МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
(повна назва випускної кафедри)

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**на тему:**

# СИСТЕМА ОНЛАЙН-ОПИТУВАНЬ З АВТОМАТИЧНИМ ПІДРАХУНКОМ РЕЗУЛЬТАТІВ

Виконав:
студент групи 1КІ-23 зі спеціальності
123 – «Комп'ютерна інженерія»
Владислав ГОЛОВЧЕНКО
Науковий керівник:
к.т.н Роксолана БРЕУС

Черкаси, 2025

# АНОТАЦІЯ

У кваліфікаційній роботі бакалавра на тему "Розробка системи онлайн-опитувань для автоматизації збору та аналізу даних" запропоновано комплексне дослідження та створення веб-додатку survey-system для організації, проведення та аналізу онлайн-опитувань із підтримкою авторизації, публічного доступу та аналітики результатів.

Основне завдання дослідження – аналіз предметної області онлайн-опитувань, вивчення сучасних технологій розробки веб-додатків і баз даних, проєктування архітектури системи, розробка та впровадження функціоналу для створення, проходження й аналізу опитувань, тестування системи, оцінка її ефективності та визначення перспектив розвитку.

У роботі проведено огляд існуючих платформ для онлайн-опитувань (Google Forms, SurveyMonkey, Typeform), визначено їхні переваги та недоліки, що дозволило сформулювати вимоги до survey-system. Запропоновано та реалізовано рішення на основі технологічного стеку Next.js, PostgreSQL, NextAuth, Prisma та Tailwind CSS. Розроблено реляційну базу даних із таблицями для користувачів, тем, опитувань, питань, варіантів відповідей та результатів, а також інтерактивний інтерфейс із підтримкою авторизації, публічного доступу до опитувань і аналітики. Тестування підтвердило працездатність системи, включаючи створення опитувань адміністраторами, проходження їх користувачами та автоматичний підрахунок результатів. Впровадження системи забезпечило зручний збір даних і їх аналіз, підвищивши ефективність опитувань.

Кваліфікаційна робота бакалавра містить 71 аркуш пояснювальної записки, 14 рисунків, 14 таблиць та 3 додатки пояснювальної записки.

Ключові слова: ОНЛАЙН-ОПИТУВАННЯ, ВЕБ-ДОДАТОК, АВТОМАТИЗАЦІЯ ЗБОРУ ДАНИХ, РЕЛЯЦІЙНА БАЗА ДАНИХ, АВТОРИЗАЦІЯ, АНАЛІТИКА, NEXT.JS, MYSQL.

# ABSTRACT

The bachelor's thesis on the topic “Development of an online survey system for automating data collection and analysis” proposes a comprehensive study and creation of a survey-system web application for organizing, conducting and analyzing online surveys with support for authorization, public access and results analysis.

The main objective of the study is to analyze the subject area of online surveys, study modern technologies for developing web applications and databases, design the system architecture, develop and implement functionality for creating, completing, and analyzing surveys, test the system, evaluate its effectiveness, and determine development prospects.

The paper reviews existing platforms for online surveys (Google Forms, SurveyMonkey, Typeform), identifies their advantages and disadvantages, and formulates requirements for a survey system. A solution based on the Next.js, PostgreSQL, NextAuth, Prisma, and Tailwind CSS technology stack was proposed and implemented. A relational database with tables for users, topics, surveys, questions, answer options, and results, as well as an interactive interface with support for authorization, public access to surveys, and analytics were developed. Testing has confirmed the system's performance, including the creation of surveys by administrators, their completion by users, and automatic calculation of results. The implementation of the system provided convenient data collection and analysis, increasing the efficiency of surveys.

The bachelor's thesis contains 71 pages of explanatory notes, 14 figures, 14 tables, and 3 appendices of explanatory notes.



Keywords: ONLINE SURVEY, WEB APPLICATION, DATA COLLECTION AUTOMATION, RELATIONAL DATABASE, AUTHORIZATION, ANALYTICS, NEXT.JS, MYSQL.

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | |
|---|---|
| **UI** | User Interface (Користувацький інтерфейс) |
| **UX** | User Experience (Користувацький досвід) |
| **SEO** | Search Engine Optimization (Оптимізація для пошукових систем) |
| **API** | Application Programming Interface (Інтерфейс програмування додатків) |
| **CRM** | Customer Relationship Management (Управління відносинами з клієнтами) |
| **ACID** | Atomicity, Consistency, Isolation, Durability (Атомарність, Консистентність, Ізоляція, Надійність) |
| **3NF** | Third Normal Form (Третя нормальна форма) |
| **WCAG 2.1** | Web Content Accessibility Guidelines 2.1 (Рекомендації з доступності веб-контенту) |
| **SSR** | Server-Side Rendering (Серверний рендеринг) |
| **SSG** | Static Site Generation (Статична генерація сайту) |
| **CSR** | Client-Side Rendering (Клієнтський рендеринг) |
| **PK** | Primary Key (Первинний ключ) |
| **FK** | Foreign Key (Зовнішній ключ) |
| **JSON** | JavaScript Object Notation |
| **JWT** | JSON Web Token |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **CSRF** | Cross-Site Request Forgery (Міжсайтова підробка запитів) |
| **2FA** | Two-Factor Authentication (Двофакторна автентифікація) |
| **npm** | Node Package Manager |

# ЗМІСТ

# ВСТУП

Актуальність теми. В сучасному цифровому середовищі онлайн-опитування є ефективним інструментом збору, аналізу та обробки інформації. Вони широко використовуються в різних сферах, зокрема в освіті, маркетингових дослідженнях, соціологічних опитуваннях та внутрішніх опитуваннях підприємств. Проте багато існуючих платформ для проведення опитувань потребують додаткової ручної обробки отриманих даних, що уповільнює процес аналізу та збільшує ймовірність помилок. Саме тому актуальним є створення системи онлайн-опитувань, яка забезпечить автоматичний підрахунок результатів та їхню зручну візуалізацію.

Метою дослідження є розробка системи онлайн-опитувань, яка дозволить користувачам створювати, проходити та аналізувати опитування з автоматичним підрахунком результатів, що підвищить ефективність збору та обробки даних.

Об'єктом дослідження є процес організації та проведення онлайн-опитувань.

Предметом дослідження є методи та засоби розробки веб-системи для автоматизованого збору й обробки відповідей у рамках онлайн-опитувань.

Завдання дослідження:

- аналіз існуючих рішень для проведення онлайн-опитувань та визначення їхніх переваг та недоліків;
- дослідження методів автоматичного підрахунку результатів у системах опитувань;
- обґрунтування вибору технологій для розробки системи;
- проєктування архітектури системи, включаючи базу даних, бекенд та фронтенд;

- - реалізація системи онлайн-опитувань із підтримкою автоматичного підрахунку результатів;
- - тестування системи для оцінки її ефективності та зручності використання;
- - розробка інструкції користувача для коректного використання функціоналу системи.

Розроблена система сприятиме оптимізації процесу створення та аналізу опитувань, дозволяючи користувачам швидко отримувати релевантні результати без необхідності в ручній обробці даних.

# РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ТЕОРЕТИЧНІ ОСНОВИ

## 1.1 Поняття та види онлайн-опитувань

Онлайн-опитування – це метод збору інформації за допомогою цифрових технологій, що дозволяє отримувати відповіді від респондентів через інтернет. Вони є ефективним засобом аналізу думок, потреб і поведінки людей у різних сферах діяльності, таких як соціологія, маркетинг, освіта, внутрішньокорпоративні дослідження та інші [1].

У сучасному світі онлайн-опитування набули широкого поширення завдяки своїм численним перевагам, зокрема:

- доступність – користувачі можуть проходити опитування в будь-який зручний час і з будь-якого пристрою з доступом до інтернету;
- швидкість збору даних – автоматизовані системи дозволяють миттєво отримувати та аналізувати відповіді;
- гнучкість налаштувань – можливість створення різних форматів запитань, умовних переходів між ними та персоналізації опитувань;
- зниження витрат – онлайн-опитування виключають необхідність друку паперових анкет, використання кур’єрських послуг або найму інтерв’юерів;
- анонімність – у деяких випадках респонденти можуть заповнювати анкети без розкриття особистої інформації, що підвищує чесність відповідей [2].

Онлайн-опитування можуть бути створені за допомогою спеціалізованих платформ (наприклад, Google Forms, SurveyMonkey, Typeform), вбудовані у веб-сайти або ж функціонувати у вигляді окремих інформаційних систем [3].

Ключовою особливістю сучасних систем онлайн-опитувань є можливість автоматичного підрахунку результатів, що значно спрощує процес аналізу даних та дозволяє швидко отримувати необхідну інформацію для прийняття рішень.

Існує кілька підходів до класифікації онлайн-опитувань, залежно від їхньої мети, форми подання запитань та методів обробки результатів. Детальний огляд класифікації наведено в додатку А.1.

Онлайн-опитування є універсальним інструментом для збору інформації у різних сферах діяльності. Завдяки можливості автоматизованого підрахунку результатів вони значно прискорюють процес аналізу отриманих даних, підвищують точність та мінімізують людський фактор [4].

Розробка ефективної системи онлайн-опитувань із підтримкою автоматичного підрахунку результатів дозволить значно спростити процес створення, проведення та аналізу опитувань, що є особливо актуальним для великих організацій, освітніх установ та маркетингових компаній.

### 1.2 Огляд існуючих рішень для онлайн-опитувань

У сучасному цифровому середовищі існує безліч платформ, які надають можливість створювати та проводити онлайн-опитування з автоматичним підрахунком результатів. Розрізняють наступні платформи [5]:

1. Google Forms (рис. 1.1) – це безкоштовний сервіс для створення онлайн-опитувань та збору відповідей, який інтегрований в екосистему Google, використовується для проведення опитувань, анкетування, тестування та збору зворотного зв’язку. Переваги і недолівки Google Forms показано в табл.1.1 [6].

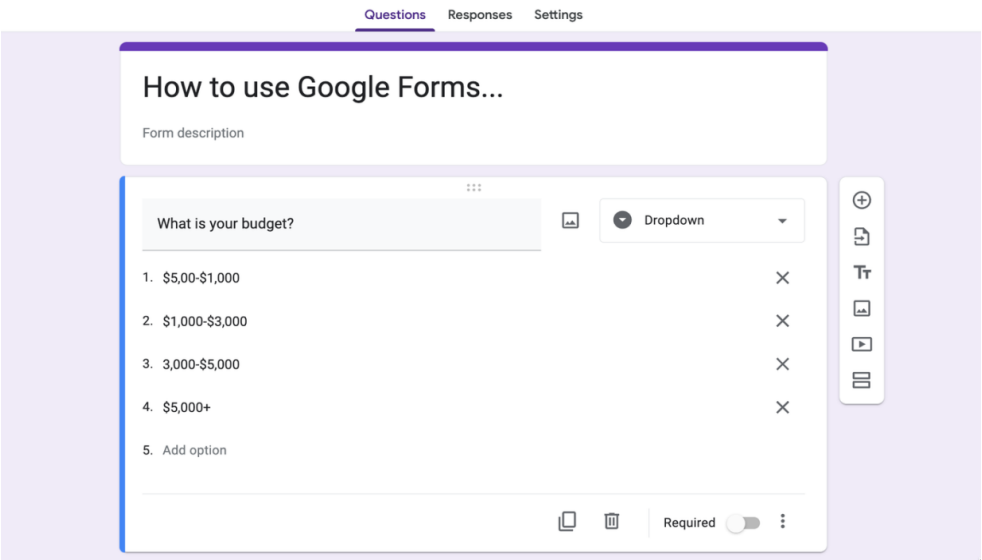


Рисунок 1.1 – Сервіс Google Forms

Можливості Google Forms [7]:

- різні типи запитань (текстові, вибіркові, шкали тощо);
- автоматичний підрахунок відповідей і створення графіків;
- інтеграція з Google Sheets для аналізу даних;
- підтримка спільного редагування;
- доступний на будь-яких пристроях без встановлення додаткового ПЗ.

Таблиця 1.1 – Переваги і недолівки Google Forms

| № | Переваги | Недоліки |
|---|---|---|
| 1 | Безкоштовність | Обмежені можливості дизайну та кастомізації |
| 2 | Простий у використанні інтерфейс | Відсутність складної логіки запитань |
| 3 | Автоматизація аналізу даних через Google Sheets | Потрібен Google-акаунт для створення опитувань |
| 4 | Підтримка роботи в команді | |

2. SurveyMonkey (рис. 1.2) – одна з найпопулярніших платформ для створення онлайн-опитувань. Вона дозволяє розробляти детальні анкети з можливістю глибокого аналізу відповідей. Переваги і недоліки SurveyMonkey SurveyMonkey показано в табл.1.3 [8].

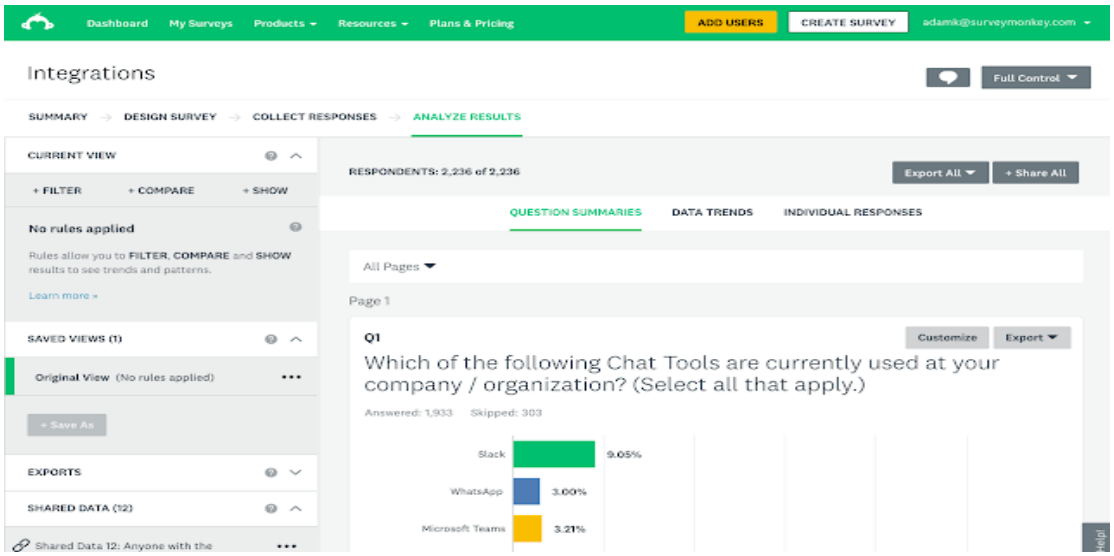


Рисунок 1.2 – Сервіс SurveyMonkey

Можливості SurveyMonkey:

- великий вибір типів запитань (від простих до складних логічних схем);
- автоматичний підрахунок та аналіз відповідей;
- інтеграція з CRM-системами, маркетинговими інструментами (HubSpot, Mailchimp тощо);
- можливість імпорту та експорту даних;
- готові шаблони та варіанти дизайну.

Таблиця 1.2 – Переваги і недоліки SurveyMonkey

| № | Переваги | Недоліки |
|---|---|---|
| 1 | Гнучка логіка опитувань (умовні переходи, фільтрація відповідей) | Безкоштовна версія має обмеження на кількість відповідей |
| 2 | Вбудований аналітичний інструмент | Більшість розширених функцій доступні лише у платній підписці |
| 3 | Інтеграція з іншими сервісами | Досить складний для новачків |
| 4 | Високий рівень безпеки даних | |

3. Typeform відрізняється стильним і сучасним дизайном. Опитування у цьому сервісі виглядають як інтерактивні розмови, що підвищує залученість респондентів (див. табл.1.3) [9].

