МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

«Захищений веб-ресурс з можливістю проведення анонімного опитування»

Завідувач
випускаючої кафедри **Любчак В.О.**

Керівник роботи **Любчак В.О.**

Студента групи КБ – 81 **Шептухін М.С.**

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра кібербезпеки

Затверджую_____

Зав. кафедрою Любчак В.О.

“_____” _____2022 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи КБ-81 спеціальності “Кібербезпека” денної форми навчання Шептухіна Матвія Сергійовича.

Тема: “**Захищений веб-ресурс з можливістю проведення анонімного опитування**”

Затверджена наказом по СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) Аналіз проблеми дослідження; 2) Методи вирішення задачі; 3) Практична реалізація.

Дата видачі завдання “_____”_____2022 р.

Керівник випускної роботи _____ Любчак В.О.

Завдання прийняв до виконання _____ Шептухін М.С.

РЕФЕРАТ

Записка: 48 стор., 20 рис., 2 додатки, 19 джерел.

Мета роботи — огляд існуючих можливостей онлайн-опитування із врахуванням захищеності та анонімності, пілотна розробка захищеного веб-додатку для опитування.

Об’єкт дослідження — процес веб-опитування і забезпечення захисту.

Предмет дослідження — моделі, алгоритми та інструменти для розробки додатку та його захисту.

Результати — було розроблено захищений веб-додаток для проведення анонімного опитування з шифруванням даних, введених респондентом та хешуванням паролів. Для розробки серверної частини додатку було використано технологію Node.js, для клієнтської – ReactJS.

ОНЛАЙН-ОПИТУВАННЯ, АНОНІМНЕ ОНЛАЙН-ОПИТУВАННЯ, ВЕБ-РЕСУРС, ШИФРУВАННЯ, ХЕШУВАННЯ, AES, SHA-256, ВЕБ-ДОДАТОК, ЗАХИЩЕНИЙ ВЕБ-ДОДАТОК, REACTJS, MONGODB, NODEJS.

ЗМІСТ

ВСТУП

Застосування інтернет-технологій для бізнесу та економіки, формування баз даних та знань, комунікацій, моніторингу процесів у суспільстві та вивчення громадської думки є ознакою та потребою сьогодення. Використовуються можливості мережі Інтернет для опитування – збору соціальної інформації про якісь конкретні факти, соціальні явища та події.

Дуже швидкими темпами впроваджуються веб-системи для анкетування та опитування у всіх сферах життєдіяльності. Наприклад: опитування та тестування в галузі освіти, проведення онлайн-референдумів і голосувань на державному рівні, вивчення громадської думки засобами ЗМІ, опитування думок та потреб учасників фірмами, кампаніями тощо.

Наказом Державного комітету телебачення та радіомовлення України в 2009 році було затверджено «Методичні рекомендації щодо використання Інтернету під час проведення опитування населення».

Веб-технології дозволяють реалізувати онлайн-опитування без використовування додаткових людських ресурсів, вибрати сприятливий час для участі в опитуванні, швидко отримати зворотній зв'язок після відповіді на питання та найголовніше – гарантувати анонімність. Саме анонімність користувача допомагає притягнути більшу кількість респондентів та підвищити рівень заповнення анкет.

Кожного року кількість веб-ресурсів збільшується, і саме тому зростає кількість конфіденційної інформації, яка знаходиться на серверах віддаленого доступу. З цього випливає не тільки ріст атак на веб-ресурси, але й певні економічні наслідки цих атак.

Метою цієї роботи є проведення огляду вже існуючих методів та технологій інтернет-анкетування, вивчення питань захисту даних та конфіденційності, аналіз вже існуючих веб-додатків для опитувань.

Завданням цієї роботи є розробка свого полегшеного захищеного веб-додатку для проведення анонімного опитування.

1 АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1 Онлайн-опитування

Онлайн-опитування – це спосіб краще дізнатися про своїх клієнтів: вивчити їхні звички та очікування, зібрати думки про продукти чи послуги та зрозуміти, в якому напрямку розвивати бізнес. Також це можливість виміряти рівень задоволеності організацією діяльності підприємства.

Опитування дозволяє також більше дізнатися про користувачів, які користуються продуктом. Наприклад, для власників або розробників сайту, соціальної мережі, будь-якого іншого онлайн-сервісу, то за допомогою опитування можливо отримати відомості про:

- користувачів та їх потреби;
- що вони вважають важливим у вашому продукті;
- подобається їм цей продукт, чи ні;
- чи знаходять вони його зручним, чи ні.

Опитування через веб-ресурс - найсучасніший спосіб отримання та обробки даних. Основними плюсами проведення онлайн-опитувань є швидкість отримання інформації та низька ціна, порівняно із традиційними техніками.[1]

Приклади вже існуючих веб-сервісів для проведення онлайн-опитування:

1) SurveyMonkey

В цього додатку є пробна та безкоштовна версії. Це веб-сайт для проведення онлайн-опитувань, який значно спрощує сам процес проходження опитування. На етапі розробки опитування цей додаток пропонує сімнадцять форматів для постановки питань - вибір декількох варіантів відповідей, відкриті питання тощо. Також цей додаток має різноманітну палітру кольорів для зміни зовнішнього вигляду опитування. Сервер розташований в США.[2]

2) Google Forms

Це програмне забезпечення для проведення та адміністрування онлайн-опитувань, яке входить до складу безкоштовного веб-редактора документів

Google, який пропонує Google. Також існує мобільна версія цього сервісу. Сервер розташований в США.[3]

3) Simpoll

Цей веб-додаток має безкоштовну версію. Це досить зручний сервіс для створення та проведення онлайн-опитувань, який можна поставити на свій веб-сайт та отримувати результати у реальному часі. Сервер розташований в Естонії.[4]

4) Survio

Це сервіс, який дозволяє досить легко створити онлайн-опитування та додати до них малюнки та навіть відео-файли. Результати опитувань можна отримувати у різних форматах у режимі реального часу. Даний веб-додаток має функцію створення резервних копій.

Стартова сторінка цього ресурсу показана на рисунку 1.1.

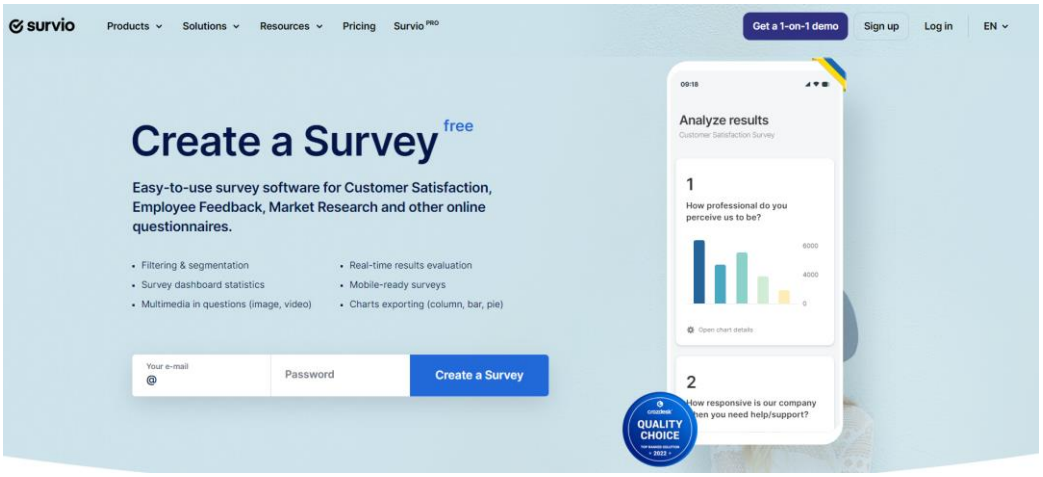

Рисунок 1.1 – Стартова сторінка Survio.com

Сервер компанії знаходиться в Чехії.[5]

5) Typeform

Хмарний сервіс для створення форм, тестів та опитувань. За допомогою цього сервісу компанії можуть створювати унікальні опитувальні форми, розміщувати їх на сайтах або ділитися ними у соціальних мережах, аналізувати відповіді респондентів. Сервіс підходить як для невеликого бізнесу, так і для великих корпорацій. Стартова сторінка цього сервісу показана на рисунку 1.2.

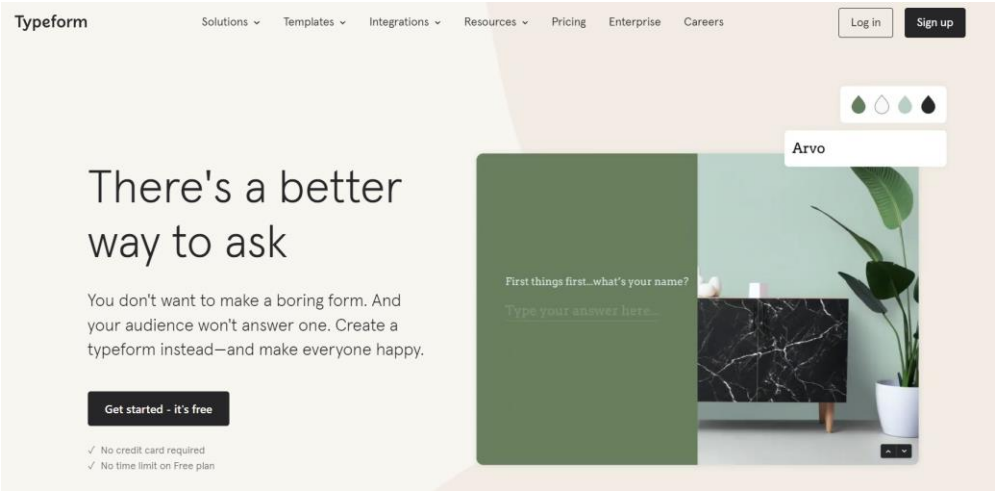

Рисунок 1.2 – Стартова сторінка Typeform.com

Форми в цьому веб-додатку створюються в конструкторі. Їх можна створювати, використовуючи готові шаблони з різноманітних тем або створити форму з нуля. Опитування складаються з кількох сторінок. На цих сторінках можна розміщувати питання, зображення та відео, посилання та інші елементи. Питання можуть бути різних типів. Це можуть бути питання з множинним вибором відповідей, тести, форми для збирання контактних даних або ж відкриті питання. Дизайн опитувань може бути будь-яким, користувачі можуть створювати теми, в яких використовуватимуться індивідуальні шрифти, кольори кнопок та інше. Сервер цього сервісу знаходиться в США.[6]

Онлайн-опитування можуть бути **анонімними**, завдяки цьому респонденти можуть дати більш чесні та відверті відповіді.

Анонімність – це стан, коли неможливо визначити особистість людини, відсутня ідентифікація в ході передачі або створенні якоїсь інформації. Конфіденційні дані не можуть бути розкриті нікому, окрім тих, хто володіє ними чи представляє власника цих даних. Також, анонімність надає можливість безборонно формулювати свої думки без будь-якого глузування, переслідувань, порушень прав людини або дискримінації. В процесі проведення анонімного онлайн-опитування потрібно запевнити користувача, що його особу не буде ідентифіковано.

