієнтського додатку, технології та програмні засоби для розробки серверного веб-додатку та серверного оточення для його роботи (Node.js, Docker, SSH), проєктування архітектури та вибір технологій і програмних засобів для розробки клієнтського додатку для Web (Java, React та Redux).
4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задач дослідження; розробка методу і моделей системи; програмна реалізація системи; тестування програми; економічна частина; висновки; перелік посилань; додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
актуальність теми; мета, об'єкт та предмет дослідження; основні задачі; загальна структурна схема IT-системи; UI/UX дизайн клієнтського додатку; програмні засоби для управління паролями і їх шифруванням у вигляді бекенду системи, що містить API для роботи з клієнтським web-додатком; результати тестування; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О. М., к.т.н, доцент кафедри ПЗ		
5	Бальзан М.В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання 18 лютого 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз, вибір та обґрунтування актуальності розробки	20.02.2021– 8.02.21	Вик.
2	Аналіз існуючих аналогів	01.03.20 – 12.03.21	Вик.
3	Розробка методу і моделей системи управління паролями	13.03.21 – 22.03.21	Вик.
4	Розробка макетів графічного інтерфейсу	23.03.21 – 31.03.21	Вик.
5	Програмна реалізація системи	01.04.21 – 25.04.21	Вик.
6	Тестування роботи системи	26.04.21 – 30.04.21	Вик.
7	Економічна частина	01.05.21 – 16.05.21	Вик.
8	Оформлення матеріалів до захисту МДР	17.05.21– 31.05.21	Вик.

Студент _____ (підпис) **Штокал А. С.** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ (підпис) **Хошаба О. М.** (прізвище та ініціали)

Рецензент магістерської кваліфікаційної роботи _____ (підпис) **д.т.н., проф. каф. КН Василевський О.М.** (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі розроблено Web-додаток для управління паролями і їх шифрування, а також розглянуто методи шифрування. Удосконалено метод оцінювання складності паролів, а також метод визначення часу автоматичних нагадувань для зміни паролю, що дозволяє підвищити безпеку системи.

Серед реалізованих методів шифрування паролів було обрано методи HMAC, а також SHA512. Було реалізовано багатофакторну автентифікацію, а також застосовано використання тимчасового пароля для входу. Додатково у додатку реалізовано функціонал автоматичного перешифрування паролів з конкретним часовим інтервалом, який може бути встановлений користувачем.

У Web-додатку реалізовано додавання нових паролів, можливість їх редагування, перешифровування всіх паролів під час зміни головного мастер-пароля, інтерфейс, що дозволяє переглянути історію змін паролів, а також можливість скасувати останні зміни. У додатку впроваджено автоматичні нагадування про зміну паролів. Розроблено також інтерфейс для адміністратора, який не має доступу до паролів користувачів, але може переглядати логи – тобто історію дій, які виконували користувачі. Такий підхід дає можливість передбачити небажані дії користувачів, а також вберегти дані користувачів від можливих атак хакерів.

Платформу реалізовано засобами фреймворку Spring Boot та React з використанням мови програмування Java та JavaScript, відповідно. Використано інтегроване середовище розробки IntelliJ IDEA Community 2021.

ANNOTATION

In the master's thesis developed a Web-application for password management and encryption, as well as methods of encryption. The method of estimating the complexity of passwords has been improved, as well as the method of determining the time of automatic reminders to change the password, which allows to increase system security.

Among the implemented methods of password encryption, the HMAC and SHA512 methods were chosen. Multifactor authentication was implemented, and a temporary login password was used. Additionally, the application implements the functionality of automatic password encryption with a specific time interval that can be set by the user.

The Web-application implements the addition of new passwords, the ability to edit them, re-encrypt all passwords when changing the master password, an interface that allows you to view the history of password changes, as well as the ability to cancel recent changes. The application has automatic reminders to change passwords. An interface has also been developed for an administrator who does not have access to user passwords, but can view logs - that is, the history of actions performed by users. This approach makes it possible to anticipate unwanted user actions, as well as protect user data from possible hacker attacks.

The platform is implemented using the Spring Boot and React framework using the Java and JavaScript programming languages, respectively. The integrated development environment of IntelliJ IDEA Community 2021 is used.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Одним з найбільш важливих кроків, які можна застосувати, щоб захистити себе – це використання унікального надійного паролю для кожного з облікових записів і додатків. На жаль, практично неможливо запам’ятати всі свої паролі. Крім того, щоб постійно вводити паролі на різних сайтах, генерувати нові паролі, відслідковувати відповіді на всі питання безпеки потрібно досить багато часу. А використовувати той самий пароль для всіх аккаунтів надзвичайно небезпечно. Однак існує рішення, що може спростити життя в безпечний спосіб – це менеджери паролів.

З назви зрозуміло, що менеджер паролів - це програма, що дозволяє зберігати всі введені паролі (і не тільки паролі) і керувати ними [1]. Це її призначення. Такі додатки виступають в ролі сховища даних, часто прив'язаного до серверів однієї компанії, яка зобов'язується ці дані оберігати від зловмисників.

Менеджер паролів допомагає швидше заповнювати форми для авторизації і автоматизує цей процес. Але що ще важливіше, він бере на себе процедуру створення пароля. Необережні користувачі вибирають собі паролі на кшталт «123456» або «пароль». Вони роблять це, щоб не запам'ятовувати щось більш складне, що підсумку ставить під загрозу особисті дані.

У більшості браузерів є хоча б примітивне управління паролями. Це краще, ніж використовувати один і той же пароль для всіх сервісів, але можливості браузерів обмежені - все-таки вони створювалися для інших цілей. Більшість з них не згенерують надійний пароль. Експерти з безпеки рекомендують використовувати спеціалізовані сервіси. Вони мають одне призначення і рік за роком отримують нові корисні функції. Тому актуальним є розробка власної системи управління паролями та їх шифруваня.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження.

Метою роботи є підвищення ефективності управління паролями та їх шифрування за рахунок методу оцінювання складності паролю, що дає можливість підвищити безпеку системи збереження паролів.

У відповідності до поставленої мети потрібно виконати такі **завдання**:

– розробити метод оцінювання складності паролю;

– розробити модель системи управління паролями та їх шифрування;

– розробити UX/UI дизайн для клієнтського web-додатку;

– розробити програмну реалізацію системи управління паролями та їх шифрування;

– розробити підсистему комунікації з серверною частиною через JSON-API за принципами REST;

– спроєктувати архітектуру бази даних веб-системи;

– налаштувати серверне оточення для розміщення системи управління паролями;

– розробити серверну частину та програмні засоби системи, що містить API для роботи з клієнтським web-додатком;

– протестувати програмний додаток.

Об’єктом дослідження є процес управління паролями та їх шифрування.

Предметом дослідження є програмні засоби управління паролями та їх шифрування.

Методи дослідження. У процесі дослідження використовувались такі методи:

– методи роботи з базами даних для побудови бази даних системи управління паролями і їх шифрування;

– методи побудови компонентів програми та їхніх взаємозв’язків для створення системи управління паролями та їх шифрування;

– методи проєктування програмного забезпечення для виконання розробки програмного продукту.

Наукова новизна одержаних результатів:

1. Подальшого розвитку дістав метод оцінювання складності введеного або генерованого паролю, який, на відмінну від існуючих, використовує оцінку генерованого коду, що надає можливість підвищити безпеку системи та захистити дані користувачів від небажаного доступу.

2. Подальшого розвитку дістав метод зміни паролю, який, на відмінну від існуючих, динамічно змінює час використання паролю з урахуванням ймовірних загроз, що дозволяє підвищити надійність роботи системи.

3. Подальшого розвитку дістала модель системи оцінювання часу встановлення нагадування на зміну старого паролю на новий, яка, на відмінну від існуючих, використовує алгоритми, що працюють зі змінною періодичністю з урахуванням навяного переліку критеріїв для розрахунку наступного часу нагадування про зміну паролю, що дозволяє підвищити захищеність системи.

Практична цінність отриманих результатів. Практична цінність результатів полягає в тому, що розроблені алгоритми і програмні засоби для управління паролями і їх шифрування можуть використовуватися у веб середовищі для підвищення рівня захисту інформаційних ресурсів.

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно.

У публікації [2] автору належать загальна функціональна модель, програмні засоби для управління паролями та їх шифрування.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати досліджень представлені на практичній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2021)»

URL: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/12975

Публікації. За тематикою дослідження опубліковано 1 наукова публікація: у матеріалах міжнародної конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2021)» [2].

Структура та обсяг роботи. Магістерська кваліфікаційна робота містить вступ, п’ять розділів, висновки, список літератури, що містить 23 найменування, та 8 додатків. Робота містить 26 рисунків та 12 таблиць. Ілюстративний матеріал до роботи подано у додатку Є.

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ УПРАВЛІННЯ ПАРОЛЯМИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз систем для управління паролями

Паролі повинні бути складними і різними для кожного сервісу. Але щоб дотримуватися цього правила, необхідно використовувати менеджери паролів. Всі дані для авторизації на різних сервісах будуть зберігатися в ньому.

Менеджери паролів допомагають створювати надійні унікальні паролі при реєстрації на веб-сайтах і зберігають їх на своїх серверах. Щоб зайти на ресурс або в додаток, можна скопіювати потрібний пароль з менеджера і вставити у відповідне поле. Часто ці програми дозволяють не тільки запам'ятовувати, але й автоматично вводити пароль на сайті.

Перед початком варто звернути увагу на способи зберігання даних в менеджерах паролів. Інформація може зберігатися або віддалено в хмарі сервісу, або локально, на комп'ютері користувача. У кожного із способів є плюси і мінуси. Хмарне зберігання і синхронізація зручні тим, що сервіси паролів доступні на всіх пристроях, де вони потрібні. Але, якщо хмара зламають, паролі можливо потраплять до зловмисників.

Локальне місце зберігання більш надійне (хіба що у вас не вкрадуть комп'ютер або ноутбук), але менш зручний [3]. Припустимо, ви створили і пароль від Фейсбуку за допомогою менеджера паролів і зберегли інформацію на ПК. Але якщо зайти в Facebook зі смартфона, то новий пароль автоматично НЕ підтягнеться і його доведеться вводити вручну. Такий підхід А якщо ми говоримо про десятки акаунтів на різних сайтах, то подібна схема стає занадто незручною.

Існують різні види подібного роду програм. Одні поширюються в вигляді розширень для популярних браузерів, інші - можна запускати в якості програмного забезпечення на комп'ютері. Також, більшість менеджерів паролів,

можна встановлювати на смартфонах і планшетах. Вибирайте відповідний варіант і зберігайте свої паролі і секретну інформацію в стовідсоткової безпеки.

1.2 Порівняльний аналіз аналогів

Перед тим як приступити до розробки власного веб-додатку для управління паролями було проведено аналіз ринку. Нижче представлено порівняльний аналіз програм до управління паролями та їх шифрування.

Спочатку було обрано параметри до порівняння. Серед головних критеріїв було визначено наступні: можливість 256-бітового AES-шифрування, наявність протоколів з нульовим розголошенням, забезпечення аутентифікації, а також наявний функціонал. Менеджери паролів пропонують безліч різних функцій: наприклад, автоматичне заповнення форм, генерація паролів, а деякі з них дозволяють навіть підключитися до VPN. Було зроблено огляд деяких з цих функцій, щоб з'ясувати, які з них є дійсно корисними, а які - просто приємними доповненнями. До уваги було брано теж такий критерій, як простота використання,адже менеджер паролів передусім повинен бути зручним. Також було проаналізовано ціну кожного з рішень.

Першим рішенням до порівняння було обрано Google Chrome - вбудований менеджер паролів, який пропонує можливість їх зберігання під обліковим записом Google (рисунок 1.1). Він зручний, доступний і зрозумілий кожному користувачеві. Менеджер Chrome має можливість генерувати паролі, але варто відзначити, що ключі, які пропонує Chrome, не здаються такими вже й складними в порівнянні з комерційними аналогами [5]. Наприклад, відсутня можливість поставити більшу кількість знаків або використовувати спеціальні символи. В цілому це - доступний і звичний інструмент, проте деякі фахівці в області інформаційної безпеки вважають його не надійним, так як відсутній майстер-пароль і при компрометації облікового запису є ризик, що зловмисник зможе заволодіти всіма даними. Варто згадати і про те, що основний продукт компанії Google - дані користувачів, які використовуються для таргетування

реклами і іншого. Тому, можливо, не варто зберігати абсолютно всю інформацію в одному вбудованому менеджері, в тому числі - особливо значиму.

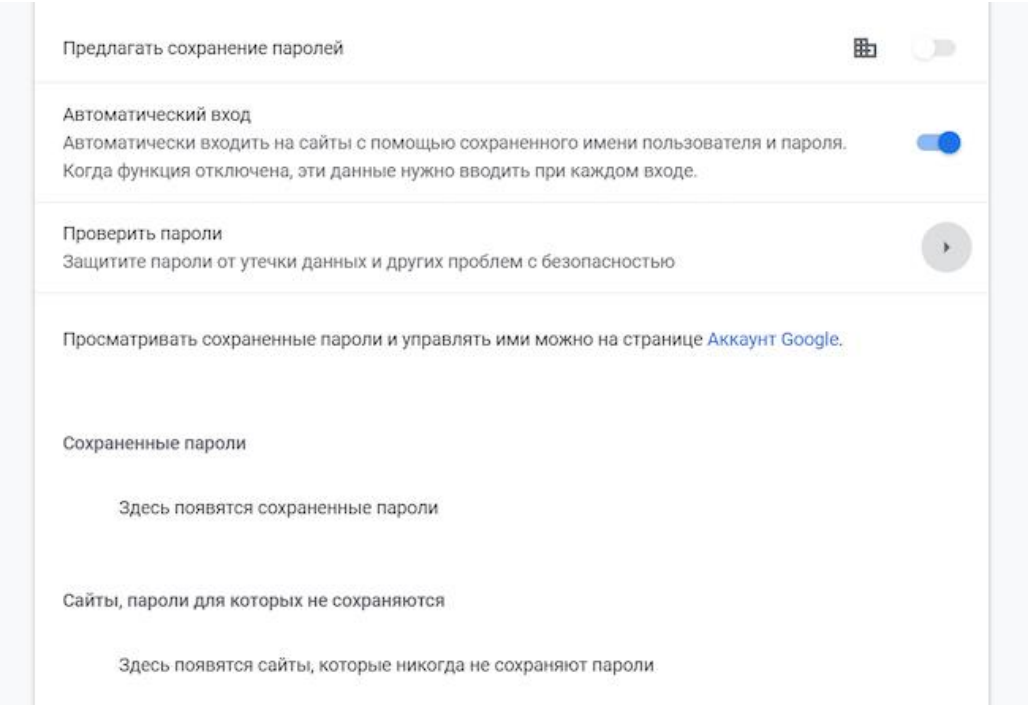

Рисунок 1.1 – Інтерфейс менеджера паролів в Google Chrome

Менеджер паролів Dashlane [6], крім основних функцій надає можливість перевірити паролі в сховище на предмет стійкості. Головне вікно додатку зображено на рисунку 1.2

Також Dashlane підтримує Windows Hello - можливість входу з використанням біометричних даних, в тому числі за допомогою сканування особи і відбитка пальця - і веде моніторинг даркнета, дозволяючи відслідковувати скомпрометовані адреси електронної пошти, паролі та фінансові відомості.

Менеджер має безкоштовну версію, проте її можливості обмежені: в ній можна зберегти не більше 50 паролів.

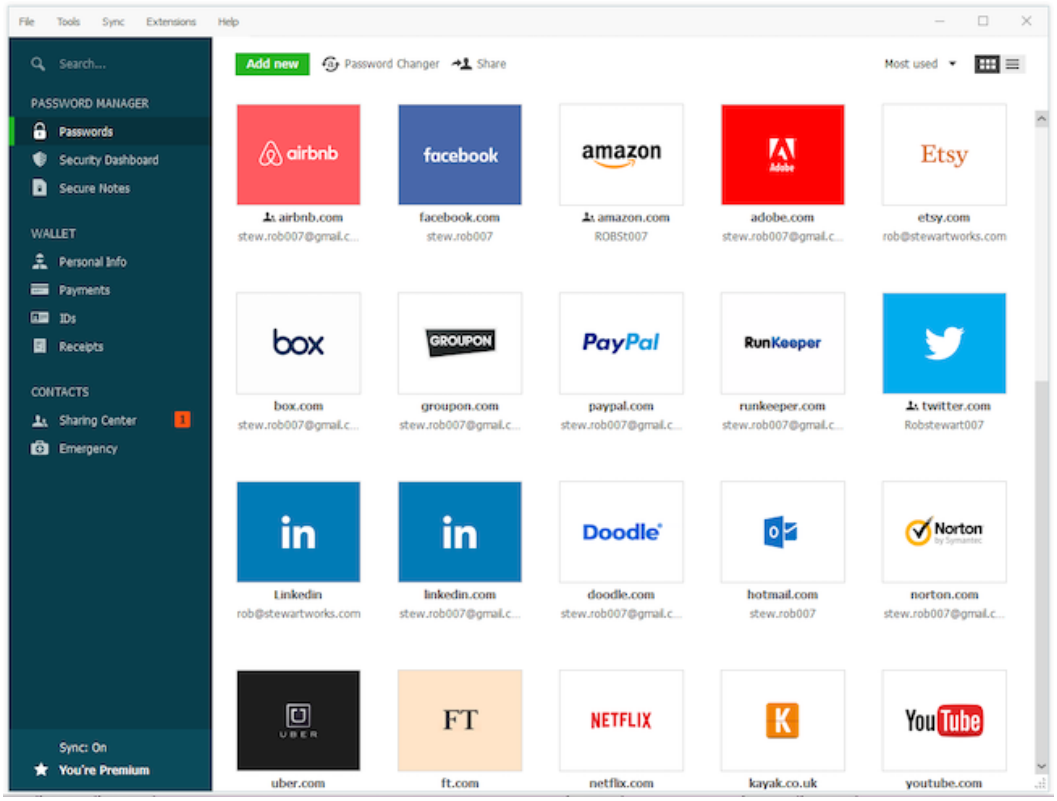

Рисунок 1.2 – Інтерфейс менеджера паролів Dashlane

Ще одним аналогом, який було взято до уваги є менеджер паролей Keeper. Це програма керування паролями та цифровий сейф, створений Keeper Security, який зберігає паролі веб-сайту, фінансову інформацію та інші конфіденційні документи, використовуючи 256-бітове шифрування AES, архітектуру нульових знань та двофакторну автентифікацію [7].

Сервіс пропонує лаконічний і зручний інтерфейс і надає безпечне сховище об'ємом в 10 ГБ. Так само як і Dashlane, він підтримує біометричну аутентифікацію з Windows Hello.

Головне вікно додатку зображено на рисунку 1.3.

Додатково цей менеджер пропонує користувачеві можливості механізму двофакторної аутентифікації (2FA) під назвою Keeper DNA, що дозволяє генерувати одноразові паролі на мобільних пристроях.

У Keeper вбудовані функції моніторингу даркнета і зашифрований чат, який дозволяє користувачам безпечно обмінюватися файлами.

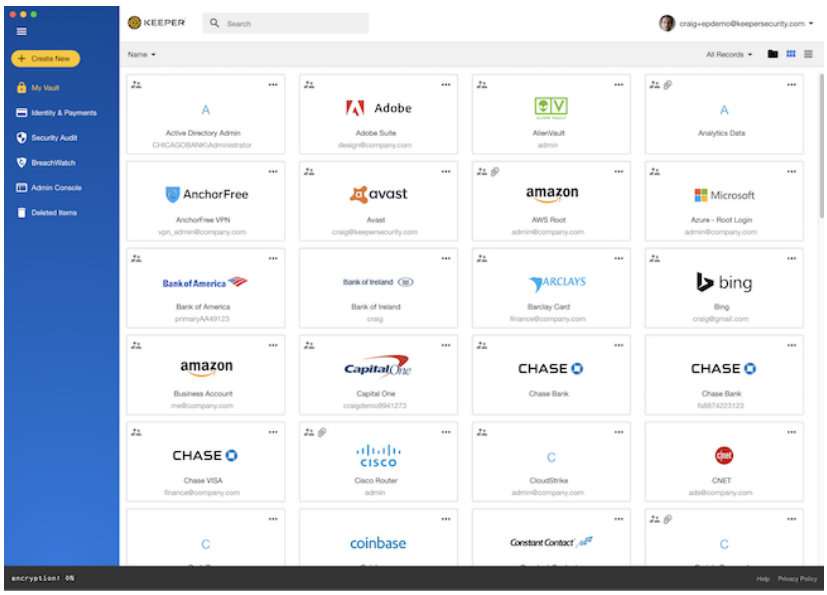

Рисунок 1.3 – Інтерфейс менеджера паролів Keeper

Як і попередні менеджери, 1Password сумісний з Windows Hello і сканує даркнет на предмет витоків інформації. Зашифроване сховище розраховане на зберігання 1 ГБ даних [8]. З унікальних функцій менеджера необхідно виділити сімейний тариф, який дозволяє приєднати одночасно до п'яти користувачів з необмеженою кількістю пристроїв, і вбудовану функцію батьківського контролю, яка дозволяє простежити за тим, щоб діти не змогли змінити паролі від важливих ресурсів. Головне вікно додатку зображено на рисунку 1.4.

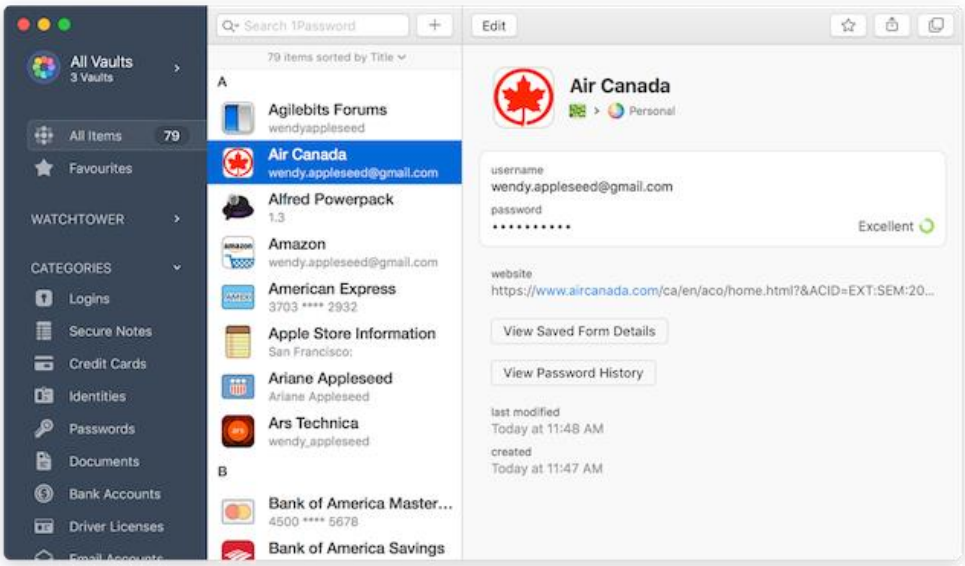

Рисунок 1.4 – Інтерфейс менеджера паролів 1Password

Безкоштовна версія LastPass [9] пропонує найширший спектр можливостей з усіх комерційних менеджерів паролів, існуючих на ринку: вона дає можливість збереження необмеженої кількості паролів на необмеженій кількості пристроїв з додатковою можливістю надання доступу до них одному користувачеві (рисунок 1.5).

Преміум-версія дозволяє давати доступ кільком користувачам, а також забезпечує біометричну ідентифікацію, 1 ГБ простору в сховище і цілодобову підтримку користувачів по електронній пошті.