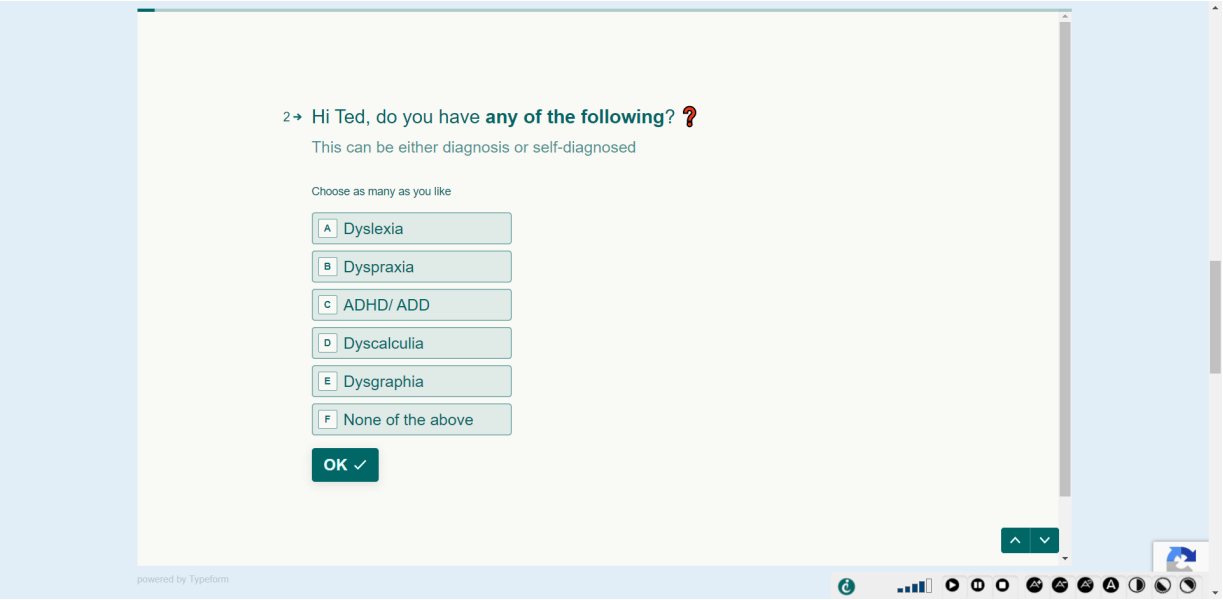


Рисунок.1.3 – Сервіс Typeform

Можливості Typeform:

- інтерактивний формат подання питань;
- вбудована логіка запитань і умовні переходи;
- автоматичний підрахунок відповідей;
- інтеграція з інструментами на кшталт Zapier, Slack, Google Sheets;

- можливість додавання зображень, відео, GIF.

Таблиця 1.3 – Переваги і недолівки Typeform

| № | Переваги | Недоліки |
|---|---|---|
| 1 | Сучасний дизайн, що підвищує взаємодію користувачів | Обмежена кількість відповідей у безкоштовній версії |
| 2 | Гнучка логіка і розгалуження питань | Вища вартість у порівнянні з конкурентами |
| 3 | Можливість створення кастомних форм | Не підходить для опитувань з великою кількістю питань |
| 4 | Інтуїтивно зрозумілий інтерфейс | |

Таблиця порівняння існуючих рішень за різними критеріями показано в додатку А.2.

Google Forms – найкраще рішення для безкоштовного використання, якщо потрібно швидко зібрати дані без складних налаштувань. SurveyMonkey підійде для бізнесу та професійних досліджень завдяки розширеним можливостям аналізу та логіки запитань. Typeform виділяється сучасним дизайном та інтерактивним підходом, що робить його ідеальним для опитувань, спрямованих на залучення користувачів [10].

## 1.3 Особливості автоматичного підрахунку результатів

Автоматичний підрахунок результатів є важливим компонентом системи онлайн-опитувань, що дозволяє спростити обробку відповідей, підвищити швидкість аналізу та зменшити вплив людського фактора на результати. Його застосування забезпечує ефективний збір та аналіз даних у таких сферах, як освіта, маркетингові дослідження, соціологія, HR-рекрутинг тощо.

Система автоматичного підрахунку результатів працює за рахунок алгоритмів обробки даних, збереження відповідей у структурованому вигляді та їх аналізу за визначеними правилами. Основними аспектами автоматизації є правильний вибір алгоритмів підрахунку, методів обробки та зберігання даних [11].

Залежно від типу опитування та його цілей застосовуються різні методи обробки відповідей. Основні підходи обробки відповідей наведено у табл.1.4.

Таблиця 1.4 – Методи обробки відповідей

| № | Метод | Опис | Приклад використання | Переваги | Недоліки |
|---|---|---|---|---|---|
| 1 | Підрахунок балів (Scoring System) [16] | Використовується у тестах з фіксованими правильними відповідями. Кожна відповідь оцінюється у балах відповідно до заданих правил. | Тестування знань, сертифікаційні тести | Простий у реалізації, швидке визначення результатів | Не підходить для відкритих питань та анкет |
| 2 | Аналіз анкетних даних (Survey Analysis) [16] | Опитування без правильних відповідей, де аналізуються статистичні показники (середнє значення, мода, медіана, частотний аналіз) | Соціологічні опитування, маркетингові дослідження | Гнучкість аналізу, можливість виявлення трендів | Вимагає додаткових методів інтерпретації |
| 3 | Визначення тенденцій та кластеризація [16] | Використання машинного навчання для виявлення закономірностей у відповідях | Аналіз поведінки користувачів, персоналізовані рекомендації | Можливість обробки великих обсягів даних, глибокий аналіз | Високі вимоги до обчислювальних ресурсів |

Для автоматичного підрахунку результатів використовуються різні алгоритми [12], які залежать від типу опитування та методів аналізу даних. У додатку А.3 наведено основні алгоритми та їх характеристики.

Для ефективної обробки результатів опитувань необхідно правильно організувати збереження даних. Вибір бази даних залежить від типу даних, що обробляються (див. табл.1.8).

Таблиця 1.5 – Збереження та обробка результатів

| № | Тип бази даних | Опис | Переваги | Недоліки |
|---|---|---|---|---|
| 1 | Реляційні бази даних (MySQL, PostgreSQL) | Використовують таблиці для зберігання структурованих даних | Висока швидкість запитів, ACID-транзакції | Менш ефективні при збереженні великих текстових відповідей |
| 2 | NoSQL-бази даних (MongoDB, Firebase) | Зберігають дані у вигляді JSON-документів | Гнучкість, масштабованість | Менше можливостей для складних запитів |
| 3 | Хмарні сервіси (Google Sheets, BigQuery) [18] | Використовуються для онлайн-аналізу та інтеграції з іншими інструментами | Зручність доступу, інтеграція з BI-системами | Можливі обмеження продуктивності |

Автоматичний підрахунок результатів онлайн-опитувань супроводжується візуалізацією даних , що допомагає краще зрозуміти закономірності та розподіл відповідей показано в табл. 1.6.

Таблиця 1.6 – Візуалізація та інтерпретація результатів

| № | Метод візуалізації | Опис | Приклад використання |
|---|---|---|---|
| 1 | Гістограма | Показує розподіл значень відповідей | Аналіз балів у тестуванні |
| 2 | Кругова діаграма | Відображає відсоткове співвідношення відповідей | Маркетингові опитування |
| 3 | Лінійний графік | Показує динаміку змін у часі | Аналіз тенденцій у великих анкетах |
| 4 | Теплова карта | Візуалізує інтенсивність вибору варіантів | Аналіз поведінкових патернів [19] |

Автоматичний підрахунок результатів онлайн-опитувань базується на різних методах аналізу даних – від простого підрахунку балів до складних алгоритмів машинного навчання. Використання статистичних методів та візуалізації допомагає отримати цінну інформацію з відповідей респондентів. Вибір алгоритму обробки та бази даних залежить від типу опитування та необхідної точності аналізу [13].

### 1.4 Вимоги до сучасних онлайн опитувань

Система онлайн-опитувань з автоматичним підрахунком результатів повинна бути доступною для широкого кола користувачів і забезпечувати можливість ефективного збору та обробки відповідей на опитування. Вона повинна містити основні функціональні можливості для створення, проходження та аналізу результатів тестів/опитувань з автоматичним підрахунком результатів. Це забезпечить спрощення процесу проведення опитувань, дозволяючи зосередитись на аналізі отриманих даних замість ручної обробки результатів [14].

Функціональні вимоги до системи:

1. Реєстрація та авторизація користувачів

- При реєстрації користувачі повинні мати можливість зареєструватися в системі, вказавши необхідні дані (електронна пошта, ім'я, пароль).

- При авторизації користувачі повинні мати можливість авторизуватися в системі за допомогою введення логіна та пароля [15].

2. Створення опитувань

- Типи опитувань:

a) Відкрите– опитування, яке доступне для всіх користувачів.

b) Закрите – опитування доступне лише тим користувачам, які мають спеціальне посилання.

- Додавання запитань – ористувач повинен мати можливість додавати різні типи запитань (один варіант відповіді, кілька варіантів, відкриті питання).

- Редагування опитувань – можливість редагувати або видаляти опитування після його створення.

- Налаштування доступу – для кожного опитування повинні бути налаштування, що визначають, чи є воно публічним, чи доступним за посиланням.

3. Проходження опитувань

- Інтерфейс для проходження опитувань – користувачі повинні мати зручний інтерфейс для заповнення опитувань.

- Перевірка результатів– після завершення проходження опитування користувач має бачити повідомлення про те, чи відповів він правильно на запитання (для тестів з правильною відповіддю) [16].

- Доступ до статистики по завершенню – користувач може побачити результат тесту, а також статистику про кількість правильних відповідей.

4. Персональний профіль

- Перегляд статистики результатів – у профілі користувач має можливість переглядати свою статистику, зокрема:

a) Статистика відповідей на власні опитування.

b) Статистика відповідей на опитування, які були пройдені (якщо такі дані відкриті автором опитування).

- Історія пройдених тестів – користувач має доступ до історії своїх тестів і результатів.

- Редагування профілю – користувач може змінювати свої особисті дані, включаючи пароль, електронну пошту, ім'я.

5. Адміністративна панель

- Управління опитуваннями – адміністратори мають можливість переглядати всі створені опитування, редагувати або видаляти їх.

- Управління користувачами – адміністратори можуть переглядати список користувачів і редагувати їхні профілі [17].

6. Автоматичний підрахунок результатів

- Підрахунок результатів тестів – система автоматично підраховує правильність відповідей для кожного тесту та виводить результати.

- Аналіз даних – на основі відповідей система може генерувати статистику для користувача (наприклад, середній бал, кількість правильних відповідей, середній час на тест).

7. Теми опитувань

- Групування за темами – користувач має можливість створювати опитування, що належать до певних тем, для зручності організації.

- Перегляд за темами – користувачі можуть фільтрувати опитування за темами для зручності пошуку [18].

Нефункціональні вимоги:

1. Безпека:

- Використання сучасних методів шифрування для збереження паролів користувачів.

- Застосування технології HTTPS для безпечної передачі даних.

2. Масштабованість:

- Система повинна підтримувати великий обсяг одночасних запитів від користувачів (підвищена навантажуваність).

3. Інтерфейс:

- Інтерфейс повинен бути адаптивним, зручним для використання на мобільних пристроях та комп'ютерах.

- Інтерфейс повинен бути зрозумілим і простим для користувача.

4. Продуктивність:

- Система повинна швидко обробляти відповіді та надавати результати [19].

5. Зручність:

- Легкість у використанні як для адміністратора, так і для кінцевого користувача.

Система онлайн-опитувань з автоматичним підрахунком результатів повинна забезпечити користувачів усіма необхідними інструментами для створення та проходження опитувань, а також аналізу результатів. Гнучкість налаштувань доступу, підтримка різних типів опитувань, інтеграція статистики, а також автоматизація процесу підрахунку результатів дозволить значно зекономити час і підвищити ефективність використання платформи.

# РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ ОНЛАЙН-ОПИТУВАНЬ

## 2.1 Вибір технологій для розробки системи

Для створення системи онлайн-опитувань з автоматичним підрахунком результатів необхідно обрати технологію, яка забезпечить високу продуктивність, зручність розробки, легкість інтеграції з бекендом та можливість подальшого розширення функціональності.

1. Next.js – це фреймворк, побудований на базі React, який розширює його можливості завдяки підтримці серверного рендерингу (SSR), статичної генерації сторінок (SSG) та клієнтського рендерингу (CSR). Він створений для спрощення розробки веб-додатків із високою продуктивністю та оптимізацією для пошукових систем [20].

Next.js оптимізує процес розробки, дозволяючи зосередитися на створенні функціональності, а не на складних налаштуваннях. Завдяки SSR сторінки генеруються на сервері, що забезпечує швидке перше відображення та відмінну індексацію пошуковими системами. SSG, у свою чергу, ідеально підходить для створення статичних опитувань, які не змінюються часто, забезпечуючи миттєве завантаження [21].

Вбудована маршрутизація спрощує створення нових сторінок. Наприклад, для системи опитувань можна легко створити маршрути типу /survey/[id], використовуючи динамічну маршрутизацію, що зменшує обсяг коду та прискорює розробку.

Оптимізація зображень і шрифтів у Next.js зменшує розмір сторінок і покращує користувацький досвід. Наприклад, якщо опитування містить графіки результатів, Next.js автоматично оптимізує їх для швидкого завантаження навіть на мобільних пристроях [22].

Серед переваг можна виділити серверний рендеринг (SSR) та статичну генерація (SSG). SSR дозволяє генерувати HTML на сервері під час кожного запиту, що покращує індексацію сторінок пошуковими системами (SEO) і забезпечує швидке перше відображення контенту. SSG генерує сторінки під час збірки проєкту, що ідеально для статичного контенту, наприклад, фіксованих опитувань, і значно прискорює їх завантаження для користувачів.

Вбудована маршрутизація: Next.js використовує файлову систему для автоматичного створення маршрутів. Наприклад, додавання файлу pages/survey.js автоматично створює маршрут /survey, що спрощує структуру проєкту та зменшує обсяг коду [23].

Оптимізація зображень і шрифтів: Next.js автоматично стискає зображення, підбирає оптимальний формат (наприклад, WebP) і завантажує їх ліниво (lazy loading), що зменшує час завантаження сторінок. Це особливо корисно для опитувань із візуальними елементами, такими як графіки чи ілюстрації.

Легка інтеграція з API: Next.js підтримує створення API-ендпоінтів у межах проєкту (наприклад, у папці pages/api), що спрощує обробку даних опитувань і їх автоматичний підрахунок [24].

Обмежена гнучкість у порівнянні з чистим React через фокус на SSR і SSG є великим недоліком. Деякі бібліотеки, які не сумісні з серверним рендерингом, можуть потребувати додаткових обхідних рішень.

Потреба в адаптації коду для специфічних сценаріїв, якщо проєкт виходить за рамки стандартних можливостей фреймворку [25].

2. React – це бібліотека для створення користувацьких інтерфейсів, яка фокусується на компонентному підході та гнучкому управлінні станом додатку. Вона є основою для Next.js, але часто використовується як самостійний інструмент [26].

React надає повну свободу у виборі архітектури та інструментів. Популярні бібліотеки, такі як React Router для маршрутизації чи Redux для управління

станом, дозволяють створювати додатки будь-якої складності. Однак для реалізації SSR чи SSG потрібно підключати додаткові фреймворки, що ускладнює процес порівняно з Next.js.

React також відкриває можливості для створення мобільних додатків через React Native. Це може бути перевагою, якщо система опитувань у майбутньому потребуватиме мобільної версії.

Перевагами React є наступні:

– Гнучкість. React дозволяє розробникам самостійно обирати інструменти для маршрутизації (наприклад, React Router), управління станом (Redux, Context API, MobX) та інших задач, що робить його універсальним для будь-яких проєктів.

– Велика екосистема. Завдяки активній спільноті React має величезну кількість бібліотек і готових рішень, що полегшує вирішення складних задач.

– Крос-платформність. За допомогою React Native можна створювати мобільні додатки, що відкриває можливості для розширення системи онлайн-опитувань на мобільні платформи в майбутньому.

Недоліки:

– Відсутність вбудованих SSR і SSG: React за замовчуванням використовує клієнтський рендеринг (CSR), де HTML генерується на стороні клієнта. Це ускладнює SEO і може уповільнити перше завантаження сторінки, якщо не додавати додаткові інструменти, такі як Next.js чи Gatsby.