Люди, які ініціюють онлайн-опитування зазвичай зацікавлені в зборі певної конфіденційної інформації від учасників опитування, щоб прояснити, хто

дійсно проходив онлайн-опитування, а хто ні, наприклад, щоб надіслати нагадування. Або, наприклад, фіксувати кінцеві пристрої та операційні системи, які були використані респондентами під час заповнення онлайн-анкет, або дані про місцезнаходження, наприклад, щоб знати місце, з якого учасники проходили це опитування. І саме тому, тут може сформуватись конфлікт інтересів, оскільки учасники опитування вважають за краще не розкривати свою особу. Саме тому необхідно дати можливість залишати анонімні відгуки, забезпечувати анонімний зворотній зв'язок та гарантувати захищеність даних.

Можливі схеми жорсткого запису анонімних опитувань можна знайти, наприклад, в опитуваннях пацієнтів або просто в опитуваннях місцевого населення. Організатори онлайн-опитування постійно публікують з посиланням на анонімне опитування, для того, щоб збільшити сприймання учасників анкетування в опитуваннях клієнтів або співробітників і таким чином підвищити процент відгуків.

Було б набагато краще проводити онлайн-опитування анонімно та збирати тільки анонімні відгуки, у якому всі персональні дані, наприклад, IP-адреса, операційна система, адреса електронної пошти респондента, не будуть потрапляти до організатора цього опитування.

Це вагомо, якщо ініціатор має на меті отримати відгук щодо будь-якої персональної інформації. А втім, пропозиція анонімності респондента, як правило, є дуже гарною практикою для будь-якого способу опитування.[7]

Прикладом може слугувати веб-додаток Xoyondo.

Це безкоштовна, багатофункціональна платформа для планування подій, створення простих та анонімних опитувань. На головній сторінці (Рисунок 1.3) можна створити анонімне опитування, подивитись на приклад такого опитування.

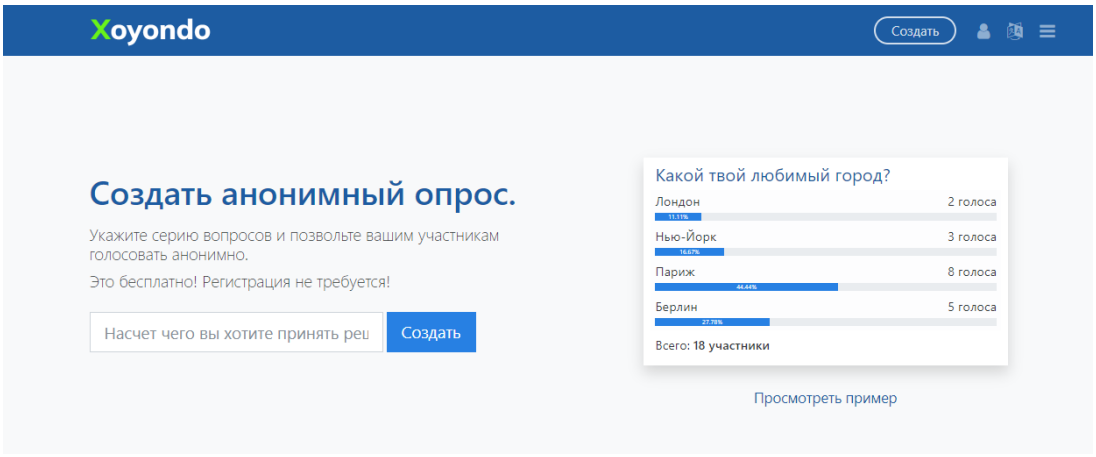

Рисунок 1.3 – Головна сторінка

Для створення анонімного опитування не потрібна реєстрація. Після того як була вибрана тема опитування та натиснута кнопка “Створити” з’являється форма з вибором теми опитування та варіантів відповідей.(Рисунок 1.4) Також, на цій сторінці є декілька додаткових налаштувань – чи можна обирати декілька варіантів відповідей, чи можна відправляти декілька відповідей з однієї IP-адреси.

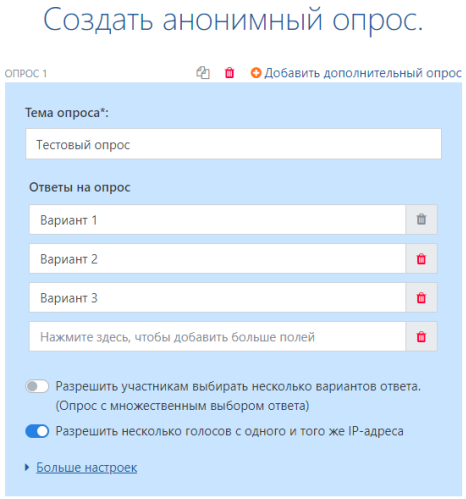

Рисунок 1.4 – Форма створення опитування

Також можна обрати ще декілька налаштувань. Показати результат – якщо цей параметр відключити, то респонденти не зможуть побачити результати опитування. Показати голоси – якщо цей параметр відключити, то учасники не зможуть побачити кількість голосів, але зможуть ознайомитись з кінцевим графіком.

Після створення опитування треба заповнити ще одну форму, в якій треба вказати ім’я того, хто створив опитування (обов’язково) та електрону

адресу(необов’язково). Електронна пошта потрібна для того, щоб додаток надіслав посилання на опитування та розділ адміністрування поштою.

Після всіх цих маніпуляцій З’являється форма (Рисунок 1.5) з посиланнями на опитування для тих, хто буде проходити його та посилання для адміністратора.[8]

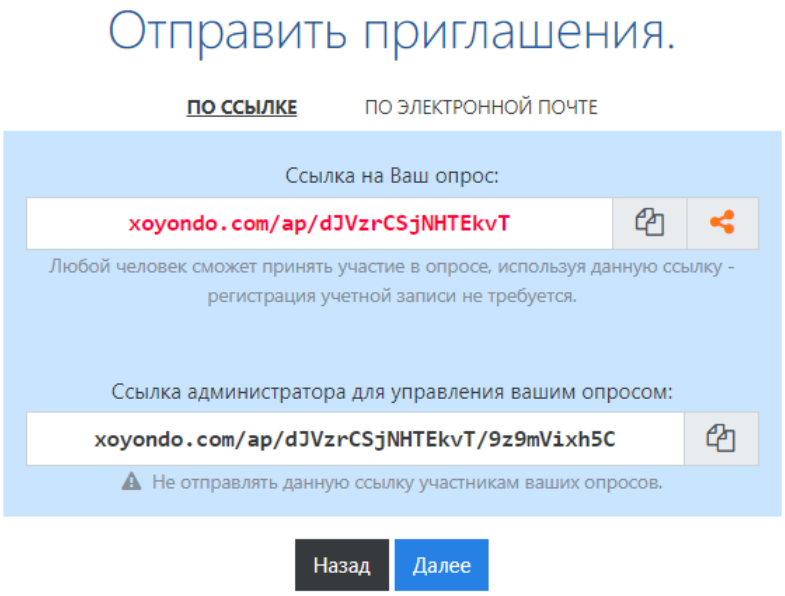

Рисунок 1.5 – Форма з посиланнями на опитування

Переваги:

- Просте налаштування опитувань;
- не потребує плати за створення опитування;
- лаконічний та зрозумілий інтерфейс;
- багато варіантів налаштування опитування;
- додатковий корисний функціонал можна “Спробувати безкоштовно” без прив’язування банківської карти (строком на 30 діб).

Недоліки:

- застарілий дизайн;
- додатковий корисний функціонал потрібно розблоковувати за гроші.

1.2. Веб-додаток

Веб-додаток – це комп’ютерна програма, яка використовує браузер для виконання певної функції. Веб-додаток це прикладна програма клієнт-сервер, тому в цьому середовищі багато комп'ютерів можуть обмінюватися такою інформацією, як збереження інформації в базі даних. "Клієнт" може використовуватися для введення інформації, а "сервер" використовується як сховище для інформації.

Веб-додаток виконує певні функціональні можливості через Інтернет, використовуючи веб-браузери та веб-технології. Програми обробляють сховище та отримують інформацію за допомогою скриптів на стороні сервера.

До найбільш розповсюджених веб-додатків можуть відноситися соціальні мережі, інтернет-магазини, онлайн-додатки для бізнесу тощо.[9]

Види веб-додатків:

- SPA або Single page application – це односторінкові програми в яких використовується як бекенд, так і фронтенд частини. За сприянням цих двох частин можна створити програму, яка буде виконувати певну роботу взагалі без перезавантаження сторінки на стороні клієнта;
- MPA або Multi Page Application – це багатосторінкові програми, які працюють за традиційною схемою. Це означає, що при кожній незначній зміні даних або завантаженні нової інформації оновлюється сторінка. Такі програми важчі, ніж односторінкові, тому їх використання доцільно лише у випадках, коли потрібно відобразити велику кількість контенту;
- Прогресивні програми або Progressive Web Application взаємодіють із користувачем, як програма. Вони можуть встановлюватися на головний екран смартфона, надсилати push-сповіщення та працювати в режимі офлайн.[10]

2 МЕТОДИ ВИРІШЕННЯ ЗАДАЧИ

2.1 Захист онлайн опитування

2.1.1. Шифрування

Шифрування — це зворотнє перетворення інформації з метою приховування від неавторизованих осіб, водночас, з наданням авторизованим користувачам доступу до неї. Шифрування служить завданням дотримання конфіденційності інформації, що передається. Головною властивістю будь-якого алгоритму шифрування є використання ключа, який затверджує вибір конкретного перетворення із сукупності можливих даного алгоритму.

Алгоритмами шифрування діляться на симетричні алгоритми та асиметричні алгоритми:

- Симетричне шифрування. Дані алгоритми використовують той самий ключ, як для зашифровування, так і для розшифровування;
- Асиметричне шифрування. Ці алгоритми використовують два різні ключі: відкритий - для зашифровування, закритий - для розшифровування.

Приклади симетричних алгоритмів:

- Data Encryption Standard (DES). Це алгоритм симетричного шифрування, він був розроблений фірмою IBM.
- Triple DES (3DES) - прямий розвиток шифру DES. У цьому алгоритмі шифрування та розшифровування виконуються шляхом триразового виконання алгоритму DES.
- AES. Цей шифр впроваджено в програмне та апаратне забезпечення по всьому світу для шифрування конфіденційних даних. Це важливо для державної комп’ютерної безпеки та захисту електронних даних.
- Blowfish. Це криптографічний алгоритм, що реалізує блокове симетричне шифрування довжиною ключа, яка може змінюватись.

Більш детально розглянемо симетричний шифр AES.

AES - це симетричний шифр, обраний урядом США для захисту секретної інформації.

У сучасній криптографії AES широко використовується і підтримується як апаратним, так і програмним забезпеченням. На сьогоднішній день жодних практичних криптоаналітичних атак проти AES не було виявлено. Крім того, AES має вбудовану гнучкість довжини ключа, що дозволяє певну міру «захистити майбутнє» від прогресу у можливості виконувати вичерпний пошук ключів.

Особливості алгоритму шифрування AES наступні:

- Симетричний ключовий симетричний блоковий шифр;
- 128-бітні дані, 128/192/256-бітні ключі;
- Сильніший і швидший, ніж Triple-DES;
- Програмне забезпечення реалізоване на C і Java;
- Високий рівень безпеки.