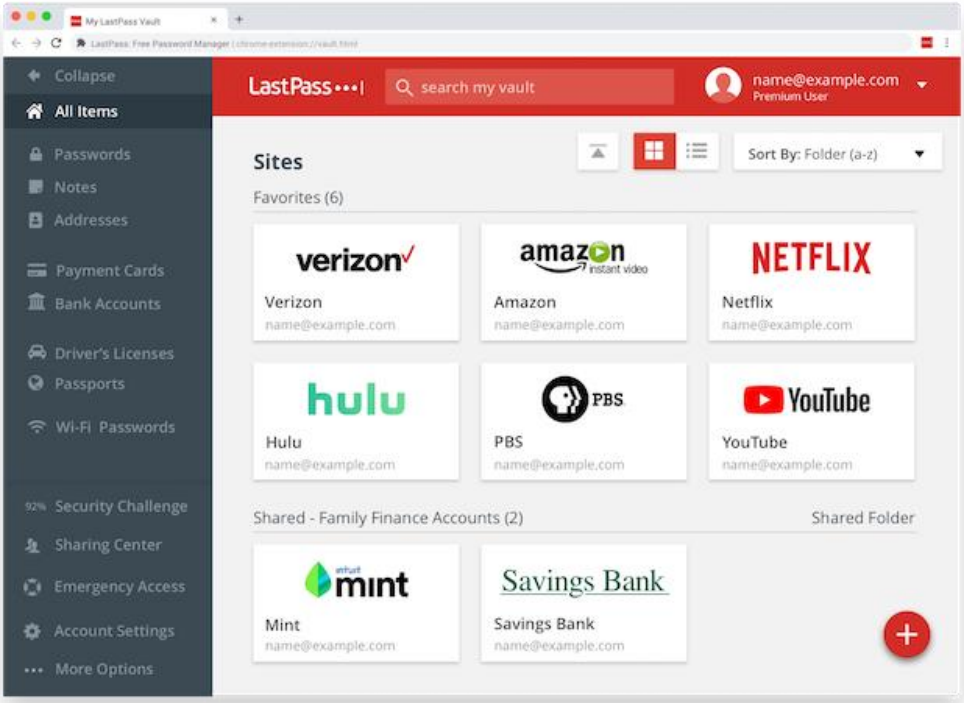

Рисунок 1.5 – Інтерфейс менеджера паролів LastPass

Підсумувавши все вищесказане, можна відзначити, що використання менеджерів паролів дозволяє значно полегшити роботу з веб-сервісами та убезпечити облікові записи. Даний спосіб зберігання паролів набагато надійніше традиційних, таких як використання одного і того ж пароля або комбінації символів з мінімальним розходженням, і можна відзначити, що все

більше користувачів починають освоювати менеджери. Однак при виборі менеджера варто враховувати його особливості та призначення.

Наприклад, вбудовані в браузер менеджери зручні, зрозумілі більшості користувачів, але поки все ще програють комерційним аналогам по здатності генерувати складні паролі, немає можливості перемикатися між браузерами або експертувати паролі до файлу. Через це більшість експертів схиляються до того, щоб розглядати такі менеджери швидше як розширення браузерів і не зберігати в подібного роду додатках важливі дані, наприклад для банківських сервісів. Комерційні менеджери володіють великим спектром можливостей, вони більш комплексні і працюють у вигляді незалежних додатків. Підсумовуюча таблиця подана нижче (таблиця 1).

Таблиця 2.1 – Порівняльна таблиця аналогів

Критерій порівняння	Google Chrome	Dashlane	Keeper	1Password	LastPass	Моє рішення
Експорт паролів до файлу	-	+	+	-	+	+
Мастер-пароль	-	+	+	+	+	+
Генератор складних паролів	+	+	+	+	+	+
Спільний доступ до сховища	-	+	+	+	-	+
Можливість rollback користувача	-	-	-	-	-	+

Продовження таблиці 2.1

Блокування ір адреси	-	+	+	+	+	+
Вартість	безкоштовно	0-5,99 долара США на місяць	2,91-6,01 долара США на місяць	3,99-7,99 долара США на місяць	0-3,9 євро в місяць	Безкоштов-но

Серед розглянутих аналогів, було відзначено наступні недоліки: блокування акаунту в цілях безпеки, блокування ір адреси, щоб запобігти спробам зламу. Деякі з представлених рішень не передбачають наявність мастер-пароля. Натомість, пропоноване рішення в цій магістерській кваліфікаційній роботі передбачає використання мастер пароля до шифрування всіх інших паролей, що підвищує безпеку такого додатку. Кожна наступна зміна мастер-пароля передбачає перешифрування всіх паролів. Також, варто зазначити, що не всі з представлених рішень передбачають можливість спільного доступу до пароля, а також експорту паролів до файлу.

Саме тому актуальним є розробка власного рішення, яке поєднає в собі реалізацію недоліків та відсутнього функціоналу в кожному з розглянутих аналогів.

1.3 Постановка задач роботи

Після порівняльного аналізу аналогів до системи у магістерській кваліфікаційній роботі було вирішено скласти список завдань, необхідних для створення програмного продукту.

Головною задачею роботи є створення системи, що дозволить спростити процес управління паролями та їх шифрування.

Виконання цієї задачі передбачає:

– проведення обґрунтування доцільності розробки;
– провести аналіз інформаційного забезпечення розробки;
– розробка методу визначення надійності паролю;
– розробка методу оцінювання необхідності змінювати пароль;
– розробка моделі системи управління паролями та їх шифрування;
– розробка повного UX/UI дизайну для клієнського web-додатку;
– програмна реалізація системи управління паролями та їх шифрування;
– проектування архітектури бази даних веб-системи;
– налаштування серверного оточення для розміщення системи управління паролями та їх шифрування;
– розробка серверної частини та програмних засобів системи, що містить API для роботи з клієнтським додатком.

1.4 Висновки

Було розглянуто роль менеджерів паролів у сучасному світі та визнано їх важливість. Досліджено види систем для управління паролями та розглянуто їх головні функції. Обрано різновид такої системи, а також розглянуто аналоги. В ході перегляду аналогів системи було визначено їх переваги, які можна покращити: замість функціоналу коментування та вподобання публікацій, було вирішено розробити власні методи оцінювання складності паролів. Було виявлено недоліки аналогів, які врахуються у магістерській кваліфікаційній роботі: можливість відновлення мастер-пароля, автоматичне перешифрування паролів з встановленою періодичністю, використання тимчасового паролю для входу, а також двофакторної автентифікації, забезпечення історії змін паролів і можливість відкату. В результаті дослідження аналогів було сформульовано основні завдання, які необхідно виконати для розробки програмного продукту.

2 РОЗРОБКА МЕТОДІВ І МОДЕЛІ РОБОТИ СИСТЕМИ

2.1 Аналіз інформаційного забезпечення системи

Для реалізації системи необхідно обрати технології, які в повній мірі зможуть задовольнити технічні та функціональні вимоги модулів.

Оскільки веб-додаток має мікросервісну архітектуру, то варто розглянути окрему частини бекенд та фронтенд. Для бекенду основним фреймворком було обрано Spring Boot. Використовуючи Spring Boot можна досягти таких самих результатів, що і на чистій Java, але набагато меншими зусиллями і меншою кількістю коду. Мета даного фреймворку - щоб програміст зосередився на вирішенні бізнес завдання замість того, щоб витрачати час на налаштування коду. Всі налаштування за додатком можна помістити в файл налаштувань замість java класів. Як правило, файл - application.properties лежить в папці resources. Але і це можна змінити і винести настройки на віддалений сервер наприклад або в інший файл.

Spring Boot при запуску завантажує файл залежностей бібліотек. Якщо це Maven то pom.xml. Spring Boot також включає в себе вбудований контейнер сервлетів (по замовчуванні Tomcat) що дозволяє запускати веб додатки як звичайні java програми: використовуючи jar файл.

Для розробки фронтенду основним фреймворком було обрано React. React - це бібліотека JavaScript, яка використовується для створення призначеного для користувача інтерфейсу. Відмінною рисою бібліотеки є концентрація на компонентах - ми можемо створити окремі компоненти і потім їх легко переносити з проекту в проект [4]. Ще одна особливість React - використання JSX. JSX представляє комбінацію коду JavaScript і XML і надає простий і інтуїтивно зрозумілий спосіб для визначення коду візуального інтерфейсу.

2.2 Розробка загальної моделі системи

Загальна функціональна структурна модель IT-системи зображена на рисунку 2.1.

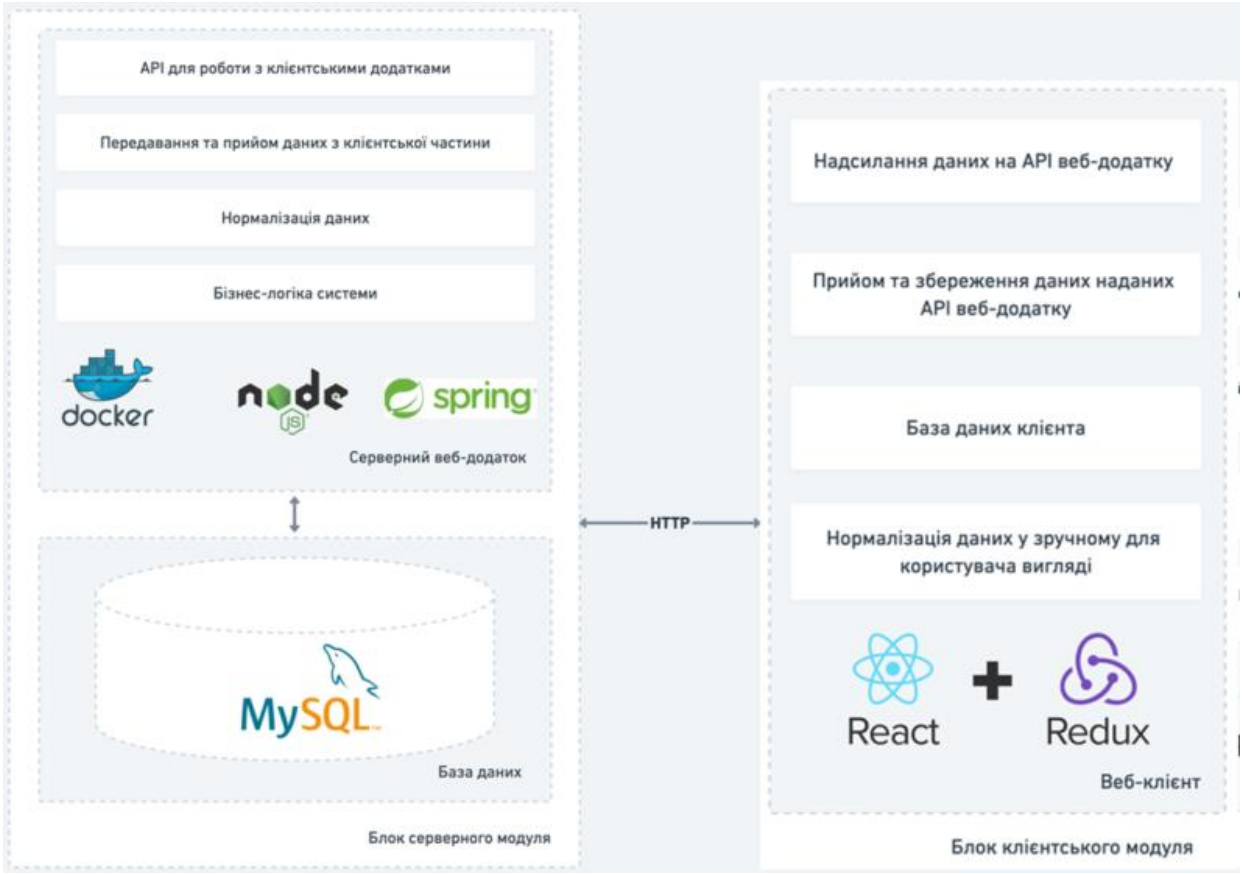

Рисунок 2.1 – Загальна структурна модель IT-системи

Система управління паролями представляє собою багатомодульну програму, що має серверний та клієнтський модулі. Система даними і записує зміни до бази даних по стороні серверної частини. Для роботи клієнтського модуля необхідно зареєструватись та увійти до системи попередньо авторизувавшись.

Серверна частина містить базу даних та всю необхідну логіку веб-системи. База даних призначена для зберігання, зміни і обробки необхідної інформації. Веб-додаток, як частина системи містить у собі бізнес логіку

системи, нормалізацію даних, що будуть записані до бази, REST API, побудоване на контролерах для роботи з клієнтськими модулями. В цілому, серверна частина являється основною та забезпечує роботу клієнтської сторони, виконуючи функцію передавання та прийому даних системи в цілому.

Вся структура веб-сторінки може бути представлена за допомогою DOM (Document Object Model) - організація елементів html, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові [4]. Для взаємодії з DOM застосовується мова JavaScript. Сценарії користувачів визначає бізнес-логіка. Бізнес-сценарії – це набір кроків, які може виконати звиклий користувач. Призначенням цього блоку є захист структури предметної області, контроль або вплив на його операції.

2.3 Розробка методів роботи системи

Для підвищення безпеки і надійності паролів користувачів було реалізовано власний метод визначення надійності паролю. Метод було застосовано як і при ручному вводі символів так і при генерації паролю.

А також було розглянуто автоматичне встановлення нагадувань про зміну старого паролбю на новий. Для цього було зреалізовано метод оцінювання необхідності змінювати пароль на новий

2.3.1 Метод оцінювання необхідності змінювати пароль на новий

Метод оцінювання формує оцінку щодо необхідності змінювати пароль на новий. Розглянемо на прикладі як працює метод. Для кожного пароля до уваги береться набір критеріїв за якими система виставляє оцінки (по 10ти бальній шкалі). Буде застосовано 5 критеріїв оцінки. У кожного критерія є своя вага, відповідно якій одні критерії оцінки є більш вагомі ніж інші. Відповідно критеріям надаються коефіцієнти для обрахувань. Сума коефіцієнтів дорівнює

1. Ці коефіцієнти встановлює користувач. У таблиці 2.1 наявні прикладові дані для обрахування необхідності змінювати пароль на новий.

Таблиця 2.1 – Початкові дані для обрахування

Критерій	Коефіцієнт	Прикладова оцінка системи
Складність паролю	0,25	7.5
Довжина паролю	0,15	8
Час останньої заміни	0,23	5.5
Час останнього використання	0,22	6.5
Загальна кількість використань	0,15	9

Оцінка виставляється в межах від 1 до 10. Обрахування необхідності змінювати пароль на новий обчислюється за формулою 2.1.

$$\mathrm{R} = \sum_{i=0}^{n} A_i * K_i \qquad (2.1)$$

де

A_i – оцінка системи;

K_i – коефіцієнт критерія;

Таким чином отримуємо результат у балах від 1 до 10.

$R = 7{,}5 * 0{,}25 + 0{,}15 * 8 + 5{,}5 * 0{,}23 + 6{,}5 * 0{,}22 + 0{,}15 * 9 = 7.12.$

З результату дістаємо оцінку необхідності змінювати пароль на новий, яка дорівнює 7,2.

2.3.2 Метод оцінювання складності введеного паролю

Було розроблено алгоритм для оцінювання рівня складності введеного пароля. Рівень складності пароля обчислюється в залежності від обчислюваної «бітової стійкості» [0;4], а також коефіцієнту подібності :

bits ≥ 128 – 4;

128 <bits ≥ 64 – 3;

64 <bits ≥ 56 – 2;

bits <56 – 1;

порожній пароль – 0.

Оцінка складності, розраховується за формулою 2.2: «бітова стійкість» помножена на коефіцієнт подібності:

$$P = \log(charset)*(\frac{length}{\log(2)})*(1 - k) \qquad (2.2)$$

де:

- P – складність;
- log – натуральний логарифм;
- ength – довжина пароля;
- charset – розмір для типу множини.

Існуючі можливі типи подано нижче. Кожен з них береться до уваги, в залежності від того чи вони зустрічаються в рядку:

- малі англійські букви [abcdefghijklmnopqrstuvwxyz];
- заголовні англійські букви [ABCDEFGHIJKLMNOPQRSTUVWXYZ];
- спеціальні символи [~ `! @ # $% ^ & * () -_ + ="];
- цифри [1234567890];
- k – коефіцієнт подібності (розраховується за даними, для користувача) в діапазоні від 0 до 1;

Коефіцієнт подібності розраховується як підрахунок входження підрядків згенерованої бази фраз у введеному користувачем паролі.

2.4 Розробка прототипів системи

Інтерфейс – інтеракція,що існує між двома функціональними об'єктами, вимоги до якої визначаються стандартом; сукупність засобів, методів і правил взаємодії між елементами системи [6]. Іншими словами, користувацький інтерфейс, скорочено UI (user interface) – це спосіб, яким користувач виконуєте будь-яку задачу за допомогою будь-якого продукту, а саме дії, що ним здійснюються і те, що він отримуєте у відповідь [7].

Перше вікно з яким має справу незареєстрований користувач, це вікно реєстрації.

Форма реєстрації містить наступні поля:

- Login – текстове поле, для введення логіну;
- Email address – адреса електронної пошти, поле з валідацією на наявність символу @;
- Password/Confirm password – поле головного мастер-пароля ;
- Password Level1/level2 – тимчасові паролі на два рівні доступу;
- Login Time – кількість невдалих логувань;
- Type of encoding – метод шифрування;

На рисунку 2.1 зображено протопи вікна реєстрації, з усіма наявними полями та кнопками.

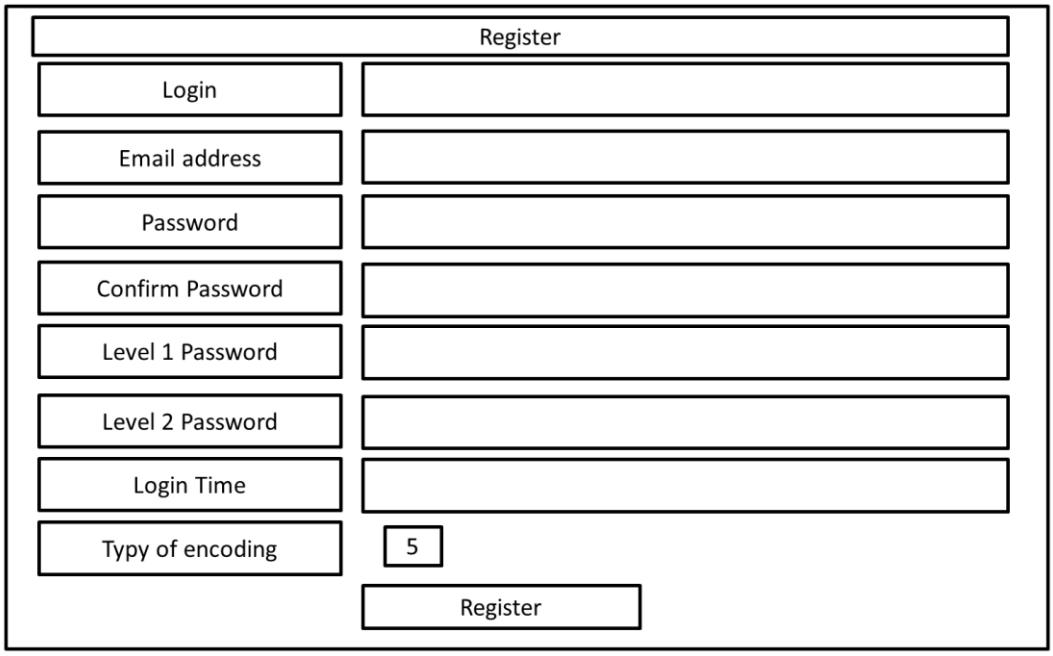

Рисунок 2.1 – Прототип екрану «Реєстрація»

Після реєстрації, користувач повинен увійти до системи, щоб мати змогу користуватися сервісом. Процес входу має автентифікацію через пошту. Первинне вікно для входу (рисунок 2.2) виглядає наступним чином:

- login - основна інформація про користувача;
- password - пароль користувача;
- remevber me – чекбокс, що відповідає за збереження сесії;
- forgot password – посилання для відновлення мастер-пароля.

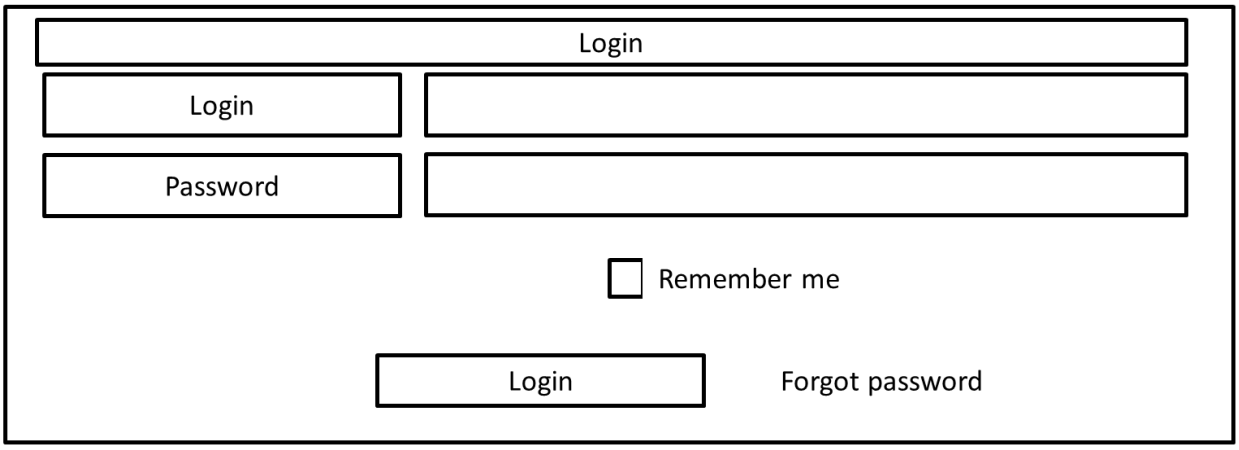

Рисунок 2.2 – Прототип екрану «Домашньої сторінки при авторизації»

Після успішного залогування користувач потрапляє на головне вікно додатку. Прототип вікна зображено на рисунка 2.3.

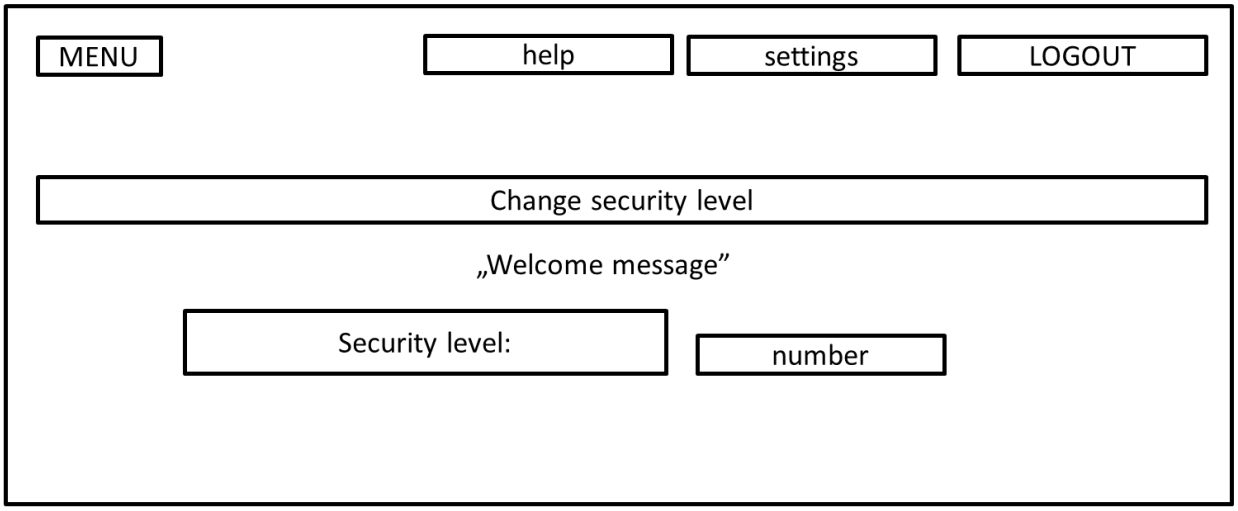

Рисунок 2.3 – Прототип екрану «Головної сторінки»

На рисунку 2.4 відображений процес зміни рівня безпеки. Для цього користувачеві знадобиться пароль, що відповідає тому рівню, на який хоче перейти користувач.