– Додаткові налаштування: Для базових функцій, таких як маршрутизація чи оптимізація продуктивності, потрібно підключати сторонні бібліотеки, що збільшує час на розробку [27].

3. Angular – це повноцінний фреймворк від Google, який включає всі необхідні інструменти для створення складних веб-додатків, такі як маршрутизація, управління формами, HTTP-клієнт і сервіси [28].

Angular підходить для великих проєктів із чіткою структурою. Його вбудовані сервіси дозволяють легко організувати логіку обробки даних, наприклад, автоматичний підрахунок результатів опитувань. Angular Universal додає підтримку SSR, але його налаштування вимагає додаткових зусиль і знань.

Сильна типізація через TypeScript є значною перевагою для команд, які працюють над великими проєктами, але для невеликої системи опитувань це може бути надмірною складністю [29].

Вбудована функціональність – Angular має все необхідне «з коробки» маршрутизацію, обробку форм, ін’єкцію залежностей через сервіси, що зменшує потребу в сторонніх бібліотеках.

Сильна типізація – використання TypeScript забезпечує статичну перевірку типів, що зменшує кількість помилок і полегшує підтримку коду в великих проєктах.

Підтримка SSR через Angular Universal: Хоча SSR не є стандартною функцією, Angular Universal дозволяє реалізувати серверний рендеринг для покращення SEO і продуктивності [30].

Серед недоліків можна виділити наступні:

– Висока складність – Angular має круту криву навчання через велику кількість концепцій (зони, модулі, сервіси), що може бути перешкодою для невеликих проєктів або команд із новачками.

– Великий розмір бандла: Навіть після оптимізації Angular створює більші бандли порівняно з React чи Next.js, що може уповільнити завантаження сторінок на слабких пристроях чи повільних з’єднаннях.

– Менша гнучкість: Жорстка структура фреймворку ускладнює швидкі зміни чи адаптацію до нестандартних вимог [31]. Порівняння можливостей систем показано в табл. 2.1.

Таблиця 2.1 – Таблиця порівняння можливостей систем

| № | Критерій | Next.js [32] | React [33] | Angular [34] |
|---|---|---|---|---|
| 1 | Тип | Фреймворк на базі React | Бібліотека для UI | Повноцінний фреймворк |
| 2 | Рендеринг | SSR, SSG, CSR | CSR (SSR з дод. інструментами) | CSR (SSR з Angular Universal) |
| 3 | Маршрутизація | Вбудована (файлова система) | Потребує бібліотек (React Router) | Вбудована |
| 4 | Управління станом | Гнучке (React-інструменти) | Гнучке (Redux, Context тощо) | Вбудоване через сервіси |
| 5 | SEO | Відмінне (SSR, SSG) | Слабке без SSR | Слабке без SSR |
| 6 | Час завантаження | Швидкий (оптимізація зображень) | Залежить від налаштувань | Може бути повільним (великий бандл) |
| 7 | Екосистема | Велика (успадкована від React) | Дуже велика | Велика, але менша за React |
| 8 | Складність | Середня | Низька до середньої | Висока |
| 9 | Спільнота | Активна, швидко розвивається | Дуже активна, багато ресурсів | Активна, підтримка від Google |
| 10 | Документація | Відмінна, з прикладами | Відмінна, багато навч. матеріалів | Відмінна, але складна для новачків |
| 11 | Продуктивність розробки | Висока (вбудовані функції) | Середня (потрібні налаштування) | Середня до низької (складність) |

Для реалізації системи онлайн-опитувань з автоматичним підрахунком результатів фреймворк Next.js є технічно обґрунтованим і оптимальним вибором. Його переваги охоплюють ключові аспекти продуктивності, гнучкості та зручності розробки [32]:

1. Продуктивність і пошукова оптимізація SEO досягається завдяки підтримці SSR та SSG, сторінки завантажуються швидко та залишаються доступними для пошукових систем. Це особливо важливо для широкого охоплення аудиторії та залучення респондентів до участі в опитуваннях.

2. Простота та швидкість розробки. Next.js має вбудовану систему маршрутизації, підтримку оптимізації зображень, а також можливість створення API-ендпоінтів без необхідності налаштування окремого бекенду. Це дозволяє

зосередитися на функціональності та логіці опитувань, зменшуючи часові витрати на інфраструктурні завдання [33].

3. Гнучкість і масштабованість. Базування на React забезпечує доступ до потужної екосистеми бібліотек і компонентів. Це дає змогу розширювати функціонал, додавати інтерактивні елементи, реалізовувати логіку перевірки, фільтрації та аналітики без суттєвих обмежень.

4. Ефективна інтеграція з бекендом. Завдяки можливості створення API-роутів безпосередньо в Next.js, обробка вхідних даних, збереження відповідей та обчислення статистичних результатів відбувається швидко й без потреби у складній інтеграції з зовнішніми сервісами [34].

React є потужним інструментом, але потребує додаткових налаштувань для досягнення рівня продуктивності Next.js, що робить його менш ефективним для цього проєкту. Angular, у свою чергу, занадто складний і громіздкий для системи опитувань, де пріоритетом є швидкість і простота. Таким чином, Next.js поєднує в собі найкращий баланс між продуктивністю, зручністю та розширюваністю, що робить його ідеальним вибором для даного проєкту.

## 2.2 Створення бази даних

Для забезпечення функціональності системи онлайн-опитувань необхідно організувати зберігання таких категорій даних:

– Опитування: назва, опис, дата створення, статус (активне / завершене).

– Питання: текст питання, тип (одновибір, множинний вибір, відкрита відповідь), прив’язка до відповідного опитування.

– Відповіді користувачів: обрані варіанти або текстові відповіді, що пов’язуються з конкретним питанням та користувачем.

– Користувачі: ідентифікаційна інформація (ім’я, email, пароль), необхідна для автентифікації.

– Результати: агреговані дані для подальшого аналізу та візуалізації.

Дані мають бути структурованими, взаємопов’язаними та оптимізованими для частих операцій читання/запису, з урахуванням перспективи розширення системи (додання нових типів питань, аналітики тощо).

Для такого типу системи доцільно використати реляційну базу даних, оскільки:

– Всі основні сутності (опитування, питання, відповіді) мають чітко виражені зв’язки один до одного (1:M, M:1).

– Реляційна модель дозволяє легко реалізувати цілісність даних за допомогою зовнішніх ключів.

– Підтримуються транзакції відповідно до ACID-властивостей, що критично важливо для збереження узгодженості відповідей у реальному часі.

PostgreSQL – оптимальний вибір з таких причин:

– Підтримка складних запитів, індексів, транзакцій, CTE (Common Table Expressions), а також типу JSON, що дозволяє гнучко зберігати додаткові дані.

– Надійна інтеграція з бекендом, написаним на Next.js або Node.js, з використанням ORM-бібліотек (наприклад, Prisma, TypeORM, Sequelize).

– Висока масштабованість і продуктивність, що дозволяє обслуговувати велику кількість одночасних користувачів.

Альтернативи:

– MySQL – підходить для простіших рішень, але має менш гнучке розширення типів.

– SQLite – зручна для прототипів або десктопних застосунків, однак не рекомендується для продуктивних хмарних систем.

Структура бази даних (таблиці) мають наступний вигляд:

– Таблиця User (табл 2.2),

Таблиця 2.2 – Поля таблиці User

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор користувача |
| 2 | email | VARCHAR(255) | UNIQUE, NOT NULL | Електронна пошта користувача |
| 3 | password | VARCHAR(255) | NOT NULL | Хеш пароля |
| 4 | name | VARCHAR(255) | NOT NULL | Ім’я користувача |
| 5 | role | VARCHAR(50) | DEFAULT 'user' | Роль: 'user' або 'admin' |
| 6 | createdAt | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата створення запису |
| 7 | updatedAt | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата останнього оновлення |

Таблиця Topic (табл.2.3) – зберігає теми опитувань. Таблиця Topic використовується для категоризації та класифікації опитувань за змістовими напрямами. Це дозволяє структурувати великий масив опитувань, спростити навігацію користувачів у системі, а також реалізувати фільтрацію, пошук і тематичний аналіз даних. Така категоризація є особливо важливою у великих інформаційних системах або освітніх, соціологічних, маркетингових проєктах.

Таблиця 2.3 – Поля таблиці Topic (Теми опитувань)

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор теми |
| 2 | name | VARCHAR(255) | NOT NULL | Назва теми (наприклад, "Освіта") |
| 3 | description | TEXT | | Опис теми |
| 4 | createdAt | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата створення запису |

Таблиця Surveys (табл.2.4) – зберігає базову метаінформацію про опитування, включаючи назву, опис, дату створення, статус активності та зв'язок з тематикою. Вона є центральним елементом для побудови логіки платформи, оскільки всі інші сутності (питання, відповіді, результати) прив’язані саме до конкретного опитування. Її функціональне призначення включає управління життєвим циклом опитування (створення, публікація, завершення), зв’язування з темами (topic_id) для категоризації, ідентифікація авторства (через зовнішній ключ user_id), забезпечення основи для побудови списків, фільтрів і звітів

Таблиця 2.4 – Поля таблиці Surveys (Опитування)

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор опитування |
| 2 | title | VARCHAR(255) | NOT NULL | Назва опитування |
| 3 | description | TEXT | | Опис опитування |
| 4 | topicId | INTEGER | FOREIGN KEY (Topic.id) | Ідентифікатор теми |
| 5 | createdAt | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата створення запису |

Таблиця Questions (табл. 2.5) зберігає всі окремі питання, що входять до складу конкретного опитування. Вона забезпечує гнучкість у розробці анкет, дозволяє варіативність типів питань (одновибір, множинний вибір, відкрита відповідь тощо) і створює основу для структурованого збору відповідей користувачів.

Функціональне навантаження таблиці Questions полягає у тому що, вона формалізує структуру опитування, дозволяє динамічне формування інтерфейсу відображення питань, пов’язує питання з відповідним опитуванням, підтримує типізацію та порядок відображення питань.

Таблиця 2.5 – Поля таблиці Questions

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор питання |
| 2 | surveyId | INTEGER | FOREIGN KEY (Survey.id) | Ідентифікатор опитування |
| 3 | text | TEXT | NOT NULL | Текст питання |

Таблиця Option (табл.2.6) використовується для зберігання попередньо визначених варіантів відповіді, які відображаються користувачеві для вибору при заповненні питань типу одновибір (radio buttons) або множинний вибір (checkboxes). Вона забезпечує структурованість і дозволяє аналітично обробляти результати на основі заданих опцій. Функціональне навантаження даної таблиці полягає в зберіганні повного переліку можливих відповідей для кожного питання, зв’язок із конкретним питанням із таблиці Questions, задання порядку відображення опцій у формі, потенційне маркування правильних відповідей (для тестів).

Таблиця 2.6 – Поля таблиці Option (Варіанти відповідей)

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор варіанту |
| 2 | questionId | INTEGER | FOREIGN KEY (Question.id) | Ідентифікатор питання |
| 3 | text | VARCHAR(255) | NOT NULL | Текст варіанту відповіді |

Таблиця UserSurvey (табл. 2.7) виконує роль проміжної таблиці (junction table) у зв’язку між таблицями Users і Surveys, фіксуючи, хто, коли та яке опитування проходив. Це важливо як для ведення журналу участі, так і для забезпечення унікальності проходження одного опитування одним користувачем (у разі обмежень). Функціональне навантаження передбачає відображення факту участі користувача в опитуванні, зберігання часу початку або завершення

опитування, підтримка механізмів захисту від повторного проходження. Таким чином це основна таблиця для зв’язку з UserAnswers.

Таблиця 2.7 – Поля таблиці UserSurvey (Опитування пройдені користувачем)

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор запису |
| 2 | userId | INTEGER | FOREIGN KEY (User.id) | Ідентифікатор користувача |
| 3 | surveyId | INTEGER | FOREIGN KEY (Survey.id) | Ідентифікатор опитування |
| 4 | completed At | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата проходження |

Таблиця UserAnswer (табл.2.8) фіксує детальні дані про відповіді, які користувач надає під час проходження конкретного опитування. Це центральна таблиця, що дозволяє відстежувати кожну відповідь користувача на конкретне питання, підтримувати різні типи питань: вибір однієї, кількох або текстову відповідь, здійснювати агрегацію для подальшого аналізу результатів опитування. Функціональне навантаження даної таблиці – підтримка множинних типів відповідей (через універсальне поле), прив’язка до UserSurvey для групування відповідей по сесіях, основне джерело інформації для аналітики та візуалізації.

Зв’язки між даними таблицями мають наступн категорії:

1. Зв’язок між таблицями User та UserSurvey має тип «один до багатьох» (1:N). Один користувач (User.id) може проходити багато опитувань, кожне з яких фіксується у таблиці UserSurvey через поле userId. Таким чином, поле UserSurvey.userId є зовнішнім ключем, що посилається на User.id і встановлює зв’язок між записами цих таблиць.

Таблиця 2.8 – Поля таблиці UserAnswer (Відповіді користувачів на питання)

| № | Поле | Тип даних | Обмеження | Опис |
|---|---|---|---|---|
| 1 | id | INTEGER | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор відповіді |
| 2 | userId | INTEGER | FOREIGN KEY (User.id) | Ідентифікатор користувача |
| 3 | questionId | INTEGER | FOREIGN KEY (Question.id) | Ідентифікатор питання |
| 4 | optionId | INTEGER | FOREIGN KEY (Option.id) | Ідентифікатор вибраного варіанту |
| 5 | createdAt | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Дата збереження відповіді |

2. Зв’язок між таблицями User та UserAnswer є типу «один до багатьох» (1:N). Один користувач (User.id) може надати багато відповідей, що зберігаються у таблиці UserAnswer через поле userId. Поле UserAnswer.userId виступає як зовнішній ключ і посилається на User.id, забезпечуючи коректну асоціацію відповідей з відповідним користувачем.

3. Зв’язок між таблицями Topic та Survey має тип «один до багатьох» (1:N). Одна тема (Topic.id) може бути пов’язана з багатьма опитуваннями, які зберігаються у таблиці Survey через поле topicId. Поле Survey.topicId є зовнішнім ключем, що посилається на Topic.id, встановлюючи зв’язок між темою та відповідними опитуваннями.

4. Зв’язок між таблицями Survey та Question є типу «один до багатьох» (1:N). Одне опитування (Survey.id) містить багато питань, що зберігаються у таблиці Question через поле surveyId. Поле Question.surveyId виступає як зовнішній ключ і посилається на Survey.id, забезпечуючи коректну асоціацію питань з відповідним опитуванням.

5. Зв’язок між таблицями Survey та UserSurvey має тип «один до багатьох» (1:N). Одне опитування (Survey.id) може бути пройдене багатьма користувачами, про що свідчать записи в таблиці UserSurvey через поле surveyId. Поле

UserSurvey.surveyId є зовнішнім ключем, що посилається на Survey.id, встановлюючи зв'язок між опитуванням і користувачами, які його проходили.

6. Зв'язок між таблицями Question та Option є типу «один до багатьох» (1:N). Одне питання (Question.id) має багато варіантів відповідей, які зберігаються у таблиці Option через поле questionId. Поле Option.questionId виступає як зовнішній ключ і посилається на Question.id, забезпечуючи коректну асоціацію варіантів відповідей з відповідним питанням.

7. Зв'язок між таблицями Question та UserAnswer має тип «один до багатьох» (1:N). Одне питання (Question.id) може отримати багато відповідей від користувачів, що зберігаються у таблиці UserAnswer через поле questionId. Поле UserAnswer.questionId є зовнішнім ключем, який посилається на Question.id, забезпечуючи зв'язок між питаннями та відповідями користувачів.

8. Зв'язок між таблицями Option та UserAnswer є типу «один до багатьох» (1:N). Один варіант відповіді (Option.id) може бути вибраний у багатьох відповідях користувачів, що зберігаються у таблиці UserAnswer через поле optionId. Поле UserAnswer.optionId виступає як зовнішній ключ і посилається на Option.id, забезпечуючи коректну асоціацію вибраних варіантів відповідей з відповідними відповідями користувачів. Схему бази даних зображено на рис. 2.1.