Атаки на шифрування AES:

Дослідження атак на шифрування AES продовжуються з моменту завершення розробки стандарту в 2000 році. Різні дослідники публікували атаки проти скорочених версій AES.

Ось кілька потенційних способів атаки на шифрування AES:

- У 2009 році вони виявили можливу атаку пов'язаного ключа. Цей криптоаналіз намагався зламати шифр, вивчаючи, як він працює за допомогою різних ключів. Атака пов’язаних ключів виявилася загрозою лише для систем AES, які неправильно налаштовані;
- У 2009 році відбулася атака з відомим ключем на AES-128. Для розпізнавання структури шифрування використовувався відомий ключ. Однак злом був спрямований лише на версію AES-128 з восьми раундів, а не на стандартну версію з 10 раундами, що робить загрозу відносно незначною.[11]

Кожен раунд алгоритму складається із чотирьох кроків. Алгоритм роботи AES описано нижче.

Крок 1 - Заміна байтів.

На першому етапі байти тексту блоку замінюються на основі правил, продиктованих зумовленими S-блоками (скорочення від блоків підстановки). (Рисунок 2.1)

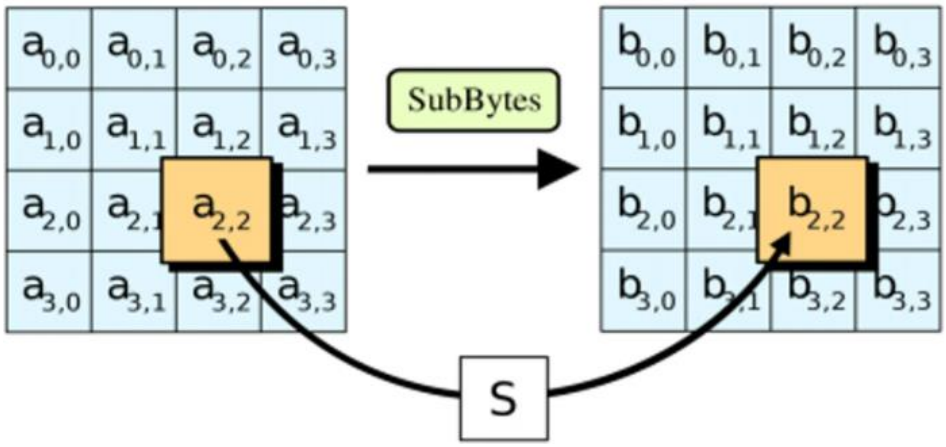

Рисунок 2.1 - Заміна байтів

Крок 2 - Зсув рядків.

Наступним кроком йде етап перестановки. На цьому кроці всі рядки, крім першого, зсуваються на одиницю, як показано на рисунку 2.2.

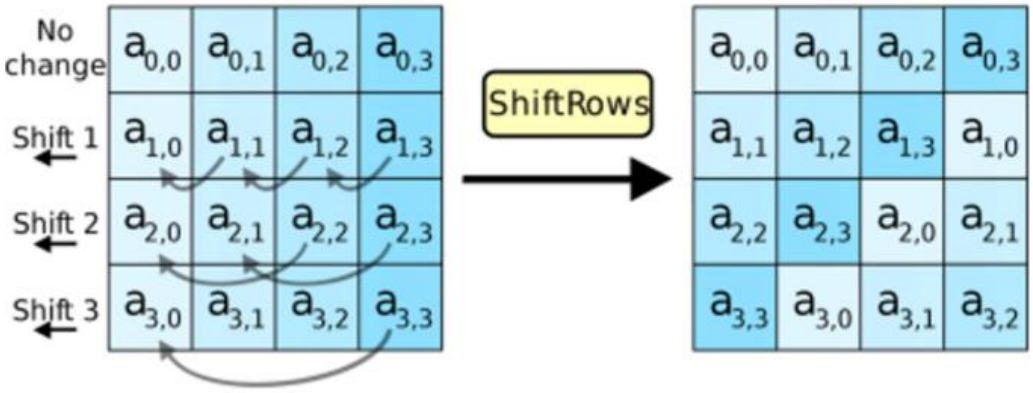

Рисунок 2.2 - Зсув рядків

Крок 3 - Змішування стовпців.

На третьому кроці використовується шифр Хілла для того, щоб ще більше заплутати повідомлення шляхом змішування стовпців блоку. (Рисунок 2.3)

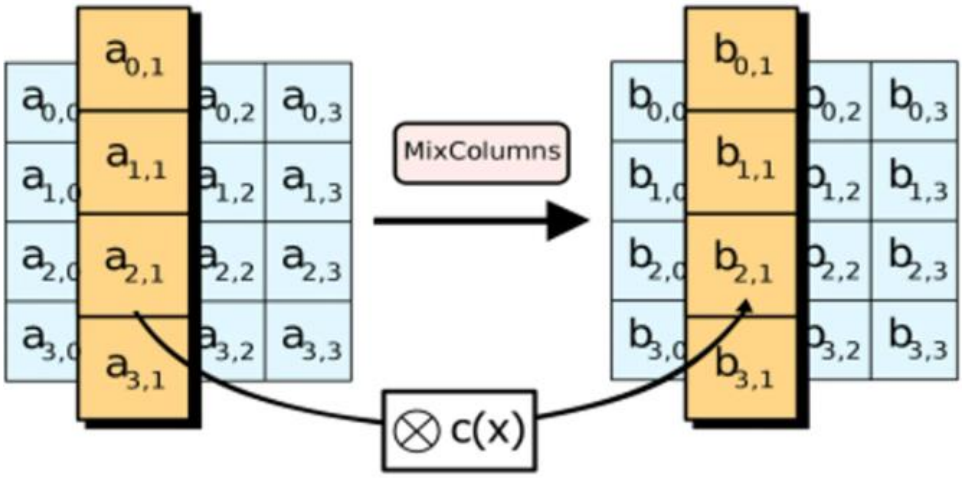

Рисунок 2.3 - Змішування стовпців

Крок 4 - Додавання раундового ключа.

На останньому етапі повідомлення піддається операції XOR із відповідним раундовим ключем. (Рисунок 2.4)

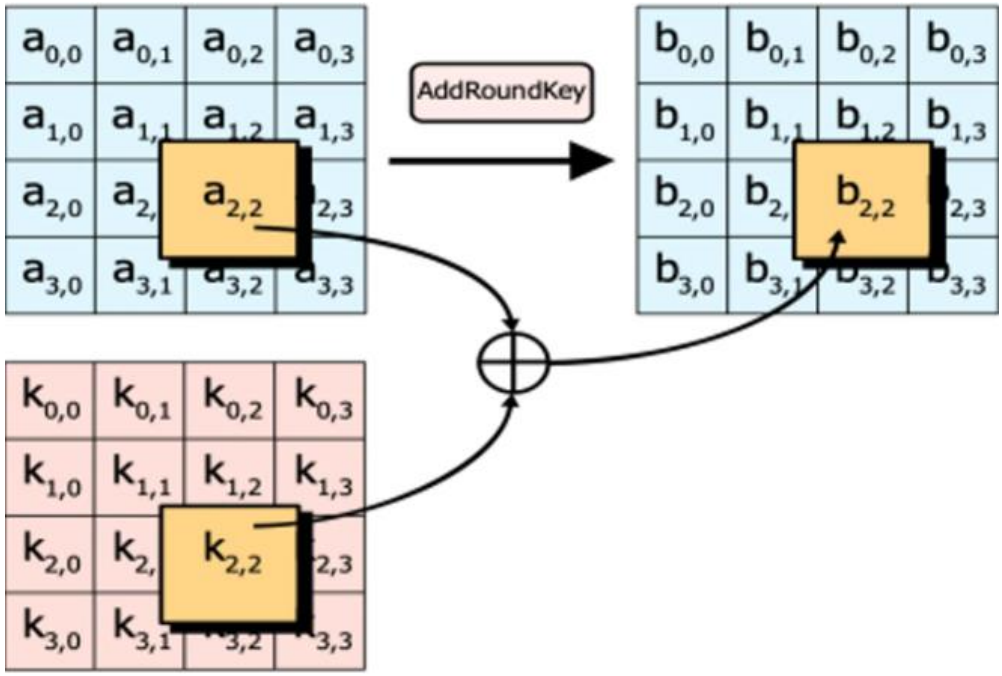

Рисунок 2.4 - Додавання раундового ключа

При багаторазовому виконанні цих кроків забезпечується захист кінцевого зашифрованого тексту.[12]

2.1.2 Хешування

Хешування — це процес перетворення будь-якого заданого ключа або рядка символів в інше значення. Поширеним використанням хешування є

реалізація хеш-таблиць, що зберігають пари ключів і значень у списку, доступному через її індекс. Тому що пари ключів і значень нескінченні, хеш-функція відобразить ключі до розміру таблиці, тоді захешоване значення буде індексом для відповідного значення.

Хеш-функція генерує нові значення згідно з математичним алгоритмом хешування, більш відомого як хеш. Щоб запобігти перетворенню хешу у зворотній вихідний ключ, використовується односторонній алгоритм хешування.

Хешування застосовується до індексації та пошуку даних, цифрових підписів, кібербезпеки та криптографії. Багато алгоритмів шифрування використовують хешування для підвищення кібербезпеки.

Хешовані рядки та не мають значення для хакерів без ключа дешифрування. Наприклад, якщо хакери зламали базу даних і знайшли конкретні дані, вони можуть використати цю інформацію. Однак захешоване значення, є марним для зловмисників, якщо вони не мають ключа для його розшифрування. Процес хешування допомагає захистити паролі, що зберігаються в базі даних.

Криптографія використовує кілька хеш-функцій для захисту даних. Деякі з найпопулярніших криптографічних хешів:

- Алгоритм безпечного хешування 1 (SHA-1);
- Алгоритм безпечного хешування 2 (SHA-2);
- Алгоритм безпечного хешування 3 (SHA-3);
- MD2;
- MD4;
- MD5.

Завдяки хеш-функціям - MD2, MD4 та MD5 можна хешувати цифрові підписи. Після хешування підпис перетворюється на коротше значення, яке називається дайджестом повідомлення.[13]

Алгоритм безпечного хешування (SHA) - це стандартний алгоритм, який використовується для створення більшого дайджеста повідомлення. Хоча це як

хеш-функція дайджесту повідомлень MD4 - і добре зберігає та витягує базу даних — це не найкращий підхід для криптографічних цілей або перевірки помилок. SHA-2 використовується для створення більшого, 224-розрядного дайджеста повідомлень. SHA-3 є наступником SHA-2.

Розглянемо більш детально алгоритм хешування SHA-256. Цей алгоритм є різновидом SHA-2. Результатом хешування за алгоритмом SHA-256 завжди має бути хеш, який має довжину 64 символи.