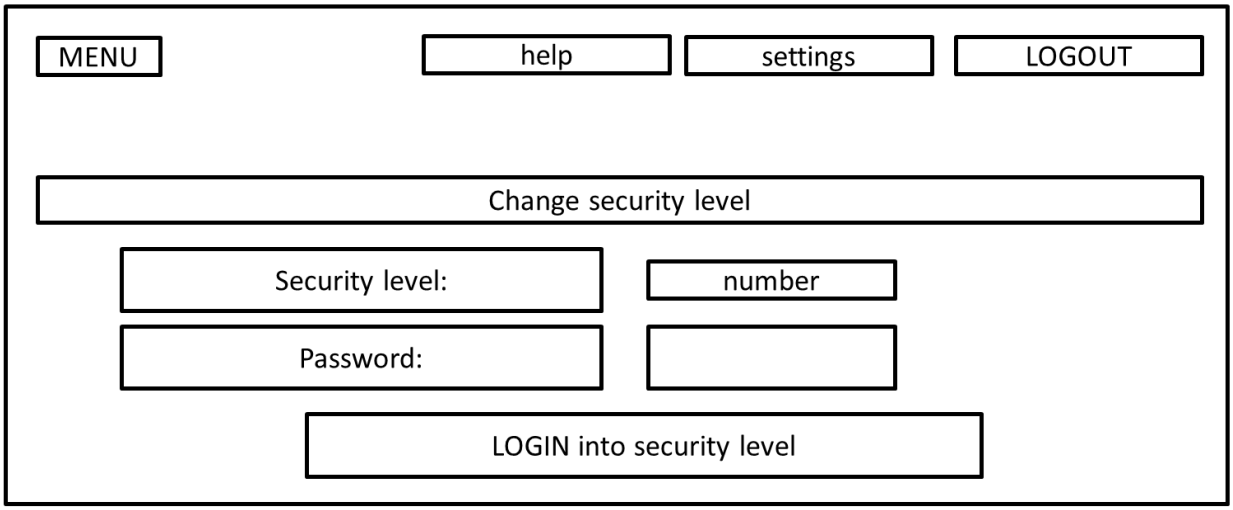

Рисунок 2.4 – Прототип екрану «Логування на другий рівень безпеки»

На другому рівні безпеки користувач може переглядати або редагувати вже додані паролі, а також додавати нові. На рисунку 2.5 відображено прототип вікна додавання/редагування паролю.

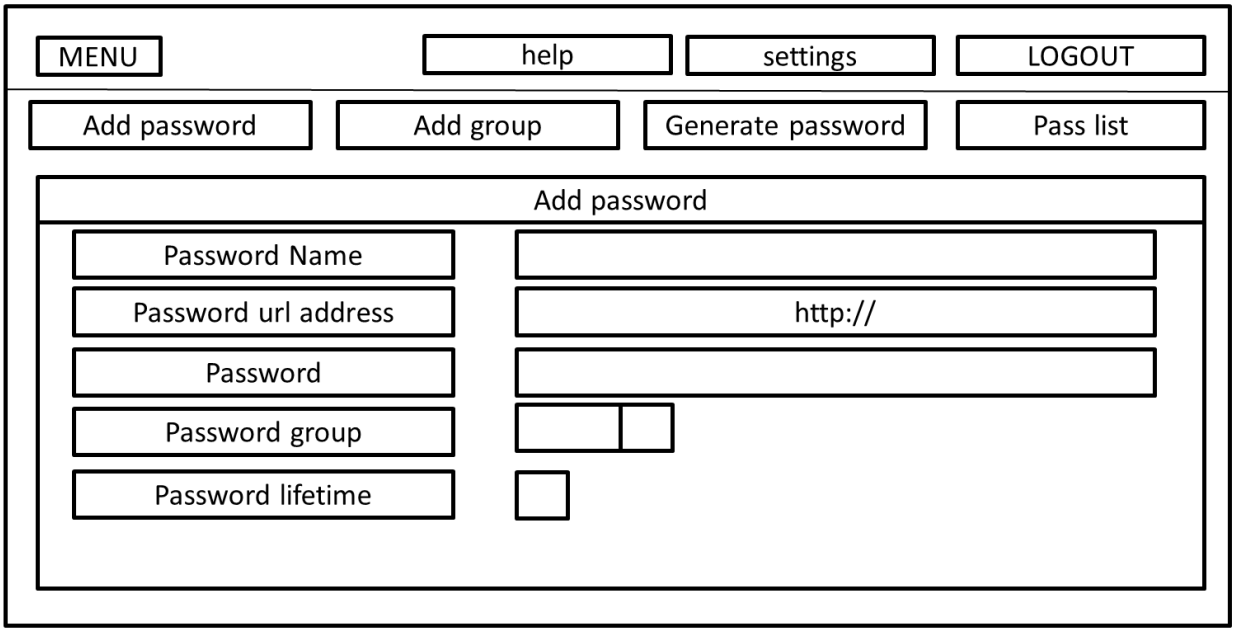

Рисунок 2.5 – Прототип екрану «Додавання/редагування паролю»

Як тільки користувач залогується до другого рівня безпеки, йому доступноою стає можливість переглядати історію своїх змін, та повернутися до якоїсь конкретної точки, щоб відмінити всі нові зроблені зміни. Прототип вікна історії зображено на рисунку 2.6

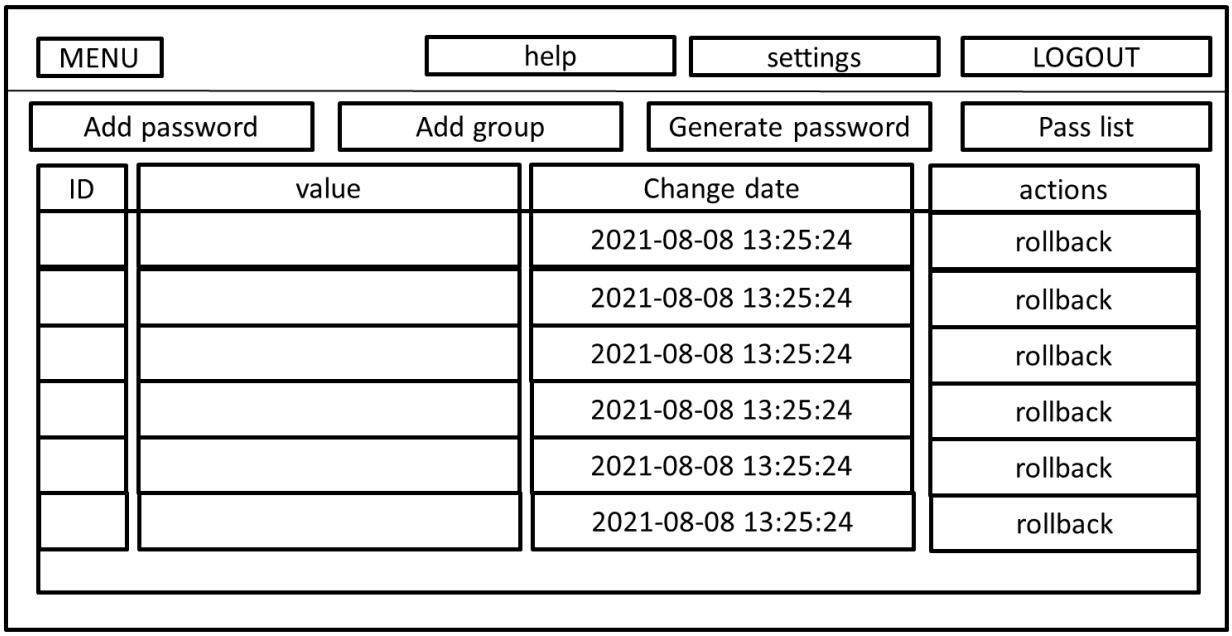

Рисунок 2.6 – Прототип екрану «Історія змін»

Під час відкату до попередньої версії паролю, до історія додається новий запис з заголовком про відновлення.

2.5 Висновки

В цьому розділі було проаналізовано інформаційне забезпечення системи та розглянуто початкову версію інтерфейсу. Розглянуто загальну функціональну структурну модель, а також розроблено власні методи оцінювання складності паролю, а також алгоритм сповіщення користувачів для зміни старого паролю на новий. Також було розглянуто принципи побудови прототипів інтерфейсу, що побудовані на засадах UI/UX, що дозволяє зробити інтерфейс інтуїтивно-зрозумілим для користувача та пришвидшить його роботу з додатком [8].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програмного продукту

Для реалізації системи необхідно обрати технології, які в повній мірі зможуть задовольнити технічні та функціональні вимоги модулів.

Оскільки веб-додаток має мікросервісну архітектуру, то варто розглянути окрему частини бекенд та фронтенд.

Для бекенду основним фреймворком було обрано Spring Boot. Використовуючи Spring Boot можна досягти таких самих результатів, що і на чистій Java, але набагато меншими зусиллями і меншою кількістю коду. Мета даного фреймворку - щоб програміст зосередився на вирішенні бізнес завдання замість того, щоб витрачати час на налаштування коду.

Серед основних переваг фреймворку можна відзначити:

- легкість запуску: пакет проектів має вбудований сервер та інші необхідні компоненти, необхідні для запуску програми;
- автоматична конфігурація: для запуску основної програми не потрібна додаткова конфігурація;
- швидкість: спрощення створення додатків за допомогою Spring Boot, що призводить до швидшого та простішого процесу розробки.

Всі налаштування за додатком можна помістити в файл налаштувань замість java класів. Як правило, файл - application.properties лежить в папці resources. Але і це можна поміняти і винести настройки на віддалений сервер наприклад або в інший файл [9].

Spring Boot при запуску завантажує файл залежностей бібліотек. Якщо це Maven то pom.xml. Він просканує всі залежності, які підключені і після використовує файл проперти, щоб підставити настройки в необхідні залежності. Візьмемо для прикладу підключення бази даних. Спочатку вказуються в файлі pom.xml залежності, що будуть використовуватись

наприклад spring-data-jpa. Spring «знає», що дана бібліотека заснована на роботі з базою даних. При запуску він буде шукати у файлі application.properties налаштування до бази (логін, пароль). Якщо він їх не знайде - то не запуститься і видасть відповідне повідомлення.

Spring Boot також включає в себе вбудований контейнер сервлетів (по замовчуванні Tomcat) що дозволяє запускати веб додатки як звичайні java програми: використовуючи jar файл. Або якщо ще простіше кажучи: просто натискаючи на зелену стрілку в IDE як Ви це робили з “Hello world” додатками. Фактично Spring Boot має автоматичну настройку майже для всіх Spring модулів, які ми зараз розберемо.

Для розробки фронтенду основним фреймворком було обрано React.

React - це бібліотека JavaScript, яка використовується для створення призначеного для користувача інтерфейсу. React був створений компанією Facebook, а перший реліз бібліотеки побачив світ у березні 2013 року. Поточної версій на даний момент (жовтень 2020 року) є версія React v17.0.

Спочатку React призначався для вебу, для створення веб-сайтів, проте пізніше з'явилася платформа React Native, яка вже призначалася для мобільних пристроїв.

React представляється ідеальний інструмент для створення масштабованих веб-додатків (в даному випадку мова йде про фронтенді), особливо в тих ситуаціях, коли додаток являє SPA (односторінкове додаток).

React відносно простий в освоєнні, має зрозумілий та лаконічний синтаксис.

Відмінною рисою бібліотеки є концентрація на компонентах - ми можемо створити окремі компоненти і потім їх легко переносити з проекту в проект. Ще одна особливість React - використання JSX. JSX представляє комбінацію коду JavaScript і XML і надає простий і інтуїтивно зрозумілий спосіб для визначення коду візуального інтерфейсу.

Вся структура веб-сторінки може бути представлена за допомогою DOM (Document Object Model) - організація елементів html, якими ми можемо

маніпулювати, змінювати, видаляти або додавати нові. Для взаємодії з DOM застосовується мова JavaScript.

3.2 Розробка алгоритмів шифрування паролів

SHA512 - це хеш-функція, яка обчислює хеш-значення для набору даних. Хеш обчислюється за спеціальним алгоритмом без додаткових даних. Це робить можливими деякі атаки - особливо при використанні таблиць. Через це під час перемішування додають сіль. Сіль - це послідовність байтів / символів, які додаються до паролю перед хешуванням. Важливо змінювати сіль під час зміни пароля [10].

Сіль зберігається в базі даних. Для кращого захисту паролів ви можете додати перець. Це також випадкова послідовність байтів / символів - але вона зберігається всередині програми, а не в базі даних. Ви можете зберігати перець у коді вашої програми або у файлі конфігурації. Фрагмент коду функції представлено на лістингу 3.1.

```
private static String calculateSHA512(String text) {
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-512");
        byte[] messageDigest = md.digest(text.getBytes());
        BigInteger no = new BigInteger(1, messageDigest);
        String hashtext = no.toString(16);
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
        return hashtext;
    }
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
```

Лістинг 3.1 – фрагмент коду реалізації алгоритму SHA512

Повний лістинг класу, що відповідає за шифрування SHA512 подано в Додатку А.

HMAC - це код автентифікації повідомлень на основі хешу. Він обчислює хеш даних, але з використанням секретного ключа. Це означає, що тільки власник ключа може обчислити хеш [11].

HMAC має багато переваг, таких як велика стійкість проти атак. Схема HMAC представлена на рисунку 3.1.

Лістинг класу, що відповідає за шифрування методом HMAC подано в Додатку В.

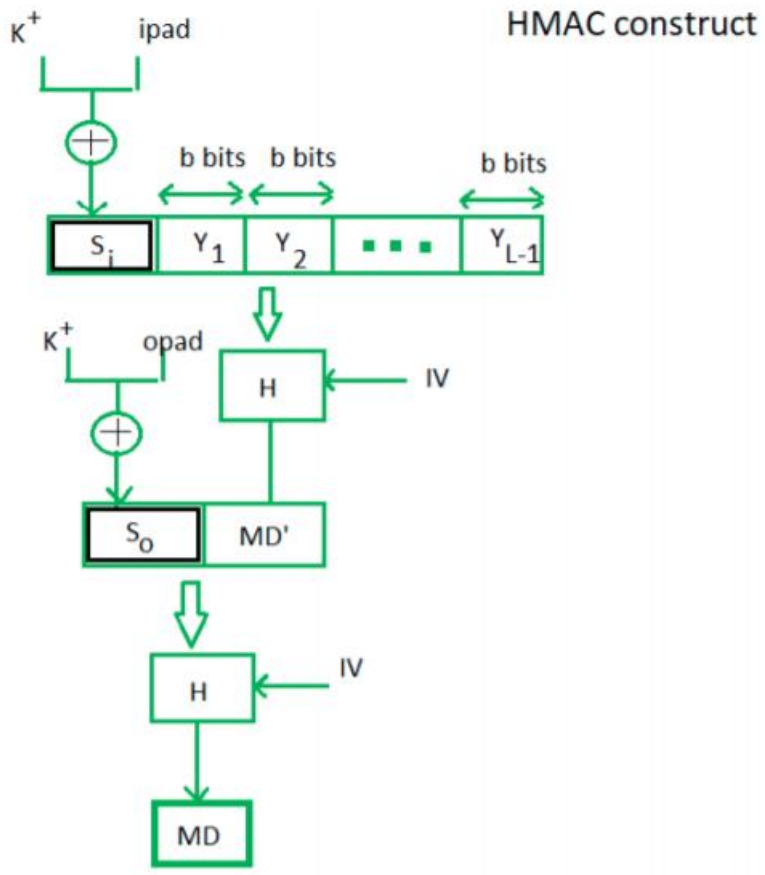

Рисунок 3.1 – Схема HMAC

Серед переваг цього алгоритму можна зазначити наступні. HMAC ідеально підходять для високопродуктивних систем, таких як маршрутизатори, завдяки використанню хеш-функцій, які швидко обчислюються та перевіряються на

відміну від систем із відкритим ключем. Цифрові підписи більші за HMAC, проте HMAC забезпечують порівняно вищий рівень безпеки. HMAC використовуються в адміністраціях, де системи відкритих ключів заборонені.

Якщо говорити про недоліки, то HMAC використовує спільний ключ. Якщо ключ відправника або одержувача пошкоджено, зловмисникам буде легко створювати несанкціоновані повідомлення.

3.3 Розробка структури бази даних

На першому етапі було запроектовано базу даних, яка буде доречною для збереження даних для входу користувачів та їх паролі.

Структура бази даних показана на показаній схемі EER на рисунку 3.2 [12].

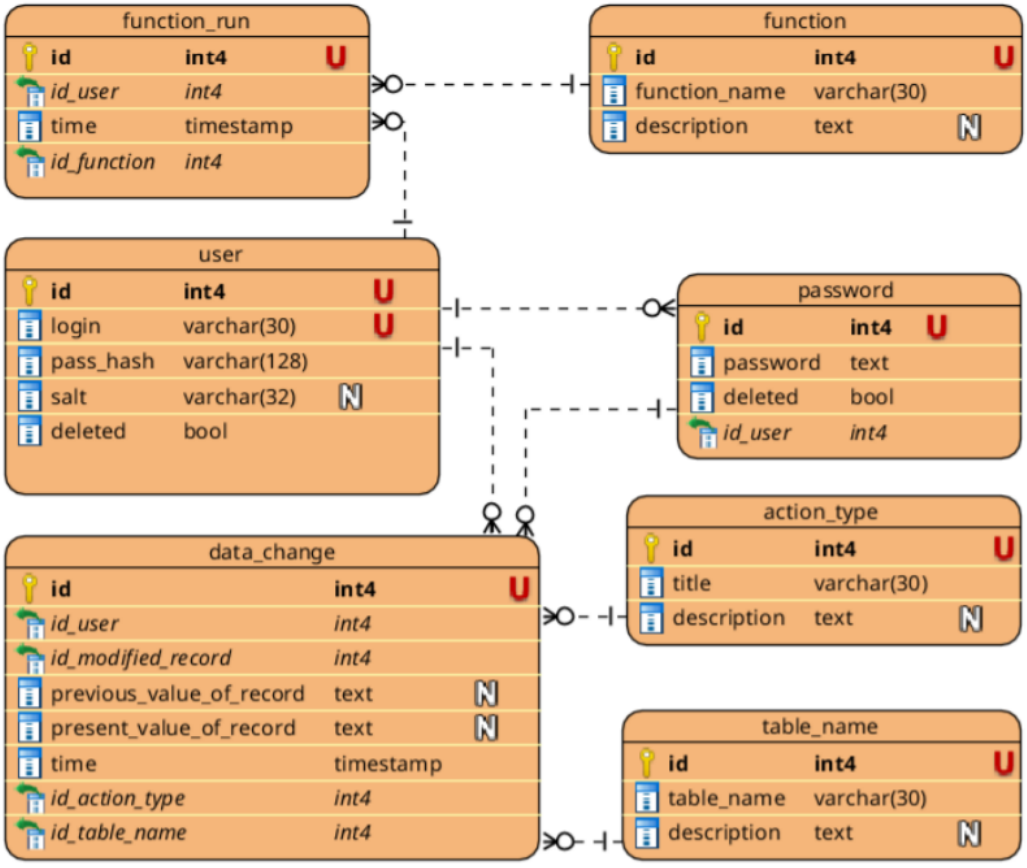

Рисунок 3.2 – Схема структури бази даних

Подана структура бази даних, створена командами мови SQL [13-15]. Код подано в додатку В.

Всі паролі в базі даних зберігаються у зашифрованому вигляді. Це означає, що користувач створює мастер-пароль для шифрування паролів, що зберігаються в базі даних. За допомогою такого головного пароля будуть шифруватися та розшифровуватися паролі користувачів. Мастер-пароль вводиться користувачем лише один раз після входу і зберігається у змінній під час роботи програми.

3.4 Розробка функціональних модулів веб-додатку

Проектування моделі веб-додатку є однією з найважливіших частин процесу розробки програмного забезпечення. Правильно розроблений додаток є швидшим та легшим у реалізації, що означає скорочення робочого часу призначеного для імплементації. Ретельне проектування також дозволяє виявити багато помилок перед тим, як почати писати програму, і дозволяє їх легко виправити. Дизайн веб-сайту включає визначення нефункціональних функціональних вимог, точне визначення структури бази даних та розробку відповідних діаграм для сприяння у впровадженні.

Першим етапом проектування веб-додатку було визначення функціональних вимог [16], яким він буде відповідати. Завдяки цьому вдалося точно визначити завдання веб-сайту, що створюється. Його основна функціональність включає:

- вибір способу шифрування, під час реєстрації нового користувача: SHA512 або HMAC;
- зміна пароля для входу;
- функціонал додавання паролів користувачів;
- функціонал перегляду збережених паролів;
- розшифрування пароля, якщо користувач просить його відобразити;

- реєстрація спроб логування користувачами (запис таких даних, як час входу, результат входу (успішно, не вдалося), IP-адресу користувача);
- представлення користувачеві даних про останній успішний і невдалих вхід;
- збереження інформації про кількість наступних неправильних пробних входів користувачів;
- у випадку, якщо користувач не зміг увійти принаймні два рази, час очікування входу користувача збільшується до 5 секунд;
- у випадку, якщо користувач не зміг увійти щонайменше тричі встановлюється продовжиення часу перевірки входу користувача до 10 секунд;
- у випадку, якщо користувачу не вдався увійти як мінімум чотири рази, обліковий запис користувача блокується на 2 хвилини;
- у випадку успішного входу в систему число неправильних входів скидається до нуля;
- функціонал перевірки IP-адреси, якою користується користувач – підрахунок правильних та неправильних спроб входу з вибраної IP-адреси;
- у випадку, якщо користувачі не змогли увійти щонайменше чотири рази з однієї і тої самої адреси - IP адреса блокується назавжди;
- у випадку успішного входу з конкретної IP-адреси, скиньте кількість наступних неправильні логіни з цієї адреси;
- можливість зняти блокування IP-адреси.

Таким чином, одним з перших кроків реалізації вищеописаних вимог було створення моделей всіх сутностей в додатку: «Пароль», «Користувач», «Лог», «Операція зміни даних», «Виконана функція». Кожна така сутність позначена анотацією Entity. Клас типу Entity вказує на клас, який на абстрактному рівні співвідноситься з таблицею в базі даних. Кожен об'єкт, інстанційований цим класом, вказує модель самої таблиці, що містить інформацію останнього. Поля

класу позначаються анотаціями @Column(), що відповідає назвам стовпців у таблиці. Лістинг кожної з поданих моделей подано в додатку Ґ.

Всі процеси, що виконуються веб-сайтом, можуть бути описані за допомогою графічних позначень, які називаються діаграмами BPMN (Bussiness Process Model and Notation). Вони однозначні, зрозумілі та гнучкі, що полегшує розуміння того, як працюють певні функціональні можливості [17-19]. На рисунку 3.3 показаний процес додавання паролів.