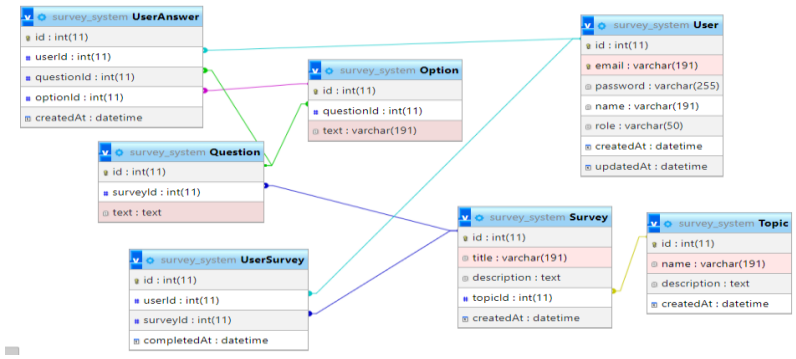


Рис.2.1 Схема бази даних

Для інтеграції бази даних із Next.js рекомендується використовувати Prisma – сучасну ORM, яка значно спрощує роботу з базою даних. Prisma підтримує TypeScript та автоматично генерує типи, що підвищує надійність і зручність коду. Завдяки об’єктно-орієнтованому підходу Prisma дозволяє легко і зрозуміло виконувати запити до бази даних. Також вона має вбудовані інструменти для управління міграціями, що допомагає зберігати структуру бази в актуальному стані.

Процес налаштування Prisma з Next.js включає кілька основних кроків. Спершу у проект встановлюють Prisma CLI та клієнтську бібліотеку. Після цього ініціалізують Prisma, що створює конфігураційні файли, зокрема файл схеми, де описується структура бази даних, і файл для налаштування підключення до бази. Наступним кроком є налаштування рядка підключення до бази даних у файлі конфігурації.

Далі описують моделі даних у файлі схеми Prisma відповідно до структури таблиць та їх зв’язків у базі. Після цього виконують міграції – процес створення таблиць і зв’язків у базі даних на основі опису моделей. В результаті міграції Prisma генерує клієнт, який використовується для роботи з базою даних у коді Next.js.

Таким чином, інтеграція Prisma з Next.js забезпечує типізований і зручний доступ до бази даних, полегшує підтримку структури даних та пришвидшує розробку функціоналу. Якщо потрібно, можна описати конкретні моделі для твоєї бази даних, враховуючи існуючі таблиці і зв’язки. Процес встановлення Prisma показано в додатку Б.1.

Створення бази даних для системи онлайн-опитувань на Next.js із використанням PostgreSQL і Prisma забезпечує надійне зберігання даних, ефективну обробку та зручність розробки. Нормалізована схема з чіткими зв’язками гарантує цілісність, а інтеграція з Next.js через Prisma спрощує доступ

до даних і дозволяє зосередитися на логіці системи, такій як автоматичний підрахунок результатів.

### 2.3 Реалізація бекенду

Бекенд системи онлайн-опитувань відповідає за обробку даних, управління користувачами, збереження та отримання опитувань, питань і відповідей, а також за автоматичний підрахунок результатів. У рамках роботи бекенд реалізується з використанням API Routes, що є частиною Next.js. Це дозволяє створювати серверну логіку безпосередньо в проєкті, без необхідності розгортати окремий сервер.

У Next.js API Routes розміщуються в папці pages/api. Кожен файл у цій папці автоматично стає окремим API-ендпоінтом. Наприклад, файл pages/api/surveys.js відповідає ендпоінту /api/surveys. Такий підхід спрощує організацію серверної логіки та її інтеграцію з фронтендом.

Для роботи з базою даних використовується ORM Prisma, яка забезпечує зручну взаємодію з реляційними базами даних, такими як PostgreSQL. У проєкті створюється файл lib/prisma.js для ініціалізації PrismaClient:

```
import { PrismaClient } from '@prisma/client';
const prisma = new PrismaClient();
export default prisma;
```

У системі, що базується на Next.js з використанням Prisma, доступ до бази даних реалізується через спеціальний модуль – Prisma Client. Цей модуль імпортується в API-ендпоінтах, які відповідають за обробку запитів до сервера. Зокрема, за допомогою Prisma Client у цих ендпоінтах можна виконувати операції створення, читання, оновлення та видалення даних.

Один із таких ендпоінтів реалізує функціонал створення нового опитування. Коли користувач надсилає запит на створення опитування, система обробляє передані дані – наприклад, заголовок опитування, його опис, а також

поточний статус (наприклад, активне чи неактивне). Після перевірки валідності даних ендпоінт формує запит до бази даних за допомогою Prisma Client, створюючи відповідний запис у таблиці Survey.

Такий підхід дозволяє централізовано керувати створенням опитувань, забезпечує перевірку даних перед їх внесенням до бази, а також гарантує, що всі нові опитування зберігаються у відповідності до структури, описаної в Prisma-схемі, детальний опис якої представлено в додатку Б.2. Використання ендпоінт представлено в додатку Б.2.

Окремий API-ендпоінт у системі опитувань відповідає за агрегацію відповідей користувачів та формування результатів опитування. Цей ендпоінт є важливою складовою аналітичного функціоналу, оскільки дозволяє отримати зведену інформацію щодо вибору користувачів по кожному питанню.

Після отримання ідентифікатора опитування, ендпоінт звертається до бази даних через Prisma Client і виконує декілька послідовних операцій:

– Отримання списку питань, що належать до заданого опитування. Це дає змогу ідентифікувати, які саме питання потребують обробки.

– Аналіз типу кожного питання – якщо питання має закритий тип (тобто передбачає вибір одного або кількох варіантів відповіді), система:

– Знаходить усі пов’язані з ним варіанти відповідей (із таблиці Option).

– Для кожного варіанту обчислює кількість голосів, тобто підраховує, скільки разів цей варіант було вибрано користувачами у таблиці UserAnswer.

– Формує агрегований результат у вигляді: варіант – кількість голосів.

– Якщо питання є відкритим (тобто користувач має надати текстову відповідь), система витягує усі текстові відповіді з таблиці UserAnswer та формує список відповідей, що дозволяє переглядати індивідуальні думки або коментарі користувачів.

– Формування структури результатів: отримані дані структуруються у форматі, що містить ідентифікатор та текст кожного питання, тип питання (закрите/відкрите), агреговану статистику (для закритих) або список текстів (для відкритих).

– Повернення результатів клієнту: після обробки всіх питань, ендпоінт надсилає зібрані результати у відповідь, які можуть бути візуалізовані на інтерфейсі у вигляді графіків, діаграм або списків.

Такий механізм дозволяє проводити ефективний аналіз результатів опитування в реальному часі та забезпечує підтримку як кількісних, так і якісних методів збору даних.

Реалізація бекенду системи онлайн-опитувань на Next.js за допомогою API Routes забезпечує створення ефективного серверного шару, інтегрованого з базою даних через Prisma. Бекенд підтримує управління опитуваннями, питаннями, відповідями та автоматичний підрахунок результатів, що відповідає вимогам дипломної роботи. Додаткові заходи, такі як автентифікація через JWT і тестування, гарантують безпеку та стабільність системи.

## 2.4 Фронтенд: створення інтерфейсу користувача

Інтерфейс користувача системи онлайн-опитувань розроблено з акцентом на зручність, інтуїтивність та естетичну привабливість, що забезпечує комфортний і позитивний досвід для кожного користувача. В основі клієнтської частини лежить сучасний фреймворк Next.js, який дозволяє створювати високопродуктивні, SEO-оптимізовані та динамічні веб-додатки.

Дизайн інтерфейсу витримано у спокійній блакитній гамі, яка сприяє приємному сприйняттю: основний колір активних елементів – насичений блакитний (#007BFF), а фоновий колір сторінок – м’який світло-блакитний (#E3F2FD). Такий кольоровий баланс створює візуально легке середовище, у

якому користувач може зосередитись на змісті опитувань, не відволікаючись на зайві елементи.

Інтерфейс системи складається з низки ключових сторінок і компонентів, кожен з яких виконує певну функцію у взаємодії користувача з платформою. Навігаційна панель є універсальним компонентом, присутнім на всіх сторінках системи. Вона виконує роль орієнтиру та швидкого доступу до основних розділів: посилань на головну сторінку, профіль користувача, вхід або реєстрацію. Залежно від статусу автентифікації користувача навігаційна панель динамічно змінює свій вміст. Візуально панель оформлена у блакитному кольорі (#007BFF) із білим текстом, що створює чіткий контраст і хорошу читабельність. Логотип "Опитування" веде на головну сторінку системи.

Головна сторінка є центральним елементом взаємодії з системою, надає огляд усіх наявних опитувань і дозволяє користувачу зручно здійснювати пошук або фільтрацію за темами чи параметрами. Опитування завантажуються з сервера й відображаються у вигляді інтерактивного списку, кожен елемент якого веде на окрему сторінку з питаннями. Вся сторінка витримана в єдиному стилі: світло-блакитний фон (#E3F2FD), блакитні заголовки й активні посилання (#007BFF).

Сторінка окремого опитування надає можливість користувачеві відповідати на питання. Кожне питання виводиться у відповідному форматі: одновибір (radio), множинний вибір (checkbox) або відкрита текстова відповідь (textarea). Після заповнення форми користувач натискає кнопку "Надіслати відповіді", яка оформлена у вигляді помітного елементу з фоном #007BFF і білим текстом. Уся сторінка підтримує загальний стиль – світло-блакитний фон і блакитні активні елементи.

Сторінка результатів реалізує аналітичну функцію: вона агрегує всі зібрані відповіді та відображає результати опитування. Для закритих питань (із вибором варіантів) результати подаються у вигляді графіків або гістограм, що дозволяє швидко оцінити переваги думок респондентів. Для відкритих питань

повертається список текстових відповідей, зібраних від учасників. Інтерфейс цієї сторінки залишається легким і лаконічним, з мінімалістичним оформленням, яке зосереджене на змісті даних.

Сторінки авторизації включають форми для реєстрації нового користувача або входу до системи. Вони мають сучасне й зручне оформлення: світло-блакитний фон, блакитні рамки полів для вводу, закруглені кути для створення м’якості форм, а також кнопку підтвердження дії з фоном #007BFF і білим текстом. Такий підхід дозволяє зробити процес авторизації простим, приємним і зрозумілим для всіх категорій користувачів.

Загалом інтерфейс онлайн-опитувальної системи поєднує сучасні технології, адаптивний дизайн і виважену кольорову палітру, що робить користування не лише функціональним, а й естетично привабливим.

Наведені приклади демонструють реалізацію інтерфейсу системи онлайн-опитувань на базі Next.js. Блакитна колірна гама забезпечує візуальну гармонію, а модульна структура компонентів полегшує підтримку та розширення системи.

### 2.5 Алгоритм автоматичного підрахунку результатів

У межах цієї роботи було розроблено ефективний алгоритм автоматичної обробки та підрахунку результатів опитувань. Він виступає ключовим компонентом системи, забезпечуючи швидкий і точний аналіз зібраних відповідей користувачів, а також їх подання у зручній та наочній формі. Алгоритм адаптований до особливостей різних типів питань, зокрема тих, що передбачають один правильний варіант, множинний вибір або відкриту текстову відповідь. Завдяки цій гнучкості система забезпечує коректне опрацювання даних незалежно від типу опитування.

Алгоритм побудовано у вигляді чіткої послідовності з п’яти логічних етапів.

Перший етап – збір даних – полягає у вибірці всіх відповідей користувачів, що стосуються конкретного опитування. На цьому етапі система формує запит до бази даних за допомогою інструментів ORM Prisma. Запит охоплює таблицю UserAnswer і витягує пов’язані записи з таблиць Question та Option, щоб забезпечити повну інформацію для подальшого аналізу.

Завдяки реляційним зв’язкам між таблицями система отримує повний набір даних: тип кожного питання, відповідні варіанти відповідей (для закритих типів), а також текстові відповіді (для відкритих питань). Цей етап є підґрунтям для подальших операцій агрегації та візуалізації, забезпечуючи повноту і достовірність вхідних даних. За допомогою Prisma виконується запит до таблиці answers, який повертає відповіді разом із інформацією про питання і виглядає наступним чином:

```
const answers = await prisma.answer.findMany({
  where: { question: { surveyId: surveyId } },
  include: { question: true },
});
```

На другому етапі алгоритму здійснюється класифікація питань, що є критично важливою для правильного застосування методів обробки відповідей. Для кожного питання, отриманого з бази даних, визначається його тип – single_choice (одновибір), multiple_choice (множинний вибір) або text (відкрита текстова відповідь). Ця інформація зберігається у відповідному полі таблиці Question і безпосередньо впливає на логіку агрегації результатів.

Залежно від типу, система використовує спеціалізовані механізми обробки: для одновибірних питань підраховується кількість голосів за кожен варіант; для питань із множинним вибором – кількість обрань кожного варіанта по всіх відповідях; для відкритих текстових відповідей – збирається список усіх текстових фраз без агрегації. Такий підхід дозволяє забезпечити гнучкість

алгоритму, його адаптивність до різних форматів запитань та точність у формуванні підсумкової інформації.

Третім етапом алгоритму є обробка відповідей, яка виконується з урахуванням типу кожного питання. На цьому кроці застосовуються відповідні методи агрегації або збору даних, що забезпечують коректне представлення результатів опитування в залежності від логіки кожного типу запитання.

Для питань типу «одновибір» (single_choice) система підраховує кількість голосів за кожен окремий варіант відповіді. Наприклад, якщо варіантів три – A, B і C – то алгоритм визначає, скільки разів кожен із них був обраний користувачами. Таким чином, формується чітка статистика за кожною опцією.

У випадку множинного вибору (multiple_choice) обробка ускладнюється тим, що користувач має можливість обрати кілька варіантів одночасно. Відповіді зберігаються у вигляді масиву значень (наприклад, [A, B]), і система підраховує кількість разів, коли кожен варіант був включений до відповідей. Це дозволяє отримати загальну популярність кожної опції незалежно від кількості користувачів.

Для відкритих текстових питань (text) алгоритм виконує збір усіх введених користувачами відповідей без попередньої агрегації. Усі текстові дані зберігаються в окремому списку, який може бути використаний для подальшого аналізу – зокрема, для тематичної класифікації, фільтрації або візуалізації. У разі потреби можна застосувати додаткову обробку: наприклад, згрупувати подібні відповіді або виявити найчастотніші формулювання.

Завдяки такій типозалежній логіці алгоритм забезпечує точність і гнучкість у формуванні статистичних результатів опитувань, адаптуючись до структури кожного конкретного запитання.

4. На четвертому етапі алгоритму відбувається агрегація результатів, що передбачає підсумовування зібраних відповідей у чітко структурованому форматі. Цей процес є необхідним для подальшого подання інформації у вигляді

таблиць, списків або графіків, а також для збереження результатів у форматах, придатних до аналітики.

Для питань із вибором варіантів (single_choice або multiple_choice) результати агрегуються у вигляді об’єкта (словника), де кожен ключ відповідає окремій опції, а значення – кількості голосів, які ця опція отримала. ля відкритих текстових питань (text) система формує масив усіх отриманих відповідей без попередньої фільтрації або обробки. Це дозволяє зберегти повноту інформації та забезпечує можливість подальшого аналізу, включаючи кластеризацію, фільтрацію за ключовими словами чи контент-аналіз. Такий підхід до агрегації результатів забезпечує узгоджене, уніфіковане представлення даних незалежно від типу питання, що значно спрощує побудову аналітичних інтерфейсів і систем візуалізації.

Приклад для одновибору виглядає наступним чином:

```
const singleChoiceResults = answers
  .filter(answer => answer.question.type === 'single_choice')
  .reduce((acc, answer) => {
    const option = answer.selectedOption;
    acc[option] = (acc[option] || 0) + 1;
    return acc;
  }, {});
```

Завершальним, п’ятим етапом роботи алгоритму є візуалізація результатів, що забезпечує наочне представлення зібраних даних у зручному та зрозумілому для користувача форматі. Візуальна інтерпретація відповідей відіграє ключову роль у процесі аналізу, дозволяючи швидко виявити закономірності, переваги учасників опитування та загальні тенденції.