Сутністю алгоритму хешування SHA-256 є те, що він робить з випадкового набору даних певне значення з фіксованою довжиною, яке буде слугувати ідентифікатором цих даних. Отримане значення порівнюється з дублікатами вихідних даних, витягти які майже неможливо.[14]

2.2 Технології для розробки додатку

2.2.1 Node.js

Node.js це платформа для роботи з JavaScript, яка працює на движку V8. Він використовує JS код, і потім перетворює його на швидший низькорівневий код. Це код, який може запускати комп'ютер без петреби його інтерпретувати. Node.js дозволяє користуватися єдиною мовою – JavaScript, для написання коду як на стороні клієнта, так і на сервері. Ці можливості Node.js є важливими для розробки програм реального часу, що базуються на подіях.[15]

Переваги Node.js:

- Швидка обробка запитів;
- Повна підтримка JSON, без перетворення між бінарними моделями за допомогою JavaScript;
- Синтаксис JavaScript;
- безліч безкоштовних інструментів.

Недоліки Node.js:

- Дуже повільна робота;
- Непротестовані оновлення.[16]

2.2.2 Express.js

Express.js - це простий фреймворк Node.js, що дає великий вибір різних можливостей для того, щоб розробити власний веб-додаток.

Він використовує протокол передачі гіпертексту, а також дає ряд готових варіантів, які спрощують створення серверної логіки.

До основних переваг фреймворку входять:

- Можливість кастомізації;
- Детальна нормативна база;
- Використання однієї мови програмування.

А до основних недоліків:

- JavaScript не пропонує рішень у сфері безпеки;
- Відсутність подальшого розвитку.[17]

На мою думку, фреймворк Express.js на платформі Node.js є чудовим вибором для реалізації цього веб-додатку.

2.2.3 MongoDB

MongoDB — це нереляційна база даних документів, яка підтримує сховище, подібне до JSON. Дана СУБД має гнучку модель даних, яка дозволяє зберігати неструктуровані дані, а також забезпечує повну підтримку індексації та реплікації за допомогою багатих та зрозумілих API. В основі цієї БД лежить концепція колекцій та документів.

Головні характеристики цієї СУБД: масштабованість, безпека та доступність.

Головними перевагами MongoDB є:

- Гнучкість;
- Всі дані накопичуються у BSON-форматі;
- Підтримка спеціальних запитів;
- Дані не потрібно наводити до одного типу;
- Ціна. MongoDB є безкоштовною СУБД.

Недоліки MongoDB:

- Обмежений розмір і вкладення даних. Розмір документа обмежується лише 16 МБ;
- Велике використання пам'яті. [18]

2.2.4 ReactJS

React – це бібліотека для розробки інтерфейсу користувача, яка базується на мові програмування JavaScript. Цю бібліотеку було створено дуже відомою компанією Facebook. Наразі ця компанія займається потужною підтримкою свого продукту.

Головними особливостями цієї бібліотеки є те, що вона здійснює концепт односторінкових додатків. Це такі додатки, в яких весь контент на сторнці змінюється динамічно. Завдяки цьому дуже серйозно збільшується продуктивність додатку. А також поділ всього контенту на сторінці на окремі компоненти для повторного використання в цьому, чи в інших проектах.

Далі пропоную подивитись на переваги та недоліки цієї бібліотеки.

Переваги ReactJS:

- Добре підходить для командної розробки, завдяки тому, що є суворе дотримання UI та шаблону робочого процесу;
- Компонентний підхід - розробка додатків на основі окремих компонентів;
- Підтримка від Facebook. Бібліотека оновлюється доволі часто;
- Висока швидкодія.

Недоліки ReactJS:

- Невелика кількість офіційної документації.
- ReactJS не підтримує браузери від Internet Explorer 8 та менше;[19]

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Опис шифрування результатів опитування та паролів

Для здійснення шифрування результатів опитування буде використано симетричний алгоритм шифрування AES.

В запропонованому мною веб-додатку дані будуть передаватися з клієнтської частини веб-додатку на серверну частину, та у зворотному напрямку. Для того, щоб було можливо зробити шифрування та дешифрування інформації, яка була засекречена за допомогою симетричного алгоритму шифрування, треба ключ передати з серверної частини на клієнтську.

Для того, щоб зашифрувати або розшифрувати результати опитування на клієнті потрібен ключ. Цей ключ отримується з унікального ідентифікатора опитування при створенні його структури. Для шифрування та дешифрування даних використовується такий алгоритм:

1. На клієнті створюється опитування та відправляється на сервер.
2. На сервері структура опитування зберігається в таблиці бази даних.
3. Після проходження опитування, результати шифруються за допомогою унікального ідентифікатора, згенерованого в базі даних.
4. Для отримання даних опитування на клієнт передається публічний ключ та зашифровані дані, які потім на клієнті будуть розшифровуватись.

3.2 Проектування веб-додатку

3.2.1 Побудова бази даних

В процесі моделювання структури сховища інформації, перш за все треба зробити аналіз всіх сутностей:

- Користувач;
- Опитування;
- Відповіді.

Конфігурація бази даних показана рисунку 3.4.

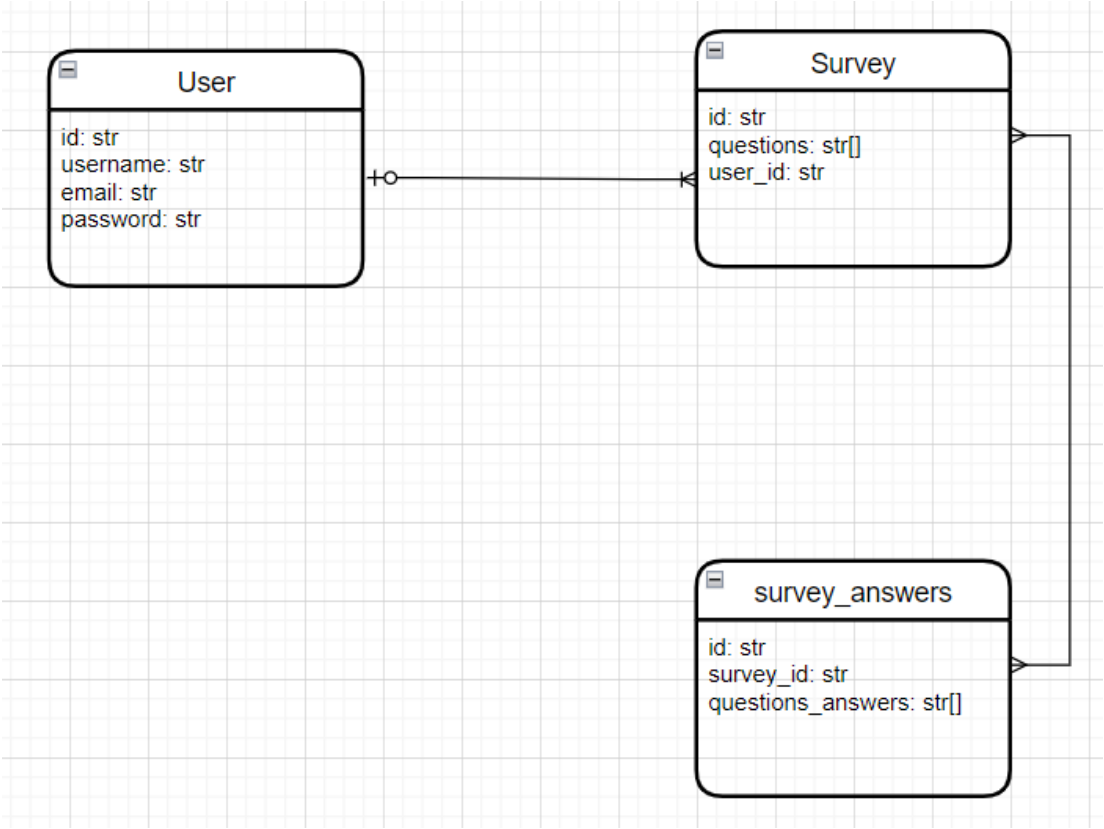

Рисунок 3.4 - Конфігурація сховища інформації

3.2.2 Моделювання інтерфейсу додатку

Наступним кроком треба визначити сторінки та частини веб-додатку.

- Перш за все, потрібно створити сторінку, на який користувач зможе пройти процедуру авторизації у свій акаунт. Поки користувач не увійде до свого власного акаунту, він не зможе створити опитування.
- Для проходження авторизації треба зробити сторінку, на який користувач зможе створити свій власний акаунт.
- Після цього треба зробити основну сторінку, на яку буде потрапляти користувач після проходження процедури авторизації. На цій сторінці у користувача буде можливість натиснути на кнопку “Створити опитування”, а також продивитись список вже створених їм опитувань.

- Після того, як користувач натисне “Створити опитування” відкриється сторінка, на який можна створити своє опитування. На цій сторінці треба ввести назву опитування та самі питання.
- Далі, коли користувач натисне на кнопку “Створити опитування” він потрапить на сторінку, де буде посилання на це опитування.
- Наступною сторінкою стане сторінка з опитуванням. На цій сторінці респондент зможе пройти опитування.
- Після цього, користувач, що створив це опитування, після переходу на головну сторінку зможе продивитись відповіді, що дали респонденти після проходження онлайн-опитування.

3.3 Розробка та опис додатку

Для того, щоб розпочати розробку додатку треба зробити певні налаштування в середовищі розробки. В моєму випадку це Visual Studio Code. Це безкоштовний та дуже популярний редактор коду від відомої компанії Microsoft.

Вся розробка цього веб-додатку буде складатися з двох основних фаз – розробка бекенду та фронтенду.

Для розробки серверної частини було використано наступні сторонні бібліотеки:

- Express.js. Це фреймворк, за допомогою якого була проведена розробка серверної частини додатку;
- Бібліотека моделювання об’єктних даних на основі Node.js для MongoDB.

Далі, на рисунку 3.11 можна побачити файли, які були створені для розробки серверної частини цього веб-додатку.

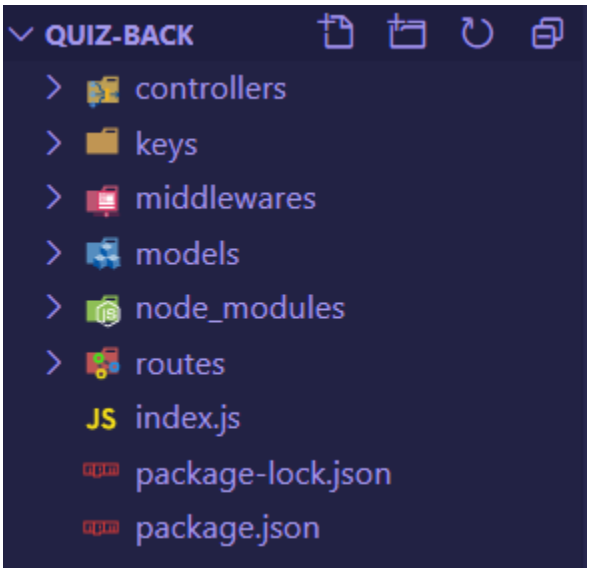

Рисунок 3.11 – Структура файлів бек-енд частини веб-додатку

Для того, щоб розпочати розробку фронтенд частини веб-додатку потрібно знову зробити конфігурування Visual Studio Code.

В бібліотеці ReactJS використовується компонентний підхід для розробки додатків. Тому ці компоненті можна розділити на:

- Компоненти, що відповідають за відрисовування контенту на сторінці;
- Компоненти, що відповідають за розташування контенту на сторінці;
- Інші службові компоненти.