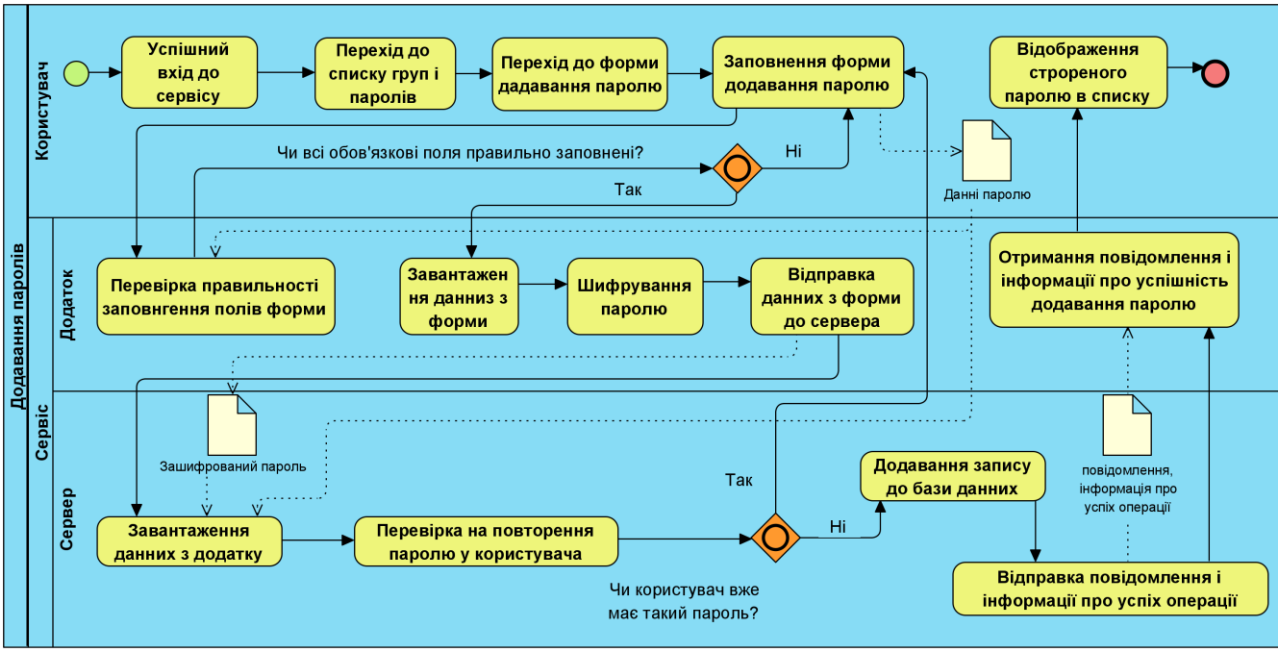

Рисунок 3.3 – Схема структури бази даних

Наступним кроком було визначення нефункціональних вимог [20], тобто таких, які не стосуються функціональності веб-додатку, але визначають спосіб його роботи:

- веб-сайт під’єднується до бази даних SQL;
- для користування веб-сайтом потрібне з’єднання з Інтернетом;
- веб-додаток доступний у браузері
- паролі в базі даних зберігаються у зашифрованому вигляді,
- пароль для входу та паролі доступу для кожного рівня безпеки зберігаються у хеш-формі, ці паролі є ключами шифрування на індивідуальному рівні безпеки;

• веб-сайт використовує алгоритм SHA-512 як хеш-функцію;

• веб-сайт використовує алгоритм AES-256 для шифрування паролів;

У аплікації реалізація вищеописаних вимог виглядає наступним чином. Завдяки контролерам, серверна частина аплікації комунікує з клієнтською [21]. Відповідно існує чотири контролери: UserController, PasswordController, FunctionRunController та DataChangeController. Контролер позначається в коді анотацією @Controller з вказанням методу передачі даних, наприклад: @PostMapping, @GetMapping тощо. Повний код кожного з контроллерів подано у додатку Д.

Головна логіка програми реалізована у сервісних класах, які позначені анотацією @Service. Сервісний клас, що реалізує логіку роботи з паролем і пов'язаний з контролером «PasswordController» подано у додатку Е.

3.5 Висновки

У третьому розділі було обрано технологію реалізації компонентів системи та розроблено алгоритми шифрування паролів. Оскільки для додатку було обрано мікросервісну архітектуру, то бекенд та фронтенд розроблялися окремо.

Для бекенду було обрано фреймворк написаний мовою Java – Spring Boot,а для фронтенду – React. У рамках розробки алгоритму робота програмного додатку, передбачено функціонал додавання паролів, їх редагування, поширення, а також перегляд історії змін паролів і вразі потреби відкат до конкретного моменту часу.Розроблено файл конфігурації програмного додатку, що регулює налаштування програми та бібліотеки, обрано необхідні поля конфігурації, що забезпечать найбільш зручне та водночас гнучке використання програмного застосунку.

Також було розроблено структуру бази даних веб-додатку. Для потреб веб-додатку було прийнято рішення використовувати реляційну базу даних MySQL.

Було визначено основні функціональна та нефункціональні вимоги веб-додатку, а також представлено діаграму процесу додавання паролів.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Аналіз методів тестування

З метою перевірки правильної роботи розробленого веб-сайту були проведені тести. Вони дають змогу виявляти помилки та виправляти їх ще до виходу остаточної версії. Тестування також дозволяє помітити будь-які незручності та труднощі, пов’язані з використанням веб-сайту. Зазвичай тести проводяться з точки зору конкретної особи (часто тієї, для кого створюється веб-сайт) і стосуються конкретного модуля. Ще однією перевагою є той факт, що частину веб-сайту можна протестувати до закінчення всього вихідного коду.

Black-box тестування - це функціональний і нефункціональне тестування без доступу до внутрішньої структурі компонентів системи. Метод тестування «чорного ящика» - процедура отримання та вибору тестових випадків на основі аналізу специфікації (функціональної або нефункціональної), компонентів або системи без посилання на їх внутрішній устрій [18].

За допомогою Black Box Testing можна протестувати будь-який додаток, просто зосередившись на входах і виходах, не знаючи його внутрішньої реалізації коду.

Існує багато видів тестування чорного ящика, але найбільш важливими є наступні.

Функціональне тестування - цей тип тестування чорного ящика пов'язаний з функціональними вимогами системи; це роблять тестери програмного забезпечення.

Нефункціональне тестування. Цей тип тестування чорного ящика пов'язаний не з тестуванням конкретної функціональності, а з нефункціональними вимогами, такими як продуктивність, масштабованість, зручність використання.

Регресійне тестування - Регресійне тестування проводиться після того, як виправлення коду, оновлення або будь-яке інше обслуговування системи для

перевірки того, що новий код не торкнувся існуючий код. Веб-додаток було протестовано в браузері Google Chrome, версія 63.0.3239.84, 64-розрядна версія, в системі Windows 10 Education.

4.2 Тестування режиму реєстрації

Правильність роботи функції, що дозволяє реєстрацію користувача, була перевірена в тесті, проведеному за сценарієм, представленим у таблиці 4.1.

Таблиця 4.1 – Тестовий сценарій реєстрації

Користувач	Новий користувач
Ціль тесту	Створення нового облікового запису
Дії користувача	1. Запуск програми в браузері. 2. Перехід до вікна реєстрації. 3. Заповнення реєстраційної форми. 4. Натискання кнопки реєстрації.
Очікуваний результат	Створення облікового запису
Альтернативний результат	Помилка реєстрації через неправильні дані

Хід тесту виглядає наступним чином. Після запуску програми та переходу до вікна реєстрації відображалася реєстраційна форма, показана на рисунку 4.1.

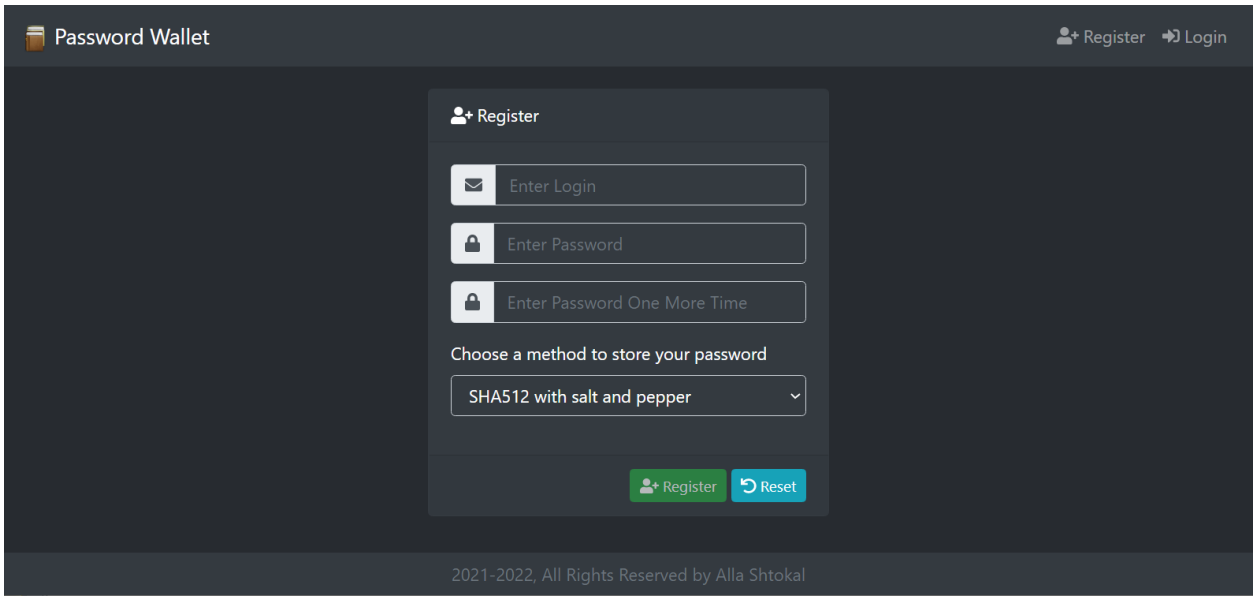

Рисунок 4.1 – Реєстрація користувача

На рисунку 4.2 показано процес реєстрації, який успішно завершено, і користувач одночасно входить на веб-сайт, тоді як на рисунку 4.3 показано ефект введення неправильних даних у форму реєстрації.

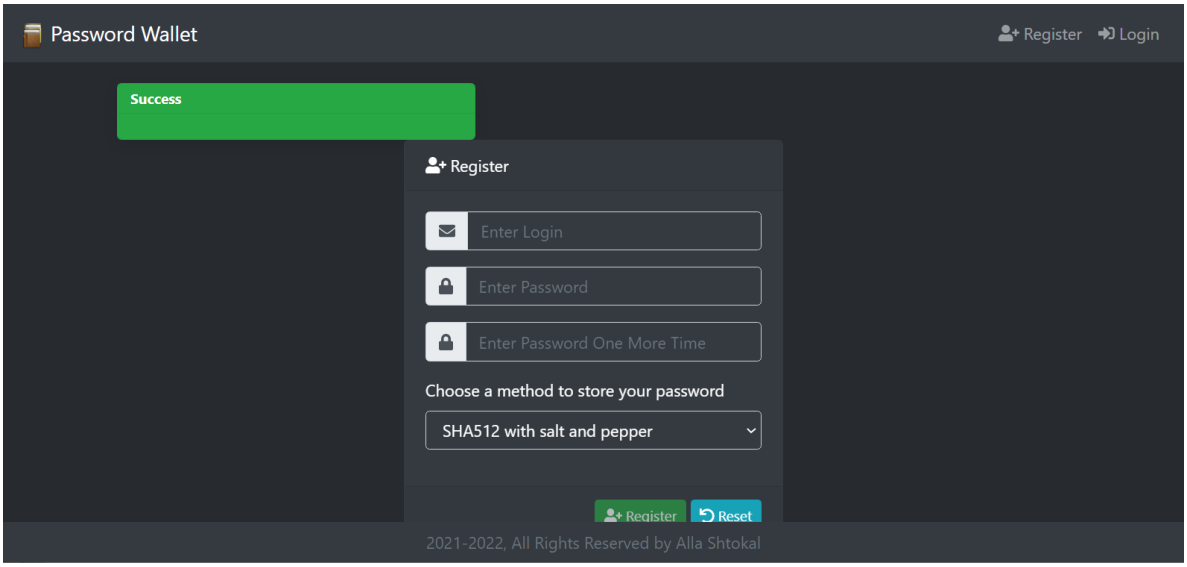

Рисунок 4.2 – Реєстрація користувача

4.3 Тестування режиму логування

Робота функції, що дозволяє користувачеві увійти на веб-сайт, була перевірена в тесті, проведеному за сценарієм, представленим у таблиці 4.2.

Таблиця 4.2 – Тестовий сценарій логування

Користувач	Користувач, що має акаунт в системі
Ціль тесту	Вхід до аплікації
Дії користувача	1. Запуск програми в браузері. 2. Перехід до вікна входу. 2. Заповнення форми входу. 3. Натиснення кнопки входу
Очікуваний результат	Успішний вхід на сайт
Альтернативний результат	Відображається повідомлення про надання неправдивих даних

Після запуску програми та переходу до вікна входу в систему відображалася форма входу, показана на рисунку 4.4.

Якщо будуть введені правильні дані, користувач увійде до системи та перейде у вікно головного меню, як показано на рисунку 4.5.

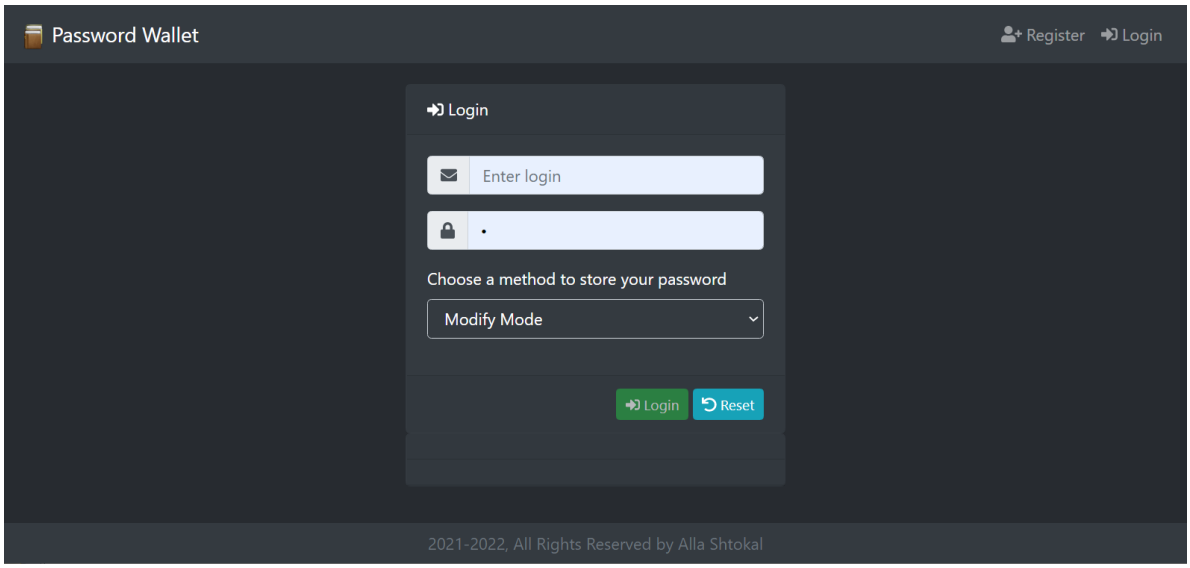

Рисунок 4.4 – Форма логування

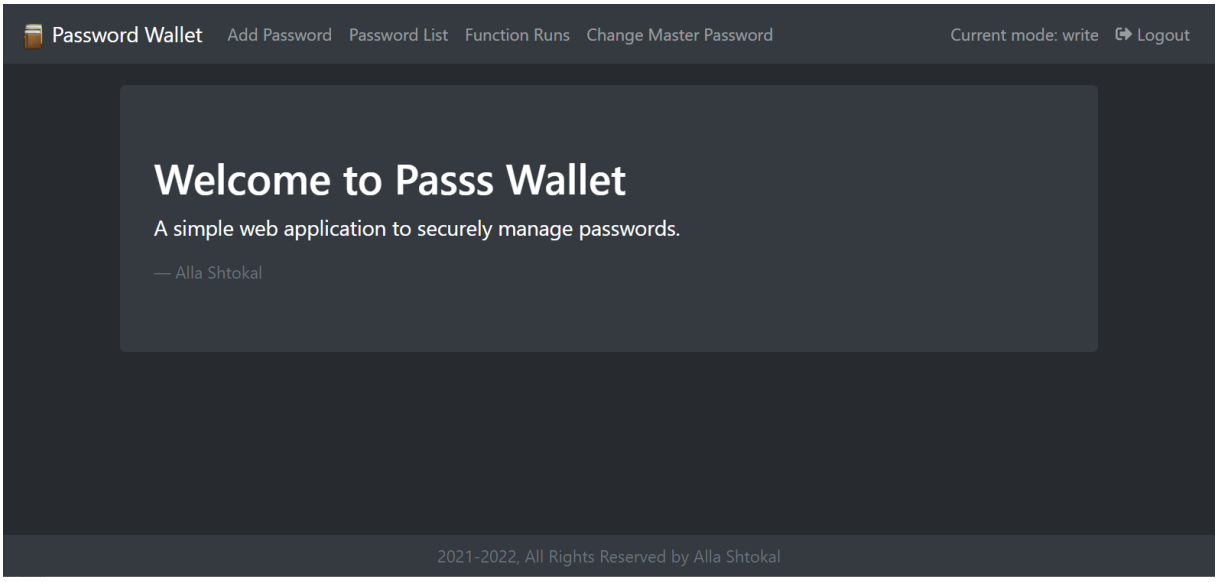

Рисунок 4.5 – Успішний вхід

Введення неправильних даних призведе до помилки входу та відображення відповідного повідомлення, яке було представлено на рисунку 4.6.

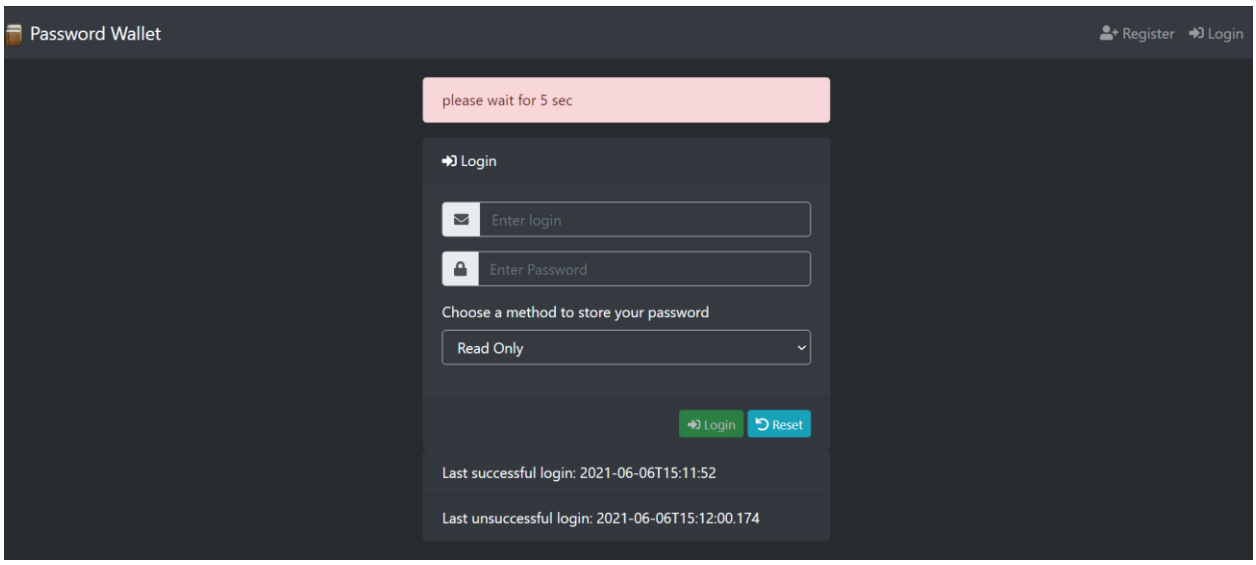

Рисунок 4.6 – Невдалий вхід

4.4 Тестування історії змін паролів

Наступною перевіреною функціональністю був механізм відображення історії змін паролів, а також можливість відкату до попереднього кроку. Це дозволяє у випадку небажаних змін повернутися до попереднього стану.

Ця функціональність була перевірена на основі сценарію, описаного в таблиці 4.3.

Таблиця 4.3 – Тестовий сценарій тестування історії зміни паролів

Користувач	Користувач, що увійшов до системи
Ціль тесту	Відображення історії змін та повернення до попередніх станів
Дії користувача	1. Перехід у вікно зі списокм паролів 2. Натиснення кнопки з переглядом історії для конкретного вибраного паролю 4. У новому вікні з історією натиснення іконки переходу до попереднього стану
Очікуваний результат	Успішний перехід до попереднього стану
Альтернативний результат	Відображення нового запису в історії змін про повернення всіх змін до вибраного стану

Залогований користувач повинен перейти у вікно зі списком усіх паролів, та обрати той, для якого хоче переглянути історію змін. Натиснувши на іконку поруч із відповідним рядком, рисунок 4.7.

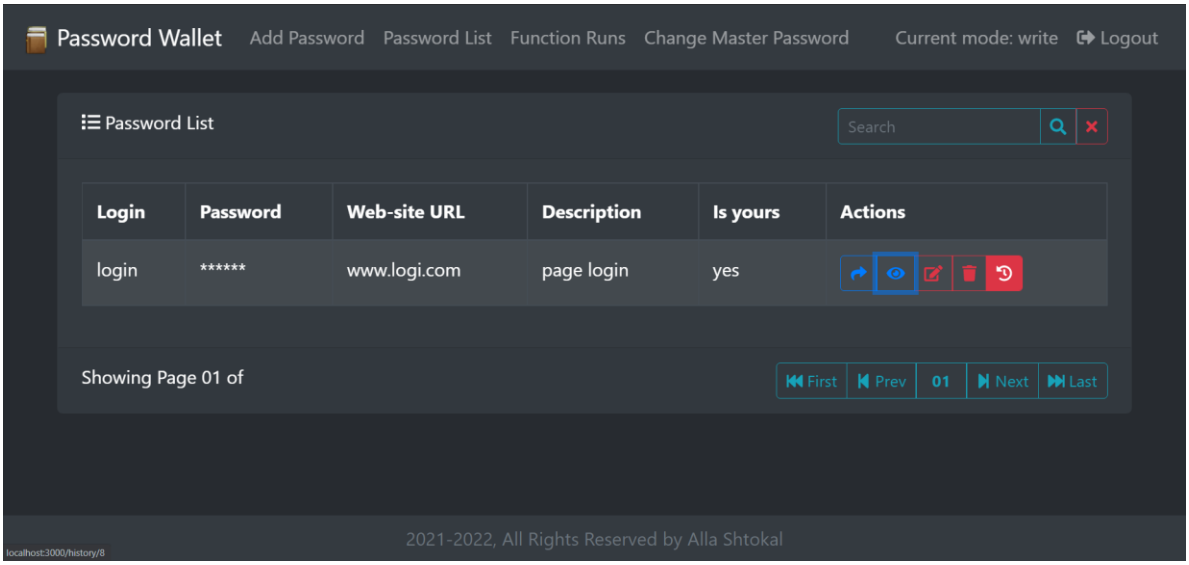

Рисунок 4.7 – Екран зі списком паролів

Після натиснення кнопки з іконкою «Історія» користувач потрапляє у нове вікно з історією. На рисунку 4.8 показано історію змін паролю.

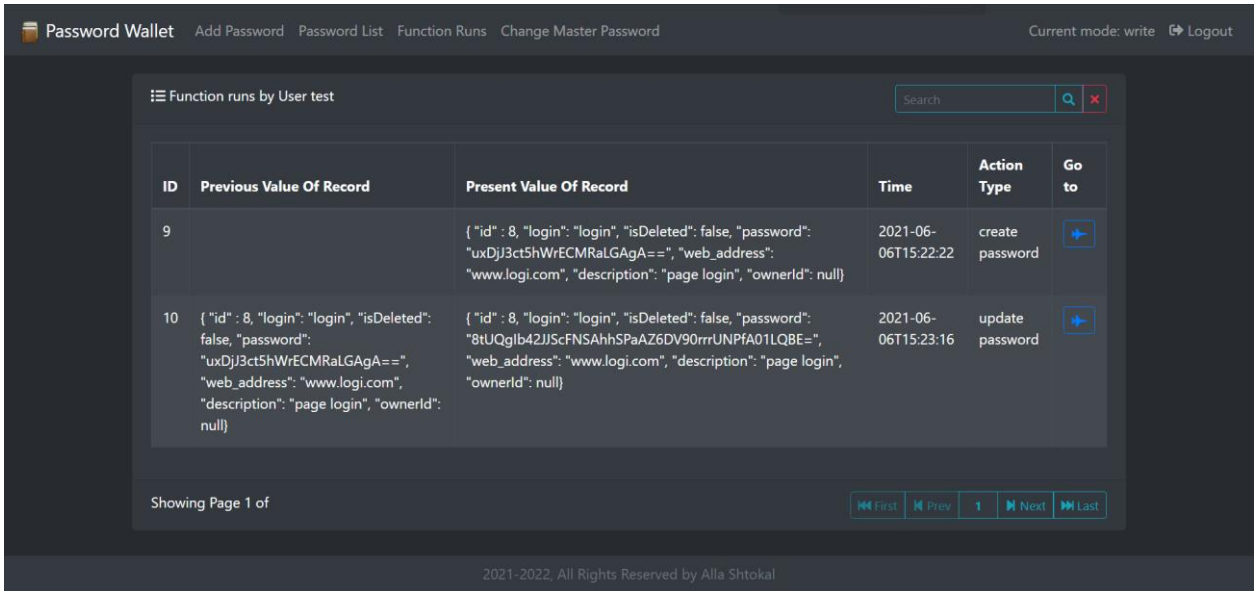

Рисунок 4.8 – Екран з відображенням історії змін паролю

Щоб повернутися до вибраного попереднього стану паролю потрібно натиснути на іконку в правому стовпці.