Для питань із вибором відповідей (як одновибір, так і множинний вибір) результати відображаються у вигляді графіків. Найчастіше використовуються стовпчасті (bar) або кругові (pie) діаграми, які реалізуються за допомогою

сучасних JavaScript-бібліотек для візуалізації даних, таких як Chart.js. Кожна опція відповіді представлена як окремий сегмент або стовпчик, висота чи розмір якого пропорційна кількості голосів, що дозволяє інтуїтивно оцінити популярність кожного варіанта.

Для відкритих текстових відповідей результати подаються у вигляді списку текстів, наданих користувачами, або хмари слів, яка дозволяє швидко ідентифікувати найчастіше вживані фрази. Такий підхід особливо корисний для виявлення ключових тем чи проблем, які хвилюють респондентів.

Основний кодовий блок, що реалізує цей функціонал, виконує попередню групування відповідей за питаннями, підраховує кількість голосів для кожної опції у випадку вибору і збирає текстові відповіді у єдиний масив. Уся обробка відбувається автоматично, завдяки чому користувач системи миттєво отримує готові до перегляду результати без потреби в ручному підрахунку чи фільтрації.

Розроблений алгоритм обробки та візуалізації результатів став невід’ємною частиною онлайн-системи опитувань, створеної на базі Next.js. Його інтеграція забезпечила швидке, точне та адаптивне опрацювання відповідей, дозволивши ефективно аналізувати великі обсяги даних і представляти їх у форматі, зручному як для користувачів, так і для адміністраторів системи.

# РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

## 3.1 Тестування системи

Система розгорнута локально у репозиторії **survey-system** із використанням сучасних технологій: Next.js версії 15.3.0, бази даних PostgreSQL, системи автентифікації NextAuth, ORM Prisma та стилізації за допомогою Tailwind CSS.

Для запуску сервера необхідно виконати команду у кореневій папці проєкту:

```
-- Тема
INSERT INTO "Topic" (id, name, description) VALUES
  (1, 'Освіта', 'Опитування про освіту та навчання')
ON CONFLICT (id) DO NOTHING;
-- Опитування
INSERT INTO "Survey" (id, title, description, topicId, createdAt) VALUES
  (2, 'Тестове опитування 2', 'Оцінка тестування', 1, CURRENT_TIMESTAMP)
ON CONFLICT (id) DO NOTHING;
-- Питання
INSERT INTO "Question" (id, surveyId, text) VALUES
  (2, 2, 'Чи подобається вам тестування?')
ON CONFLICT (id) DO NOTHING;
-- Варіанти відповідей
INSERT INTO "Option" (id, questionId, text) VALUES
  (5, 2, 'Так'),
  (6, 2, 'Ні')
ON CONFLICT (id) DO NOTHING;
-- Користувач
```

INSERT INTO "User" (email, password, name, createdAt, updatedAt) VALUES

('test@example.com', '$2b$10$<вставте_хеш_password123>', 'Тестовий Користувач', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP)

ON CONFLICT (email) DO NOTHING;

Для створення хешу пароля рекомендується використовувати надійні алгоритми хешування, такі як **bcrypt**, які забезпечують безпеку зберігання паролів завдяки сольовим вставкам і стійкості до атак типу «перебір».

Наприклад, у середовищі Node.js можна створити хеш пароля за допомогою бібліотеки bcrypt наступним чином:

Кроки тестування виглядають наступним чином:

1. Реєстрація користувачамає на меті перевірку можливостіь успішного створення нового користувача в системі.

Кроки виконання: Необхідно відкрити у браузері сторінку реєстрації за адресою http://localhost:3000/auth/signup. У відповідні поля необхідно ввести наступні дані: електронну адресу – newuser@example.com, ім’я – Новий Користувач, пароль – newpassword123. Після заповнення форми необхідно натиснути кнопку «Зареєструватися».

Очікуваний результат – після успішної реєстрації користувач буде автоматично перенаправлений на сторінку входу (/auth/signin). У базі даних у таблиці "User" з’явиться новий запис із введеними даними. Для перевірки створення запису можна виконати SQL-запит:

SELECT email, name FROM "User" WHERE email = 'newuser@example.com';

В результаті запиту має бути виведено відповідний запис із електронною адресою та іменем нового користувача, зображений на рис.3.1.

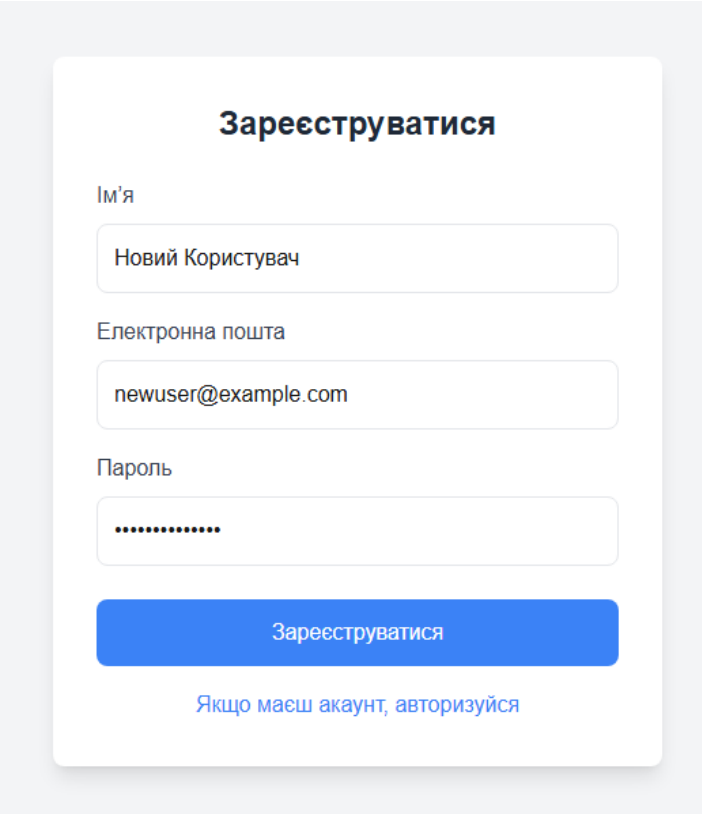


Рисунок 3.1 – Форма реєстрації

2. Авторизація користувача має на меті перевірірку входу авторизованого користувача.

Кроки виконання: Користувач переходить на сторінку авторизації за адресою http://localhost:3000/auth/signin. У форму входу вводяться такі дані:

– Email: test@example.com

– Пароль: password123

Після заповнення форми натискає кнопку «Увійти».

Очікуваний результат: Після натискання кнопки користувач має бути автоматично перенаправлений на головну сторінку системи за адресою http://localhost:3000.

Для додаткової перевірки сесії у консолі браузера (відкривається через F12 Console) виконується наступна команда:

```
fetch('/api/auth/session')
  .then(res => res.json())
```

```
  .then(data => console.log(data));
```

Результатом виконання має бути об’єкт, що містить інформацію про авторизованого користувача, зокрема:

```
{
  user: {
    email: "test@example.com",
    ...
  },
  ...
}
```

Фактичний результат (на момент попереднього тестування): Авторизація була успішною. Після відправлення форми отримано відповідь від сервера у форматі:

SignIn - Result: { error: null, status: 200 }

Це підтверджує, що облікові дані були валідними, і сесія користувача створена коректно. Головна сторінка після входу відображена на рис. 3.2.

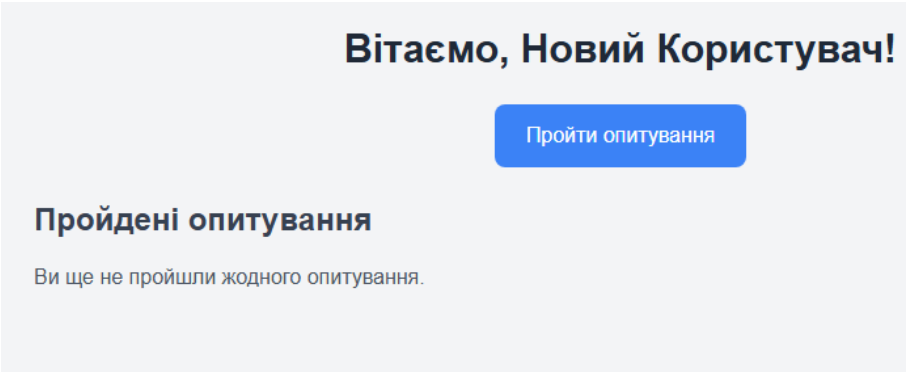


Рисунок 3.2 – Головна сторінка після входу

3. Перегляд доступних опитувань має на меті перевірити чи коректно відображається список наявних опитувань для авторизованого користувача на головній сторінці.

Кроки виконання:

Авторизація у системі, використовуючи облікові дані користувача test@example.com. Після входу необхідно перейти на головну сторінку веб-додатку за адресою http://localhost:3000. Ознайомлення зі списком доступних опитувань, що відображаються на сторінці.

Очікуваний результат – на головній сторінці має з’явитися перелік опитувань, доступних для проходження. Зокрема, в списку має бути присутнє опитування з назвою «Тестове опитування 2» та ідентифікатором id = 2.

Назва опитування повинна бути клікабельною. При натисканні на нього користувач має бути перенаправлений на сторінку з URL-адресою: http://localhost:3000/survey/2. Список тем по опитуванням відображено на рис. 3.3.

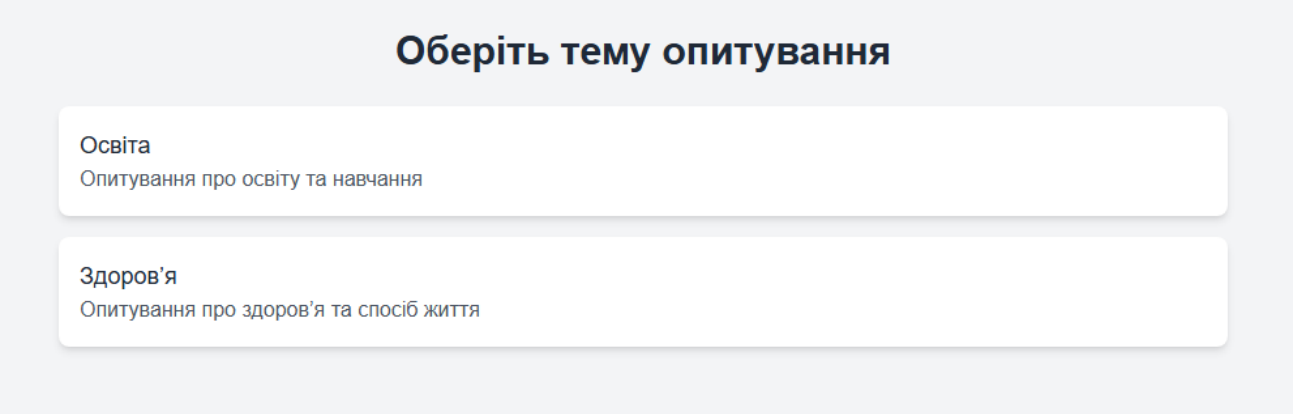


Рисунок 3.3 – Список тем по опитуванням

4. Проходження опитування має на меті перевірку функціональності проходження опитування користувачем, а також коректність збереження його відповідей у базі даних і передбачає наступні кроки:

Кроки тестування:

- Авторизація у системі під обліковим записом test@example.com.
- Перехід на сторінку опитування за посиланням: http://localhost:3000/survey/2

- Ознайомлення із запитанням та вибір одного із запропонованих варіантів відповіді (наприклад, «Так»).
- Натиснення кнопки «Зберегти відповіді», щоб надіслати відповіді на сервер.

Очікуваний результат – після успішного збереження відповідей система має перенаправити користувача на головну сторінку:

http://localhost:3000

У базі даних мають з’явитися нові записи:

у таблиці UserSurvey, що фіксує сам факт проходження користувачем опитування:

SELECT * FROM "UserSurvey" WHERE surveyId = 2;

у таблиці UserAnswer, що містить відповіді користувача на конкретні питання:

SELECT * FROM "UserAnswer" WHERE questionId = 2;

Після переходу на сторінку http://localhost:3000/survey/2 відображається повідомлення «Опитування не знайдено», оскільки змінна survey має значення null. Це свідчить про проблему із завантаженням даних опитування з бази, можливо, через відсутність опитування з ідентифікатором 2 або помилку в логіці завантаження. Сторінку опитування зображено на рис. 3.4.

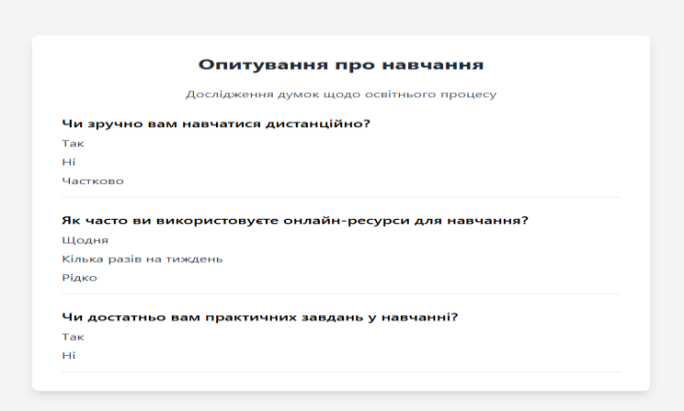


Рисунок 3.4 – Сторінка опитування

Тестування системи survey-system підтвердило стабільну роботу основних функціональних компонентів. Зокрема, перевірено коректність реалізації процесів реєстрації нового користувача, авторизації, перегляду доступних опитувань, а також проходження опитування з фіксацією відповідей у базі даних. Усі ключові дії виконувались відповідно до очікуваної логіки, що свідчить про правильність налаштування маршрутизації, взаємодії з базою даних через Prisma, а також інтеграції UI з серверною частиною. Система демонструє працездатність і готовність до подальшого розширення функціоналу.

### 3.2 Інструкція користувача

1. Реєстрація нового користувача є першим етапом взаємодії користувача з системою онлайн-опитувань. Вона дозволяє створити персональний обліковий запис, необхідний для проходження опитувань і збереження індивідуальних відповідей у базі даних.

Процес починається з переходу на сторінку реєстрації за адресою http://localhost:3000/auth/signup. Користувач заповнює реєстраційну форму, вказуючи унікальну адресу електронної пошти (наприклад, user@example.com), ім’я (наприклад, Користувач) та пароль (наприклад, password123). Після натискання кнопки «Зареєструватися», форма надсилається на сервер, де відбувається перевірка даних, хешування пароля та збереження нового облікового запису в базі даних.

У разі успішної реєстрації система автоматично перенаправляє користувача на сторінку входу http://localhost:3000/auth/signin, де він може авторизуватись у системі за щойно створеними обліковими даними. Цей механізм забезпечує безпечний та зручний початок роботи з платформою. Форма реєстрації нового користувача відображена на рис. 3.5.

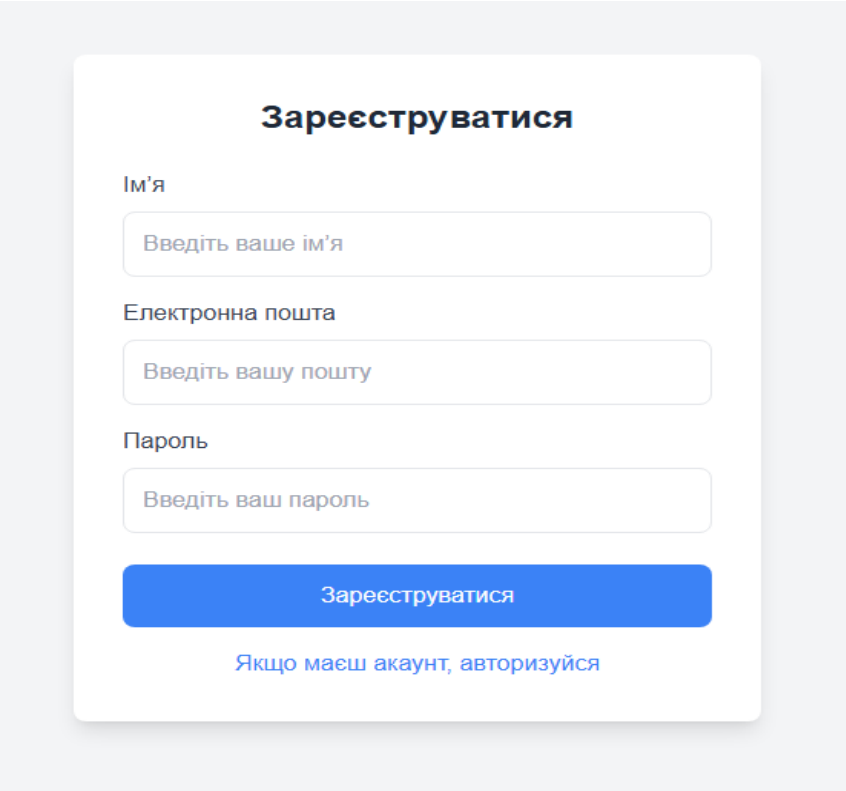


Рисунок 3.5 – Форма реєстрації нового користувача

2. Вхід у систему – забезпечує доступ авторизованих користувачів до особистого кабінету та участі в опитуваннях. Цей процес є необхідною умовою для забезпечення персоналізованого досвіду користування та фіксації результатів опитувань за конкретним користувачем.