Структуру файлів фронт-енд частини веб-додатку можна побачити на рисунку 3.12.

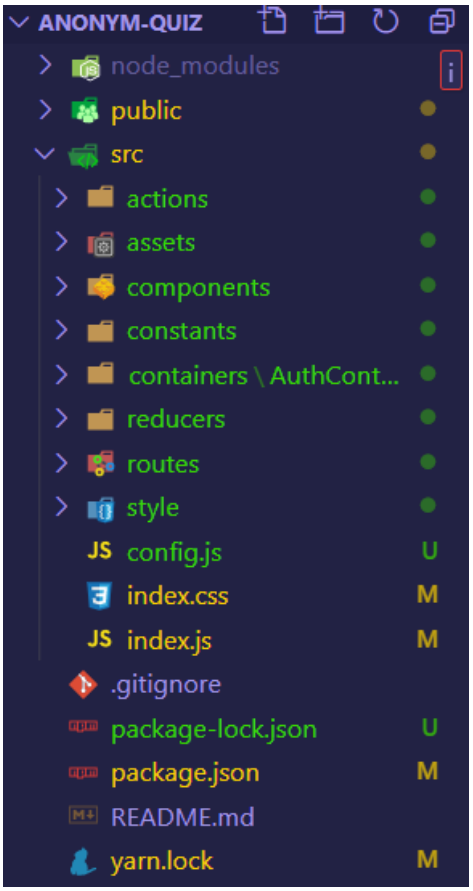

Рисунок 3.12 - Структура файлів фронт-енд частини веб-додатку

Хешування паролів було реалізовано за допомогою алгоритму SHA-256, з використанням солі. Реалізація хешування паролів на серверній частині надана в Додатку А.1.

Шифрування було реалізовано за допомогою симетричного алгоритму шифрування AES. Реалізація функції для шифрування даних на серверній частині надана в Додатку А.2. Реалізація функції для шифрування даних на клієнтській частині надана в Додатку А.3.

В Додатку Б наданий код для створення серверної (Додаток Б.1) та клієнтської (Додаток Б.2) частин веб-додатку. Клієнтська частина потрібна для відображення інтерфейсу всього додатку, а серверна – для збереження даних, які надходять з клієнтської частини, для взаємодії з базою даних.

3.4 Результати розробки власного веб-додатку

Наслідком розробки став захищений веб-додаток, з можливістю проведення анонімного опитування. Ресурсом для експлуатації може бути майже

будь-який браузер, виключенням може бути лише браузер Internet Explorer 8 та версії нижче.

Далі, пропоную розглянути вже останню версію цього веб-додатку. На рисунках, що надані нижче, показаний основний функціонал розробленого веб-додатку.

1) Сторінка для проведення процедури реєстрації.(Рисунок 3.13)

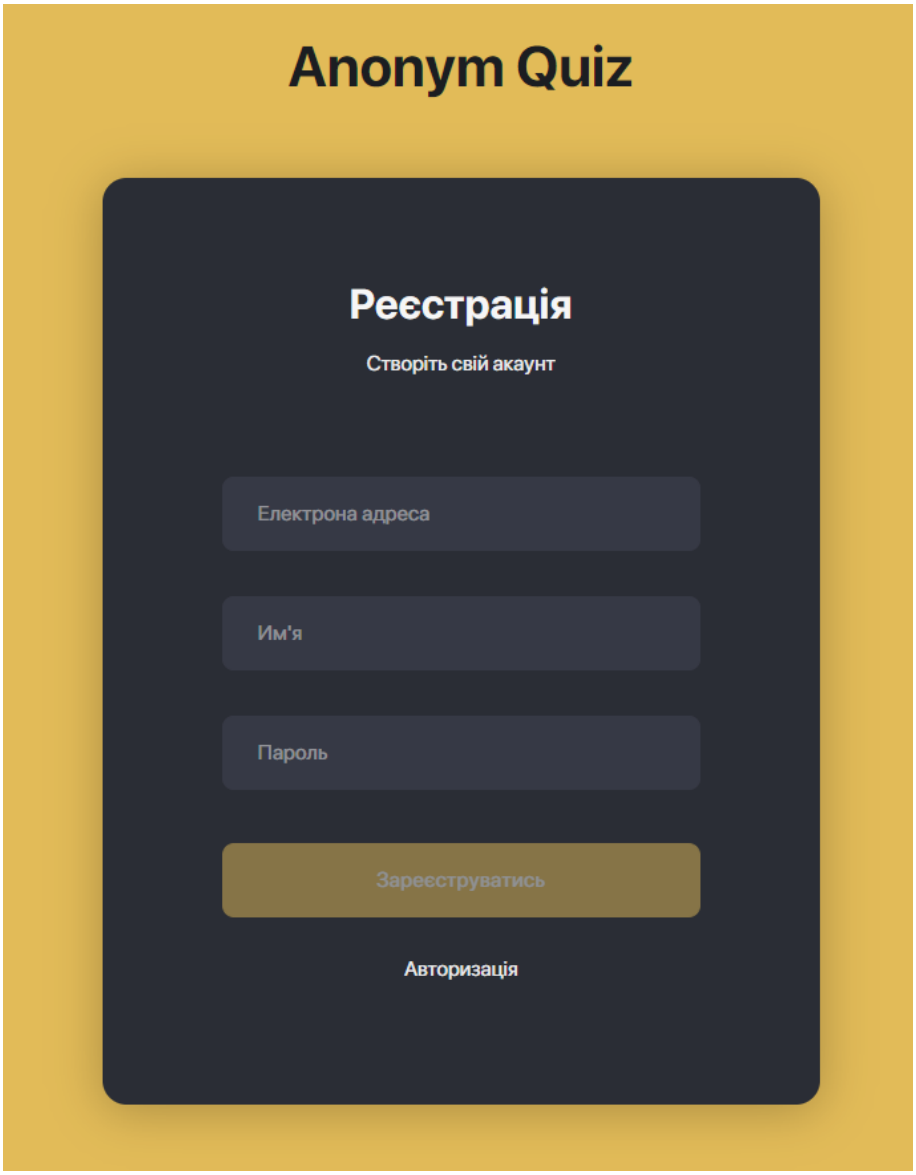

Рисунок 3.13 – Сторінка реєстрації

2) Сторінка для проведення процедури авторизації.(Рисунок 3.14)

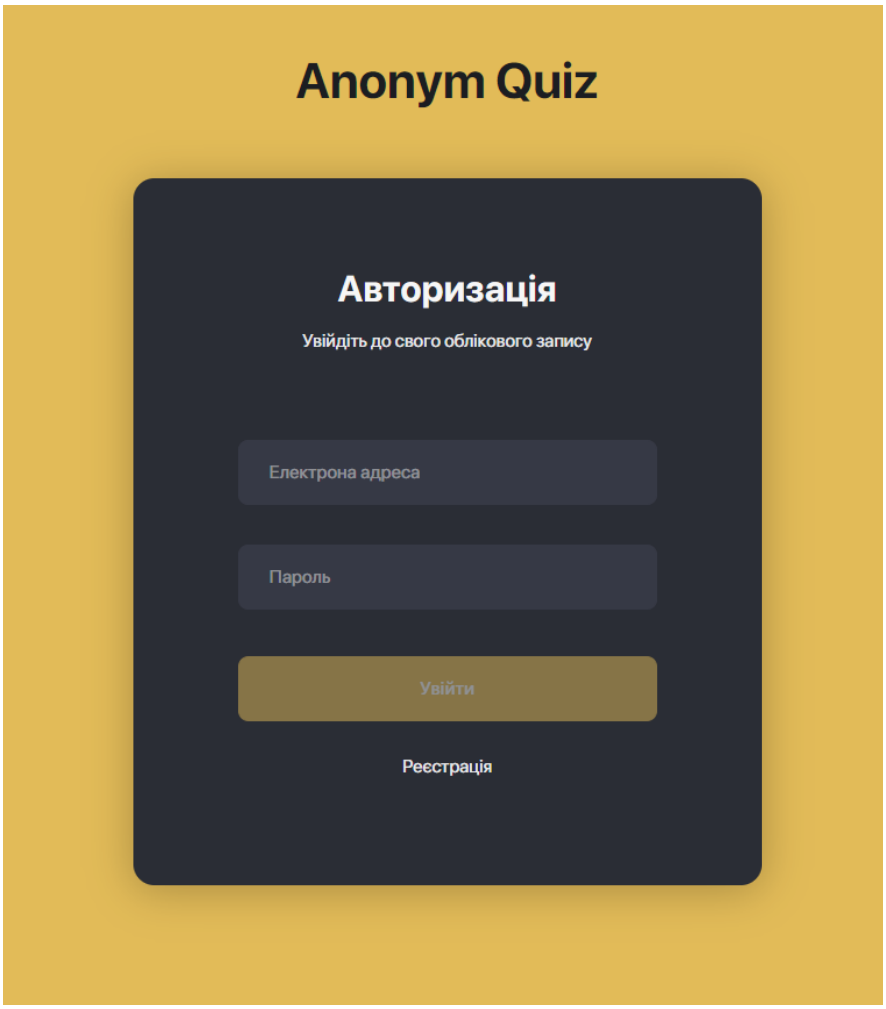

Рисунок 3.14 – Сторінка авторизації

3) Після авторизації користувач потрапляє на головну сторінку, де можна створити нове опитування або продивить вже існуючі. Зараз ще не має створених опитувань.(Рисунок 3.15)

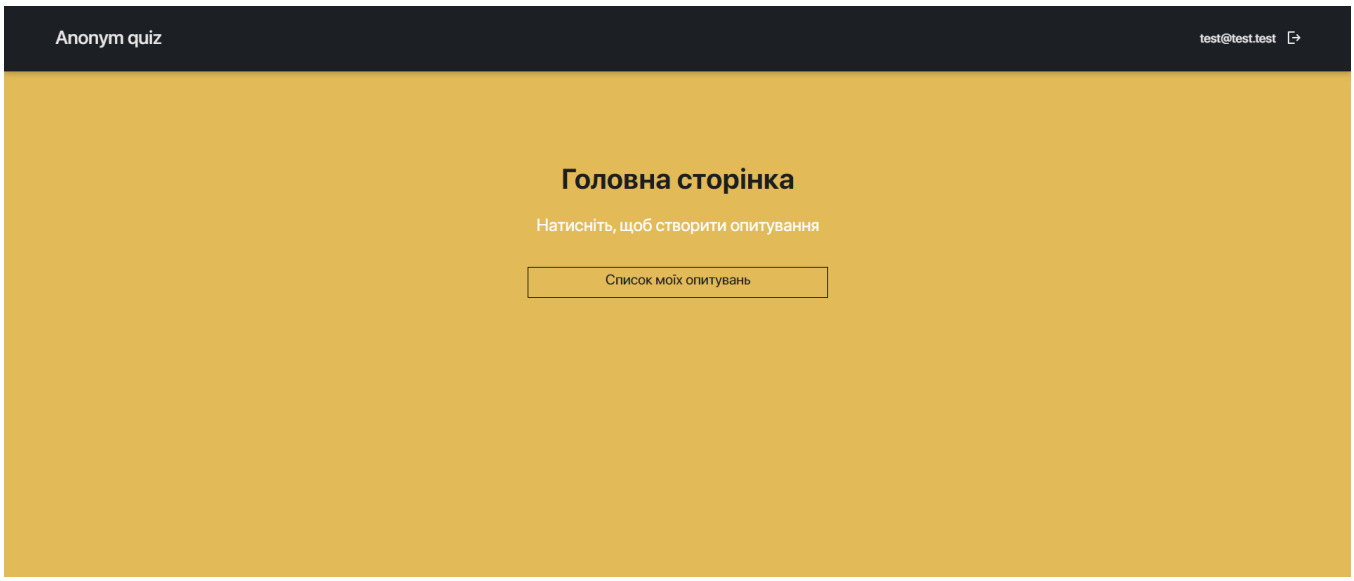

Рисунок 3.15– Головна сторінка

4) Після того як користувач захоче створити нове опитування, буде відкрита інша сторінка зі створюванням свого опитування.(Рисунок 3.16)