На рисунку 4.9 показано ефект вдалої спроби повернення до попередньої версії паролю.

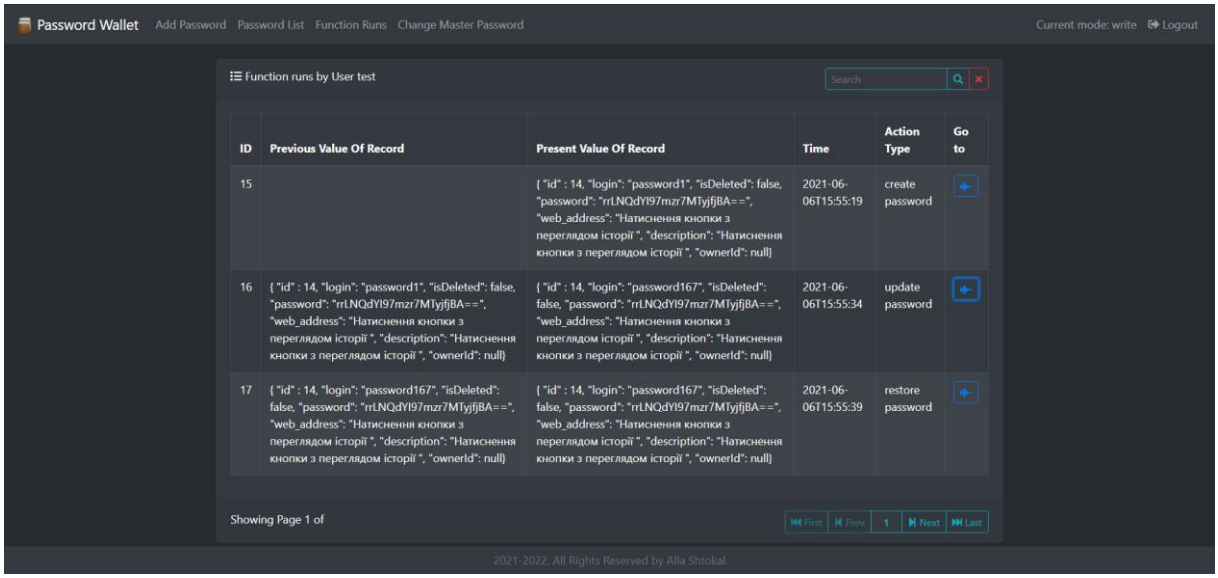

Рисунок 4.9 – Вдала спроба повернення до попереднього стану паролю

4.5 Тестування додавання паролів

Для того, щоб перевірити правильність функціональних можливостей додавання паролів, було проведено тест. Сценарій тесту подано у таблиці 4.4.

Таблиця 4.4 – Тестовий сценарій додавання паролів

Користувач	Користувач, що увійшов до системи
Ціль тесту	Додавання паролю
Дії користувача	1. Перехід до вікна зі списком паролів та груп. 2. Перехід до вікна додавання пароля. 3. Заповнення відповідних полів. 4. Натиснення кнопки, що додає пароль.
Очікуваний результат	Успішне додання паролю
Альтернативний результат	Відображення повідомлення про неправильні дані

Для того, щоб створити пароль, потрібно перейти у вікно, що містить список паролів, а потім вибирати опцію додавання пароля у верхній панелі. В результаті користувач перенаправляється у вікно, представлене на рисунку 4.10

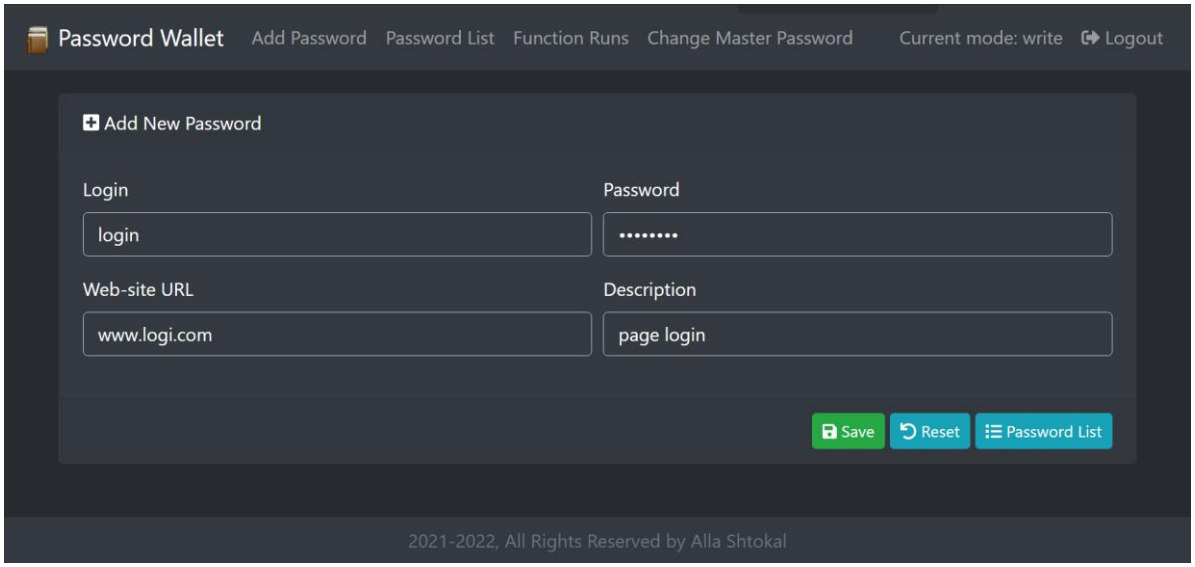

Рисунок 4.10 – Додавання паролю

Додаючи пароль, необхідно вказати його зміст та назву веб-сайту, якому він призначений. Адреса веб-сайту не є обов'язковою.

4.6 Розробка інструкції користувача

Інструкція користувача призначена для швидкого та лаконічного опису роботи з програмним продуктом. Інструкція дозволяє скоротити час ознайомлення користувача з системою, а також у разі потреби може допомогти вирішити конкретну проблему.

Користуватися системою можуть лише зареєстровані та авторизовані користувачі. На головному екрані належи створити новий акаунт або ж увійти вже з існуючого на відповідний рівень безпеки

Для того, щоб почати управління своїми паролями користувач може:

1. Перейти з домашньої сторінки на сторінку своїх паролів.
2. Додати новий пароль натиснувши кнопку «Додати пароль»

3. Після успішного додавання паролю його можна відредагувати, переглянути або видалити.

Усі дії проведені над паролями логуються.

4. Щоб переглянути історію змін потрібно перейти у головне меню і натиснути "Історія"

5. У вікні з історією можна повернутися до конкретного запису в історії записів..

6. Користувач, що не увійшов до другого рівня безпеки може лише переглядати паролі, але не може їх редагувати чи видаляти..

7. На першому рівні безпеки доступна можливість поділитися паролем з іншим користувачем.

Якщо якась дія доступна на першому рівні безпеки, то це означає, що вона так само доступна і на другому рівні безпеки.

Паролі можна додавати в групи. Можна переглянути список всіх паролів або паролів, що належать до конкретної групи.

Для пошуку потрібно на головній сторінці біля пошукової стрічки вибрати групу, за якою буде виконуватись пошук і ввести ключове слово.

Якщо користувач вже має додані паролі та виконував над нимиоперації усування, вносив зміни, то у правому верхньому куті буде відображатися кнопка «Історія». Після натискання на кнопку користувач потрапляє до вікна з відображеною історією змін, де може зробити відкат до вибраної ним зміни.

Таким чином, вся базова інформація буде надана користувачеві перед початком роботи з додатком.

4.7 Висновки

У четвертому розділі магістерської кваліфікаційної роботи розглянуто види та методики тестування, обрано методику тестування. Було проведено опис роботи головних сценаріїв веб-додатку. Додаток для кожного з розглянутих сценаріїв працює правильно. Розроблено інструкцію користувача.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Виконання кожної дослідницької роботи передбачає певні витрати. Зазвичай це витрати на створення та впровадження конкретного програмного забезпечення. Такі витрати необхідно постійно зменшувати, адже у цьому полягає прогрес будь-якого суспільства. Без такого прогресу ніяка науково-технічна розробка не буде реалізована на практиці, адже така розробка не буде ефективнішою за існуючі на ринку аналоги. На основі економічних розрахунків можна довести економічну доцільність та ефективність впровадження отриманих результатів виконаних науково-дослідних робіт у виробництво, тобто здійснити так звану комерціалізацію наукових розробок. У цілях проведення технологічного аудиту було залучено 2 незалежних експертіи. Такими експертами є Войтко Вікторія Володимирівна (к.т.н., доц. кафедри ПЗ ВНТУ), Хошаба Олександр Мирославович (к.т.н., доц. кафедри ПЗ ВНТУ).

Оцінка комерційного потенціалу розробки здійснюється за 5-ти бальною шкалою, до уваги взято 12 критеріїв. Критерії наведено в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки та їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна Конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне значне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні Незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Войтко В. В.	2. Хошаба О.М.
	Бали, виставлені експертами:	
1	3	4
2	3	2
3	2	3
4	4	4
5	2	3
6	3	4
7	4	3
8	4	2
9	3	3
10	2	3
11	4	4
12	2	2
Сума балів	$СБ_1 = 36$	$СБ_2 = 37$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 36{,}5$	

Отже, з отриманих даних таблиці 5.2 видно, що нова розробка має достатній рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Необхідні витрати для для розробки програмного продукту подано нижче. Основна заробітна плата для розробників визначається за формулою 5.1:

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р = 22$ дні;

t - число днів роботи розробника, t = 45 днів.

Остаточні значення заробітних плат для керівника і програміста наведені в таблиці 5.3.

Таблиця 5.3 – Розрахунки основної заробітної плати

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	30000	1363,63	5	6818,15
Frontend-програміст	25000	1136,36	20	22727,2
Всього:				29545,35

Таким чином, додаткова заробітна плата буде дорівнювати:

$$\text{Здод} = \ 0{,}1 \cdot 29545{,}35 = 2954{,}535 \ (\text{грн.})$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати за формулою 5.2:

$$\text{Нзп} = \ (\text{Зо} \ + \ \text{Зр}) \cdot \frac{\beta}{100}, \qquad (5.2)$$

$$\text{Нзп} = \ (29545{,}35 + 2954{,}535) \cdot \frac{22}{100} = 7149{,}97 \ (\text{грн.}).$$

Розрахунок амортизаційних витрат для ПЗ обчислюється за формулою 5.3, а результати занесено до таблиці 5.4:

$$\text{А} = \frac{\text{Ц} \cdot \text{Н}_\text{а}}{100} \cdot \frac{\text{Т}}{12}, \qquad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$\text{Н}_\text{а}$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.4 – Розрахунок амортизаційних відрахувань

Найменування	Ціна, грн.	Норма амортизації, %	Термін використання, м.	Сума амортизації
ПК + прилади керування	10000	20	3	500
Прилади маніпуляції ПК	1000	20	3	50
Всього	550			

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою 5.4, а результати занесемо до таблиці 5.5:

$$K = \sum_1^n \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і}, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.5 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	200	1	200
Фліпчарт	шт.	350	1	350
Маркери	шт.	25	4	100
Всього з урахуванням транспортних витрат				750

Витрати на силову електроенергію розраховуються за формулою 5.5:

$$\text{В}_\text{е} = \text{В} \cdot \text{П} \cdot \text{Ф} \cdot \text{К}_\text{п}\ ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=9,5 грн/кВт);

П – установлена потужність комп’ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=250 год.);

$\text{К}_\text{п}$ – коефіцієнт використання потужності ($\text{К}_\text{п} < 1$, $\text{К}_\text{п} = 0{,}8$).

$$Ве = 9{,}5 \cdot 0{,}6 \cdot 250 \cdot 0{,}8 = 1140 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто за формулою 5.6 маємо:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (29545{,}35 + 2954{,}535) = 32\,499{,}885 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 29545,35+2954,535+7149,97+550+750+1140+32 499,885 = 74 589,74 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою 5.7:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{74\,589{,}74}{1} = 74\,589{,}74$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою 5.8:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, отримуємо загальні витрати:

$$ЗВ = \frac{74\,589{,}74}{0{,}9} = 82\,877{,}48 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою 5.9:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшаться на 60 грн, а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 450 користувачів, протягом другого року – на 400 користувачів, протягом третього року – 350 користувачів.

Реалізація програмного продукту до впровадження результатів наукової розробки складала 50 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 100 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2020} = 60 \cdot 50 + (100 + 60) \cdot 450 = 75000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2021} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400) = 139000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2022} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400 + 350) = 195000 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою 5.10:

$$Е_{абс} = (ПП - PV), \quad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

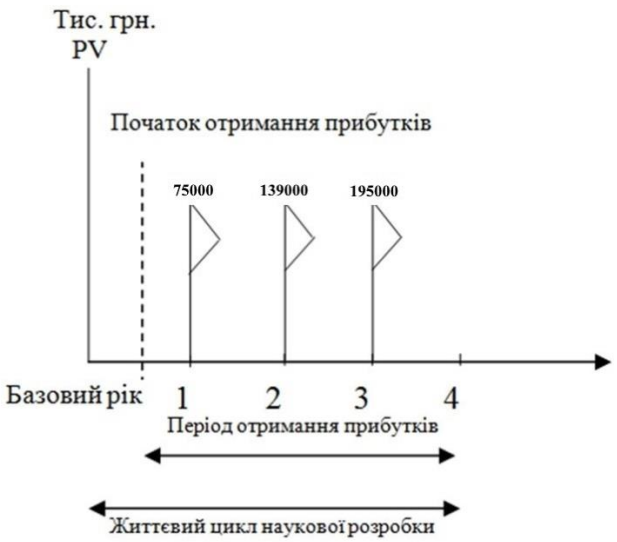

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою 5.11:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{82\,877{,}48}{(1+0{,}1)^0} + \frac{75000}{(1+0{,}1)^2} + \frac{139000}{(1+0{,}1)^3} + \frac{195000}{(1+0{,}1)^4} = 382481{,}32 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$\text{Е}_{\text{абс}} = 382481{,}32 - 82\,877{,}48 = 299\,603{,}84\text{грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\text{Е}_\text{в}$ за формулою 5.12:

$$\text{Е}_\text{в} = \sqrt[\text{Т}]{1 + \frac{\text{Е}_{\text{абс}}}{\text{PV}}} - 1 \qquad (5.12)$$

де $\text{Е}_{\text{абс}}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$\text{Т}_\text{ж}$ – життєвий цикл наукової розробки, роки.

Таким чином будемо мати:

$$\text{Е}_\text{в} = \sqrt[3]{1 + \frac{299\,603{,}84}{82\,877{,}48}} - 1 = 0{,}58 \text{ або } 58\ \%$$

Далі, розраховану величина $\text{Е}_\text{в}$ порівнюємо з мінімальною ставкою дисконтування $\tau_{\text{мін}}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{\text{мін}}$ визначається за формулою 5.13:

$$\tau = d + f, \qquad (5.13)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 58% > 30%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проєкту інвестицій. Термін окупності вкладених у реалізацію наукового проєкту інвестицій $T_{ок}$ розраховується за формулою 5.14:

$$T_{ок} = \frac{1}{0,44} = 2,27 \text{ року} \qquad (5.14)$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

Під час оцінки комерційного потенціалу розробки можна зробити висновок, що розробка має досить високий рівень комерційного потенціалу. В ході оцінки було розраховано витрати на виконання науково-дослідної та конструкторсько-технологічної роботи, а також показники збільшення чистого прибутку протягом трьох років.

Таким чином, обрахунок терміну окупності розробки показав, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи було розроблено функціонал ІТ-системи для управління паролями та їх шифрування.

Детальний аналіз предметної області показав, що проблема потребує вирішення. Було виконано аналіз відомих аналогів проекту. Було виявлено недоліки, які врахуються у магістерській кваліфікаційній роботі: можливість відновлення мастер-пароля, автоматичне перешифрування паролів з встановленою періодичністю, використання тимчасового паролю для входу, а також двофакторної автентифікації, забезпечення історії змін паролів і можливість відкату. Створено перелік завдань та план дій для розробки програмного забезпечення.

Розроблено загальну функціональну структурну модель, яка описує основні процеси системи.

Під час розробки було реалізовано:

– розроблено метод формування оцінки терміну щодо необхідності змінювати пароль на новий;

– розроблено метод визначення рівня складності паролю;

– розроблено модель системи управління паролями та їх шифрування;

– здійснено програмну реалізацію системи управління паролями та їх шифрування;

– спроектовано архітектуру бази даних веб-системи;

– налаштовано серверне оточення для розміщення системи управління паролями та ъх шифрування;

– розроблено серверну частину та програмні засоби системи, що містить API для роботи з клієнським додатком для веб.

Магістерську кваліфікаційну роботу було оформлено відповідно до вимог [23]. Розроблена система була протестована за допомогою технології «чорної скриньки». Тестування показало коректність відображення інформації та довело працездатність системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Менеджер паролів. URL: https://uk.wikipedia.org/wiki/Менеджер_паролів
2. Штокал А.С., Войтко В.В., Хошаба О.М. Тези доповідей науково-практичної конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2021)». URL: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/12975
3. Ю.А. Тарнавський Технології захисту інформації, 2018. 161 с.
4. Современный бэкенд для фронтенда на Node.js. URL: https://www.youtube.com/watch?v=XdtbRkN97go
5. Google passwords: https://passwords.google.com/
6. Dashlane. URL: https://www.dashlane.com/
7. Keeper. URL: https://www.keepersecurity.com/
8. 1Password. URL: https://1password.com/ru/
9. LastPass. URL: https://www.lastpass.com/
10. Niels Provos, David Mazières A Future-Adaptable Password Scheme. Paper - 1999 USENIX Annual Technical Conference, 1999.
11. Сіль (криптографія). URL: https://uk.wikipedia.org/wiki/Сіль_криптографія
12. Кристофер Дейт, Введение в системы баз данных, 1975.
13. Hash-based message authentication codes (HMAC). URL: https://cryptography.io/en/latest/hazmat/primitives/mac/hmac/
14. Node.js для фронтенд-разработчиков. URL: https://www.cat-in-web.ru/node-js-guide/
15. Сложность пароля. URL: https://dic.academic.ru/dic.nsf/ruwiki/1629819
16. С. Нікіфоров Методы защиты информации. Пароли, скрытие, шифрование, 2016. 124 с.
17. Кей С. Хорстманн, Гари Корнелл "Java. Библиотека профессионала", 2018. 310 с.
18. Google Material Design. URL: https://material.io/design.

19. CLI command reference. URL: https://docs.nestjs.com/cli/usages
20. Забезпечення безпеки контейнеризованого додатку Node.js за допомогою Nginx, Let's Encrypt і Docker Compose. URL: https://www.digitalocean.com/community/tutorials/how-to-secure-a-containerized-node-js-application-with-nginx-let-s-encrypt-and-docker-compose-ru
21. Степанченко И.В. Методы тестирования программного обеспечения : учебное пособие. Волгоград, 2006. 74с.
22. Бейзер Б. Тестирование черного ящика. Питер, 2004. 365 с.
23. О. Н. Романюк, Р. Р. Обертюх, Т. О. Савчук, Л. П. Громова. Положення про кваліфікаційну роботу у Вінницькому національному технічному університеті. Вінниця, 2015. 27 с.

ДОДАТКИ

Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
17 лютого 2021 року

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів та програмного додатку управління паролями і їх шифрування»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доцент Хошаба О.М.
18 лютого 2021 року
Виконала:
студентка гр. 1ПІ-19м Штокал А.С.
18 лютого 2021 року

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів та програмного додатку управління паролями і їх шифрування».

Галузь застосування - навчальна сфера.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ від 9 березня 2021 року № 65 ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності управління паролями та їх шифрування за рахунок методу оцінювання складності паролю, що дає можливість підвищити безпеку системи збереження паролів.

Призначення роботи – розробка методів і програмних засобів для управління паролями та їх шифрування.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Ю.А. Тарнавський Технології захисту інформації, 2018. 161 с.
2. Niels Provos, David Mazières (June 1999). A Future-Adaptable Password Scheme. Paper - 1999 USENIX Annual Technical Conference
3. Google Material Design. URL: https://material.io/design
4. Современный бэкенд для фронтенда на Node.js. URL: https://www.youtube.com/watch?v=XdtbRkN97go
5. Node.js для фронтенд-разработчиков. URL: https://www.cat-in-web.ru/node-js-guide/

4. Технічні вимоги

Оскільки система складається з 2-х частин (клієнт, сервер), розглянемо вимоги до кожної з них.

1. Клієнт. Оскільки клієнт виконаний в форматі сайту і не має важких обчислень на стороні клієнта, варто звертати увагу лише на вимоги браузера. Розглянемо мінімльні системні характеристики необхідні для роботи з сайтом: Windows 7, Windows 8, Windows 8.1, Windows 10 або пізнішої версії. Процесор Intel Pentium 4 або більш пізніша версія з підтримкою SSE3.
2. Сервер. Мінімальні вимоги: 1 VCPU, 2 GB RAM, 20 GB DISK LOCAL
3. DOCKER. Серверна частина працює на ОС Linux засобами Docker. Якщо маються на увазі апаратні вимоги, то докер сам по собі не має мінімальних апаратних вимог. Це лише утиліта для віртуалізації. Мінімальні системні вимоги залежать від програми, яка буде розвернута в контейнерах. З апаратних вимог лише одна – підтримка віртуалізації. Що стосується системних (програмних) вимог, то для систем на базі ОС Linux потрібно лише: бітність 64, ядро не старіше 3.10.
4. NODE. Якщо маються на увазі апаратні вимоги - все залежить від стороніх бібліотек які використовуються в самому додатку. Розроблюваний додаток не має важких залежностей.

5. Конструктивні вимоги.