Щоб увійти до системи, користувач переходить за посиланням http://localhost:3000/auth/signin, де відкривається форма автентифікації. У відповідні поля вводяться облікові дані: електронна пошта (наприклад, test@example.com) та пароль (наприклад, password123). Після натискання кнопки «Увійти» система перевіряє введену інформацію, звіряє пароль із хешованим значенням у базі даних і, у разі успішного підтвердження, здійснює автентифікацію користувача.

Очікуваним результатом є автоматичне перенаправлення на головну сторінку http://localhost:3000, де авторизований користувач бачить список

доступних опитувань. Успішна авторизація також підтверджується на рівні сесії, що дозволяє ідентифікувати користувача при подальших запитах до API. Форма авторизації користувача показана на рис. 3.6.

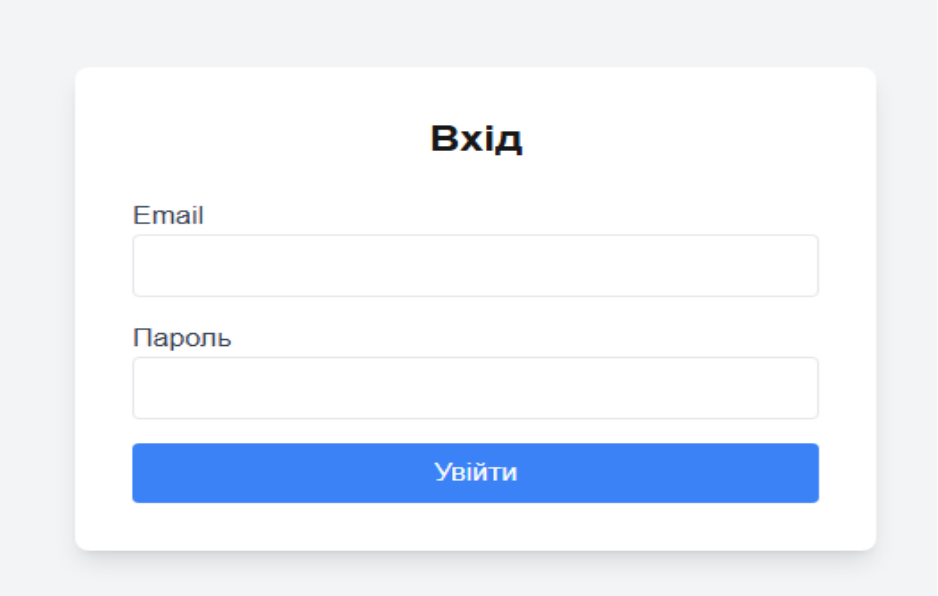


Рисунок 3.6 – Форма авторизації користувача

3. Після успішного входу в систему користувач отримує доступ до головної сторінки, яка розташована за адресою http://localhost:3000. Основне призначення цієї сторінки – представити список усіх опитувань, доступних для проходження. Це дозволяє користувачам швидко зорієнтуватися у тематиці, обрати цікаве опитування та перейти до його проходження.

Інтерфейс сторінки автоматично завантажує перелік опитувань із бази даних через API-запит. Кожен елемент списку містить назву опитування, короткий опис та, за потреби, вказівку на його статус (наприклад, активне чи завершене). Клік по назві опитування здійснює перехід на сторінку з питаннями, яка має маршрут /survey/{id}, де {id} – унікальний ідентифікатор відповідного опитування.

Таким чином, користувач у зручному форматі може ознайомитися з усіма доступними опитуваннями, обрати актуальне для себе та одразу розпочати його проходження. Список тем опитувань відображено на рис. 3.7.

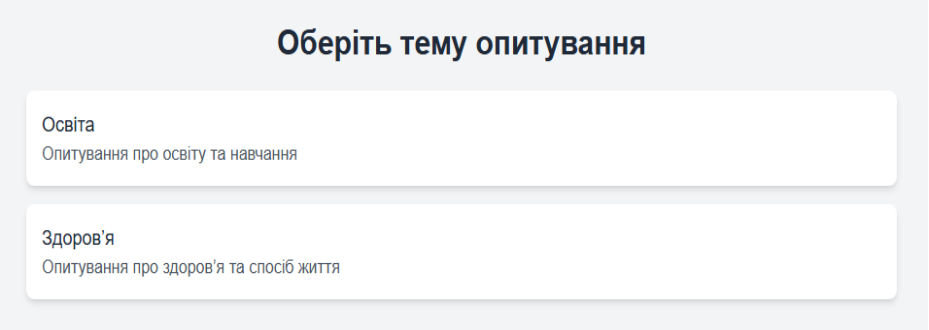


Рисунок 3.7 – Список тем опитувань

При натисканні на тему, наприклад, «Освіта» далі відкриється список опитувань даної теми, як показано на рис.3.8.

Очікуваний результат – перенаправлення на сторінку опитування (http://localhost:3000/survey/2). Далі відображається форма з питаннями і варіантами відповідей.

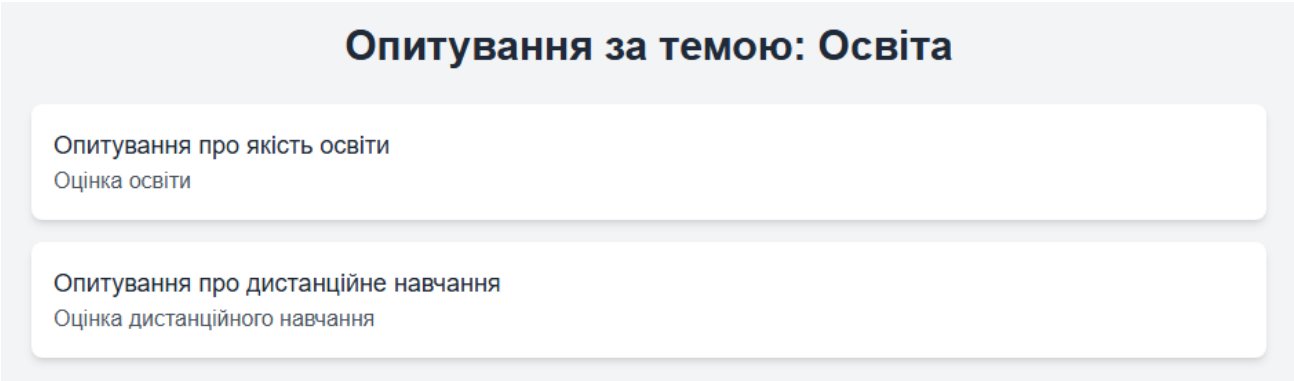


Рисунок 3.8 – Опитування з певної теми

4. Система онлайн-опитувань дозволяє авторизованим користувачам брати участь в активних опитуваннях, відповідаючи на запропоновані запитання. Після входу до системи користувач може перейти на сторінку конкретного опитування,

наприклад, за адресою http://localhost:3000/survey/2, де 2 – це унікальний ідентифікатор опитування в базі даних Форму з питаннями відображено на рис. 3.9.

На сторінці опитування відображаються всі питання, пов’язані з відповідним опитуванням. Кожне питання має відповідний тип: одновибір (radio), множинний вибір (checkbox) або текстова відповідь (input). У випадку з одновибірковими питаннями користувач обирає лише одну відповідь із запропонованих варіантів. Наприклад, для питання «Чи подобається вам тестування?» можна вибрати опцію «Так».

Після вибору відповідей на всі питання користувач натискає кнопку «Зберегти відповіді», яка надсилає відповіді через API-запит на сервер. Система зберігає ці дані у відповідних таблицях бази даних (UserAnswer, UserSurvey), фіксуючи, що конкретний користувач пройшов конкретне опитування.

У разі успішного збереження відповідей користувач автоматично перенаправляється на головну сторінку або на сторінку з результатами, залежно від реалізації системи. Це забезпечує зручний і послідовний досвід взаємодії, дозволяючи ефективно збирати дані для подальшого аналізу.

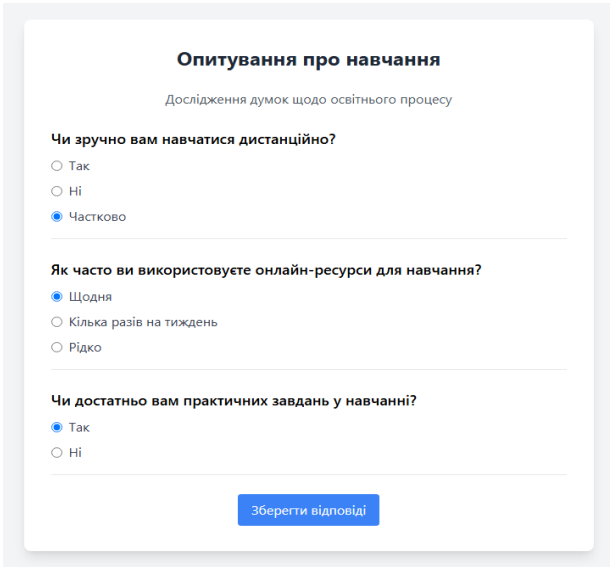


Рисунок 3.9 – Форма з питаннями

5. Перегляд результатів опитування – система онлайн-опитувань надає можливість користувачам переглядати зведену інформацію за результатами опитувань, у яких вони брали участь. Після авторизації в системі користувач переходить на головну сторінку за адресою http://localhost:3000 , де відображається список доступних опитувань. Для кожного з них передбачена можливість перегляду результатів, зокрема для опитування з ідентифікатором id=2.

Натиснувши кнопку «Переглянути результати», користувач перенаправляється на спеціальну сторінку результатів, яка агрегує і візуалізує дані по кожному питанню. Для запитань із варіантами відповідей (одновибір або множинний вибір) система автоматично підраховує кількість голосів за кожен варіант та формує відсоткове співвідношення. Наприклад, для питання «Чи подобається вам тестування?» може відображатися результат: Так: 100%, Ні: 0%.

Представлення результатів реалізується у зручному візуальному форматі, як-от стовпчасті діаграми, кругові діаграми або списки, з використанням бібліотек на зразок Chart.js. Такий підхід дозволяє користувачам швидко оцінити переважаючі думки та прийняти відповідні висновки на основі колективних відповідей.

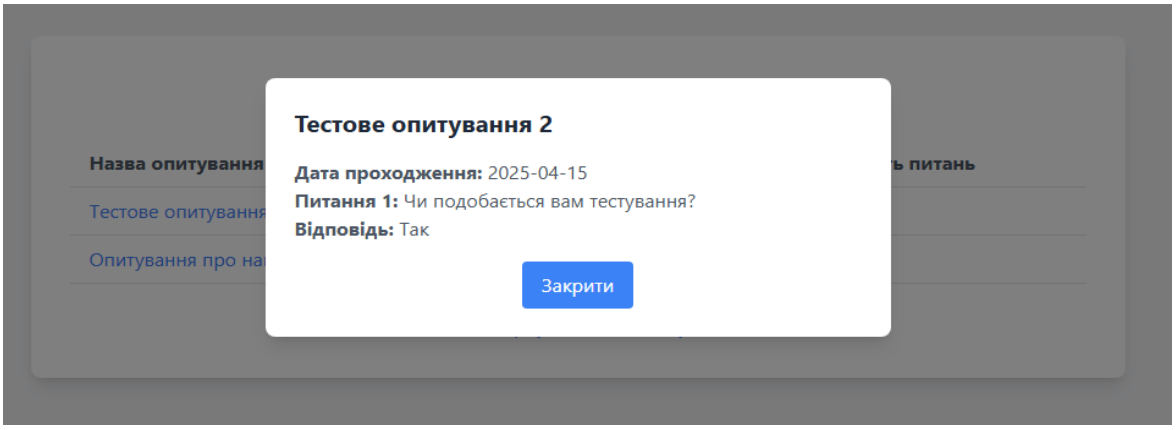


Рисунок 3.10 – Результати опитування

7. Аналітика опитувань полягає в тому, що крім базового перегляду результатів, система онлайн-опитувань також надає функціональність для перегляду аналітики пройдених опитувань. Ця можливість дозволяє не лише бачити підсумкові відповіді, а й аналізувати поведінку користувачів, рівень залученості та загальні тенденції відповідей.

Аналітика включає такі елементи, як:

– Кількість респондентів, які взяли участь в опитуванні.

– Частка відповідей для кожного варіанта у відсотковому співвідношенні.

– Розподіл відповідей у вигляді графіків (стовпчастих, кругових або лінійних).

– Час проходження опитувань, що дозволяє оцінити активність користувачів у динаміці.

– Аналіз відкритих відповідей, наприклад, формування хмари слів або пошук ключових тем.

Дані аналітики генеруються автоматично після завершення опитування та оновлюються в режимі реального часу. Візуалізація реалізується за допомогою інструментів, таких як Chart.js або D3.js, що забезпечує зрозуміле й ефективне представлення інформації. Таким чином, аналітика опитувань дозволяє не лише переглянути, як голосували користувачі, але й отримати цінні інсайти щодо змісту та ефективності самого опитування. Аналітика пройдених опитувань показана на рис. 3.11.

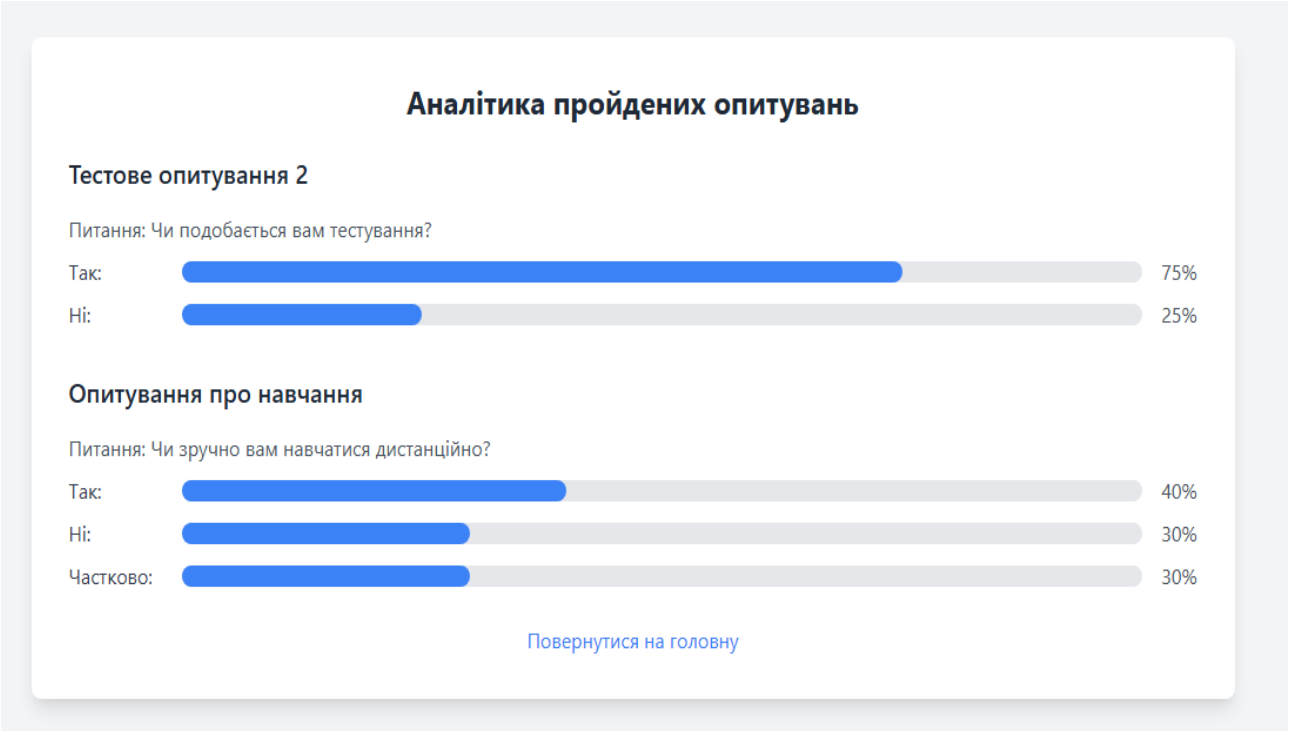


Рисунок 3.11 – Аналітика пройдених опитувань

Інструкція користувача чітко описує кроки для взаємодії з системою, ілюструючи процес скріншотами для зручності. Ці розділи забезпечують повне розуміння можливостей системи та полегшують її використання для користувачів.