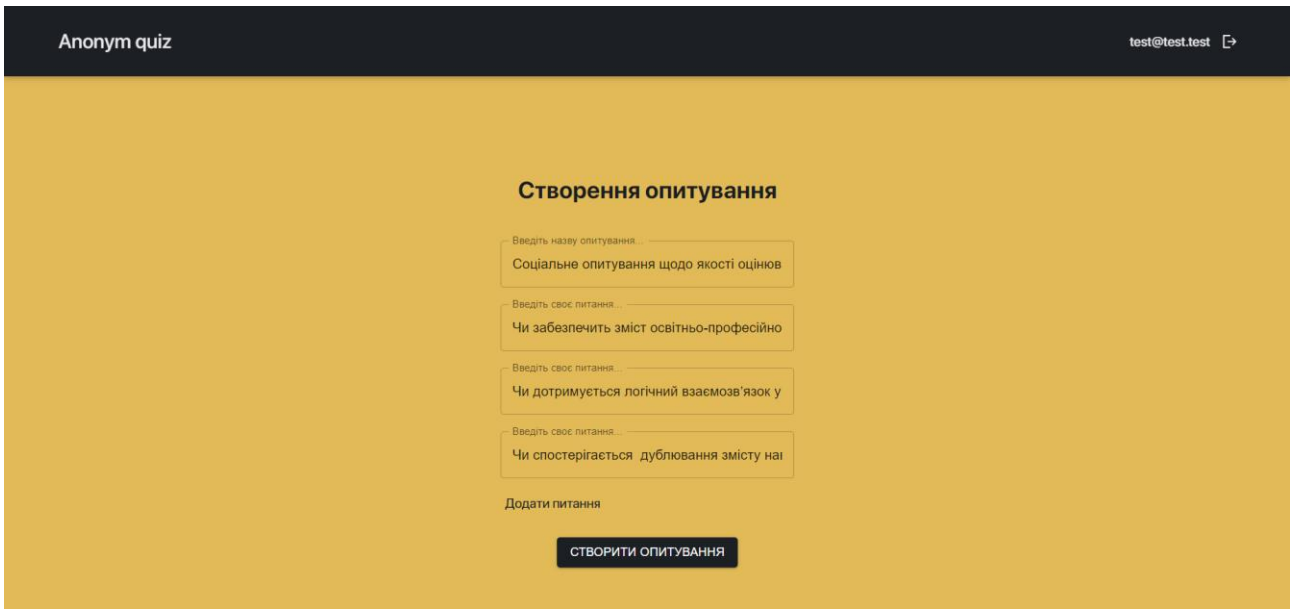

Рисунок 3.16– Сторінка створення опитування

5) Далі, сторінка із згенерованим посиланням на проходження опитування.(Рисунок 3.17)

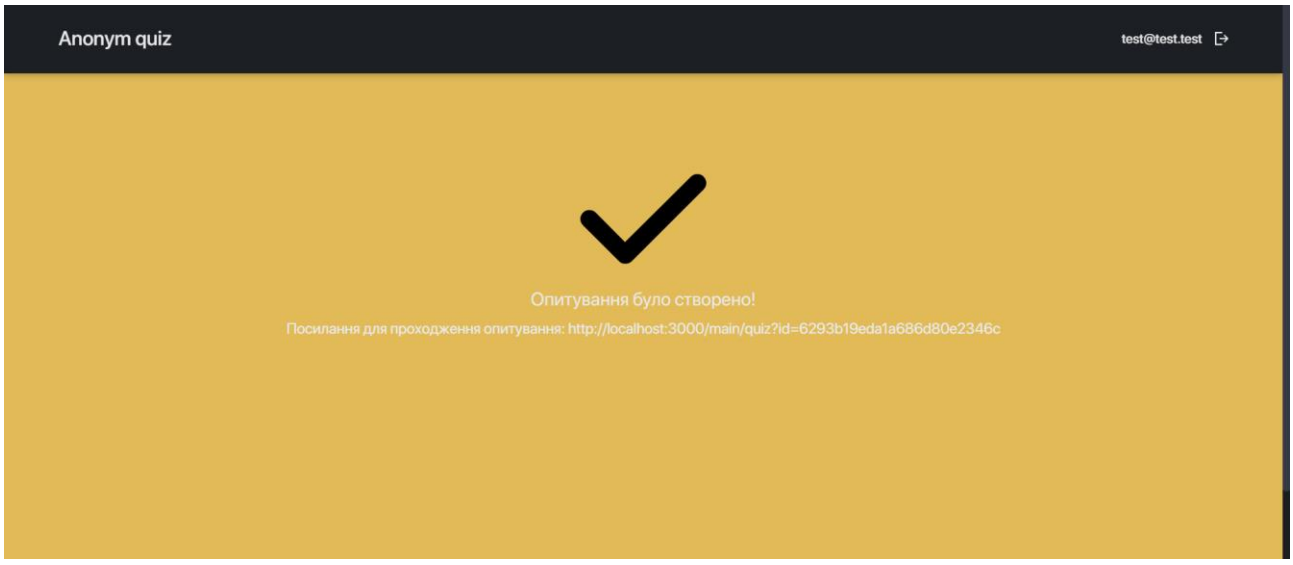

Рисунок 3.17 – Сторінка з посиланням на опитування

6) Після переходу за цим посиланням, респондент потрапляє на сторінку самого опитування. Для проходження опитування реєстрація не потрібна(Рисунок 3.18)

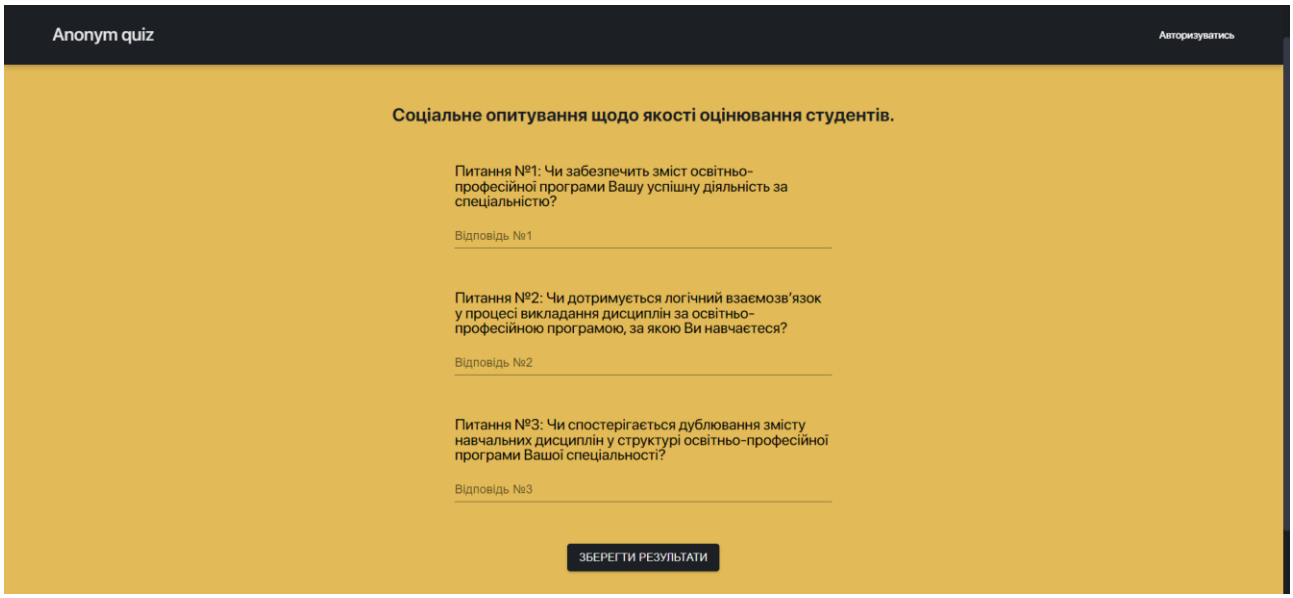

Рисунок 3.18 – Сторінка опитування

7) Після надання відповідей користувач, що створив це опитування може продивитись його результати.(Рисунок 3.19)

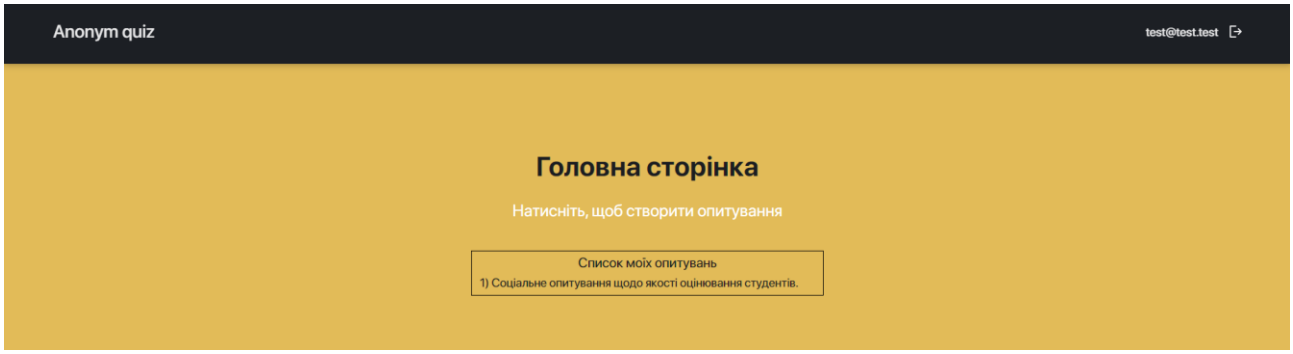

Рисунок 3.19 – Сторінка з поповненим списком створених опитувань

8) На цій сторінці можна побачити відповіді, що вже дав один респондент.(Рисунок 3.20)