Система повинна бути зручною у використанні та обслуговуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до МКР;

- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз, вибір та обґрунтування актуальності розробки	20.02.2021– 8.02.21
2	Аналіз існуючих аналогів	01.03.20 – 12.03.21
3	Розробка методу і моделей системи управління паролями	13.03.21 – 22.03.21
4	Розробка макетів графічного інтерфейсу	23.03.21 – 31.03.21
5	Програмна реалізація системи	01.04.21 – 25.04.21
6	Тестування роботи системи	26.04.21 – 30.04.21
7	Економічна частина	01.05.21 – 16.05.21
8	Оформлення матеріалів до захисту МДР	17.05.21– 31.05.21

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Лістинг алгоритму шифрування SHA512

```
package com.shtokal.passs.config;
import org.apache.tomcat.util.codec.binary.Base64;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Random;
public class SHAAlgorithm {
    public static String calculateHashSHA512(String pepper, String salt, String text) {
        return calculateSHA512(pepper + salt + text);
    }
    private static String calculateSHA512(String text) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-512");
            byte[] messageDigest = md.digest(text.getBytes());
            BigInteger no = new BigInteger(1, messageDigest);
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }    }
    public static String generateSalt() {
        final Random r = new SecureRandom();
        byte[] salt = new byte[32];
        r.nextBytes(salt);
        return Base64.encodeBase64String(salt);   }}
```

Додаток В – Лістинг алгоритму шифрування HMAC

```
package com.shtokal.passs.config;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
public class HMACAlgorithm {
    private static final String HMAC_SHA512 = "HmacSHA512";
    public static String calculateHMAC(String text, String key) {
        Mac sha512Hmac;
        String result = "";
        try {
            final byte[] byteKey = key.getBytes(StandardCharsets.UTF_8);
            sha512Hmac = Mac.getInstance(HMAC_SHA512);
            SecretKeySpec keySpec = new SecretKeySpec(byteKey, HMAC_SHA512);
            sha512Hmac.init(keySpec);
            byte[] macData = sha512Hmac.doFinal(text.getBytes(StandardCharsets.UTF_8));
            // Can either base64 encode or put it right into hex
            result = java.util.Base64.getEncoder().encodeToString(macData);
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
        }
        return result;
    }
}
```

```
package com.shtokal.passs.config;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
public class PassAlgor {
   private static final String ALGO = "AES";
   public static Key generateKey(String password) throws Exception {
      return new SecretKeySpec(calculateMD5(password), ALGO);
   }
   public static String encrypt(String data, Key key) throws Exception {
      Cipher c = Cipher.getInstance(ALGO);
      c.init(Cipher.ENCRYPT_MODE, key);
      byte[] encVal = c.doFinal(data.getBytes());
      return Base64.getEncoder().encodeToString(encVal);
   }
   public static String decrypt(String encryptedData, Key key) throws Exception {
      Cipher c = Cipher.getInstance(ALGO);
      c.init(Cipher.DECRYPT_MODE, key);
      byte[] decodedValue = Base64.getDecoder().decode(encryptedData);
      byte[] decValue = c.doFinal(decodedValue);
      return new String(decValue);
   }
   public static byte[] calculateMD5(String text) {
      try {
         MessageDigest md = MessageDigest.getInstance("MD5");
         byte[] messageDigest = md.digest(text.getBytes());
         return messageDigest;
      } catch (NoSuchAlgorithmException e) {
         throw new RuntimeException(e);      }    }}
```

Додаток Г – Лістинг команд SQL для створення бази даних

```
CREATE TABLE action_type (
  id SERIAL NOT NULL,
  title varchar(30) NOT NULL,
  description text,
  PRIMARY KEY (id)
);
CREATE TABLE data_change (
  id SERIAL NOT NULL,
  id_user int4 NOT NULL,
  id_message_record int4 NOT NULL,
  previous_value_of_record text,
  present_value_of_record text,
  time timestamp NOT NULL,
  id_action_type int4 NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE "function" (
  id SERIAL NOT NULL,
  function_name varchar(30) NOT NULL,
  description text,
  PRIMARY KEY (id)
);
CREATE TABLE function_run (
  id SERIAL NOT NULL,
  3 id_user int4 NOT NULL,
  time timestamp NOT NULL,
  id_function int4 NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE password (
  id SERIAL NOT NULL,
  password text NOT NULL,
  deleted bool DEFAULT 'false' NOT NULL,
  id_user int4 NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE "user" (
  id SERIAL NOT NULL,
  login varchar(30) NOT NULL UNIQUE,
  pass_hash varchar(128) NOT NULL,
  salt varchar(32),
  deleted bool DEFAULT 'false' NOT NULL,
  PRIMARY KEY (id)
);
ALTER TABLE
  data_change
ADD
  CONSTRAINT FKdata_chang739617 FOREIGN KEY (id_message_record) REFERENCES
password (id) ON UPDATE Cascade ON DELETE Restrict;
```

```
ALTER TABLE
  data_change
ADD
  CONSTRAINT FKdata_chang567764 FOREIGN KEY (id_user) REFERENCES "user" (id) ON
UPDATE Cascade ON DELETE Restrict;
ALTER TABLE
  data_change
ADD
  CONSTRAINT FKdata_chang126695 FOREIGN KEY (id_action_type) REFERENCES
action_type (id) ON UPDATE Cascade ON DELETE Restrict;
ALTER TABLE
  function_run
ADD
  CONSTRAINT FKfunction_r736743 FOREIGN KEY (id_function) REFERENCES "function"
(id) ON UPDATE Cascade ON DELETE Restrict;
ALTER TABLE
  function_run
ADD
  CONSTRAINT FKfunction_r609630 FOREIGN KEY (id_user) REFERENCES "user" (id) ON
UPDATE Cascade ON DELETE Cascade;
ALTER TABLE
  password
ADD
  CONSTRAINT FKpassword525405 FOREIGN KEY (id_user) REFERENCES "user" (id) ON
UPDATE Cascade ON DELETE Cascade;
```

Додаток Ґ – Лістинг класів, що описують моделі в додатку

```
package com.shtokal.passs.model;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;
import static javax.persistence.GenerationType.AUTO;
@Entity
@Table(name = "password")
@Getter
@Setter
public class Password {
    @Id
    @GeneratedValue(strategy = AUTO)
    @Column(name = "id")
    private Long id;
    @Column(name="login")
    private String login;
    @Column(name="is_deleted")
    private Boolean isDeleted = false;
    @Column(name="password")
    private String password;
    @Column(name="web_address")
    private String web_address;
    @Column(name="description")
    private String description;
    @Column(name="owner_id")
    private Long ownerId;
    @ManyToOne
    @JoinColumn(name="user_id", nullable=false)
```

```
    private User user;
    @OneToMany(mappedBy = "user")
    private Set<DataChange> dataChanges = new HashSet<>();

    public void addDataChange(DataChange dataChange) {
        this.dataChanges.add(dataChange);
        dataChange.setPassword(this);
    }

    @Override
    public String toString() {
        return "{" +
                " \"id\" : " + id  +
                ", \"login\": \"" + login + '\"' +
                ", \"isDeleted\": " + isDeleted +
                ", \"password\": \"" + password + '\"' +
                ", \"web_address\": \"" + web_address + '\"' +
                ", \"description\": \"" + description + '\"' +
                ", \"ownerId\": " + ownerId +
                '}';
    }
}

package com.shtokal.passs.model;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;
import static javax.persistence.GenerationType.AUTO;

@Entity
```

```
@Table(name = "user")
@Getter
@Setter
public class User {
    @Id
    @GeneratedValue(strategy = AUTO)
    @Column(name = "id")
    private Long id;

    @Column(name = "login")
    private String login;

    @Column(name = "password_hash")
    private String password_hash;

    @Column(name = "salt")
    private String salt;

    @Column(name = "isPasswordKeptAsHash")
    private Boolean isPasswordKeptAsHash;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "user_roles",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();

    @OneToMany(mappedBy = "user")
    private Set<Password> passwords = new HashSet<>();

    @OneToMany(mappedBy = "user")
    private Set<DataChange> dataChanges = new HashSet<>();
```

```
    @OneToMany(mappedBy = "user")
    private Set<FunctionRun> functionRuns = new HashSet<>();

    public void addPassword(Password password) {
        this.passwords.add(password);
        password.setUser(this);
    }

    public void addDataChange(DataChange dataChange) {
        this.dataChanges.add(dataChange);
        dataChange.setUser(this);
    }

    public void addFunctionRun(FunctionRun functionRun) {
        this.functionRuns.add(functionRun);
        functionRun.setUser(this);
    }
}

package com.shtokal.passs.model;

import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
import java.time.LocalDateTime;
import static javax.persistence.GenerationType.AUTO;

@Entity
@Table(name = "log")
@Getter
```

```
@Setter
public class Log {

    @Id
    @GeneratedValue(strategy = AUTO)
    @Column(name = "id")
    private Long id;

    @Column(name = "attempt")
    private int attempt;

    @Column(name = "ip_address")
    private String ipAddress;

    @Column(name = "login")
    private String login;

    @Column(name = "time")
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime time;

}
package com.shtokal.passs.model;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Getter;
import lombok.Setter;
import org.hibernate.annotations.GenericGenerator;
import javax.persistence.*;
import java.time.LocalDateTime;
@Entity
@Table(name = "function_run")
@Getter
```

```
@Setter
public class FunctionRun {
    @Id
    @GeneratedValue(
            strategy= GenerationType.AUTO,
            generator="native"
    )
    @GenericGenerator(
            name = "native",
            strategy = "native"
    )
    @Column(name = "id")
    private Long id;
    @ManyToOne
    @JoinColumn(name="user_id", nullable=false)
    private User user;
    @Column(name = "time")
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime time;
    @Column(name = "functionName")
    private String functionName;
}


package com.shtokal.passs.model;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
import java.time.LocalDateTime;
import static javax.persistence.GenerationType.AUTO;
@Entity
@Table(name = "data_change")
```

```
@Getter
@Setter
public class DataChange {
    @Id
    @GeneratedValue(strategy = AUTO)
    @Column(name = "id")
    private Long id;
    @ManyToOne
    @JoinColumn(name="user_id", nullable=false)
    private User user;
    @ManyToOne
    @JoinColumn(name="password_id", nullable=false)
    private Password password;
    @Column(name="previous_value_of_record")
    private String previousValueOfRecord ;
    @Column(name="present_value_of_record")
    private String presentValueOfRecord ;
    @Column(name = "time")
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime time;
    @Column(name="action_type")
    private String actionType;}
```

Додаток Д – Лістинг класів-контролерів

```
package com.shtokal.passs.controller;
import com.shtokal.passs.dto.*;
import com.shtokal.passs.model.Log;
import com.shtokal.passs.repository.LogRepository;
import com.shtokal.passs.security.JwtUtils;
import com.shtokal.passs.service.LogService;
import com.shtokal.passs.service.UserService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.web.bind.annotation.*;
import javax.servlet.http.HttpServletRequest;
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/api/v1/user")
public class UserController {
   private final UserService userService;
   private final LogService logService;
   public UserController(UserService userService,
                         AuthenticationManager authenticationManager,
                         JwtUtils jwtUtils, LogService logService) {
      this.userService = userService;
      this.logService = logService;
   }
   @PostMapping(value = "/add")
   public ResponseEntity<UserResponse> save(@RequestBody UserDTORegister
userDTORegister) {
      if (userDTORegister == null) {
         return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
      }
      if (userService.existsByLogin(userDTORegister.getLogin())) {
         return ResponseEntity
```

```
                .badRequest()
                .body(new UserResponse());
        }
        HttpHeaders headers = new HttpHeaders();
        UserResponse add = userService.add(userDTORegister);
        return new ResponseEntity<>(add, headers, HttpStatus.CREATED);
    }
    @PostMapping(value = "/login")
    public ResponseEntity<LoginResponse> login(@RequestBody UserDTO userDTO,
                                               HttpServletRequest request) {
        if (userDTO == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        if (!userService.existsByLogin(userDTO.getLogin())) return new
ResponseEntity<>(HttpStatus.BAD_REQUEST);
        HttpHeaders headers = new HttpHeaders();
        String remoteAddr = request.getRemoteAddr();
        Boolean result = userService.login(userDTO);
        Integer status = logService.getLoginStatus(userDTO.getLogin(), result, remoteAddr);
        LoginResponse loginResponse = new LoginResponse();
        loginResponse.setStatus(status.toString());
        Log lastSuccessful = logService.findLastSuccessful(remoteAddr,
userDTO.getLogin());
        Log lastUnSuccessful = logService.findLastUnSuccessful(remoteAddr,
userDTO.getLogin());
        if(lastSuccessful!=null)
        loginResponse.setLastSuccess(lastSuccessful.getTime() );
        else  loginResponse.setLastSuccess(null);
        if(lastUnSuccessful!=null)
            loginResponse.setLastUnsuccessful(lastUnSuccessful.getTime() );
        else  loginResponse.setLastUnsuccessful(null);
        return ResponseEntity.ok(loginResponse);
    }
    @PostMapping(value = "/reset")
    public ResponseEntity<Boolean> resetIp(HttpServletRequest request) {
```

```
            String remoteAddr = request.getRemoteAddr();
            logService.resetIp(remoteAddr);
            return ResponseEntity.ok(true);
        }
        @PostMapping(value = "/change")
        public ResponseEntity<Boolean> changePassword(@RequestBody
ChangeUserPasswordRequest uRequest) throws Exception {
            HttpHeaders headers = new HttpHeaders();
            if (uRequest == null) {
                return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
            }
            Boolean changeResult = this.userService.changePassword(uRequest);
            return new ResponseEntity<>(changeResult, headers, HttpStatus.OK);
        }
    }

package com.shtokal.passs.controller;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.shtokal.passs.dto.*;
import com.shtokal.passs.service.PasswordService;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/api/v1/password")
public class PasswordController {
   private final PasswordService passwordService;
   public PasswordController(PasswordService passwordService) {
      this.passwordService = passwordService;
   }
```

```
    @PostMapping(value = "/add")
    public ResponseEntity<PasswordRequest> save(@RequestBody AddPasswordRequest passwordRequest) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (passwordRequest == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        PasswordRequest add = this.passwordService.add(passwordRequest);
        return new ResponseEntity<>(add, headers, HttpStatus.CREATED);
    }
    @PostMapping(value = "/share")
    public ResponseEntity<Boolean> share(@RequestBody SharePassRequest sharePassRequest) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (sharePassRequest == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        Boolean add = passwordService.share(sharePassRequest);
        return new ResponseEntity<>(add, headers, HttpStatus.CREATED);
    }
    @PutMapping(value = "/edit")
    public ResponseEntity<Boolean> edit(@RequestBody EditRequest editPassRequest) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (editPassRequest == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        Boolean add = passwordService.edit(editPassRequest);
        return new ResponseEntity<>(add, headers, HttpStatus.CREATED);
    }
    @PutMapping(value = "/delete")
    public ResponseEntity<Boolean> delete(@RequestParam("passwordId") String passwordId,
                                          @RequestParam("userLogin")  String userLogin  ) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (passwordId == null || userLogin == null) {
```

```
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        Boolean deleted = passwordService.delete(passwordId, userLogin);
        return new ResponseEntity<>(deleted, headers, HttpStatus.OK);
    }
    @PostMapping(value = "/show")
    public ResponseEntity<String> showPassword(@RequestBody ShowPasswordRequest showPasswordRequest) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (showPasswordRequest == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        String password = passwordService.showPassword(showPasswordRequest);
        return new ResponseEntity<>(password, headers, HttpStatus.OK);
    }
    @GetMapping(value = "/allbylogin")
    public ResponseEntity<PasswordResponse> getAllByLogin(@RequestParam String login,
                                                          @RequestParam String pageNumber,
                                                          @RequestParam String pageSize) {
        HttpHeaders headers = new HttpHeaders();
        if (login == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        Pageable pageable = PageRequest.of(Integer.parseInt(pageNumber), Integer.parseInt(pageSize));
        PasswordResponse resp = passwordService.findAllByUserLogin(login, pageable);
        return new ResponseEntity<>(resp, headers, HttpStatus.OK);
    }
    @GetMapping(value = "/id")
    public ResponseEntity<OnePasswordResponse> getPasswordById(@RequestParam String id,
                                                               @RequestParam String login ) throws Exception {
        HttpHeaders headers = new HttpHeaders();
        if (id == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
```

```
        OnePasswordResponse resp = passwordService.fidPasswordById(id, login);
        return new ResponseEntity<>(resp, headers, HttpStatus.OK);
    }
    @PutMapping(value = "/restore")
    public ResponseEntity<Boolean> restore(@RequestParam("changeId") String changeId,
                                           @RequestParam("login") String login
    ) throws JsonProcessingException {

        HttpHeaders headers = new HttpHeaders();
        if (changeId == null || login == null  ) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        Boolean restored = passwordService.restore(changeId, login);
        return new ResponseEntity<>(restored, headers, HttpStatus.OK);
    }}
package com.shtokal.passs.controller;
import com.shtokal.passs.dto.DataChangeResponse;
import com.shtokal.passs.dto.FunctionRunResponse;
import com.shtokal.passs.model.DataChange;
import com.shtokal.passs.model.FunctionRun;
import com.shtokal.passs.repository.FunctionRunRepository;
import org.modelmapper.ModelMapper;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.List;
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/api/v1/function")
public class FunctionRunController {
    private final FunctionRunRepository functionRunRepository;
    private final ModelMapper modelMapper;
```

```
    public FunctionRunController(FunctionRunRepository functionRunRepository, ModelMapper modelMapper) {

        this.functionRunRepository = functionRunRepository;
        this.modelMapper = modelMapper;
    }
    @GetMapping(value = "/all")
    public ResponseEntity<List<FunctionRunResponse>> getAllByLogin(@RequestParam("login") String login) {
        if (login == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        HttpHeaders headers = new HttpHeaders();
        List<FunctionRun> functionRuns = functionRunRepository.findAllByUser_Login(login);
        List<FunctionRunResponse> functionRunResponses = new ArrayList<>();
        for (FunctionRun functionRun : functionRuns) {
            FunctionRunResponse map = modelMapper.map(functionRun, FunctionRunResponse.class);
            functionRunResponses.add(map);
        }
        return new ResponseEntity<>(functionRunResponses, headers, HttpStatus.OK);
    }}
package com.shtokal.passs.controller;
import com.shtokal.passs.dto.DataChangeResponse;
import com.shtokal.passs.service.DataChangeService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/api/v1/datachange")
public class DataChangeController {
    private final DataChangeService dataChangeService;
```

```
    public DataChangeController(DataChangeService dataChangeService) {
        this.dataChangeService = dataChangeService;
    }
    @GetMapping(value = "/all")
    public  ResponseEntity<List<DataChangeResponse>>  getAllByLogin(@RequestParam("login")
String login,
                                                    @RequestParam("passwordId") String passwordId){
        if (login == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
        HttpHeaders headers = new HttpHeaders();
        List<DataChangeResponse>                                                dataChangeResponses=
dataChangeService.findAllByUserLoginAndPasswordId(login,passwordId);
        return new ResponseEntity<>(dataChangeResponses, headers, HttpStatus.OK);
    }}
```