### 3.3 Результати впровадження та перспективи розвитку

Система опитувань survey-system, розроблена на основі сучасних веб-технологій – Next.js 15.3.0, PostgreSQL, NextAuth, Prisma та Tailwind CSS – являє собою повнофункціональне рішення для створення, проходження та аналізу онлайн-опитувань. Архітектура системи орієнтована на безпечну взаємодію користувачів з опитуваннями та надання зручного, адаптивного інтерфейсу.

У рамках реалізації впроваджено автентифікацію за допомогою NextAuth, що гарантує захищений доступ до системи. Користувачі можуть реєструватися, входити до облікового запису, переглядати доступні опитування та брати участь

у них, обираючи відповідні варіанти відповідей. Усі дії супроводжуються збереженням результатів у базі даних для подальшого аналізу.

Перспективи подальшого розвитку системи:

1. Усунення поточних недоліків – одним із пріоритетів є виправлення помилки survey: null, яка виникає при завантаженні окремих опитувань. Для її усунення необхідно:

- перевірити наповнення таблиць Survey, Question, Option, Topic;
- налагодити відповідні API-роути (/api/surveys/:surveyId, /api/surveys/public/:surveyId);
- забезпечити стабільне зчитування інформації про користувача через механізми сесій у NextAuth.

2. Розширення адміністративного функціоналу – планується впровадження повноцінної панелі адміністратора з можливостями:

- редагування, архівування та видалення опитувань;
- управління користувачами (зміна ролей, блокування акаунтів);
- моніторингу активності користувачів та історії проходження опитувань.

3. Поглиблення аналітики результатів Для підвищення інформативності аналітичної частини системи доцільно реалізувати:

- інтерактивну візуалізацію результатів (стовпчасті, кругові та лінійні діаграми з використанням Chart.js або D3.js);
- фільтрацію відповідей за різними параметрами (дати, теми, аудиторії);
- експорт результатів у формати PDF та CSV для подальшого використання в дослідженнях або звітності.

4. Підтримка різноманітних типів питань, адже планується розширення функціоналу опитувань за рахунок додавання нових типів питань: текстових відповідей, множинного вибору, рейтингових шкал, відкритих запитань. Це

дозволить зробити систему більш універсальною та придатною для широкого спектру дослідницьких завдань.

5. Оптимізація продуктивності для забезпечення високої швидкодії при зростанні кількості користувачів, оскільки передбачено впровадження індексації бази даних, а також використання механізмів кешування (наприклад, через Redis). Це значно знизить навантаження на сервер і прискорить обробку запитів.

6. Покращення користувацького інтерфейсу за рахунок доповнення анімаціями для попапів, плавних переходів між сторінками та індикації завантаження даних. Також заплановано впровадження темної теми оформлення та адаптивних макетів, що підвищить комфорт використання системи на різних пристроях.

7. Масштабування системи – для забезпечення публічного доступу та підтримки великої кількості одночасних користувачів. Планується розгортання системи на хмарних платформах (Vercel, AWS тощо). Це сприятиме стабільній роботі при збільшенні аудиторії.

8. Посилення безпеки шляхом впровадження додаткових заходів захисту, зокрема двофакторної автентифікації, захисту від CSRF-атак, а також шифрування чутливих даних. Це підвищить безпеку збереження та обробки інформації користувачів.

9. Забезпечення доступності (Accessibility) – інтерфейс системи буде приведено у відповідність зі стандартами WCAG 2.1, додано підтримку клавіатурної навігації, сумісність з програмами читання екрану та реалізовано висококонтрастні режими. Це забезпечить зручність використання системи людьми з різними можливостями.

7. Інтеграція з зовнішніми сервісами – планується реалізація функціоналу імпорту та експорту опитувань через API, інтеграція з платформами для розсилок (наприклад, EmailJS) та соціальними мережами. Це розширить можливості залучення аудиторії та автоматизації роботи з опитуваннями.

Таким чином, система survey-system вже досягла базового рівня функціональності, який забезпечує авторизацію, створення, проходження та аналіз опитувань. Реалізація запропонованих напрямків розвитку дозволить створити сучасну, безпечну та універсальну платформу, що відповідатиме потребам як локальних, так і глобальних користувачів.

# ВИСНОВКИ

Розробка системи онлайн-опитувань survey-system стала комплексним проєктом, що охопив усі етапи створення сучасної платформи для організації, проведення та аналізу опитувань, як визначено у плані роботи. У вступі та першому розділі проведено ґрунтовний аналіз проблематики онлайн-опитувань, розглянуто їх види (анонімні, авторизовані, одноразові, періодичні) та проаналізовано існуючі рішення, такі як Google Forms, SurveyMonkey і Typeform, що дозволило визначити їхні переваги й недоліки. Теоретичні основи автоматичного підрахунку результатів, включаючи статистичні методи й алгоритми агрегації даних, лягли в основу чіткої постановки задачі – створення безпечної, зручної та масштабованої системи з адаптивним інтерфейсом і розширеною аналітикою.

Другий розділ присвячено проєктуванню системи: обрано оптимальний стек технологій (Next.js 15.3.0, PostgreSQL, NextAuth, Prisma, Tailwind CSS), який забезпечує продуктивність, безпеку та сучасний дизайн. Розроблено детальну структуру бази даних для зберігання інформації про опитування, питання, відповіді та користувачів, що забезпечує ефективне управління даними. Бекенд, реалізований через API-роути, підтримує авторизацію, обробку відповідей і аналітику, тоді як фронтенд із адаптивним інтерфейсом, створеним за допомогою Tailwind CSS, забезпечує інтуїтивно зрозумілу взаємодію. Алгоритм автоматичного підрахунку результатів дозволяє миттєво обробляти відповіді та генерувати статистичні звіти, що підвищує цінність системи для адміністраторів.

У третьому розділі описано реалізацію та тестування системи. Тестування підтвердило повну працездатність усіх компонентів: авторизації, створення та проходження опитувань, перегляду результатів і аналітика. Інструкція користувача детально роз’яснює, як користувачі можуть взаємодіяти з системою, включаючи перегляд аналітики та управління темами. Результати впровадження

демонструють, що survey-system успішно виконує всі поставлені завдання, забезпечуючи стабільну роботу та зручність використання.

Система survey-system є повністю готовою та функціональною платформою, що відповідає сучасним вимогам до онлайн-опитувань. Вона забезпечує безпечну авторизацію, інтуїтивний інтерфейс, гнучке створення опитувань і потужну аналітику, що робить її зручною як для користувачів. Перспективи розвитку, такі як додавання нових типів питань (текстові, множинний вибір), інтеграція з зовнішніми сервісами, розширена аналітика з графіками, оптимізація продуктивності через кешування та розгортання на хмарних платформах, відкривають шлях до перетворення системи на конкурентоспроможну платформу міжнародного рівня. Survey-system поєднує простоту використання, функціональність і потенціал для масштабування, що робить її цінним інструментом для збору та аналізу даних у різних сферах, від освіти до бізнесу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Богаченко Владислав Костянтинович. Сучасні онлайн ресурси як засоби контролю навчальної успішності учнів інформатики. Creative Commons BY 4.0 International license. 2024.
2. Онлайн-опитування. URL: https://cpd.com.ua/uk/online-opytuvannya/ (дата звернення: 18.02.2025).
3. Переваги онлайн-опитувань. URL: https://esurvey.com.ua/info/поради-щодо-опитування/переваги-онлайн-опитувань-11 (дата звернення: 10.03.2025).
4. The most common Types of Online Surveys. URL: https://www.resonio.com/market-research/types-of-online-surveys/ (дата звернення: 10.03.2025).
5. The Anatomy of Online Surveys: Definition, Types, and Characteristics. URL: https://survalyzer.com/anatomy-of-online-surveys-definition-types-characteristics/ (дата звернення: 15.03.2025).
6. Швидко отримуйте дані за допомогою онлайн-форм. URL: https://workspace.google.com/products/forms/ (дата звернення: 15.03.2025).
7. Google Forms: what is it and how to create it?. URL: https://wedex.com.ua/en/blog/google-forms-what-is-it-and-how-to-create-it/ (дата звернення: 15.03.2025).
8. Advantages and disadvantages of Google forms. URL: https://datascope.io/en/blog/advantages-and-disadvantages-of-google-forms/ (дата звернення: 15.03.2025).
9. SurveyMonkey. URL: https://www.surveymonkey.com/about/ (дата звернення: 12.04.2025).
10. What is SurveyMonkey?. URL: https://www.genroe.com/blog/what-is-surveymonkey/7979 (дата звернення: 13.04.2025).

11. Reviews of SurveyMonkey. URL: https://www.capterra.com/p/253516/SurveyMonkey/reviews/ (дата звернення: 13.04.2025).
12. Typeform. URL: https://www.typeform.com/about-us (дата звернення: 17.04.2025).
13. Typeform Review – Complete Guide. URL: https://www.windwardstudios.com/blog/typeform-review (дата звернення: 17.04.2025).
14. Typeform Reviews. URL: https://www.joinsecret.com/typeform/reviews (дата звернення: 04.05.2025).
15. de leeuw, Edith. Counting and Measuring Online: The Quality of Internet Surveys. Bull Sociol Method. 114. 10.1177/0759106312437290. 2012.
16. Valerie M. Sue, Lois A. Ritter. Processing and Analyzing the Survey Data. URL: https://methods.sagepub.com/book/mono/conducting-online-surveys/chpt/processing-analyzing-survey-data (дата звернення: 10.05.2025).
17. Cleland, Jamie & Dixon, Kevin & Kilvington, Daniel. Online Surveys. 10.4324/9780367809300-4. 2019.
18. Analyzing Survey Results: The Ultimate Guide. URL: https://survicate.com/blog/survey-analysis/ (дата звернення: 10.05.2025).
19. Online Survey Data – Analysis and Reporting A comprehensive guide. URL: https://www.resonio.com/market-research/online-survey-data-analysis-and-reporting/ (дата звернення: 17.05.2025).
20. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 19.05.2025).
21. Martin Krause. The Complete Developer: Master the Full Stack with TypeScript, React, Next.js, MongoDB, and Docker. c. 344. 2024.

22. Morgan Devline. Next.js: Building Modern React Applications with Server-Side Rendering (SSR), Static Site Generation (SSG), and Hands-On Projects. с. 342. 2025.
23. Michele Riva. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production. Packt Publishing. с. 366. 2022.
24. Alex Banks, Eve Porcello. Learning React: Modern Patterns for Developing React Apps 2nd Edition. O'Reilly Media. с. 310. 2020.
25. Robin Wieruch. The Road to React: Your journey to master plain yet pragmatic React.js. с. 286. 2018.
26. Stoyan Stefanov. React: Up & Running: Building Web Applications 1st Edition. O'Reilly Media. с. 222. 2016.
27. React.js benefits: Why It’s perfect for your business applications. URL: https://medium.com/@reactmasters.in/advantages-and-disadvantages-of-react-js-e6c80b25763b (дата звернення: 19.05.2025).
28. Roberto Heckers. Effective Angular: Develop applications of any size by effectively using Angular with Nx, RxJS, NgRx, and Cypress. Packt Publishing. с. 400. 2024.
29. Aristeidis Bampakos, Pablo Deeleman. Learning Angular: A no-nonsense beginner's guide to building web applications with Angular 10 and TypeScript, 3rd Edition. Packt Publishing. с. 430. 2020.
30. Adam Freeman. Pro Angular Build Powerful and Dynamic Web Apps. APress. с. 880. 2022.
31. Pros and Cons of Angular Framework You Need to Know. URL: https://www.robinwaite.com/blog/pros-and-cons-of-angular-framework-you-need-to-know (дата звернення: 20.05.2025).
32. Main Features of Next.js. URL: https://medium.com/@navtechsolution16/main-features-of-next-js-57302ab7a49d (дата звернення: 23.05.2025).

33. Top React Features for Web Developers. URL: https://dev.to/brilworks/top-react-features-for-web-developers-2afi (дата звернення: 25.05.2025).
34. Top 20 Key Features Of Angular. URL: https://medium.com/@aqeelabbas3972/modular-development-angular-uses-a-modular-approach-to-development-which-helps-developers-create-7e9686a51eb4 (дата звернення: 25.05.2025).

# ДОДАТОК А – ТАБЛИЦІ

## ДОДАТОК А.1 – Класифікація онлайн-опитувань

| № | Назва | Тип |
|---|---|---|
| 1 | 2 | 3 |
| 1 | За формою взаємодії з респондентами | Анонімні опитування – відповіді респондентів не прив'язуються до конкретних осіб. Це дозволяє забезпечити конфіденційність та отримати більш відверті відповіді. Такі опитування часто використовуються у внутрішньокорпоративних дослідженнях або соціологічних опитуваннях. |
| | | Ідентифіковані опитування – відповіді зберігаються разом із даними про респондента (ім'я, електронна пошта, унікальний ідентифікатор). Використовуються для тестування знань, персоналізованих опитувань клієнтів або внутрішніх оцінювань у компаніях. |
| 2 | За типом запитань | Закриті опитування – містять чітко визначені варіанти відповідей, серед яких респондент повинен вибрати правильний або найбільш відповідний варіант. Наприклад, питання з варіантами "Так/Ні" або з вибором однієї чи кількох відповідей із запропонованого списку. |
| | | Відкриті опитування – передбачають самостійне формулювання відповіді респондентом. Такі опитування дозволяють отримати більш розгорнуті відповіді, проте їхній аналіз є складнішим і часто потребує додаткової обробки. |
| | | Комбіновані опитування – поєднують закриті та відкриті запитання, що дозволяє отримати як структуровані відповіді, так і додаткові коментарі чи пояснення [4]. |
| 3 | За призначенням | Соціологічні опитування – використовуються для дослідження громадської думки, аналізу соціальних тенденцій, політичних уподобань населення тощо. Такі опитування проводяться соціологічними організаціями, урядовими установами або громадськими організаціями. |
| | | Маркетингові опитування – призначені для дослідження поведінки споживачів, оцінки попиту на товари та послуги, аналізу задоволеності клієнтів. Наприклад, компанії можуть проводити опитування перед запуском нового продукту, щоб з'ясувати потреби потенційних покупців. |
| | | Освітні опитування та тести – використовуються у навчальних закладах та корпоративному навчанні для перевірки знань, оцінювання успішності студентів або збору зворотного зв'язку від слухачів курсів. |
| | | Внутрішньокорпоративні опитування – проводяться в компаніях для оцінки рівня задоволеності співробітників, аналізу ефективності управлінських рішень або збору ідей для розвитку бізнесу. |

| 1 | 2 | 3 |
|---|---|---|
| 4 | За способом обробки результатів | Опитування з ручним підрахунком результатів – дані збираються у вигляді таблиць або текстових документів, після чого їх аналіз проводиться вручну або за допомогою стандартних засобів статистичної обробки. Такий підхід є менш ефективним для великих обсягів даних. |
| | | Опитування з автоматичним підрахунком результатів – система самостійно аналізує відповіді, генерує статистику, графіки та аналітичні звіти. Це значно прискорює процес обробки даних та мінімізує ризик людських помилок. |
| 5 | За рівнем інтерактивності | Статичні опитування – стандартні анкети без додаткових елементів взаємодії. Вони складаються з набору запитань, на які респондент послідовно відповідає. |
| | | Динамічні опитування – містять адаптивні механізми, коли наступні запитання змінюються залежно від попередніх відповідей респондента. Наприклад, якщо респондент відповів, що не користується певним продуктом, система може пропустити запитання, пов’язані з його оцінкою [5]. |