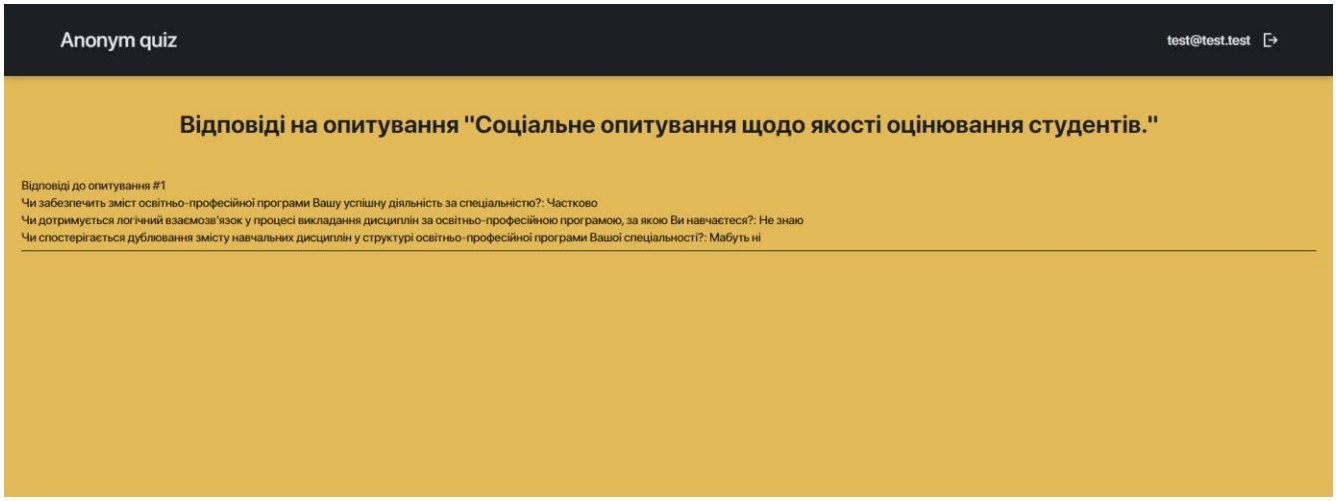

Рисунок 3.20 – Сторінка з відповідями

Цей веб-додаток зберігає повну анонімність. Для того, щоб респондент не був ідентифікований, йому не треба проходити реєстрацію при опитуванні. Достатньо лише перейти за посиланням, що створив користувач та пройти це опитування. В результаті його особа не буде ідентифікована.

В такому випадку виникає проблема. Вона полягає в тому, що будь-який респондент може проходити опитування багато разів, тим самим пошкодити збору статистики. Про цю проблему відомо, але я цим питанням не займався, тому що даний веб-додаток не включає в себе збір статистики та її обробку. Якщо треба збирати та обробляти певну статистику, тоді звісно треба заборонити користувачеві проходити опитування декілька разів.

ВИСНОВКИ

Отже, в ході виконання даної роботи було розглянуто: що таке онлайн-опитування та для чого вони взагалі потрібні, приклади вже існуючих веб-сервісів для опитування, що таке веб-додаток та їх види, шифрування та хешування.

У ході огляду шифрування були розглянуті різні види шифрування – симетричне та асиметричне. Були поверхнево розглянуті різні алгоритми, а шифр AES був розглянутий більш детально. Було прийнято рішення обрати саме цей шифр, тому що він новій та значно надійніший та кращий по всіх параметрах. У ході огляду хешування було розглянуто Алгоритм Безпечного Хешування (SHA), а також SHA-2. Далі більш детально було розглянуто алгоритм хешування SHA-256, що є різновидом алгоритму SHA-2. Зробивши висновки, було прийнято рішення обрати саме цей алгоритм. У наш час SHA-256 перемагає по всіх параметрах.

Підбиваючи підсумки теоретичної частини, спочатку було змодельовано, а потім і розроблено захищений веб-додаток з можливістю проведення анонімного опитування. В цьому додатку було реалізовано шифрування даних, що вводяться користувачем, за допомогою симетричного алгоритму шифрування AES. А також, хешування паролів за допомогою алгоритму SHA-256.

СПИСОК ЛІТЕРАТУРИ

1. Introduction to Online Surveys / Vera Toepoel // 2016. – С. 2 – 12.
2. Веб-ресурс для проведення онлайн-опитувань / SurveyMonkey [Електронний ресурс] – Режим доступу до ресурсу: https://surveymonkey.com/
3. Веб-ресурс для проведення онлайн-опитувань / Google Forms [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/intl/ru/forms/about/
4. Веб-ресурс для проведення онлайн-опитувань / Simpoll [Електронний ресурс] – Режим доступу до ресурсу: https://simpoll.ru/
5. Веб-ресурс для проведення онлайн-опитувань / Survio [Електронний ресурс] – Режим доступу до ресурсу: https://www.survio.com/ru/
6. Веб-ресурс для проведення онлайн-опитувань / TypeForm [Електронний ресурс] – Режим доступу до ресурсу: https://www.typeform.com/
7. Explorations and Surveys / Anonymous // Nabu Press, 2010. – С. 5-15.
8. Веб-ресурс для проведення простих та анонімних онлайн-опитувань / Xoyondo [Електронний ресурс] – Режим доступу до ресурсу: https://xoyondo.com/
9. Learning Web App Development: Build Quickly with Proven JavaScript Techniques / Semmy Purewal, 2014. С. 67 – 84.
10. The survey form / Arlene Fink // SAGE Publications, 2017. – С. 24 – 29.
11. Інформаційна безпека: навч. посібник / Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселічник, А.П. Бондарєв, С.С. Войтусік, А. Я. Горпенюк, О. А. Нємкова, І. М. Журавель, Б. М. Березюк, Є. І. Яковенко, В. І. Отенко, І. Я. Тишик; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. – Львів : Видавництво Львівської політехніки, 2019. – 580 с.
12. What is the AES algorithm? / Anushesh Zohair Mustafeez. [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/edpresso/what-is-the-aes-algorithm

13. Кібербезпека : сучасні технології захисту. Навчальний посібник для студентів вищих навчальних закладів. / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Львів: “Новий світ - 2000”, 2020. – 678 с.

14. The Design of Rijndael: The Advanced Encryption Standard / Joan Daemen, Vincent Rijmen // Nabu Press, 2019. – С.84-96.

15. Node.js introduction / w3schools. [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/nodejs/nodejs_intro.asp

16. Плюси та мінуси розробки додатків на Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://artjoker.ua/ru/blog/v-chem-preimushchestva-nodejs/

17. Express/Node introduction / MDN contributors. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/Introduction#

18. MongoDB / MongoDB, inc. [Електронний ресурс] - Режим доступу до ресурсу: https://www.mongodb.com/

19. Що таке ReactJS і як він працює? / Olha L. [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com.ua/rukovodstva/chto-takoe-react

ДОДАТОК А

А.1 Хешування паролів на серверній частині:

```
let userAccount = new User({
      email: request.body.email,
      password: createHash("sha256", keys.SALT)
        .update(request.body.password)
        .digest("hex"),
      username: request.body.username,
  });
```

А.2 Реалізація функції шифрування даних на серверній частині:

```
exports.sendSecretKeys = (request, response) => {
  try {
    const quizId = request.params.id;
    const generateAESKey = (quizId) => seed(quizId)(32);
    const key = generateAESKey(quizId);
    response.send({ key });
  } catch (e) {
    console.log(e);
  }
};
```

А.3 Реалізація функції шифрування даних на клієнтській частині:

```
  const encryptWord = (key, word) => {
    const bytes = aesjs.utils.utf8.toBytes(word);
    const ctr = new aesjs.ModeOfOperation.ctr(key);
    const encrypted = ctr.encrypt(bytes);
    const hex = aesjs.utils.hex.fromBytes(encrypted);
    return hex;
  };
```

ДОДАТОК Б

Б.1 Серверна частина

quiz.controller.js:

```
const db = require("../models");
const Quiz = db.quiz;
const jwt = require("jsonwebtoken");
const mongoose = require("mongoose");
const keys = require("../keys");
const seed = require("random-bytes-seed");
const Answers = db.answers;

exports.createQuiz = (req, res) => {
   try {
      jwt.verify(
         req.headers["x-access-token"],
         keys.JWT_SECRET,
         (err, decoded) => {
            if (err) {
               return res.status(401).send({ message: "Не авторізовано!" });
            }
            const quiz = new Quiz({
               questions: req.body.questions,
               creatorId: mongoose.Types.ObjectId(decoded.id),
               name: req.body.name,
            });
            quiz.save((err, quiz) => {
               if (err) {
                  res.status(500).send({ message: err });
                  return;
               }
               res.send({ quizId: quiz.id });
            });
         }
      );
   } catch (e) {
      console.log(e);
   }
};

exports.getQuiz = (req, res) => {
   try {
      const id = req.query.id;

      Quiz.findById(mongoose.Types.ObjectId(id))
         .populate("-__v")
         .exec((err, quiz) => {
```

```
            res.status(200).send(quiz);
        });
    } catch (e) {
        console.log(e);
    }
};

exports.answerQuiz = (req, res) => {
    try {
        const answers = new Answers({
            quizId: mongoose.Types.ObjectId(req.params.id),
            answers: req.body.answers,
        });
        answers.save((err) => {
            if (err) {
                res.status(500).send({ message: err });
                return;
            }
            res.status(201).send("Success!");
        });
    } catch (e) {
        console.log(e);
    }
};

exports.getMyQuizes = (req, res) => {
    try {
        jwt.verify(
            req.headers["x-access-token"],
            keys.JWT_SECRET,
            (err, decoded) => {
                if (err) {
                    return res.status(401).send({ message: "Не авторізовано!" });
                }
                Quiz.find({ creatorId: mongoose.Types.ObjectId(decoded.id) })
                    .populate("-__v")
                    .exec((err, results) => {
                        res.status(200).send(results);
                    });
            }
        );
    } catch (e) {
        console.log(e);
    }
};

exports.getAnswers = (req, res) => {
    try {
        jwt.verify(
            req.headers["x-access-token"],
```

```
            keys.JWT_SECRET,
            (err, decoded) => {
                if (err) {
                    return res.status(401).send({ message: "Не авторізовано!" });
                }
                Quiz.findById(mongoose.Types.ObjectId(req.params.id))
                    .populate("-__v")
                    .exec((err, quiz) => {
                        Answers.find({"quizId": mongoose.Types.ObjectId(req.params.id)})
                            .populate("-__v")
                            .exec((err, results) => {
                                if (err) {
                                    res.status(500).send({ message: err });
                                    return;
                                }
                                res.status(200).send({...quiz._doc, answers: results.map(el => el.answers)})
                            });
                    });
            }
        );
    } catch (e) {
        console.log(e);
    }
};
```

Answers.js:

```
const { model, Schema } = require("mongoose");

const Answers = model(
    "Answers",
    new Schema({
        quizId: {
            type: Schema.Types.ObjectId, ref: "Quiz",
            required: true,
        },
        answers: [
            {
                type: String,
            },
        ],
    })
);

module.exports = Answers;
```

quiz.js:

```
const { model, Schema } = require("mongoose");
```

```
const Quiz = model(
    "Quiz",
    new Schema({
        creatorId: {
            type: Schema.Types.ObjectId, ref: "User",
            required: true,
        },
        questions: [
            {
                type: String,
            },
        ],
        name: {
            type: String,
            required: true,
        },
    })
);

module.exports = Quiz;
```

quiz.js:

```
const { authJwt } = require("../middlewares");
const controller = require("../controllers/quiz.controller");

module.exports = function (app) {
    app.post("/api/createQuiz", [authJwt.verifyToken], controller.createQuiz);
    app.get("/api/myQuizes", [authJwt.verifyToken], controller.getMyQuizes);
    app.get("/api/answers/:id", [authJwt.verifyToken], controller.getAnswers);
    app.get("/api/getQuiz", controller.getQuiz);
    app.post("/api/quiz/:id", controller.answerQuiz);
    app.post("/api/quiz/:id/keys", controller.sendKeys);
};
```