Додаток Е – Лістинг сервісного класу для роботи з паролем

```
package com.shtokal.passs.service;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.shtokal.passs.config.PassAlgor;
import com.shtokal.passs.dto.*;
import com.shtokal.passs.model.DataChange;
import com.shtokal.passs.model.FunctionRun;
import com.shtokal.passs.model.Password;
import com.shtokal.passs.model.User;
import com.shtokal.passs.repository.DataChangeRepository;
import com.shtokal.passs.repository.FunctionRunRepository;
import com.shtokal.passs.repository.PasswordRepository;
import com.shtokal.passs.repository.UserRepository;
import org.modelmapper.ModelMapper;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import java.security.Key;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
@Service
public class PasswordServiceImplementation implements PasswordService {
    private final UserRepository userService;
    private final  PasswordRepository passwordRepository;
    private final ModelMapper modelMapper;
    private final  DataChangeRepository dataChangeRepository;
    private final FunctionRunRepository functionRunRepository;
    public PasswordServiceImplementation(ModelMapper modelMapper,
                                         UserRepository userService,
                                         PasswordRepository passwordRepository,
```

```
                        DataChangeRepository dataChangeRepository,
                        FunctionRunRepository functionRunRepository) {
        this.modelMapper = modelMapper;
        this.userService = userService;
        this.passwordRepository = passwordRepository;
        this.dataChangeRepository = dataChangeRepository;
        this.functionRunRepository = functionRunRepository;
    }
    @Override
    public PasswordResponse findAllByUserLogin(String login, Pageable pageable) {
        Integer totalElements =
passwordRepository.findAllByUser_LoginAndIsDeletedIsFalse(login).size();
        Integer totalPages;
        if (totalElements % pageable.getPageSize() == 0)
            totalPages = totalElements / pageable.getPageSize();
        else totalPages = totalElements / pageable.getPageSize() + 1;
        List<Password> allByUser_login =
passwordRepository.findAllByUser_LoginAndIsDeletedIsFalse(login, pageable);
        PasswordResponse passwordRespons = new PasswordResponse();
        List<PasswordsContent> content = new ArrayList<>();
        for (Password pass : allByUser_login) {
            PasswordsContent map = modelMapper.map(pass, PasswordsContent.class);
            content.add(map);
        }
        passwordRespons.setContent(content);
        passwordRespons.setTatalElements(totalElements.toString());
        passwordRespons.setTatalPages(totalPages.toString());
        passwordRespons.setNumber(String.valueOf(pageable.getPageNumber()));
        return passwordRespons;
    }
    @Override
    public String showPassword(ShowPasswordRequest showPasswordRequest) throws Exception {
```

```
        Password password =
passwordRepository.findById(Long.parseLong(showPasswordRequest.getPasswordId())).get();
        User byLogin = userService.findByLogin(password.getUser().getLogin());
        //FunctionRun
        FunctionRun functionRun = new FunctionRun();
        functionRun.setFunctionName("view password");
        functionRun.setTime(LocalDateTime.now());
        byLogin.addFunctionRun(functionRun);
        functionRunRepository.save(functionRun);
        String password_hash = password.getUser().getPassword_hash();
        Key key = PassAlgor.generateKey(password_hash);
        return PassAlgor.decrypt(password.getPassword(), key);
    }
    @Override
    public Boolean changeAllUsersPasswords(String login, String oldMasterPass, String
NewMasterPass) throws Exception {
        List<Password> allByUser_login = passwordRepository.findAllByUser_Login(login);
        for (Password p : allByUser_login) {
            String decrypt = PassAlgor.decrypt(p.getPassword(),
PassAlgor.generateKey(oldMasterPass));
            p.setPassword(PassAlgor.encrypt(decrypt, PassAlgor.generateKey(NewMasterPass)));
            passwordRepository.save(p);
        }
        return true;
    }
    @Override
    public PasswordRequest add(AddPasswordRequest addPasswordRequest) throws Exception {
        User userByLogin = userService.findByLogin(addPasswordRequest.getUserLogin());
        Password password = new Password();
        password.setLogin(addPasswordRequest.getLogin());
        Key key = PassAlgor.generateKey(userByLogin.getPassword_hash());
        password.setPassword(PassAlgor.encrypt(addPasswordRequest.getPassword(), key));
```

```
        password.setWeb_address(addPasswordRequest.getWeb_address());
        password.setDescription(addPasswordRequest.getDescription());
        password.setOwnerId(null);
        userByLogin.addPassword(password);
        passwordRepository.save(password);
        DataChange dataChange = new DataChange();
        dataChange.setPreviousValueOfRecord(null);
        dataChange.setPresentValueOfRecord(password.toString());
        dataChange.setTime(LocalDateTime.now());
        dataChange.setActionType("create password");
        userByLogin.addDataChange(dataChange);
        password.addDataChange(dataChange);
        dataChangeRepository.save(dataChange);
        FunctionRun functionRun = new FunctionRun();
        functionRun.setFunctionName("create password");
        functionRun.setTime(LocalDateTime.now());
        userByLogin.addFunctionRun(functionRun);
        functionRunRepository.save(functionRun);
       return modelMapper.map(addPasswordRequest, PasswordRequest.class);
    }
    @Override
    public Boolean share(SharePassRequest sharePassRequest) throws Exception {
        User ownerByLogin = userService.findByLogin(sharePassRequest.getUserLogin());
        User sharedUserByLogin = userService.findByLogin(sharePassRequest.getSharedLogin());
        Password passwordById =
passwordRepository.findById(Long.parseLong(sharePassRequest.getPasswordId())).get();
        Password firstByUser_loginAndIsOwnerTrue =
passwordRepository.findByUser_LoginAndIdAndOwnerIdIsNull(sharePassRequest.getUserLogin(
),
  Long.valueOf(sharePassRequest.getPasswordId()));
        if (firstByUser_loginAndIsOwnerTrue != null) {
            //робимо новий запис до бази
            Password password = new Password();
```

```
            password.setOwnerId(passwordById.getId());
            password.setLogin(passwordById.getLogin());
            password.setDescription(passwordById.getDescription());
            password.setWeb_address(passwordById.getWeb_address());
            String password_hash = ownerByLogin.getPassword_hash();
            Key key = PassAlgor.generateKey(password_hash);
            String decryptedPass = PassAlgor.decrypt(passwordById.getPassword(), key);
            Key new_key = PassAlgor.generateKey(sharedUserByLogin.getPassword_hash());
            password.setPassword(PassAlgor.encrypt(decryptedPass, new_key));
            sharedUserByLogin.addPassword(password);
            passwordRepository.save(password);
            FunctionRun functionRun = new FunctionRun();
            functionRun.setFunctionName("share password");
            functionRun.setTime(LocalDateTime.now());
            ownerByLogin.addFunctionRun(functionRun);
            functionRunRepository.save(functionRun);
            return true;
        } else return false;
    }
    @Override
    public Boolean edit(EditRequest editPassRequest) throws Exception {
        Password firstByUser_loginAndIsOwnerTrue =
passwordRepository.findByUser_LoginAndIdAndOwnerIdIsNull(editPassRequest.getUserLogin(),
                Long.valueOf(editPassRequest.getPasswordId()));
        if (firstByUser_loginAndIsOwnerTrue != null) {
            User ownerByLogin = userService.findByLogin(editPassRequest.getUserLogin());
            Password passwordById =
passwordRepository.findById(Long.parseLong(editPassRequest.getPasswordId())).get();
            DataChange dataChange = new DataChange();
            dataChange.setPreviousValueOfRecord(passwordById.toString());
            List<Password> allByOwnerId =
passwordRepository.findAllByOwnerId(Long.parseLong(editPassRequest.getPasswordId()));
            if (editPassRequest.getNewWeb() != null) {
```

```
            passwordById.setWeb_address(editPassRequest.getNewWeb());
            allByOwnerId.forEach(password ->
password.setWeb_address(editPassRequest.getNewWeb())); }
        if (editPassRequest.getNewDescription() != null) {
            passwordById.setDescription(editPassRequest.getNewDescription());
            allByOwnerId.forEach(password ->
password.setDescription(editPassRequest.getNewDescription()));  }
        if (editPassRequest.getNewPassword() != null) {
            Key new_key = PassAlgor.generateKey(ownerByLogin.getPassword_hash());
            passwordById.setPassword(PassAlgor.encrypt(editPassRequest.getNewPassword(),
new_key));
            for (Password pass : allByOwnerId) {
                Key keyForEachUser = PassAlgor.generateKey(pass.getUser().getPassword_hash());
                pass.setPassword(PassAlgor.encrypt(editPassRequest.getNewPassword(),
keyForEachUser));    }  }
        if (editPassRequest.getNewLogin() != null) {
            passwordById.setLogin(editPassRequest.getNewLogin());
            allByOwnerId.forEach(password ->
password.setLogin(editPassRequest.getNewLogin()));
        }
        allByOwnerId.forEach(password -> passwordRepository.save(password));
        passwordRepository.save(passwordById);
        dataChange.setPresentValueOfRecord(passwordById.toString());
        dataChange.setTime(LocalDateTime.now());
        dataChange.setActionType("update password");
        ownerByLogin.addDataChange(dataChange);
        passwordById.addDataChange(dataChange);
        dataChangeRepository.save(dataChange);
        FunctionRun functionRun = new FunctionRun();
        functionRun.setFunctionName("update password");
        functionRun.setTime(LocalDateTime.now());
        ownerByLogin.addFunctionRun(functionRun);
        functionRunRepository.save(functionRun);
```

```
            return true;
        } else return false;
    }
    @Override
    public Boolean delete(String passwordId, String userLogin) {

        User byLogin = userService.findByLogin(userLogin);
        Password firstByUser_loginAndIsOwnerTrue =
                passwordRepository.findByUser_LoginAndIdAndOwnerIdIsNull(userLogin,
                        Long.valueOf(passwordId));
        if (firstByUser_loginAndIsOwnerTrue != null) {
            Password passwordById =
passwordRepository.findById(Long.parseLong(passwordId)).get();
            DataChange dataChange = new DataChange();
            dataChange.setPreviousValueOfRecord(passwordById.toString());
            passwordById.setIsDeleted(true);
            passwordRepository.save(passwordById);
            List<Password> passwordsList =
passwordRepository.findAllByOwnerId(Long.parseLong(passwordId));
            passwordsList.forEach(
                    password -> password.setIsDeleted(true)
            );
            dataChange.setPresentValueOfRecord(passwordById.toString());
            dataChange.setTime(LocalDateTime.now());
            dataChange.setActionType("delete password");
            byLogin.addDataChange(dataChange);
            passwordById.addDataChange(dataChange);
            dataChangeRepository.save(dataChange);
            FunctionRun functionRun = new FunctionRun();
            functionRun.setFunctionName("delete password");
            functionRun.setTime(LocalDateTime.now());
            byLogin.addFunctionRun(functionRun);
```

```
            functionRunRepository.save(functionRun);
            return true;
        } else return false;
    }
    @Override
    public OnePasswordResponse fidPasswordById(String id, String login) throws Exception {
        Password password = passwordRepository.findById(Long.valueOf(id)).get();
        User ownerByLogin = userService.findByLogin(login);
        String password_hash = ownerByLogin.getPassword_hash();
        Key key = PassAlgor.generateKey(password_hash);
        String decryptedPass = PassAlgor.decrypt(password.getPassword(), key);
        OnePasswordResponse map = modelMapper.map(password, OnePasswordResponse.class);
        map.setPassword(decryptedPass);
        return map;
    }
    @Override
    public Integer getKfromPasswordByPhraze(String password, String phrase) throws Exception {
        return NotificationManager.countMatches(password, phrase);
    }
    @Override
    public Boolean restore(String changeId, String userLogin) throws JsonProcessingException {


        User byLogin = userService.findByLogin(userLogin);


        DataChange dataChange = dataChangeRepository.findById(Long.valueOf(changeId)).get();
        Password password ;
        if( passwordRepository.findById(Long.valueOf(dataChange.getPassword().getId())).get()!=
null)
        {
password=passwordRepository.findById(Long.valueOf(dataChange.getPassword().getId())).get();
        }       else
        { password = new Password();}
```

```
        DataChange dataChangeToWrite = new DataChange();
        dataChangeToWrite.setPreviousValueOfRecord(password.toString());
        ObjectMapper mapper = new ObjectMapper();
        Password restoreToPassword = mapper.readValue(dataChange.getPresentValueOfRecord(),
Password.class);
        password.setDescription(restoreToPassword.getDescription());
        password.setIsDeleted(restoreToPassword.getIsDeleted());
        password.setLogin(restoreToPassword.getLogin());
        password.setOwnerId(restoreToPassword.getOwnerId());
        password.setPassword(restoreToPassword.getPassword());
        password.setWeb_address(restoreToPassword.getWeb_address());
        byLogin.addPassword(password);
        passwordRepository.save(password);
        dataChangeToWrite.setPresentValueOfRecord(password.toString());
        dataChangeToWrite.setTime(LocalDateTime.now());
        dataChangeToWrite.setActionType("restore password");
        byLogin.addDataChange(dataChangeToWrite);
        password.addDataChange(dataChangeToWrite);
        dataChangeRepository.save(dataChangeToWrite);
        FunctionRun functionRun = new FunctionRun();
        functionRun.setFunctionName("restore password");
        functionRun.setTime(LocalDateTime.now());
        byLogin.addFunctionRun(functionRun);
        functionRunRepository.save(functionRun);
        return true;
    }
}
```

Додаток Є – Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доц. кафедри ПЗ _____ О. М. Хошаба

Рецензент, д.т.н., проф. кафедри КН _____ О.М. Василевський

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____ О. М. Хошаба

Виконавець, студентка групи 1ПІ–19м _____ А. С. Штокал

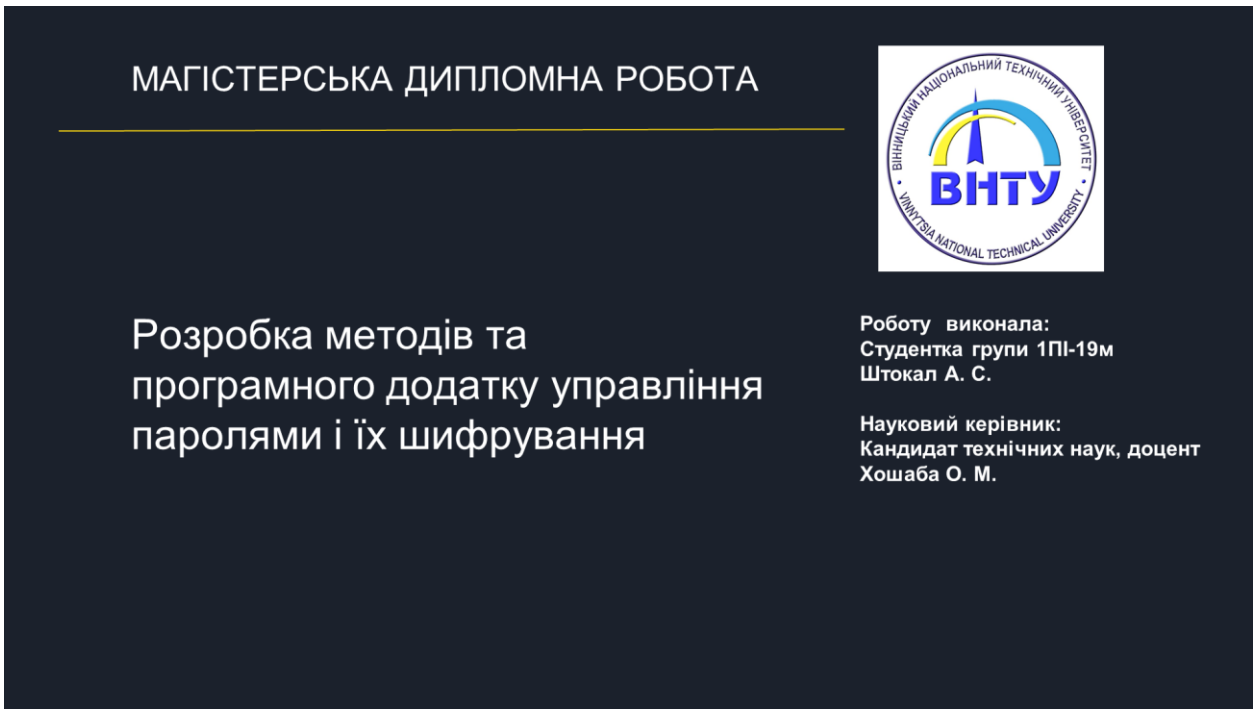

Рисунок Є.1 – Титульна сторінка

Рисунок Є.2 – Мета, об’єкт та предмет дослідження

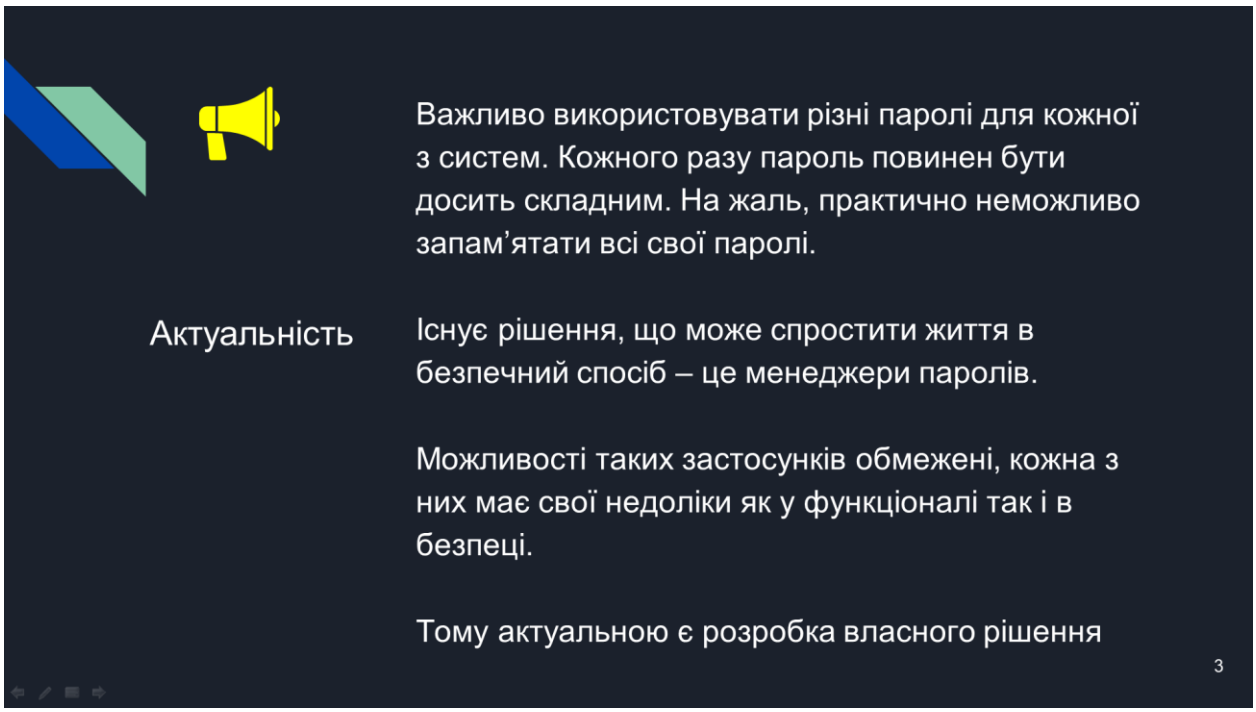

Рисунок Є.3 – Актуальність дослідження

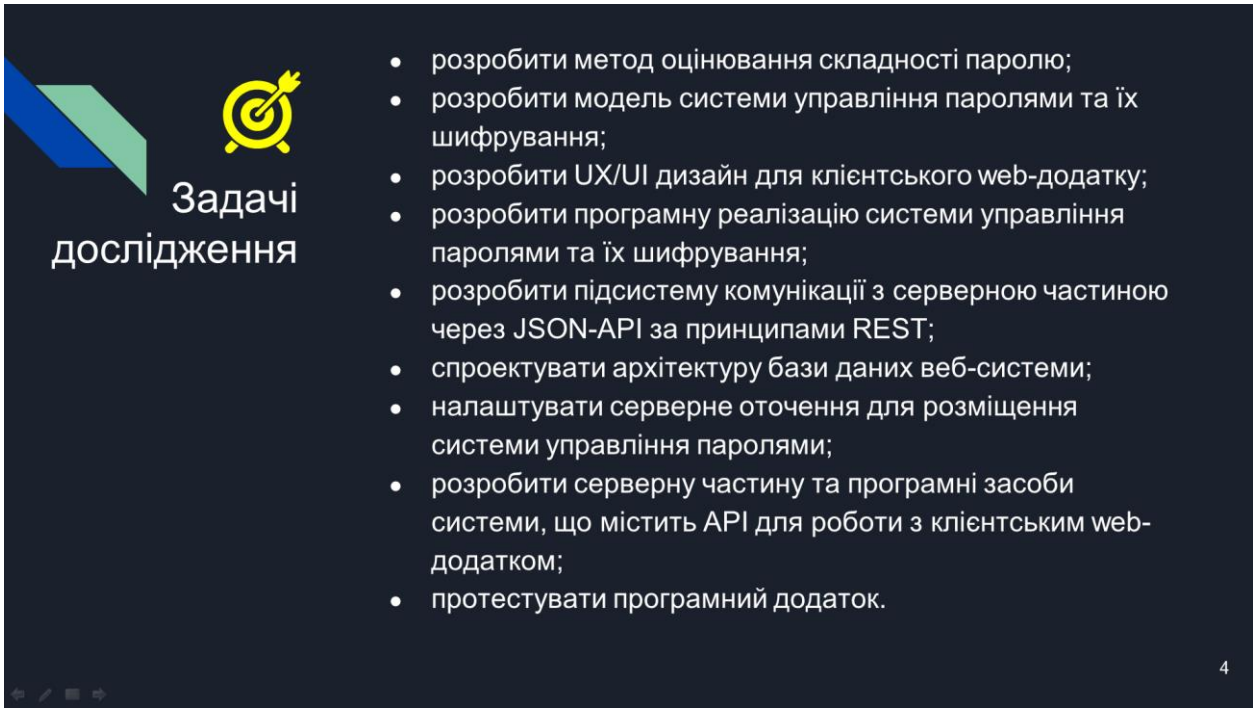

Рисунок Є.4 – Задачі дослідження

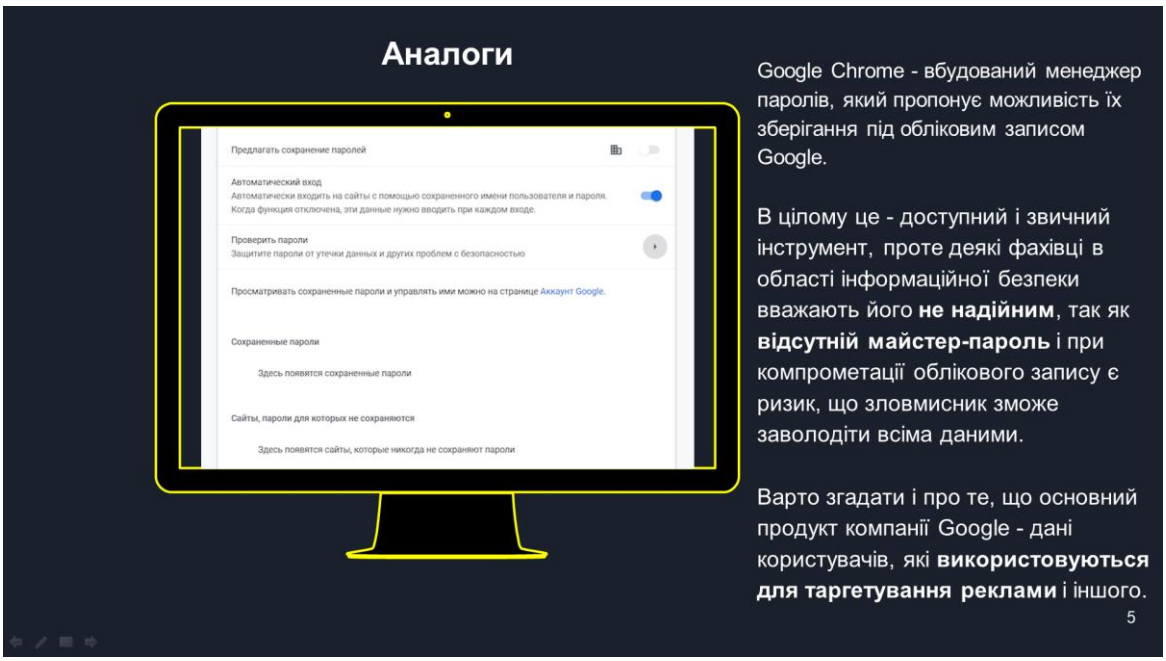

Рисунок Є.5 – Аналог «Google Chrome»

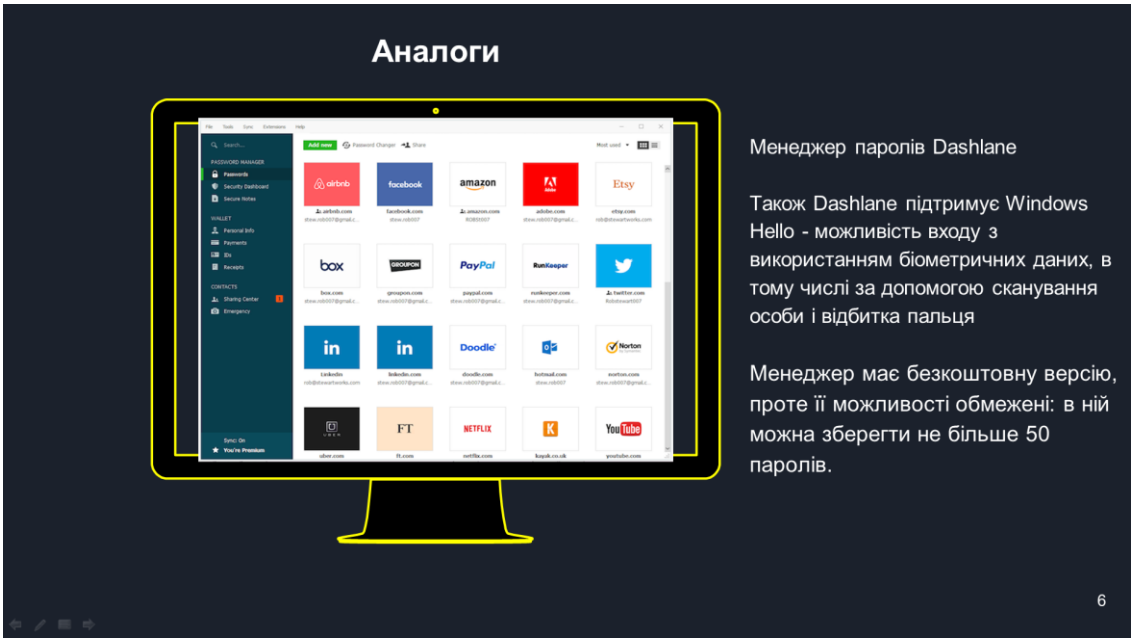

Рисунок Є.6 – Аналог «Dashlane»

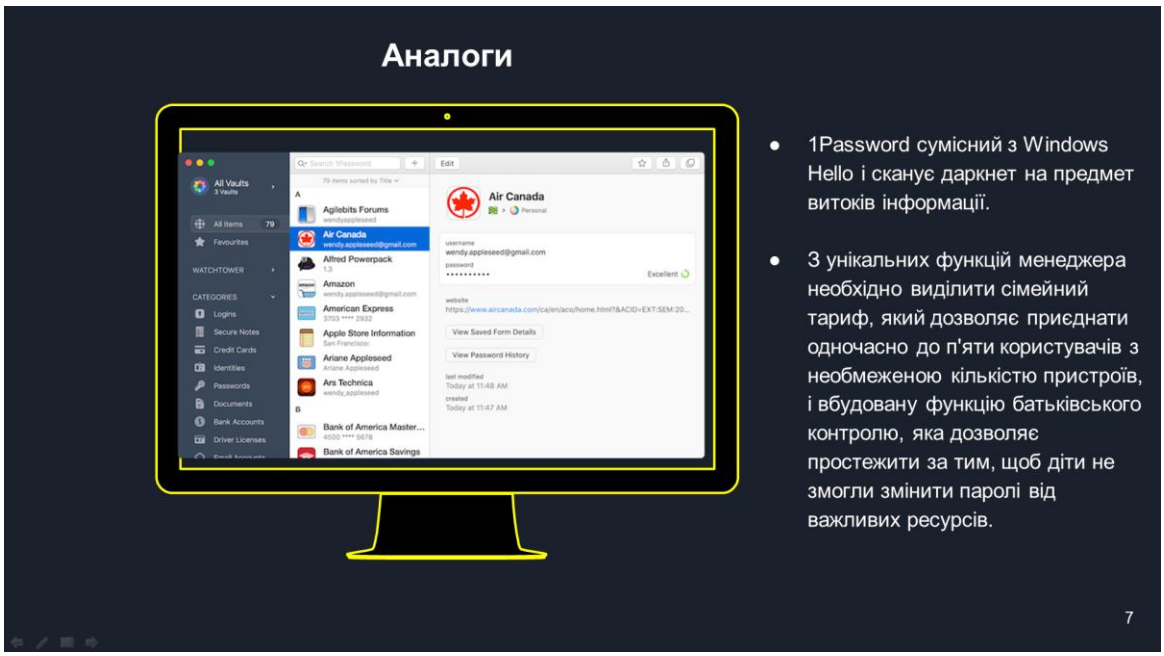

Рисунок Є.7 – Аналог «1Password»