**ДОДАТОК А.2 – Порівняльна таблиця платформ для онлайн-опитувань**

| № | Критерій | Google Forms [6] | SurveyMonkey [9] | Typeform [12] |
|---|---|---|---|---|
| 1 | Вартість | Безкоштовно (з базовими функціями) | Обмежена безкоштовна версія, платні тарифи від $25/міс | Безкоштовно до 10 відповідей/міс, платні тарифи від $29/міс |
| 2 | Простота використання | Дуже проста | Середня (багато налаштувань) | Інтуїтивна, сучасний UX |
| 3 | Дизайн та кастомізація | Мінімальний | Гнучкий (шаблони та теми) | Сучасний, кастомізований |
| 4 | Типи запитань | Основні (текст, вибір, шкала) | Розширені (логічні переходи, фільтрація) | Інтерактивні, мультимедійні |
| 5 | Автоматичний підрахунок відповідей | Так | Так | Так |
| 6 | Логіка запитань (умовні переходи) | Обмежена | Розширена | Розширена |
| 7 | Аналітика | Базові графіки, інтеграція з Google Sheets | Потужний вбудований аналіз | Мінімальна аналітика |
| 8 | Інтеграції | Google Drive, Sheets | CRM, маркетингові платформи | Zapier, Slack, Google Sheets |
| 9 | Обмеження безкоштовної версії | Без ліміту відповідей | 10 запитань, 40 відповідей | 10 відповідей на місяць |
| 10 | Найкраще підходить для | Простих опитувань | Бізнесу та досліджень | Інтерактивних опитувань |

## ДОДАТОК А.3 – Алгоритми автоматизації підрахунку результатів

| № | Алгоритм | Принцип роботи | Застосування | Переваги | Недоліки |
|---|---|---|---|---|---|
| 1 | Прямий підрахунок (Direct Counting Algorithm) [17] | Підсумовування відповідей у реальному часі | Просте опитування з вибором одного варіанта відповіді | Швидкість, простота реалізації | Обмежена функціональність |
| 2 | Логічні правила (Rule-Based Scoring) [17] | Використання умовних операторів (if-else) для нарахування балів | Тестування знань, квізи | Гнучке налаштування оцінювання | Потрібна ручна розробка правил |
| 3 | Статистичний аналіз (Statistical Analysis Methods) [17] | Обчислення середнього значення, моди, медіани, стандартного відхилення | Маркетингові дослідження, соціологічні опитування | Дає загальну картину розподілу відповідей | Не дозволяє персоналізований аналіз |
| 4 | Машинне навчання (ML-based Analysis) [17] | Аналіз великих обсягів відповідей та виявлення прихованих закономірностей | Персоналізовані опитування, поведінковий аналіз | Висока точність прогнозів | Високі обчислювальні ресурси |

# ДОДАТОК Б – ЛІСТИНГИ

## ДОДАТОК Б.1 – Встановлення Prisma

```
npm install prisma --save-dev
npx prisma init
```

У файлі prisma/schema.prisma:

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

Визначення моделей:

```
model User {
  id           Int          @id @default(autoincrement())
  email        String       @unique
  password     String?
  name         String?
  createdAt    DateTime     @default(now())
  updatedAt    DateTime     @updatedAt
  accounts     Account[]
  sessions     Session[]
  userSurveys  UserSurvey[]
  userAnswers  UserAnswer[]
}

model Account {
  id             String   @id @default(cuid())
  userId          Int
  type            String
```

```
  provider          String
  providerAccountId String
  refresh_token     String?  @db.Text
  access_token      String?  @db.Text
  expires_at        Int?
  token_type        String?
  scope             String?
  id_token          String?  @db.Text
  session_state     String?
  user              User     @relation(fields: [userId], references: [id], onDelete: Cascade)
  @@unique([provider, providerAccountId])
}
model Session {
  id           String   @id @default(cuid())
  sessionToken String   @unique
  userId       Int
  expires      DateTime
  user         User     @relation(fields: [userId], references: [id], onDelete: Cascade)
}
model VerificationToken {
  identifier String
  token      String   @unique
  expires    DateTime
  @@unique([identifier, token])
}
model UserSurvey {
  id          Int      @id @default(autoincrement())
  userId      Int
  surveyId    Int
  completedAt DateTime @default(now())
  user        User     @relation(fields: [userId], references: [id], onDelete: Cascade)
  survey      Survey   @relation(fields: [surveyId], references: [id], onDelete: Cascade)
  @@unique([userId, surveyId])
}
```

```
model Survey {
  id          Int          @id @default(autoincrement())
  title       String
  description String?
  topicId     Int
  topic       Topic        @relation(fields: [topicId], references: [id], onDelete: Cascade)
  createdAt   DateTime     @default(now())
  userSurveys UserSurvey[]
  questions   Question[]
}
model Topic {
  id          Int          @id @default(autoincrement())
  name        String
  description String?
  surveys     Survey[]
}
model Question {
  id          Int          @id @default(autoincrement())
  surveyId    Int
  text        String
  survey      Survey       @relation(fields: [surveyId], references: [id], onDelete: Cascade)
  options     Option[]
  userAnswers UserAnswer[]
}
model Option {
  id          Int          @id @default(autoincrement())
  questionId  Int
  text        String
  question    Question     @relation(fields: [questionId], references: [id], onDelete: Cascade)
  userAnswers UserAnswer[]
}
```

```
model UserAnswer {
  id         Int      @id @default(autoincrement())
  userId     Int
  questionId Int
  optionId   Int
  answeredAt DateTime @default(now())
  user       User     @relation(fields: [userId], references: [id], onDelete: Cascade)
  question   Question @relation(fields: [questionId], references: [id], onDelete: Cascade)
  option     Option   @relation(fields: [optionId], references: [id], onDelete: Cascade)
  @@unique([userId, questionId])
}
```

Налаштування .env:

```
DATABASE_URL="postgresql://postgres:@localhost:5432/survey_system?schema=public"
```

Для створення таблиць у базі даних необхідно виконати, оскільки, це згенерує SQL-міграцію та застосує її до PostgreSQL :

```
npx prisma migrate dev --name init
```

## ДОДАТОК Б.2 – Використання ендпоінта

```
import prisma from '../../lib/prisma';

export default async function handler(req, res) {
  if (req.method === 'POST') {
    const { title, description, status } = req.body;
    const survey = await prisma.survey.create({
      data: { title, description, status },
    });
    res.status(201).json(survey);
  } else {
    res.setHeader('Allow', ['POST']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

Цей ендпоінт повертає список усіх доступних опитувань.

```
import prisma from '../../lib/prisma';
export default async function handler(req, res) {
  if (req.method === 'GET') {
    const surveys = await prisma.survey.findMany();
    res.status(200).json(surveys);
  } else {
    res.setHeader('Allow', ['GET']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

Ендпоінт додає нове питання до конкретного опитування, включаючи текст, тип і можливі варіанти відповідей.

```
import prisma from '../../lib/prisma';
export default async function handler(req, res) {
  if (req.method === 'POST') {
    const { surveyId, text, type, options } = req.body;
    const question = await prisma.question.create({
```

```
      data: {
        survey: { connect: { id: surveyId } },
        text,
        type,
        options: options ? JSON.stringify(options) : null,
      },
    });
    res.status(201).json(question);
  } else {
    res.setHeader('Allow', ['POST']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

Цей ендпоінт повертає всі питання, пов’язані з конкретним опитуванням.

```
import prisma from '../../../lib/prisma';
export default async function handler(req, res) {
  const { id } = req.query;
  if (req.method === 'GET') {
    const questions = await prisma.question.findMany({
      where: { surveyId: parseInt(id) },
    });
    res.status(200).json(questions);
  } else {
    res.setHeader('Allow', ['GET']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

Ендпоінт дозволяє користувачам надсилати відповіді на питання, зберігаючи їх у базі даних.

```
import prisma from '../../lib/prisma';
export default async function handler(req, res) {
  if (req.method === 'POST') {
    const { questionId, userId, selectedOption, textAnswer } = req.body;
```

```
    const answer = await prisma.answer.create({
      data: {
        question: { connect: { id: questionId } },
        user: { connect: { id: userId } },
        selectedOption,
        textAnswer,
      },
    });
    res.status(201).json(answer);
  } else {
    res.setHeader('Allow', ['POST']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

## ДОДАТОК Б.3 – Окремий API-ендпоінт у системі опитувань

```
import prisma from '../../../lib/prisma';
export default async function handler(req, res) {
  const { id } = req.query;
  if (req.method === 'GET') {
    const questions = await prisma.question.findMany({
      where: { surveyId: parseInt(id) },
      include: { answers: true },
    });
    const results = questions.map(question => {
      if (question.type === 'single_choice' || question.type === 'multiple_choice') {
        const optionsCount = {};
        question.answers.forEach(answer => {
          optionsCount[answer.selectedOption] = (optionsCount[answer.selectedOption] || 0)
+ 1;
        });
        return { questionId: question.id, type: question.type, results: optionsCount };
      } else {
        return { questionId: question.id, type: question.type, answers:
question.answers.map(a => a.textAnswer) };
      }
    });
    res.status(200).json(results);
  } else {
    res.setHeader('Allow', ['GET']);
    res.status(405).end(`Method ${req.method} Not Allowed`);
  }
}
```

**ДОДАТОК Б.4 – Лістинг файлу index.tsx**

```
"use client";
import { useSession } from "next-auth/react";
import { useRouter } from "next/router";
import { useEffect, useState } from "react";
export default function Home() {
  const { data: session, status } = useSession();
  const router = useRouter();
  const [surveys, setSurveys] = useState([]);
  const [loading, setLoading] = useState(true);
  const [selectedSurvey, setSelectedSurvey] = useState(null);
  const [answers, setAnswers] = useState([]);
  const [popupLoading, setPopupLoading] = useState(false);

  useEffect(() => {
    if (status === "unauthenticated") {
      router.push("/auth/signin");
    } else if (status === "authenticated") {
      const fetchSurveys = async () => {
        try {
          const response = await fetch("/api/user-surveys");
          if (response.ok) {
            const data = await response.json();
            setSurveys(data);
          }
        } catch (error) {
          console.error("Помилка:", error);
        } finally {
          setLoading(false);
```

```
        }
      };
      fetchSurveys();
    }
  }, [status, router]);

  const handleViewAnswers = async (surveyId) => {
    setPopupLoading(true);
    try {
      const response = await fetch(`/api/answers?surveyId=${surveyId}&userId=${session.user.id}`);
      if (response.ok) {
        const data = await response.json();
        setAnswers(data);
        setSelectedSurvey(surveys.find((s) => s.surveyId === surveyId));
      }
    } catch (error) {
      console.error("Помилка:", error);
    } finally {
      setPopupLoading(false);
    }
  };

  const closePopup = () => {
    setSelectedSurvey(null);
    setAnswers([]);
  };

  const handleTakeSurvey = () => {
```

```
  router.push("/survey/new");
 };

 if (status === "loading" || loading) {
  return <div className="min-h-screen flex items-center justify-center">Завантаження...</div>;
 }

 return (
  <div className="min-h-screen bg-gray-100 p-6">
   <div className="max-w-4xl mx-auto">
    <h1 className="text-3xl font-bold mb-6 text-center text-gray-800">
     Вітаємо, {session?.user?.name || "Користувачу"}!
    </h1>
    <div className="mb-6 text-center">
     <button
      onClick={handleTakeSurvey}
      className="bg-blue-500 text-white px-6 py-3 rounded-lg hover:bg-blue-600 transition-colors"
     >
      Пройти опитування
     </button>
    </div>
    <h2 className="text-2xl font-semibold mb-4 text-gray-700">Пройдені опитування</h2>
    {surveys.length === 0 ? (
     <p className="text-gray-600">Ви ще не пройшли жодного опитування.</p>
    ) : (
```

```
            <ul className="space-y-4">
              {surveys.map((survey) => (
                <li
                  key={survey.id}
                  className="bg-white p-4 rounded-lg shadow-md flex justify-between items-center"
                >
                  <div>
                    <button
                      onClick={() => handleViewAnswers(survey.surveyId)}
                      className="text-lg font-medium text-blue-500 hover:underline"
                    >
                      {survey.survey.title}
                    </button>
                    <p className="text-gray-600">{survey.survey.description}</p>
                    <p className="text-sm text-gray-500">
                      Пройдено: {new Date(survey.completedAt).toLocaleDateString("uk-UA")}
                    </p>
                  </div>
                </li>
              ))}
            </ul>
          )}
          {selectedSurvey && (
            <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center">
              <div className="bg-white p-6 rounded-lg max-w-2xl w-full max-h-[80vh] overflow-y-auto">
```

```
                <h2 className="text-2xl font-bold mb-4 text-gray-
800">{selectedSurvey.survey.title}</h2>
                {popupLoading ? (
                  <p className="text-gray-600">Завантаження відповідей...</p>
                ) : answers.length === 0 ? (
                  <p className="text-gray-600">Відповідей не знайдено.</p>
                ) : (
                  <ul className="space-y-4">
                    {answers.map((answer) => (
                      <li key={answer.id}>
                        <p                  className="font-medium                  text-gray-
800">{answer.question.text}</p>
                        <p                              className="text-gray-600">Відповідь:
{answer.option.text}</p>
                      </li>
                    ))}
                  </ul>
                )}
                <button
                  onClick={closePopup}
                  className="mt-4 w-full bg-blue-500 text-white p-3 rounded-lg
hover:bg-blue-600 transition-colors"
                >
                  Закрити
                </button>
              </div>
            </div>
          )}
        </div>
      </div>
    );
  }
```

**ДОДАТОК Б.5 – Лістинг файлу [...nextauth].ts**

```
import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import { PrismaAdapter } from "@next-auth/prisma-adapter";
import { PrismaClient } from "@prisma/client";
import bcrypt from "bcrypt";

const prisma = new PrismaClient();

export default NextAuth({
  adapter: PrismaAdapter(prisma),
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: { label: "Email", type: "email" },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        console.log("Authorize - Credentials:", credentials); // Дебаг
        if (!credentials?.email || !credentials?.password) {
          console.log("Authorize - Missing email or password");
          throw new Error("Please enter email and password");
        }

        const user = await prisma.user.findUnique({
          where: { email: credentials.email },
        });
```

```
            console.log("Authorize - User:", user); // Дебаг

            if (!user || !user.password) {
              console.log("Authorize - No user found");
              throw new Error("No user found");
            }

            const isValid = await bcrypt.compare(credentials.password,
user.password);
            console.log("Authorize - Password valid:", isValid); // Дебаг

            if (!isValid) {
              console.log("Authorize - Invalid password");
              throw new Error("Invalid password");
            }

            console.log("Authorize - Success:", { id: user.id.toString(), name:
user.name, email: user.email });
            return { id: user.id.toString(), name: user.name, email: user.email };
          },
        }),
      ],
      pages: {
        signIn: "/auth/signin",
        signUp: "/auth/signup",
        error: "/auth/error",
      },
      session: {
```

```
  strategy: "jwt",
```

*Продовження ДОДАТКА Б.5*

```
 },
 callbacks: {
  async jwt({ token, user }) {
   if (user) {
    token.id = user.id;
   }
   return token;
  },
  async session({ session, token }) {
   console.log("Session callback - Token:", token); // Дебаг
   if (session.user) {
    session.user.id = token.id as string;
   }
   return session;
  },
 },
 secret: process.env.NEXTAUTH_SECRET,
});
```