Б.2 Клієнтська частина

QuizActions.jsx:

```
import { QUIZ } from "../constants";

export function createQuiz(data) {
    return {
        type: QUIZ.QUIZ,
        payload: {
            client: "default",
            request: {
                url: `createQuiz`,
                method: "post",
```

```
            data,
        },
    },
  };
}

export function getQuiz(id) {
  return {
    type: QUIZ.GET_QUIZ,
    payload: {
      client: "default",
      request: {
        url: `getQuiz?id=${id}`,
        method: "get",
      },
    },
  };
}

export function postKey(id) {
  return {
    type: QUIZ.POST_KEY,
    payload: {
      client: "default",
      request: {
        url: `/quiz/${id}/keys`,
        method: "post",
      },
    },
  };
}

export function answerQuiz(id, data) {
  return {
    type: QUIZ.ANSWER_QUIZ,
    payload: {
      client: "default",
      request: {
        url: `/quiz/${id}`,
        method: "post",
        data,
      },
    },
  };
}

export function getMyQuizes() {
  return {
    type: QUIZ.GET_MY_QUIZES,
    payload: {
```

```
            client: "default",
            request: {
                url: `/myQuizes`,
                method: "get",
            },
        },
    };
}

export function getAnswers(id) {
    return {
        type: QUIZ.GET_ANSWERS,
        payload: {
            client: "default",
            request: {
                url: `/answers/${id}`,
                method: "get",
            },
        },
    };
}
```

Answers.jsx:

```
import { connect } from "react-redux";
import { getAnswers, postKey } from "../../actions/quizActions";
import { useEffect, useState } from "react";
import aesjs from "aes-js";
import "./Answers.scss";

const Answers = ({
    getAnswers,
    location: { search },
    data,
    postKey,
    history,
}) => {
    const [key, setKey] = useState(null);

    useEffect(() => {
        const id = new URLSearchParams(search).get("id");
        getAnswers(id);
        postKey(id).then((res) => {
        setKey(res.payload.data.aesKey.data);
            } else {
                history.push("/main");
            }
        });
    }, []);
```

```
    const { questions, answers } = data;

    return (
        <div className="answers-wrapper">
            <h1>Відповіді на опитування "{data?.name}"</h1>
            {answers && answers.length > 0 ? (
                answers.map((ans, ansIdx) => {
                    return (
                        <div className="single-answer">
                            <p>
                                Відповіді до опитування #{ansIdx + 1}
                            </p>
                            {questions?.map((el, idx) => {
                                return (
                                    <div>
                                        <span>{el}</span>: <span>{decrypt(ans[idx], key)}</span>
                                    </div>
                                );
                            })}
                        </div>
                    );
                })
            ) : (
                <p>Відповідей ще немає.</p>
            )}
        </div>
    );
};

const mapStateToProps = ({ quiz }) => {
    return {
        data: quiz.answers,
    };
};

const mapDispatchToProps = {
    getAnswers,
    postKey,
};

export default connect(mapStateToProps, mapDispatchToProps)(Answers);
```

CreateQuiz.jsx:

```
import Button from "@mui/material/Button";
import "./CreateQuiz.scss";
import { connect } from "react-redux";
import { createQuiz } from "../../actions/quizActions";
import { useState } from "react";
import TextField from "@mui/material/TextField";
```

```
const CreateQuiz = ({ createQuiz, history }) => {
    const [questions, setQuestions] = useState([""]);
    const [name, setName] = useState("");

    return (
        <div className="create-quiz-wrapper">
            <h1>Створення опитування</h1>
            <div className="inputs-wrapper">
                <TextField
                    id={"quiz-name"}
                    sx={{
                        width: "100%",
                        marginBottom: "20px",
                    }}
                    label="Введіть назву опитування..."
                    variant="outlined"
                    value={name}
                    onChange={(e) => setName(e.target.value)}
                />
                {questions.map((el, idx) => {
                    return (
                        <TextField
                            id={idx}
                            sx={{
                                width: "100%",
                                marginBottom: "20px",
                            }}
                            label="Введіть своє питання..."
                            variant="outlined"
                            value={questions[idx]}
                            onChange={(e) =>
                                setQuestions((value) => {
                                    return [
                                        ...value.slice(0, idx),
                                        e.target.value,
                                        ...value.slice(idx + 1),
                                    ];
                                })
                            }
                        />
                    );
                })}

                <p
                    className="add-question"
                    onClick={() =>
                        setQuestions((value) => {
                            return [...value, ""];
                        })
```

```
                }
              >
                Додати питання
              </p>
            </div>
            <Button
              variant="contained"
              onClick={() =>
                createQuiz({ questions, name }).then((data) =>
                  history.push(`success-quiz?id=${data.payload.data.quizId}`)
                )
              }
              sx={{
                backgroundColor: "#1c1f24",
                "&:hover": {
                  backgroundColor: "#1c1f24",
                },
                "&:active": {
                  backgroundColor: "#1c1f24",
                },
              }}
            >
              Створити опитування
            </Button>
          </div>
        );
};

const mapDispatchToProps = {
    createQuiz,
};

export default connect(null, mapDispatchToProps)(CreateQuiz);
```

MainPage.jsx:

```
import { connect } from "react-redux";
import { Link } from "react-router-dom";
import "./MainPage.scss";
import { getMyQuizes } from "../../actions/quizActions";
import { useEffect } from "react";

const MainPage = ({ history, getMyQuizes, data }) => {
    useEffect(() => {
        if (localStorage.token) {
            getMyQuizes();
        }
    }, []);
```

```
    return (
        <div className="main-page-wrapper">
            <h1>Головна сторінка</h1>
            {localStorage.token ? (
                <div className="main-with-token">
                    <button
                        onClick={() => {
                            history.push("/main/create-quiz");
                        }}
                    >
                        Натисніть, щоб створити опитування
                    </button>
                    <div>
                        <p>Список моїх опитувань</p>
                        {data.map((el, idx) => (
                            <Link to={`/main/answers?id=${el._id}`} key={el._id}>
                                <p className="">
                                    {idx + 1}) {el.name}
                                </p>
                            </Link>
                        ))}
                    </div>
                </div>
            ) : (
                <div className="main-without-token">
                    <p>Щоб створити опитування потрібно авторизуватись!</p>
                    <button
                        onClick={() => {
                            history.push("/auth/login");
                        }}
                    >
                        Авторизуватись
                    </button>
                </div>
            )}
        </div>
    );
};

const mapStateToProps = ({ quiz }) => {
    return {
        data: quiz.myQuizes,
    };
};

const mapDispatchToProps = {
    getMyQuizes,
};

export default connect(mapStateToProps, mapDispatchToProps)(MainPage);
```

Quiz.jsx:

```
import "./Quiz.scss";
import { useState, useEffect } from "react";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import Image from "../../assets/image/check-solid.svg";
import aesjs from "aes-js";
import { connect } from "react-redux";
import { getQuiz, postKey, answerQuiz } from "../../actions/quizActions";

const Quiz = ({
    getQuiz,
    location: { search },
    quiz,
    postKey,
    history,
    answerQuiz,
}) => {
    const [answers, setAnswers] = useState([]);
    const [key, setKey] = useState(null);
    const [isSaved, setIsSaved] = useState(false);

    useEffect(() => {
        const id = new URLSearchParams(search).get("id");
        getQuiz(id);

        postKey(id).then((res) => {
            setKey(res.payload.data.aesKey.data);
            } else {
                history.push("/main");
            }
        });
    }, []);

    const { questions, name } = quiz;

    useEffect(() => {
        console.log(quiz, 123);
        let arr = [];

        if (Object.keys(quiz).length !== 0) {
            arr = quiz.questions;
        }
        setAnswers([...arr.map((el) => "")]);
    }, [quiz]);

    const handleChange = (e, idx) => {
```

```
    setAnswers((value) => [
      ...value.slice(0, idx),
      e.target.value,
      ...value.slice(idx + 1),
    ]);
  };

  const handleSubmit = (key, data) => {
    const arr = data.map((el) => encrypt(key, el));
    const id = new URLSearchParams(search).get("id");
    answerQuiz(id, { answers: arr }).then((res) => {
      if (
        res.payload &&
        res.payload.status &&
        res.payload.status === 201
      ) {
        setIsSaved(true);
      }
    });
  };

  return (
    <div className="quiz-wrapper">
      <h1>{name}</h1>
      {isSaved ? (
        <>
          <img src={Image} />
          <p className="saved">Ваші відповіді збережені!</p>
        </>
      ) : (
        <>
          {questions?.map((el, idx) => (
            <div key={idx}>
              <p>
                Питання №{idx + 1}: {el}
              </p>
              <TextField
                id={idx}
                label={`Відповідь №${idx + 1}`}
                variant="standard"
                value={answers[idx]}
                sx={{
                  width: "100%",
                }}
                onChange={(e) => handleChange(e, idx)}
              />
            </div>
          ))}
          <Button
            variant="contained"
```

```
                    onClick={() => handleSubmit(key, answers)}
                    sx={{
                        backgroundColor: "#1c1f24",
                        "&:hover": {
                            backgroundColor: "#1c1f24",
                        },
                        "&:active": {
                            backgroundColor: "#1c1f24",
                        },
                    }}
                >
                    Зберегти результати
                </Button>
            </>
        )}
    </div>
  );
};

const mapStateToProps = ({ quiz }) => {
  return {
    quiz: quiz.quiz,
  };
};

const mapDispatchToProps = {
  getQuiz,
  postKey,
  answerQuiz,
};

export default connect(mapStateToProps, mapDispatchToProps)(Quiz);
```

SuccessQuiz.jsx:

```
import Image from "../../assets/image/check-solid.svg";
import "./SuccessQuiz.scss";

const SuccessQuiz = ({ location: { search } }) => {
  const {
    location: { origin },
  } = window;

  return (
    <div className="success-quiz-wrapper">
      <img src={Image} />
      <p className="saved">Опитування було створено!</p>
      <div className="success-link">
        <span>Посилання для проходження опитування: </span>
```

```
            <a
               target="_blank"
               href={`${origin}/main/quiz${search}`}
            >{`${origin}/main/quiz${search}`}</a>
         </div>
      </div>
   );
};

export default SuccessQuiz;
```

reduceQuiz:

```
import { QUIZ } from "../constants";

const INITIAL_STATE = {
   quiz: {},
   myQuizes: [],
   answers: {},
};

export default function (state = INITIAL_STATE, action) {
   switch (action.type) {
      case QUIZ.GET_QUIZ_SUCCESS:
         return { ...state, quiz: action.payload.data };
      case QUIZ.GET_MY_QUIZES_SUCCESS:
         return { ...state, myQuizes: action.payload.data };
      case QUIZ.GET_ANSWERS_SUCCESS:
         return { ...state, answers: action.payload.data };
      default:
         return state;
   }
}
```