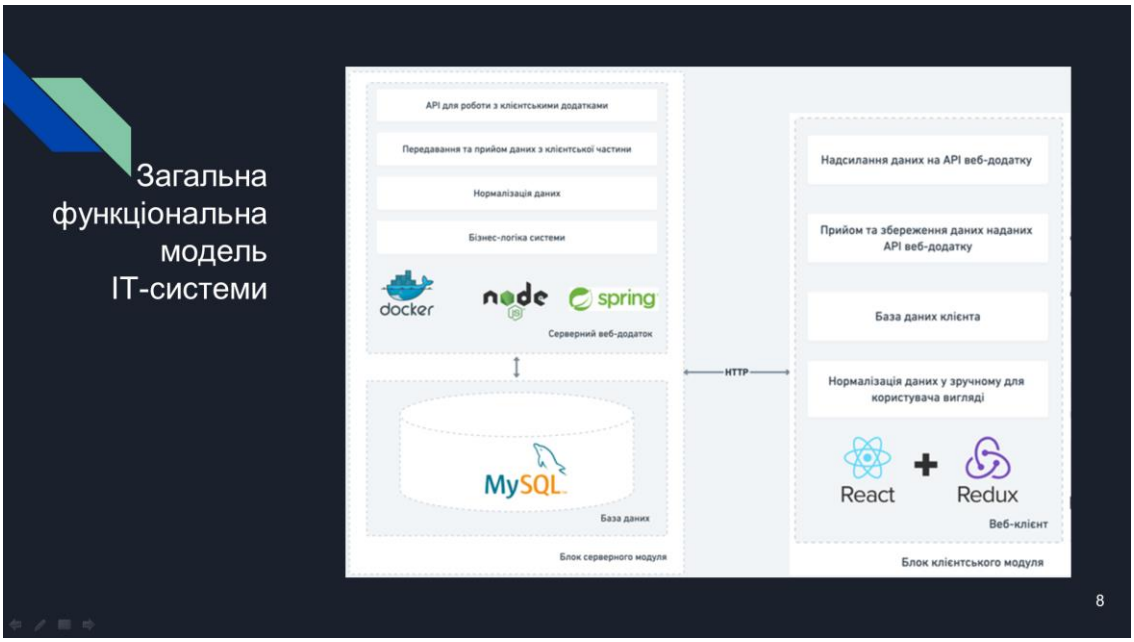

Рисунок Є.8 – Загальна функціональна можель IT-системи

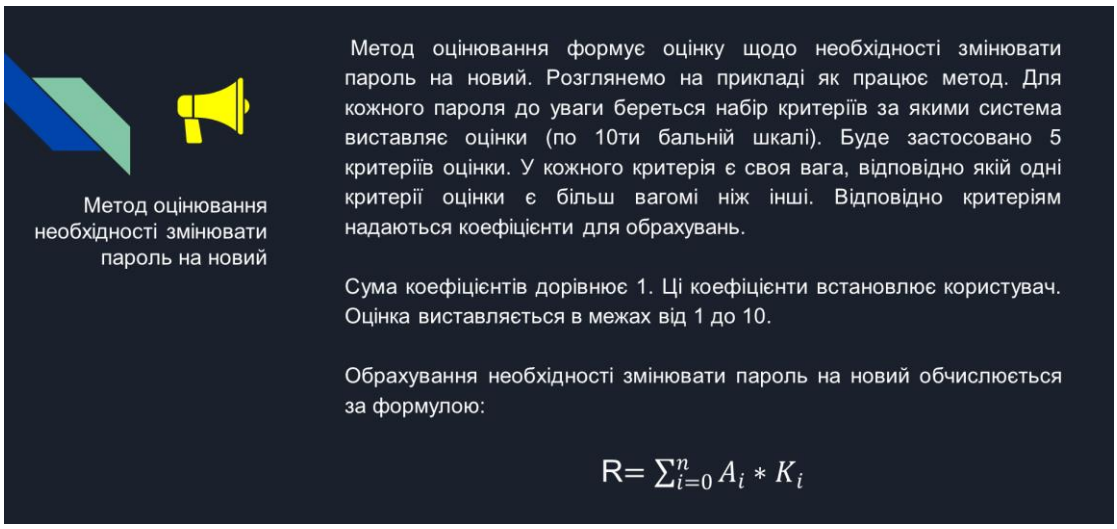

Рисунок Є.9 – Метод оцінювання необхідності змінювати пароль

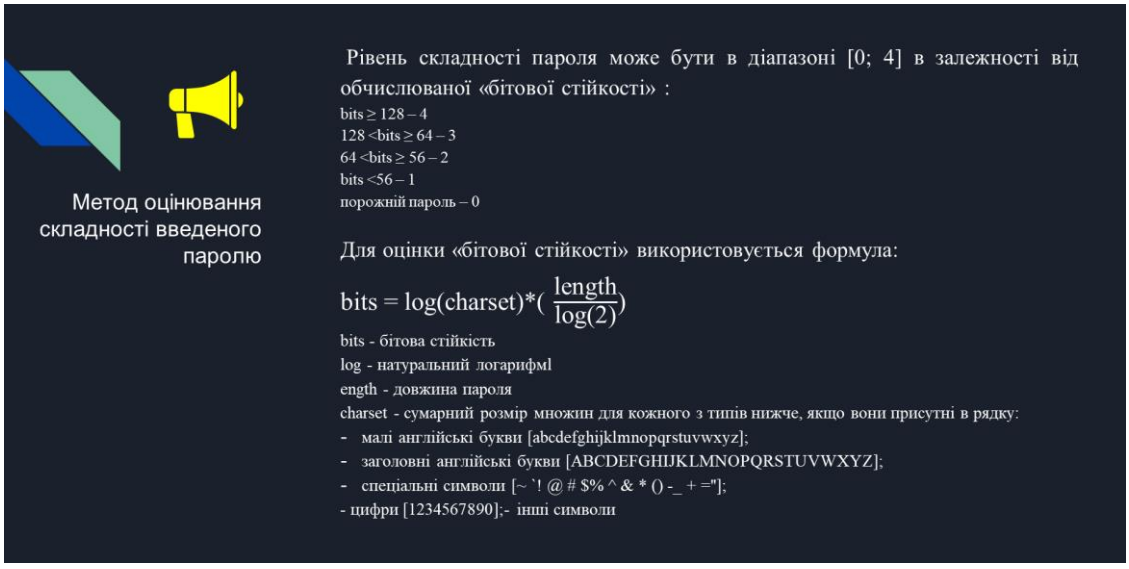

Рисунок Є.10 – Метод оцінювання складності введеного паролю

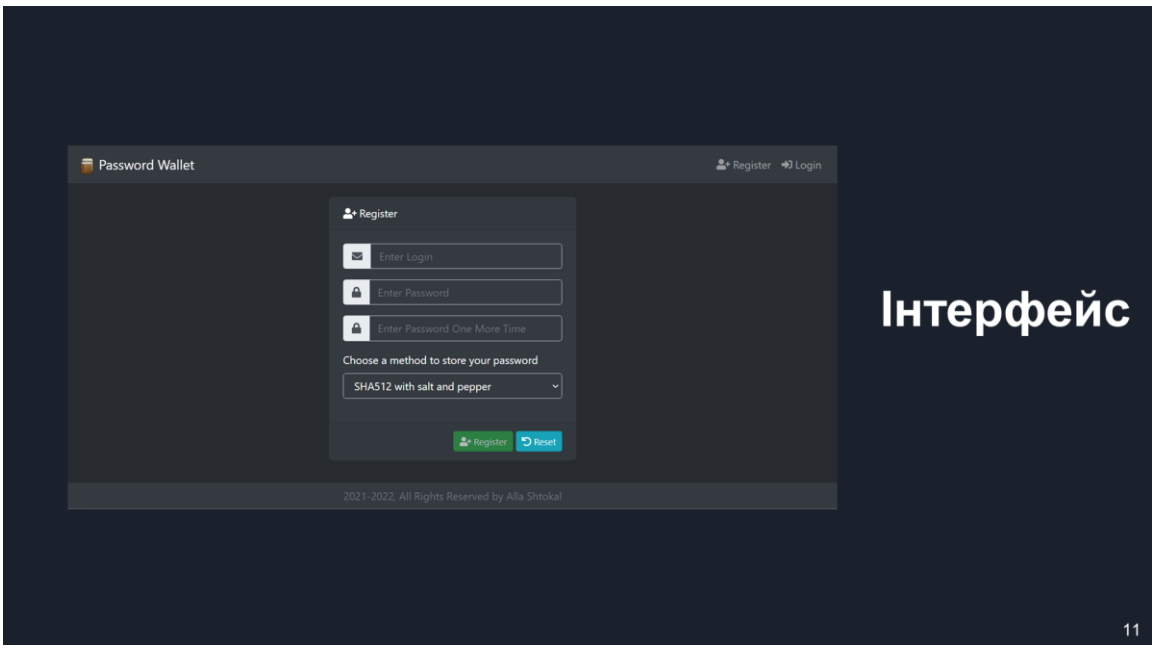

Рисунок Є.11 – Інтерфейс системи

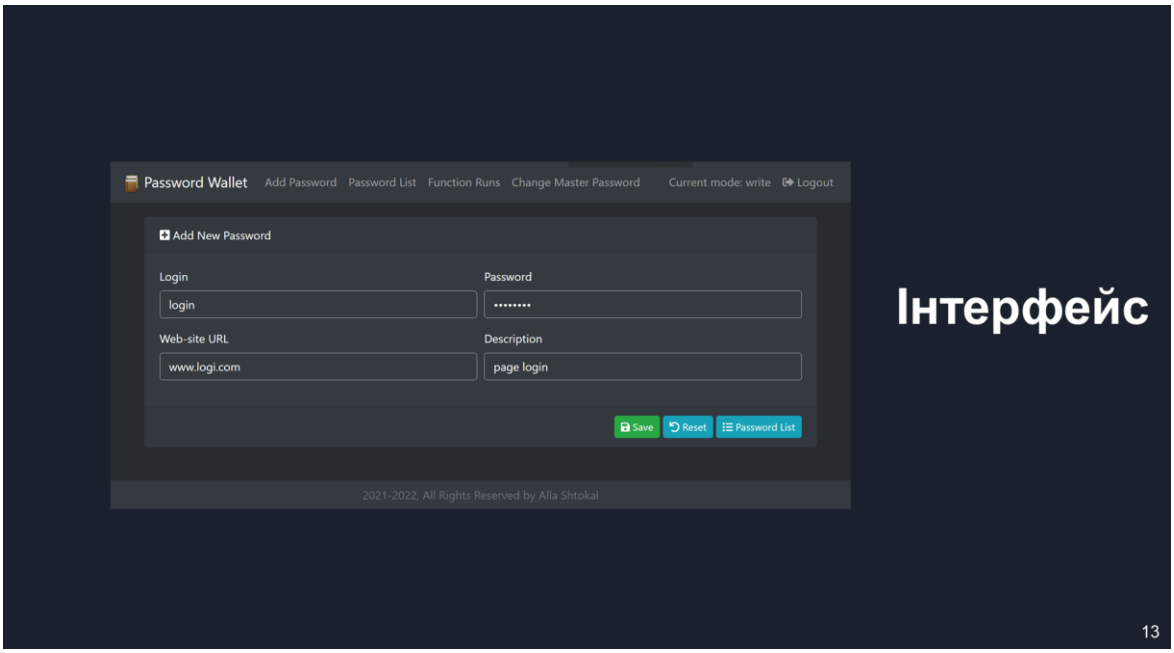

Рисунок Є.12 – Інтерфейс системи

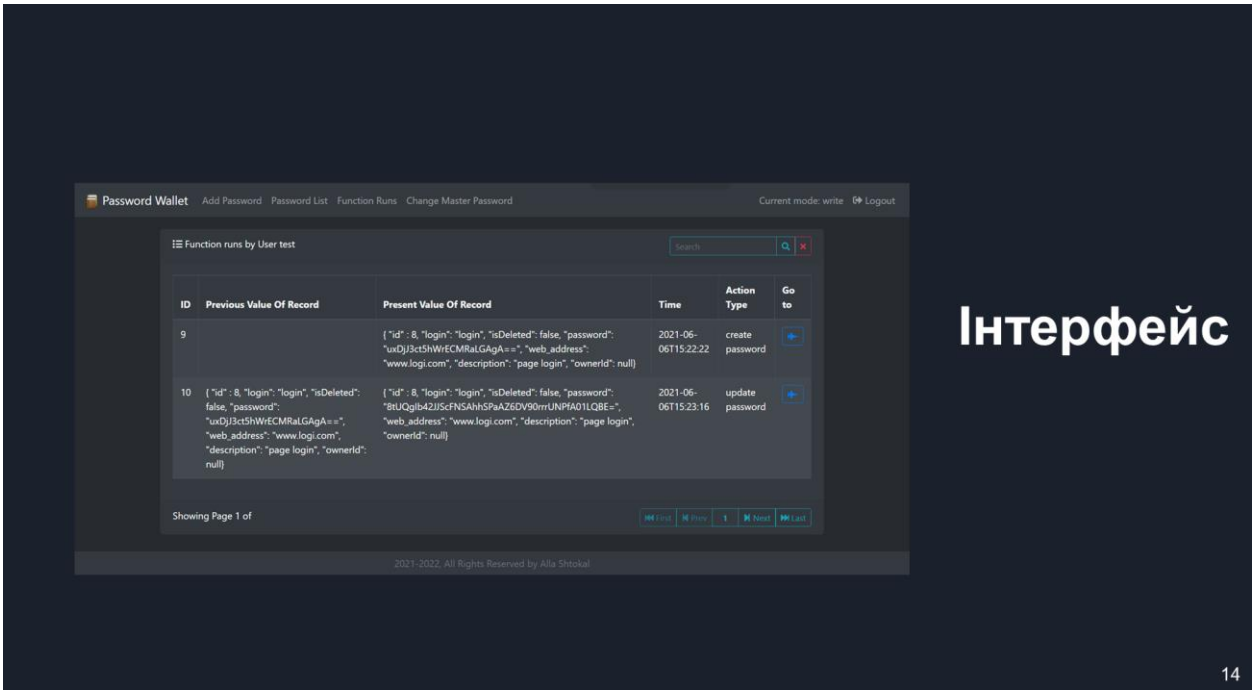

Рисунок Є.13 – Інтерфейс системи

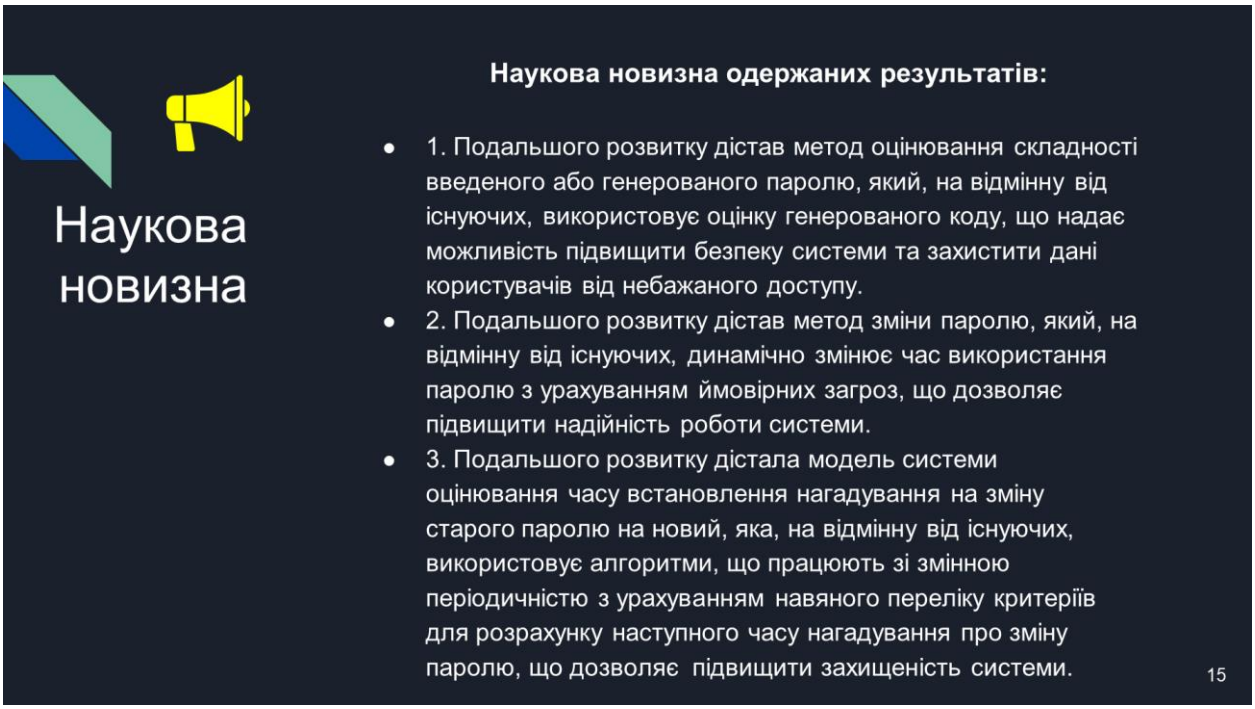

Рисунок Є.14 – Наукова новизна

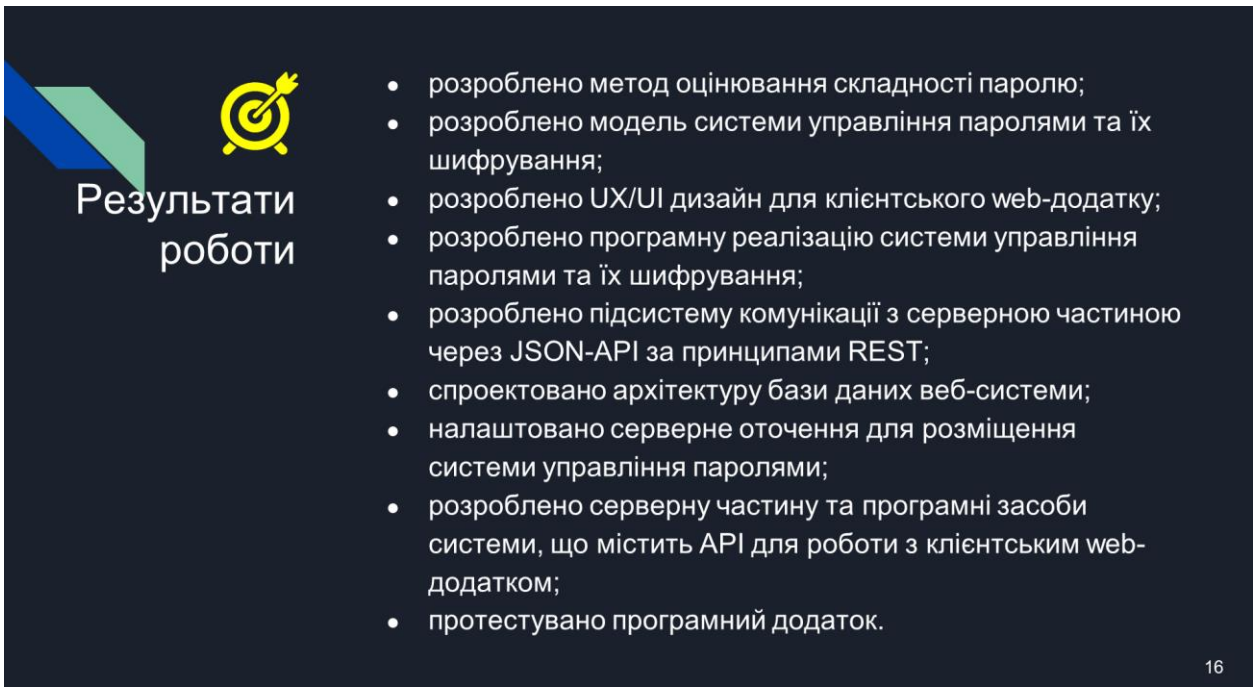

Рисунок Є.15 – Результати роботи

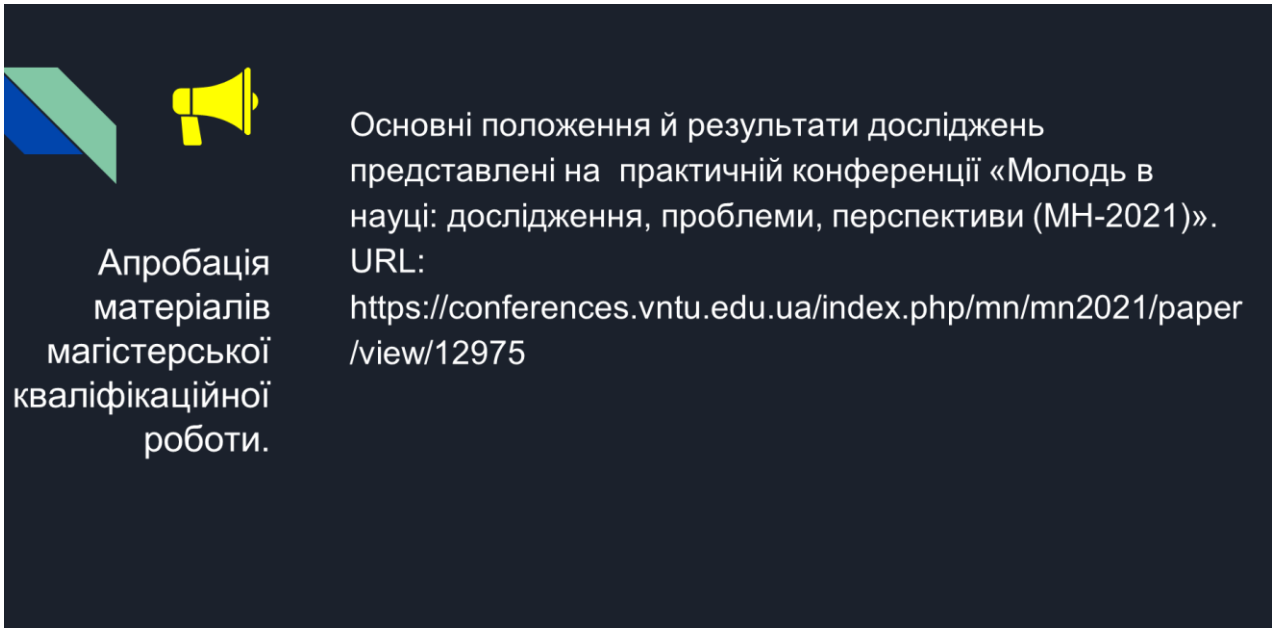

Рисунок Є.17 – Апробація матеріалів

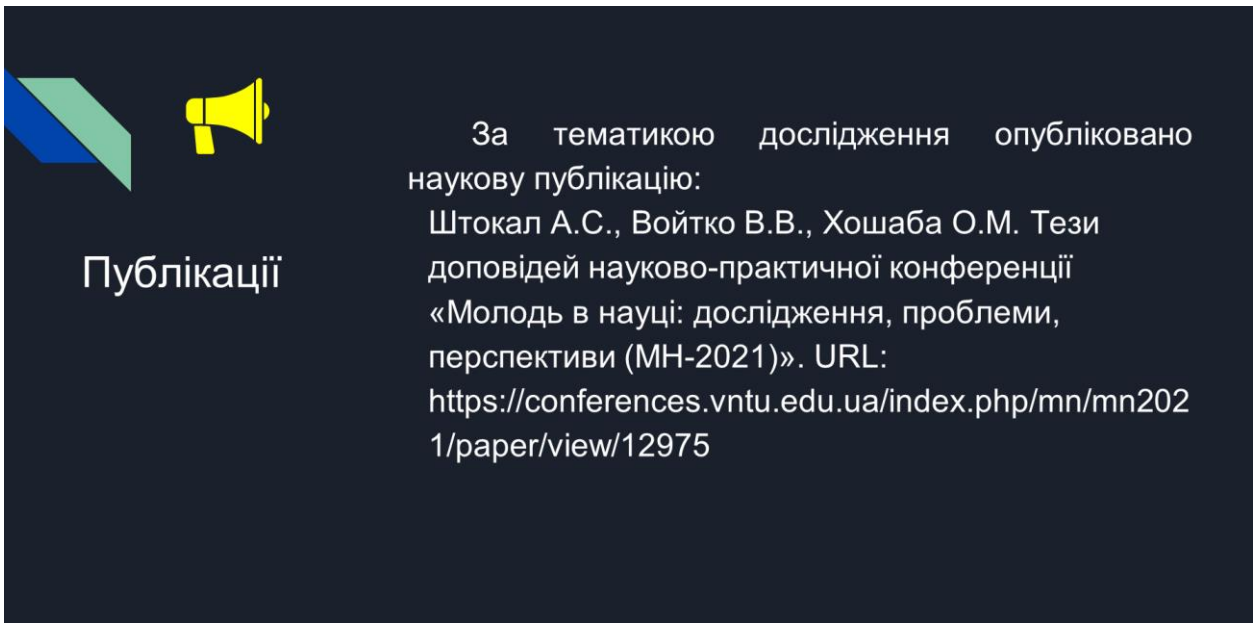

Рисунок Є.18 – Публікації